International PhD School in
Algorithms for Advanced Processor Architectures - AFAPA

Word RAM Algorithms

Gerth Stølting Brodal

MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS

University of Aarhus

Monday June 9, 2008, IT University of Copenhagen, Denmark

Lecture Material

DONALD E. KNUTH

Stanford University

THE ART OF
COMPUTER PROGRAMMING

VOLUME 4 PRE-FASCICLE 1A

A DRAFT OF SECTION 7.1.3:
BITWISE TRICKS AND TECHNIQUES

Background...

- Computer *word sizes* have increased over time

  (4 bits, 8 bits, 12 bits, 16 bits, 32 bits, 64 bits, 128 bits, ...GPU...)

- What is the power and limitations of *word computations*?

- How can we exploit *word parallellism*?

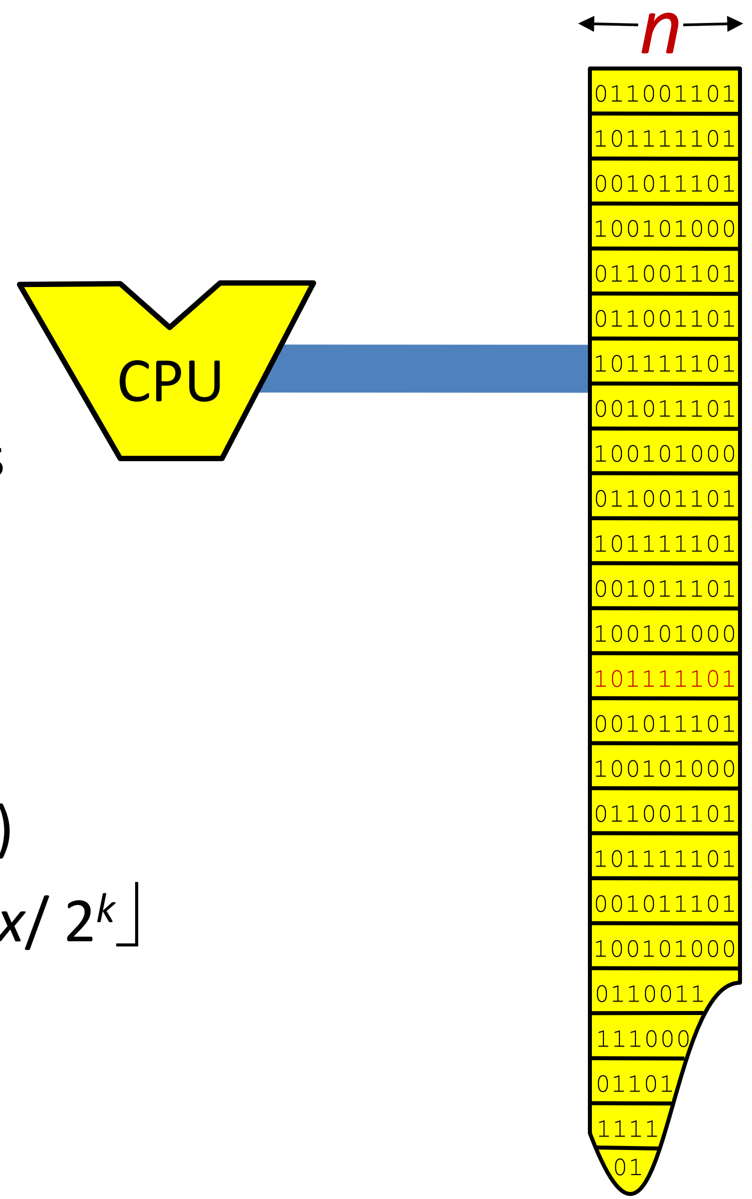
Overview

- Word RAM model
- Words as sets
- Bit-manipulation on words
- Trees
- Searching
- Sorting
- Word RAM results

Word RAM Model

Word RAM (Random Access Machine)

- Unlimited memory
- Word = n bits
- CPU, O(1) registers
- CPU, read & write memory words
  - set[i,v], get[i]
- CPU, computation:
  - Boolean operations
  - Arithmetic operations: +, -, (*)
  - Shifting: $x \ll k = x \cdot 2^k$, $x \gg k = \lfloor x / 2^k \rfloor$
- Operations take O(1) time



Word RAM – Boolean operations

AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

XOR	0	1
0	0	1
1	1	0

x	$\sim x$
0	1
1	0

0 = False, 1 = True

Corresponding word operations work on all n bits in one or two words in parallel.

