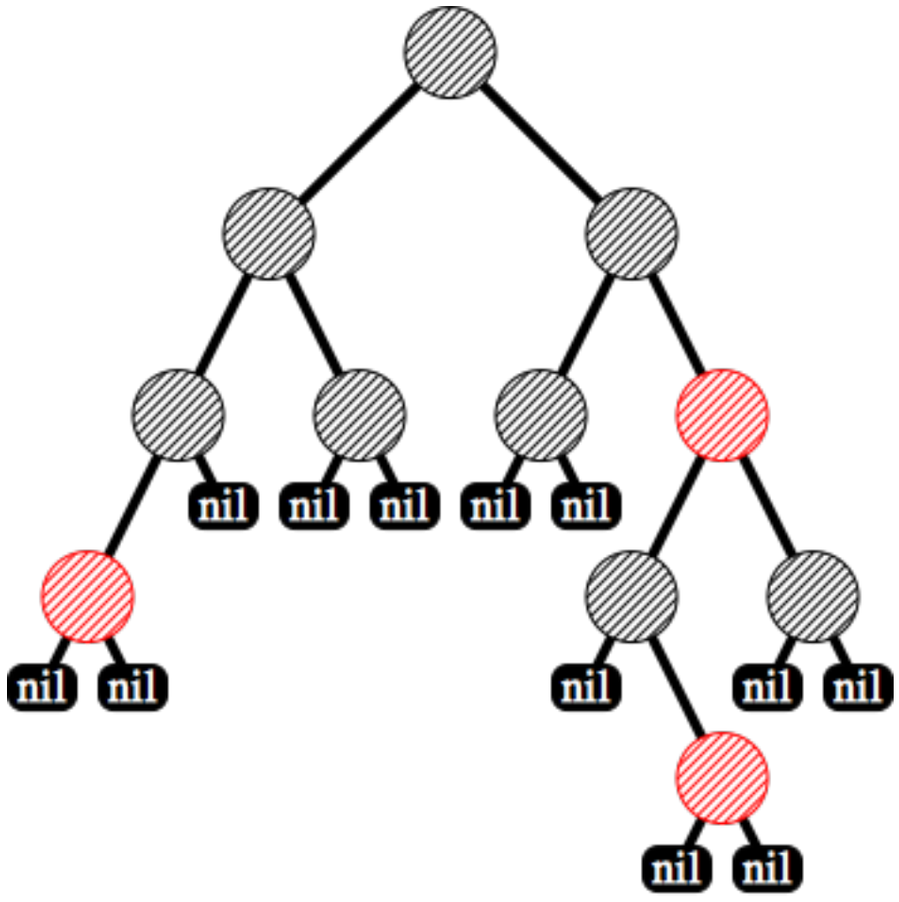
# Red-Black Stromy

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

# Vlastnosti Red-Black Stromů

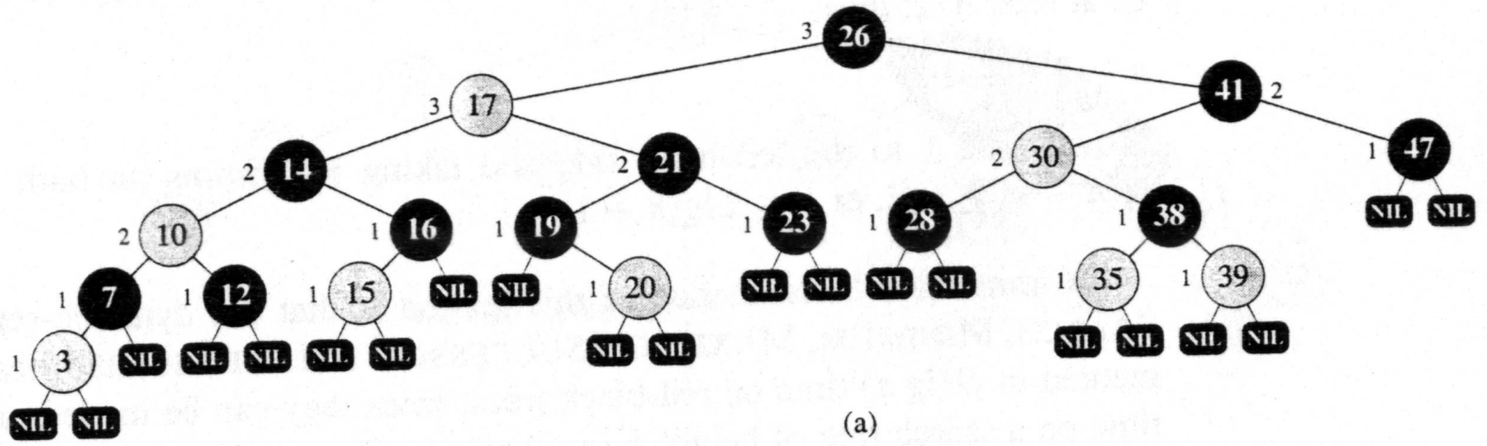



## Vlastnosti Red-Black stromů

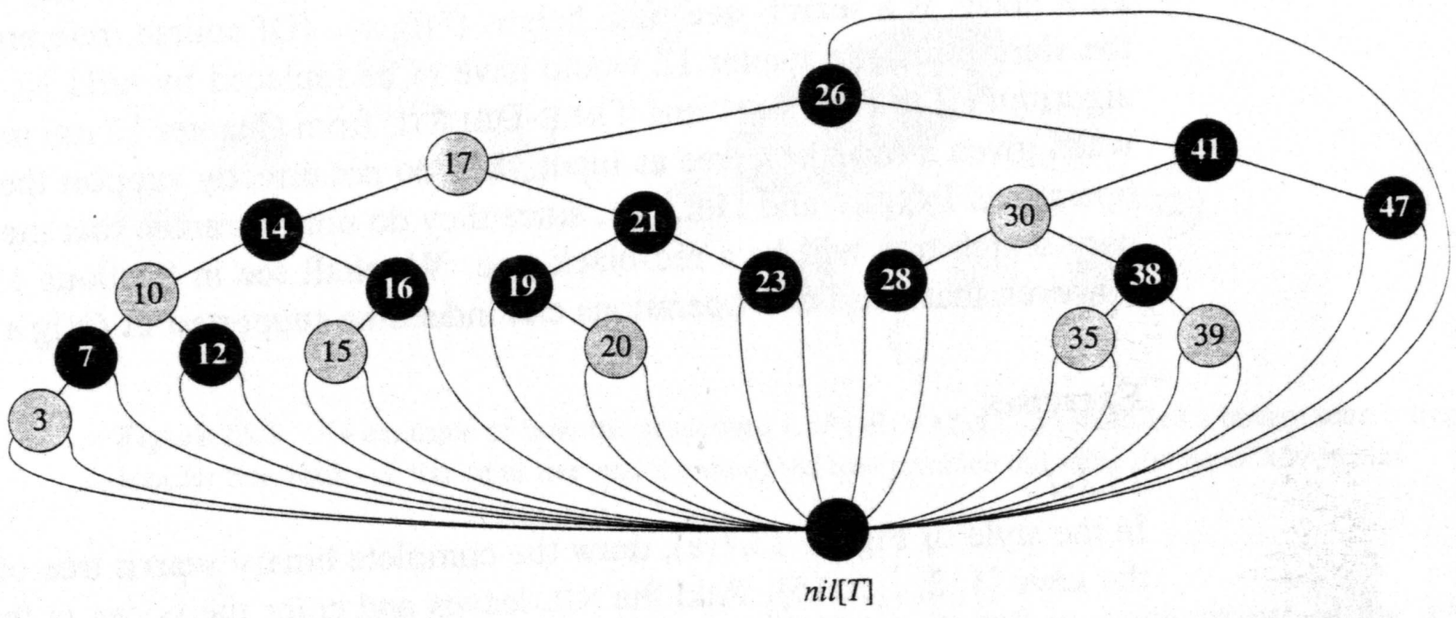
- **Každý uzel stromu je obarven červenou nebo černou barvou.**
- **Kořen stromu je obarven černě.**
- **Listy (nil) jsou černé.**
- **Červený uzel má pouze černé syny.**
- **Na kterékoliv cestě z kořene do listu leží stejný počet černých uzlů**

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.

