Contents

Asymptotic notation

Problem 1 (6 %) check

In the following, $\log n$ denotes the base-2 logarithm of n.

	Yes	No	
$(\log n)^3 = O(n^2)$			1.1
$4\log(n^2) = O(n^{3/2})$			1.2
$n^2 + 3\log n = O(\log n)$			1.3
$n^2 = O(n)$			1.4
$2n \cdot \log n = O(2^n)$			1.5
$3^3 + \sqrt{n} = O(n!)$			1.6
$3\sqrt{n} + 7n^{1/3} = O(n^{0.1})$			1.7
$4(\log n)^4 = O(2^n)$			1.8
$n \cdot \log n = O(\sqrt{n})$			1.9
$\log n = \Omega(\sqrt{n})$			1.10
$n \cdot \log n = \Omega(n^{0.1})$			1.11
$n^{0.01}/5 = \Omega(\log n)$			1.12

Problem 2 (6 %) check

In the following, $\log n$ denotes the base-2 logarithm of n.

	Yes	No	
$6\sqrt{n} \cdot \log n = O(n^{2/3})$			2.1
$n^{0.1} = O(n)$			2.2
$7(\log n)^2 = O(n^{0.001})$			2.3
$n \cdot \log n = O(\sqrt{n} \cdot \log n)$			2.4
$2^n = O(n^3)$			2.5
$n^2 \log n = O(\log n)$			2.6
$n^3 = O(\sqrt{n})$			2.7
$n = O(\log(n!))$			2.8
$7n \cdot \log n = O(n^{0.001})$			2.9
$n \cdot \log n = \Omega(8^{\log n})$			2.10
$\sqrt{n} = \Omega((\log n)^2)$			2.11
$n \cdot \log n = \Omega(n^{2/3})$			2.12

Problem 3 (6 %) check

In the following, $\log n$ denotes the base-2 logarithm of n.

	Yes	No	
$\sqrt{n} = O(n^{2/3})$			3.1
$\log(n^2) = O(n \cdot \log n)$			3.2
$n = O(n^{0.01})$			3.3
$4n = O(n)$			3.4
$8^{\log n} = O(\log n)$			3.5
$2^n = O(\log(n^2))$			3.6
$\log(n!) = O(n)$			3.7
$n^2 = O(n^{0.001})$			3.8
$(\log n)^2 = O(n^{0.01})$			3.9
$n^{0.001} = \Omega(\log(n^2))$			3.10
$n^2 \log n = \Omega(\sqrt{n})$			3.11
$3^n = \Omega(\log n)$			3.12

Problem 4 (6 %) check

In the following, $\log n$ denotes the base-2 logarithm of n.

	Yes	No	
$n + \sqrt{n} = O(n \cdot \log n)$			4.1
$2^{3 \log n} = O(n^n)$			4.2
$7n^3 = O(3^3)$			4.3
$n^2 \log n = O(\sqrt{n})$			4.4
$n \cdot \log n = O(n^3)$			4.5
$\log(n!) = O(n^3)$			4.6
$2^n = O(2^{3 \log n})$			4.7
$5n^{2/3} \cdot n^{1/3} = O(n^n)$			4.8
$\sqrt{n} = O(2^{\log n})$			4.9
$\log n + \log n = \Theta(\log n)$			4.10
$4\sqrt{n} \cdot \log n = \Omega(\sum_{i=1}^{n} i)$			4.11
$(\log n)^3 = \Theta((\log n)^6)$			4.12

Problem 5 (6 %) check

In the following, $\log n$ denotes the base-2 logarithm of n.

	Yes	No	
$\log(n!) = O(n!)$			5.1
$2^n = O(n^{0.1})$			5.2
$n^{3/2} = O(n^2 \log n)$			5.3
$6n^2 = O(2^{\log n})$			5.4
$\log n = O(\sqrt{n} \cdot \log n)$			5.5
$5 \log n + n^2 = O(\log(n^2))$			5.6
$2^{3 \log n} = O(n)$			5.7
$1 = O(n^{0.1})$			5.8
$n \cdot \log n = O(n)$			5.9
$1 = \Omega(n^{1/3})$			5.10
$6 \log n = \Theta(4^{\log n})$			5.11
$n^{0.001} = \Omega(\sqrt{n} \cdot \log n)$			5.12

Problem 6 (6 %) check

In the following, $\log n$ denotes the base-2 logarithm of n.

	Yes	No	
$\log n = O(2^{\log n})$			6.1
$8^{\log n} = O(n^{2/3})$			6.2
$\sqrt{n} = O(n^{2/3})$			6.3
$2^{\log n} = O(2^{2 \log n})$			6.4
$\log(n^2) = O(\log n)$			6.5
$\sqrt{n} \cdot \log n = O(\sqrt{n})$			6.6
$\log(n!)/7 = O(n^2 \log n)$			6.7
$n! + \sqrt{n} \cdot (\log n)/3 = O(\sqrt{n})$			6.8
$\sqrt{n} = O((\log n)^3)$			6.9
$5\sqrt{n} + 2 \cdot 8^{\log n} = \Omega(n!)$			6.10
$8^{\log n} = \Theta(2^{3 \log n})$			6.11
$n \cdot \log n = \Omega(2^{3 \log n})$			6.12

Problem 7 (6 %) check

In the following, $\log n$ denotes the base-2 logarithm of n.

	Yes	No	
$1 + \log(n^2)/2 = O(\log n)$			7.1
$2n = O(n \cdot \log n)$			7.2
$1 + 3^3 = O(2^{3 \log n})$			7.3
$7n \cdot \log n = O(\log n)$			7.4
$n! = O((\log n)^3)$			7.5
$n^2 \log n = O(\sqrt{n})$			7.6
$n \cdot \log n = O(\log(n^2))$			7.7
$n^{2/3} = O(2^{3 \log n})$			7.8
$3n \cdot \log n = O(1)$			7.9
$n = \Theta(n^{0.1})$			7.10
$\log n = \Omega(n \cdot \log n)$			7.11
$4 \log n = \Theta(\log n)$			7.12

Problem 8 (6 %) check

In the following, $\log n$ denotes the base-2 logarithm of n.

	Yes	No	
$n \cdot \log n = O(\sqrt{n})$			8.1
$n^{0.01} = O(\sqrt{n} \cdot \log n)$			8.2
$4 \cdot 3^n = O(\log n)$			8.3
$n^{2/3} = O(n^{2/3} \cdot n^{1/3})$			8.4
$5^5 + n^{0.001} = O(n \cdot \log n)$			8.5
$\log(n^2) = O(\log n)$			8.6
$5n^{1/3} = O(\log n)$			8.7
$2^{3 \log n} = O(n^3)$			8.8
$6\sqrt{n} \cdot \log n = O(\sqrt{n})$			8.9
$n/6 = \Theta(n)$			8.10
$2^{2 \log n} = \Theta(n\sqrt{n})$			8.11
$n^3 = \Theta(n \cdot \log n)$			8.12

Problem 9 (6 %) check

In the following, $\log n$ denotes the base-2 logarithm of n.

	Yes	No	
$\sqrt{n}/2 = O(\log(n^2))$			9.1
$2^n = O(n \cdot \log n)$			9.2
$6\log(n!) + n \cdot \log n = O(2^{\log n})$			9.3
$\log n = O(\sqrt{n} \cdot \log n)$			9.4
$2 = O((\log n)^2)$			9.5
$(\log n)^3 = O(n \cdot \log n)$			9.6
$n \cdot \log n = O(2^n)$			9.7
$n^3 + n \cdot \log n = O(n \cdot \log n)$			9.8
$8^{\log n} = O(2^n)$			9.9
$n^{0.1}/6 = \Theta(n^n)$			9.10
$4^{\log n} = \Theta(\sum_{i=1}^{n} i)$			9.11
$1 = \Theta(5^5)$			9.12

Problem 10 (6 %) check

In the following, $\log n$ denotes the base-2 logarithm of n.

	Yes	No	
$n \cdot \log n = O(2^{2\log n})$			10.1
$2^{3\log n} = O(n!)$			10.2
$n = O(n \cdot \log n)$			10.3
$n = O(n\sqrt{n})$			10.4
$3n = O(n)$			10.5
$\sqrt{n} + \sqrt{n} = O(n^{0.1})$			10.6
$\log(n!) + 6n^{3/2} = O(\log n)$			10.7
$n^{0.1} = O(\sqrt{n} \cdot \log n)$			10.8
$n \cdot (\log n)/3 = O(2^n)$			10.9
$n \cdot \log n = \Theta(\log(n!))$			10.10
$1 + 4 = \Theta(4^4)$			10.11
$\sqrt{n} \cdot \log n = \Omega(\sqrt{n})$			10.12

Analysis of loops

Problem 11 (6 %) check

```
Algorithm loop1(n)
  s = 1
  while s ≤ n
    s = s + 1
```

```
Algorithm loop2(n)
  s = 0
  for i = 1 to n
    for j = 1 to n
      for k = 1 to n
        s = s + 1
```

```
Algorithm loop3(n)
  i = 0
  j = n
  while i < j
    i = i + 2
    j = j + 1
```

```
Algorithm loop4(n)
  for i = 1 to n
    j = i
    while j ≤ n
      j = 2 * j
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta((\log n)^2)$	$\Theta(n^3)$	$\Theta(n \log n)$	$\Theta(n)$	$\Theta(\log \log n)$	$\Theta(\log n)$	$\Theta(n^2)$	$\Theta(n^2 \cdot \log n)$	
loop1									11.1
loop2									11.2
loop3									11.3
loop4									11.4

Problem 12 (6 %) check

```
Algorithm loop1(n)
  i = 1
  while i ≤ n
    i = 2 * i
```

```
Algorithm loop2(n)
  s = 1
  for i = 1 to n
    for j = i to n
      s = s + 1
```

```
Algorithm loop3(n)
  i = 1
  j = n
  while i ≤ j
    i = i * 2
    j = ⌊j/2⌋
```

```
Algorithm loop4(n)
  for i = 1 to n
    j = 0
    while j ≤ n
      j = j + i
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta(\log n)$	$\Theta((\log n)^2)$	$\Theta(\log \log n)$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^3)$	$\Theta(n^2)$	$\Theta(n^2 \cdot \log n)$	
loop1									12.1
loop2									12.2
loop3									12.3
loop4									12.4

Problem 13 (6 %) check

```
Algorithm loop1(n)
  s = 0
  for i = 1 to n
      for j = 1 to i * i
          s = s + 1

Algorithm loop2(n)
  i = 1
  while i ≤ n
      i = 2 * i

Algorithm loop3(n)
  i = 1
  while i ≤ n * n
      i = 2 * i

Algorithm loop4(n)
  i = 1
  while i ≤ n
      j = 1
      while j ≤ i
          j = 2 * j
      i = 2 * i
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta(n^2)$	$\Theta((\log n)^2)$	$\Theta(\log n)$	$\Theta(n^3)$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(\sqrt{n})$	$\Theta(n^2 \cdot \log n)$	
loop1									13.1
loop2									13.2
loop3									13.3
loop4									13.4

Problem 14 (6 %) check

```
Algorithm loop1(n)
  i = 1
  while i ≤ n
      i = 2 * i

Algorithm loop2(n)
  i = n
  while i > 0
      i = i - 1

Algorithm loop3(n)
  i = 1
  while i ≤ n * n
      i = 3 * i

Algorithm loop4(n)
  i = 1
  j = 1
  s = 0
  while s ≤ n
      while j ≤ s
          j = 2 * j
      s = s + i
      i = i + 1
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta(n^2)$	$\Theta(2^n)$	$\Theta(\sqrt{n})$	$\Theta(n^3)$	$\Theta(\log n)$	$\Theta(n \log n)$	$\Theta(n)$	$\Theta((\log n)^2)$	
loop1									14.1
loop2									14.2
loop3									14.3
loop4									14.4

Problem 15 (6 %) check

```
Algorithm loop1(n)
  s = 0
  for i = 1 to n
      for j = 1 to n
          for k = 1 to n
              s = s + 1
```

```
Algorithm loop2(n)
  for i = 1 to n
      j = i
      while j > 0
          j = j − 1
```

```
Algorithm loop3(n)
  i = 1
  while i ≤ n
      j = 0
      while j ≤ n
          j = j + 1
      i = 2 ∗ i
```

```
Algorithm loop4(n)
  i = n
  j = 0
  while i > 0
      if j < i
          j = j + 1
      else
          j = 0
          i = i − 1
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

$\Theta(n \log n)$ $\Theta((\log n)^2)$ $\Theta(n)$ $\Theta(n^3)$ $\Theta(\sqrt{n})$ $\Theta(\frac{\log n}{\log \log n})$ $\Theta(n^2)$ $\Theta(\log n)$

loop1 15.1

loop2 15.2

loop3 15.3

loop4 15.4

Problem 16 (6 %) **check**

```
Algorithm loop1(n)
  i = 1
  while i ≤ n
      j = 1
      while j ≤ n
          j = j + 1
      i = i + 1
```

```
Algorithm loop2(n)
  s = 0
  for i = 1 to n
      for j = 1 to i * i
          s = s + 1
```

```
Algorithm loop3(n)
  i = n
  while i ≤ n * n
      i = 2 * i
```

```
Algorithm loop4(n)
  i = 1
  while i ≤ n
      j = i
      while j ≤ n
          j = j + 1
      i = 2 * i
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta(n \log n)$	$\Theta(\log \log n)$	$\Theta(n^3)$	$\Theta(\log n)$	$\Theta(n)$	$\Theta((\log n)^2)$	$\Theta(n^2)$	$\Theta(n\sqrt{n})$	
loop1									16.1
loop2									16.2
loop3									16.3
loop4									16.4

Problem 17 (6 %) check

```
Algorithm loop1(n)
  s = 1
  for i = 1 to n * n
    for j = 1 to n
      s = s + 1
```

```
Algorithm loop2(n)
  s = 1
  for i = 1 to n
    s = s + 1
```

```
Algorithm loop3(n)
  s = 0
  i = n
  while i > 0
    for j = 1 to i
      s = s + 1
    i = i - 1
```

```
Algorithm loop4(n)
  i = n
  j = 0
  while i > 0
    if j < i
      j = j + 1
    else
      j = 0
      i = i - 1
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta(\sqrt{n}\log n)$	$\Theta((\log n)^2)$	$\Theta(n\log n)$	$\Theta(n^2)$	$\Theta(n^3)$	$\Theta(n)$	$\Theta(\log n)$	$\Theta(\sqrt{n})$	
loop1									17.1
loop2									17.2
loop3									17.3
loop4									17.4

Problem 18 (6 %) check

```
Algorithm loop1(n)
  s = 0
  for i = 1 to n
    for j = 1 to n
      for k = 1 to n
        s = s + 1
```

```
Algorithm loop2(n)
  i = n
  while i > 0
    i = i - 1
```

```
Algorithm loop3(n)
  i = 2
  while i ≤ n
    i = i * i
```

```
Algorithm loop4(n)
  for i = 1 to n
    j = i
    while j ≤ n
      j = 2 * j
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta(n^2)$	$\Theta(n)$	$\Theta(\log\log n)$	$\Theta(\sqrt{n})$	$\Theta((\log n)^2)$	$\Theta(n\log n)$	$\Theta(n^3)$	$\Theta(\log n)$	
loop1									18.1
loop2									18.2
loop3									18.3
loop4									18.4

Problem 19 (6 %) check

```
Algorithm loop1(n)
  for i = 1 to n
    j = i
    while j > 0
      j = j − 1

Algorithm loop2(n)
  i = 1
  while i ≤ n
    i = 2 ∗ i

Algorithm loop3(n)
  s = 0
  i = 1
  while s ≤ n
    s = s + i
    i = i + 1

Algorithm loop4(n)
  i = 1
  while i ≤ n
    j = i
    while j ≤ n
      j = j + 1
    i = 2 ∗ i
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta((\log n)^2)$	$\Theta(\sqrt{n})$	$\Theta(n^2 \cdot \log n)$	$\Theta(n^3)$	$\Theta(n^2)$	$\Theta(\log n)$	$\Theta(n)$	$\Theta(n \log n)$	
loop1									19.1
loop2									19.2
loop3									19.3
loop4									19.4

Problem 20 (6 %) check

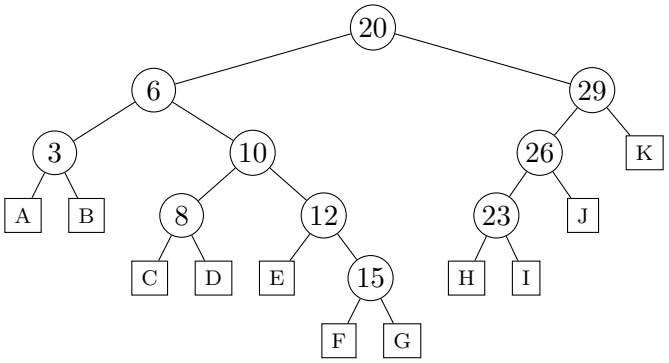
```
Algorithm loop1(n)
  for i = 1 to n
    j = i
    while j > 0
      j = j − 1

Algorithm loop2(n)
  s = 1
  for i = n to 1 step −1
    for j = n to 1 step −1
      s = s + 1

Algorithm loop3(n)
  i = 1
  while i ≤ n
    j = 1
    while j ≤ i
      j = 2 ∗ j
    i = i + 1

Algorithm loop4(n)
  i = 1
  j = n
  while i ≤ j
    i = 4 ∗ i
    j = 2 ∗ j
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta(n)$	$\Theta(n^3)$	$\Theta(\log n)$	$\Theta(\frac{\log n}{\log \log n})$	$\Theta(2^n)$	$\Theta((\log n)^2)$	$\Theta(n \log n)$	$\Theta(n^2)$	
loop1									20.1
loop2									20.2
loop3									20.3
loop4									20.4

Insertions into search trees

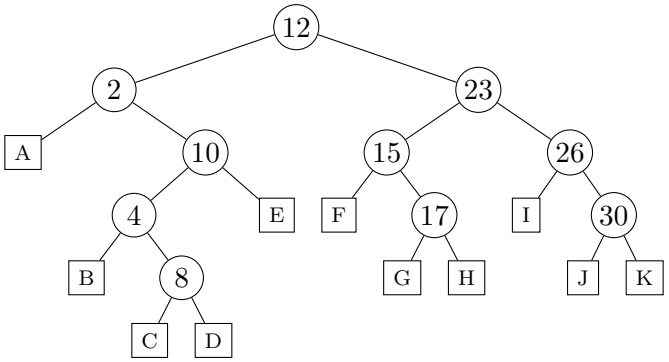
Problem 21 (4 %) check



Indicate in which leaves A–K in the above unbalanced binary search tree the elements 18, 2, 11, 25 and 22 should be inserted (it is assumed that before each insertion the tree contains only the above ten elements).

	A	B	C	D	E	F	G	H	I	J	K	
INSERT(18)												21.1
INSERT(2)												21.2
INSERT(11)												21.3
INSERT(25)												21.4
INSERT(22)												21.5

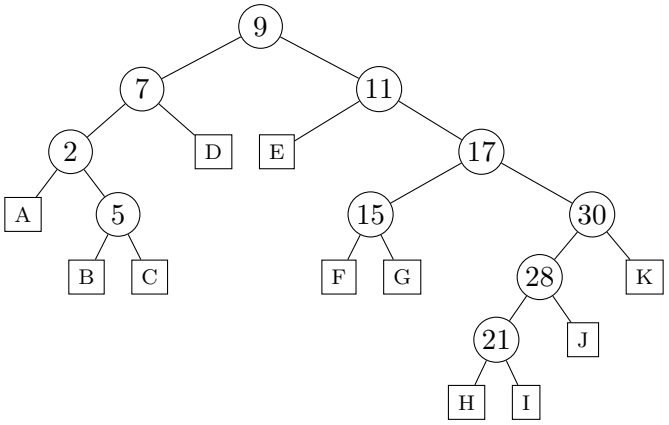
Problem 22 (4 %) check



Indicate in which leaves A–K in the above unbalanced binary search tree the elements 19, 28, 27, 1 and 21 should be inserted (it is assumed that before each insertion the tree contains only the above ten elements).

	A	B	C	D	E	F	G	H	I	J	K	
INSERT(19)												22.1
INSERT(28)												22.2
INSERT(27)												22.3
INSERT(1)												22.4
INSERT(21)												22.5

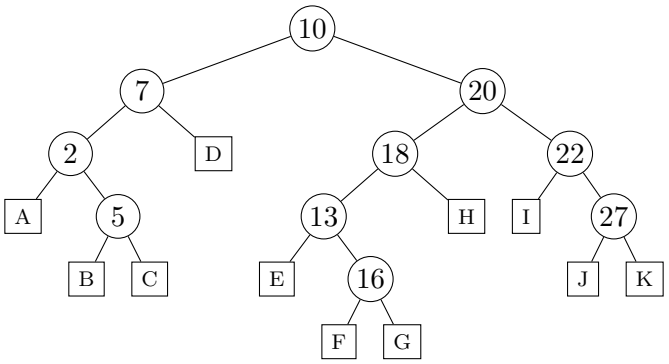
Problem 23 (4 %) **check**



Indicate in which leaves A–K in the above unbalanced binary search tree the elements 10, 3, 22, 25 and 20 should be inserted (it is assumed that before each insertion the tree contains only the above ten elements).

	A	B	C	D	E	F	G	H	I	J	K	
INSERT(10)												23.1
INSERT(3)												23.2
INSERT(22)												23.3
INSERT(25)												23.4
INSERT(20)												23.5

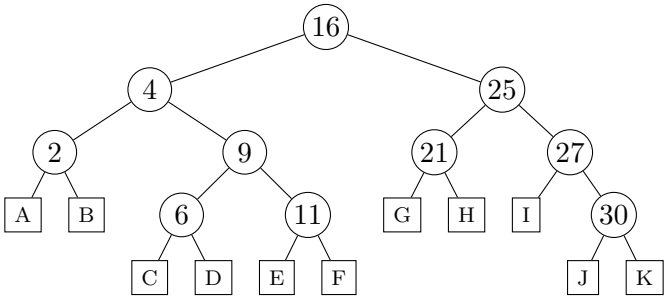
Problem 24 (4 %) **check**



Indicate in which leaves A–K in the above unbalanced binary search tree the elements 11, 19, 17, 28 and 1 should be inserted (it is assumed that before each insertion the tree contains only the above ten elements).

	A	B	C	D	E	F	G	H	I	J	K	
INSERT(11)												24.1
INSERT(19)												24.2
INSERT(17)												24.3
INSERT(28)												24.4
INSERT(1)												24.5

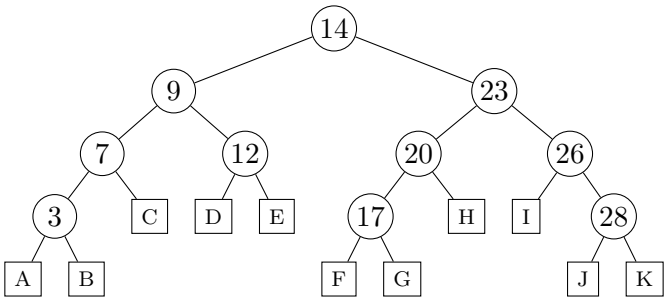
Problem 25 (4 %) **check**



Indicate in which leaves A–K in the above unbalanced binary search tree the elements 18, 5, 12, 19 and 22 should be inserted (it is assumed that before each insertion the tree contains only the above ten elements).

	A	B	C	D	E	F	G	H	I	J	K	
INSERT(18)												25.1
INSERT(5)												25.2
INSERT(12)												25.3
INSERT(19)												25.4
INSERT(22)												25.5

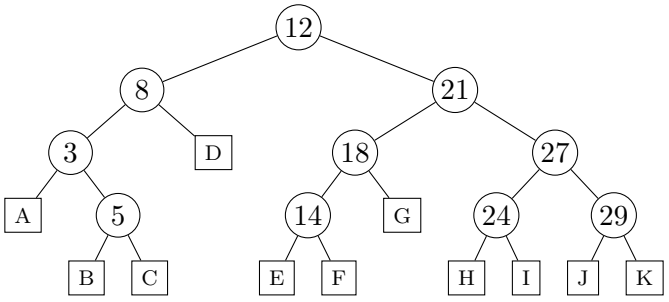
Problem 26 (4 %) **check**



Indicate in which leaves A–K in the above unbalanced binary search tree the elements 16, 2, 4, 19 and 22 should be inserted (it is assumed that before each insertion the tree contains only the above ten elements).

	A	B	C	D	E	F	G	H	I	J	K	
INSERT(16)												26.1
INSERT(2)												26.2
INSERT(4)												26.3
INSERT(19)												26.4
INSERT(22)												26.5

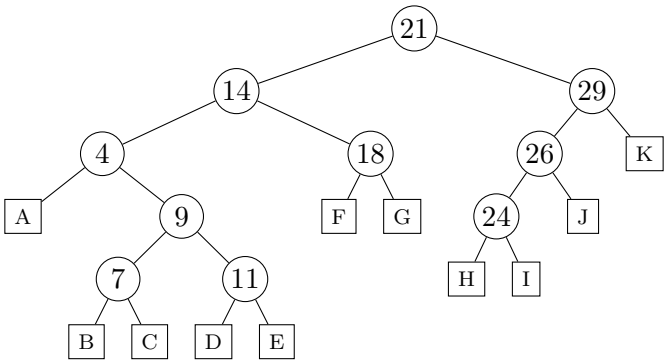
Problem 27 (4 %) **check**



Indicate in which leaves A–K in the above unbalanced binary search tree the elements 17, 10, 19, 15 and 23 should be inserted (it is assumed that before each insertion the tree contains only the above ten elements).

	A	B	C	D	E	F	G	H	I	J	K	
INSERT(17)												27.1
INSERT(10)												27.2
INSERT(19)												27.3
INSERT(15)												27.4
INSERT(23)												27.5

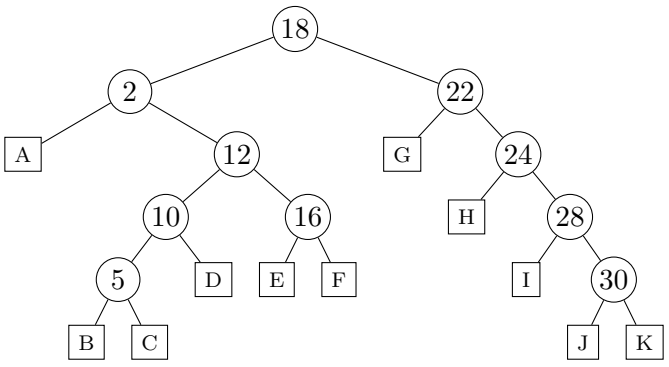
Problem 28 (4 %) **check**



Indicate in which leaves A–K in the above unbalanced binary search tree the elements 15, 20, 28, 16 and 12 should be inserted (it is assumed that before each insertion the tree contains only the above ten elements).

	A	B	C	D	E	F	G	H	I	J	K	
INSERT(15)												28.1
INSERT(20)												28.2
INSERT(28)												28.3
INSERT(16)												28.4
INSERT(12)												28.5

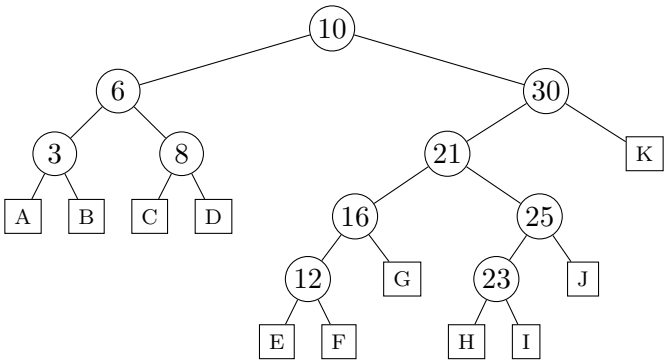
Problem 29 (4 %) **check**



Indicate in which leaves A–K in the above unbalanced binary search tree the elements 26, 29, 21, 8 and 6 should be inserted (it is assumed that before each insertion the tree contains only the above ten elements).

	A	B	C	D	E	F	G	H	I	J	K	
INSERT(26)												29.1
INSERT(29)												29.2
INSERT(21)												29.3
INSERT(8)												29.4
INSERT(6)												29.5

Problem 30 (4 %) **check**



Indicate in which leaves A–K in the above unbalanced binary search tree the elements 18, 7, 4, 24 and 9 should be inserted (it is assumed that before each insertion the tree contains only the above ten elements).

	A	B	C	D	E	F	G	H	I	J	K	
INSERT(18)												30.1
INSERT(7)												30.2
INSERT(4)												30.3
INSERT(24)												30.4
INSERT(9)												30.5

Max-Heap-Insert

Problem 31 (4 %) **check**

What is the binary max-heap after inserting the elements 3, 11, 13, 14, 1, 8 and 5 in the given order with MAX-HEAP-INSERT, starting with the empty heap.

1	2	3	4	5	6	7
14	13	11	3	1	8	5

1	2	3	4	5	6	7
14	11	13	3	1	8	5

1	2	3	4	5	6	7
3	11	13	14	1	8	5

1	2	3	4	5	6	7
14	13	11	8	5	3	1

1	2	3	4	5	6	7
13	14	8	11	1	3	5

Problem 32 (4 %) check

What is the binary max-heap after inserting the elements 2, 7, 4, 9, 8, 11 and 3 in the given order with Max-Heap-Insert, starting with the empty heap.

