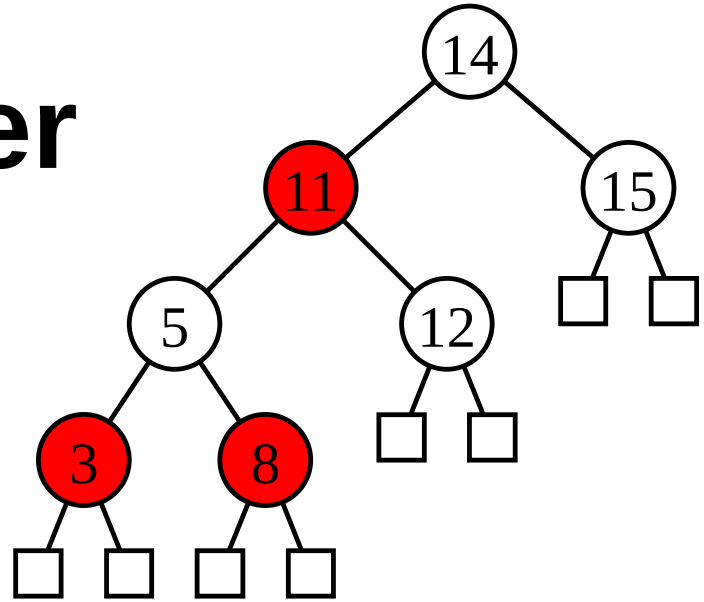
Algoritmer og Datastrukturer

Balancerede søgetræer: Rød-sorte søgetræer
[CLRS, kapitel 13]

Rød-Sorte Søgetræer



Mål

Søgetræer med dybde $O(\log n)$

Invarianter

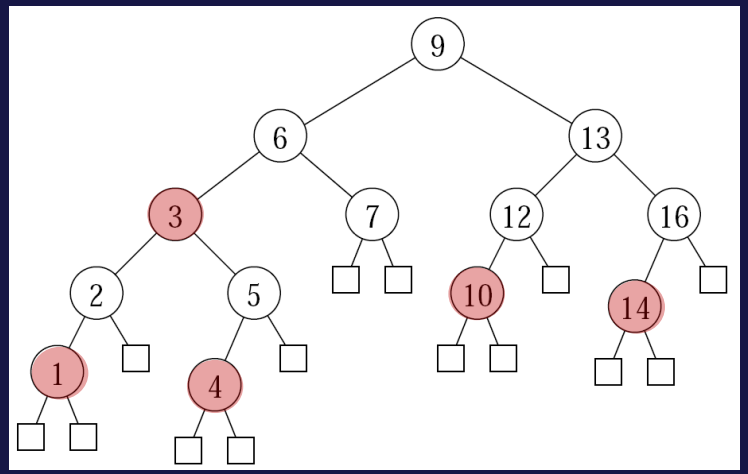
- Hver knude er enten **RØD** eller **SORT**
- Hver **RØD** knude har en **SORT** far
- Alle stier fra roden til et eksternt blad har **samme antal sorte knuder**

Sætning

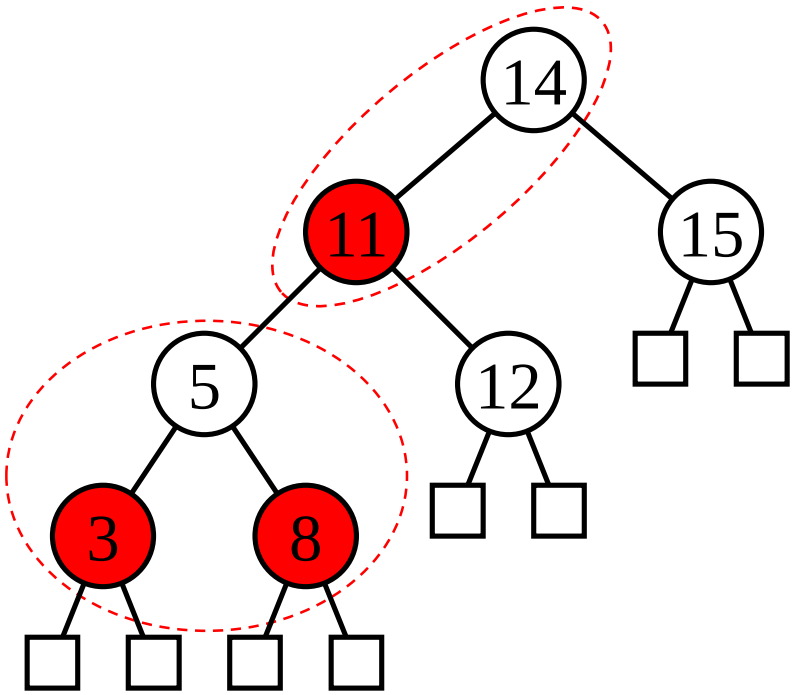
Et rød-sort træ har højde $\leq 2 \cdot \log (n+1)$

Hvilke knuder skal farves røde for at træet bliver et rød-sort søgetræ?