Example: Clear a set of bits using AND

	0	0	1	1	1	0	1	0	1	1	1	1
AND	0	1	1	1	0	1	1	1	0	1	1	1
	0	0	1	1	0	0	1	0	0	1	1	1

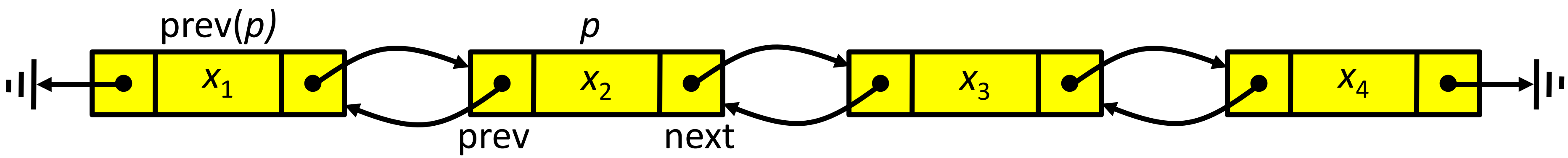
The first tricks…

Exercise 1

Consider a double-linked list, where each node has three fields: prev, next, and an element.

Usually prev and next require one word each.

Question. Describe how prev and next for a node can be combined into *one word*, such that navigation in a double-linked list is still possible.



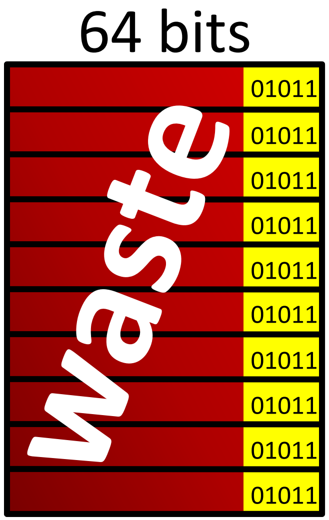
Exercise 2

Question.

How can we pack an array of N 5-bit integers into an array of 64-bit words, such that

a) we only use $\approx N \cdot 5/64$ words, and

b) we can access the i'th 5-bit integer efficiently ?



how not to do it

Words as Sets

Words as Sets

Would like to store *subsets* of {0,1,2,...,*n*-1} in an *n*-bit word.

The set {2,5,7,13} can e.g. be represented by the following word (bit-vector):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	0	0	0	1	0	1	0	0	1	0	0

Exercise 3

Question.

How can we perform the following set operations efficiently, given two words representing S_1 and S_2:

a) $S_3 = S_1 \cap S_2$
b) $S_3 = S_1 \cup S_2$
c) $S_3 = S_1 \setminus S_2$

Exercise 4

Question.

How can we perform the following set queries, given words representing the sets:

a) $x \in S$?
b) $S_1 \subseteq S_2$?
c) Disjoint(S_1, S_2) ?
d) Disjoint(S_1, S_2,..., S_k) ?

Exercise 5

Question.

How can we perform compute $|S|$, given S as a word (i.e. numer of bits = 1)?

a) without using multiplication
b) using multiplication

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	0	0	1	0	0	0	0	0	1	0	1	0	0	1	0	0

$|S| = 4$

Bit-manipulations on Words

Exercise 6

Question.

Describe how to efficiently *reverse* a word *S*.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	0	1	0	0	0	0	0	0	1	0	1	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reverse(S)	0	0	1	0	0	1	0	1	0	0	0	0	0	0	1	0

Exercise 7

Question.

How can we efficiently compute the *zipper*

$$y_{n/2-1}x_{n/2-1}\ldots y_2x_2y_1x_1y_0x_0$$

of two half-words $x_{n/2-1}\ldots x_2x_1x_0$ and $y_{n/2-1}\ldots y_2y_1y_0$?

