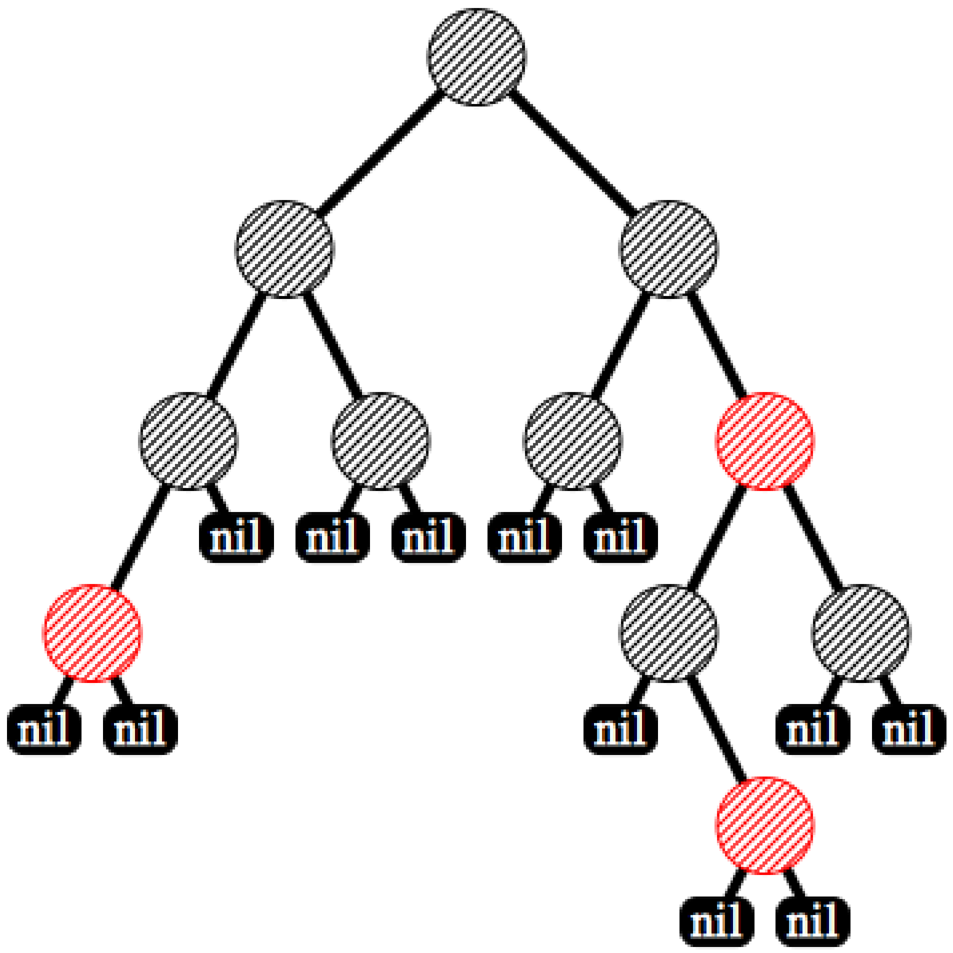
Stromy

Binární Vyhledávací Stromy, u kterých je časová složitost operací v nejhorším případě rovná O(log n)

Vlastnosti Red-Black Stromů

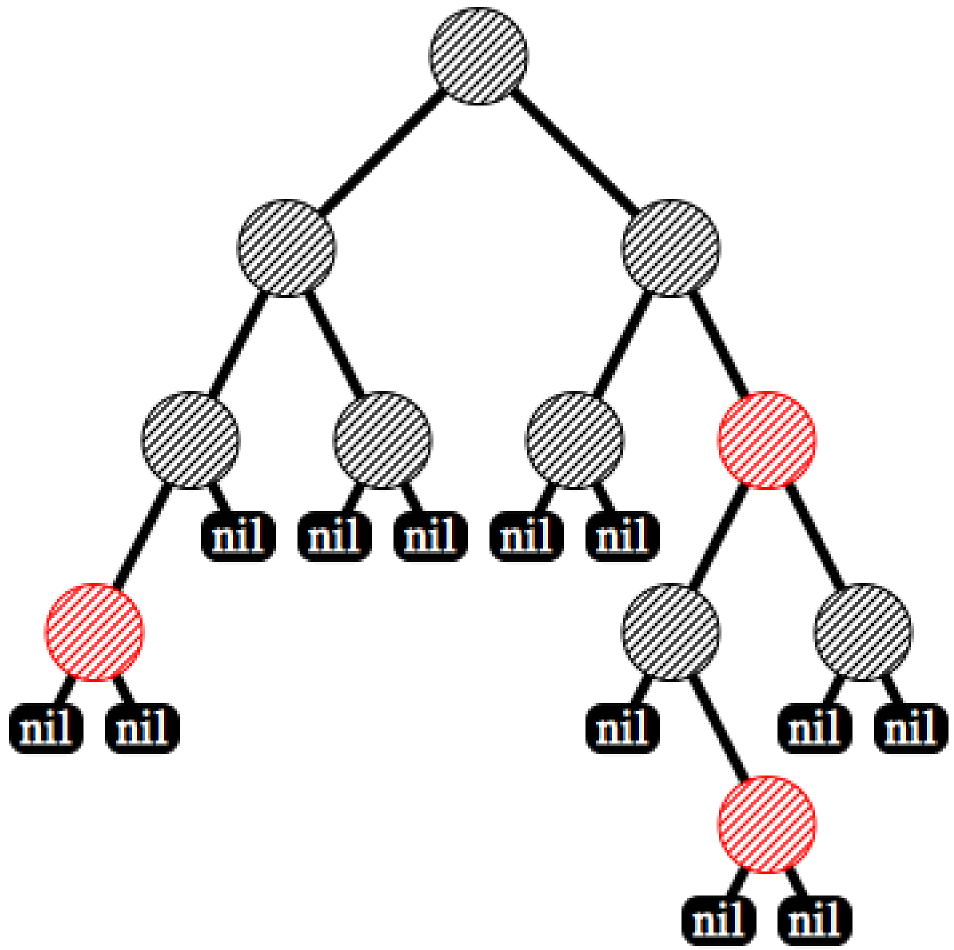


Vlastnosti Red-Black stromů

- **Každý uzel stromu je obarven červenou nebo černou barvou.**
- **Kořen stromu je obarven černě.**
- **Listy (nil) jsou černé.**
- **Červený uzel má pouze černé syny.**
- **Na kterékoliv cestě z kořene do listu leží stejný počet černých uzlů**

Černá výška (***black-height bh(x))*** uzlu ***x*** je počet černých uzlů na cestě z uzlu ***x*** (ale bez uzlu x) k listu. **Bh(T)** stromu T je rovna **bh(r),** kde r je kořen stromu.

Výška Red-Black stromů s n uzly



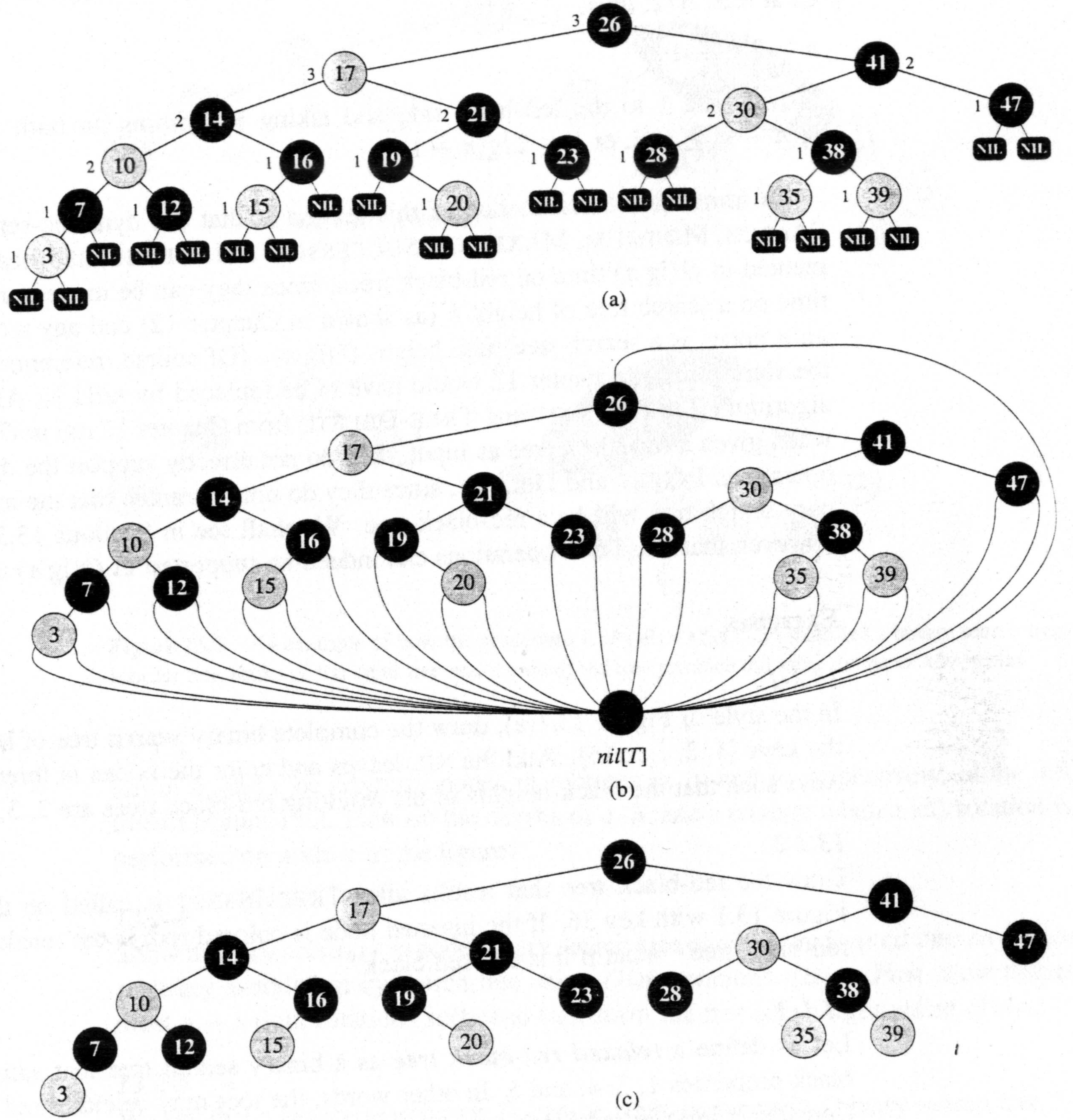
Lemma: *Red-black strom T s bh(T)=h má minimálně* $2^h - 1$ *uzlů.*

Lemma: *Red-black strom T s bh(T)=h* má maximálně $2h$ *uzlú.*

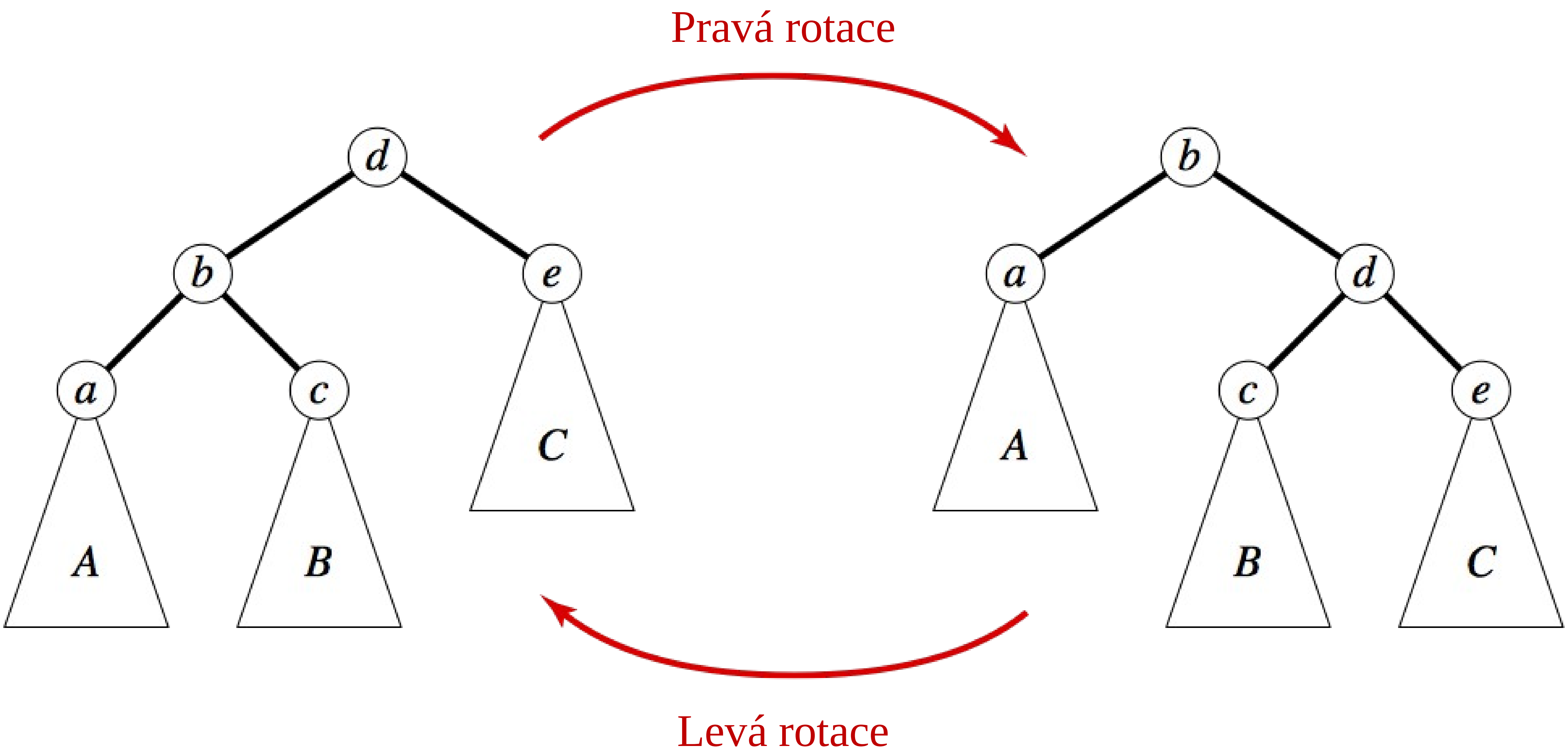
Důsledek: *Red-black strom s n uzly má výšku maximálně* $2 \log (n + 1)$.