1	2	3	4	5	6	7
11	8	9	2	7	4	3

1	2	3	4	5	6	7
11	9	8	7	4	3	2

1	2	3	4	5	6	7
7	9	11	2	8	4	3

1	2	3	4	5	6	7
11	9	4	7	8	2	3

1	2	3	4	5	6	7
2	7	4	9	8	11	3

Problem 33 (4 %) check

What is the binary max-heap after inserting the elements 10, 2, 1, 13, 12, 14 and 6 in the given order with Max-Heap-Insert, starting with the empty heap.

1	2	3	4	5	6	7
14	13	10	2	12	1	6

1	2	3	4	5	6	7
14	12	13	2	10	1	6

1	2	3	4	5	6	7
14	13	12	10	6	2	1

1	2	3	4	5	6	7
10	13	14	2	12	1	6

1	2	3	4	5	6	7
10	2	1	13	12	14	6

Problem 34 (4 %) check

What is the binary max-heap after inserting the elements 10, 12, 1, 14, 6, 11 and 13 in the given order with Max-Heap-Insert, starting with the empty heap.

1	2	3	4	5	6	7
10	12	1	14	6	11	13

1	2	3	4	5	6	7
14	13	12	11	10	6	1

1	2	3	4	5	6	7
12	14	13	10	6	11	1

1	2	3	4	5	6	7
14	12	13	10	6	1	11

1	2	3	4	5	6	7
14	12	13	10	6	11	1

Problem 35 (4 %) check

What is the binary max-heap after inserting the elements 4, 10, 1, 2, 5, 12 and 3 in the given order with Max-Heap-Insert, starting with the empty heap.

1	2	3	4	5	6	7
12	10	5	4	3	2	1

1	2	3	4	5	6	7
12	10	4	2	5	1	3

1	2	3	4	5	6	7
12	5	10	2	4	1	3

1	2	3	4	5	6	7
10	5	12	2	4	1	3

1	2	3	4	5	6	7
4	10	1	2	5	12	3

Problem 36 (4 %) check

What is the binary max-heap after inserting the elements 6, 3, 9, 11, 12, 8 and 14 in the given order with Max-Heap-Insert, starting with the empty heap.

1	2	3	4	5	6	7
6	3	9	11	12	8	14

1	2	3	4	5	6	7
9	12	14	11	3	8	6

1	2	3	4	5	6	7
14	12	11	9	8	6	3

1	2	3	4	5	6	7
14	11	12	3	9	6	8

1	2	3	4	5	6	7
14	12	9	11	3	8	6

Problem 37 (4 %) check

What is the binary max-heap after inserting the elements 3, 12, 4, 10, 11, 2 and 14 in the given order with Max-Heap-Insert, starting with the empty heap.

1	2	3	4	5	6	7
14	12	4	10	11	2	3

1	2	3	4	5	6	7
3	12	4	10	11	2	14

1	2	3	4	5	6	7
14	11	12	3	10	2	4

1	2	3	4	5	6	7
12	11	14	10	3	2	4

1	2	3	4	5	6	7
14	12	11	10	4	3	2

Problem 38 (4 %) check

What is the binary max-heap after inserting the elements 6, 8, 10, 14, 7, 1 and 12 in the given order with Max-Heap-Insert, starting with the empty heap.

1	2	3	4	5	6	7
14	8	12	6	7	1	10

1	2	3	4	5	6	7
14	12	10	8	7	6	1

1	2	3	4	5	6	7
10	14	12	8	7	1	6

1	2	3	4	5	6	7
14	10	12	6	7	1	8

1	2	3	4	5	6	7
6	8	10	14	7	1	12

Problem 39 (4 %) check

What is the binary max-heap after inserting the elements 2, 8, 1, 5, 10, 9 and 6 in the given order with Max-Heap-Insert, starting with the empty heap.

1	2	3	4	5	6	7
10	8	9	2	5	1	6

1	2	3	4	5	6	7
10	8	9	5	2	1	6

1	2	3	4	5	6	7
8	10	9	5	2	1	6

1	2	3	4	5	6	7
2	8	1	5	10	9	6

1	2	3	4	5	6	7
10	9	8	6	5	2	1

Problem 40 (4 %) check

What is the binary max-heap after inserting the elements 10, 5, 6, 9, 8, 11 and 14 in the given order with Max-Heap-Insert, starting with the empty heap.

1	2	3	4	5	6	7
10	9	14	5	8	11	6

1	2	3	4	5	6	7
14	9	11	5	8	6	10

1	2	3	4	5	6	7
10	5	6	9	8	11	14

1	2	3	4	5	6	7
14	9	11	5	8	10	6

1	2	3	4	5	6	7
14	11	10	9	8	6	5

Build-Max-Heap

Problem 41 (4 %) check

1	2	3	4	5	6	7	8	9
2	5	6	1	7	9	4	8	3

What is the result of applying Build-Max-Heap on the above array?

1	2	3	4	5	6	7	8	9
9	8	6	5	7	2	4	1	3

1	2	3	4	5	6	7	8	9
9	8	7	6	5	4	3	2	1

1	2	3	4	5	6	7	8	9
6	7	9	8	5	2	4	1	3

1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9

1	2	3	4	5	6	7	8	9
9	8	7	6	2	5	4	1	3

Problem 42 (4 %) check

1	2	3	4	5	6	7	8	9
4	3	1	8	2	7	9	5	6

What is the result of applying Build-Max-Heap on the above array?

1	2	3	4	5	6	7	8	9
9	8	7	6	2	4	1	5	3

1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9

1	2	3	4	5	6	7	8	9
9	8	7	6	5	4	3	2	1

1	2	3	4	5	6	7	8	9
4	8	9	6	2	7	1	5	3

1	2	3	4	5	6	7	8	9
9	6	8	5	2	1	7	3	4

Problem 43 (4 %) check

1	2	3	4	5	6	7	8	9
5	6	9	2	7	3	1	8	4

What is the result of applying Build-Max-Heap on the above array?

1	2	3	4	5	6	7	8	9
9	7	5	8	6	3	1	2	4

1	2	3	4	5	6	7	8	9
9	8	5	6	7	3	1	2	4

1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9

1	2	3	4	5	6	7	8	9
9	8	7	6	5	4	3	2	1

1	2	3	4	5	6	7	8	9
9	8	6	7	5	3	1	2	4

Problem 44 (4 %) check

1	2	3	4	5	6	7	8	9
9	3	6	5	7	2	1	8	4

What is the result of applying Build-Max-Heap on the above array?

1	2	3	4	5	6	7	8	9
9	8	6	7	5	2	1	3	4

1	2	3	4	5	6	7	8	9
9	8	7	6	5	4	3	2	1

1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9

1	2	3	4	5	6	7	8	9
9	8	6	5	7	2	1	3	4

1	2	3	4	5	6	7	8	9
9	7	6	8	3	2	1	5	4

Problem 45 (4 %) check

1	2	3	4	5	6	7	8	9
3	7	5	8	1	9	2	6	4

What is the result of applying Build-Max-Heap on the above array?

1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9

1	2	3	4	5	6	7	8	9
7	8	9	6	1	5	2	3	4

1	2	3	4	5	6	7	8	9
9	7	8	6	1	5	2	3	4

1	2	3	4	5	6	7	8	9
9	8	5	7	1	3	2	6	4

1	2	3	4	5	6	7	8	9
9	8	7	6	5	4	3	2	1

Problem 46 (4 %) check

1	2	3	4	5	6	7	8	9
8	1	7	6	5	2	9	3	4

What is the result of applying Build-Max-Heap on the above array?

1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9

1	2	3	4	5	6	7	8	9
8	6	9	4	5	2	7	3	1

1	2	3	4	5	6	7	8	9
9	8	7	6	5	4	3	2	1

1	2	3	4	5	6	7	8	9
9	6	8	4	5	2	7	1	3

1	2	3	4	5	6	7	8	9
9	6	8	4	5	2	7	3	1

Problem 47 (4 %) check

1	2	3	4	5	6	7	8	9
1	4	7	6	8	9	2	3	5

What is the result of applying Build-Max-Heap on the above array?

1	2	3	4	5	6	7	8	9
9	7	8	5	6	4	2	1	3

1	2	3	4	5	6	7	8	9
9	8	7	6	5	4	3	2	1

1	2	3	4	5	6	7	8	9
9	8	7	6	4	1	2	3	5

1	2	3	4	5	6	7	8	9
7	8	9	6	4	1	2	3	5

1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9

Problem 48 (4 %) check

1	2	3	4	5	6	7	8	9
2	1	5	6	7	3	9	8	4

What is the result of applying Build-Max-Heap on the above array?

1	2	3	4	5	6	7	8	9
9	8	7	6	5	2	3	1	4

1	2	3	4	5	6	7	8	9
5	7	9	8	1	3	2	6	4

1	2	3	4	5	6	7	8	9
9	8	7	6	5	4	3	2	1

1	2	3	4	5	6	7	8	9
9	8	5	6	7	3	2	1	4

1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9

Problem 49 (4 %) check

1	2	3	4	5	6	7	8	9
7	3	5	8	6	9	1	4	2

What is the result of applying Build-Max-Heap on the above array?

1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9

1	2	3	4	5	6	7	8	9
7	8	9	4	6	5	1	3	2

1	2	3	4	5	6	7	8	9
9	8	7	6	5	4	3	2	1

1	2	3	4	5	6	7	8	9
9	8	7	4	6	5	1	3	2

1	2	3	4	5	6	7	8	9
9	7	8	4	6	5	1	3	2

Problem 50 (4 %) check

1	2	3	4	5	6	7	8	9
7	8	4	3	6	1	2	5	9

What is the result of applying Build-Max-Heap on the above array?

1	2	3	4	5	6	7	8	9
9	8	4	7	6	1	2	3	5

1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9

1	2	3	4	5	6	7	8	9
8	7	4	9	6	1	2	5	3

1	2	3	4	5	6	7	8	9
9	8	4	7	6	1	2	5	3

1	2	3	4	5	6	7	8	9
9	8	7	6	5	4	3	2	1

Heap-Extract-Max

Problem 51 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13
25	24	19	18	23	10	16	14	6	1	7	4	8

What is the result of applying Heap-Extract-Max to the above max-heap?

1	2	3	4	5	6	7	8	9	10	11	12
24	23	19	18	8	10	16	14	6	1	7	4

1	2	3	4	5	6	7	8	9	10	11	12	13
24	23	19	18	7	10	16	14	6	1		4	8

1	2	3	4	5	6	7	8	9	10	11	12
24	23	19	18	7	10	16	14	6	1	8	4

1	2	3	4	5	6	7	8	9	10	11	12
24	23	19	18	7	10	16	14	6	1	4	8

1	2	3	4	5	6	7	8	9	10	11	12
24	23	18	19	10	16	14	6	1	7	4	8

Problem 52 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13
25	24	23	8	16	18	9	4	3	7	10	13	15

What is the result of applying Heap-Extract-Max to the above max-heap?

1	2	3	4	5	6	7	8	9	10	11	12
24	16	23	8	10	18	9	4	3	7	15	13

1	2	3	4	5	6	7	8	9	10	11	12
24	23	15	16	18	9	4	3	7	10	13	8

1	2	3	4	5	6	7	8	9	10	11	12
24	16	23	8	10	18	9	4	3	7	13	15

1	2	3	4	5	6	7	8	9	10	11	12	13
24	16	23	8	10	18	9	4	3	7		13	15

1	2	3	4	5	6	7	8	9	10	11	12
24	16	23	8	15	18	9	4	3	7	10	13

Problem 53 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13
25	24	22	19	17	13	20	7	6	11	5	9	8

What is the result of applying Heap-Extract-Max to the above max-heap?

1	2	3	4	5	6	7	8	9	10	11	12
24	19	22	7	17	13	20	8	6	11	5	9

1	2	3	4	5	6	7	8	9	10	11	12	13
24	19	22	7	17	13	20		6	11	5	9	8

1	2	3	4	5	6	7	8	9	10	11	12
24	19	22	8	17	13	20	7	6	11	5	9

1	2	3	4	5	6	7	8	9	10	11	12
24	22	20	17	13	19	7	6	11	5	9	8

1	2	3	4	5	6	7	8	9	10	11	12
24	19	22	7	17	13	20	6	11	5	9	8

Problem 54 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13
25	18	16	13	12	15	8	5	4	7	1	14	6

What is the result of applying Heap-Extract-Max to the above max-heap?

1	2	3	4	5	6	7	8	9	10	11	12
18	13	16	5	12	15	8	6	4	7	1	14

1	2	3	4	5	6	7	8	9	10	11	12
18	13	16	5	12	15	8	4	7	1	14	6

1	2	3	4	5	6	7	8	9	10	11	12
18	13	16	6	12	15	8	5	4	7	1	14

1	2	3	4	5	6	7	8	9	10	11	12	13
18	13	16	5	12	15	8		4	7	1	14	6

1	2	3	4	5	6	7	8	9	10	11	12
18	16	13	12	15	8	5	4	7	1	14	6

Problem 55 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13
25	24	22	21	19	20	5	1	16	14	11	3	18

What is the result of applying Heap-Extract-Max to the above max-heap?

1	2	3	4	5	6	7	8	9	10	11	12
24	22	21	19	20	18	1	16	14	11	3	5

1	2	3	4	5	6	7	8	9	10	11	12
24	21	22	16	19	20	5	1	14	11	3	18

1	2	3	4	5	6	7	8	9	10	11	12
24	21	22	18	19	20	5	1	16	14	11	3

1	2	3	4	5	6	7	8	9	10	11	12	13
24	21	22	16	19	20	5	1		14	11	3	18

1	2	3	4	5	6	7	8	9	10	11	12
24	21	22	16	19	20	5	1	18	14	11	3

Problem 56 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13
25	24	18	22	19	13	4	7	9	17	12	6	10

What is the result of applying Heap-Extract-Max to the above max-heap?

1	2	3	4	5	6	7	8	9	10	11	12
24	22	18	9	19	13	4	7	17	12	6	10

1	2	3	4	5	6	7	8	9	10	11	12
24	22	18	10	19	13	4	7	9	17	12	6

1	2	3	4	5	6	7	8	9	10	11	12
24	19	22	18	13	10	7	9	17	12	6	4

1	2	3	4	5	6	7	8	9	10	11	12
24	22	18	9	19	13	4	7	10	17	12	6

1	2	3	4	5	6	7	8	9	10	11	12	13
24	22	18	9	19	13	4	7		17	12	6	10

Problem 57 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13
25	24	21	18	14	19	11	8	4	9	6	2	13

What is the result of applying Heap-Extract-Max to the above max-heap?

1	2	3	4	5	6	7	8	9	10	11	12
24	21	18	14	19	13	8	4	9	6	2	11

1	2	3	4	5	6	7	8	9	10	11	12
24	18	21	13	14	19	11	8	4	9	6	2

1	2	3	4	5	6	7	8	9	10	11	12
24	18	21	8	14	19	11	4	9	6	2	13

1	2	3	4	5	6	7	8	9	10	11	12
24	18	21	8	14	19	11	13	4	9	6	2

1	2	3	4	5	6	7	8	9	10	11	12	13
24	18	21	8	14	19	11		4	9	6	2	13

Problem 58 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13
23	22	18	21	15	16	13	6	9	11	14	5	12

What is the result of applying Heap-Extract-Max to the above max-heap?

1	2	3	4	5	6	7	8	9	10	11	12	13
22	21	18	9	15	16	13	6		11	14	5	12

1	2	3	4	5	6	7	8	9	10	11	12
22	21	18	9	15	16	13	6	12	11	14	5

1	2	3	4	5	6	7	8	9	10	11	12
22	18	21	15	16	13	6	9	11	14	5	12

1	2	3	4	5	6	7	8	9	10	11	12
22	21	18	9	15	16	13	6	11	14	5	12

1	2	3	4	5	6	7	8	9	10	11	12
22	21	18	12	15	16	13	6	9	11	14	5

Problem 59 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13
26	25	22	19	10	13	20	8	7	5	1	6	9

What is the result of applying Heap-Extract-Max to the above max-heap?

1	2	3	4	5	6	7	8	9	10	11	12
25	19	22	8	10	13	20	9	7	5	1	6

1	2	3	4	5	6	7	8	9	10	11	12
25	19	22	9	10	13	20	8	7	5	1	6

1	2	3	4	5	6	7	8	9	10	11	12
25	22	20	10	13	19	8	7	5	1	6	9

1	2	3	4	5	6	7	8	9	10	11	12
25	19	22	8	10	13	20	7	5	1	6	9

1	2	3	4	5	6	7	8	9	10	11	12	13
25	19	22	8	10	13	20		7	5	1	6	9

Problem 60 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13
26	24	22	13	23	19	8	11	12	2	10	4	15

What is the result of applying Heap-Extract-Max to the above max-heap?

1	2	3	4	5	6	7	8	9	10	11	12	13
24	23	22	13	10	19	8	11	12	2		4	15

1	2	3	4	5	6	7	8	9	10	11	12
24	23	22	13	10	19	8	11	12	2	4	15

1	2	3	4	5	6	7	8	9	10	11	12
24	23	15	22	19	13	11	12	2	10	4	8

1	2	3	4	5	6	7	8	9	10	11	12
24	23	22	13	15	19	8	11	12	2	10	4

1	2	3	4	5	6	7	8	9	10	11	12
24	23	22	13	10	19	8	11	12	2	15	4

Partition

Problem 61 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14	17	1	25	3	13	22	24	9	7	29	30	10	23	12

Indicate the result of applying PARTITION$(A, 3, 14)$ to the above array A.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14	17	1	3	7	9	10	13	22	23	24	25	29	30	12

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14	17	1	3	13	22	9	7	10	23	29	30	25	24	12

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14	17	1	3	13	22	9	7	10	23	25	24	29	30	12

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	3	7	9	10	12	13	14	17	22	23	24	25	29	30

Problem 62 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	21	26	23	28	18	22	19	8	4	20	6	24	12	3

Indicate the result of applying PARTITION$(A, 2, 12)$ to the above array A.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	4	6	23	28	18	22	19	8	21	20	26	24	12	3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
3	4	6	8	12	15	18	19	20	21	22	23	24	26	28

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	4	6	8	18	19	20	21	22	23	26	28	24	12	3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	4	6	21	26	23	28	18	22	19	8	20	24	12	3

Problem 63 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
21	17	8	23	29	2	18	26	22	20	25	11	1	9	28

Indicate the result of applying PARTITION$(A, 3, 13)$ to the above array A.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	8	9	11	17	18	20	21	22	23	25	26	28	29

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
21	17	1	8	23	29	2	18	26	22	20	25	11	9	28

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
21	17	1	2	8	11	18	20	22	23	25	26	29	9	28

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
21	17	1	23	29	2	18	26	22	20	25	11	8	9	28

Problem 64 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
17	13	3	15	4	27	30	21	22	1	20	8	12	11	25

Indicate the result of applying PARTITION$(A, 4, 12)$ to the above array A.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
17	13	3	4	1	8	30	21	22	15	20	27	12	11	25

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
17	13	3	1	4	8	15	20	21	22	27	30	12	11	25

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
17	13	3	4	1	8	15	27	30	21	22	20	12	11	25

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	3	4	8	11	12	13	15	17	20	21	22	25	27	30

Problem 65 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
21	24	30	25	27	3	17	29	4	15	23	2	6	5	10

Indicate the result of applying PARTITION$(A, 4, 12)$ to the above array A.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
21	24	30	2	3	4	15	17	23	25	27	29	6	5	10

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
21	24	30	2	27	3	17	29	4	15	23	25	6	5	10

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	3	4	5	6	10	15	17	21	23	24	25	27	29	30

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
21	24	30	2	25	27	3	17	29	4	15	23	6	5	10

Problem 66 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
30	12	26	1	5	19	8	28	29	9	22	18	2	27	21

Indicate the result of applying PARTITION$(A, 4, 13)$ to the above array A.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
30	12	26	1	2	19	8	28	29	9	22	18	5	27	21

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
30	12	26	1	2	5	8	9	18	19	22	28	29	27	21

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
30	12	26	1	2	5	19	8	28	29	9	22	18	27	21

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	5	8	9	12	18	19	21	22	26	27	28	29	30

Problem 67 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	26	15	10	27	25	19	6	8	3	16	23	4	1	22

Indicate the result of applying PARTITION$(A, 3, 14)$ to the above array A.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	26	1	10	27	25	19	6	8	3	16	23	4	15	22

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	3	4	6	8	10	12	15	16	19	22	23	25	26	27

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	26	1	15	10	27	25	19	6	8	3	16	23	4	22

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	26	1	3	4	6	8	10	15	16	19	23	25	27	22

Problem 68 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
19	24	1	11	23	6	18	21	22	10	3	2	25	13	12

Indicate the result of applying PARTITION$(A, 4, 14)$ to the above array A.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
19	24	1	11	6	10	3	2	13	23	18	21	22	25	12

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
19	24	1	2	3	6	10	11	13	18	21	22	23	25	12

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	3	6	10	11	12	13	18	19	21	22	23	24	25

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
19	24	1	11	6	10	3	2	13	23	18	21	25	22	12

Problem 69 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	27	11	15	21	23	10	12	1	5	16	19	13	18	28

Indicate the result of applying PARTITION$(A, 2, 13)$ to the above array A.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	11	10	12	1	5	13	15	21	23	16	19	27	18	28

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	11	10	12	1	5	13	27	15	21	23	16	19	18	28

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	5	7	10	11	12	13	15	16	18	19	21	23	27	28

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	1	5	10	11	12	13	15	16	19	21	23	27	18	28

Problem 70 (4 %) check

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
11	5	19	13	28	22	15	29	30	17	2	18	12	14	10

Indicate the result of applying PARTITION$(A, 3, 14)$ to the above array A.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
11	5	2	12	13	14	15	17	18	19	22	28	29	30	10

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	5	10	11	12	13	14	15	17	18	19	22	28	29	30

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
11	5	13	2	12	14	19	28	22	15	29	30	17	18	10

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
11	5	13	2	12	14	15	29	30	17	19	18	28	22	10

Radix-sort

Problem 71 (4 %) check

2143 2102 0424 3324 1143 1224

Consider Radix-Sort applied to the above list of numbers ($d = 4$, $k = 5$). Indicate the partially sorted list after Radix-Sort has sorted the numbers according to the *two* least significant digits.

2102	0424	3324	1224	2143	1143
2102	2143	1143	0424	3324	1224
0424	1143	1224	2102	2143	3324
0424	1143	1224	2143	2102	3324
2102	0424	1224	3324	1143	2143

Problem 72 (4 %) check

4010 3330 4410 1013 1430 1010

Consider Radix-Sort applied to the above list of numbers ($d = 4$, $k = 5$). Indicate the partially sorted list after Radix-Sort has sorted the numbers according to the *two* least significant digits.

1010	1013	1430	3330	4010	4410
4010	4410	1010	1013	3330	1430
4010	4410	1010	3330	1430	1013
1013	1010	1430	3330	4010	4410
1010	4010	4410	1013	1430	3330

Problem 73 (4 %) check

2124 3404 2024 4324 1013 2013

Consider Radix-Sort applied to the above list of numbers ($d = 4$, $k = 5$). Indicate the partially sorted list after Radix-Sort has sorted the numbers according to the *two* least significant digits.

1013	2013	3404	2124	2024	4324
1013	2024	2013	2124	3404	4324
1013	2013	2024	2124	3404	4324
3404	1013	2013	2024	2124	4324
3404	1013	2013	2124	2024	4324

Problem 74 (4 %) check

1143 4432 0234 1134 1432 4034

Consider Radix-Sort applied to the above list of numbers ($d = 4$, $k = 5$). Indicate the partially sorted list after Radix-Sort has sorted the numbers according to the *two* least significant digits.

4432	1432	0234	1134	4034	1143
0234	1143	1134	1432	4034	4432
0234	1134	1143	1432	4034	4432
1432	4432	0234	1134	4034	1143
4432	1432	1143	0234	1134	4034

Problem 75 (4 %) check

2042 0041 4041 2241 4112 2212

Consider Radix-Sort applied to the above list of numbers ($d = 4$, $k = 5$). Indicate the partially sorted list after Radix-Sort has sorted the numbers according to the *two* least significant digits.

4112	2212	0041	4041	2241	2042
0041	2042	2241	2212	4041	4112
0041	2042	2212	2241	4041	4112
0041	4041	2241	4112	2212	2042
2212	4112	0041	2241	4041	2042

Problem 76 (4 %) check

4040 2123 0430 0330 0423 4123

Consider Radix-Sort applied to the above list of numbers ($d = 4$, $k = 5$). Indicate the partially sorted list after Radix-Sort has sorted the numbers according to the *two* least significant digits.

0430	0330	4040	2123	0423	4123
0330	0423	0430	2123	4040	4123
0330	0430	0423	2123	4040	4123
0423	2123	4123	0330	0430	4040
2123	0423	4123	0430	0330	4040

Problem 77 (4 %) check

3331 4143 2031 0031 4012 2012

Consider Radix-Sort applied to the above list of numbers ($d = 4$, $k = 5$). Indicate the partially sorted list after Radix-Sort has sorted the numbers according to the *two* least significant digits.

4012	2012	3331	2031	0031	4143
0031	2031	2012	3331	4012	4143
0031	2012	2031	3331	4012	4143
2012	4012	0031	2031	3331	4143
3331	2031	0031	4012	2012	4143

Problem 78 (4 %) check

0321 4000 2121 2302 2300 1102

Consider Radix-Sort applied to the above list of numbers ($d = 4$, $k = 5$). Indicate the partially sorted list after Radix-Sort has sorted the numbers according to the *two* least significant digits.

4000	2300	0321	2121	2302	1102
4000	2300	2302	1102	0321	2121
2300	4000	1102	2302	0321	2121
0321	1102	2121	2300	2302	4000
0321	1102	2121	2302	2300	4000

Problem 79 (4 %) check

4033 4313 2433 3120 4333 4320

Consider Radix-Sort applied to the above list of numbers ($d = 4$, $k = 5$). Indicate the partially sorted list after Radix-Sort has sorted the numbers according to the *two* least significant digits.

4313	3120	4320	4033	2433	4333
4313	3120	4320	2433	4033	4333
2433	3120	4033	4313	4333	4320
3120	4320	4313	4033	2433	4333
2433	3120	4033	4313	4320	4333

Problem 80 (4 %) check

3241 2123 1023 0023 3233 0433

Consider RADIX-SORT applied to the above list of numbers ($d = 4$, $k = 5$). Indicate the partially sorted list after RADIX-SORT has sorted the numbers according to the *two* least significant digits.