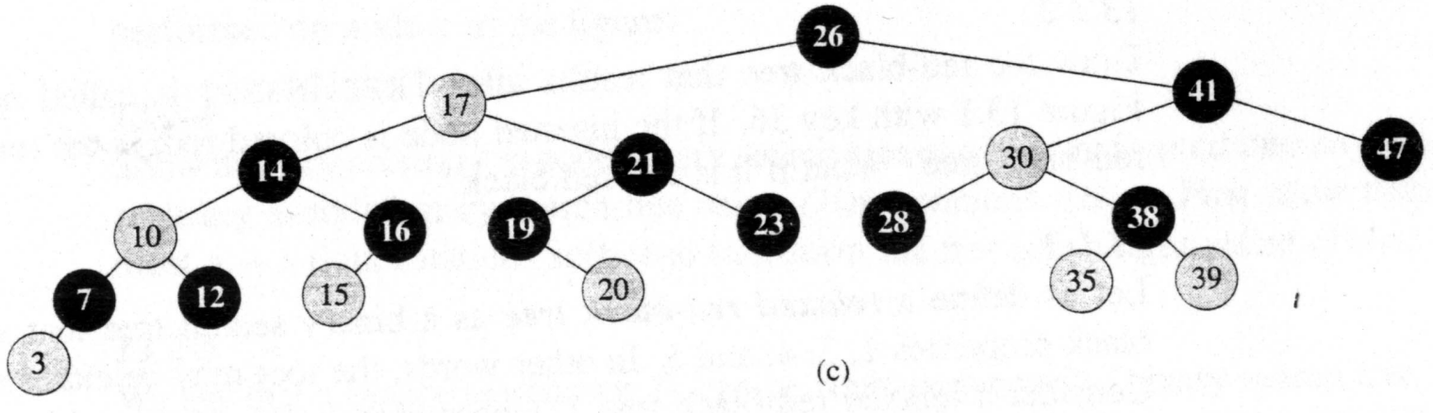
# RB strom a implementace




(a)

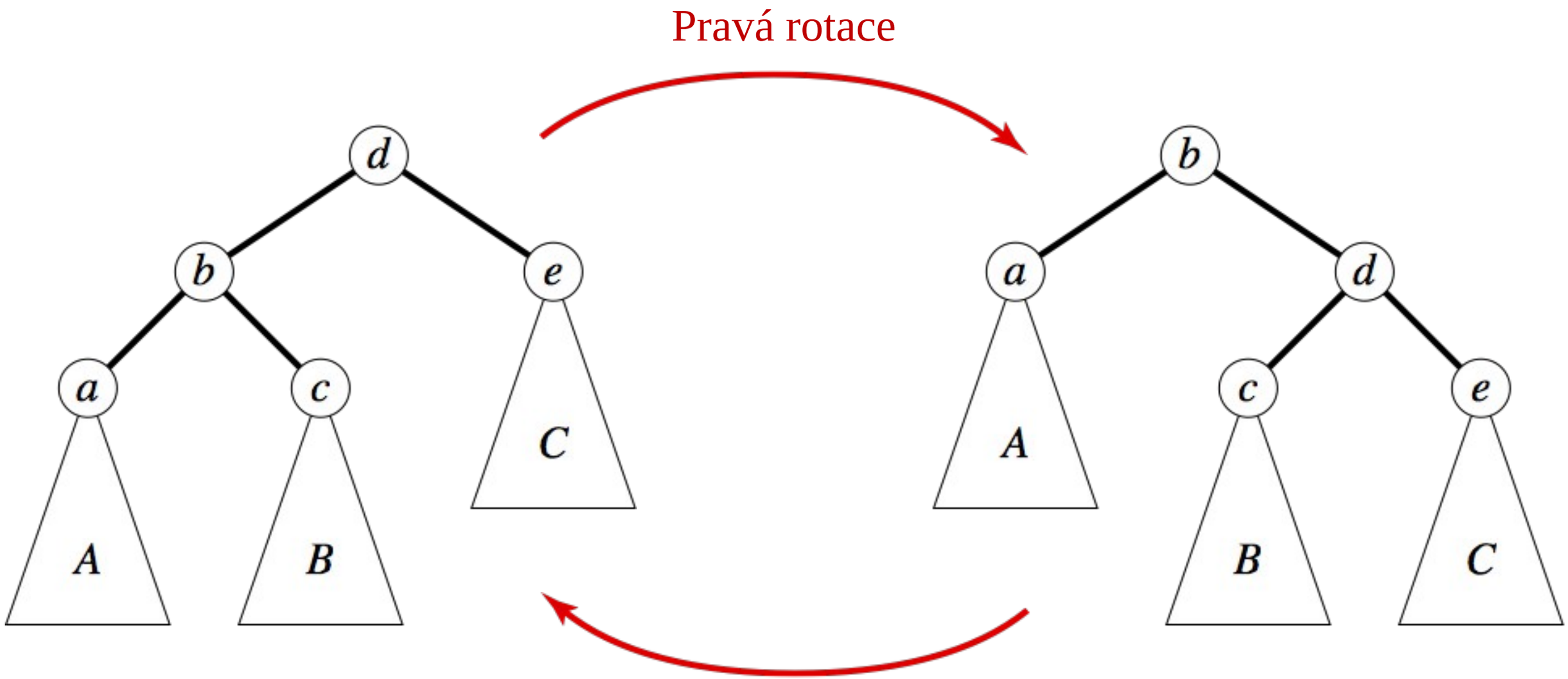



(b)