Důsledek: *Operace nad red-black stromem velikosti n mají časovou složitost* $O(\log n)$

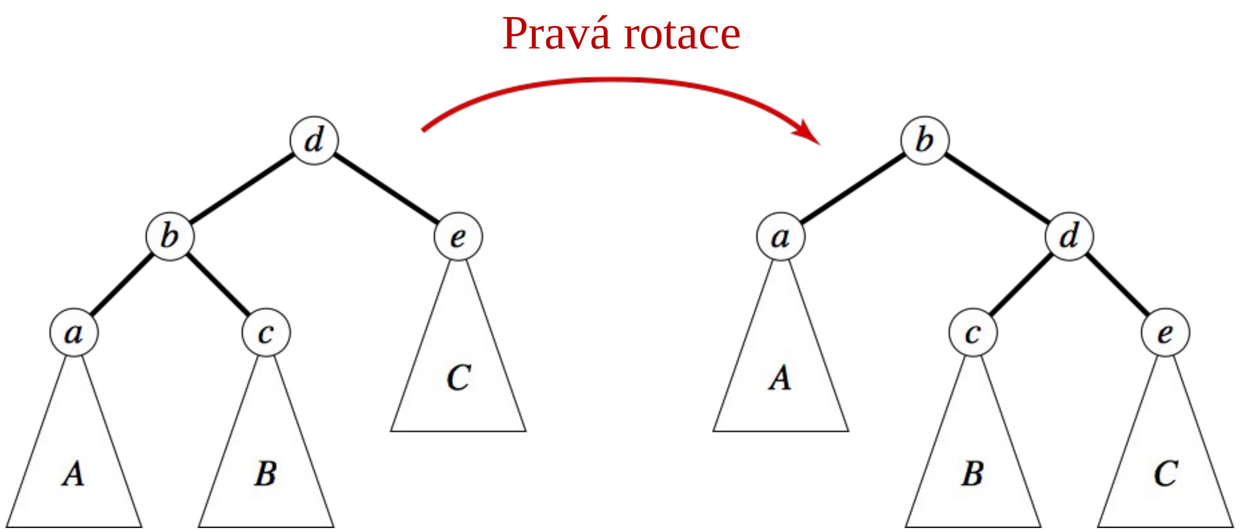
RB strom a implementace

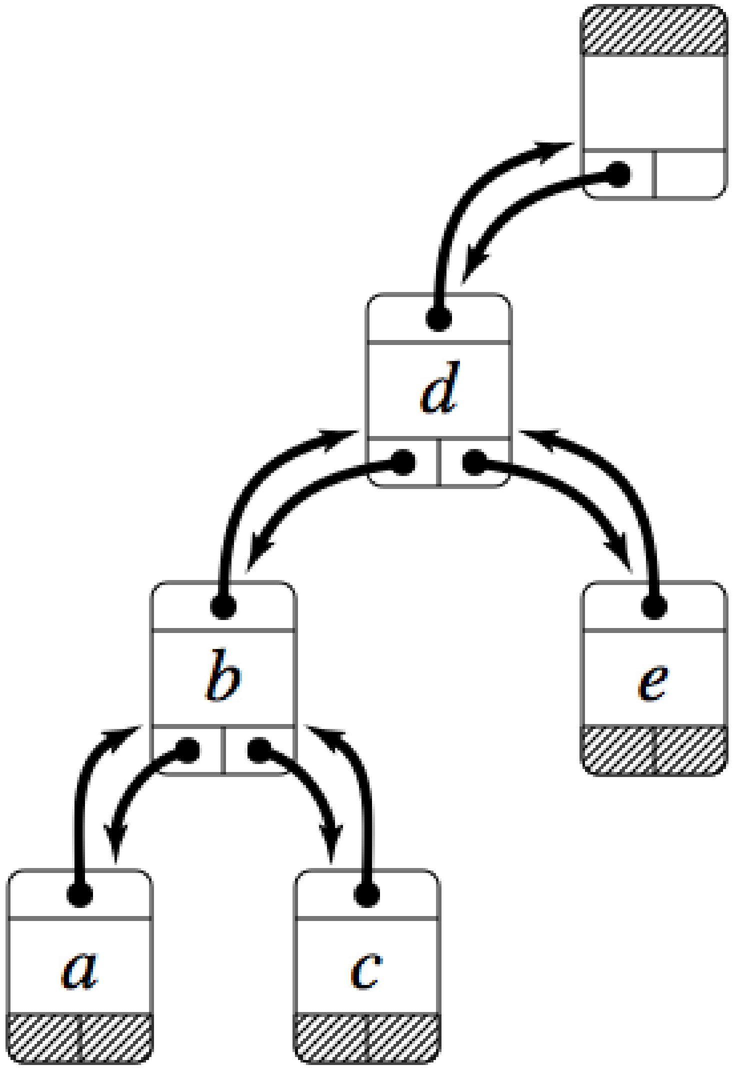


Rotace

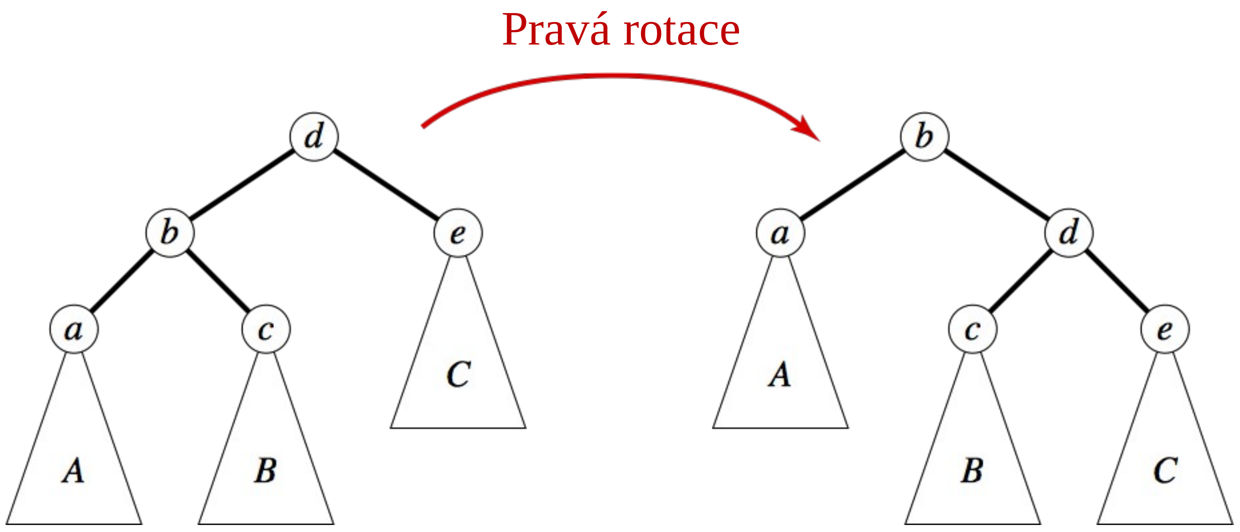


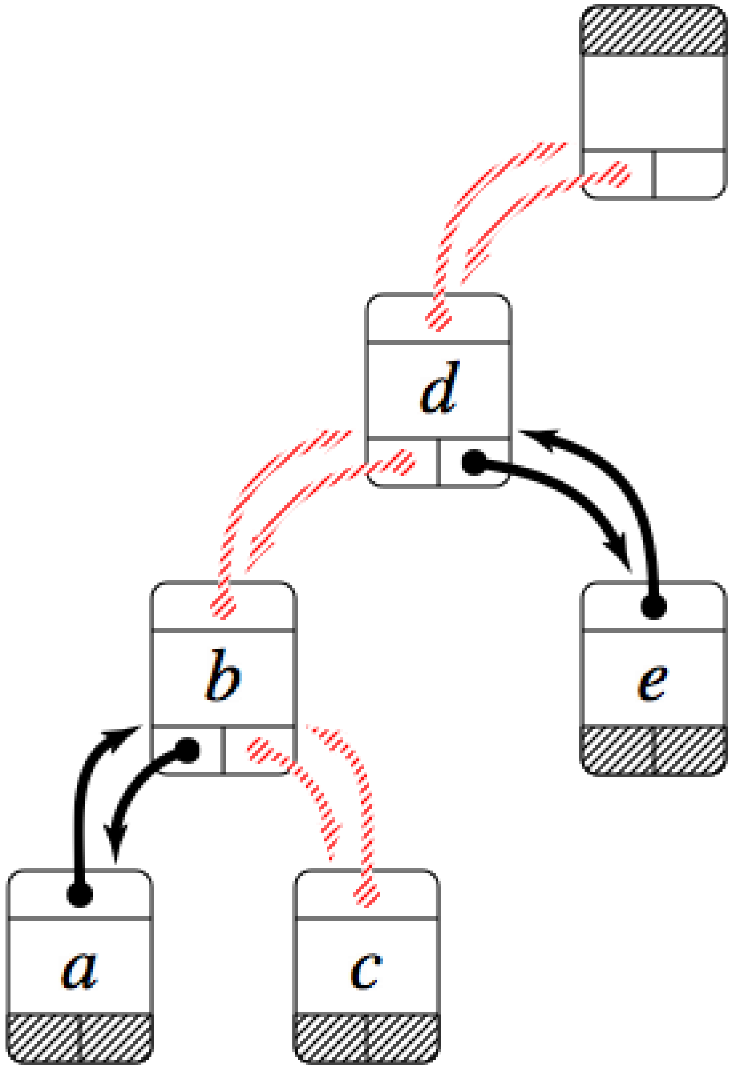
Operace rotace se provádí v konstantním čase