0023	0433	1023	2123	3241	3233
2123	1023	0023	3233	0433	3241
3241	2123	1023	0023	3233	0433
0023	0433	1023	2123	3233	3241
0023	1023	2123	0433	3233	3241

Linear probing

Problem 81 (4 %) check

0	1	2	3	4	5	6	7	8	9	10
11				5	16				3	14

In the above hash table of size 11, *linear probing* is used with hash function $h(k) = 3k \bmod 11$. Indicate the positions the five elements 0, 2, 7, 8 and 10 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 3, 5, 11, 14 and 16).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(0)												81.1
INSERT(2)												81.2
INSERT(7)												81.3
INSERT(8)												81.4
INSERT(10)												81.5

Problem 82 (4 %) check

0	1	2	3	4	5	6	7	8	9	10
22	6							15	4	21

In the above hash table of size 11, *linear probing* is used with hash function $h(k) = 2k \bmod 11$. Indicate the positions the five elements 1, 2, 3, 5 and 11 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 4, 6, 15, 21 and 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(1)												82.1
INSERT(2)												82.2
INSERT(3)												82.3
INSERT(5)												82.4
INSERT(11)												82.5

Problem 83 (4 %) check

0	1	2	3	4	5	6	7	8	9	10
22	0							6	17	15

In the above hash table of size 11, *linear probing* is used with hash function $h(k) = 5k \bmod 11$. Indicate the positions the five elements 1, 3, 4, 9 and 10 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 0, 6, 15, 17 and 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(1)												83.1
INSERT(3)												83.2
INSERT(4)												83.3
INSERT(9)												83.4
INSERT(10)												83.5

Problem 84 (4 %) **check**

0	1	2	3	4	5	6	7	8	9	10
11	22	20							4	13

In the above hash table of size 11, *linear probing* is used with hash function $h(k) = 5k \bmod 11$. Indicate the positions the five elements 0, 2, 3, 5 and 6 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 4, 11, 13, 20 and 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(0)												84.1
INSERT(2)												84.2
INSERT(3)												84.3
INSERT(5)												84.4
INSERT(6)												84.5

Problem 85 (4 %) **check**

0	1	2	3	4	5	6	7	8	9	10
		1				3	14		21	5

In the above hash table of size 11, *linear probing* is used with hash function $h(k) = 2k \bmod 11$. Indicate the positions the five elements 2, 4, 6, 9 and 10 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 1, 3, 5, 14 and 21).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(2)												85.1
INSERT(4)												85.2
INSERT(6)												85.3
INSERT(9)												85.4
INSERT(10)												85.5

Problem 86 (4 %) **check**

0	1	2	3	4	5	6	7	8	9	10
11	0					21			4	15

In the above hash table of size 11, *linear probing* is used with hash function $h(k) = 5k \bmod 11$. Indicate the positions the five elements 2, 3, 5, 7 and 9 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 0, 4, 11, 15 and 21).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(2)												86.1
INSERT(3)												86.2
INSERT(5)												86.3
INSERT(7)												86.4
INSERT(9)												86.5

Problem 87 (4 %) **check**

0	1	2	3	4	5	6	7	8	9	10
0				2		14	3			16

In the above hash table of size 11, *linear probing* is used with hash function $h(k) = 2k \bmod 11$. Indicate the positions the five elements 4, 5, 6, 10 and 11 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 0, 2, 3, 14 and 16).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(4)												87.1
INSERT(5)												87.2
INSERT(6)												87.3
INSERT(10)												87.4
INSERT(11)												87.5

Problem 88 (4 %) check

0	1	2	3	4	5	6	7	8	9	10
22			7	18				15		16

In the above hash table of size 11, *linear probing* is used with hash function $h(k) = 2k \bmod 11$. Indicate the positions the five elements 0, 2, 3, 5 and 6 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 7, 15, 16, 18 and 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(0)												88.1
INSERT(2)												88.2
INSERT(3)												88.3
INSERT(5)												88.4
INSERT(6)												88.5

Problem 89 (4 %) check

0	1	2	3	4	5	6	7	8	9	10
22	20	0							15	2

In the above hash table of size 11, *linear probing* is used with hash function $h(k) = 5k \bmod 11$. Indicate the positions the five elements 4, 6, 7, 8 and 9 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 0, 2, 15, 20 and 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(4)												89.1
INSERT(6)												89.2
INSERT(7)												89.3
INSERT(8)												89.4
INSERT(9)												89.5

Problem 90 (4 %) check

0	1	2	3	4	5	6	7	8	9	10
		1			19			15	4	16

In the above hash table of size 11, *linear probing* is used with hash function $h(k) = 2k \bmod 11$. Indicate the positions the five elements 2, 6, 7, 8 and 10 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 1, 4, 15, 16 and 19).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(2)												90.1
INSERT(6)												90.2
INSERT(7)												90.3
INSERT(8)												90.4
INSERT(10)												90.5

Quadratic probing

Problem 91 (4 %) check

0	1	2	3	4	5	6	7	8	9	10
		1	12			13			18	7

In the above hash table of size 11, *quadratic probing* is used with hash functions $h'(k) = 3k \bmod 11$ and $h(k,i) = (h'(k) + 5i + 5i^2) \bmod 11$. Indicate the positions the five elements 0, 2, 3, 5 and 9 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 1, 7, 12, 13 and 18).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(0)												91.1
INSERT(2)												91.2
INSERT(3)												91.3
INSERT(5)												91.4
INSERT(9)												91.5

Problem 92 (4 %) **check**

0	1	2	3	4	5	6	7	8	9	10
22					0	18	10		5	

In the above hash table of size 11, *quadratic probing* is used with hash functions $h'(k) = 4k \bmod 11$ and $h(k,i) = (h'(k) + 2i + 3i^2) \bmod 11$. Indicate the positions the five elements 2, 3, 7, 9 and 11 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 0, 5, 10, 18 and 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(2)												92.1
INSERT(3)												92.2
INSERT(7)												92.3
INSERT(9)												92.4
INSERT(11)												92.5

Problem 93 (4 %) **check**

0	1	2	3	4	5	6	7	8	9	10
22	6					10		15	17	

In the above hash table of size 11, *quadratic probing* is used with hash functions $h'(k) = 2k \bmod 11$ and $h(k,i) = (h'(k) + 3i + 5i^2) \bmod 11$. Indicate the positions the five elements 0, 2, 4, 7 and 9 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 6, 10, 15, 17 and 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(0)												93.1
INSERT(2)												93.2
INSERT(4)												93.3
INSERT(7)												93.4
INSERT(9)												93.5

Problem 94 (4 %) check

0	1	2	3	4	5	6	7	8	9	10
	15	6			20		17			18

In the above hash table of size 11, *quadratic probing* is used with hash functions $h'(k) = 3k \bmod 11$ and $h(k,i) = (h'(k) + 2i + 4i^2) \bmod 11$. Indicate the positions the five elements 0, 4, 7, 9 and 10 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 6, 15, 17, 18 and 20).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(0)												94.1
INSERT(4)												94.2
INSERT(7)												94.3
INSERT(9)												94.4
INSERT(10)												94.5

Problem 95 (4 %) check

0	1	2	3	4	5	6	7	8	9	10
0					12	10		17	11	

In the above hash table of size 11, *quadratic probing* is used with hash functions $h'(k) = 5k \bmod 11$ and $h(k,i) = (h'(k) + 5i^2) \bmod 11$. Indicate the positions the five elements 1, 3, 4, 5 and 9 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 0, 10, 11, 12 and 17).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(1)												95.1
INSERT(3)												95.2
INSERT(4)												95.3
INSERT(5)												95.4
INSERT(9)												95.5

Problem 96 (4 %) check

0	1	2	3	4	5	6	7	8	9	10
22		11				3		4	21	

In the above hash table of size 11, *quadratic probing* is used with hash functions $h'(k) = 2k \bmod 11$ and $h(k,i) = (h'(k) + i + i^2) \bmod 11$. Indicate the positions the five elements 1, 2, 5, 7 and 10 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 3, 4, 11, 21 and 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(1)												96.1
INSERT(2)												96.2
INSERT(5)												96.3
INSERT(7)												96.4
INSERT(10)												96.5

Problem 97 (4 %) check

0	1	2	3	4	5	6	7	8	9	10
3	17					14	20		21	

In the above hash table of size 11, *quadratic probing* is used with hash functions $h'(k) = 2k \bmod 11$ and $h(k,i) = (h'(k) + 3i + 2i^2) \bmod 11$. Indicate the positions the five elements 2, 4, 7, 9 and 10 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 3, 14, 17, 20 and 21).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(2)												97.1
INSERT(4)												97.2
INSERT(7)												97.3
INSERT(9)												97.4
INSERT(10)												97.5

Problem 98 (4 %) check

0	1	2	3	4	5	6	7	8	9	10
22		18					19	6	11	

In the above hash table of size 11, *quadratic probing* is used with hash functions $h'(k) = 5k \bmod 11$ and $h(k,i) = (h'(k) + 5i + 2i^2) \bmod 11$. Indicate the positions the five elements 0, 1, 2, 3 and 8 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 6, 11, 18, 19 and 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(0)												98.1
INSERT(1)												98.2
INSERT(2)												98.3
INSERT(3)												98.4
INSERT(8)												98.5

Problem 99 (4 %) check

0	1	2	3	4	5	6	7	8	9	10
11			18	13			20	22		

In the above hash table of size 11, *quadratic probing* is used with hash functions $h'(k) = 2k \bmod 11$ and $h(k,i) = (h'(k) + 3i + 5i^2) \bmod 11$. Indicate the positions the five elements 2, 5, 6, 7 and 9 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 11, 13, 18, 20 and 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(2)												99.1
INSERT(5)												99.2
INSERT(6)												99.3
INSERT(7)												99.4
INSERT(9)												99.5

Problem 100 (4 %) **check**

0	1	2	3	4	5	6	7	8	9	10
0	22			13	8			15		

In the above hash table of size 11, *quadratic probing* is used with hash functions $h'(k) = 2k \bmod 11$ and $h(k,i) = (h'(k) + i^2) \bmod 11$. Indicate the positions the five elements 2, 4, 6, 9 and 11 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 0, 8, 13, 15 and 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(2)												100.1
INSERT(4)												100.2
INSERT(6)												100.3
INSERT(9)												100.4
INSERT(11)												100.5

Double hashing

Problem 101 (4 %) **check**

0	1	2	3	4	5	6	7	8	9	10
		19	12		14		6		3	

In the above hash table of size 11, *double hashing* is used with hash functions $h_1(k) = 3k \bmod 11$ and $h_2(k) = 1 + (4k \bmod 10)$. Indicate the positions the five elements 2, 5, 8, 9 and 11 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 3, 6, 12, 14 and 19).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(2)												101.1
INSERT(5)												101.2
INSERT(8)												101.3
INSERT(9)												101.4
INSERT(11)												101.5

Problem 102 (4 %) **check**

0	1	2	3	4	5	6	7	8	9	10
		19	7		8		9	18		

In the above hash table of size 11, *double hashing* is used with hash functions $h_1(k) = 2k \bmod 11$ and $h_2(k) = 1 + (3k \bmod 10)$. Indicate the positions the five elements 1, 2, 4, 6 and 11 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 7, 8, 9, 18 and 19).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(1)												102.1
INSERT(2)												102.2
INSERT(4)												102.3
INSERT(6)												102.4
INSERT(11)												102.5

Problem 103 (4 %) **check**

0	1	2	3	4	5	6	7	8	9	10
11	9			13			22			16

In the above hash table of size 11, *double hashing* is used with hash functions $h_1(k) = 2k \bmod 11$ and $h_2(k) = 1 + (3k \bmod 10)$. Indicate the positions the five elements 0, 4, 5, 7 and 8 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 9, 11, 13, 16 and 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(0)												103.1
INSERT(4)												103.2
INSERT(5)												103.3
INSERT(7)												103.4
INSERT(8)												103.5

Problem 104 (4 %) **check**

0	1	2	3	4	5	6	7	8	9	10
10							21	19	16	8

In the above hash table of size 11, *double hashing* is used with hash functions $h_1(k) = 4k \bmod 11$ and $h_2(k) = 1 + (2k \bmod 10)$. Indicate the positions the five elements 0, 1, 2, 3 and 4 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 8, 10, 16, 19 and 21).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(0)												104.1
INSERT(1)												104.2
INSERT(2)												104.3
INSERT(3)												104.4
INSERT(4)												104.5

Problem 105 (4 %) **check**

0	1	2	3	4	5	6	7	8	9	10
22	0	6							5	19

In the above hash table of size 11, *double hashing* is used with hash functions $h_1(k) = 4k \bmod 11$ and $h_2(k) = 1 + (3k \bmod 10)$. Indicate the positions the five elements 2, 3, 4, 8 and 9 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 0, 5, 6, 19 and 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(2)												105.1
INSERT(3)												105.2
INSERT(4)												105.3
INSERT(8)												105.4
INSERT(9)												105.5

Problem 106 (4 %) check

0	1	2	3	4	5	6	7	8	9	10
11				1	7	18		2		

In the above hash table of size 11, *double hashing* is used with hash functions $h_1(k) = 4k \bmod 11$ and $h_2(k) = 1 + (2k \bmod 10)$. Indicate the positions the five elements 0, 3, 4, 5 and 9 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 1, 2, 7, 11 and 18).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(0)												106.1
INSERT(3)												106.2
INSERT(4)												106.3
INSERT(5)												106.4
INSERT(9)												106.5

Problem 107 (4 %) check

0	1	2	3	4	5	6	7	8	9	10
11	14			1		7				3

In the above hash table of size 11, *double hashing* is used with hash functions $h_1(k) = 4k \bmod 11$ and $h_2(k) = 1 + (3k \bmod 10)$. Indicate the positions the five elements 0, 4, 6, 8 and 10 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 1, 3, 7, 11 and 14).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(0)												107.1
INSERT(4)												107.2
INSERT(6)												107.3
INSERT(8)												107.4
INSERT(10)												107.5

Problem 108 (4 %) **check**

0	1	2	3	4	5	6	7	8	9	10
22			5		12		8			11

In the above hash table of size 11, *double hashing* is used with hash functions $h_1(k) = 5k \bmod 11$ and $h_2(k) = 1 + (4k \bmod 10)$. Indicate the positions the five elements 0, 1, 2, 6 and 9 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 5, 8, 11, 12 and 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(0)												108.1
INSERT(1)												108.2
INSERT(2)												108.3
INSERT(6)												108.4
INSERT(9)												108.5

Problem 109 (4 %) **check**

0	1	2	3	4	5	6	7	8	9	10
22		1			12				21	10

In the above hash table of size 11, *double hashing* is used with hash functions $h_1(k) = 2k \bmod 11$ and $h_2(k) = 1 + (3k \bmod 10)$. Indicate the positions the five elements 3, 4, 5, 6 and 11 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 1, 10, 12, 21 and 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(3)												109.1
INSERT(4)												109.2
INSERT(5)												109.3
INSERT(6)												109.4
INSERT(11)												109.5

Problem 110 (4 %) check

0	1	2	3	4	5	6	7	8	9	10
0		1	18		22		12			

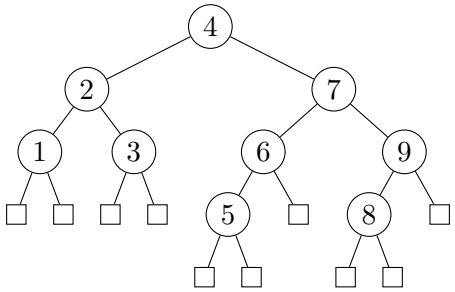
In the above hash table of size 11, *double hashing* is used with hash functions $h_1(k) = 2k \bmod 11$ and $h_2(k) = 1 + (2k \bmod 10)$. Indicate the positions the five elements 3, 7, 8, 9 and 10 will be inserted at in the hash table (for each insertion, we assume the hash table only contains the elements 0, 1, 12, 18 and 22).

	0	1	2	3	4	5	6	7	8	9	10	
INSERT(3)												110.1
INSERT(7)												110.2
INSERT(8)												110.3
INSERT(9)												110.4
INSERT(10)												110.5

Valid red-black trees

Problem 111 (4 %) check

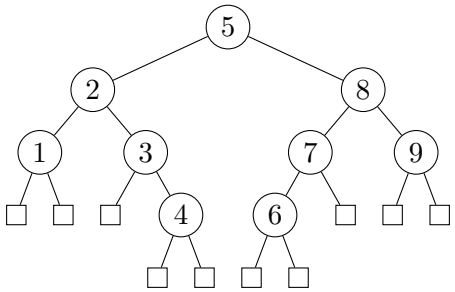
For each of the following subsets, indicate whether the binary tree below is a valid red-black tree if these nodes are colored red.



	Yes	No	
1, 3, 5, 7, 8			111.1
5, 8			111.2
4, 5, 8			111.3
2, 5, 6, 8, 9			111.4
2, 5, 7, 8			111.5

Problem 112 (4 %) check

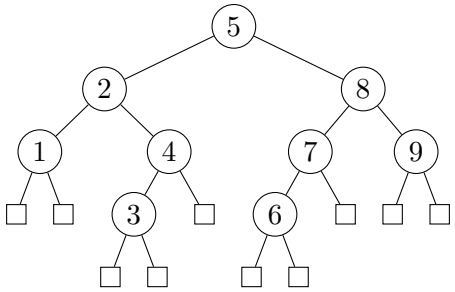
For each of the following subsets, indicate whether the binary tree below is a valid red-black tree if these nodes are colored red.



	Yes	No	
2, 4, 6, 7, 9			112.1
1, 3, 4, 6, 8			112.2
4, 5, 6			112.3
2, 4, 6, 8			112.4
4, 6			112.5

Problem 113 (4 %) check

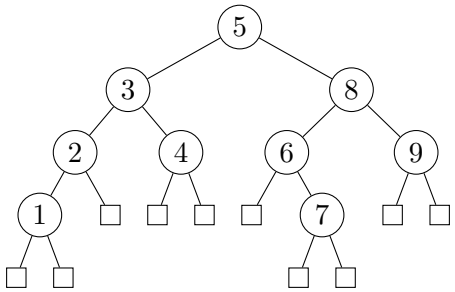
For each of the following subsets, indicate whether the binary tree below is a valid red-black tree if these nodes are colored red.



	Yes	No	
3, 5, 6			113.1
2, 3, 6, 7, 9			113.2
2, 3, 6, 8			113.3
1, 3, 4, 6, 8			113.4
3, 6			113.5

Problem 114 (4 %) check

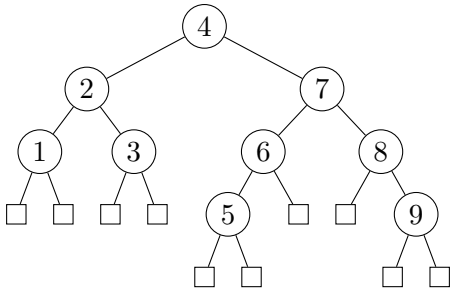
For each of the following subsets, indicate whether the binary tree below is a valid red-black tree if these nodes are colored red.



	Yes	No	
1, 7			114.1
1, 5, 7			114.2
1, 3, 7, 8			114.3
1, 2, 4, 7, 8			114.4
1, 3, 6, 7, 9			114.5

Problem 115 (4 %) check

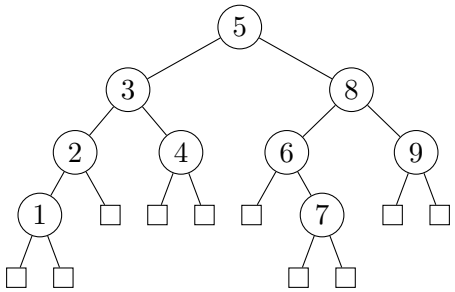
For each of the following subsets, indicate whether the binary tree below is a valid red-black tree if these nodes are colored red.



	Yes	No	
4, 5, 9			115.1
1, 3, 5, 6, 8, 9			115.2
1, 3, 5, 7, 9			115.3
5, 9			115.4
2, 5, 7, 9			115.5

Problem 116 (4 %) check

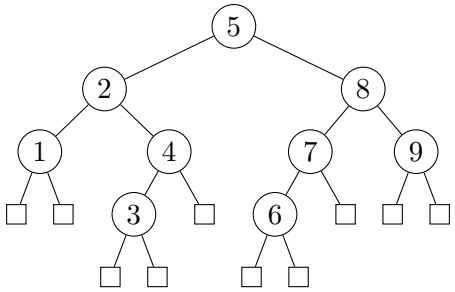
For each of the following subsets, indicate whether the binary tree below is a valid red-black tree if these nodes are colored red.



	Yes	No	
1, 5, 7			116.1
1, 3, 7, 8			116.2
1, 2, 4, 7, 8			116.3
1, 7			116.4
1, 3, 6, 7, 9			116.5

Problem 117 (4 %) check

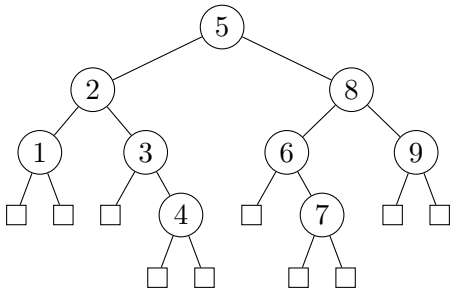
For each of the following subsets, indicate whether the binary tree below is a valid red-black tree if these nodes are colored red.



	Yes	No	
3, 6			117.1
1, 3, 4, 6, 8			117.2
3, 5, 6			117.3
2, 3, 6, 8			117.4
2, 3, 6, 7, 9			117.5

Problem 118 (4 %) check

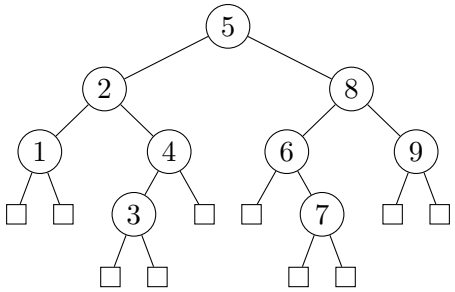
For each of the following subsets, indicate whether the binary tree below is a valid red-black tree if these nodes are colored red.



	Yes	No	
4, 7			118.1
1, 3, 4, 7, 8			118.2
2, 4, 7, 8			118.3
2, 4, 6, 7, 9			118.4
4, 5, 7			118.5

Problem 119 (4 %) check

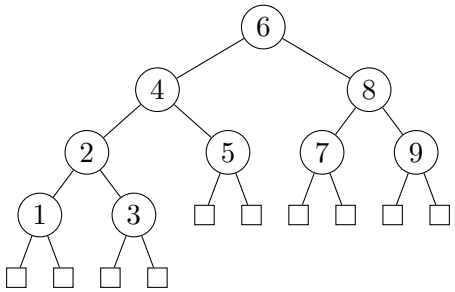
For each of the following subsets, indicate whether the binary tree below is a valid red-black tree if these nodes are colored red.



	Yes	No	
3, 7			119.1
1, 3, 4, 7, 8			119.2
3, 5, 7			119.3
2, 3, 7, 8			119.4
2, 3, 6, 7, 9			119.5

Problem 120 (4 %) check

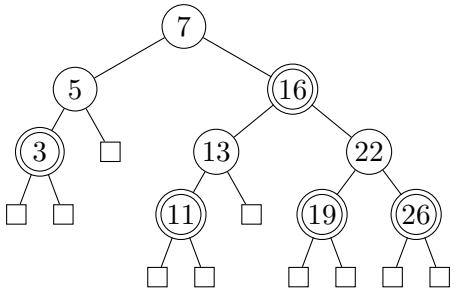
For each of the following subsets, indicate whether the binary tree below is a valid red-black tree if these nodes are colored red.



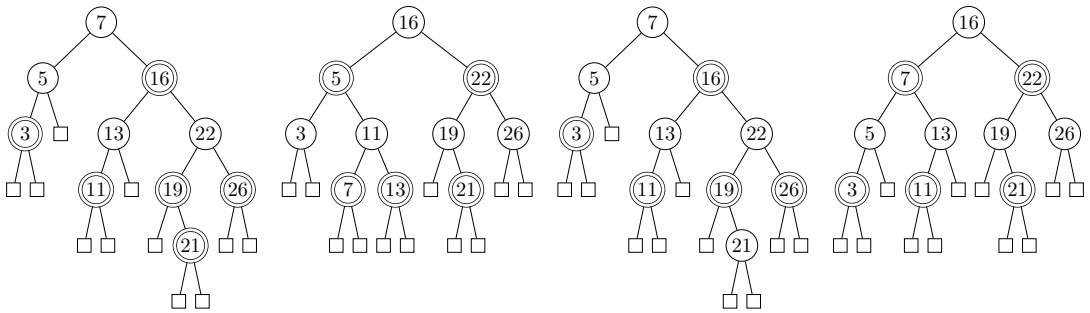
	Yes	No	
1, 3, 4, 7, 9			120.1
1, 3, 6			120.2
1, 2, 3, 5, 7, 9			120.3
1, 3, 4, 8			120.4
1, 3			120.5

Red-black tree insertion

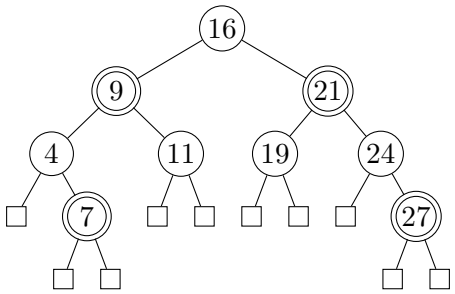
Problem 121 (4 %) check



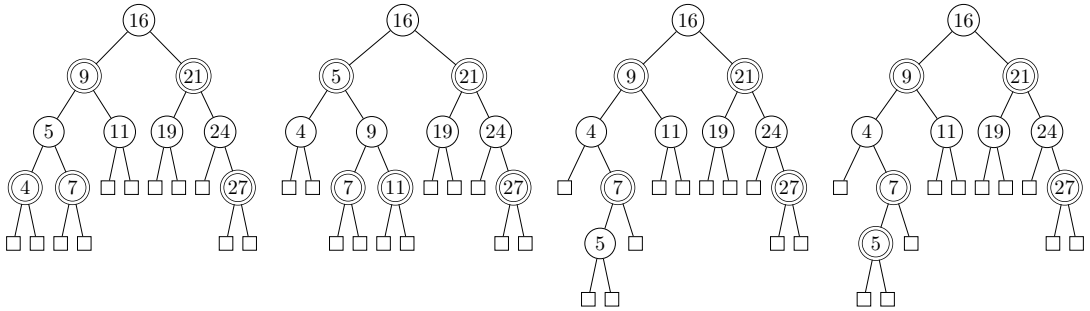
Indicate the resulting red-black tree when inserting 21 into the above red-black tree (double circles indicate red nodes).



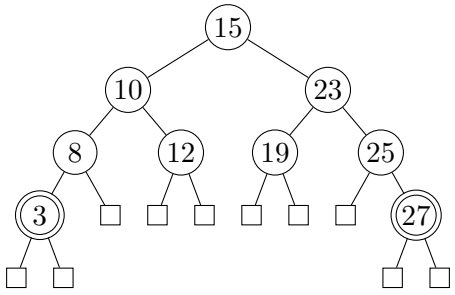
Problem 122 (4 %) check



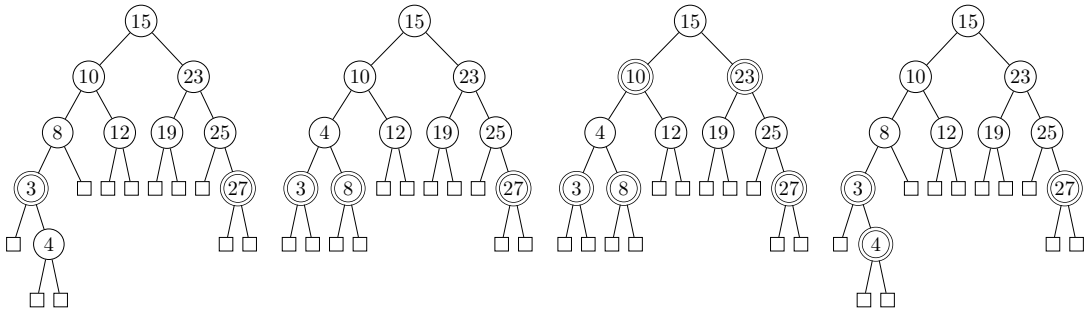
Indicate the resulting red-black tree when inserting 5 into the above red-black tree (double circles indicate red nodes).



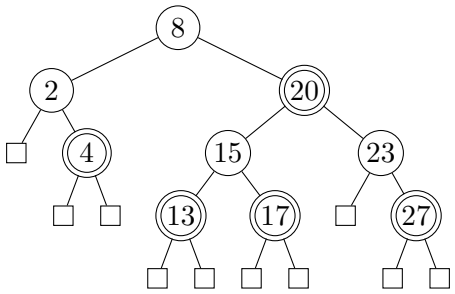
Problem 123 (4 %) check



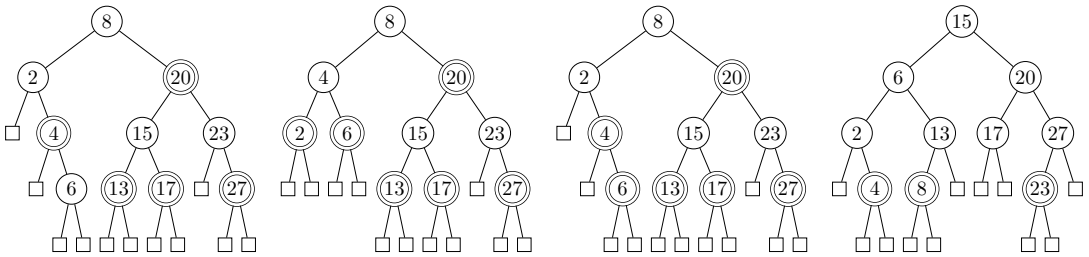
Indicate the resulting red-black tree when inserting 4 into the above red-black tree (double circles indicate red nodes).



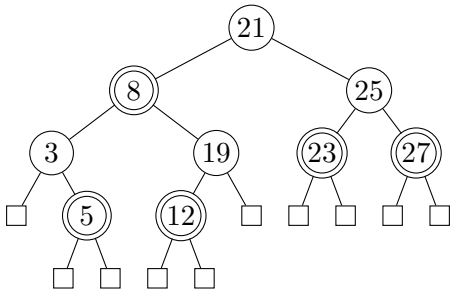
Problem 124 (4 %) check



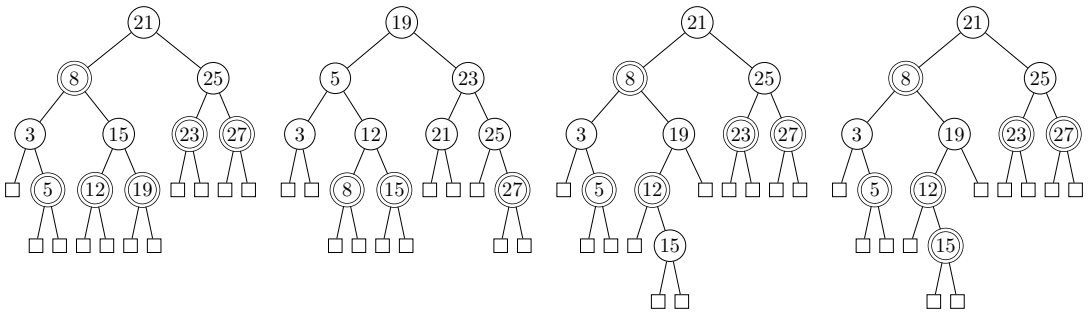
Indicate the resulting red-black tree when inserting 6 into the above red-black tree (double circles indicate red nodes).



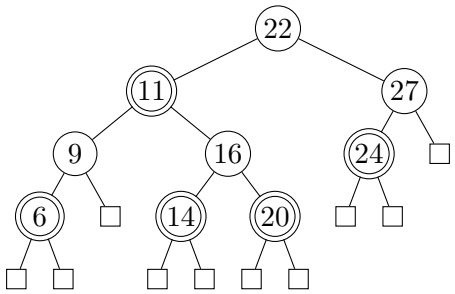
Problem 125 (4 %) check



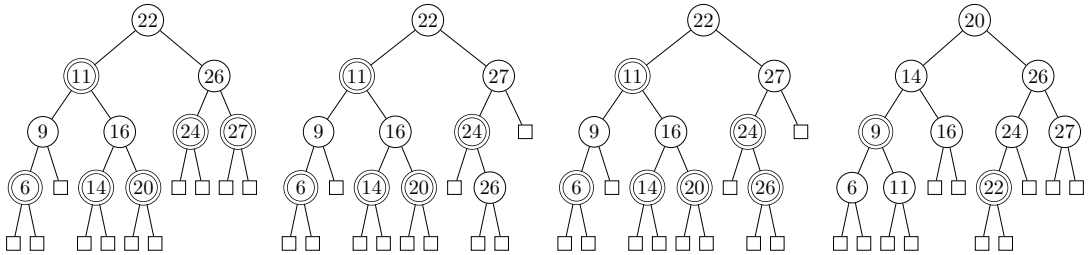
Indicate the resulting red-black tree when inserting 15 into the above red-black tree (double circles indicate red nodes).