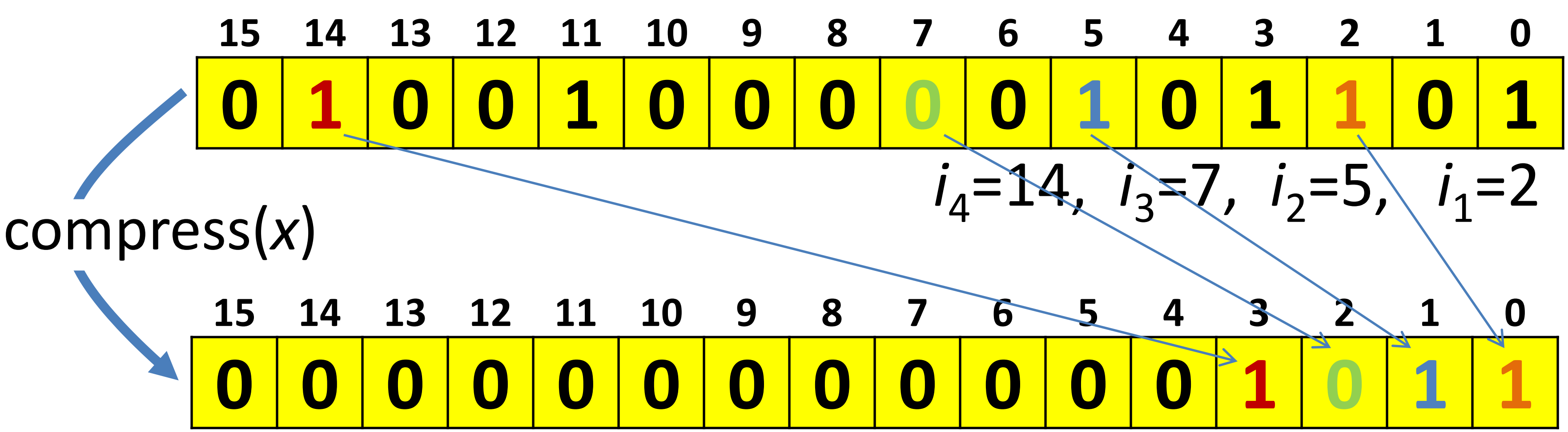
Whitcomb Judson developed the first commercial zipper (named the Clasp Locker) in 1893.

Exercise 8

Question.

Describe how to *compress* a subset of the bits w.r.t. an arbitrary set of bit positions $i_k > \cdots > i_2 > i_1$:

$$\text{compress}(x_{n-1},\ldots,x_2,x_1,x_0) = 0\ldots0x_{i_k}\ldots x_{i_2}x_{i_1}$$



Exercise 9

Question.

a) Describe how to remove the rightmost 1

b) Describe how to extract the rightmost 1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	1	1	0	0	0	1	1	0	1	1	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
remove	0	0	1	1	0	0	0	1	1	0	1	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
extract	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Exercise 10

Question.

Describe how to compute the *position* $\rho(x)$ of the rightmost 1 in a word x

a) without using multiplication
b) using multiplication
c) using integer-to-float conversion

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
x	0	0	1	1	0	0	0	1	1	0	1	1	0	0	0	0

$\rho(x) = 4$

Exercise 11

Let $\lambda(x)$ be the *position* of the leftmost 1 in a word x (i.e. $\lambda(x) = \lfloor \log_2(x) \rfloor$).

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
x	0	0	0	0	1	1	0	1	0	0	1	1	0	1	0	0

$\lambda(x) = 11$

Question.

Describe how to test if $\lambda(x) = \lambda(y)$, without actually computing $\lambda(x)$ and $\lambda(y)$.

Exercise 12*

Question.