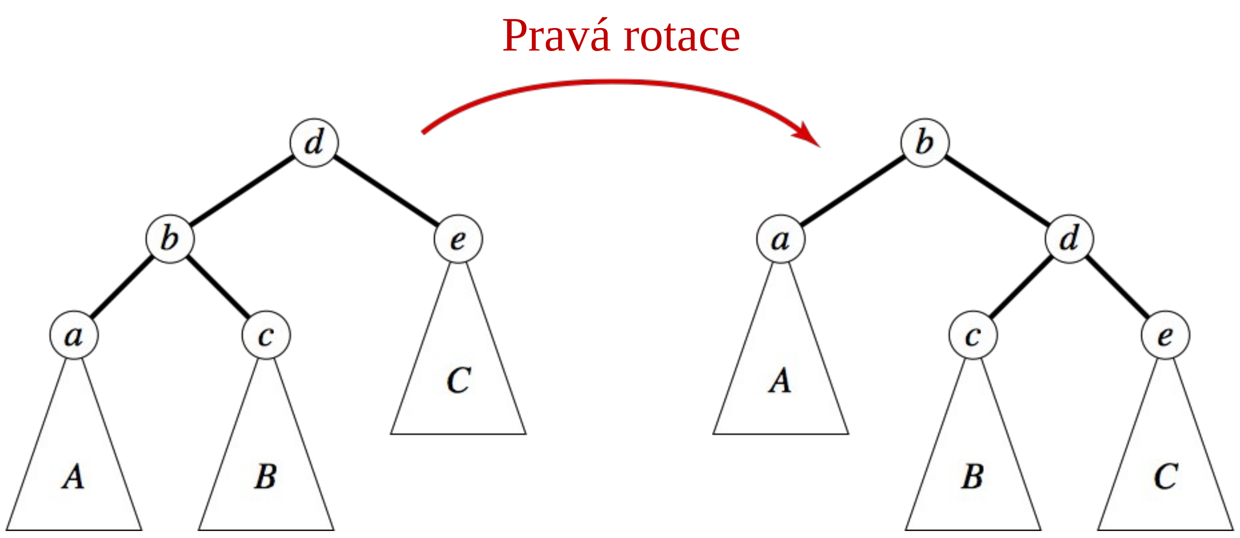


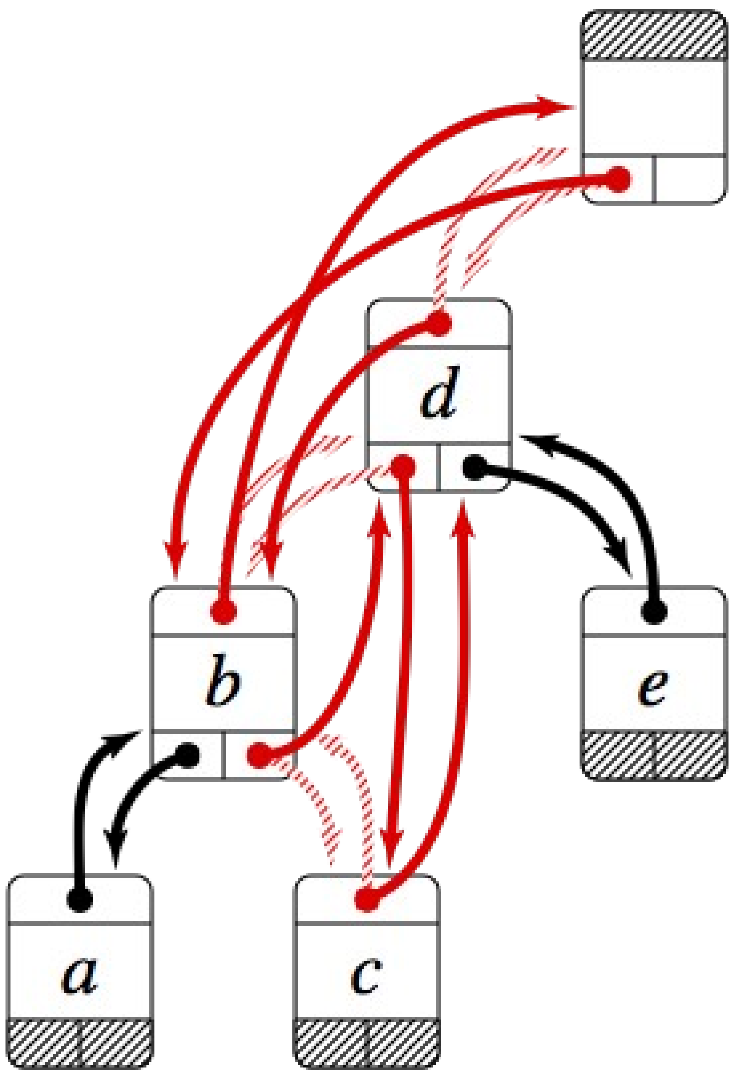
Operace rotace se provádí v konstantním čase



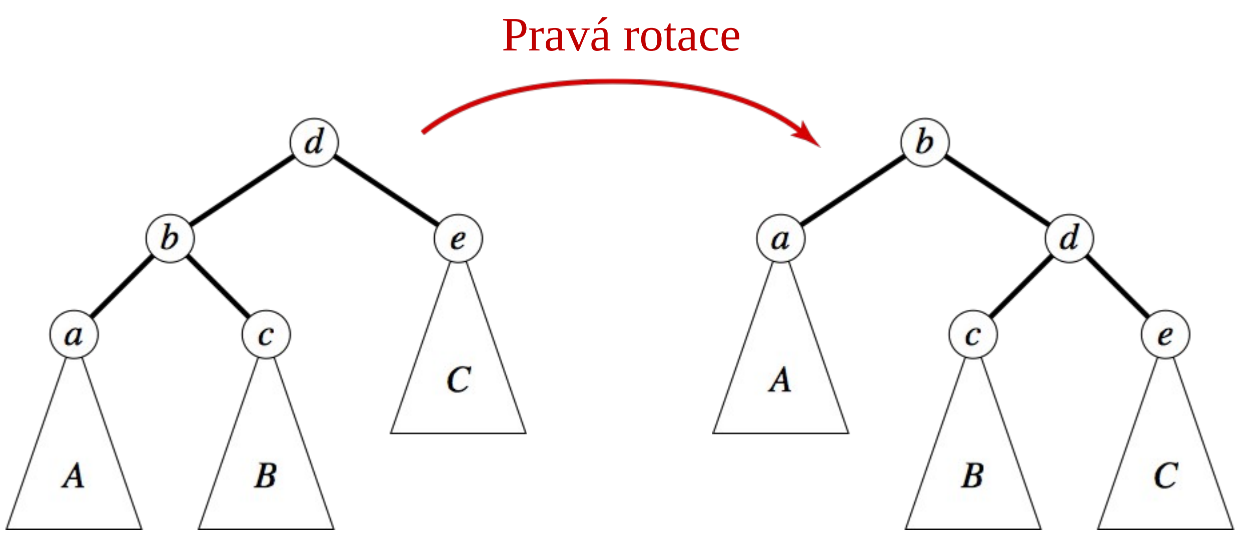


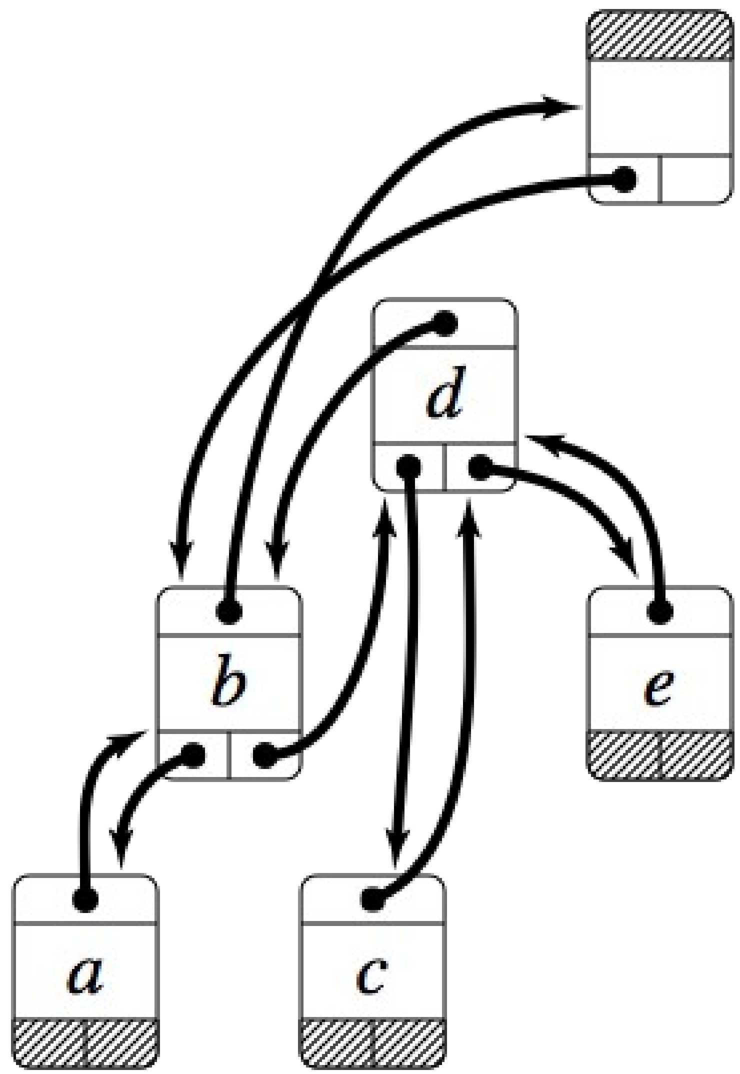
Operace rotace se provádí v konstantním čase



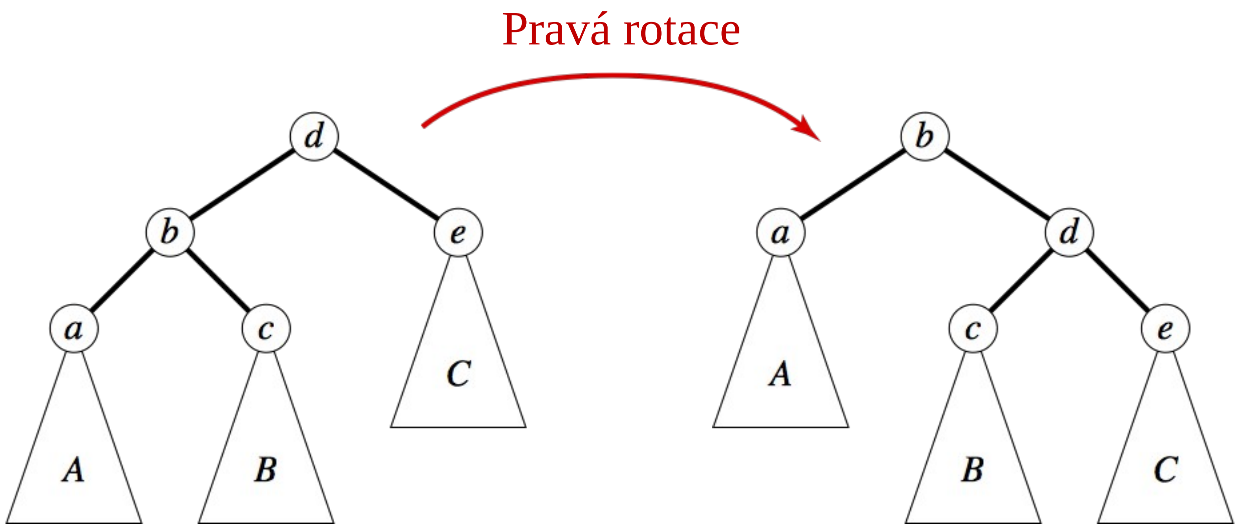


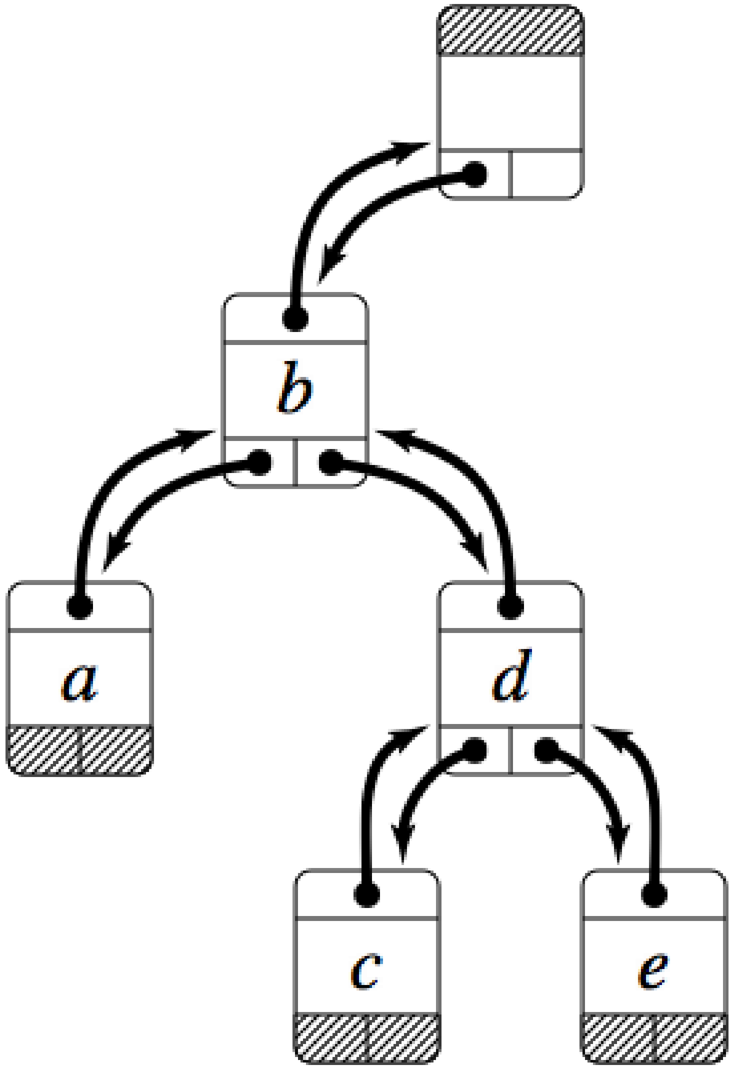
Operace rotace se provádí v konstantním čase