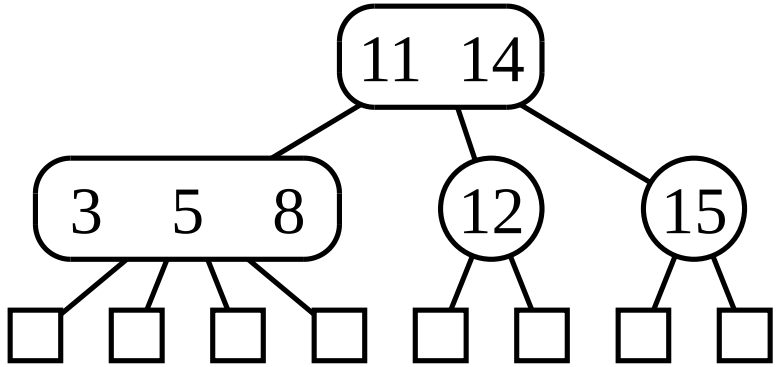
a) 1, 4, 10, 14
b) 2, 5, 10, 14
c) 1, 3, 4, 7, 12, 16
d) 1, 3, 4, 10, 14
e) ingen
f) ved ikke



Rød-Sorte vs (2-4)-træer



Rød-sort træ



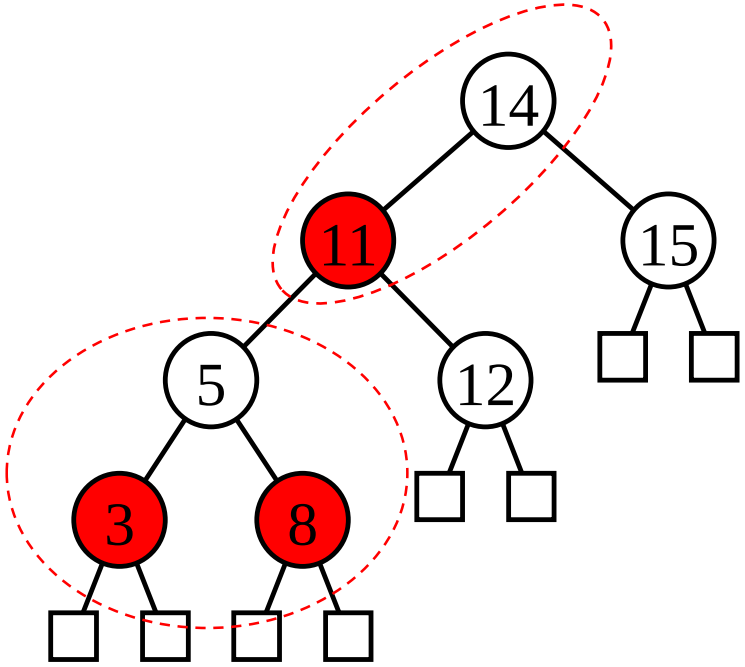
(2,4)-træ

(2,4)-træ ≡ **rød-sort træ** hvor alle røde knuder er slået sammen med faderen

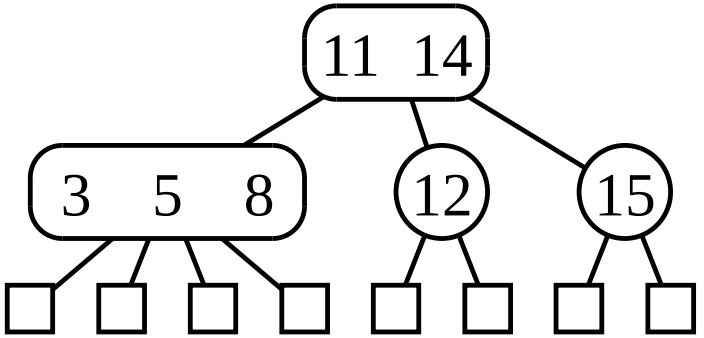
[2-4-træer = CLRS kapitel 18.1 & problem 18-2]

Egenskaber ved (2,4)-træer

- Alle interne knuder har mellem **2 og 4 børn**
- Knude med i **børn** gemmer i-1 **elementer**
- Stier fra roden til et eksternt blad er lige lange