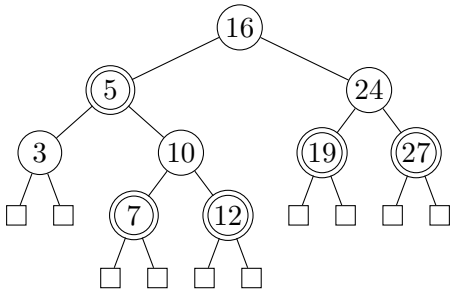
Problem 126 (4 %) check



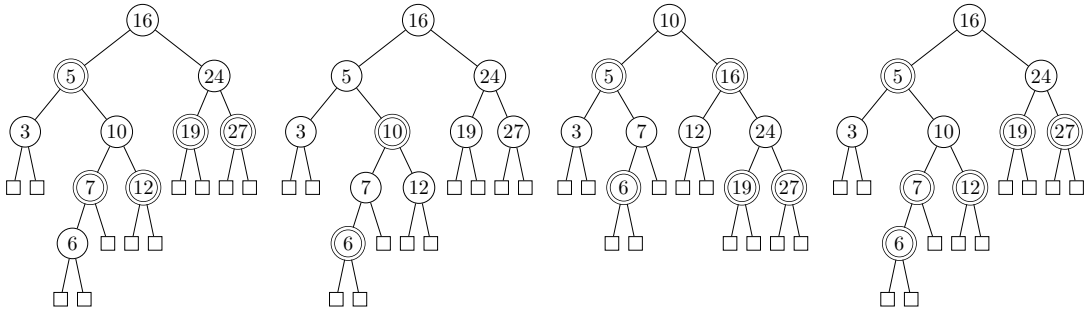
Indicate the resulting red-black tree when inserting 26 into the above red-black tree (double circles indicate red nodes).



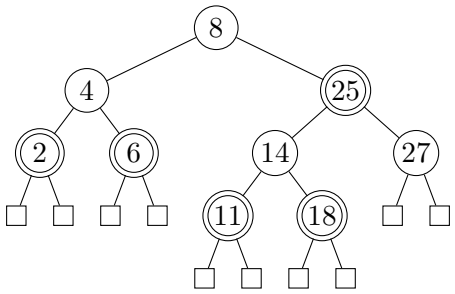
Problem 127 (4 %) check



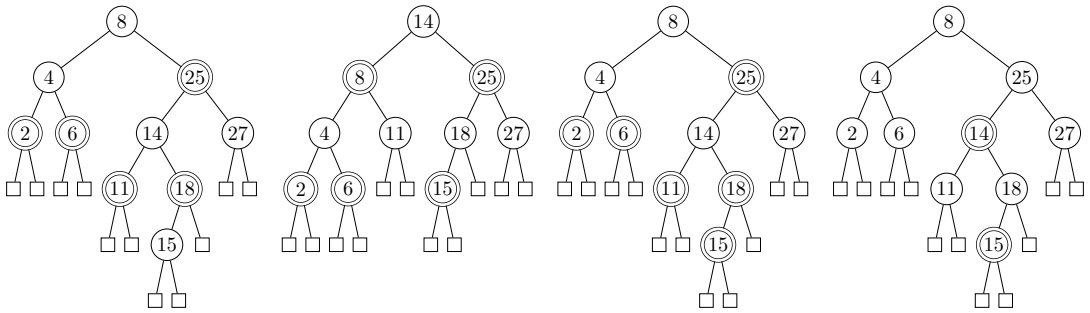
Indicate the resulting red-black tree when inserting 6 into the above red-black tree (double circles indicate red nodes).



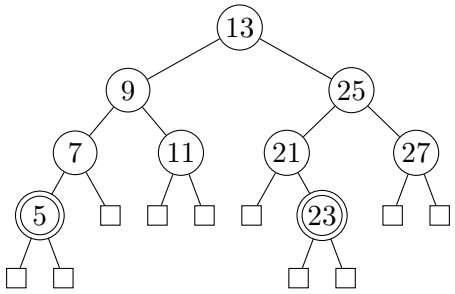
Problem 128 (4 %) check



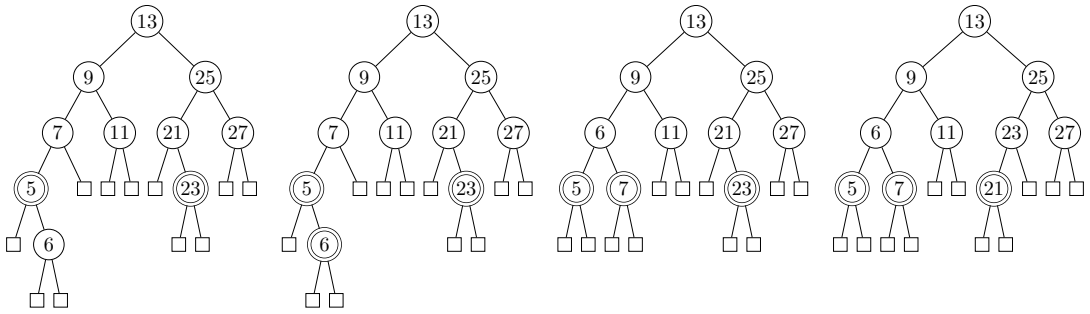
Indicate the resulting red-black tree when inserting 15 into the above red-black tree (double circles indicate red nodes).



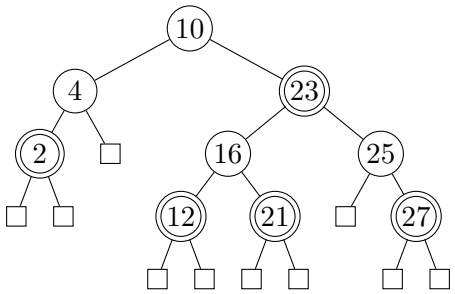
Problem 129 (4 %) check



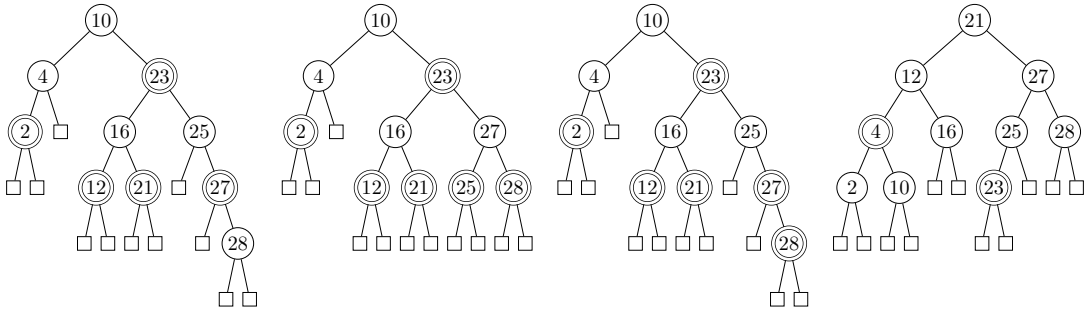
Indicate the resulting red-black tree when inserting 6 into the above red-black tree (double circles indicate red nodes).



Problem 130 (4 %) **check**



Indicate the resulting red-black tree when inserting 28 into the above red-black tree (double circles indicate red nodes).

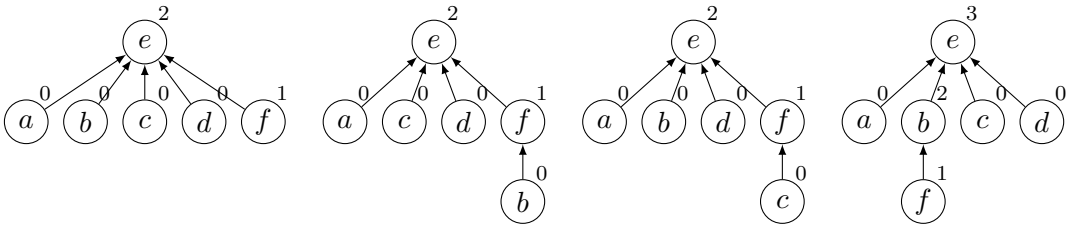


Union-find

Problem 131 (4 %) check

Indicate the resulting union-find structure after the following sequence of operations, when using union-by-rank and path compression.

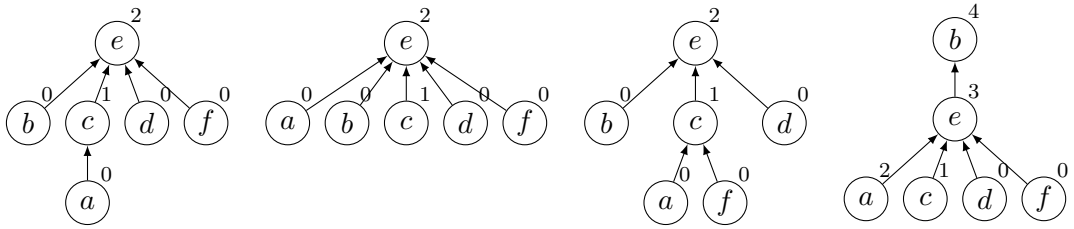
MAKESET(a)
MAKESET(b)
MAKESET(c)
MAKESET(d)
MAKESET(e)
MAKESET(f)
UNION(c, f)
UNION(c, b)
UNION(d, e)
UNION(c, e)
UNION(a, c)
FIND-SET(b)



Problem 132 (4 %) **check**

Indicate the resulting union-find structure after the following sequence of operations, when using union-by-rank and path compression.

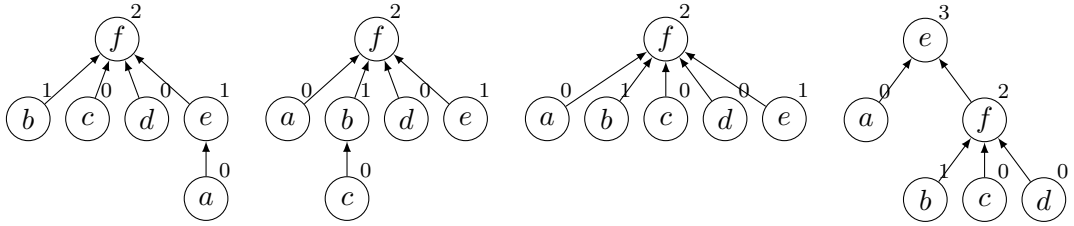
MAKESET(a)
MAKESET(b)
MAKESET(c)
MAKESET(d)
MAKESET(e)
MAKESET(f)
UNION(f, c)
UNION(c, a)
UNION(d, e)
UNION(a, e)
UNION(f, b)
FIND-SET(b)



Problem 133 (4 %) **check**

Indicate the resulting union-find structure after the following sequence of operations, when using union-by-rank and path compression.

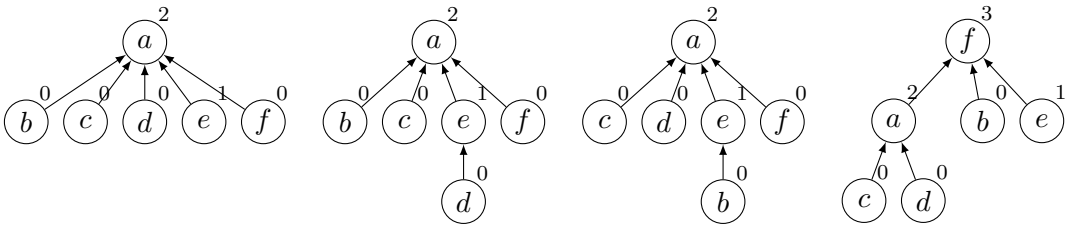
MAKESET(a)
MAKESET(b)
MAKESET(c)
MAKESET(d)
MAKESET(e)
MAKESET(f)
UNION(c, b)
UNION(d, f)
UNION(c, d)
UNION(a, e)
UNION(c, e)
FIND-SET(a)



Problem 134 (4 %) **check**

Indicate the resulting union-find structure after the following sequence of operations, when using union-by-rank and path compression.

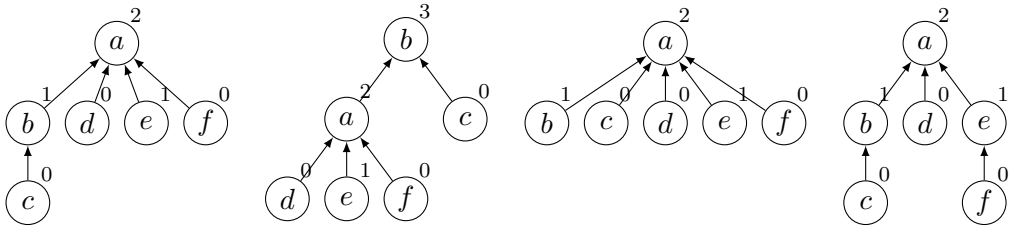
MAKESET(a)
MAKESET(b)
MAKESET(c)
MAKESET(d)
MAKESET(e)
MAKESET(f)
UNION(d, e)
UNION(b, e)
UNION(c, a)
UNION(e, a)
UNION(d, f)
FIND-SET(b)



Problem 135 (4 %) **check**

Indicate the resulting union-find structure after the following sequence of operations, when using union-by-rank and path compression.

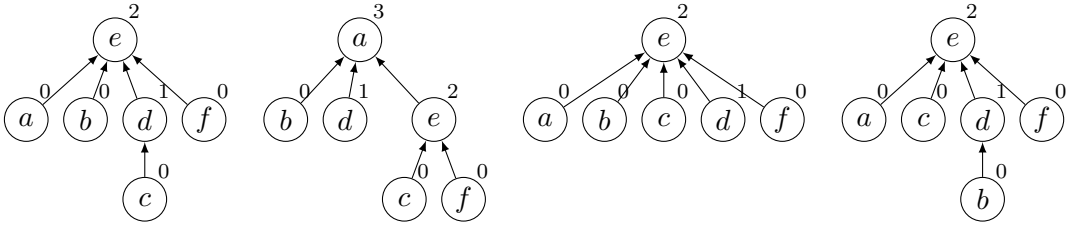
MAKESET(a)
MAKESET(b)
MAKESET(c)
MAKESET(d)
MAKESET(e)
MAKESET(f)
UNION(f, e)
UNION(d, a)
UNION(f, d)
UNION(c, b)
UNION(f, c)
FIND-SET(b)



Problem 136 (4 %) **check**

Indicate the resulting union-find structure after the following sequence of operations, when using union-by-rank and path compression.

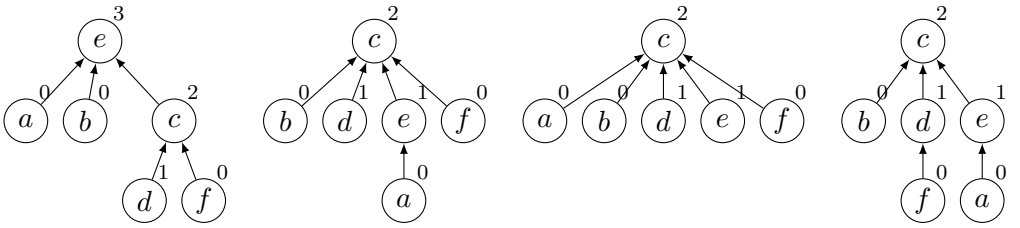
MAKESET(a)
MAKESET(b)
MAKESET(c)
MAKESET(d)
MAKESET(e)
MAKESET(f)
UNION(b, d)
UNION(c, b)
UNION(f, e)
UNION(c, f)
UNION(c, a)
FIND-SET(b)



Problem 137 (4 %) **check**

Indicate the resulting union-find structure after the following sequence of operations, when using union-by-rank and path compression.

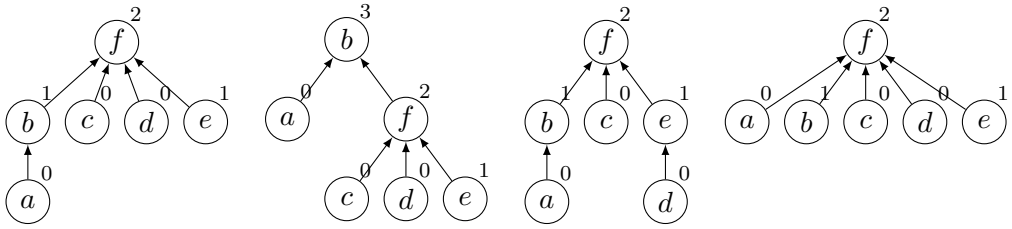
MAKESET(a)
MAKESET(b)
MAKESET(c)
MAKESET(d)
MAKESET(e)
MAKESET(f)
UNION(f, d)
UNION(b, c)
UNION(f, c)
UNION(a, e)
UNION(f, a)
FIND-SET(b)



Problem 138 (4 %) **check**

Indicate the resulting union-find structure after the following sequence of operations, when using union-by-rank and path compression.

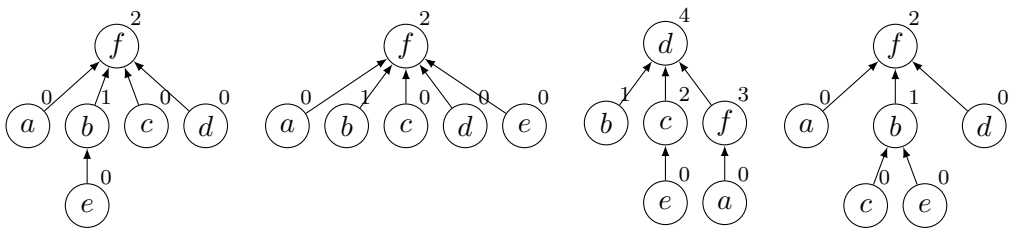
MAKESET(a)
MAKESET(b)
MAKESET(c)
MAKESET(d)
MAKESET(e)
MAKESET(f)
UNION(d, e)
UNION(c, f)
UNION(e, f)
UNION(a, b)
UNION(d, a)
FIND-SET(b)



Problem 139 (4 %) **check**

Indicate the resulting union-find structure after the following sequence of operations, when using union-by-rank and path compression.

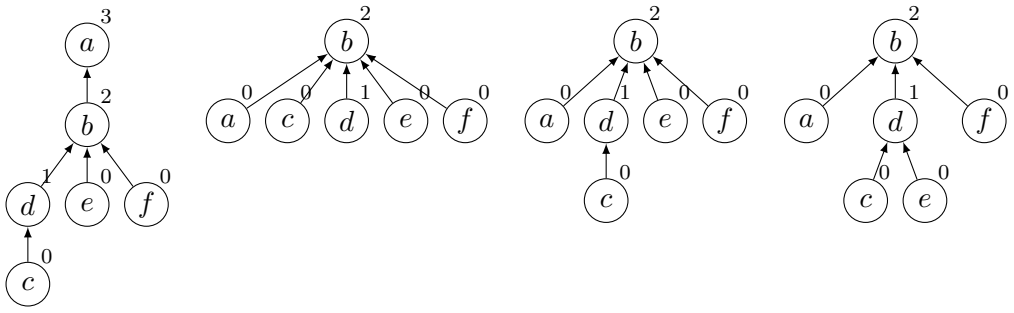
MAKESET(a)
MAKESET(b)
MAKESET(c)
MAKESET(d)
MAKESET(e)
MAKESET(f)
UNION(e, b)
UNION(b, c)
UNION(a, f)
UNION(e, a)
UNION(c, d)
FIND-SET(b)



Problem 140 (4 %) **check**

Indicate the resulting union-find structure after the following sequence of operations, when using union-by-rank and path compression.

MAKESET(a)
MAKESET(b)
MAKESET(c)
MAKESET(d)
MAKESET(e)
MAKESET(f)
UNION(c, d)
UNION(e, d)
UNION(f, b)
UNION(c, b)
UNION(e, a)
FIND-SET(b)



Huffman encoding

Problem 141 (4 %) **check**

Letter	a	b	c	d	e	f	g
Frequency	20	80	10	40	70	40	30

Assume we have a file with the above letters and frequencies. If one constructs a Huffman tree for these frequencies, what are the lengths of the codes for `a`, `d` and `f`?

1 2 3 4 5 6

`a` 141.1

`d` 141.2

`f` 141.3

The file contains 20 + 80 + 10 + 40 + 70 + 40 + 30 = 290 letters. How many bits would a Huffman encoding of the string use?

750 790 800 810 830 840

141.4

Problem 142 (4 %) **check**

Letter	a	b	c	d	e	f	g
Frequency	60	30	50	80	80	40	30

Assume we have a file with the above letters and frequencies. If one constructs a Huffman tree for these frequencies, what are the lengths of the codes for `b`, `d` and `g`?

1 2 3 4 5 6

`b` 142.1

`d` 142.2

`g` 142.3

The file contains 60 + 30 + 50 + 80 + 80 + 40 + 30 = 370 letters. How many bits would a Huffman encoding of the string use?

1010 1020 1030 1040 1050 1060

142.4

Problem 143 (4 %) check

Letter	a	b	c	d	e	f	g
Frequency	40	10	30	20	40	70	80

Assume we have a file with the above letters and frequencies. If one constructs a Huffman tree for these frequencies, what are the lengths of the codes for `c`, `d` and `e`?

	1	2	3	4	5	6	
`c`							143.1
`d`							143.2
`e`							143.3

The file contains 40 + 10 + 30 + 20 + 40 + 70 + 80 = 290 letters. How many bits would a Huffman encoding of the string use?

730 740 750 780 790 800

143.4

Problem 144 (4 %) check

Letter	a	b	c	d	e	f	g
Frequency	40	20	70	60	80	50	10

Assume we have a file with the above letters and frequencies. If one constructs a Huffman tree for these frequencies, what are the lengths of the codes for `d`, `e` and `g`?

	1	2	3	4	5	6	
`d`							144.1
`e`							144.2
`g`							144.3

The file contains 40 + 20 + 70 + 60 + 80 + 50 + 10 = 330 letters. How many bits would a Huffman encoding of the string use?

830 850 860 870 900 910

144.4

Problem 145 (4 %) **check**

Letter	a	b	c	d	e	f	g
Frequency	70	10	50	50	10	30	20

Assume we have a file with the above letters and frequencies. If one constructs a Huffman tree for these frequencies, what are the lengths of the codes for `a`, `c` and `f`?

	1	2	3	4	5	6	
`a`							145.1
`c`							145.2
`f`							145.3

The file contains 70 + 10 + 50 + 50 + 10 + 30 + 20 = 240 letters. How many bits would a Huffman encoding of the string use?

590 600 610 620 630 640

145.4

Problem 146 (4 %) **check**

Letter	a	b	c	d	e	f	g
Frequency	40	80	60	20	10	40	30

Assume we have a file with the above letters and frequencies. If one constructs a Huffman tree for these frequencies, what are the lengths of the codes for `c`, `e` and `g`?

	1	2	3	4	5	6	
`c`							146.1
`e`							146.2
`g`							146.3

The file contains 40 + 80 + 60 + 20 + 10 + 40 + 30 = 280 letters. How many bits would a Huffman encoding of the string use?

720 730 760 770 780 790

146.4

Problem 147 (4 %) **check**

Letter	a	b	c	d	e	f	g
Frequency	70	40	50	30	20	20	30

Assume we have a file with the above letters and frequencies. If one constructs a Huffman tree for these frequencies, what are the lengths of the codes for `c`, `d` and `f`?

	1	2	3	4	5	6	
`c`							147.1
`d`							147.2
`f`							147.3

The file contains 70 + 40 + 50 + 30 + 20 + 20 + 30 = 260 letters. How many bits would a Huffman encoding of the string use?

660 670 680 700 710 720

147.4

Problem 148 (4 %) **check**

Letter	a	b	c	d	e	f	g
Frequency	80	50	50	60	10	40	20

Assume we have a file with the above letters and frequencies. If one constructs a Huffman tree for these frequencies, what are the lengths of the codes for `b`, `c` and `g`?

	1	2	3	4	5	6	
`b`							148.1
`c`							148.2
`g`							148.3

The file contains 80 + 50 + 50 + 60 + 10 + 40 + 20 = 310 letters. How many bits would a Huffman encoding of the string use?

780 800 810 820 830 850

148.4

Problem 149 (4 %) **check**

Letter	a	b	c	d	e	f	g
Frequency	50	20	70	30	30	40	20

Assume we have a file with the above letters and frequencies. If one constructs a Huffman tree for these frequencies, what are the lengths of the codes for `b`, `d` and `e`?

	1	2	3	4	5	6	
`b`							149.1
`d`							149.2
`e`							149.3

The file contains 50 + 20 + 70 + 30 + 30 + 40 + 20 = 260 letters. How many bits would a Huffman encoding of the string use?

700 710 720 730 740 750

149.4

Problem 150 (4 %) **check**

Letter	a	b	c	d	e	f	g
Frequency	40	60	60	10	50	50	20

Assume we have a file with the above letters and frequencies. If one constructs a Huffman tree for these frequencies, what are the lengths of the codes for `b`, `e` and `f`?

	1	2	3	4	5	6	
`b`							150.1
`e`							150.2
`f`							150.3

The file contains 40 + 60 + 60 + 10 + 50 + 50 + 20 = 290 letters. How many bits would a Huffman encoding of the string use?

720 730 740 760 770 780

150.4

Recurrence relations

Problem 151 (4 %) check

State the solution for each of the below recurrence relations, where $T(n) = 1$ for $n \leq 1$.

$$\Theta(\log n) \quad \Theta(\sqrt{n}) \quad \Theta(n) \quad \Theta(n \log n) \quad \Theta(n^2) \quad \Theta(n^2 \log n) \quad \Theta(n^3)$$

$T(n) = T(n-1) + \log n$ 151.1

$T(n) = 4 \cdot T(n/5) + n^3$ 151.2

$T(n) = 4 \cdot T(n/2) + n^2$ 151.3

$T(n) = 3 \cdot T(n/5) + n$ 151.4

$T(n) = T(n-1) + n^2$ 151.5

Problem 152 (4 %) check

State the solution for each of the below recurrence relations, where $T(n) = 1$ for $n \leq 1$.

$$\Theta(\log n) \quad \Theta(\sqrt{n}) \quad \Theta(n) \quad \Theta(n \log n) \quad \Theta(n^2) \quad \Theta(n^2 \log n) \quad \Theta(n^3)$$

$T(n) = T(n-1) + \log n$ 152.1

$T(n) = 4 \cdot T(n/5) + n$ 152.2

$T(n) = T(n/3) + 3$ 152.3

$T(n) = 2 \cdot T(n/2) + n$ 152.4

$T(n) = 3 \cdot T(n/9) + 3$ 152.5

Problem 153 (4 %) check

State the solution for each of the below recurrence relations, where $T(n) = 1$ for $n \leq 1$.

$$\Theta(\log n) \quad \Theta(\sqrt{n}) \quad \Theta(n) \quad \Theta(n \log n) \quad \Theta(n^2) \quad \Theta(n^2 \log n) \quad \Theta(n^3)$$

$T(n) = 2 \cdot T(n/4) + 2$ 153.1

$T(n) = 5 \cdot T(n/5) + n$ 153.2

$T(n) = 3 \cdot T(n/9) + 1$ 153.3

$T(n) = 8 \cdot T(n/2) + 2$ 153.4

$T(n) = 4 \cdot T(n/2) + n^2$ 153.5

Problem 154 (4 %) check

State the solution for each of the below recurrence relations, where $T(n) = 1$ for $n \leq 1$.

$$\Theta(\log n) \quad \Theta(\sqrt{n}) \quad \Theta(n) \quad \Theta(n \log n) \quad \Theta(n^2) \quad \Theta(n^2 \log n) \quad \Theta(n^3)$$

$T(n) = 2 \cdot T(n/5) + n$ 154.1

$T(n) = 4 \cdot T(n/2) + 1$ 154.2

$T(n) = 4 \cdot T(n/4) + n$ 154.3

$T(n) = T(n/4) + 3$ 154.4

$T(n) = T(n-1) + n$ 154.5

Problem 155 (4 %) check

State the solution for each of the below recurrence relations, where $T(n) = 1$ for $n \leq 1$.

$$\Theta(\log n) \quad \Theta(\sqrt{n}) \quad \Theta(n) \quad \Theta(n \log n) \quad \Theta(n^2) \quad \Theta(n^2 \log n) \quad \Theta(n^3)$$

$T(n) = 2 \cdot T(n/4) + 1$ 155.1

$T(n) = 5 \cdot T(n/5) + n$ 155.2

$T(n) = 8 \cdot T(n/2) + 1$ 155.3

$T(n) = T(n-1) + 2$ 155.4

$T(n) = T(n-1) + n$ 155.5

Problem 156 (4 %) check

State the solution for each of the below recurrence relations, where $T(n) = 1$ for $n \leq 1$.

$$\Theta(\log n) \quad \Theta(\sqrt{n}) \quad \Theta(n) \quad \Theta(n \log n) \quad \Theta(n^2) \quad \Theta(n^2 \log n) \quad \Theta(n^3)$$

$T(n) = 4 \cdot T(n/2) + n^2$ 156.1

$T(n) = 9 \cdot T(n/3) + 3$ 156.2

$T(n) = 4 \cdot T(n/2) + 2$ 156.3

$T(n) = 2 \cdot T(n/4) + 2$ 156.4

$T(n) = 8 \cdot T(n/2) + 1$ 156.5

Problem 157 (4 %) check

State the solution for each of the below recurrence relations, where $T(n) = 1$ for $n \leq 1$.

$$\Theta(\log n) \quad \Theta(\sqrt{n}) \quad \Theta(n) \quad \Theta(n \log n) \quad \Theta(n^2) \quad \Theta(n^2 \log n) \quad \Theta(n^3)$$

$T(n) = 8 \cdot T(n/2) + 2$ 157.1

$T(n) = T(n/3) + 5$ 157.2

$T(n) = T(n-1) + n$ 157.3

$T(n) = 2 \cdot T(n/4) + n$ 157.4

$T(n) = 3 \cdot T(n/4) + n^3$ 157.5

Problem 158 (4 %) check

State the solution for each of the below recurrence relations, where $T(n) = 1$ for $n \leq 1$.

$$\Theta(\log n) \quad \Theta(\sqrt{n}) \quad \Theta(n) \quad \Theta(n \log n) \quad \Theta(n^2) \quad \Theta(n^2 \log n) \quad \Theta(n^3)$$

$T(n) = 9 \cdot T(n/3) + n^2$ 158.1

$T(n) = 4 \cdot T(n/2) + 2$ 158.2

$T(n) = T(n/4) + 3$ 158.3

$T(n) = 3 \cdot T(n/9) + 1$ 158.4

$T(n) = 3 \cdot T(n/5) + n$ 158.5

Problem 159 (4 %) check

State the solution for each of the below recurrence relations, where $T(n) = 1$ for $n \leq 1$.