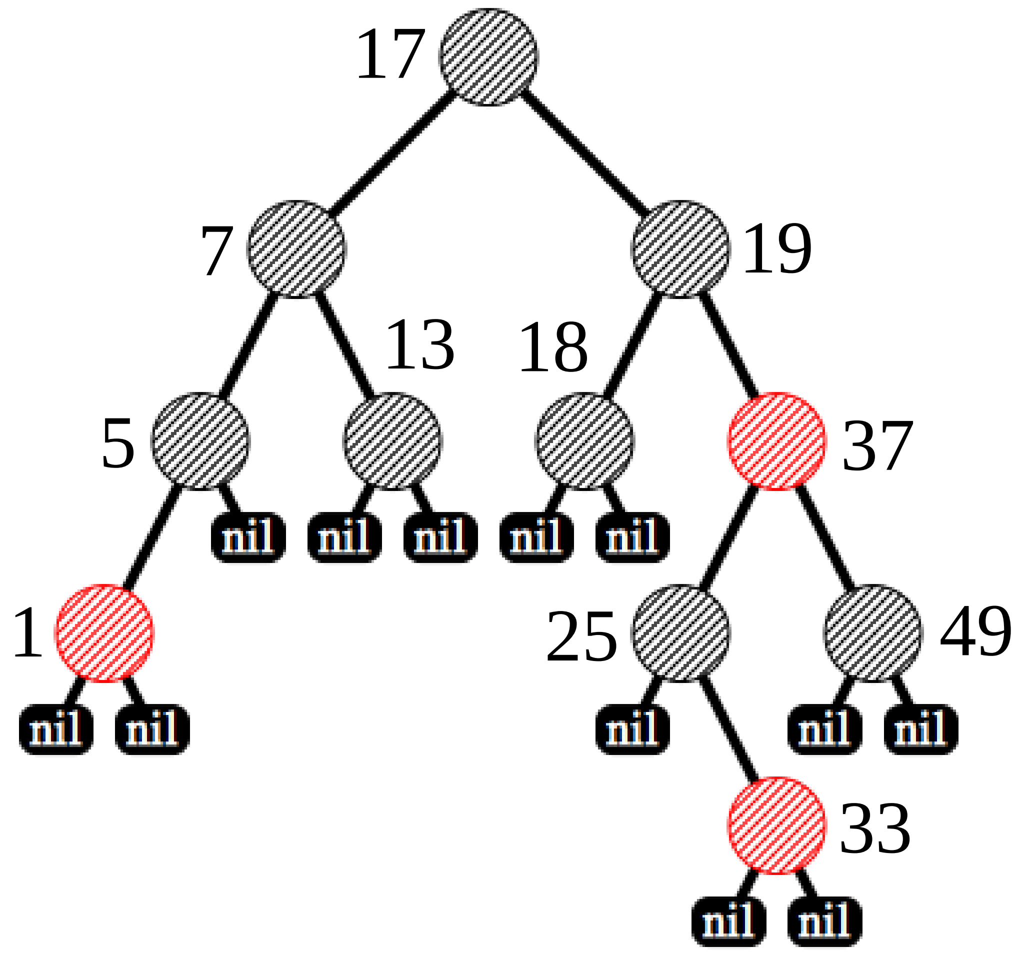


Operace rotace se provádí v konstantním čase



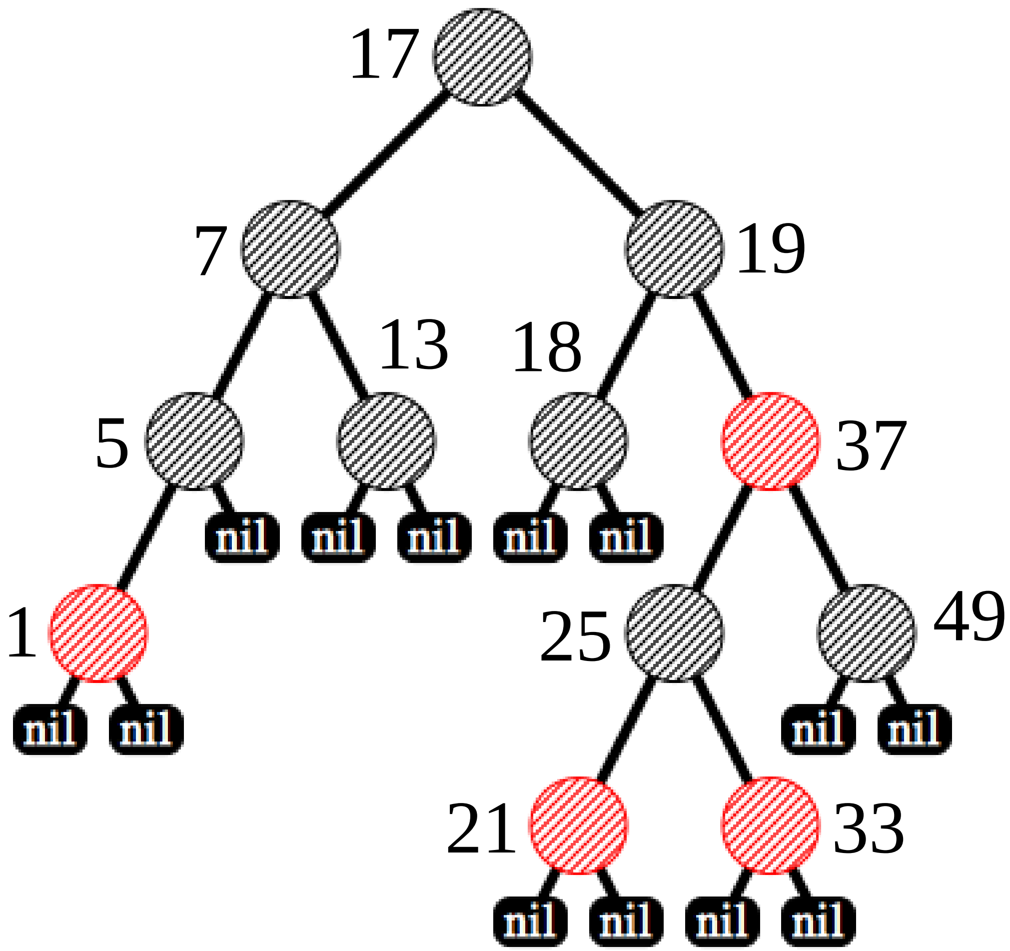


Vložení prvku do Red-Black stromu



Vkládaný prvek: 21

Vložení prvku do Red-Black stromu

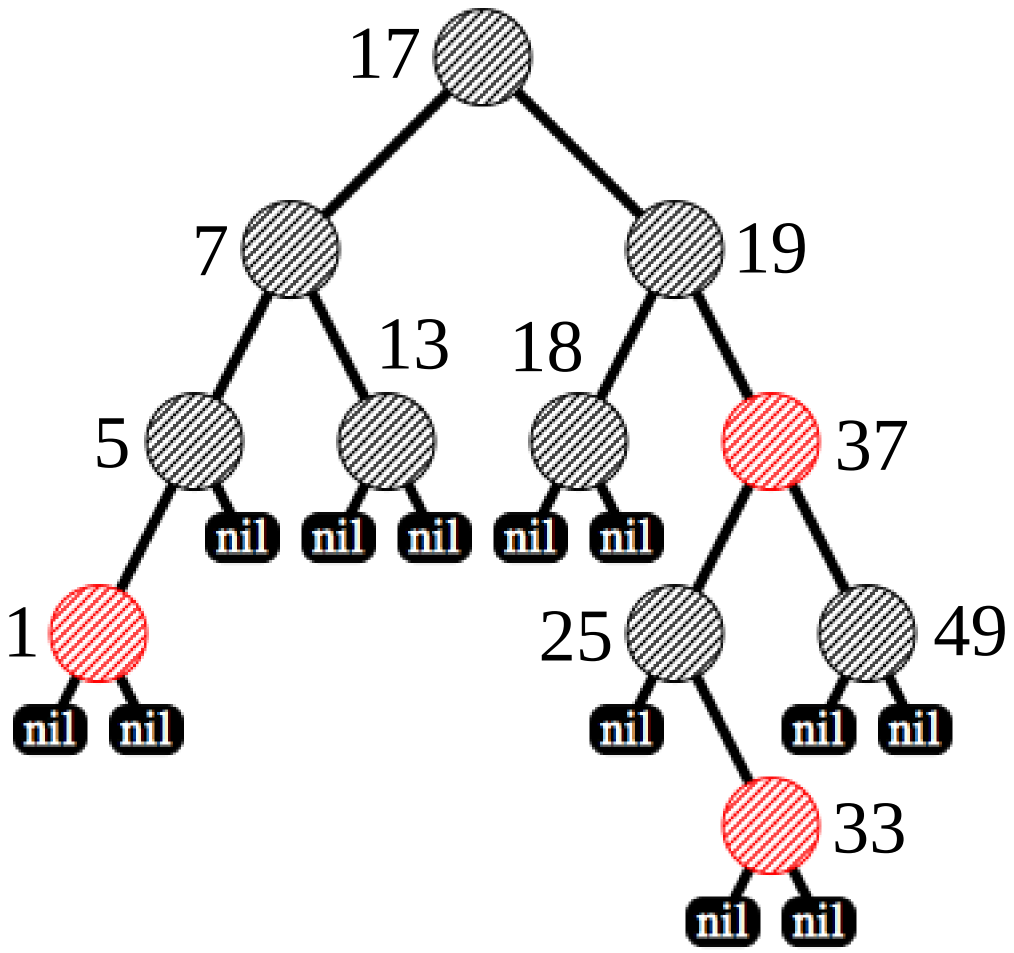


Vkládaný prvek: 21

Vlastnosti Red-Black stromů

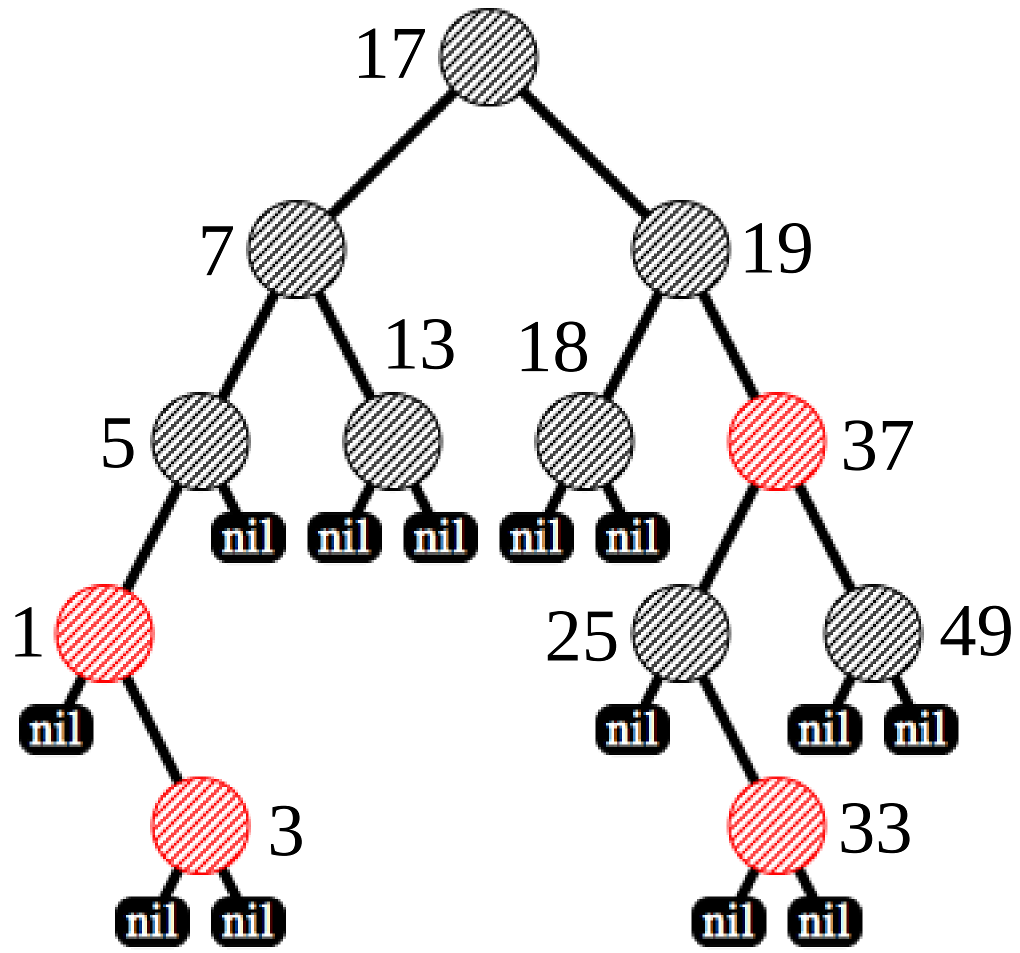
- **Každý uzel stromu je obarven červenou nebo černou barvou.** ✔
- **Kořen stromu je obarven černě.** ✔
- **Listy (nil) jsou černé.** ✔
- **Červený uzel má pouze černé syny.** ✔
- **Na kterékoliv cestě z kořene do listu leží stejný počet černých uzlů** ✔