Describe how to compute the *position* $\lambda(x)$ of the leftmost 1 in a word x (i.e. $\lambda(x) = \lfloor \log_2(x) \rfloor$)

a) without using multiplication
b) using multiplication
c) using integer-to-float conversion

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
x	0	0	0	0	1	1	0	1	0	0	1	1	0	1	0	0

$\lambda(x) = 11$

Fredman & Willard

Computation of λ(*x*) in O(1) steps using 5 multiplications

$n = g \cdot g$, g a power of 2

$t_1 \leftarrow h \,\&\, (x \mid ((x \mid h) - l)),\quad$ where $h = 2^{g-1} l$ and $l = (2^n - 1)/(2^g - 1)$;
$y \leftarrow (((a \cdot t_1) \bmod 2^n) \gg (n - g)) \cdot l,\quad$ where $a = (2^{n-g} - 1)/(2^{g-1} - 1)$;
$t_2 \leftarrow h \,\&\, (y \mid ((y \mid h) - b)),\quad$ where $b = (2^{n+g} - 1)/(2^{g+1} - 1)$;
$m \leftarrow (t_2 \ll 1) - (t_2 \gg (g - 1)),\ m \leftarrow m \oplus (m \gg g)$;
$z \leftarrow (((l \cdot (x \,\&\, m)) \bmod 2^n) \gg (n - g)) \cdot l$;
$t_3 \leftarrow h \,\&\, (z \mid ((z \mid h) - b))$;
$\lambda \leftarrow ((l \cdot ((t_2 \gg (2g - \lg g - 1)) + (t_3 \gg (2g - 1)))) \bmod 2^n) \gg (n - g)$.

Exercise 13

Question.

Describe how to compute the length of the *longest common prefix* of two words

$$x_{n-1} \ldots x_2 x_1 x_0 \quad \text{and} \quad y_{n-1} \ldots y_2 y_1 y_0$$

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
x	0	0	1	1	0	0	0	1	1	0	1	1	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
y	0	0	1	1	0	0	1	1	0	0	1	0	1	1	0	0

$\mathrm{lcp}(x,y) = 6$

Trees

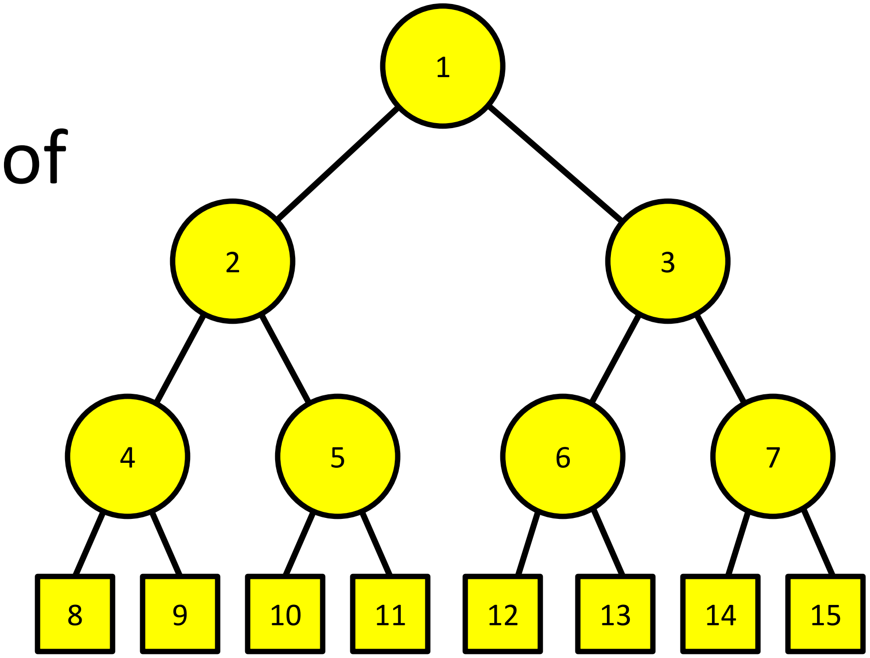
Exercise 14

Question.

Consider the nodes of a complete binary tree being numbered level-by-level and the root being numbered 1.

a) What are the numbers of the children of node *i* ?

b) What is the number of the parent of node *i* ?