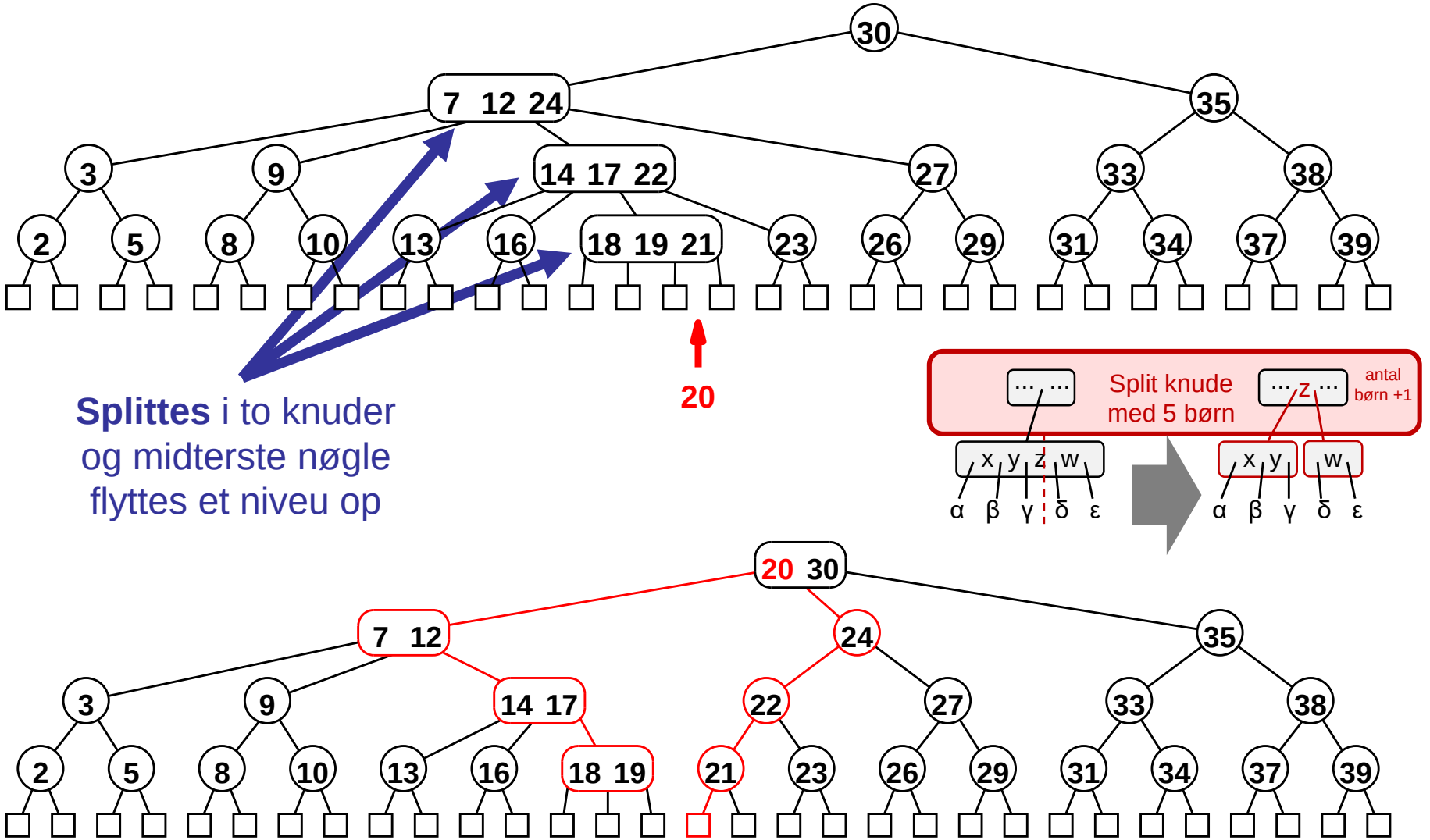


Rød-sort træ



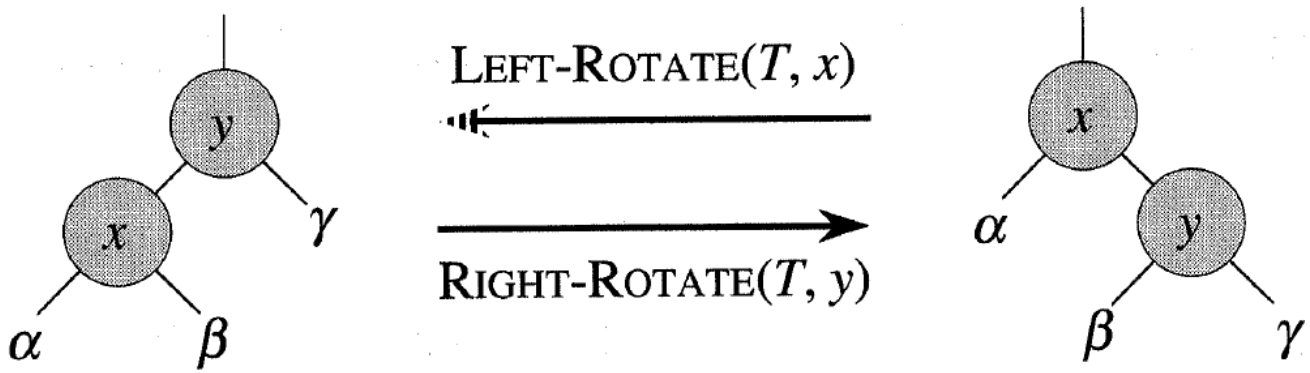
(2,4)-træ

Indsættelse i et (2,4)-træ



Opdateringer af Rød-Sorte Træer

Rotationer



LEFT-ROTATE(T, x)

```
1   y = x.right                    // set y
2   x.right = y.left               // turn y's left subtree into x's right subtree
3   if y.left ≠ T.nil
4       y.left.p = x
5   y.p = x.p                      // link x's parent to y
6   if x.p == T.nil
7       T.root = y
8   elseif x == x.p.left
9       x.p.left = y
10  else x.p.right = y
11  y.left = x                     // put x on y's left
12  x.p = y
```