Vložení prvku do Red-Black stromu



Vkládaný prvek: 3

Vložení prvku do Red-Black Stromu



Vkládaný prvek: 3

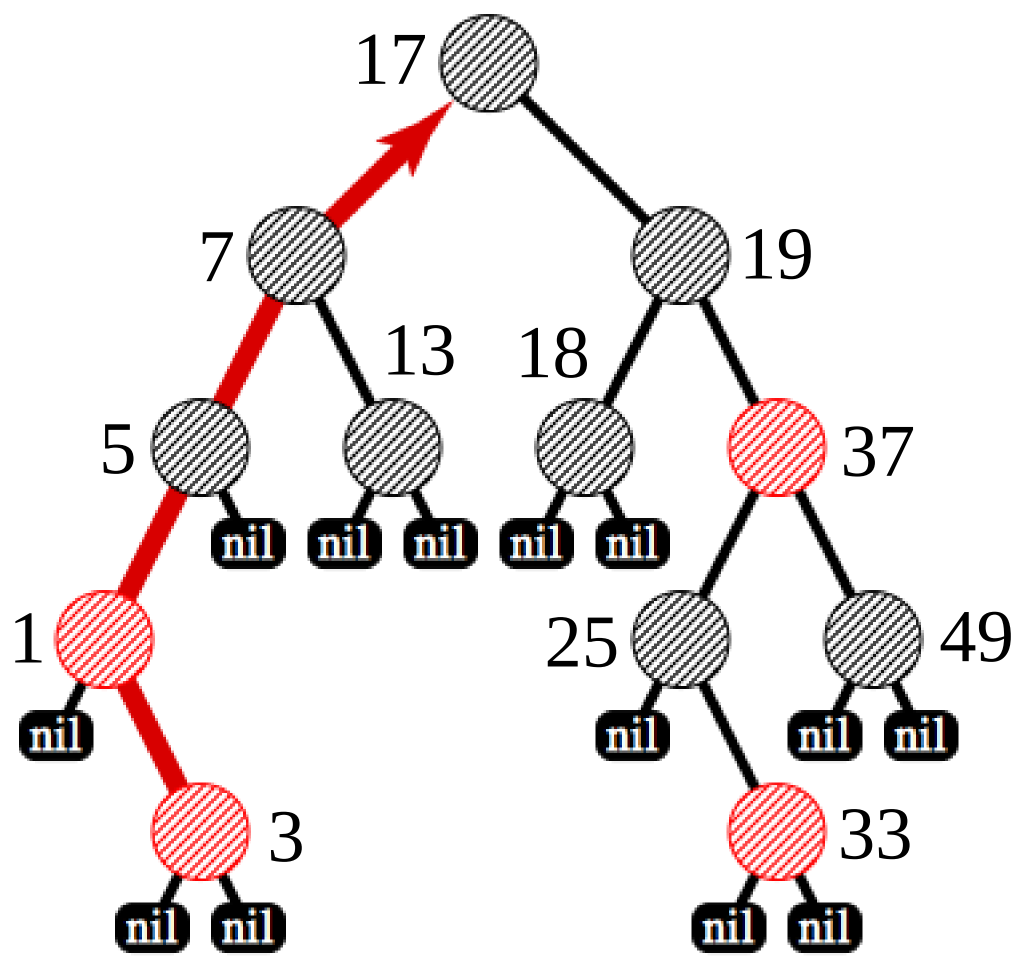
Vlastnosti Red-Black stromu

- **Každý uzel stromu je obarven červenou nebo černou barvou.** ✔
- **Kořen stromu je obarven černě.** ✔
- **Listy (nil) jsou černé.** ✔

Problém dvou červených uzlů

- **Červený uzel má pouze černé syny.** ✘
- **Na kterékoliv cestě z kořene do listu leží stejný počet černých uzlů** ✔

Cíl: Obnovit vlastnosti **Red-black** stromu přebarvením uzlů popř. provedením rotací



Vladaný prvek: 3

Vlastnosti Red-Black stromu

- **Každý uzel stromu je obarven červenou nebo černou barvou.** ✔
- **Kořen stromu je obarven černě.** ✔
- **Listy (nil) jsou černé.** ✔

Problém dvou červených uzlů

- **Červený uzel má pouze černé syny.** ✘
- **Na kterékoliv cestě z kořene do listu leží stejný počet černých uzlů** ✔

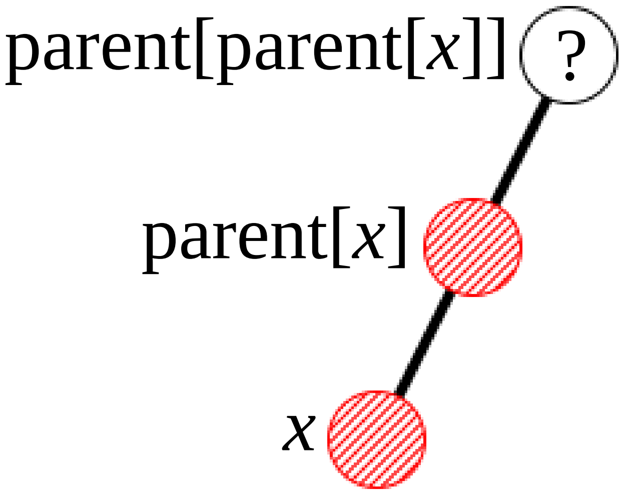
Cíl: Obnovit vlastnosti **Red-black** stromu přebarvením uzlů popř. provedením rotací

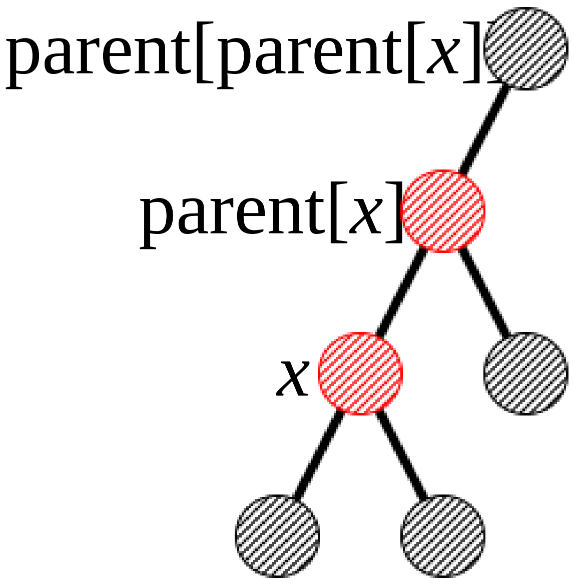
Obnovení vlastností Red-black stromu

- **Ve stromu existuje pouze jeden červený uzel x jehož předchůdce je červený**

- **Postup:**
- Opravit problém dvou červených uzlů *x*
- Oprava může způsobit stejný problém u předka -> je nutné postupovat směrem ke kořeni a upravit totéž i u předků

- **Platí následující:**
- **Jelikož *x* má červeného předka pak *x* není kořen stromu**.
- **Je-li parent[*x*] červený, pak ani on není kořenem tj. existuje parent[parent[*x*]].**
- **Je-li parent[*x*] černý, pak úpravy končí**.





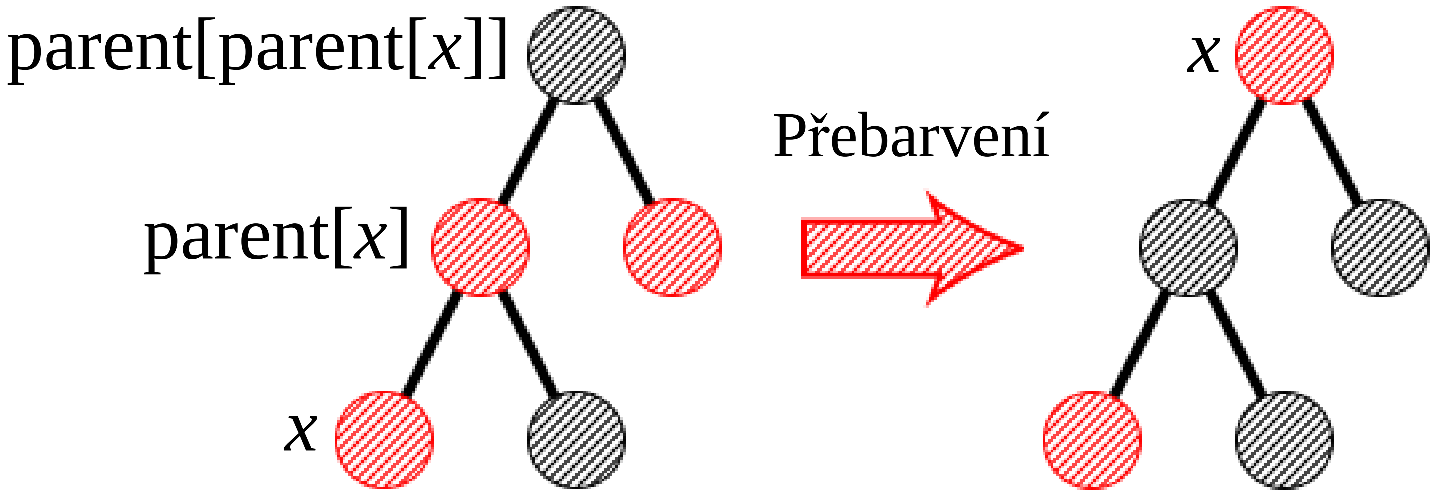
Algoritmus vložená prvků do RB stromu

```
RB-INSERT(T, z)
y ← nil[T]
x ← root[T]
while x ≠ nil[T]
     do y ← x
        if key[z] < key[x]
           then x ← left[x]
           else x ← right[x]
p[z] ← y
if y = nil[T]
   then root[T] ← z
   else if key[z] < key[y]
           then left[y] ← z
           else right[y] ← z
left[z] ← nil[T]
right[z] ← nil[T]
color[z] ← RED
RB-INSERT-FIXUP(T, z)
```

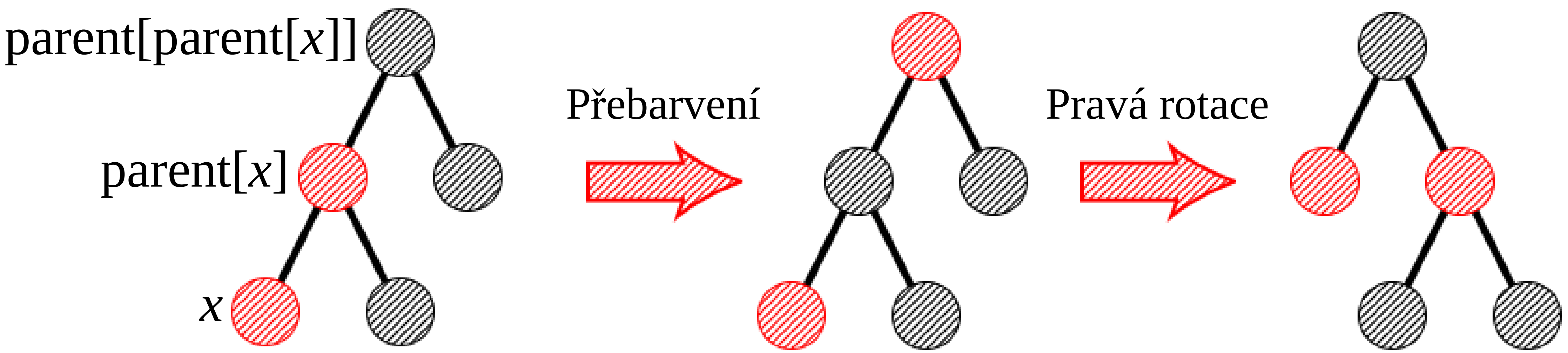
```
RB-INSERT-FIXUP(T, z)
while color[p[z]] = RED
     do if p[z] = left[p[p[z]]]
           then y ← right[p[p[z]]]
                if color[y] = RED
                  then color[p[z]] ← BLACK                    ▷ Case 1
                       color[y] ← BLACK                       ▷ Case 1
                       color[p[p[z]]] ← RED                   ▷ Case 1
                       z ← p[p[z]]                            ▷ Case 1
                  else if z = right[p[z]]
                          then z ← p[z]                       ▷ Case 3
                               LEFT-ROTATE(T, z)              ▷ Case 3
                       color[p[z]] ← BLACK                    ▷ Case 2
                       color[p[p[z]]] ← RED                   ▷ Case 2
                       RIGHT-ROTATE(T, p[p[z]])               ▷ Case 2
           else (same as then clause
                       with "right" and "left" exchanged)
color[root[T]] ← BLACK
```

Obnovení vlastností Red-black stromu

- **Existují 3 případy:**
- parent[*x*] a jeho bratr jsou červení
- parent[*x*] je červený, jeho bratr je černý, and *x* je levý syn svého rodiče
- jako v případě 2; ale *x* je pravý syn svého rodiče