$\Theta(\log n)$ $\Theta(\sqrt{n})$ $\Theta(n)$ $\Theta(n \log n)$ $\Theta(n^2)$ $\Theta(n^2 \log n)$ $\Theta(n^3)$

159.1 $T(n) = 3 \cdot T(n/9) + 3$

159.2 $T(n) = T(n-1) + \log n$

159.3 $T(n) = 2 \cdot T(n/4) + 2$

159.4 $T(n) = T(n-1) + 3$

159.5 $T(n) = T(n/3) + 5$

Problem 160 (4 %) check

State the solution for each of the below recurrence relations, where $T(n) = 1$ for $n \leq 1$.

$\Theta(\log n)$ $\Theta(\sqrt{n})$ $\Theta(n)$ $\Theta(n \log n)$ $\Theta(n^2)$ $\Theta(n^2 \log n)$ $\Theta(n^3)$

160.1 $T(n) = 9 \cdot T(n/3) + 3$

160.2 $T(n) = 4 \cdot T(n/5) + n^2$

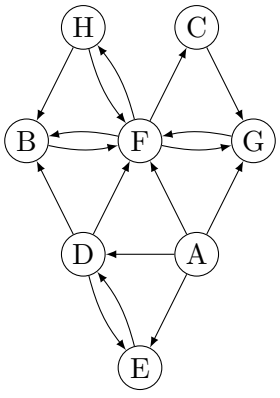
160.3 $T(n) = 4 \cdot T(n/2) + n^2$

160.4 $T(n) = 3 \cdot T(n/9) + 2$

160.5 $T(n) = 2 \cdot T(n/4) + n^3$

BFS

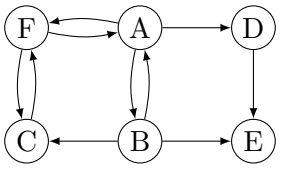
Problem 161 (4 %) check



For a breadth first traversal (BFS) of the graph above **starting in node A**, indicate the order in which the nodes are inserted into the queue Q in the BFS algorithm. It is assumed that the graph is given by alphabetically sorted adjacency lists.

A D E F G B C H A D E F G B H C A G D E F B C H A D B F C G H E

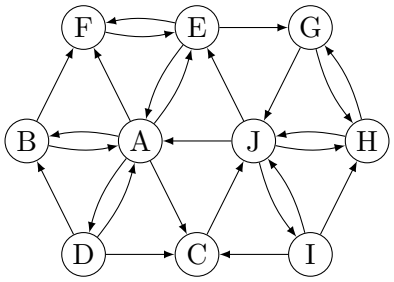
Problem 162 (4 %) check



For a breadth first traversal (BFS) of the graph above **starting in node A**, indicate the order in which the nodes are inserted into the queue Q in the BFS algorithm. It is assumed that the graph is given by alphabetically sorted adjacency lists.

A B C F E D A B D F C E A B D F E C A F D B C E

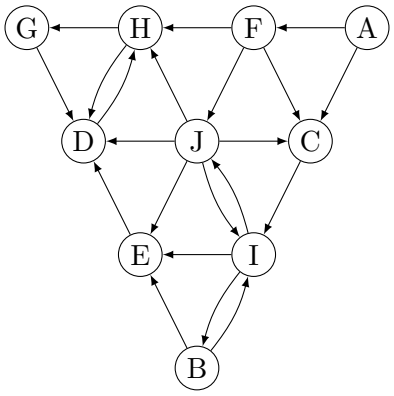
Problem 163 (4 %) check



For a breadth first traversal (BFS) of the graph above **starting in node A**, indicate the order in which the nodes are inserted into the queue Q in the BFS algorithm. It is assumed that the graph is given by alphabetically sorted adjacency lists.

A B F E G H J I C D A B C D E F J G I H A E B F D C G J H I A B C D E F J G H I

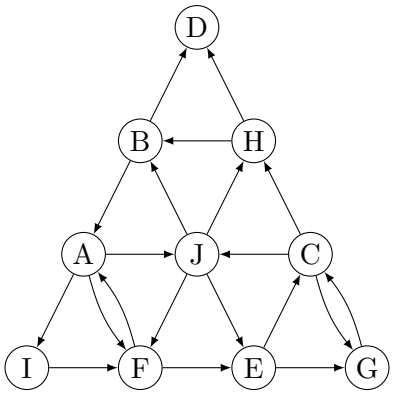
Problem 164 (4 %) check



For a breadth first traversal (BFS) of the graph above **starting in node A**, indicate the order in which the nodes are inserted into the queue Q in the BFS algorithm. It is assumed that the graph is given by alphabetically sorted adjacency lists.

A C I B E D H G J F A C F I H J B E G D A C F I H J E B G D A C F I H J B E D G

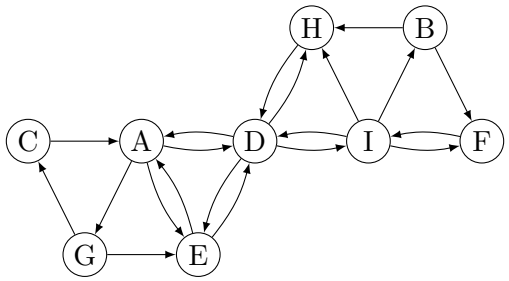
Problem 165 (4 %) check



For a breadth first traversal (BFS) of the graph above **starting in node A**, indicate the order in which the nodes are inserted into the queue Q in the BFS algorithm. It is assumed that the graph is given by alphabetically sorted adjacency lists.

A F I J E B H C G D A J F I H B E D C G A F I J E B H G C D A F E C G H B D J I

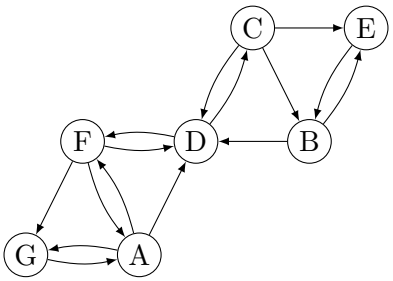
Problem 166 (4 %) check



For a breadth first traversal (BFS) of the graph above **starting in node A**, indicate the order in which the nodes are inserted into the queue Q in the BFS algorithm. It is assumed that the graph is given by alphabetically sorted adjacency lists.

A D E G H I C B F A D E H I B F G C A D E G H I C F B A D E G I H C F B

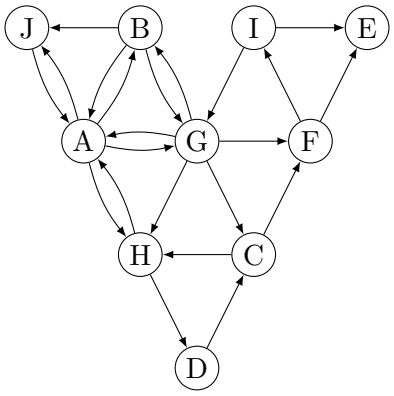
Problem 167 (4 %) check



For a breadth first traversal (BFS) of the graph above **starting in node A**, indicate the order in which the nodes are removed from the queue Q in the BFS algorithm. It is assumed that the graph is given by alphabetically sorted adjacency lists.

A D F G C B E A D G F C E B A D F G C E B A D C B E F G

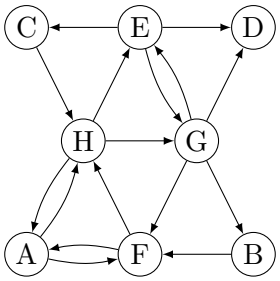
Problem 168 (4 %) check



For a breadth first traversal (BFS) of the graph above **starting in node A**, indicate the order in which the nodes are removed from the queue Q in the BFS algorithm. It is assumed that the graph is given by alphabetically sorted adjacency lists.

A B G C F E I H D J A B G H J C F D I E A B G H J C F D E I A B H J G D C F E I

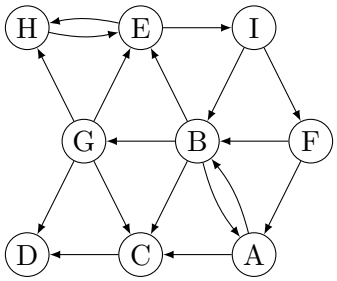
Problem 169 (4 %) check



For a breadth first traversal (BFS) of the graph above **starting in node A**, indicate the order in which the nodes are inserted into the queue Q in the BFS algorithm. It is assumed that the graph is given by alphabetically sorted adjacency lists.

A F H E C D G B A H F G E B D C A F H E G D C B A F H E G C D B

Problem 170 (4 %) check

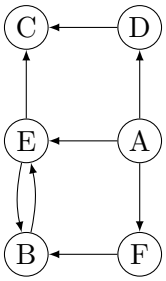


For a breadth first traversal (BFS) of the graph above **starting in node A**, indicate the order in which the nodes are inserted into the queue Q in the BFS algorithm. It is assumed that the graph is given by alphabetically sorted adjacency lists.

A B C D E H I F G A C B D G E H I F A B C E G D H I F A B C E G D I H F

Valid BFS trees

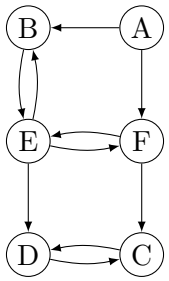
Problem 171 (4 %) check



Indicate for each of the sets of edges below whether they form a valid BFS tree for a breadth first traversal of the graph above **starting in node A** and for a suitable ordering of the graph's adjacency lists.

	Yes	No	
(A,D) (A,F) (B,E) (E,C) (F,B)			171.1
(A,D) (A,E) (A,F) (E,B) (E,C)			171.2
(A,D) (A,E) (A,F) (D,C) (E,B)			171.3
(A,D) (A,F) (B,E) (D,C) (F,B)			171.4
(A,D) (A,E) (A,F) (E,C) (F,B)			171.5

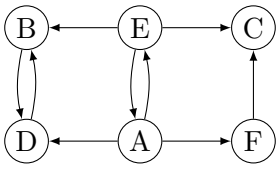
Problem 172 (4 %) check



Indicate for each of the sets of edges below whether they form a valid BFS tree for a breadth first traversal of the graph above **starting in node A** and for a suitable ordering of the graph's adjacency lists.

	Yes	No	
(A,B) (A,F) (E,D) (F,C) (F,E)			172.1
(A,B) (B,E) (D,C) (E,D) (E,F)			172.2
(A,B) (A,F) (D,C) (E,D) (F,E)			172.3
(A,F) (E,B) (E,D) (F,C) (F,E)			172.4
(A,B) (A,F) (C,D) (F,C) (F,E)			172.5

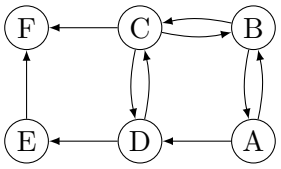
Problem 173 (4 %) check



Indicate for each of the sets of edges below whether they form a valid BFS tree for a breadth first traversal of the graph above **starting in node A** and for a suitable ordering of the graph's adjacency lists.

	Yes	No	
(A,D) (A,E) (A,F) (D,B) (F,C)			173.1
(A,E) (A,F) (B,D) (E,B) (E,C)			173.2
(A,D) (A,E) (A,F) (E,B) (F,C)			173.3
(A,D) (A,E) (A,F) (E,B) (E,C)			173.4
(A,D) (A,E) (A,F) (D,B) (E,C)			173.5

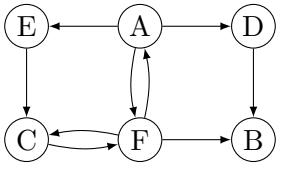
Problem 174 (4 %) check



Indicate for each of the sets of edges below whether they form a valid BFS tree for a breadth first traversal of the graph above **starting in node A** and for a suitable ordering of the graph's adjacency lists.

	Yes	No	
(A,B) (A,D) (C,F) (D,C) (D,E)			174.1
(A,D) (C,B) (D,C) (D,E) (E,F)			174.2
(A,B) (A,D) (D,C) (D,E) (E,F)			174.3
(A,B) (A,D) (B,C) (C,F) (D,E)			174.4
(A,B) (A,D) (B,C) (D,E) (E,F)			174.5

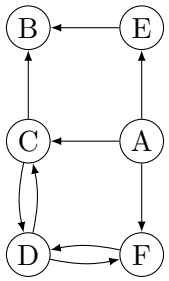
Problem 175 (4 %) check



Indicate for each of the sets of edges below whether they form a valid BFS tree for a breadth first traversal of the graph above **starting in node A** and for a suitable ordering of the graph's adjacency lists.

	Yes	No	
(A,D) (A,E) (A,F) (D,B) (F,C)			175.1
(A,D) (A,E) (A,F) (D,B) (E,C)			175.2
(A,D) (A,E) (A,F) (E,C) (F,B)			175.3
(A,D) (A,E) (C,F) (E,C) (F,B)			175.4
(A,D) (A,E) (A,F) (F,B) (F,C)			175.5

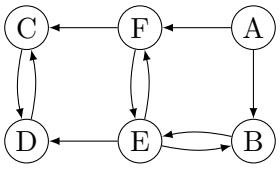
Problem 176 (4 %) check



Indicate for each of the sets of edges below whether they form a valid BFS tree for a breadth first traversal of the graph above **starting in node A** and for a suitable ordering of the graph's adjacency lists.

	Yes	No	
(A,C) (A,E) (A,F) (C,D) (E,B)			176.1
(A,C) (A,E) (A,F) (E,B) (F,D)			176.2
(A,E) (A,F) (C,B) (D,C) (F,D)			176.3
(A,C) (A,E) (A,F) (C,B) (C,D)			176.4
(A,C) (A,E) (A,F) (C,B) (F,D)			176.5

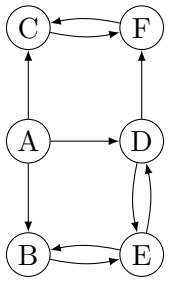
Problem 177 (4 %) check



Indicate for each of the sets of edges below whether they form a valid BFS tree for a breadth first traversal of the graph above **starting in node A** and for a suitable ordering of the graph's adjacency lists.

	Yes	No	
(A,B) (A,F) (E,D) (F,C) (F,E)			177.1
(A,B) (B,E) (D,C) (E,D) (E,F)			177.2
(A,B) (A,F) (B,E) (E,D) (F,C)			177.3
(A,F) (C,D) (E,B) (F,C) (F,E)			177.4
(A,B) (A,F) (B,E) (D,C) (E,D)			177.5

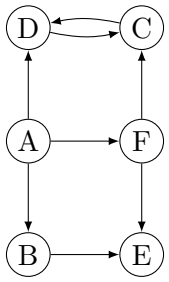
Problem 178 (4 %) check



Indicate for each of the sets of edges below whether they form a valid BFS tree for a breadth first traversal of the graph above **starting in node A** and for a suitable ordering of the graph's adjacency lists.

	Yes	No	
(A,D) (D,E) (D,F) (E,B) (F,C)			178.1
(A,B) (A,C) (A,D) (C,F) (D,E)			178.2
(A,B) (A,C) (A,D) (B,E) (C,F)			178.3
(A,B) (A,C) (B,E) (C,F) (E,D)			178.4
(A,B) (A,C) (A,D) (D,E) (D,F)			178.5

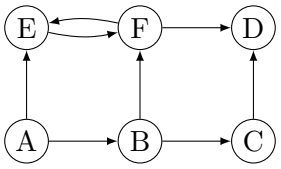
Problem 179 (4 %) check



Indicate for each of the sets of edges below whether they form a valid BFS tree for a breadth first traversal of the graph above **starting in node A** and for a suitable ordering of the graph's adjacency lists.

	Yes	No	
(A,B) (A,F) (C,D) (F,C) (F,E)			179.1
(A,B) (A,D) (A,F) (B,E) (F,C)			179.2
(A,B) (A,D) (A,F) (F,C) (F,E)			179.3
(A,B) (A,D) (A,F) (B,E) (D,C)			179.4
(A,B) (A,D) (A,F) (D,C) (F,E)			179.5

Problem 180 (4 %) **check**

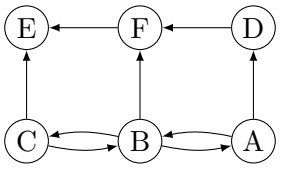


Indicate for each of the sets of edges below whether they form a valid BFS tree for a breadth first traversal of the graph above **starting in node A** and for a suitable ordering of the graph's adjacency lists.

	Yes	No	
(A,B) (B,C) (B,F) (F,D) (F,E)			180.1
(A,B) (A,E) (B,C) (C,D) (E,F)			180.2
(A,B) (A,E) (B,C) (B,F) (C,D)			180.3
(A,B) (A,E) (B,C) (E,F) (F,D)			180.4
(A,B) (A,E) (B,C) (B,F) (F,D)			180.5

DFS

Problem 181 (4 %) **check**



Consider a depth first traversal (DFS) of the graph above, where the DFS traversal starts in **node A**, and where the outgoing edges of a node are visited in alphabetical order. Indicate the order in which the nodes receive their **discovery time**.

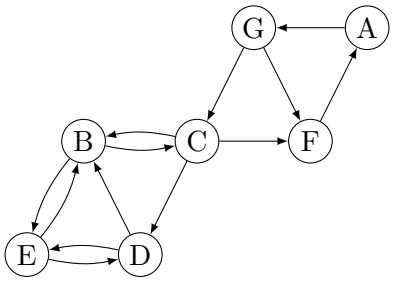
A B C E F D A D B F C E A B D C F E A D F E B C

181.1

Indicate for each of the edges below which type the edge is in the DFS traversal.

	Tree edge	Back edge	Cross edge	Forward edge	
(A, D)					181.2
(C, B)					181.3
(F, E)					181.4
(B, A)					181.5

Problem 182 (4 %) **check**



Consider a depth first traversal (DFS) of the graph above, where the DFS traversal starts in **node A**, and where the outgoing edges of a node are visited in alphabetical order. Indicate the order in which the nodes receive their **discovery time**.

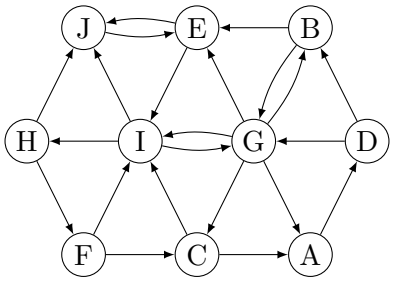
A G C F B E D A G C B E D F A G C F B D E A G F C B E D

182.1

Indicate for each of the edges below which type the edge is in the DFS traversal.

	Tree edge	Back edge	Cross edge	Forward edge	
(C, D)					182.2
(D, E)					182.3
(D, B)					182.4
(C, B)					182.5

Problem 183 (4 %) **check**



Consider a depth first traversal (DFS) of the graph above, where the DFS traversal starts in **node A**, and where the outgoing edges of a node are visited in alphabetical order. Indicate the order in which the nodes receive their **discovery time**.

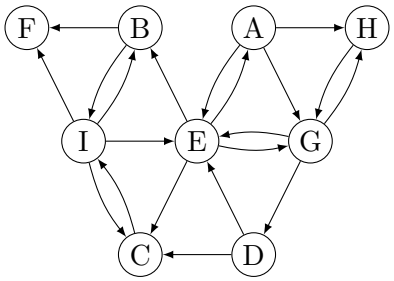
A D B G E C I J H F A D B E I H J F G C A D B E I G C H F J A D G B E J I H F C

183.1

Indicate for each of the edges below which type the edge is in the DFS traversal.

	Tree edge	Back edge	Cross edge	Forward edge	
(F, C)					183.2
(E, J)					183.3
(F, I)					183.4
(B, E)					183.5

Problem 184 (4 %) **check**



Consider a depth first traversal (DFS) of the graph above, where the DFS traversal starts in **node A**, and where the outgoing edges of a node are visited in alphabetical order. Indicate the order in which the nodes receive their **finishing time**.

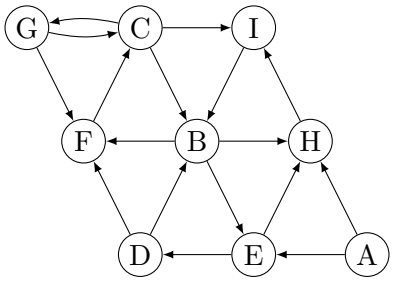
F C I B D H G E A H D G C I F B E A F B I C D H G E A I F D C B H G E A

184.1

Indicate for each of the edges below which type the edge is in the DFS traversal.

	Tree edge	Back edge	Cross edge	Forward edge	
(D, E)					184.2
(A, H)					184.3
(A, E)					184.4
(I, F)					184.5

Problem 185 (4 %) **check**



Consider a depth first traversal (DFS) of the graph above, where the DFS traversal starts in **node A**, and where the outgoing edges of a node are visited in alphabetical order. Indicate the order in which the nodes receive their **discovery time**.

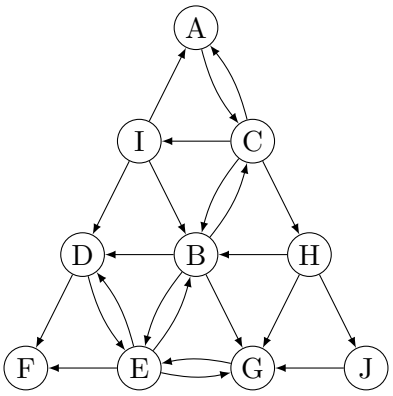
A E D B H F C I G A E D B H I F C G A E D B F C G I H A E H D I B F C G

185.1

Indicate for each of the edges below which type the edge is in the DFS traversal.

	Tree edge	Back edge	Cross edge	Forward edge	
(D, F)					185.2
(I, B)					185.3
(B, H)					185.4
(H, I)					185.5

Problem 186 (4 %) **check**



Consider a depth first traversal (DFS) of the graph above, where the DFS traversal starts in **node A**, and where the outgoing edges of a node are visited in alphabetical order. Indicate the order in which the nodes receive their **finishing time**.

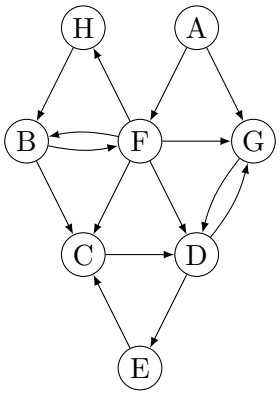
I J H G F E D B C A F G B E D I J H C A F J G E D I H B C A F G E D B J H I C A

186.1

Indicate for each of the edges below which type the edge is in the DFS traversal.

	Tree edge	Back edge	Cross edge	Forward edge	
(D, F)					186.2
(D, E)					186.3
(I, A)					186.4
(I, D)					186.5

Problem 187 (4 %) **check**



Consider a depth first traversal (DFS) of the graph above, where the DFS traversal starts in **node A**, and where the outgoing edges of a node are visited in alphabetical order. Indicate the order in which the nodes receive their **finishing time**.

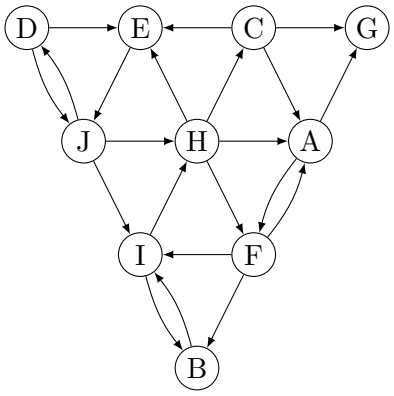
E G D C B H F A E H D C B G F A H G E D C B F A G E D C B H F A

187.1

Indicate for each of the edges below which type the edge is in the DFS traversal.

	Tree edge	Back edge	Cross edge	Forward edge	
(G, D)					187.2
(H, B)					187.3
(D, G)					187.4
(A, G)					187.5

Problem 188 (4 %) **check**



Consider a depth first traversal (DFS) of the graph above, where the DFS traversal starts in **node A**, and where the outgoing edges of a node are visited in alphabetical order. Indicate the order in which the nodes receive their **discovery time**.

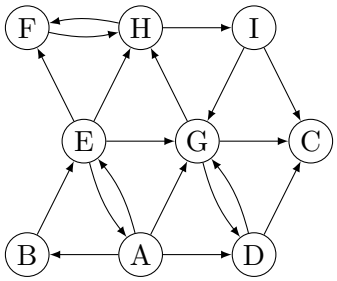
A F G B I H C E J D A F B I H C G E J D A F B I H C E J D G A G F I H C E J D B

188.1

Indicate for each of the edges below which type the edge is in the DFS traversal.

	Tree edge	Back edge	Cross edge	Forward edge
(H, C)				188.2
(D, J)				188.3
(F, I)				188.4
(I, B)				188.5

Problem 189 (4 %) **check**



Consider a depth first traversal (DFS) of the graph above, where the DFS traversal starts in **node A**, and where the outgoing edges of a node are visited in alphabetical order. Indicate the order in which the nodes receive their **discovery time**.

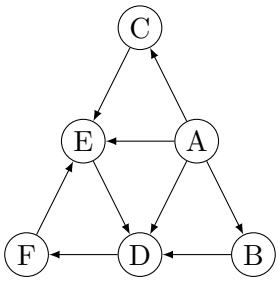
A B E F H I C G D A B D E G C F H I A B E F H I G D C A E F H I G C D B

189.1

Indicate for each of the edges below which type the edge is in the DFS traversal.

	Tree edge	Back edge	Cross edge	Forward edge	
(E, H)					189.2
(G, C)					189.3
(E, F)					189.4
(H, F)					189.5

Problem 190 (4 %) check



Consider a depth first traversal (DFS) of the graph above, where the DFS traversal starts in **node A**, and where the outgoing edges of a node are visited in alphabetical order. Indicate the order in which the nodes receive their **finishing time**.

C E F D B A E F D B C A F D E B C A F E D C B A

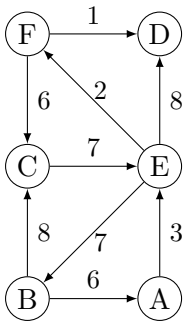
190.1

Indicate for each of the edges below which type the edge is in the DFS traversal.

	Tree edge	Back edge	Cross edge	Forward edge
(F, E)				190.2
(E, D)				190.3
(C, E)				190.4
(A, E)				190.5

Dijkstra's algorithm

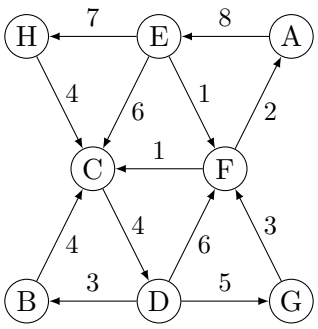
Problem 191 (4 %) check



Assume Dijkstra's algorithm is used to find shortest distances from **node A** to all nodes in the graph above. Indicate the order in which the nodes are removed from the priority queue in Dijkstra's algorithm.

A E B C D F A E B D F C A E F D C B A E F D B C

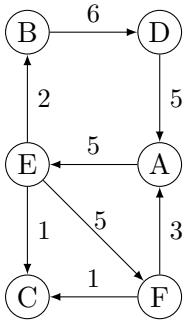
Problem 192 (4 %) check



Assume Dijkstra's algorithm is used to find shortest distances from **node A** to all nodes in the graph above. Indicate the order in which the nodes are removed from the priority queue in Dijkstra's algorithm.

A E C D B F G H A E F C D B G H A E C F H D B G A E F C D H B G

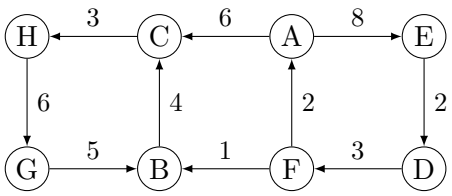
Problem 193 (4 %) check



Assume Dijkstra's algorithm is used to find shortest distances from **node A** to all nodes in the graph above. Indicate the order in which the nodes are removed from the priority queue in Dijkstra's algorithm.

A E B C F D A E C B F D A E C B D F A E B D C F

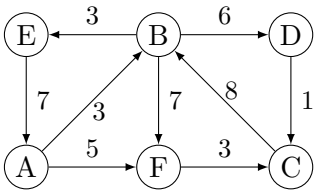
Problem 194 (4 %) check



Assume Dijkstra's algorithm is used to find shortest distances from **node A** to all nodes in the graph above. Indicate the order in which the nodes are removed from the priority queue in Dijkstra's algorithm.