Insert

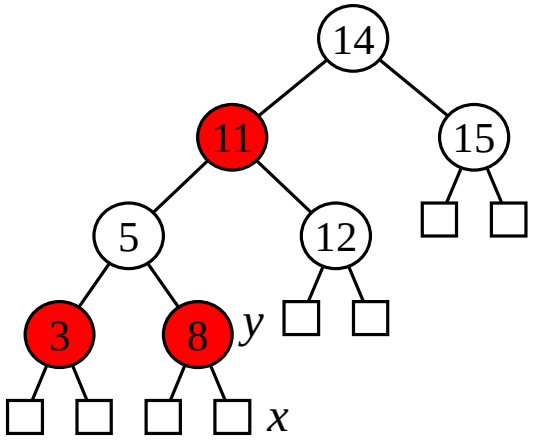
Ubalanceret indsættelse

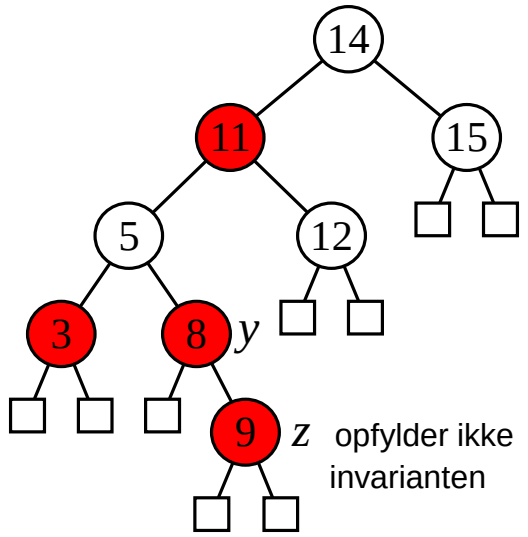
søg

opret z

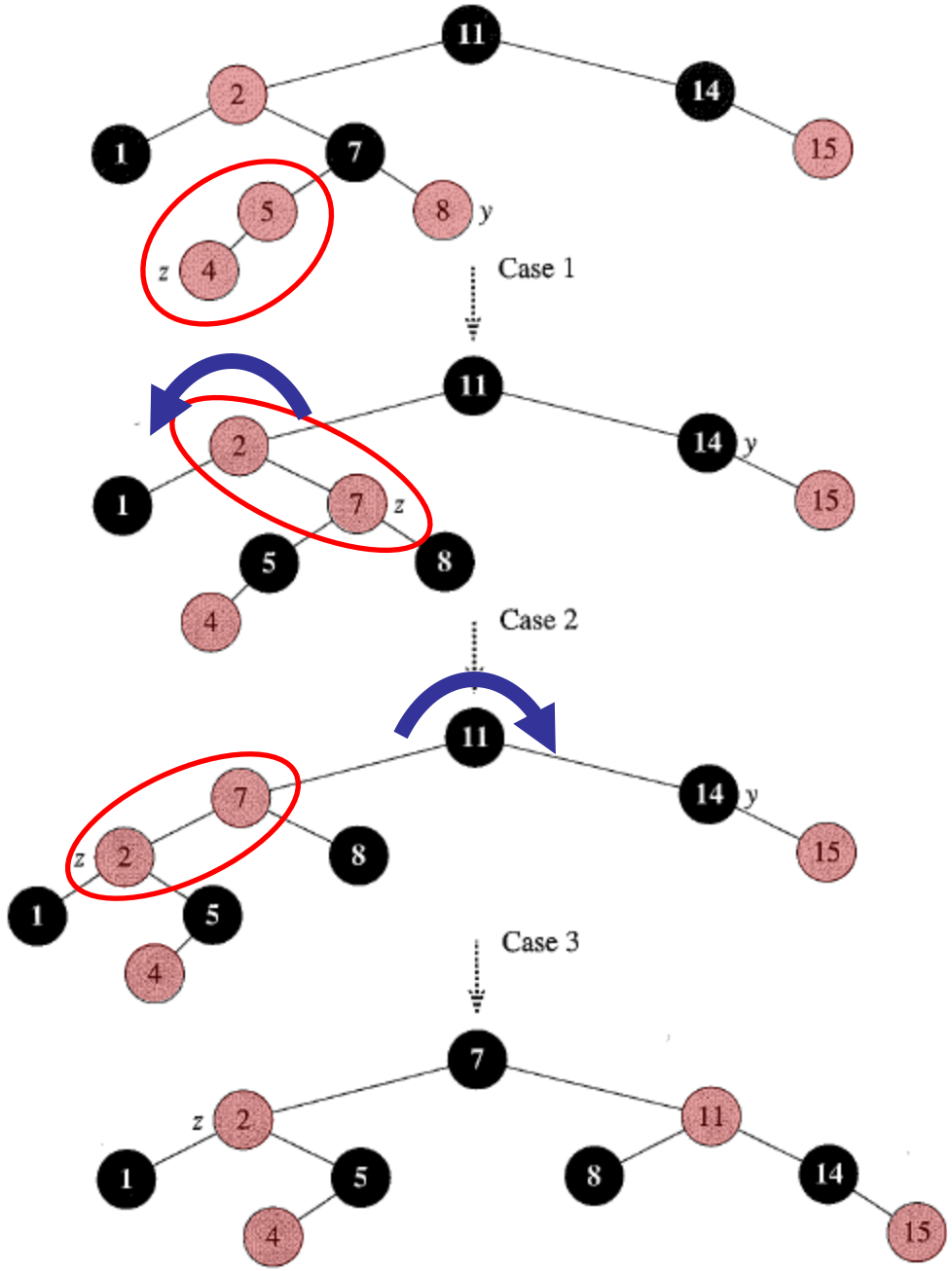
```
RB-INSERT(T, z)
 1  y = T.nil
 2  x = T.root
 3  while x ≠ T.nil
 4      y = x
 5      if z.key < x.key
 6          x = x.left
 7      else x = x.right
 8  z.p = y
 9  if y == T.nil
10      T.root = z
11  elseif z.key < y.key
12      y.left = z
13  else y.right = z
14  z.left = T.nil
15  z.right = T.nil
16  z.color = RED
17  RB-INSERT-FIXUP(T, z)
```

Insert(9)