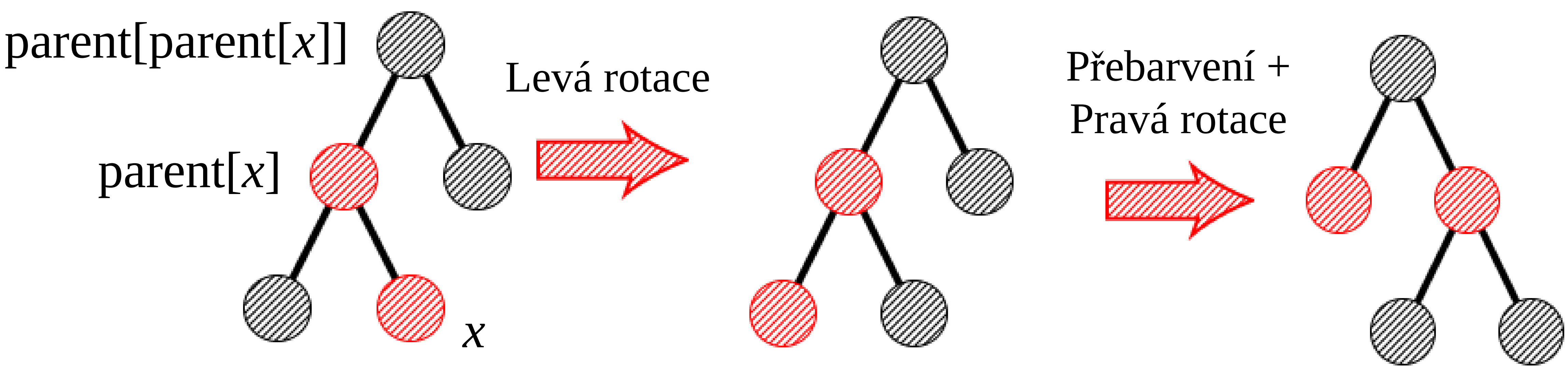
Případ 1



Případ 2

Obnovení vlastností Red-black stromu

- **Existují 3 případy:**
- parent[*x*] jeho bratr jsou červení
- parent[*x*] je červrný, and *x* je levý syn svého rodiče
- jako v případě 2; ale *x* je pravý syn svého rodiče



Případ 2

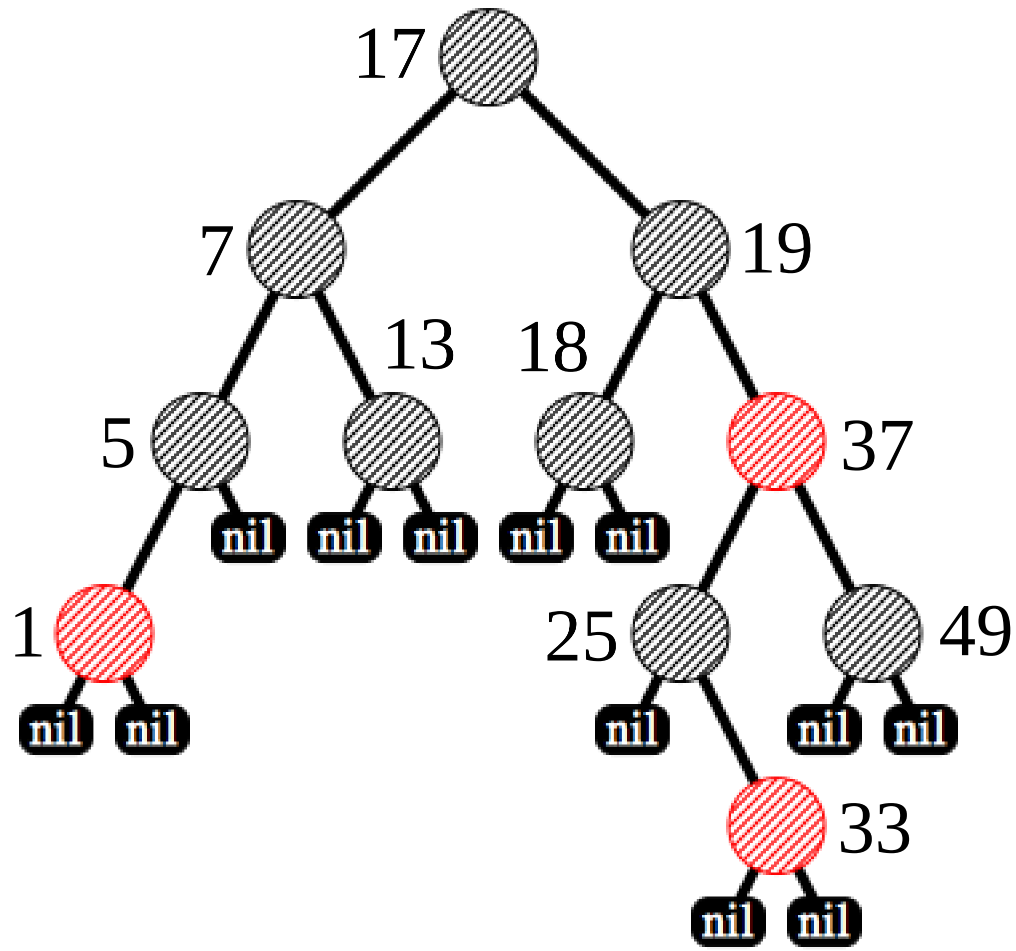
Případ 3

Vložení prvku do Red-black stromu - shrnutí

- **Pozorování:**
- **Každý z uvedených případů je proveden v konstantním čase**
- ***Případ 1* přesouvá *x* o dva kroky blíže ke kořeni a neprovádí se v něm žádné rotace – pouze se přebarvují uzly**
- **V *Případě 2* a 3, se provádí 1 nebo 2 rotace; pak úpravy končí**

Lemma: Vložení prvku do red-black stromu s n uzly má časovou složitost $O(\log n)$ a provádí se v něm pouze 2 rotace.

Zrušení uzlu v Red-Black stromu

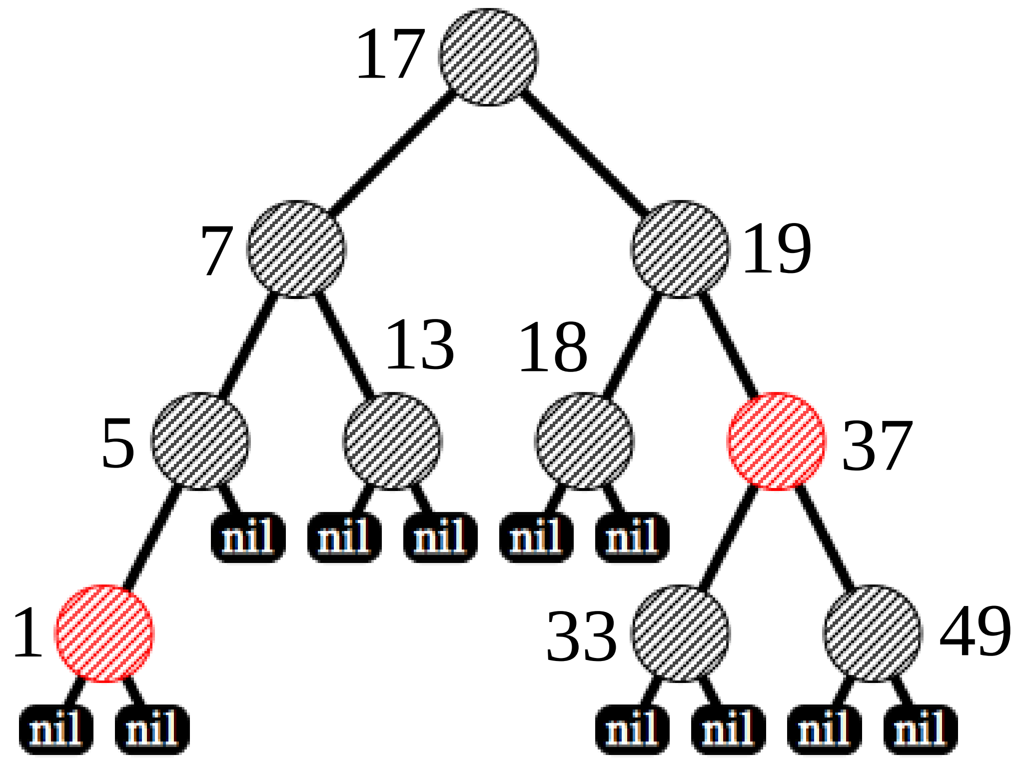


Rušený prvek: 25

Vlastnosti Red-Black stromů

- **Každý uzel stromu je obarven červenou nebo černou barvou.** ✔
- **Kořen stromu je obarven černě.** ✔
- **Listy (nil) jsou černé.** ✔
- **Červený uzel má pouze černé syny.** ✔
- **Na kterékoliv cestě z kořene do listu leží stejný počet černých uzlů** ✔

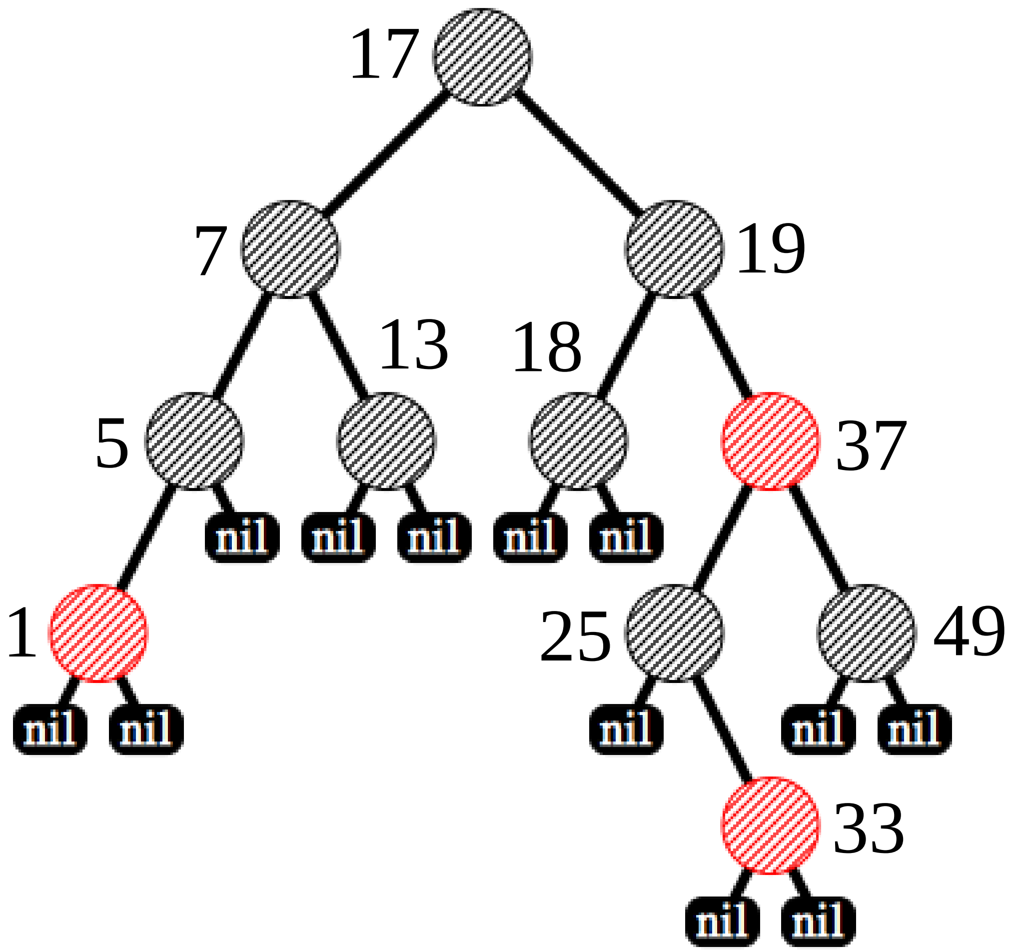
Vlastnosti Red-Black stromů



- **Každý uzel stromu je obarven červenou nebo černou barvou.** ✔
- **Kořen stromu je obarven černě.** ✔
- **Listy (nil) jsou černé.** ✔
- **Červený uzel má pouze černé syny.** ✔
- **Na kterékoliv cestě z kořene do listu leží stejný počet černých uzlů** ✔