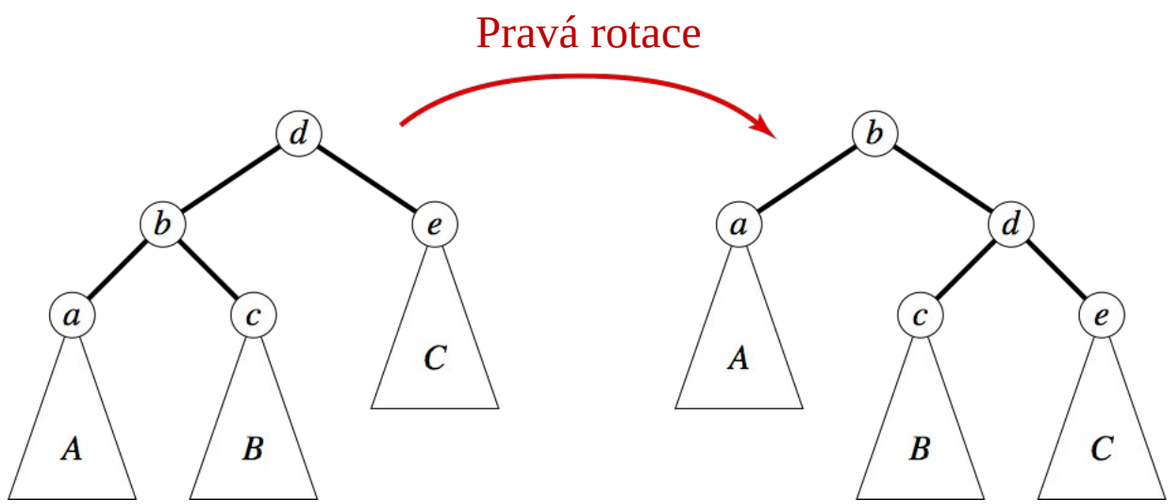


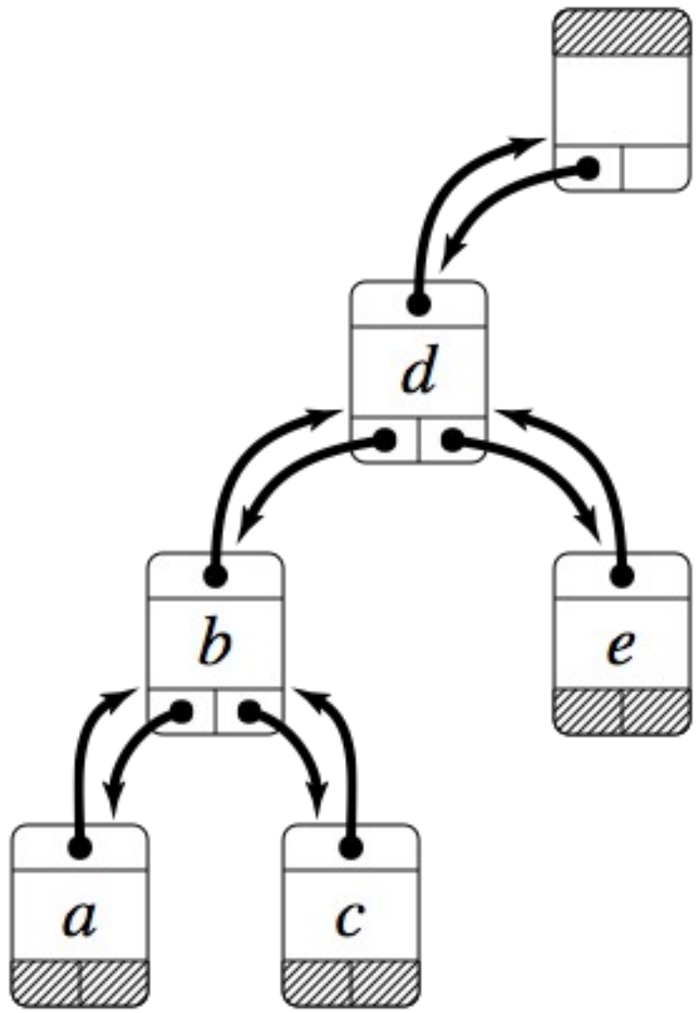

(c)

# Rotace

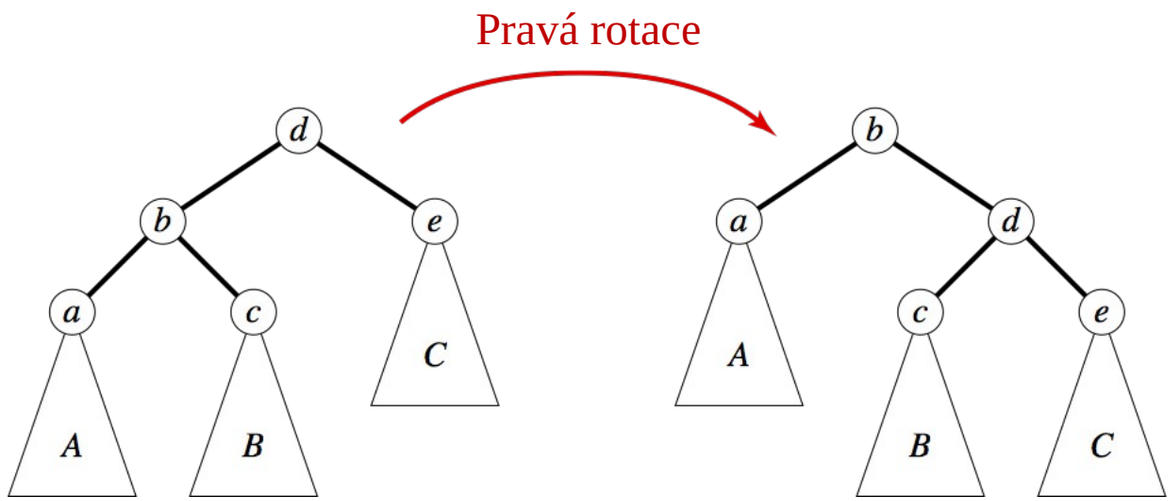


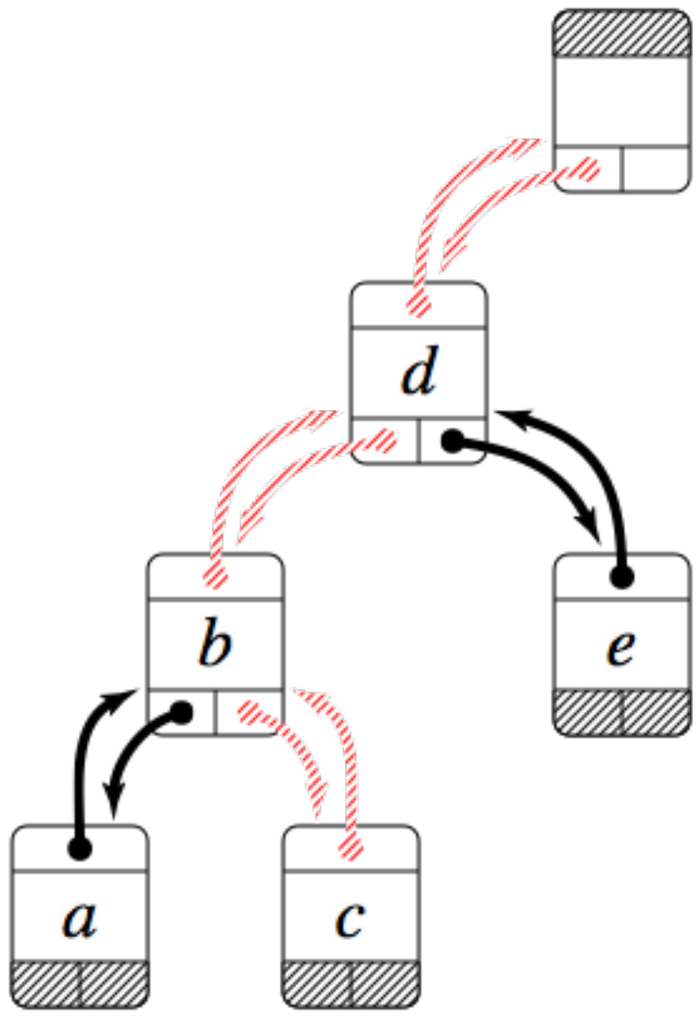
# Operace rotace se provádí v konstantním čase