A C E H G B D F A C E H D G F B A C E H D F B G A C H G B E D F

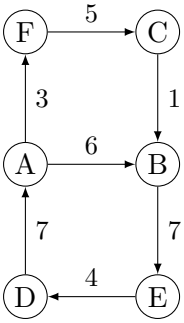
Problem 195 (4 %) check



Assume Dijkstra's algorithm is used to find shortest distances from **node A** to all nodes in the graph above. Indicate the order in which the nodes are removed from the priority queue in Dijkstra's algorithm.

A B F C E D A B D C E F A B F E C D A B F D E C

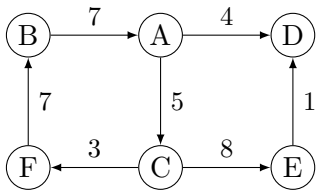
Problem 196 (4 %) check



Assume Dijkstra's algorithm is used to find shortest distances from **node A** to all nodes in the graph above. Indicate the order in which the nodes are removed from the priority queue in Dijkstra's algorithm.

A F B C E D A F B E D C A B F E C D A B E D F C

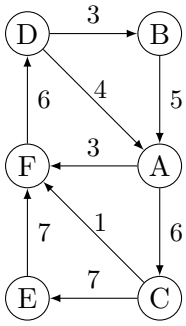
Problem 197 (4 %) check



Assume Dijkstra's algorithm is used to find shortest distances from **node A** to all nodes in the graph above. Indicate the order in which the nodes are removed from the priority queue in Dijkstra's algorithm.

A D C F B E A D C F E B A C D E F B A C E D F B

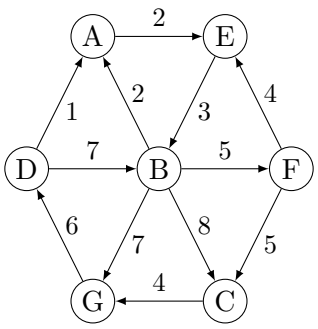
Problem 198 (4 %) check



Assume Dijkstra's algorithm is used to find shortest distances from **node A** to all nodes in the graph above. Indicate the order in which the nodes are removed from the priority queue in Dijkstra's algorithm.

A C E F D B A F C E D B A C F E D B A F C D B E

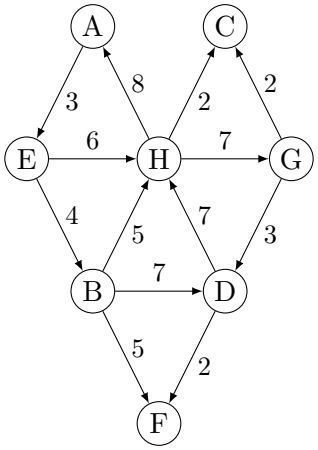
Problem 199 (4 %) check



Assume Dijkstra's algorithm is used to find shortest distances from **node A** to all nodes in the graph above. Indicate the order in which the nodes are removed from the priority queue in Dijkstra's algorithm.

A E B C F G D A E B C G D F A E B F C G D A E B F G C D

Problem 200 (4 %) check

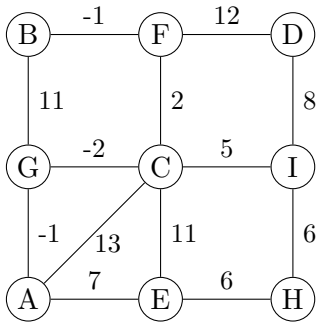


Assume Dijkstra's algorithm is used to find shortest distances from **node A** to all nodes in the graph above. Indicate the order in which the nodes are removed from the priority queue in Dijkstra's algorithm.

A E B H C G D F A E B H C F D G A E B D F H C G A E B H D F C G

Prim's algorithm

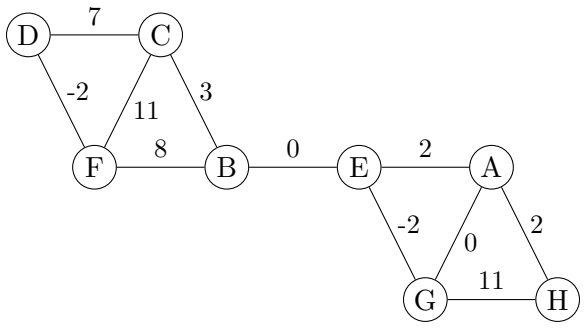
Problem 201 (4 %) check



Assume Prim's algorithm is used to find a minimum spanning tree for the graph above, and the algorithm starts in **node A**. Indicate the order in which the nodes are included in the minimum spanning tree (taken out of the priority queue in Prim's algorithm).

A G C F B I H E D A G C F B D I H E A G C F B I H D E A G C F B I E H D

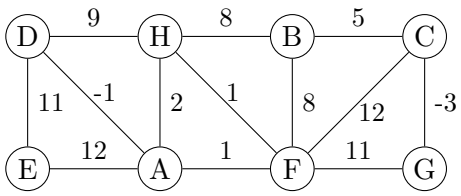
Problem 202 (4 %) check



Assume Prim's algorithm is used to find a minimum spanning tree for the graph above, and the algorithm starts in **node A**. Indicate the order in which the nodes are included in the minimum spanning tree (taken out of the priority queue in Prim's algorithm).

A G E B H C D F A G E B C H F D A G E B C D F H A G E B C H D F

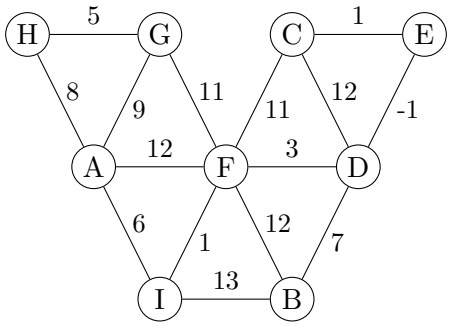
Problem 203 (4 %) check



Assume Prim's algorithm is used to find a minimum spanning tree for the graph above, and the algorithm starts in **node A**. Indicate the order in which the nodes are included in the minimum spanning tree (taken out of the priority queue in Prim's algorithm).

A D F H B E G C A D F H B C G E A D F H B E C G A D H F B C G E

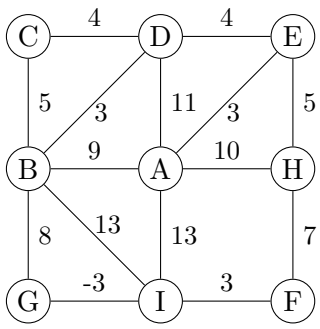
Problem 204 (4 %) check



Assume Prim's algorithm is used to find a minimum spanning tree for the graph above, and the algorithm starts in **node A**. Indicate the order in which the nodes are included in the minimum spanning tree (taken out of the priority queue in Prim's algorithm).

A I F D E C B H G A I F H G D E C B A I F D E C B G H A I F H D E C G B

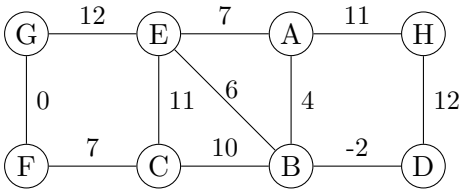
Problem 205 (4 %) check



Assume Prim's algorithm is used to find a minimum spanning tree for the graph above, and the algorithm starts in **node A**. Indicate the order in which the nodes are included in the minimum spanning tree (taken out of the priority queue in Prim's algorithm).

A E D B C G I F H A E D H B C I G F A E D B C H F I G A E D H B C F I G

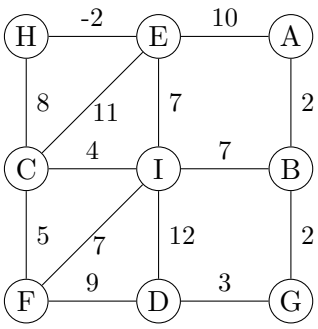
Problem 206 (4 %) check



Assume Prim's algorithm is used to find a minimum spanning tree for the graph above, and the algorithm starts in **node A**. Indicate the order in which the nodes are included in the minimum spanning tree (taken out of the priority queue in Prim's algorithm).

A B D E H C G F A B D H E C F G A B D E H C F G A B D E C F G H

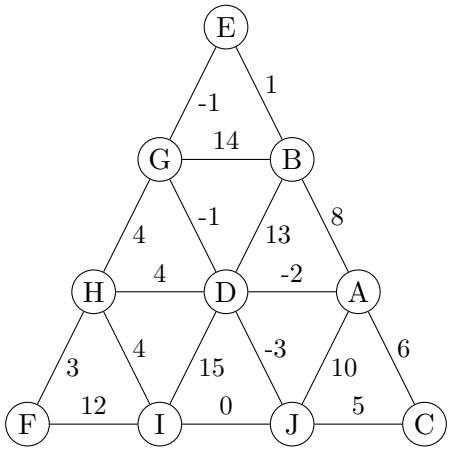
Problem 207 (4 %) check



Assume Prim's algorithm is used to find a minimum spanning tree for the graph above, and the algorithm starts in **node A**. Indicate the order in which the nodes are included in the minimum spanning tree (taken out of the priority queue in Prim's algorithm).

A B G D F C I E H A B G D I E H C F A B G D I C F E H A B G D I C E H F

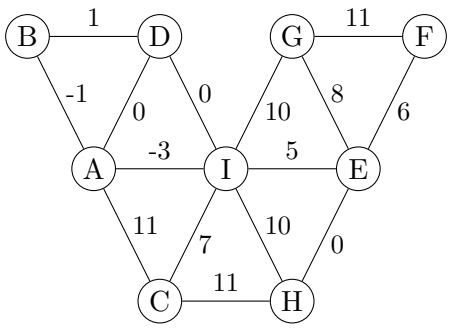
Problem 208 (4 %) check



Assume Prim's algorithm is used to find a minimum spanning tree for the graph above, and the algorithm starts in **node A**. Indicate the order in which the nodes are included in the minimum spanning tree (taken out of the priority queue in Prim's algorithm).

A D J I H F G E B C A D J G E I B H F C A D J I G E B C H F A D J I G E B H C F

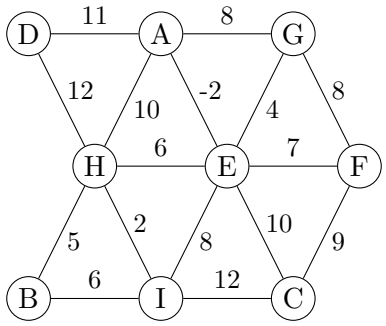
Problem 209 (4 %) **check**



Assume Prim's algorithm is used to find a minimum spanning tree for the graph above, and the algorithm starts in **node A**. Indicate the order in which the nodes are included in the minimum spanning tree (taken out of the priority queue in Prim's algorithm).

A I B D E H C F G A I D B E H C G F A I D B E H C F G A I B D E H F C G

Problem 210 (4 %) **check**

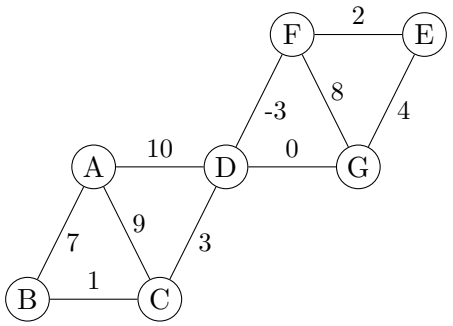


Assume Prim's algorithm is used to find a minimum spanning tree for the graph above, and the algorithm starts in **node A**. Indicate the order in which the nodes are included in the minimum spanning tree (taken out of the priority queue in Prim's algorithm).

A E G H F I B D C A E G F C I H B D A E G H I B F C D A E G H F I C B D

Kruskal's algorithm

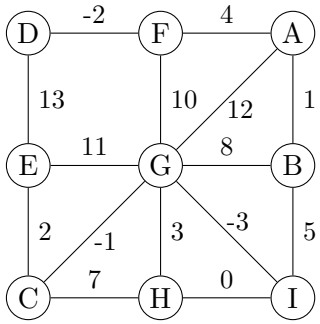
Problem 211 (4 %) check



Assume Kruskal's algorithm is used to find a minimum spanning tree for the graph above. Indicate which edge is **last** included in the minimum spanning tree.

(B , C) (A , B) (E , G) (E , F) (D , G)

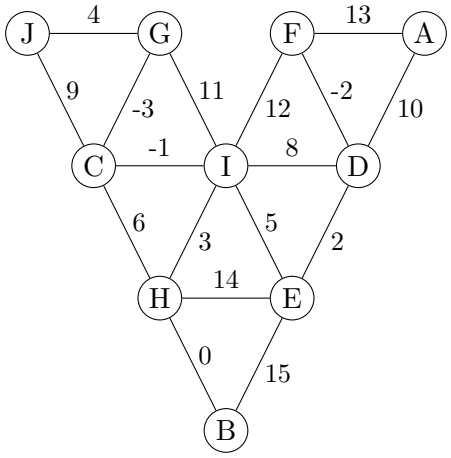
Problem 212 (4 %) check



Assume Kruskal's algorithm is used to find a minimum spanning tree for the graph above. Indicate which edge is **last** included in the minimum spanning tree.

(B , I) (C , H) (C , E) (H , I) (G , I)

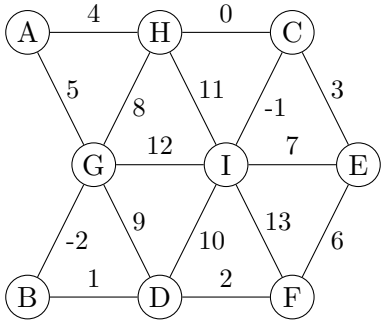
Problem 213 (4 %) check



Assume Kruskal's algorithm is used to find a minimum spanning tree for the graph above. Indicate which edge is **last** included in the minimum spanning tree.

(B , H) (B , E) (A , D) (C , I) (G , J)

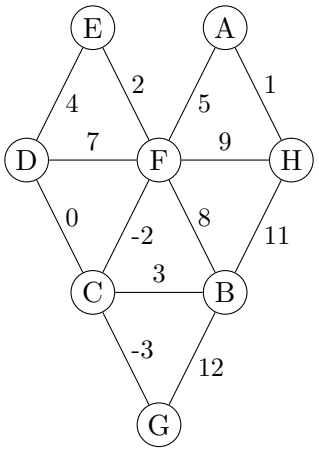
Problem 214 (4 %) check



Assume Kruskal's algorithm is used to find a minimum spanning tree for the graph above. Indicate which edge is **last** included in the minimum spanning tree.

(B , G) (A , G) (C , E) (A , H) (D , F)

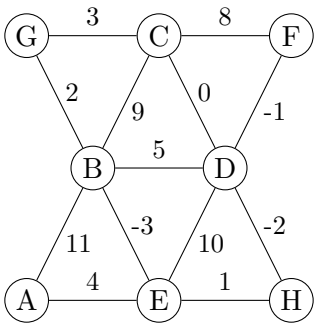
Problem 215 (4 %) check



Assume Kruskal's algorithm is used to find a minimum spanning tree for the graph above. Indicate which edge is **last** included in the minimum spanning tree.

(D , E) (A , F) (E , F) (C , F) (B , C)

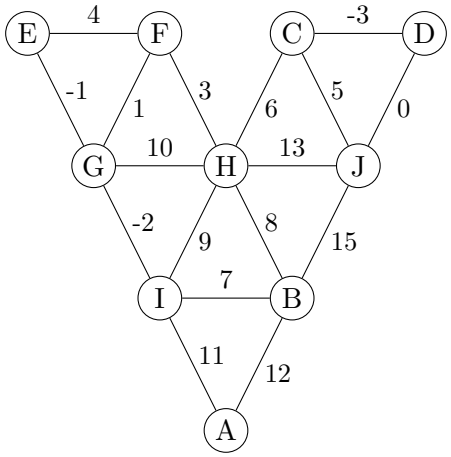
Problem 216 (4 %) check



Assume Kruskal's algorithm is used to find a minimum spanning tree for the graph above. Indicate which edge is **last** included in the minimum spanning tree.

(D , F) (C , D) (A , E) (B , G) (B , E)

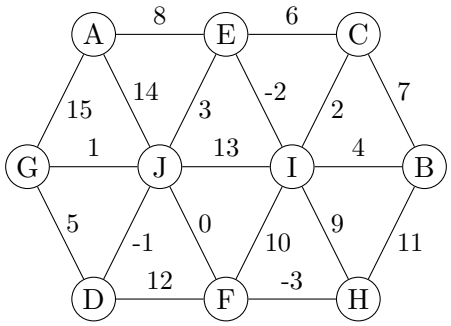
Problem 217 (4 %) check



Assume Kruskal's algorithm is used to find a minimum spanning tree for the graph above. Indicate which edge is **last** included in the minimum spanning tree.

(B , I) (C , H) (D , J) (B , J) (A , I)

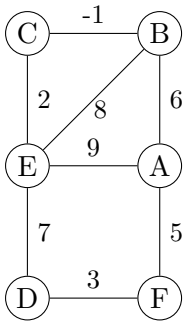
Problem 218 (4 %) check



Assume Kruskal's algorithm is used to find a minimum spanning tree for the graph above. Indicate which edge is **last** included in the minimum spanning tree.

(D , G) (B , I) (C , I) (A , E) (G , J)

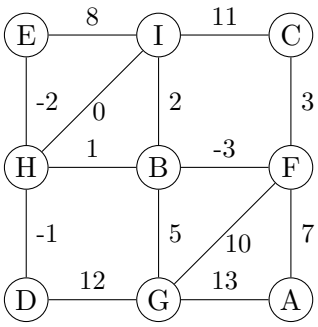
Problem 219 (4 %) check



Assume Kruskal's algorithm is used to find a minimum spanning tree for the graph above. Indicate which edge is **last** included in the minimum spanning tree.

(A , F) (B , C) (D , F) (C , E) (A , B)

Problem 220 (4 %) check

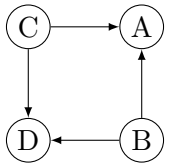


Assume Kruskal's algorithm is used to find a minimum spanning tree for the graph above. Indicate which edge is **last** included in the minimum spanning tree.

(C , F) (B , G) (B , H) (D , G) (A , F)

Topological sorting

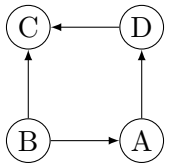
Problem 221 (4 %) check



For each of the following orderings of the nodes in the graph above, indicate whether it is a valid topological sort.

	Yes	No	
B C D A			221.1
C B A D			221.2
C D A B			221.3
B C A D			221.4
D B A C			221.5

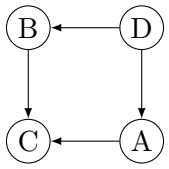
Problem 222 (4 %) check



For each of the following orderings of the nodes in the graph above, indicate whether it is a valid topological sort.

	Yes	No	
A B D C			222.1
C A D B			222.2
B D A C			222.3
B A D C			222.4
B A C D			222.5

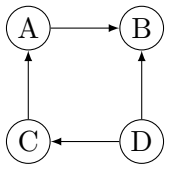
Problem 223 (4 %) check



For each of the following orderings of the nodes in the graph above, indicate whether it is a valid topological sort.

	Yes	No	
D C A B			223.1
D A B C			223.2
D B A C			223.3
C A B D			223.4
A D B C			223.5

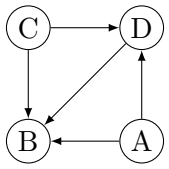
Problem 224 (4 %) check



For each of the following orderings of the nodes in the graph above, indicate whether it is a valid topological sort.

	Yes	No	
D C B A			224.1
D C A B			224.2
D A C B			224.3
B C A D			224.4
C D A B			224.5

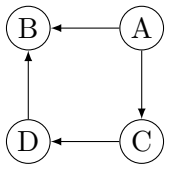
Problem 225 (4 %) check



For each of the following orderings of the nodes in the graph above, indicate whether it is a valid topological sort.

	Yes	No	
A C D B			225.1
B C D A			225.2
C A B D			225.3
C A D B			225.4
C D A B			225.5

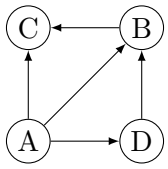
Problem 226 (4 %) check



For each of the following orderings of the nodes in the graph above, indicate whether it is a valid topological sort.

	Yes	No	
A C D B			226.1
A D C B			226.2
A B D C			226.3
D C A B			226.4
C A D B			226.5

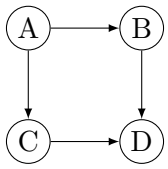
Problem 227 (4 %) check



For each of the following orderings of the nodes in the graph above, indicate whether it is a valid topological sort.

	Yes	No	
A D C B			227.1
A D B C			227.2
C D B A			227.3
A C B D			227.4
D A B C			227.5

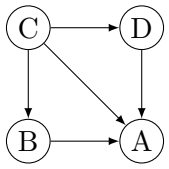
Problem 228 (4 %) check



For each of the following orderings of the nodes in the graph above, indicate whether it is a valid topological sort.

	Yes	No	
B A C D			228.1
C B A D			228.2
A C B D			228.3
A D B C			228.4
A B C D			228.5

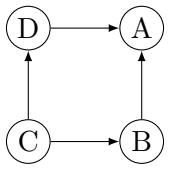
Problem 229 (4 %) **check**



For each of the following orderings of the nodes in the graph above, indicate whether it is a valid topological sort.

	Yes	No	
D B C A			229.1
C B A D			229.2
A B D C			229.3
C B D A			229.4
C D B A			229.5

Problem 230 (4 %) **check**

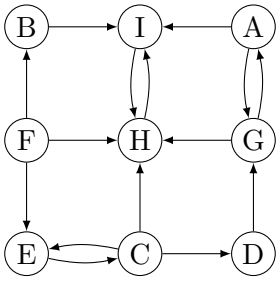


For each of the following orderings of the nodes in the graph above, indicate whether it is a valid topological sort.

	Yes	No	
C D B A			230.1
C B D A			230.2
C D A B			230.3
C B A D			230.4
D B C A			230.5

Strongly connected components

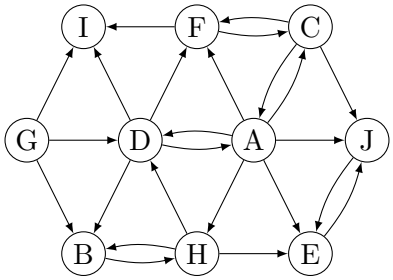
Problem 231 (4 %) check



What is the number of strongly connected components in the graph above?

1 2 3 4 5 6 7 8 9

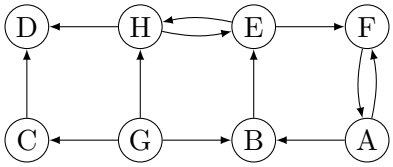
Problem 232 (4 %) check



What is the number of strongly connected components in the graph above?

1 2 3 4 5 6 7 8 9 10

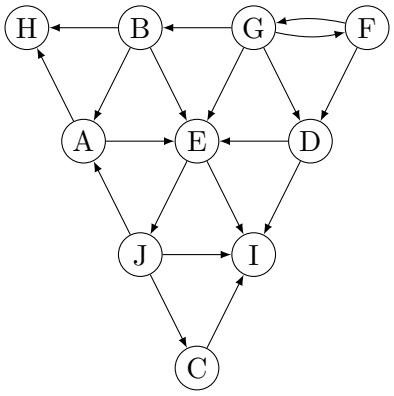
Problem 233 (4 %) check



What is the number of strongly connected components in the graph above?

1 2 3 4 5 6 7 8

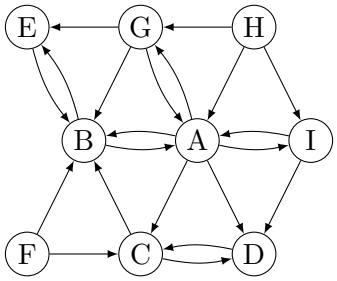
Problem 234 (4 %) check



What is the number of strongly connected components in the graph above?

1 2 3 4 5 6 7 8 9 10

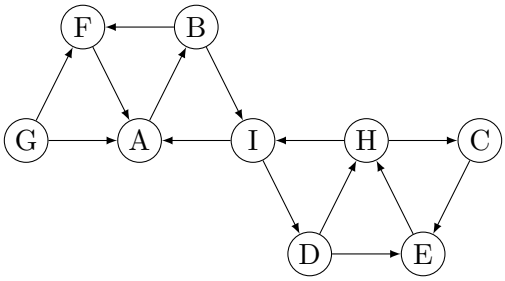
Problem 235 (4 %) check



What is the number of strongly connected components in the graph above?

1 2 3 4 5 6 7 8 9

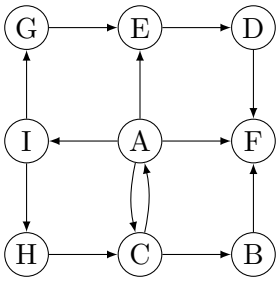
Problem 236 (4 %) check



What is the number of strongly connected components in the graph above?

1 2 3 4 5 6 7 8 9

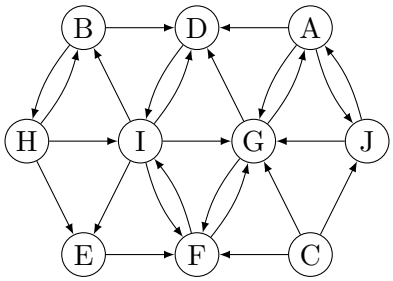
Problem 237 (4 %) check



What is the number of strongly connected components in the graph above?

1 2 3 4 5 6 7 8 9

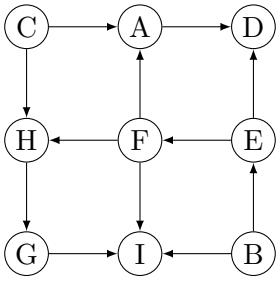
Problem 238 (4 %) check



What is the number of strongly connected components in the graph above?

1 2 3 4 5 6 7 8 9 10

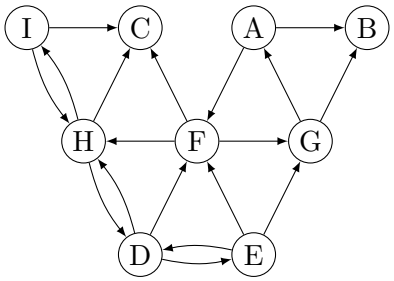
Problem 239 (4 %) check



What is the number of strongly connected components in the graph above?

1 2 3 4 5 6 7 8 9

Problem 240 (4 %) **check**



What is the number of strongly connected components in the graph above?