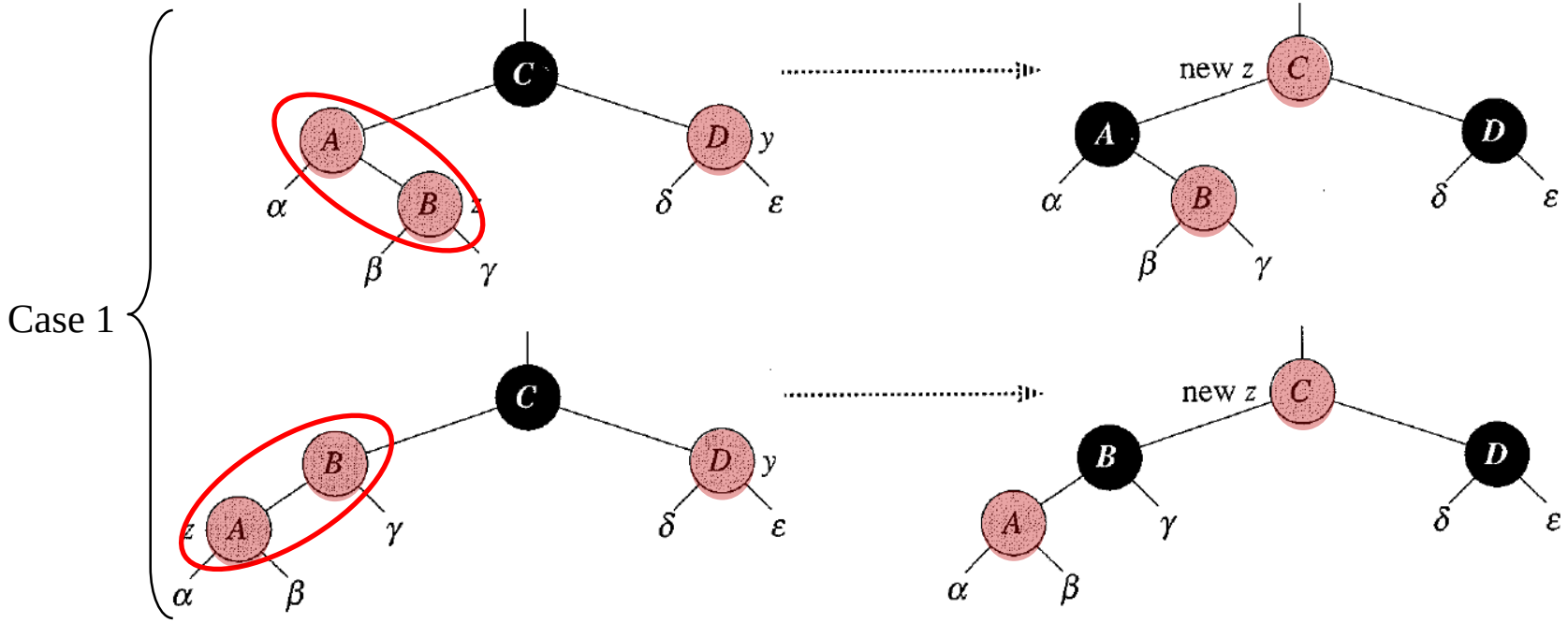


z opfylder ikke invarianten

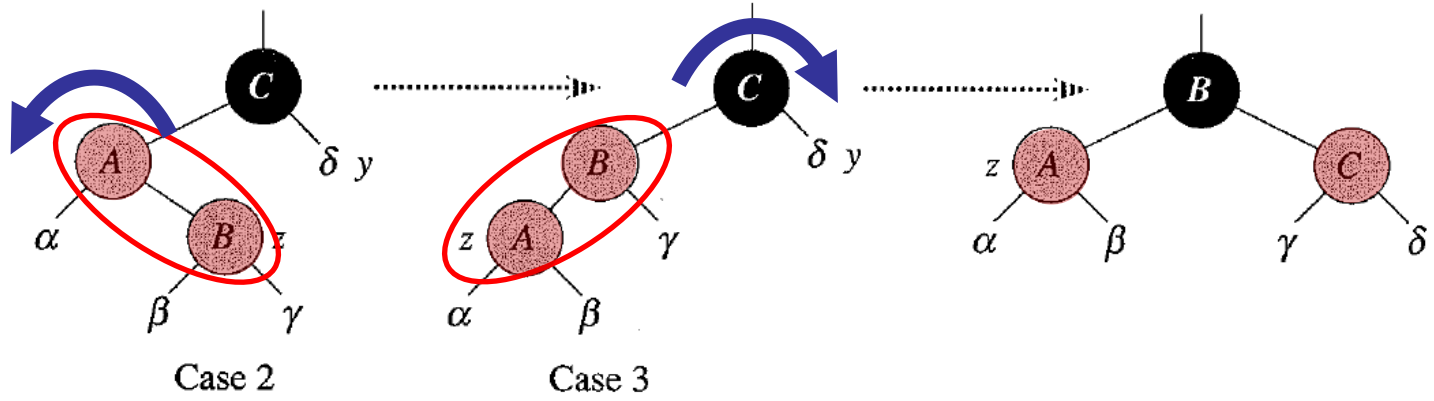

11
2
14
1
7
15
5
8
y
z
4
Case 1
11
2
14
y
1
7
z
15
5
8
4
Case 2
11
7
14
y
z
2
8
15
1
5
4
Case 3
7
z
2
11
1
5
8
14
4
15

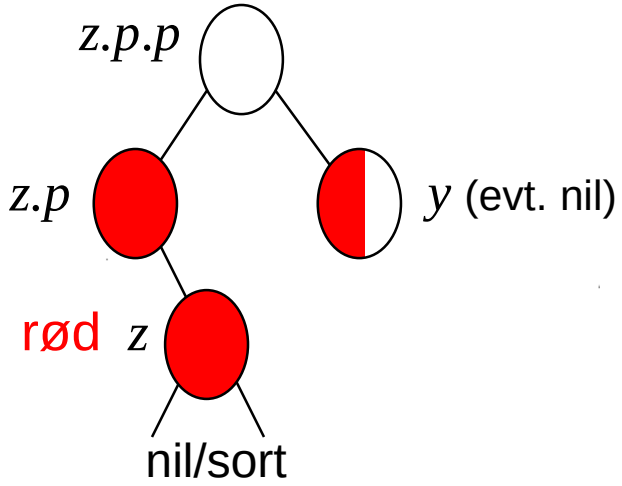
Indsættelse i rød-sorte træer: rebalancering

Omfarv *C* og dens to røde børn



Afslut






```
RB-INSERT-FIXUP(T, z)
 1  while z.p.color == RED
 2      if z.p == z.p.p.left
 3          y = z.p.p.right
 4          if y.color == RED
 5              z.p.color = BLACK                    // case 1
 6              y.color = BLACK                      // case 1
 7              z.p.p.color = RED                    // case 1
 8              z = z.p.p                            // case 1
 9          else if z == z.p.right
10                  z = z.p                          // case 2
11                  LEFT-ROTATE(T, z)                // case 2
12              z.p.color = BLACK                    // case 3
13              z.p.p.color = RED                    // case 3
14              RIGHT-ROTATE(T, z.p.p)               // case 3
15      else (same as then clause
                with "right" and "left" exchanged)
16  T.root.color = BLACK
```