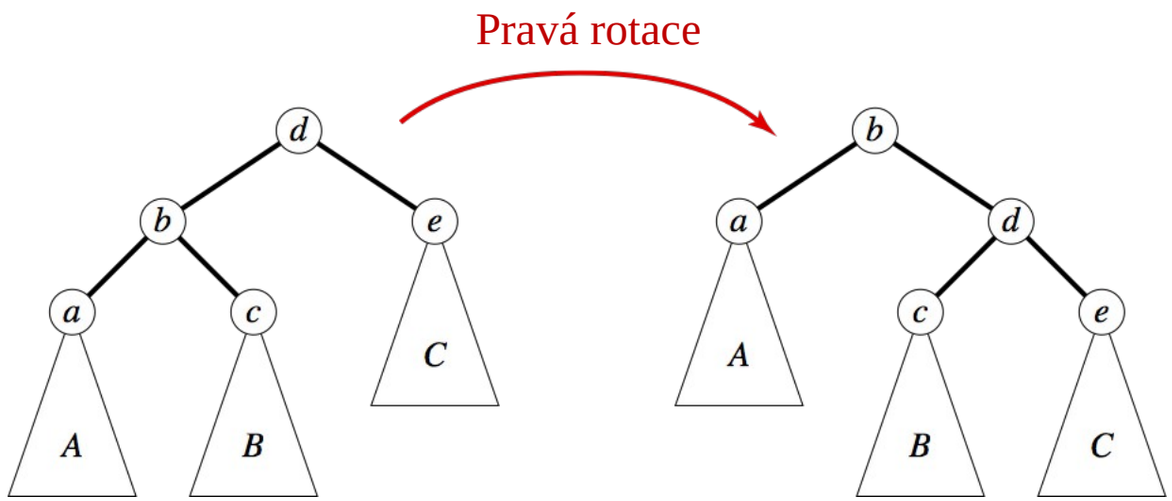



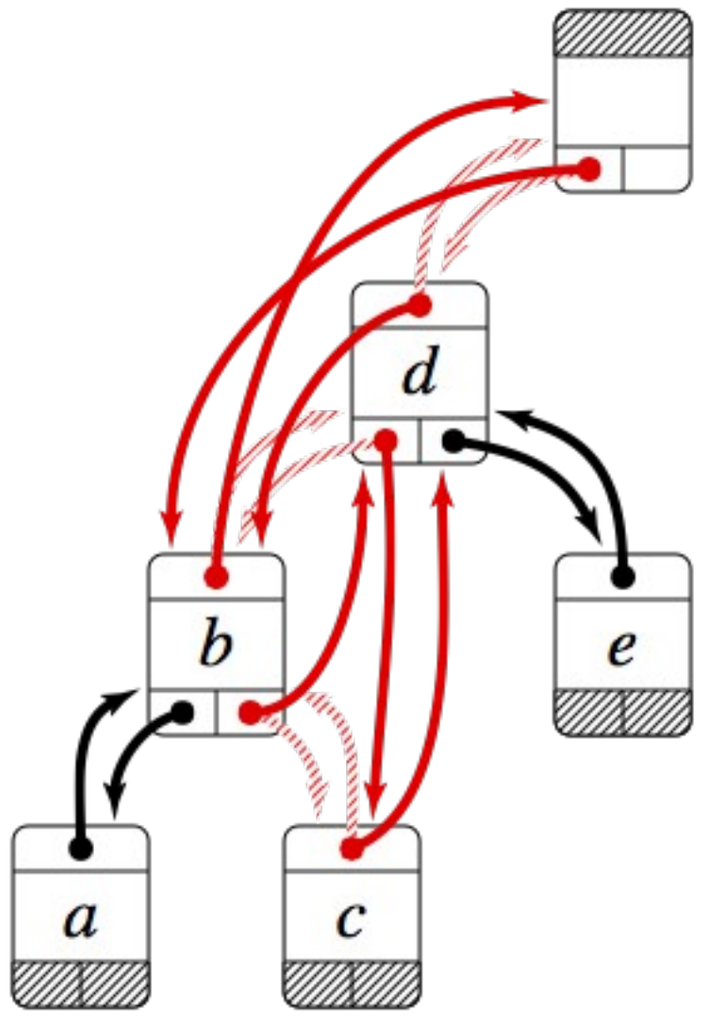
# Operace rotace se provádí v konstantním čase



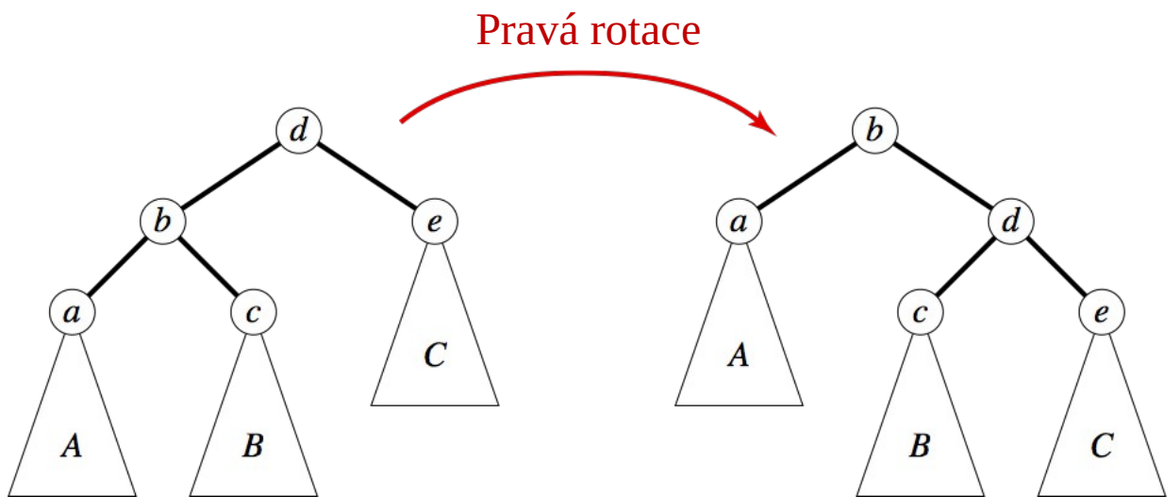



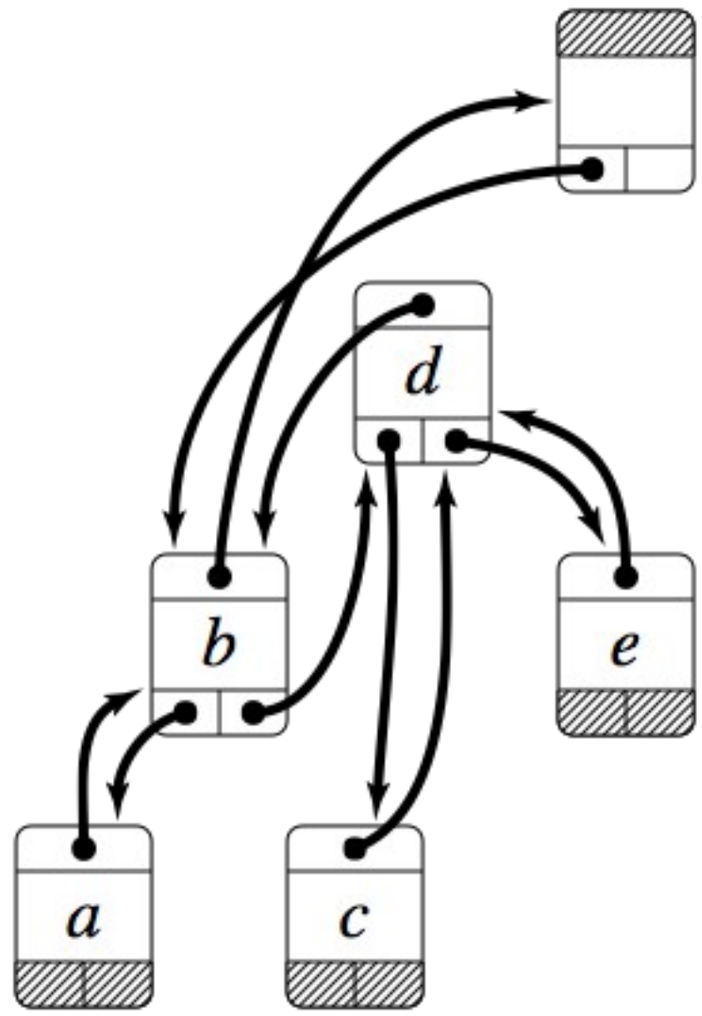
# Operace rotace se provádí v konstantním čase