1 2 3 4 5 6 7 8 9

Loop problems

Problem 241 check

Algorithm loop1(n)
$i = 1$
while $i \leq n$
  $j = 1$
  while $j \leq i$
    $j = 2 * j$
  $i = 2 * i$

Algorithm loop2(n)
$i = 1$
while $i \leq n$
  $j = 1$
  while $j \leq n$
    $j = 2 * j$
  $i = 2 * i$

Algorithm loop3(n)
$i = 1$
while $i \leq n$
  $j = i$
  while $j \leq n$
    $j = 2 * j$
  $i = 2 * i$

Algorithm loop4(n)
$i = 1$
while $i \leq n$
  $j = i$
  while $j > 0$
    $j = \lfloor j/2 \rfloor$
  $i = i + i$

Algorithm loop5(n)
$i = n$
while $i > 0$
  $j = i$
  while $j > 0$
    $j = \lfloor j/2 \rfloor$
  $i = \lfloor i/2 \rfloor$

Algorithm loop6(n)
$s = 1$
for $i = 1$ **to** n
  $j = 1$
  while $j \leq s$
    $j = j + 1$
  $s = 2 * s$

Algorithm loop7(n)
$s = 1$
for $i = 1$ **to** n
  $j = s$
  while $j > 0$
    $s = s + 1$
    $j = j - 1$

Algorithm loop8(n)
$i = 1$
$p = 1$
while $p \leq n$
  $i = i + 1$
  $p = p * i$

Algorithm loop9(n)
$i = 1$
$j = n$
while $i < j$
  $i = 2 * i$
  $j = j + n$

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta(2^n)$	$\Theta(n^2)$	$\Theta(n^3)$	$\Theta(\log n)$	$\Theta(\frac{\log n}{\log \log n})$	$\Theta(n \log n)$	$\Theta(n)$	$\Theta((\log n)^2)$	
loop1									241.1
loop2									241.2
loop3									241.3
loop4									241.4
loop5									241.5
loop6									241.6
loop7									241.7
loop8									241.8
loop9									241.9

Problem 242 check

Algorithm loop1(n)
$i = 1$
$j = n$
while $i \leq j$
  $i = 4 * i$
  $j = 2 * j$

Algorithm loop2(n)
$i = 1$
$j = n$
while $i \leq j$
  $i = i * 2$
  $j = \lfloor j/2 \rfloor$

Algorithm loop3(n)
$i = 1$
while $i * i \leq n$
  $i = i + i$

Algorithm loop4(n)
$i = 1$
while $i \leq n$
  $i = 2 * i$

Algorithm loop5(n)
$i = 1$
while $i \leq n$
  $i = 3 * i$

Algorithm loop6(n)
$i = 1$
while $i \leq n$
  $i = i + i$

Algorithm loop7(n)
$i = 1$
while $i \leq n * n$
  $i = 2 * i$

Algorithm loop8(n)
$i = 1$
while $i \leq n * n$
  $i = 3 * i$

Algorithm loop9(n)
$i = 1$
$j = n * n$
while $i \leq j$
  $i = 2 * i$
  $j = j - 1$

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta(n \log n)$	$\Theta(n^3)$	$\Theta(\log n)$	$\Theta(\sqrt{n} \log n)$	$\Theta(n^2)$	$\Theta(\frac{\log n}{\log \log n})$	$\Theta((\log n)^2)$	$\Theta(n)$	
loop1									242.1
loop2									242.2
loop3									242.3
loop4									242.4
loop5									242.5
loop6									242.6
loop7									242.7
loop8									242.8
loop9									242.9

Problem 243 check

Algorithm loop1(n)
```
i = n
while i > 0
    if i ulige then
        i = i − 1
    else
        i = i/2
```

Algorithm loop2(n)
```
i = n
while i ≤ n ∗ n
    i = 2 ∗ i
```

Algorithm loop3(n)
```
s = n
while s > 0
    s = ⌊s/2⌋
```

Algorithm loop4(n)
```
i = 2
while i ≤ n
    i = i ∗ i
```

Algorithm loop5(n)
```
s = 2
while s ≤ n
    s = s ∗ s
```

Algorithm loop6(n)
```
i = 0
s = 0
q = 0
while q ≤ n
    i = i + 1
    s = s + i
    q = q + s
```

Algorithm loop7(n)
```
i = 0
s = 0
while s ≤ n
    i = i + 1
    s = s + i
```

Algorithm loop8(n)
```
i = 1
j = 1
s = 0
while s ≤ n
    while j ≤ s
        j = 2 ∗ j
    s = s + i
    i = i + 1
```

Algorithm loop9(n)
```
j = n
i = 1
while j ≥ 0
    j = j − i
    i = i + 1
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta(n^2)$	$\Theta(n \log n)$	$\Theta(\sqrt[3]{n})$	$\Theta(\log \log n)$	$\Theta(n)$	$\Theta((\log n)^2)$	$\Theta(\log n)$	$\Theta(n^3)$	$\Theta(\sqrt{n})$	
loop1										243.1
loop2										243.2
loop3										243.3
loop4										243.4
loop5										243.5
loop6										243.6
loop7										243.7
loop8										243.8
loop9										243.9

Problem 244 check

```
Algorithm loop1(n)
  s = 0
  i = 1
  while s ≤ n
    s = s + i
    i = i + 1
```

```
Algorithm loop2(n)
  i = 1
  while i ≤ n
    j = 1
    k = 1
    while k ≤ n
      j = j + 1
      k = k + j
    i = 2 * i
```

```
Algorithm loop3(n)
  for i = 1 to n
    j = i
    while j ≤ n
      j = 2 * j
```

```
Algorithm loop4(n)
  i = 0
  j = n
  while i ≤ j
    i = i + 1
    j = j − 1
```

```
Algorithm loop5(n)
  i = 0
  j = n
  while i < j
    i = i + 2
    j = j + 1
```

```
Algorithm loop6(n)
  i = 1
  j = 0
  while i ≤ n
    i = i + i
    while j < i
      j = j + 1
```

```
Algorithm loop7(n)
  i = 1
  j = 1
  while i ≤ n
    while j ≤ i
      j = j + 1
    i = 2 * i
```

```
Algorithm loop8(n)
  i = 1
  s = 0
  while s ≤ n
    j = 1
    while j ≤ i
      j = j + 1
    s = s + i
    i = i + 1
```

```
Algorithm loop9(n)
  i = 1
  s = 1
  while s ≤ n * n
    i = i + 1
    s = s + i
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta(\log n)$	$\Theta(\sqrt{n})$	$\Theta((\log n)^2)$	$\Theta(n^2)$	$\Theta(n^3)$	$\Theta(\sqrt{n}\log n)$	$\Theta(n)$	$\Theta(n \log n)$	
loop1									244.1
loop2									244.2
loop3									244.3
loop4									244.4
loop5									244.5
loop6									244.6
loop7									244.7
loop8									244.8
loop9									244.9

Problem 245 check

```
Algorithm loop1(n)
  i = 1
  while i ≤ n
    j = 0
    while j ≤ i
      j = j + 1
    i = 2 * i
```

```
Algorithm loop2(n)
  i = 1
  while i ≤ n
    j = 0
    while j ≤ n
      j = j + i
    i = 2 * i
```

```
Algorithm loop3(n)
  i = 1
  while i ≤ n
    j = 1
    while j ≤ i
      j = j + 1
    i = 2 * i
```

```
Algorithm loop4(n)
  i = n
  while i > 0
    i = i − 1
```

```
Algorithm loop5(n)
  i = n
  while i ≥ 1
    j = i
    while j ≤ n
      j = 2 * j
    i = i − 1
```

```
Algorithm loop6(n)
  s = 0
  i = 1
  while i * i ≤ n
    for j = 1 to i
      s = s + 1
    i = i + 1
```

```
Algorithm loop7(n)
  s = 0
  i = n
  while i > 1
    for j = 1 to i
      s = s + 1
    i = ⌊i/2⌋
```

```
Algorithm loop8(n)
  s = 1
  for i = 1 to n
    s = s + 1
```

```
Algorithm loop9(n)
  s = 1
  for i = n to 1 step −1
    s = s + 1
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta(n)$	$\Theta(\sqrt{n}\log n)$	$\Theta(n\sqrt{n})$	$\Theta((\log n)^2)$	$\Theta(n^2)$	$\Theta(n\log n)$	$\Theta(\log n)$	$\Theta(n^3)$	
loop1									245.1
loop2									245.2
loop3									245.3
loop4									245.4
loop5									245.5
loop6									245.6
loop7									245.7
loop8									245.8
loop9									245.9

Problem 246

check

Algorithm loop1(n)
```
s = 1
i = 1
while i ≤ n
    for j = 1 to i
        s = s + 1
    i = 2 * i
```

Algorithm loop2(n)
```
s = 1
while s ≤ n
    s = s + 1
```

Algorithm loop3(n)
```
for i = 1 to n
    j = 0
    while j ≤ n
        j = j + i
```

Algorithm loop4(n)
```
for i = 1 to n
    j = 1
    while j ≤ i
        j = 2 * j
```

Algorithm loop5(n)
```
for i = 1 to n
    j = 1
    while j ≤ n
        j = 2 * j
```

Algorithm loop6(n)
```
for i = 1 to n
    j = i
    while j > 1
        j = ⌊j/2⌋
```

Algorithm loop7(n)
```
i = 0
while i ≤ n
    j = i
    while j > 0
        j = ⌊j/2⌋
    i = i + 1
```

Algorithm loop8(n)
```
i = 1
j = 1
s = 0
while i ≤ n
    if i = j then
        for k = 1 to n
            s = s + 1
        j = 2 * j
    i = i + 1
```

Algorithm loop9(n)
```
i = 1
s = 0
while i ≤ n
    for j = i to n
        s = s + 1
    i = i + i
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta(n^2)$	$\Theta(n \log n)$	$\Theta(\log n)$	$\Theta(\frac{\log n}{\log \log n})$	$\Theta(n)$	$\Theta(n^3)$	$\Theta(n\sqrt{n})$	$\Theta((\log n)^2)$	
loop1									246.1
loop2									246.2
loop3									246.3
loop4									246.4
loop5									246.5
loop6									246.6
loop7									246.7
loop8									246.8
loop9									246.9

Problem 247 check

```
Algorithm loop1(n)
  i = 1
  while i ≤ n
    j = 0
    while j ≤ n
      j = j + 1
    i = 2 * i
```

```
Algorithm loop2(n)
  i = 1
  while i ≤ n
    j = 1
    while j ≤ i
      j = 2 * j
    i = i + 1
```

```
Algorithm loop3(n)
  i = 1
  while i ≤ n
    j = i
    while j ≤ n
      j = j + 1
    i = 2 * i
```

```
Algorithm loop4(n)
  i = 1
  while i ≤ n
    j = n
    while j > 1
      j = j - 1
    i = 2 * i
```

```
Algorithm loop5(n)
  s = 0
  i = n
  while i > 1
    for j = 1 to n
      s = s + 1
    i = ⌊i/2⌋
```

```
Algorithm loop6(n)
  for i = 0 to n
    j = 0
    s = 0
    while s ≤ i
      j = j + 1
      s = s + j
```

```
Algorithm loop7(n)
  for i = 1 to n
    j = 1
    while j ≤ i
      j = j + 1
```

```
Algorithm loop8(n)
  for i = 1 to n
    j = i
    while j > 0
      j = j - 1
```

```
Algorithm loop9(n)
  i = 0
  j = 0
  while i ≤ n
    if i < j then
      i = i + 1
    else
      j = j + 1
      i = 0
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta(n \log n)$	$\Theta(2^n)$	$\Theta(n^3)$	$\Theta(n^2)$	$\Theta((\log n)^2)$	$\Theta(n)$	$\Theta(\log n)$	$\Theta(n\sqrt{n})$	
loop1									247.1
loop2									247.2
loop3									247.3
loop4									247.4
loop5									247.5
loop6									247.6
loop7									247.7
loop8									247.8
loop9									247.9

Problem 248 check

```
Algorithm loop1(n)
  i = 1
  while i ≤ n
      j = 1
      while j ≤ i
          j = j + 1
      i = i + 1
```

```
Algorithm loop2(n)
  i = 1
  while i ≤ n
      j = 1
      while j ≤ n
          j = j + 1
      i = i + 1
```

```
Algorithm loop3(n)
  i = n
  j = 0
  while i > 0
      if j < i
          j = j + 1
      else
          j = 0
          i = i − 1
```

```
Algorithm loop4(n)
  s = 0
  for i = 1 to n
      for j = 1 to n
          if i = j then
              for k = 1 to n
                  s = s + 1
```

```
Algorithm loop5(n)
  s = 0
  i = n
  while i > 0
      for j = 1 to i
          s = s + 1
      i = i − 1
```

```
Algorithm loop6(n)
  s = 1
  for i = 1 to n
      for j = 1 to n
          s = s + 1
```

```
Algorithm loop7(n)
  s = 1
  for i = 1 to n
      for j = 1 to n
          s = s + 1
      for k = 1 to n
          s = s + 1
```

```
Algorithm loop8(n)
  s = 1
  for i = 1 to n
      for j = i to n
          s = s + 1
```

```
Algorithm loop9(n)
  s = 1
  for i = n to 1 step −1
      for j = n to 1 step −1
          s = s + 1
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta(n \log n)$	$\Theta(n)$	$\Theta(\log n)$	$\Theta(\sqrt{n})$	$\Theta(n^3)$	$\Theta((\log n)^2)$	$\Theta(\sqrt{n} \log n)$	$\Theta(n^2)$	
loop1									248.1
loop2									248.2
loop3									248.3
loop4									248.4
loop5									248.5
loop6									248.6
loop7									248.7
loop8									248.8
loop9									248.9

Problem 249 check

```
Algorithm loop1(n)
  for i = 1 to n
    for j = 1 to i
      k = 1
      while k ≤ i + j
        k = 2 * k
```

```
Algorithm loop2(n)
  s = 0
  for i = 1 to n
    for j = 1 to i * i
      s = s + 1
```

```
Algorithm loop3(n)
  s = 0
  for i = 1 to n
    for j = 1 to n
      for k = 1 to n
        s = s + 1
```

```
Algorithm loop4(n)
  s = 0
  for i = 1 to n
    for j = i to n
      for k = i to j
        s = s + 1
```

```
Algorithm loop5(n)
  s = 0
  j = 0
  for i = 1 to n
    j = j + i
    for k = 1 to j
      s = s + 1
```

```
Algorithm loop6(n)
  s = 1
  for i = 1 to n
    for j = 1 to i
      for k = j to i
        s = s + 1
```

```
Algorithm loop7(n)
  s = 1
  for i = 1 to n * n
    for j = 1 to n
      s = s + 1
```

Indicate for each of the above algorithms the running time as a function of n in Θ-notation.

	$\Theta(n^2 \cdot \log n)$	$\Theta(n^3)$	$\Theta(n \log n)$	$\Theta(\log n)$	$\Theta(n^2)$	$\Theta((\log n)^2)$	$\Theta(n)$	$\Theta(n\sqrt{n})$	
loop1									249.1
loop2									249.2
loop3									249.3
loop4									249.4
loop5									249.5
loop6									249.6
loop7									249.7

Amortized analysis

Problem 250 (4 %) check

Assume a list L is used to store a set of distinct integers. The following two operations are to be supported: Add(x) adds x to the list (where x is assumed to be distinct from all integers already in the list), and RemoveLargerHalf removes and returns the $\lceil |L|/2 \rceil$ largest elements from the list. Add(x) simply appends the new element to the end of the list L in worst-case $O(1)$ time. RemoveLargerHalf first uses deterministic selection to find the $\lfloor |L|/2 \rfloor + 1$ smallest element e in L in worst-case $O(|L|)$ time, after which L is scanned and all elements greater than or equal to e are removed.

For each of the following functions, indicate whether it is a potential function with which one can argue that both operations take amortized $O(1)$ time.

	Yes	No	
$\|L\|$			250.1
$\log \|L\|$			250.2
$2\|L\|$			250.3
$\|L\|/2$			250.4
$\|L\|^2$			250.5
$\|L\| \cdot \log \|L\|$			250.6

Problem 251 (4 %) check

Assume an array X of size n contains two stacks S and T of sizes s and t, respectively, such that $X[1..s] = S$ and $X[n+1-t..n] = T$, where the tops of the two stacks are $X[s]$ and $X[n+1-t]$, respectively. When X becomes full, i.e., $s+t = n$, a new array of double the size is allocated for X, and S and T are copied into this array.

For each of the following functions, indicate whether it is a potential function with which one can argue that the stack operations Push and Pop on the two stacks take amortized $O(1)$ time.

	Yes	No	
$s + t$			251.1
$t - s$			251.2
$s + n - t$			251.3
$\max\{0, 2(s+t) - n\}$			251.4
$n - s - t$			251.5
$\max\{0, 2s + 3t - n\}$			251.6

Problem 252 (4 %) check

Given a sorted list L with $N = 2^k - 1$ elements, for a positive integer k, one can in $O(N)$ time construct a perfectly balanced binary search tree containing L. In the following, we assume that we only perform deletions in the search tree, as described in [CLRS, Chapter 12.3], i.e., deletions do not attempt to keep the tree balanced. Let n denote the current number of elements in the tree. To keep the height of the tree logarithmic in n, the tree is reconstructed as a perfectly balanced binary search tree in $O(n)$ time when half of the elements have been deleted, i.e., when $n < N/2$, where N denotes the number of elements in the tree the last time it was reconstructed.

For each of the following functions, indicate whether it is a potential function with which one can argue that deletions take amortized $O(\log n)$ time.

	Yes	No	
N			252.1
n			252.2
$N - n$			252.3
$(N - n) \cdot \log n$			252.4
$\log n$			252.5
$(N - n) \cdot \log N$			252.6

Problem 253 (4 %) check

Consider a list $L = (x_1, \ldots, x_N)$ of N integers, on which we can perform the following two operations: Append(x) appends the integer x to the end of the list, and AddPairs replaces for all $i = 1..\lfloor |L|/2 \rfloor$ the $2i-1$'th and $2i$'th integers with their sum, such that the new list has length $\lceil |L|/2 \rceil$. For example, AddPairs on the list $3, 5, 7, 4, 11, 2, 6$ results in the new list $8, 11, 13, 6$. The worst-case time for Append and AddPairs is $O(1)$ and $O(|L|)$, respectively.

For each of the following functions, indicate whether it is a potential function with which one can argue that both operations take amortized $O(1)$ time.

	Yes	No	
N			253.1
$\log N$			253.2
$N + \log N$			253.3
$N/2$			253.4
$N \cdot \log N$			253.5
$\sqrt{N}$			253.6

Problem 254 (4 %) check

Consider an unordered list of n integers, where we can in $O(1)$ time insert a new integer, and in $O(n)$ time perform the operation NegateRemove, which removes all non-positive integers ($x \leq 0$) from the list and replaces each positive integer $y > 0$ with the corresponding negative integer $-y$. NegateRemove$(3, -4, -2, 7, 6, -2) = (-3, -7, -6)$.

For each of the following functions, indicate whether it is a potential function with which one can argue that both insertions and NegateRemove take amortized $O(1)$ time, where P is the number of positive integers in the list and N is the number of non-positive integers in the list.

	Yes	No	
N			254.1
P			254.2
$2N + P$			254.3
$N + P$			254.4
$N + 2P$			254.5
$P + N/2$			254.6

Problem 255 (4 %) check

Consider a red-black search tree extended with the operation InsertCut(x), which inserts the element x into the search tree and deletes all elements less than x from the search tree. If the insertion deletes k elements, this takes worst-case $O((k+1)\log n)$ time, where n is the number of elements in the tree before InsertCut is performed.

For each of the following functions, indicate whether it is a potential function with which one can argue that InsertCut takes amortized $O(\log n)$.

	Yes	No	
$\log n$			255.1
$n \cdot \log n$			255.2
n			255.3
$\sum_{i=1}^{n} \log i$			255.4
$k \cdot \log n$			255.5

Problem 256 (4 %) check

Assume we want to store a set of n integers using hashing with linear probing in an array of size N. We guarantee that the array is always between 1/4 and 3/4 full. If there are fewer than $N/4$ or more than $3N/4$ integers in the set, we reinsert all integers into a new array of size $2n$, i.e., the new array is 1/2 full.

For each of the following functions, indicate whether it is a potential function with which one can argue that the total number of reinsertions in the hash tables is amortized $O(1)$ per insertion and deletion in the set.

	Yes	No	
$\lvert 2n - N \rvert$			256.1
$\min(n, N - n)$			256.2
$N - n$			256.3
$4\lvert N/2 - n \rvert$			256.4
N			256.5
$N/2 - \min(n, N - n)$			256.6

Problem 257 (4 %) check

A binary max-heap supports Insert and Heap-Extract-Max on a max-heap with n elements in worst-case $O(\log n)$ time.

For each of the following functions, indicate whether it is a potential function with which one can argue that Insert takes amortized $O(\log n)$ time and Heap-Extract-Max takes amortized $O(1)$ time.

	Yes	No	
$n \cdot \log n$			257.1
$\sum_{i=1}^{n} \log i$			257.2
n			257.3
$\log n$			257.4
$(\log n)^2$			257.5

Problem 258 (4 %) check

Consider a binary max-heap implemented in an array. Overflow is handled by allocating a new array of double size and copying the contents of the old array to the new array. Let the current size of the array be N and the number of elements in the heap be n.

For each of the following functions, indicate whether it is a potential function with which one can argue that INSERT and HEAP-EXTRACT-MAX require amortized $O(\log n)$ time.

	Yes	No
n		258.1
$\lvert 2n - N \rvert$		258.2
$\max(0, 2n - N)$		258.3
$N - 2n$		258.4
$\log n$		258.5
$\log(N/n)$		258.6

Problem 259 (4 %) check

Consider a stack implemented in an array, where overflow is handled by allocating a new larger array and copying the contents of the old array to the new array. Indicate the amortized time for a PUSH operation, when the new array has the following size N and n is the number of elements on the stack before the PUSH operation.

	$O(1)$	$O(\log n)$	$O(\sqrt{n})$	$O(n)$
$N = n + 1$				259.1
$N = n + \lceil \sqrt{n} \rceil$				259.2
$N = \lceil \frac{3}{2} n \rceil$				259.3
$N = 2n$				259.4
$N = 3n$				259.5

Problem 260 (4 %) check

A binary max-heap supports Max-Heap-Insert and Heap-Extract-Max in worst-case time $O(\log n)$, where n is the number of elements in the heap. We now want to support the operation Delete, which given a pointer to an element in the heap, deletes the element from the heap. We implement Delete by simply marking the element as deleted in worst-case $O(1)$ time. When we perform Heap-Extract-Max, we repeat this until the first unmarked element is returned. I.e., if Heap-Extract-Max deletes D marked elements, the worst-case time is $O((D+1)\log N)$, where N is the number of marked and unmarked elements in the heap.

For each of the following functions, indicate whether it is a potential function with which one can argue that the operations Max-Heap-Insert, Delete, and Heap-Extract-Max take amortized $O(\log N)$ time, where M denotes the number of marked elements in the heap.

	Yes	No	
N			260.1
M			260.2
$N \cdot \log N$			260.3
$M \cdot \log N$			260.4
$N \cdot \log M$			260.5
$M \cdot \log M$			260.6

Problem 261 (4 %) check

A binary max-heap supports Insert and Heap-Extract-Max on a heap with n elements in worst-case $O(\log n)$ time. Note that the same value can be inserted multiple times in a max-heap. We now want to change Heap-Extract-Max, such that it removes all occurrences of the maximum value from the heap, i.e., the original Heap-Extract-Max operation is repeated until the root contains a smaller value or the heap is empty. If the maximum occurs m times in the heap, then the Heap-Extract-Max operation will take worst-case $O(m \log n)$ time.

For each of the following functions, indicate whether it is a potential function with which one can argue that the operations Insert and Heap-Extract-Max take amortized $O(\log n)$ time. The number of elements in the heap is denoted n and the number of distinct elements in the heap N, where $N \leq n$.

	Yes	No	
n			261.1
N			261.2
$n - N$			261.3
$n \cdot \log n$			261.4
$N \cdot \log n$			261.5
$(n - N) \cdot \log n$			261.6

Invariants

Problem 262 (4 %) check

Given two non-negative integers n and m, the below algorithm computes $n \cdot m$.

Algorithm Multiplication(n, m)
Input : $n \geq 0 \ \wedge \ m \geq 0$
Output : $r = n_0 \cdot m_0$
Method : $r \leftarrow 0$
$\{I\}$ **while** $n > 0$ **do**
if n is odd **then**
$r \leftarrow r + m$
$n \leftarrow n - 1$
else
$m \leftarrow m * 2$
$n \leftarrow n/2$

For each of the following statements, indicate whether it is an invariant I for the algorithm Multiplication, where n_0 and m_0 denote the initial values of n and m, respectively.

	Yes	No	
$0 \leq n \leq n_0$			262.1
$0 \leq m \leq m_0$			262.2
$r = m \cdot n$			262.3
$n_0 \cdot m_0 = r + n \cdot m$			262.4
$n_0 \cdot m_0 + r = n \cdot m$			262.5

Problem 263 (4 %) check

Given positive integers x and y, the below algorithm computes x^y.

Algorithm Power(x, y)
Input : $x \geq 1 \ \wedge \ y \geq 1$
Output : $r = x^y$
Method : $r \leftarrow 1$
$\{I\}$ **while** $y \geq 1$ **do**
if y is odd **then**
$y \leftarrow y - 1$
$r \leftarrow r * x$
else
$y \leftarrow y/2$
$x \leftarrow x * x$

For each of the following statements, indicate whether it is an invariant I for the algorithm Power, where x_0 and y_0 denote the initial values of x and y.

	Yes	No	
$r = x^y$			263.1
$r = x_0^{y_0}$			263.2
$r = x_0^{y_0 - y}$			263.3
$x_0^{y_0} = r \cdot x^y$			263.4
$x^y = r \cdot x_0^{y_0}$			263.5

Problem 264 (4 %) check

Given a positive integer n, the below algorithm computes n^2.

Algorithm Square(n)
Input : $n \geq 1$
Output : $r = n^2$
Method : $r \leftarrow 0$
$i \leftarrow 0$
$\{I\}$ **while** $i < n$ **do**
$i \leftarrow i + 1$
$r \leftarrow r + i$
$r \leftarrow 2 * r - n$

For each of the following statements, indicate whether it is an invariant I for the algorithm Square.

	Yes	No	
$i \geq 0 \ \wedge \ r = i^2$			264.1
$0 \leq i \leq n$			264.2
$i \geq 0 \ \wedge \ r = i(i+1)/2$			264.3
$r \geq i \geq 0$			264.4
$0 \leq r \leq n$			264.5

Problem 265 (4 %) check

Given an array $A[1..n]$ containing $n \geq 1$ integers and an integer x, the below algorithm computes the number of occurrences of x in A, i.e., $count(x, A) = |\{i \mid 1 \leq i \leq n \wedge A[i] = x\}|$.

Algorithm Count(x, A)
Input : $x \wedge A[1..n] \wedge n \geq 1$
Output : $r = count(x, A)$
Method : $i \leftarrow 0$
$r \leftarrow 0$
$\{I\}$ **while** $i < n$ **do**
$i \leftarrow i + 1$
if $x = A[i]$ **then**
$r \leftarrow r + 1$

For each of the following statements, indicate whether it is an invariant I for the above algorithm Count.

	Yes	No	
$i \leq n$			265.1
$r = count(x, A[1..i])$			265.2
$r = count(x, A[1..i+1])$			265.3
$i < n$			265.4
$r = 0$			265.5

Problem 266 (4 %) check

Assume that a sorted array $A[1..n]$ contains $n \geq 2$ different integers, i.e., $A[1] < A[2] < \cdots < A[n-1] < A[n]$. Given a positive integer $x > 0$, the below algorithm identifies whether there exist $1 \leq i < j \leq n$ such that $x = A[j] - A[i]$.

Algorithm FindDiff($A[1..n], x$)
Input : $x > 0 \ \wedge \ A[1..n] \ \wedge \ n \geq 2 \ \wedge \ A[1] < \cdots < A[n]$
Output : $1 \leq i < j \leq n$, where $x = A[j] - A[i]$;
or $j = n + 1$ if no such pair exists
Method : $i \leftarrow 1$;
$j \leftarrow 1$;
$\{I\}$ **while** $j \leq n$ **and** $x \neq A[j] - A[i]$ **do**
if $x > A[j] - A[i]$ **then**
$j \leftarrow j + 1$
else
$i \leftarrow i + 1$

For each of the following statements, indicate whether it is an invariant I for the algorithm FindDiff.

	Yes	No	
$i \leq j$			266.1
$1 \leq i \leq j \leq n$			266.2
$x \neq A[j] - A[i]$			266.3
$\forall\ 1 \leq i' < j' \leq n : (i' \geq i \ \vee \ j' \geq j) \Rightarrow x \neq A[j'] - A[i']$			266.4
$\forall\ 1 \leq i' < j' \leq n : (i' < i \ \vee \ j' < j) \Rightarrow x \neq A[j'] - A[i']$			266.5

Problem 267 (4 %) check

Let $\|x\|$ denote the number of bits with value 1 in the binary representation of a non-negative integer x, e.g., $\|14_{10}\| = \|1110_2\| = 3$. The below algorithm computes $\|x\|$.

Algorithm Bits(x)
Input : $x \geq 0$
Output : $r = \|x\|$
Method : $r \leftarrow 0$;
$\{I\}$ **while** $x > 0$ **do**
if x is odd **then**
$x \leftarrow x - 1$;
$r \leftarrow r + 1$
$x \leftarrow x/2$

For each of the following statements, indicate whether it is an invariant I for the algorithm Bits, where x_0 denotes the initial value of x.

	Yes	No	
$r = \|x\|$			267.1
$r + \|x\| = \|x_0\|$			267.2
$r + \|x_0\| = \|x\|$			267.3
$r + \|x\| \leq x_0$			267.4
$\|x\| + 2^r = x_0$			267.5

Problem 268 (4 %) check

The below algorithm computes the integer base-2 logarithm of n, i.e., $\text{intlog}(n) = \lfloor \log_2 n \rfloor$.

Algorithm Log2(n)
Input : $n \geq 2$
Output : $r = \text{intlog}(n) = \lfloor \log_2 n \rfloor$
Method : $i \leftarrow 1$;
$r \leftarrow 1$;
$p \leftarrow 2$;
$\{I\}$ **while** $2p \leq n$ **do**
if $p * p \leq n$ **then**
$p \leftarrow p * p$;
$r \leftarrow 2 * r$
else
$p \leftarrow 2 * p$;
$r \leftarrow r + 1$

For each of the following statements, indicate whether it is an invariant I for the algorithm Log2.

	Yes	No	
$1 \leq r < p$			268.1
$2p \leq n$			268.2
$p = 2^r$			268.3
$p = 2r$			268.4
$p = 2^{\text{intlog}(p)}$			268.5

Problem 269 (4 %) check

Assume that an array $A[1..n-1]$ contains $n-1$ different numbers from the set $\{1,2,3,\ldots,n\}$, where $n \geq 2$. The below algorithm Missing identifies the element $r \in \{1,2,3,\ldots,n\}$ that is not in A.

Algorithm Missing($A[1..n-1]$)
Input : $A[1..n-1] \wedge n \geq 2 \wedge A[i] \in \{1,2,\ldots,n\} \wedge A[i] \neq A[j]$ for $1 \leq i < j < n$
Output : $r \in \{1,2,3,\ldots,n\} \setminus A$
Method : $i \leftarrow 1$;
$x \leftarrow 1$;
$y \leftarrow A[1]$;
$\{I\}$ **while** $i < n-1$ **do**
$i \leftarrow i+1$;
$x \leftarrow x+i$;
$y \leftarrow y+A[i]$
$r \leftarrow n+x-y$

For each of the following statements, indicate whether it is an invariant I for the algorithm Missing.

	Yes	No	
$1 \leq i \leq n-1$			269.1
$y = x - i + A[i]$			269.2
$x = i(i+1)/2$			269.3
$y = \sum_{j=1}^{i} A[j]$			269.4
$i = i+1$			269.5

Problem 270 (4 %) check

Assume that an array $A[1..n]$ contains $n \geq 1$ integers. Position i in A is said to be *dominating* if $A[i] > A[j]$ for all $1 \leq j < i$. Let $\mathrm{dom}(A)$ denote the number of dominating positions i in A, where $1 \leq i \leq n$. The below algorithm computes $\mathrm{dom}(A)$.