Worst-case antal rotationer ved indsættelse i et rød-sort træ ?

a) 1

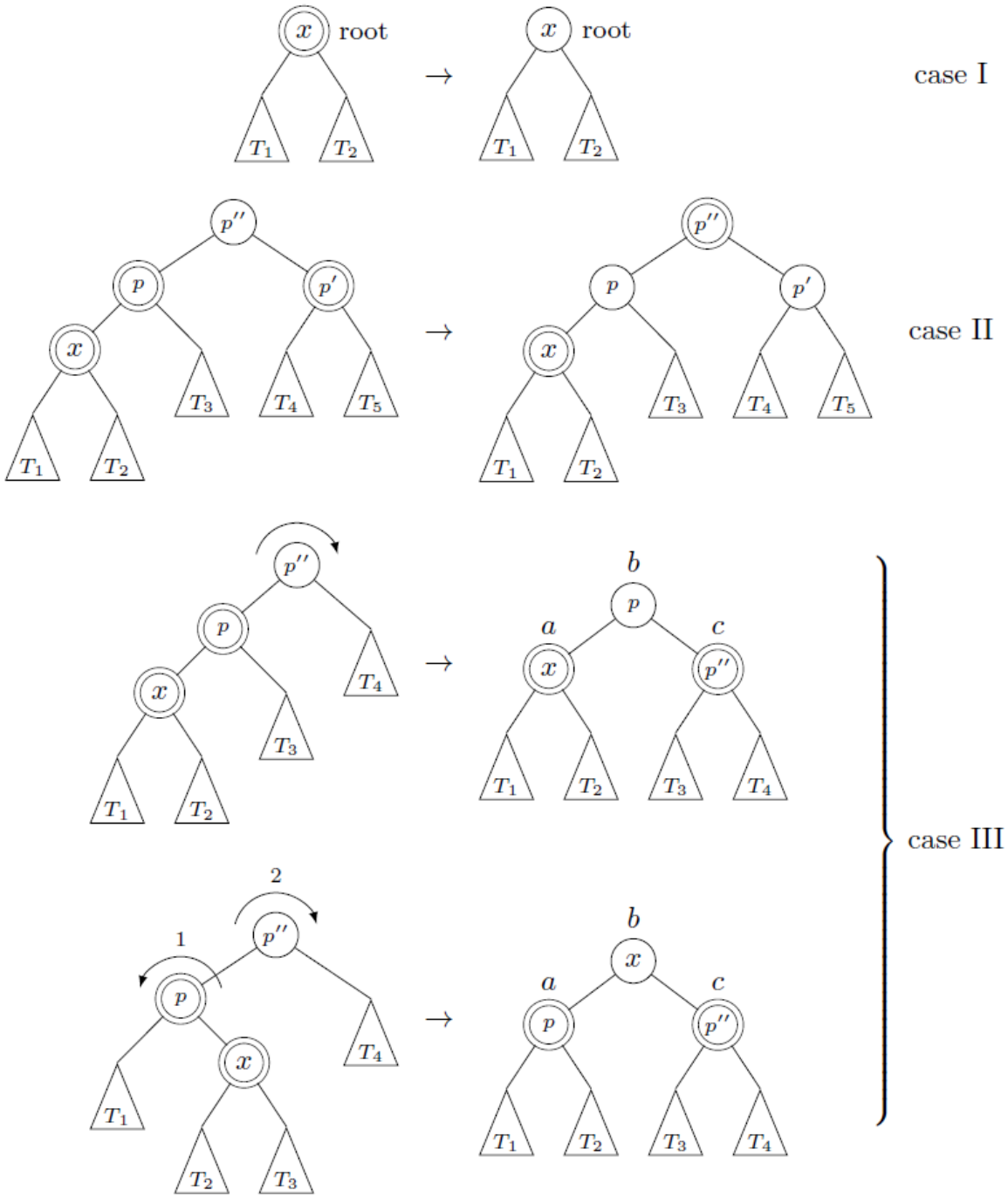
b) 2

c) log *n*

d) 2·log *n*

e) Ved ikke

[B]