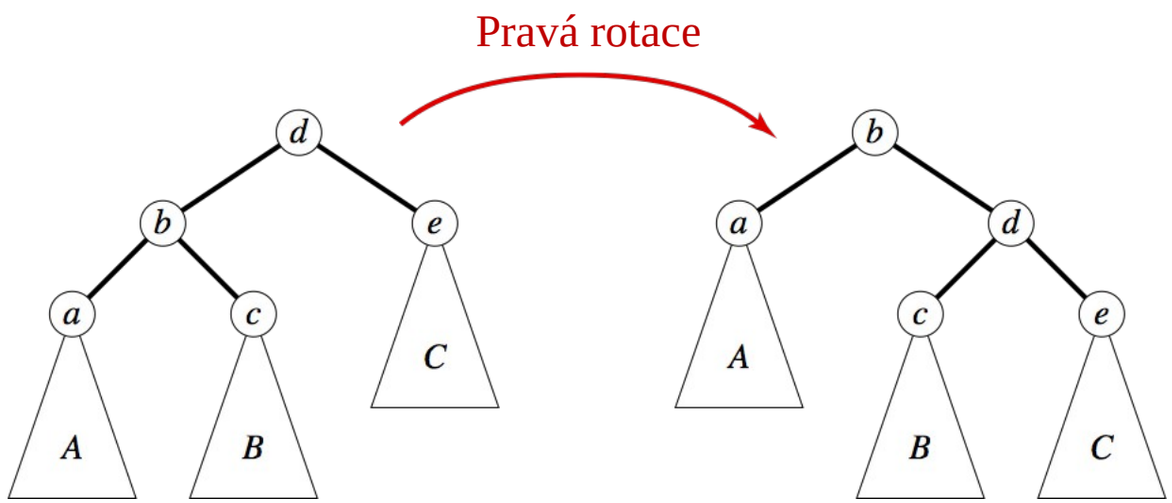



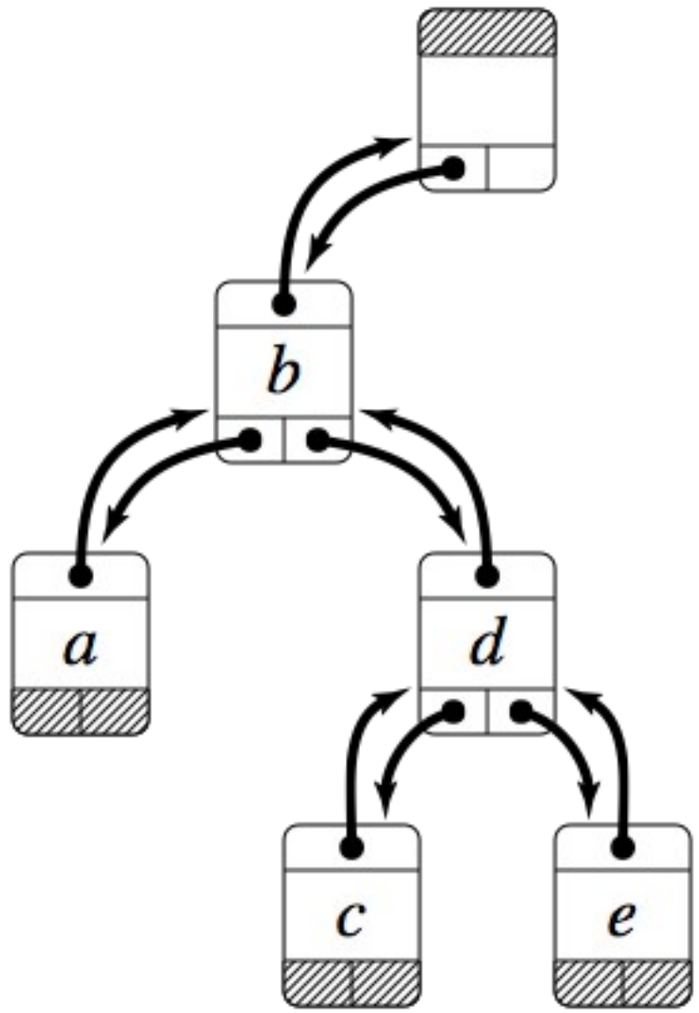
# Operace rotace se provádí v konstantním čase