Algorithm Domination(A)
Input : Array $A[1..n] \ \wedge \ n \geq 1$
Output : $r = \mathrm{dom}(A)$
Method : $i \leftarrow 1$;
$x \leftarrow A[1]$;
$r \leftarrow 1$;
$\{I\}$ **while** $i < n$ **do**
$i \leftarrow i + 1$;
if $A[i] > x$ **then**
$x \leftarrow A[i]$;
$r \leftarrow r + 1$

For each of the following statements, indicate whether it is an invariant I for the algorithm.

	Yes	No	
$1 \leq i < n$			270.1
$x = A[i]$			270.2
$x \geq A[i]$			270.3
$r = \mathrm{dom}(A[1..n])$			270.4
$r = \mathrm{dom}(A[1..i])$			270.5

Problem 271 (4 %) check

Assume $A[1..n]$ is a sorted array with n different positive integers. Let squares(A) denote the number of $A[i]$ such that $A[i]^2$ also occurs in A. For example, squares$(1, 3, 4, 7, 9, 16) = 3$, since $1^2 = 1$, $3^2 = 9$ and $4^2 = 16$. The below algorithm Squares computes squares(A).

Algorithm Squares($A[1..n]$)
Input : Array $A[1..n] \ \wedge \ 0 < A[1] < A[2] < \cdots < A[n]$
Output : $r = \text{squares}(A)$
Method : $i \leftarrow 1$;
$j \leftarrow 1$;
$r \leftarrow 0$;
$\{I\}$ **while** $j \leq n$ **do**
if $A[i]^2 < A[j]$ **then**
$i \leftarrow i + 1$
else if $A[i]^2 = A[j]$ **then**
$i \leftarrow i + 1$; $j \leftarrow j + 1$; $r \leftarrow r + 1$
else if $A[i]^2 > A[j]$ **then**
$j \leftarrow j + 1$

For each of the following statements, indicate whether it is an invariant I for the above algorithm Squares. It is assumed that $A[0] = 0$ and $A[n + 1] = +\infty$.

	Yes	No	
$1 \leq i \leq j \leq n$			271.1
$i \leq j$			271.2
$A[i-1]^2 < A[j] \ \wedge \ r = \text{squares}(A[1..j])$			271.3
$A[i-1]^2 < A[j] \ \wedge \ r = \text{squares}(A[1..j-1])$			271.4
$r = j - 1$			271.5

Problem 272 (4 %) check

Given a positive integer n, the below algorithm computes n^3.

Algorithm Power3(n)
Input : $n \geq 1$
Output : $r = n^3$
Method : $i \leftarrow 1$
$s \leftarrow 1$
$r \leftarrow 1$
$\{I\}$ **while** $i < n$ **do**
$i \leftarrow i + 1$
$s \leftarrow s + 2i - 1$
$r \leftarrow r + 3s - 3i + 1$

For each of the following statements, indicate whether it is an invariant I for the algorithm Power3.

	Yes	No	
$1 \leq i < n$			272.1
$1 \leq i \leq s \leq r$			272.2
$r = i^3$			272.3
$r = n^3$			272.4
$s = s + 2i - 1$			272.5

Problem 273 (4 %) check

Given a non-negative integer x and a positive integer y, the below algorithm computes $\lfloor x/y \rfloor$.

Algorithm Division(x, y)
Input : $x \geq 0 \ \wedge \ y \geq 1$
Output : $r = \lfloor x/y \rfloor$
Method : $r \leftarrow 0$
$\{I\}$ **while** $x \geq y$ **do**
$x \leftarrow x - y$
$r \leftarrow r + 1$

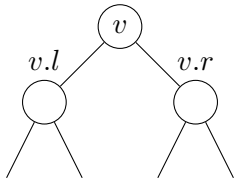
For each of the following statements, indicate whether it is an invariant I for the algorithm Division, where x_0 and y_0 denote the initial values of x and y, respectively.

	Yes	No	
$r = \lfloor x/y \rfloor$			273.1
$r = \lfloor x_0/y_0 \rfloor$			273.2
$r = \lfloor (x_0 - x)/y \rfloor$			273.3
$x + ry = x_0$			273.4
$r(x - x_0) = y$			273.5

Augmented search trees

Problem 274 (4 %) check

Consider a red-black tree where each node stores a pair of integers $(element, weight)$, and the pairs are sorted from left-to-right according to increasing *element* value. For a node v in the tree we let $v.e$ and $v.w$ denote the pair (e, w) stored in the node. Furthermore, v stores the value $v.W$ which is the sum of the weights in all nodes in v's subtree, and $v.prefix$ which is the maximum sum of the weights a *prefix* of the pairs in v's subtree can have (when the pairs are sorted according to element value). Indicate how $v.prefix$ can be computed when the corresponding information is known for the two children $v.l$ and $v.r$ (it can be assumed that both exist).



$v.prefix = \max(v.l.prefix, v.l.W + v.w, v.l.prefix + v.w + v.r.prefix)$

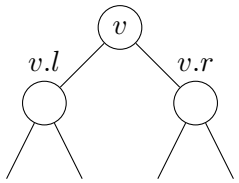
$v.prefix = \max(v.l.prefix, v.l.W + v.w, v.l.W + v.w + v.r.prefix)$

$v.prefix = \max(v.l.prefix, v.W + v.r.prefix)$

$v.prefix = \max(v.l.W, v.l.W + v.w, v.l.W + v.w + v.r.W)$

Problem 275 (4 %) check

An interval tree is a red-black tree where each node stores exactly one interval $(low, high)$, and the intervals are sorted from left-to-right according to increasing *low* value. For a node v in the tree we let $v.low$ and $v.high$ denote the endpoints of the interval stored in the node. Furthermore, v stores the value $v.max$ which is the maximum value in an interval stored in v's subtree. Indicate how $v.max$ can be computed when the corresponding information is known for the two children $v.l$ and $v.r$ (it can be assumed that both exist).



$v.max = v.r.max$

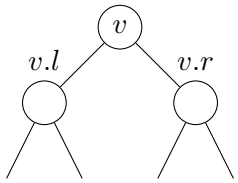
$v.max = \max(v.r.max, v.high)$

$v.max = \max(v.r.max, v.l.max, v.high)$

$v.max = \max(v.l.high, v.high, v.r.high)$

Problem 276 (4 %) **check**

Consider a search tree where each node v stores a number $v.x$, and the nodes are ordered left-to-right according to increasing $v.x$. Furthermore, a node v stores two values $v.size$ and $v.avg$, which are respectively the number of elements in v's subtree and the average of all numbers in v's subtree. Indicate how $v.size$ and $v.avg$ can be computed when the corresponding information is known for the two children $v.l$ and $v.r$ (it can be assumed that both exist).



$$v.size = v.size + v.l.size + v.r.size$$

$$v.size = 1 + v.l.size + v.r.size$$

$$v.size = v.l.size + v.r.size$$

$$v.size = 1 + (v.r.size - v.l.size)$$

$$v.avg = (v.x + v.l.avg + v.r.avg)/3$$

$$v.avg = (v.x + v.l.avg * v.l.size + v.r.avg * v.r.size)/v.size$$

$$v.avg = (v.x + v.l.avg + v.r.avg)/v.size$$

$$v.avg = v.x + (v.l.avg + v.r.avg)/2$$

Problem 277 (4 %) **check**

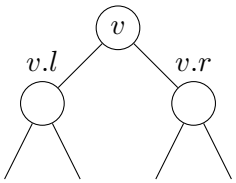
Consider a list of n points $(x_1, y_1), \ldots, (x_n, y_n)$, where x_i and y_i are real numbers and $x_1 < x_2 < \cdots < x_n$. We want to find a contiguous sequence of points

$$(x_i, y_i), (x_{i+1}, y_{i+1}), \ldots, (x_{j-1}, y_{j-1}), (x_j, y_j),$$

where $i \leq j$, such that $Y_{i,j} = \sum_{k=i}^{j} y_k = y_i + y_{i+1} + \cdots + y_{j-1} + y_j$ is as large as possible. We want for a dynamic list of points to maintain this maximum contiguous y-sum, *maxysum*.

Consider a search tree where each node v stores a point $(v.x, v.y)$, and the nodes are ordered left-to-right according to increasing $v.x$. Furthermore, a node v, which in its subtree contains the points $(x_k, y_k), \ldots, (x_\ell, y_\ell)$, stores four values $v.sum = Y_{k,\ell}$, $v.maxysum = \max_{k \leq i \leq j \leq \ell} Y_{i,j}$, $v.pre = \max\{0, \max_{k \leq j \leq \ell} Y_{k,j}\}$, and $v.suf = \max\{0, \max_{k \leq i \leq \ell} Y_{i,\ell}\}$.

Indicate how $v.sum$ and $v.maxysum$ can be computed when the corresponding information (incl. *pre* and *suf*) is known for the two children $v.l$ and $v.r$ (it can be assumed that both exist).



$$v.sum = v.l.sum + v.r.sum$$

$$v.sum = v.l.sum + v.y + v.r.sum$$

$$v.sum = v.l.suf + v.r.pre$$

$$v.sum = v.l.suf + v.y + v.r.pre$$

$$v.maxysum = \max\{v.l.maxysum, v.r.maxysum\}$$

$$v.maxysum = \max\{v.l.maxysum, v.y, v.r.maxysum\}$$

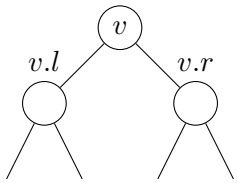
$$v.maxysum = \max\{v.l.maxysum, v.l.suf + v.y + v.r.pre, v.r.maxysum\}$$

$$v.maxysum = \max\{v.l.maxysum, v.l.suf + v.r.pre, v.r.maxysum\}$$

Problem 278 (4 %) **check**

Consider a list of n points $(x_1, y_1), \ldots, (x_n, y_n)$, where x_i and y_i are real numbers and $x_1 < x_2 < \cdots < x_n$. A point (x_i, y_i) is dominated by a point (x_j, y_j) if $x_i \leq x_j$ and $y_i \leq y_j$. A list of points is *dominance-free* if and only if no point in the list is dominated by another point. For example, the list $(3, 7), (5, 3), (7, 6), (13, 4)$ is *not* dominance-free, since $(5, 3)$ is dominated by $(13, 4)$. We want for a dynamic list of points to maintain the statement of whether the list is dominance-free.

Consider a search tree where each node v stores a point $(v.x, v.y)$, and the nodes are ordered left-to-right according to increasing $v.x$. Furthermore, v also stores $v.DF$ and $v.miny$ and $v.maxy$, where $v.DF$ is true if and only if the set of points in v's subtree is dominance-free, and $v.miny$ and $v.maxy$ are the smallest and largest y value stored in v's subtree. Indicate how $v.miny$ and $v.DF$ can be computed when the corresponding information (incl. *maxy*) is known for the two children $v.l$ and $v.r$ (it can be assumed that both exist).



$$v.miny = v.l.miny$$

$$v.miny = \min\{v.l.miny, v.r.miny\}$$

$$v.miny = \min\{v.l.miny, v.y, v.r.miny\}$$

$$v.DF = v.l.DF \wedge v.r.DF\text{r}$$

$$v.DF = v.l.DF \wedge v.r.DF \wedge v.y < v.l.miny \wedge v.y > v.r.maxy$$

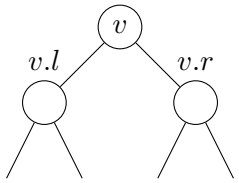
$$v.DF = v.l.DF \wedge v.r.DF \wedge v.y > v.l.miny \wedge v.y < v.r.maxy$$

$$v.DF = v.l.DF \wedge v.r.DF \wedge v.y < v.l.miny$$

Problem 279 (4 %) **check**

Consider a list of n pairs $(x_1, c_1), \ldots, (x_n, c_n)$, where $x_1 < x_2 < \cdots < x_n$ are real numbers and each $c_i \in \{\text{Red}, \text{Green}, \text{Blue}\}$ is a color.

Consider a search tree where each node v stores a pair $(v.x, v.c)$, and the nodes are ordered left-to-right according to increasing $v.x$. Furthermore, a node v stores a boolean value $v.mono$, which indicates whether all elements in v's subtree have the same color, and the set $v.missing$ of the colors that *do not* occur in v's subtree. Indicate how $v.mono$ and $v.missing$ can be computed when the corresponding information is known for the two children $v.l$ and $v.r$ (it can be assumed that both exist).



$$v.mono = v.l.mono \ \wedge \ v.r.mono$$

$$v.mono = v.l.mono \ \wedge \ v.r.mono \ \wedge \ v.l.c = v.r.c \ \wedge \ v.c = v.r.c$$

$$v.mono = v.l.mono \ \wedge \ v.r.mono \ \wedge \ v.l.c = v.r.c$$

$$v.mono = v.l.mono \ \wedge \ v.r.mono \ \wedge \ v.mono$$

$$v.missing = v.l.missing \ \cup \ v.r.missing$$

$$v.missing = v.l.missing \ \cap \ v.r.missing$$

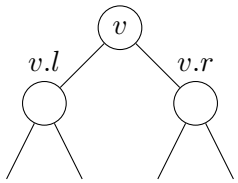
$$v.missing = \{v.c\} \cup v.l.missing \ \cup \ v.r.missing$$

$$v.missing = (v.l.missing \ \cap \ v.r.missing) \setminus \{v.c\}$$

Problem 280 (4 %) **check**

Given a string T containing letters and opening and closing parentheses `(` and `)`, we assume that all positions with parentheses are stored in a search tree, sorted from left-to-right according to increasing position. A node v stores a position $v.p$ and the corresponding parenthesis $v.c = T[v.p]$ from T. For $T =$"`a)b(cd(x)dc(a`" the $\langle v.p, v.c\rangle$ pairs $\langle 2,)\rangle, \langle 4, (\rangle, \langle 7, (\rangle, \langle 9,)\rangle$ and $\langle 12, (\rangle$ are stored in the tree.

We want to maintain information about whether the parentheses are balanced. In the above example the parentheses "`) ( ( ) (`" are not balanced, since only the marked parentheses match each other. The remaining parentheses "`) ( (`" will always be R `)`-parentheses followed by L `(`-parentheses, where $R \geq 0$ and $L \geq 0$. In the example we have $R = 1$ and $L = 2$. In a node v in the tree these values $v.R$ and $v.L$ are stored for the subsequence of parentheses in v's subtree. Indicate how $v.R$ can be computed when $v.c =$ `)` and the R and L values are known for the two children $v.l$ and $v.r$ (it can be assumed that both exist).



$$v.R = v.l.R + 1 + v.r.R$$

$$v.R = v.l.R + v.r.R + 1 - v.l.L$$

$$v.R = v.l.R + \max\{0, v.r.R + 1 - v.l.L\}$$

$$v.R = v.l.R + 1 + \max\{0, v.r.R - v.l.L\}$$

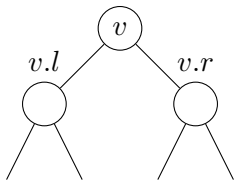
Problem 281 (4 %) **check**

For n numbers $x_1, \ldots, x_n$ we want to compute the coefficients a and b for the polynomial

$$P(y) = \sum_{i=1}^{n} (x_i - y)^2 = n \cdot y^2 + a \cdot y + b \; .$$

For example, for $x_1 = 2$, $x_2 = 3$, and $x_3 = 5$, we have the polynomial $P(y) = (2-y)^2 + (3-y)^2 + (5-y)^2 = 3y^2 - 20y + 38$, i.e., $a = -20$ and $b = 38$.

Consider a search tree where each node v stores a number $v.x$, and the nodes are ordered left-to-right according to increasing $v.x$. Furthermore, a node v stores two values $v.a$ and $v.b$, which are the a and b coefficients for the $P(y)$ polynomial defined by all x values in v's subtree. Indicate how $v.a$ and $v.b$ can be computed when the corresponding information is known for the two children $v.l$ and $v.r$ (it can be assumed that both exist).



$v.a = v.l.a + v.r.a + v.x$

$v.a = v.l.a + v.r.a - 2 * v.x$

$v.a = v.l.a * v.r.a * (v.x)^2$

$v.a = v.l.a + v.r.a + (v.x)^2$

$v.b = v.l.b + v.r.b + v.x$

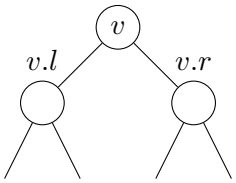
$v.b = v.l.b + v.r.b + (v.x)^2$

$v.b = v.l.b + v.r.b - 2 * v.x$

$v.b = v.l.b + v.r.b + 2 * v.x$

Problem 282 (4 %) **check**

Consider a search tree where each node v stores a number $v.x$, and the nodes are ordered left-to-right according to increasing $v.x$. Furthermore, a node v stores three values $v.min$, $v.max$, and $v.closest$. The values $v.min$ and $v.max$ are respectively the smallest and largest number in v's subtree, and $v.closest$ is the smallest difference between two numbers in v's subtree. If v's subtree contains only one number then $v.closest = +\infty$. Indicate how $v.closest$ can be computed when the *min*, *max*, and *closest* values are known for the two children $v.l$ and $v.r$ (it can be assumed that both exist).



$v.closest = \min(v.l.closest, v.r.x - v.l.x, v.r.closest)$

$v.closest = \min(v.l.closest, v.x - v.l.min, v.r.max - v.x, v.r.closest)$

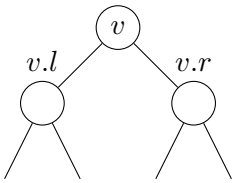
$v.closest = \min(v.l.closest, v.x - v.l.max, v.r.min - v.x, v.r.closest)$

$v.closest = \min(v.l.closest, v.r.min - v.l.max, v.r.closest)$

$v.closest = v.r.closest - v.l.closest$

Problem 283 (4 %) **check**

For a sorted list of numbers $x_1 \leq x_2 \leq \cdots \leq x_n$ we define the *sum of square gaps* (*ssg*) as $\sum_{i=2..n}(x_i - x_{i-1})^2$. Consider a red-black tree where each node v stores an integer $v.x$, and the nodes are ordered left-to-right according to increasing $v.x$. Furthermore, v stores the values $v.min$, $v.max$, and $v.ssg$, which are respectively the smallest, largest, and *sum of square gaps* of the elements in the subtree rooted at v. Indicate how $v.ssg$ can be computed when $v.min$, $v.max$, and $v.ssg$ are known for the two children $v.l$ and $v.r$ (it can be assumed that both exist).



$v.ssg = v.l.ssg + v.r.ssg$

$v.ssg = v.l.ssg + (v.r.min - v.l.max)^2 + v.r.ssg$

$v.ssg = v.l.ssg + (v.x - v.l.max)^2 + (v.x - v.r.min)^2 + v.r.ssg$

$v.ssg = v.l.ssg + (v.r.x - v.l.x)^2 + v.r.ssg$

$v.ssg = v.l.ssg + (v.x - v.l.x)^2 + (v.x - v.r.x)^2 + v.r.ssg$

Miscellaneous questions

Problem 284 (4 %) check

For each of the following algorithms, give the best-case, worst-case, and expected running time on input of size n, where the input may contain identical elements.

$\Theta(\log n)$ $\Theta(n)$ $\Theta(n \log n)$ $\Theta(n^2)$ $\Theta(2^n)$

Worst-case time for Insertion-Sort 284.1

Worst-case time for Merge-Sort 284.2

Worst-case time for Heapsort 284.3

Worst-case time for Quicksort 284.4

Best-case time for Insertion-Sort 284.5

Best-case time for Merge-Sort 284.6

Best-case time for Heapsort 284.7

Best-case time for Quicksort 284.8

Expected time for Insertion-Sort 284.9

Expected time for Merge-Sort 284.10

Expected time for Heapsort 284.11

Expected time for Quicksort 284.12

Problem 285 (4 %) check

Give the worst-case time for HeapSort on an array of n *identical* elements.

$\Theta(\sqrt{n})$ $\Theta(n)$ $\Theta(n \log n)$ $\Theta(n\sqrt{n})$ $\Theta(n^2)$

Problem 286 (4 %) check

Give the number of times the *largest element* in an array of n elements can be compared with other elements in the worst case during the execution of Merge-Sort.

$\Theta(1)$ $\Theta(\log n)$ $\Theta(\sqrt{n})$ $\Theta(n)$ $\Theta(n \log n)$

Problem 287 (4 %) check

Given a sorted array $A[1..n]$ ($A[1] < A[2] < \cdots < A[n]$) and an element x, we want to find the index ℓ such that $A[\ell] \leq x < A[\ell+1]$. It is assumed that $A[1] \leq x < A[n]$. Which of the following algorithms is correct (only lines 2 and 4 vary in the algorithms).

```
ℓ = 1, h = n + 1
while ℓ < h
  m = ⌊(h + ℓ)/2⌋
  if A[m] > x
    ℓ = m
  else
    h = m
```

```
ℓ = 1, h = n + 1
while ℓ + 1 < h
  m = ⌊(h + ℓ)/2⌋
  if A[m] > x
    ℓ = m
  else
    h = m
```

```
ℓ = 1, h = n + 1
while ℓ < h
  m = ⌊(h + ℓ)/2⌋
  if A[m] ≤ x
    ℓ = m
  else
    h = m
```

```
ℓ = 1, h = n + 1
while ℓ + 1 < h
  m = ⌊(h + ℓ)/2⌋
  if A[m] ≤ x
    ℓ = m
  else
    h = m
```

Problem 288 (4 %) check

Strassen's algorithm for multiplication of square $n \times n$ matrices is a divide-and-conquer algorithm. Indicate which recurrence relation describes the execution time of Strassen's algorithm.

$$T(n) \leq 7 \cdot T(n/2) + c \cdot n$$

$$T(n) \leq 7 \cdot T(n/4) + c \cdot n^2$$

$$T(n) \leq 7 \cdot T(n/2) + c \cdot n^2$$

$$T(n) \leq 7 \cdot T(n/2) + c \cdot n^3$$

Problem 289 (4 %) check

Give the worst-case execution time for each of the following algorithms when the input is an array of size n.

	$\Theta(1)$	$\Theta(\log n)$	$\Theta(\sqrt{n})$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$	$\Theta(n^3)$	
Binary-Search								289.1
Insertion-Sort								289.2
Merge-Sort								289.3
Heapsort								289.4
Quicksort								289.5
Partition								289.6

Problem 290 (4 %) check

Given an array of size n containing the numbers $1, 2, \ldots, n$ in ascending order, what is the worst-case time for the following sorting algorithms when applied to the array?

	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$	
Insertion-Sort				290.1
Heapsort				290.2
Merge-Sort				290.3
Quicksort				290.4

Problem 291 (4 %) check

The selection algorithm to find the ith smallest element in an unsorted list in worst-case time $O(n)$ [CLRS, Chapter 9.3], divides the input into groups of 5 elements, recursively finds the median of the groups' medians, uses the found element as a pivot for a partition, and recursively calls on one of the two parts. The execution time can be described by the following recurrence relation (ignoring rounding):

$$T(n) = T(n/5) + T(7n/10) + c \cdot n$$

What will the execution time be if you change the groups in the algorithm to have size 3 instead of 5?

$\Theta(\log n)$ $\Theta(n)$ $\Theta(n \log n)$ $\Theta(n^2)$

Problem 292 (4 %) check

For a binary max-heap of size n, give the best-case and worst-case execution time for the following operations.

	$\Theta(1)$	$\Theta(\log n)$	$\Theta(n)$	$\Theta(n \log n)$	
INSERT, worst-case					292.1
INSERT, best-case					292.2
HEAP-EXTRACT-MAX, worst-case					292.3
HEAP-EXTRACT-MAX, best-case					292.4
BUILD-MAX-HEAP, worst-case					292.5
BUILD-MAX-HEAP, best-case					292.6

Problem 293 (4 %) check

For QUICKSORT on input of size n, and a given element e in the input, how many comparisons will this element e be involved in during the execution of QuickSort? The expected number of comparisons here is the expected number of comparisons for a random permutation of the input.

	$\Theta(1)$	$\Theta(\log n)$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$	
Worst-case number of comparisons						293.1
Best-case number of comparisons						293.2
Expected number of comparisons						293.3

Problem 294 (4 %) check

Which of the following statements are true for all binary search trees with n elements.

	Yes	No	
The element in a leaf is always $\leq$ the element in the root			294.1
Elements at the same depth in the tree are sorted from left to right			294.2
Longest root-to-leaf path contains $O(\log n)$ nodes			294.3
The right child of a node with rank r has rank $r+1$			294.4
Along the path from the root to the largest element, the elements are increasing			294.5

Problem 295 (4 %) check

Consider an $n \times n$ matrix M of integers, where all rows and columns are sorted, i.e., $M[i,j] \leq M[i',j']$ for all $1 \leq i \leq i' \leq n$ and $1 \leq j \leq j' \leq n$.

What is the best worst-case time one can achieve to search for an integer in M?

$O(\log n)$ $O(n)$ $O(n \log n)$ $O(n^2)$ $O(n\sqrt{n})$ $O(\sqrt{n})$ $O((\log n)^2)$

Problem 296 (4 %) check

Consider a queue implemented in a circular array Q, where $Q.head$ indicates the head of the queue and $Q.tail$ is the next free space in the queue (as described in [CLRS]).

Which of the following expressions correctly computes the size of the queue?

	Yes	No	
$Q.tail - Q.head$			296.1
$\lvert Q.tail - Q.head \rvert$			296.2
$\lvert Q.tail - Q.head \rvert \bmod Q.length$			296.3
$(Q.tail - Q.head + Q.length - 1) \bmod Q.length$			296.4
$(Q.tail - Q.head + Q.length) \bmod Q.length$			296.5

Problem 297 (4 %) check

Consider an arbitrary binary search tree, and let k be an element in a leaf of the search tree. Let B be all elements on the path from the root down to k, and let A be all elements to the left of the path, and C all elements to the right of the path. Does the following statement always hold for all $a \in A$, $b \in B$ and $c \in C$?

	Yes	No	
$a \leq b$			297.1
$b \leq c$			297.2
$a \leq c$			297.3

Problem 298 (4 %) check

Given two search trees T_1 and T_2 containing the same n elements, T_1 can be transformed into T_2 by a series of rotations. What is the worst-case number of rotations required to transform a search tree with n elements into another search tree T_2 containing the same n elements?

$$\Theta(\log n) \quad \Theta(n) \quad \Theta(n \log n) \quad \Theta(n^2)$$

Problem 299 (4 %) check

For each of the sums below, give their value in Θ-notation. It is assumed that n is a power of two.

	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$	
$\sum_{i=1}^{n} i = 1 + 2 + 3 + \cdots + n$				299.1
$\sum_{i=0}^{\log n} 2^i = 1 + 2 + 4 + 8 + \cdots + n$				299.2
$\sum_{i=1}^{\log n} i\frac{n}{2^i} = 1\frac{n}{2^1} + 2\frac{n}{2^2} + 3\frac{n}{2^3} + \cdots + \log n\frac{n}{2^{\log n}}$				299.3
$\sum_{i=1}^{n} \log i = \log 1 + \log 2 + \log 3 + \cdots + \log n$				299.4
$\sum_{i=0}^{\log n} \frac{n}{2^i} = n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \cdots + \frac{n}{n}$				299.5

Problem 300 (4 %) check

Consider the variant of the binary max-heap where each node has d children instead of two, where $d \geq 2$ is a parameter. What is the time for Insert and Extract-Max expressed as a function of n and d.

	$\Theta(d)$	$\Theta(\log_2 n)$	$\Theta(\log_d n)$	$\Theta(d \log_2 n)$	$\Theta(d \log_d n)$	
Insert						300.1
Extract-Max						300.2

Problem 301 (4 %) check

Consider a binary max-heap with $n = 2^k - 1$ different elements, for an integer $k \geq 1$.

How many different nodes in the max-heap can the smallest element be located in?

$$1 \quad n \quad k \quad k-1 \quad 2^{k-1} \quad 2^k - 1$$

Problem 302 (4 %) check

How many times is RANDOMIZED-PARTITION called in RANDOMIZED-SELECT on an array of size n?

$\Theta(1)$ $\Theta(\log n)$ $\Theta(\sqrt{n})$ $\Theta(n)$ $\Theta(n \log n)$

Expected number of calls 302.1

Fewest number of calls (best case) 302.2

Most number of calls (worst case) 302.3

Problem 303 (4 %) check

Let v be a node in a red-black search tree, and assume there are n elements in v's left subtree. How many elements can there be at most in v's right subtree, i.e., how unbalanced can a node v be in a red-black search tree?

$\Theta(n)$ $\Theta(n \log n)$ $\Theta(n\sqrt{n})$ $\Theta(n^2)$ $\Theta(2^n)$

Problem 304 (4 %) check

Assume an array $A[1..n]$ represents a binary max-heap containing n elements. How quickly can one construct a search tree (not necessarily balanced) containing the elements $A[1..n]$?

$\Theta(\log n)$ $\Theta(n)$ $\Theta(n \log n)$ $\Theta(n^2)$

Problem 305 (4 %) check

Assume MERGE-SORT is performed on an input of size n and containing two elements x and y. What is the worst-case number of comparisons of x with y during the execution of MERGE-SORT?

$\Theta(1)$ $\Theta(\log n)$ $\Theta(n)$ $\Theta(n \log n)$ $\Theta(n^2)$

Problem 306 (4 %) check

Assume we have a directed graph with n nodes and positively weighted edges, where we continuously add additional edges. We want to maintain a distance table of the shortest distances between all pairs of nodes.

What is the best worst-case time we can achieve to update the distance table, when adding a new edge with positive weight to the graph?

$\Theta(1)$ $\Theta(\sqrt{n})$ $\Theta(n)$ $\Theta(n \cdot \log n)$ $\Theta(n \cdot \sqrt{n})$ $\Theta(n^2)$ $\Theta(n^3)$