Rušený prvek: 25

Zrušení uzlu v Red-Black stromu

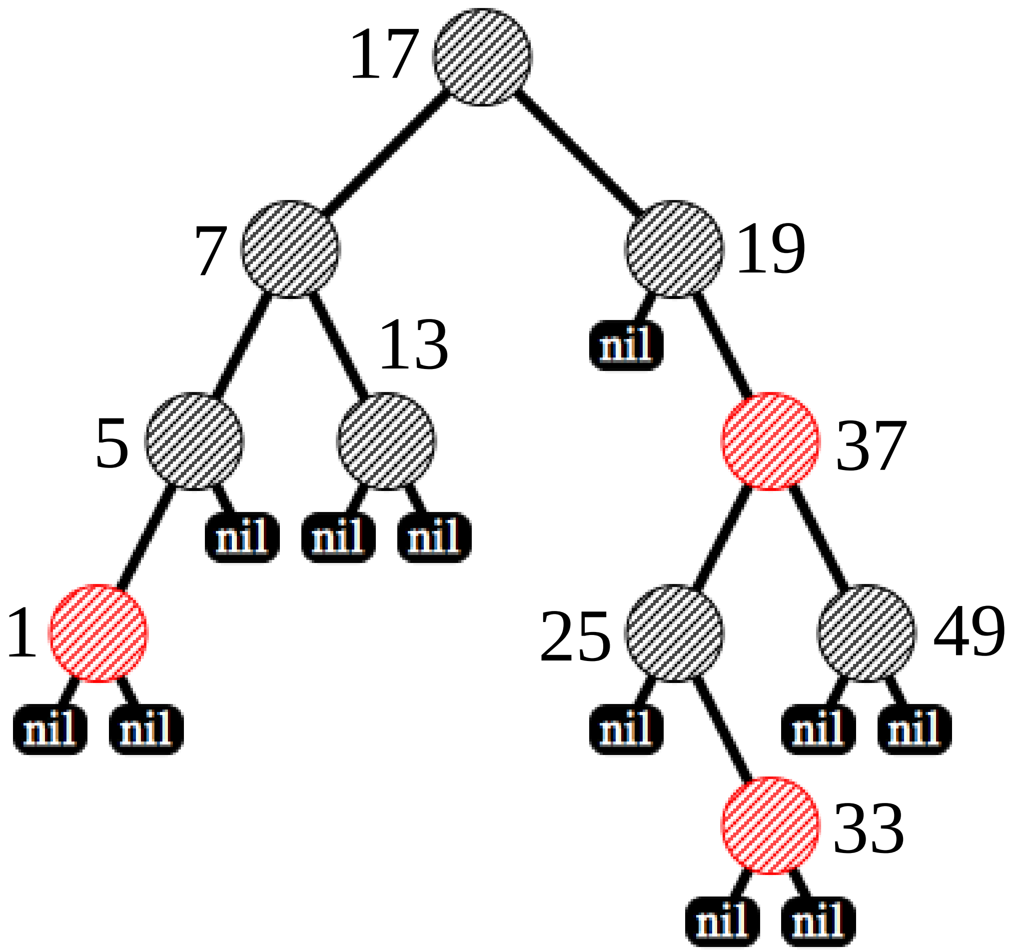


Vlastnosti Red-Black stromů

- **Každý uzel stromu je obarven červenou nebo černou barvou.** ✔
- **Kořen stromu je obarven černě.** ✔
- **Listy (nil) jsou černé.** ✔
- **Červený uzel má pouze černé syny.** ✔
- **Na kterékoliv cestě z kořene do listu leží stejný počet černých uzlů** ✔

Rušený prvek: 18

Zrušení uzlu v Red-Black stromu

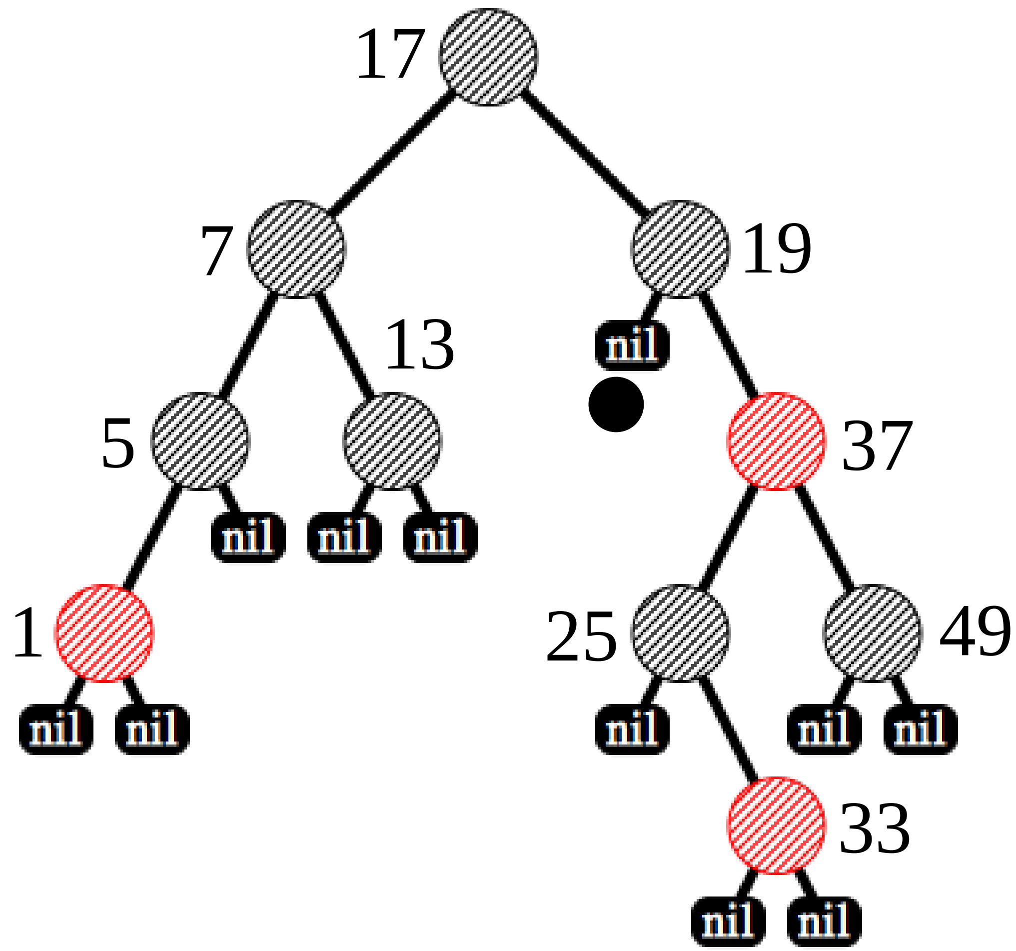


Vlastnosti Red-Black stromů

- **Každý uzel stromu je obarven červenou nebo černou barvou.** ✔
- **Kořen stromu je obarven černě.** ✔
- **Listy (nil) jsou černé.** ✔
- **Červený uzel má pouze černé syny.** ✔
- **Na kterékoliv cestě z kořene do listu leží stejný počet černých uzlů** ✔

Rušený prvek: 18

Zrušení uzlu v Red-Black stromu



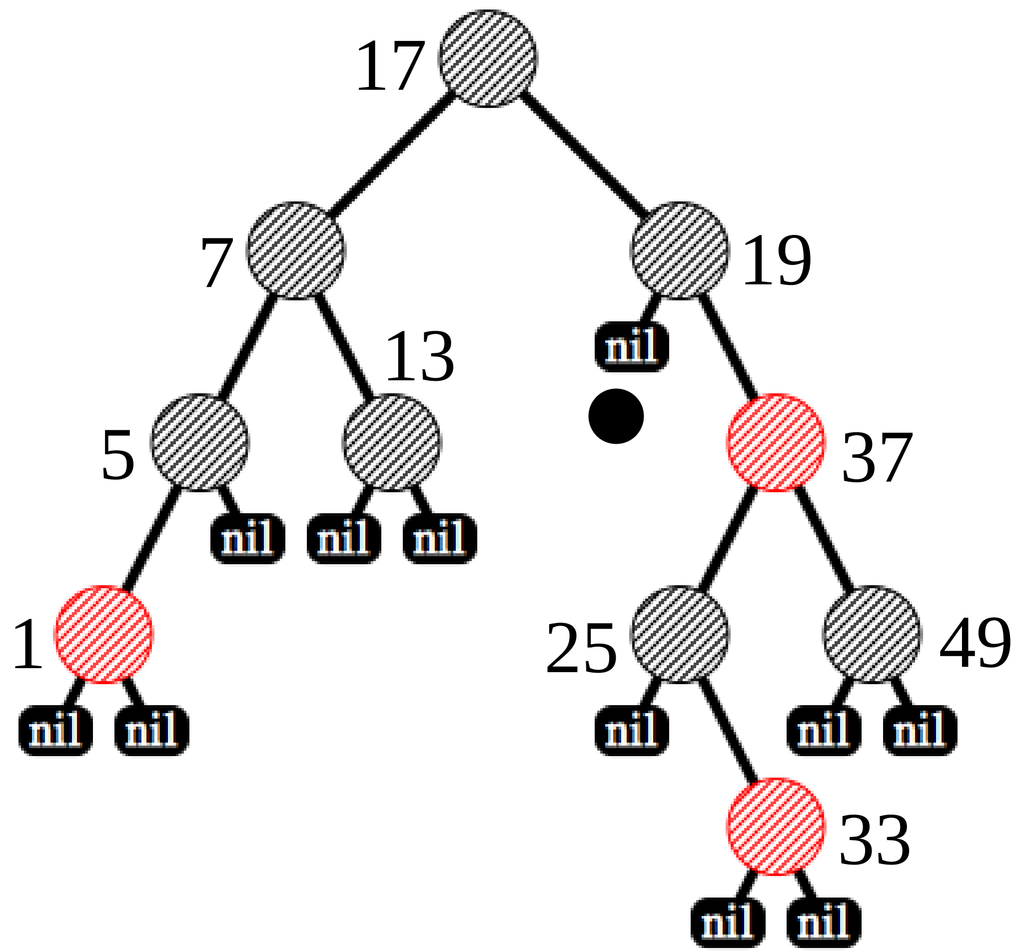
Rušený prvek: 18

Vlastnosti Red-Black stromů

- **Každý uzel stromu je obarven červenou nebo černou barvou.** ✔
- **Kořen stromu je obarven černě.** ✔
- **Listy (nil) jsou černé.** ✔
- **Červený uzel má pouze černé syny.** ✔
- **Na kterékoliv cestě z kořene do listu leží stejný počet černých uzlů** ✔

První krok: Označíme syna zrušeného uzlu jako extra černý ("doubly black")

Zrušení uzlu v Red-Black stromu



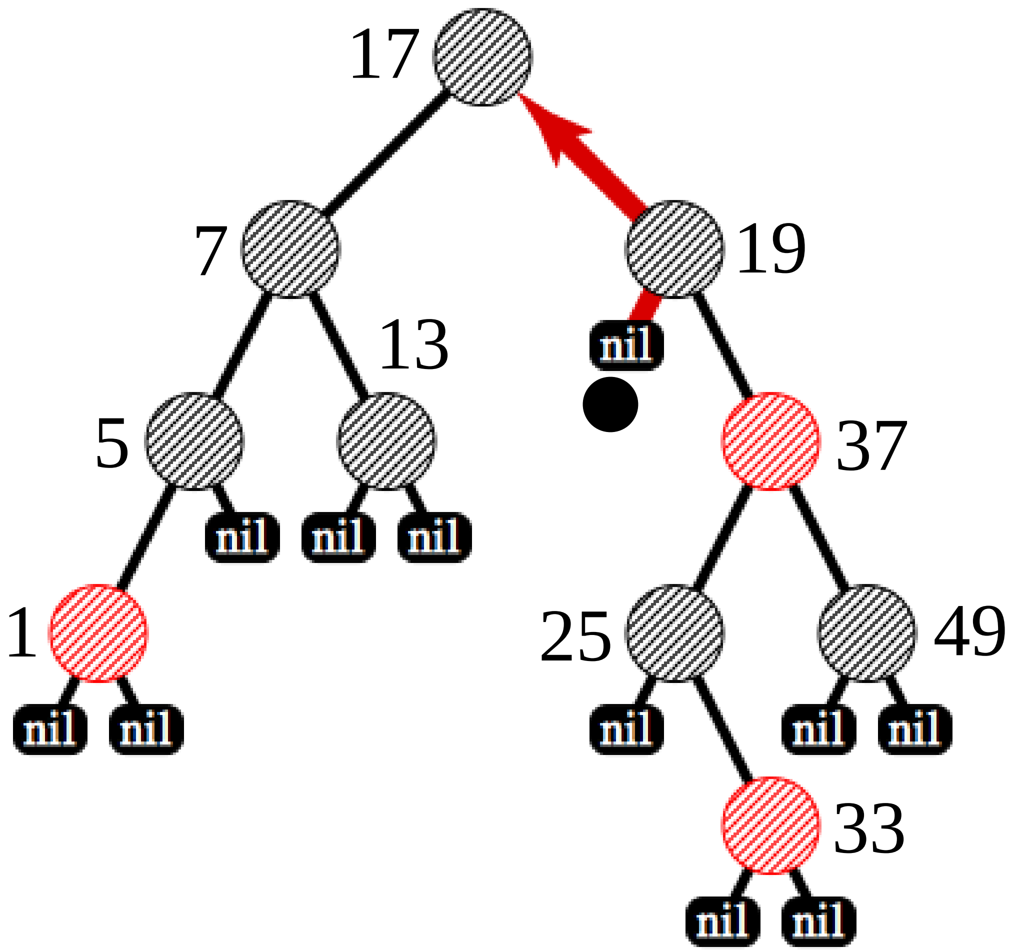
Rušený prvek: 18

Vlastnosti Red-Black stromů

Barva neodpovídá

- **Každý uzel stromu je obarven červenou nebo černou barvou.** ✘
- **Kořen stromu je obarven černě.** ✔
- **Listy (nil) jsou černé.** ✔
- **Červený uzel má pouze černé syny.** ✔
- **Na kterékoliv cestě z kořene do listu leží stejný počet černých uzlů** ✔