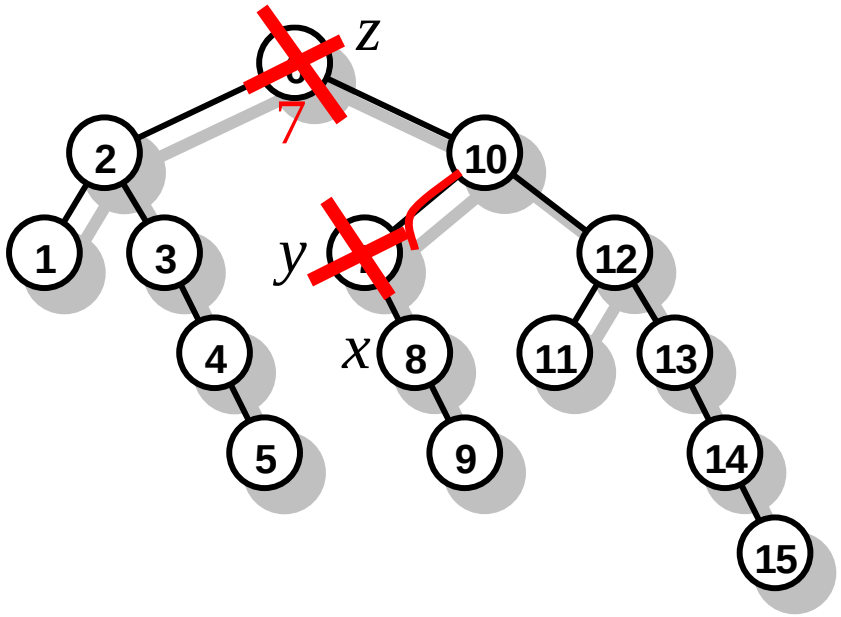


Delete

[CLRS, Ed. 3]

```
RB-DELETE(T, z)
 1  y = z
 2  y-original-color = y.color
 3  if z.left == T.nil
 4       x = z.right
 5       RB-TRANSPLANT(T, z, z.right)
 6  elseif z.right == T.nil
 7       x = z.left
 8       RB-TRANSPLANT(T, z, z.left)
 9  else y = TREE-MINIMUM(z.right)
10       y-original-color = y.color
11       x = y.right
12       if y.p == z
13            x.p = y
14       else RB-TRANSPLANT(T, y, y.right)
15            y.right = z.right
16            y.right.p = y
17       RB-TRANSPLANT(T, z, y)
18       y.left = z.left
19       y.left.p = y
20       y.color = z.color
21  if y-original-color == BLACK
22       RB-DELETE-FIXUP(T, x)
```

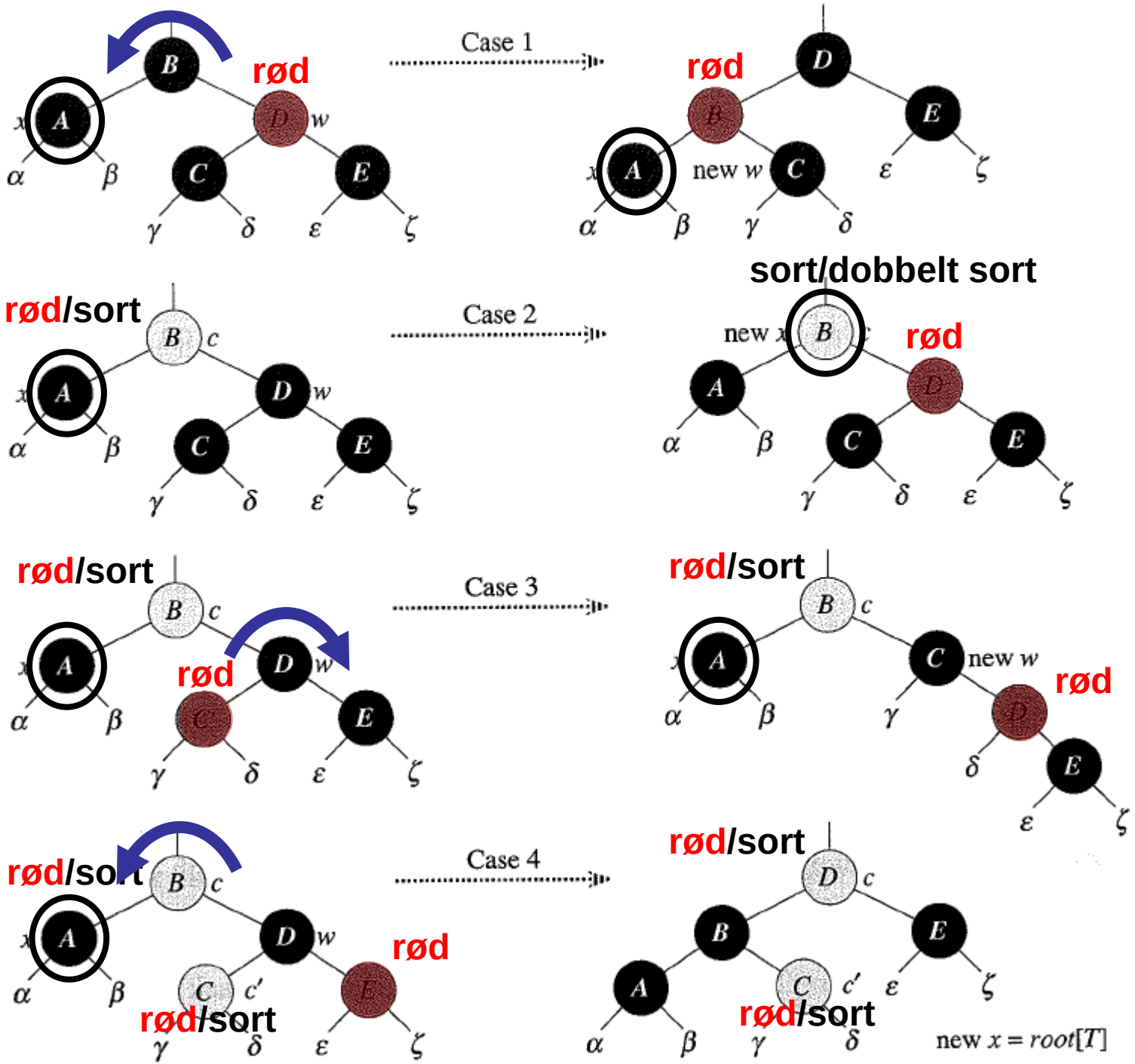
Ubalanceret slettelse



***x* og *y.right* kan være T.nil !!!**

```
RB-TRANSPLANT(T, u, v)
1  if u.p == T.nil
2       T.root = v
3  elseif u == u.p.left
4       u.p.left = v
5  else u.p.right = v
6  v.p = u.p
```

x
”dobbelt sort”