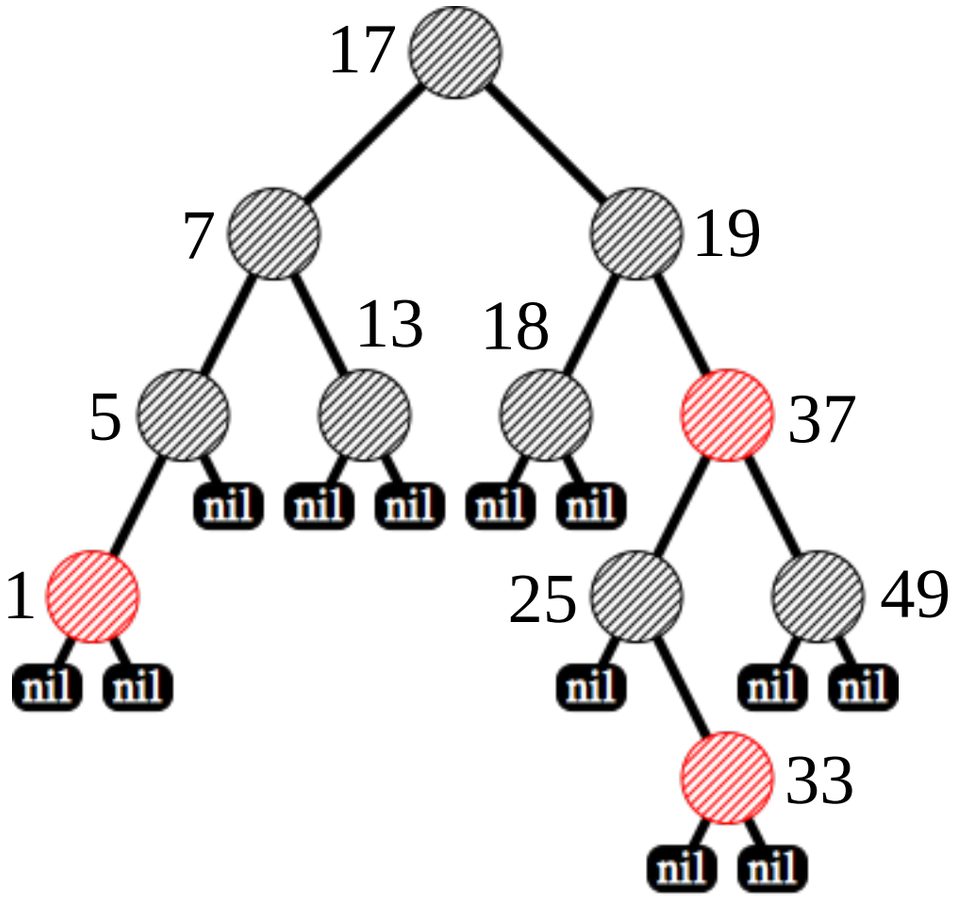



# Operace rotace se provádí v konstantním čase



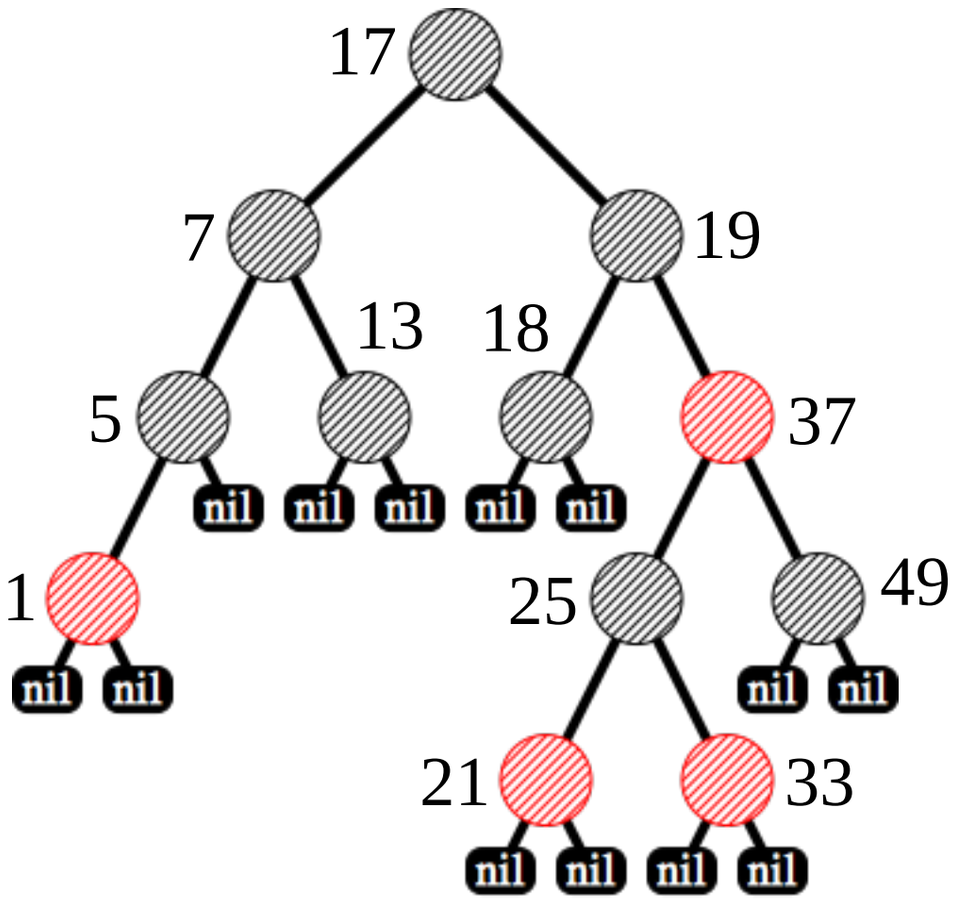



# Vložení prvku do  Red-Black stromu




Vkládaný prvek:  21

# Vložení prvku do Red-Black stromu

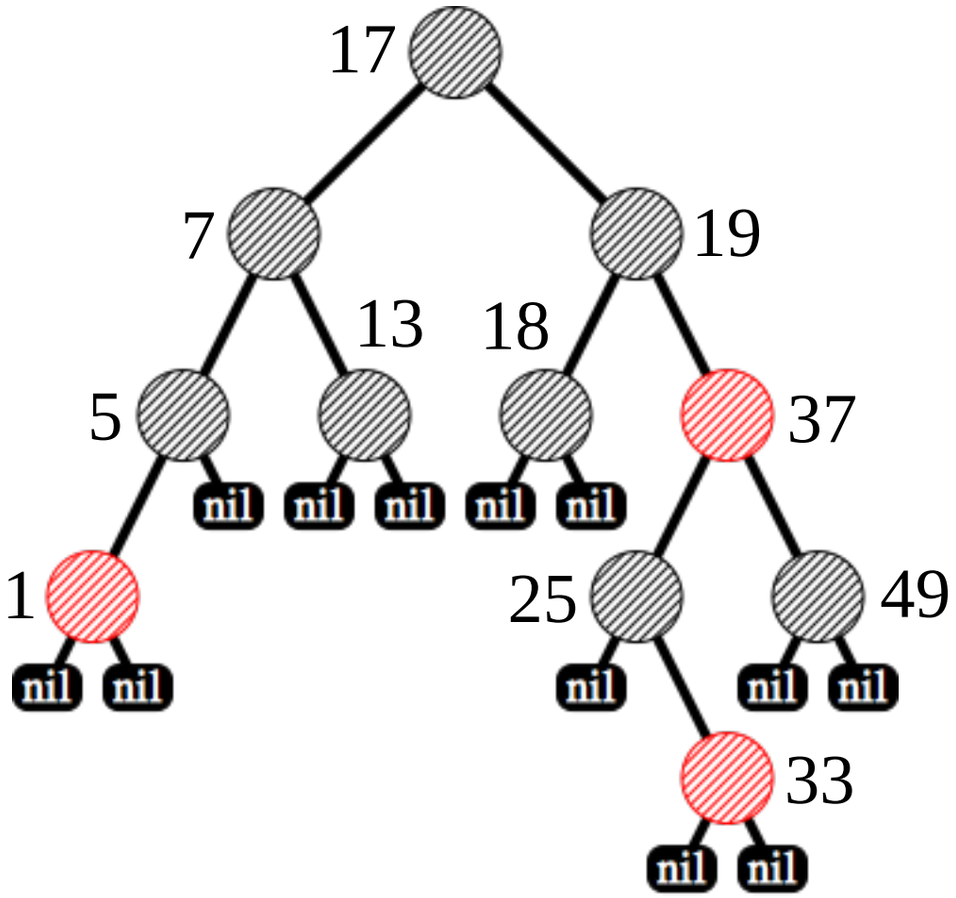



## Vlastnosti Red-Black stromů

- **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** ✔

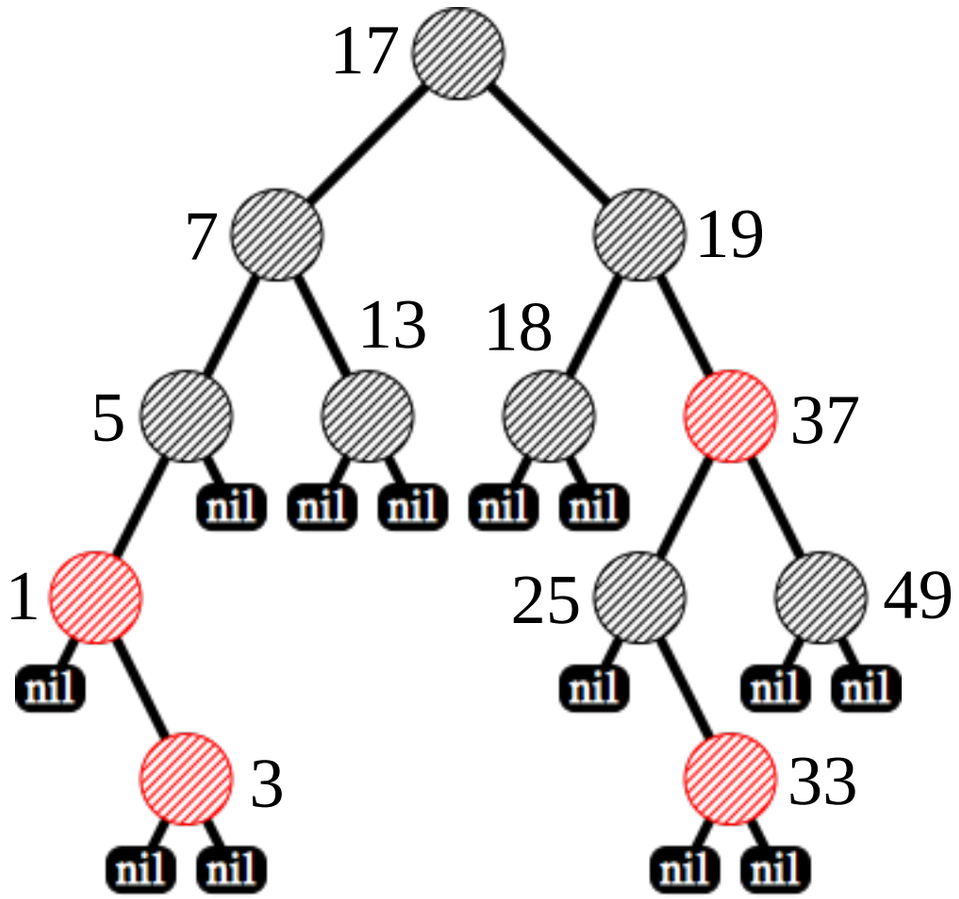