Zrušení uzlu v Red-Black stromu

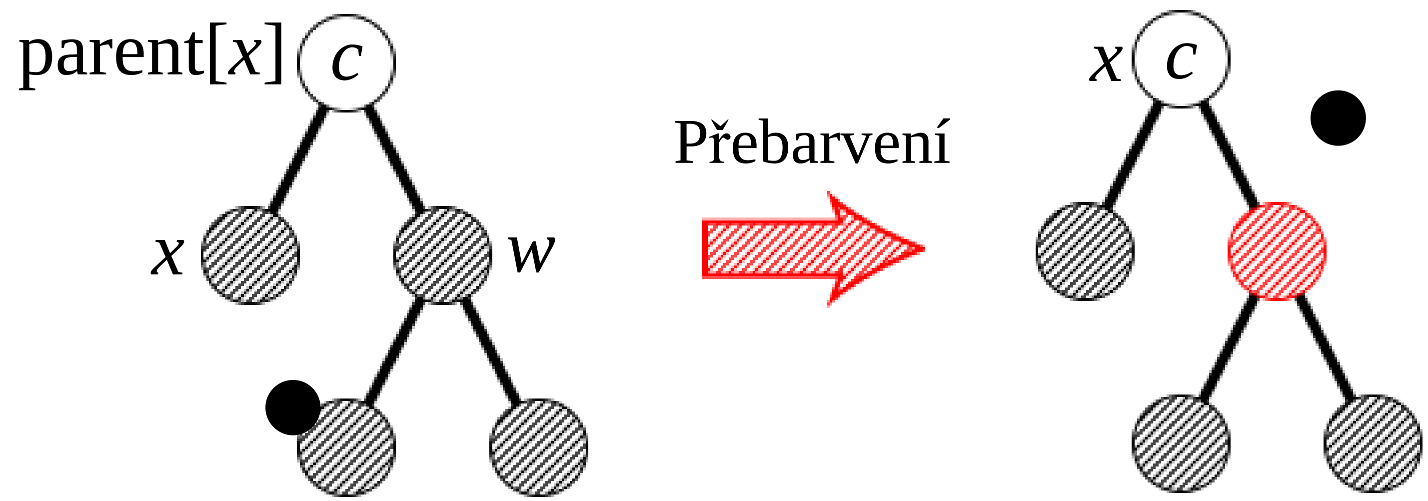


Rušený prvek: 18

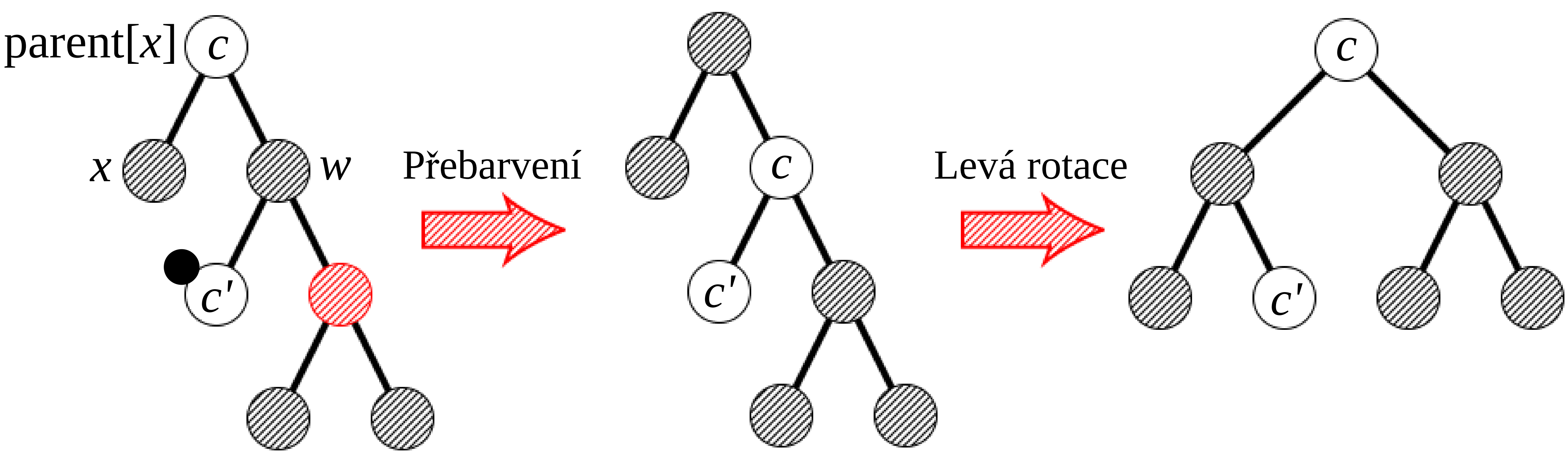
Korekce barvy uzlů v RB stromu

Předpokládejme že *x* je levý syn svého rodiče a jeho bratr *w* je černý.

- **Existují 3 případy, závisející na barvě synů w:**
- Oba synové w jsou černí
- Pravý syn w je červený
- Levý syn w je červený a pravý je černý



Případ 1

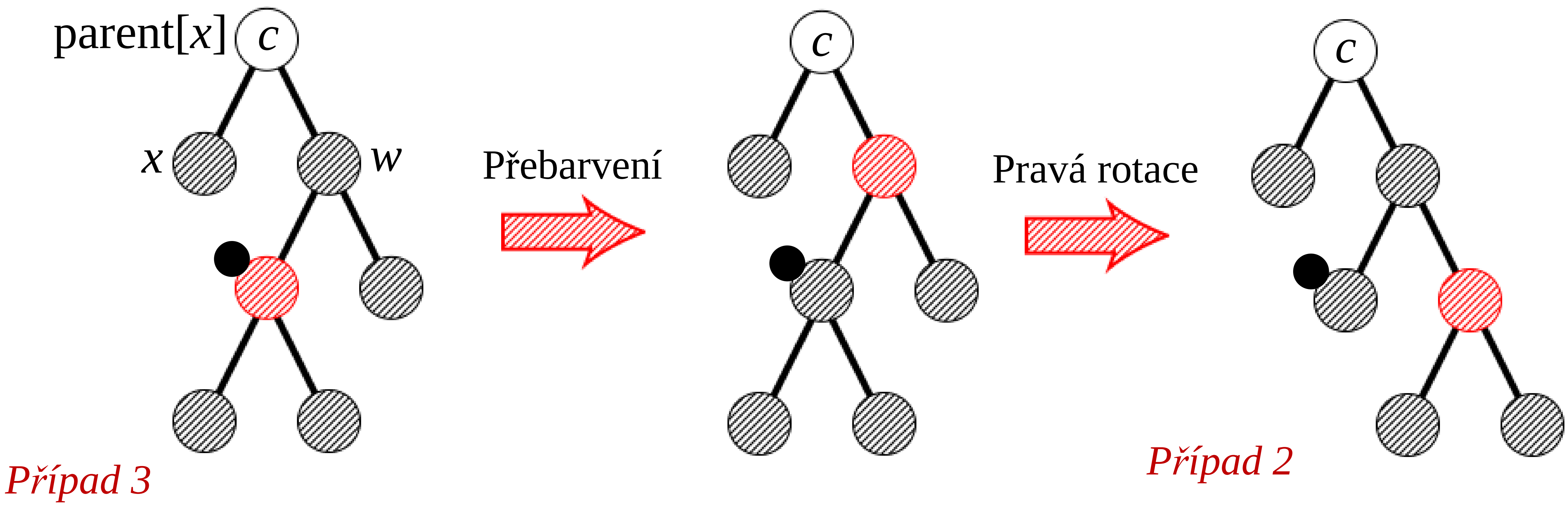


Případ 2

Korekce barvy uzlů v RB stromu

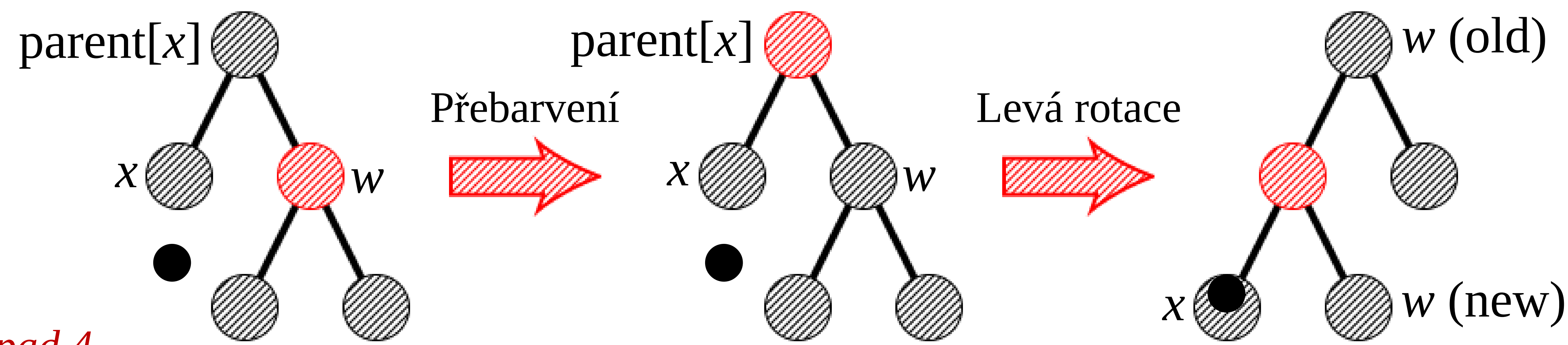
Předpokládejme že *x* je levý syn svého rodiče a jeho bratr *w* je černý.

- **Existují 3 případy, závisející na barvě synů w:**
- Oba synové w jsou černí
- Pravý syn w je červený
- Levý syn w je červený a pravý je černý



Korekce barvy uzlů v RB stromu

Případ 4: Pravý bratr *w* uzlu *x* je červený.



Případ 4

- **Pozorování:**
- Případy 2 a 3 provádí maximálně 2 rotace; pak je vše hotovo
- Případ 1 pouze přebarvuje a přesouvá korekci o jeden krok blíže ke kořeni
- Případ 4 provádí pouze jedinou rotaci a přesouvá korekci o **jeden krok dále od kořene!**

Lemma: Zrušení uzlu v red-black stromu s *n* uzly má časovou složitost $O(\log n)$ a provádí maximálně tři rotace.

Algoritmus zrušení prvku RB stromu

```
RB-DELETE(T, z)
if left[z] = nil[T] or right[z] = nil[T]
    then y ← z
    else y ← TREE-SUCCESSOR(z)
if left[y] ≠ nil[T]
    then x ← left[y]
    else x ← right[y]
p[x] ← p[y]
if p[y] = nil[T]
    then root[T] ← x
    else if y = left[p[y]]
            then left[p[y]] ← x
            else right[p[y]] ← x
if y ≠ z
    then key[z] ← key[y]
         copy y's satellite data into z
if color[y] = BLACK
    then RB-DELETE-FIXUP(T, x)
return y
```

Algoritmus zrušení prvku RB stromu

```
RB-DELETE-FIXUP(T, x)
while x ≠ root[T] and color[x] = BLACK
    do if x = left[p[x]]
          then w ← right[p[x]]
               if color[w] = RED
                  then color[w] ← BLACK                                  ▷ Case 4
                       color[p[x]] ← RED                                 ▷ Case 4
                       LEFT-ROTATE(T, p[x])                              ▷ Case 4
                       w ← right[p[x]]                                   ▷ Case 4
               if color[left[w]] = BLACK and color[right[w]] = BLACK
                  then color[w] ← RED                                    ▷ Case 1
                       x ← p[x]                                          ▷ Case 1
                  else if color[right[w]] = BLACK
                          then color[left[w]] ← BLACK                    ▷ Case 3
                               color[w] ← RED                            ▷ Case 3
                               RIGHT-ROTATE(T, w)                        ▷ Case 3
                               w ← right[p[x]]                           ▷ Case 3
                       color[w] ← color[p[x]]                            ▷ Case 2
                       color[p[x]] ← BLACK                               ▷ Case 2
                       color[right[w]] ← BLACK                           ▷ Case 2
                       LEFT-ROTATE(T, p[x])                              ▷ Case 2
                       x ← root[T]                                       ▷ Case 2
          else (same as then clause with "right" and "left" exchanged)
color[x] ← BLACK
```