Case 1 → 2,3,4

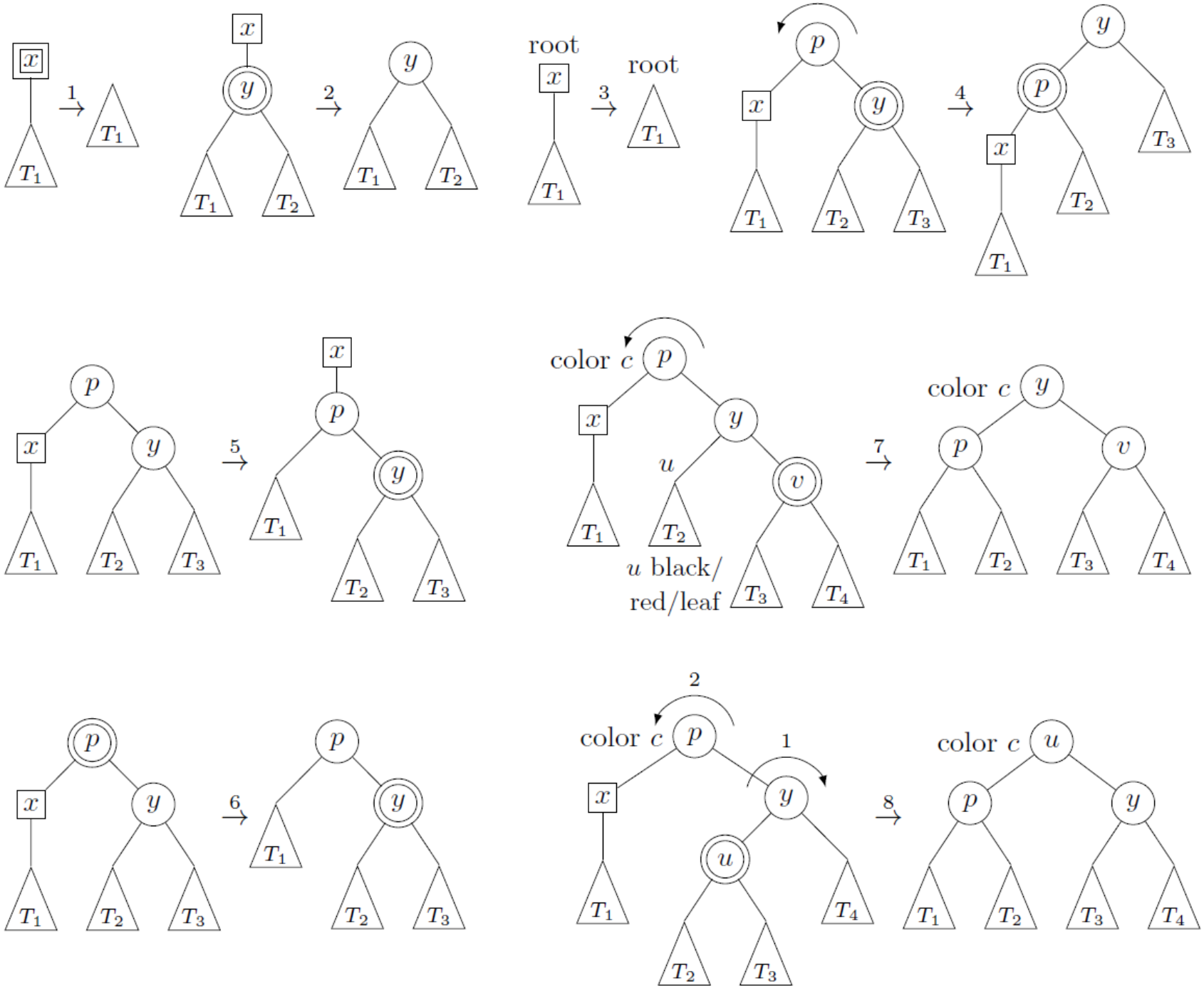
Case 2 → Problemet x flyttet tættere på roden eller færdig

Case 3 → 4 (og færdig)

Case 4 → Færdig

```
RB-DELETE-FIXUP(T, x)
 1  while x ≠ T.root and x.color == BLACK
 2      if x == x.p.left
 3          w = x.p.right
 4          if w.color == RED
 5              w.color = BLACK                          // case 1
 6              x.p.color = RED                          // case 1
 7              LEFT-ROTATE(T, x.p)                      // case 1
 8              w = x.p.right                            // case 1
 9          if w.left.color == BLACK and w.right.color == BLACK
10              w.color = RED                            // case 2
11              x = x.p                                  // case 2
12          else if w.right.color == BLACK
13                  w.left.color = BLACK                 // case 3
14                  w.color = RED                        // case 3
15                  RIGHT-ROTATE(T, w)                   // case 3
16                  w = x.p.right                        // case 3
17              w.color = x.p.color                      // case 4
18              x.p.color = BLACK                        // case 4
19              w.right.color = BLACK                    // case 4
20              LEFT-ROTATE(T, x.p)                      // case 4
21              x = T.root                               // case 4
22      else (same as then clause with "right" and "left" exchanged)
23  x.color = BLACK
```

[B]



Dynamisk Ordbog : Rød-Sorte Træer

Search(S, x**)**	$O(\log n)$
Insert(S, x**)**	
Delete(S, x**)**	

Ikke pensum

Rød-Sorte Træer i Java og C++

	Java (JDK SE 12.0.1)	C++ (GCC 9.1)
Metode	java.util.TreeMap	std::map
Dokumentation	https://docs.oracle.com/javase/10/docs/api/java/util/TreeMap.html	http://www.cplusplus.com/reference/map/map/
Kildekode	Installer JDK fra www.oracle.com/technetwork/java/javase/downloads/ Fil java.base\java\util\TreeMap.java fra C:\ProgramFiles\Java\jdk-12.0.1\lib\src.zip	https://github.com/gcc-mirror/gcc/blob/master/libstdc++-v3/include/bits/stl_tree.h