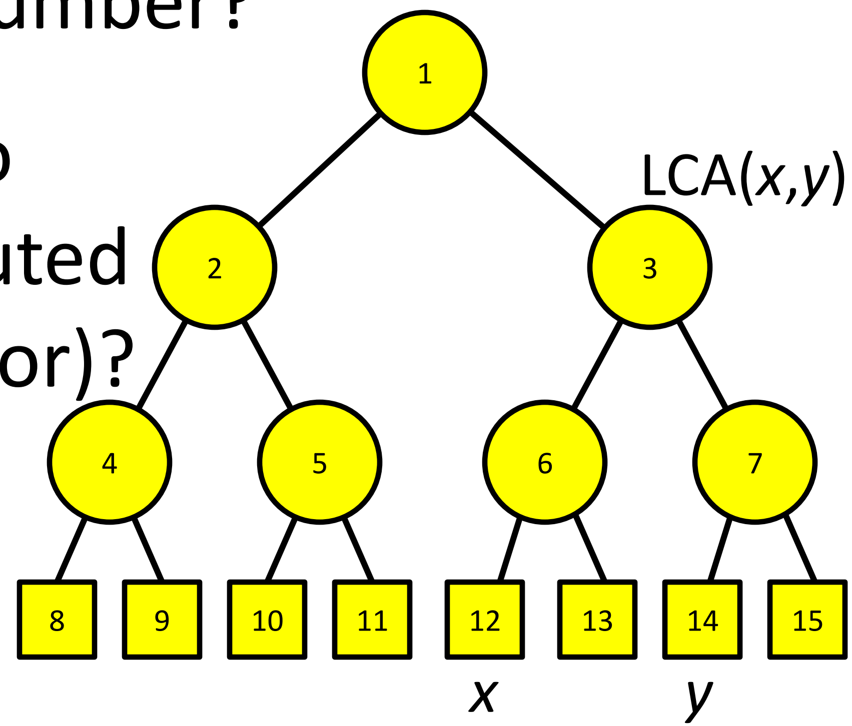


Exercise 15

Question.

a) How can the height of the tree be computed from a leaf number?

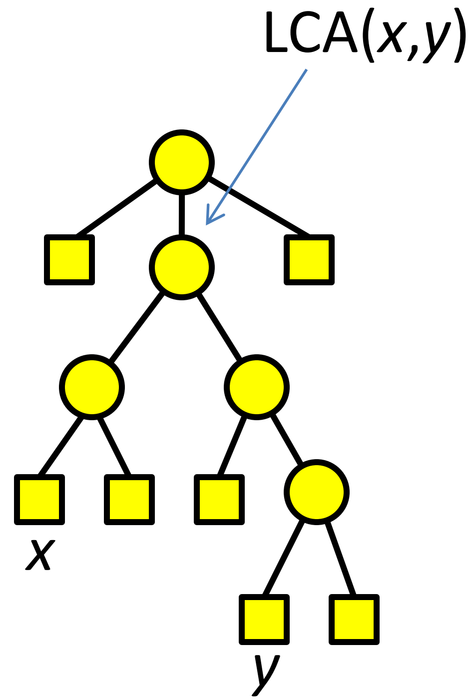
b) How can LCA(x,y) of two leaves x and y be computed (lowest common ancestor)?



Exercise 16*

Question.

Describe how to assign O(1) words to each node in an *arbitrary tree*, such that LCA(x,y) queries can be answered in O(1) time.



Searching

Exercise 17

Question. Consider a n-bit word x storing k n/k-bit values $v_0,\ldots,v_{k-1}$

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
x	0	1	0	0	1	1	1	0	0	0	0	1	1	0	0	0
	v_3				v_2				v_1				v_0			