Vkládaný prvek: 21

# 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.** ✔

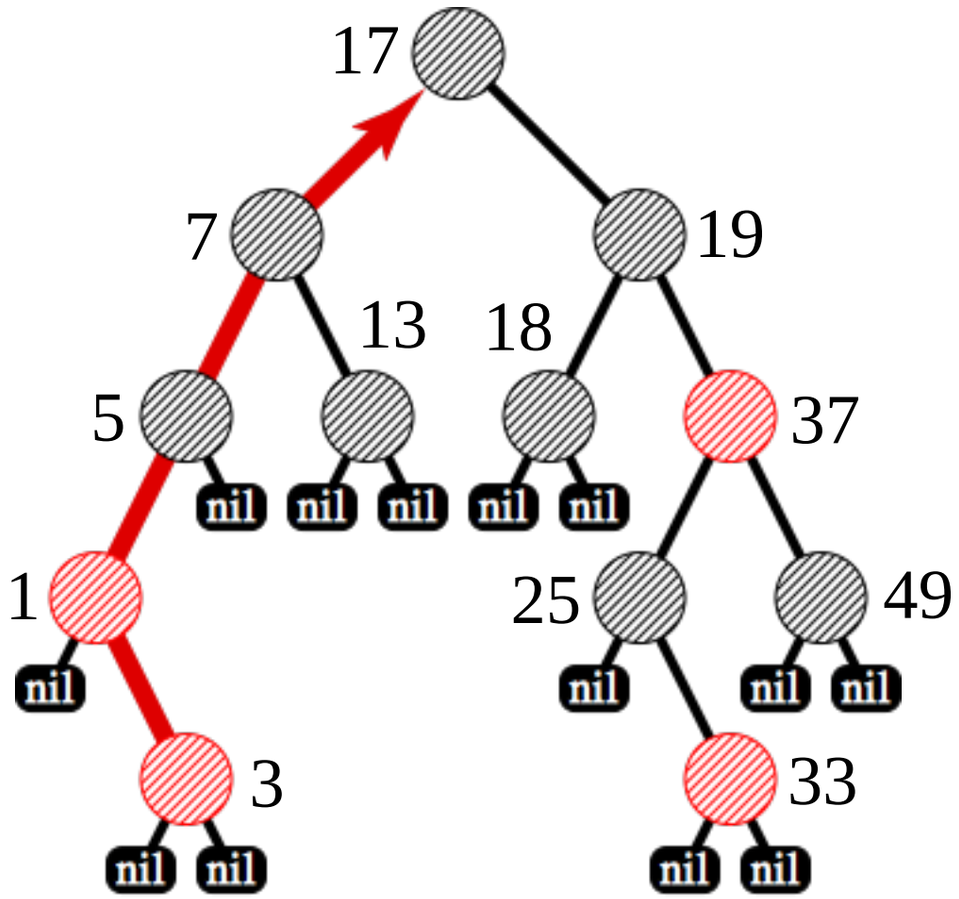
**Problém dvou červených uzlů**

- **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

# 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** ✔

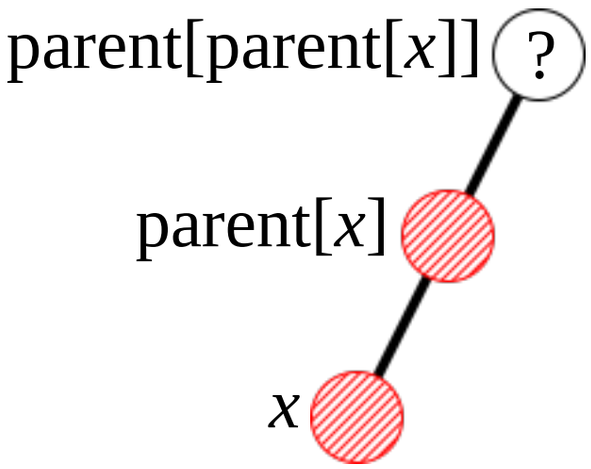
**Cíl:** Obnovit vlastnosti **Red-black** stromu přebarvením uzlů popř. provedením rotací