a) Describe how to decide if all v_i are non-zero

b) Describe how to find the first v_i equal to zero

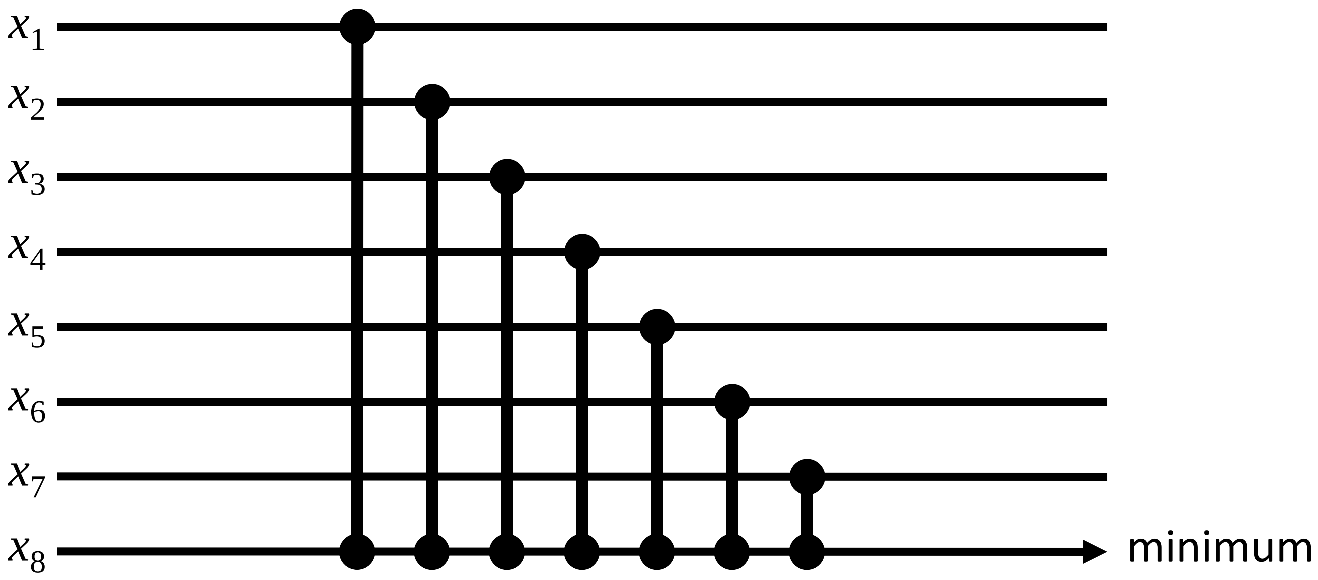
c) Describe how implement Search(x,u), that returns a i such that v_i=u (if such a v_i exists)

Sorting :
Sorting Networks

Exercise 18

Question. Construct a comparison network that outputs the *minimum* of 8 input lines.

What is the number of comparators and the depth of the comparison network?

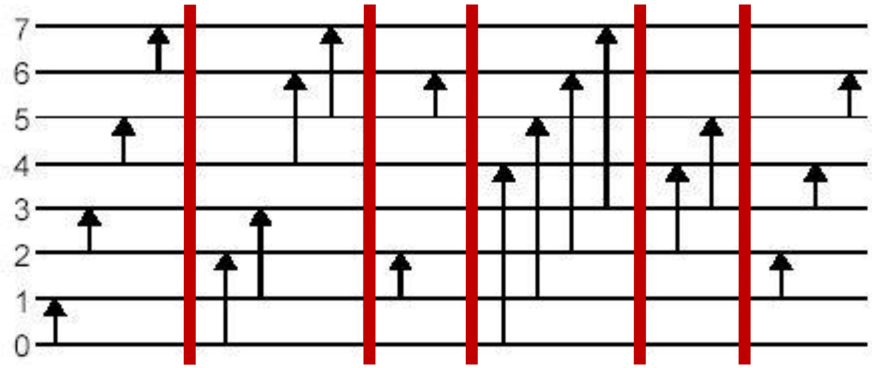


Exercise 19

Question. Construct a comparison network that outputs the *minimum* and *maximum* of 8 input lines. What is the number of comparators and the depth of the comparison network?

Odd-Even Merge Sort

K.E. Batcher 1968



Odd-even merge sort for N=8.

Size $O(N \cdot (\log N)^2)$ and depth $O((\log N)^2)$

Fact. At each depth all compators have equal length

[Ajtai, Komlós, Szemerédi 1983: depth $O(\log N)$, size $O(N \cdot \log N)$]

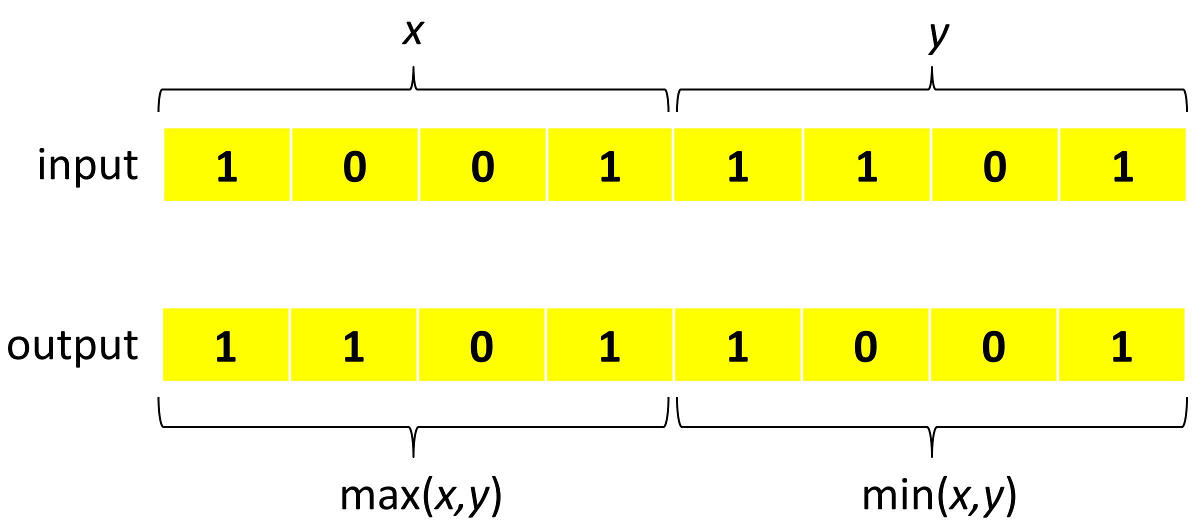
Sorting :
Word RAM implementations of Sorting Networks

Exercise 20

Question.

Descibe how to sort two sub-words stored in a single word on a Word RAM — without using branch-instructions

(implementation of a comparator)

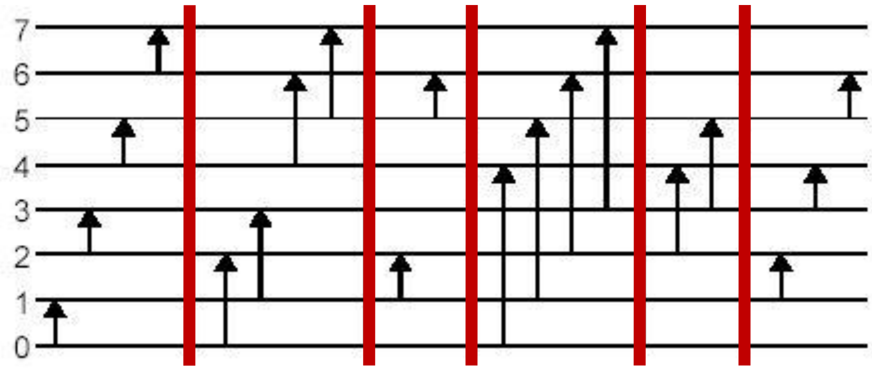


Exercise 21

Question.

Consider a n-bit word x storing n/k-bit values $v_0,\ldots,v_{k-1}$.

Describe a Word RAM implementation of odd-even merge sort with running $O((\log k)^2)$.



Odd-even merge sort for N=8.

More about
Sorting & Searching

More about Sorting & Searching

Sorting N words

Randomized	$O(N \cdot (\log\log N)^{1/2})$	Han & Thorup 2002
Deterministic	$O(N \cdot \log\log N)$	Han 2002
Randomized AC^0	$O(N \cdot \log\log N)$	Thorup 1997
Deterministic AC^0	$O(N \cdot (\log\log N)^{1+\varepsilon})$	Han & Thorup 2002

Dynamic dictionaries storing N words

Deterministic	$O((\log N/\log\log N)^{1/2})$	Andersson & Thorup 2001
Deterministic AC^0	$O((\log N)^{3/4+o(1)})$	

Summary

Summary

- Many operations on words can be efficiently without using multiplication
- $\lambda(x)$ and $\rho(x)$ can be computed in O(1) time using multiplication, and O(loglog n) time without mult.
- Parallellism can be achieved by packing several elements into one word
- The great (theory) question:

  Can N words be sorted on a Word RAM in O(N) time?