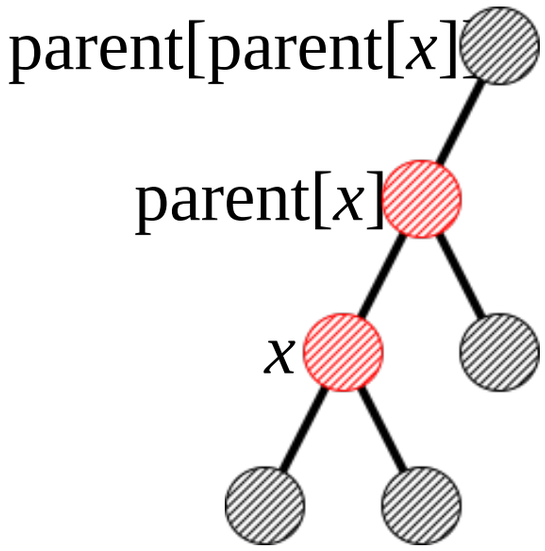
# Obnovení vlastností Red-black stromu

- **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.**



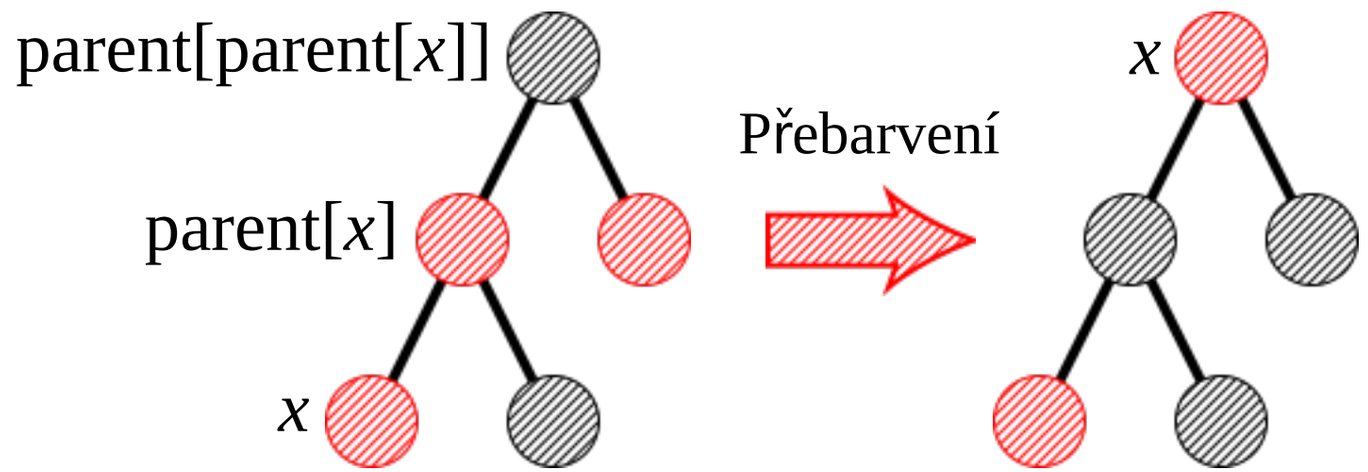
# Algoritmus vložená prvků do RB stromu

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

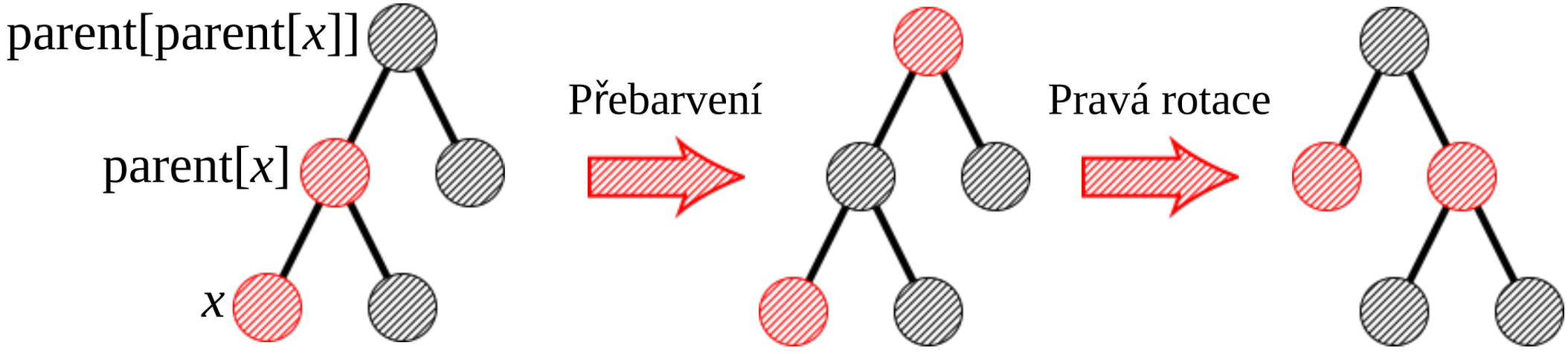
```
RB-INSERT-FIXUP(T, z)
 1  while color[p[z]] = RED
 2      do if p[z] = left[p[p[z]]]
 3            then y ← right[p[p[z]]]
 4                 if color[y] = RED
 5                    then color[p[z]] ← BLACK                    ▷ Case 1
 6                         color[y] ← BLACK                  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



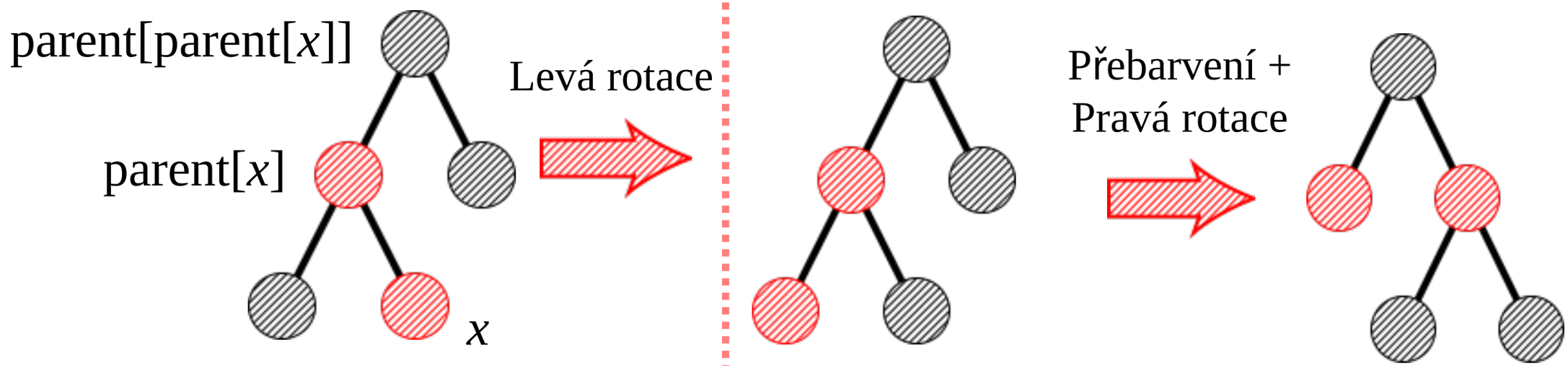

*Případ 1*




*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

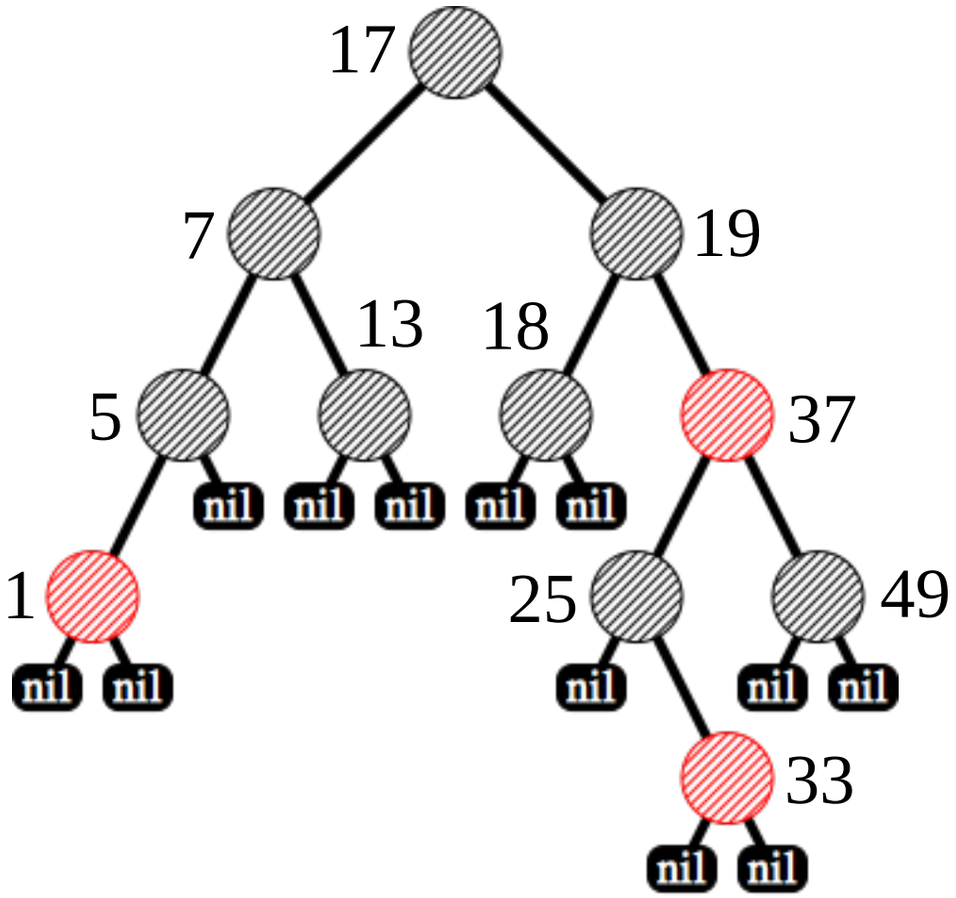


# Vložení prvku do Red-black stromu - shrnutí

- **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

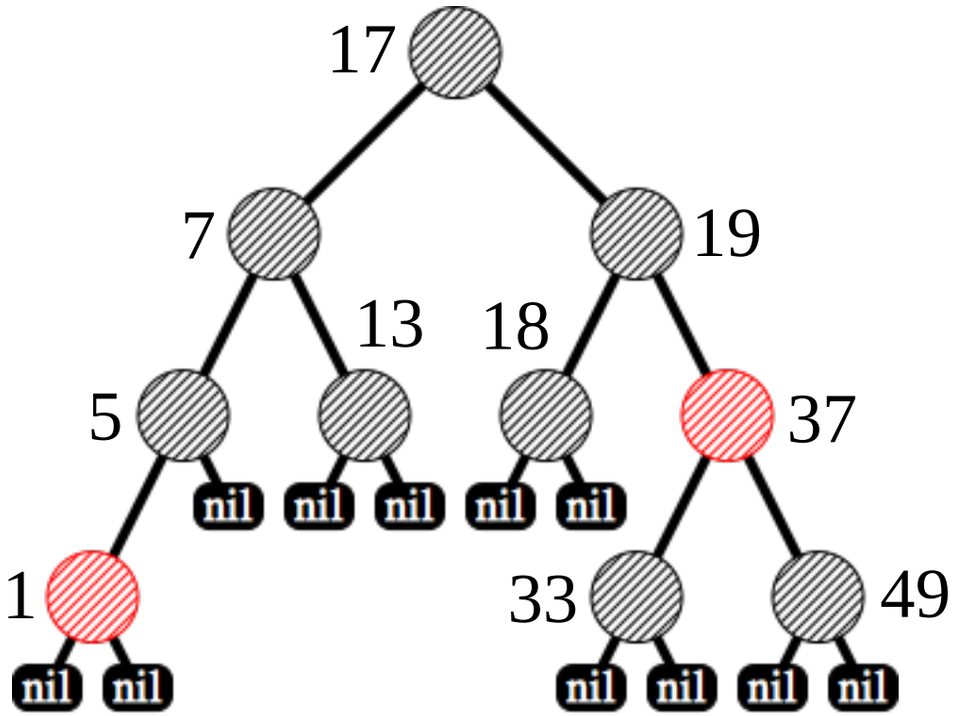



## Vlastnosti Red-Black stromů

- **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

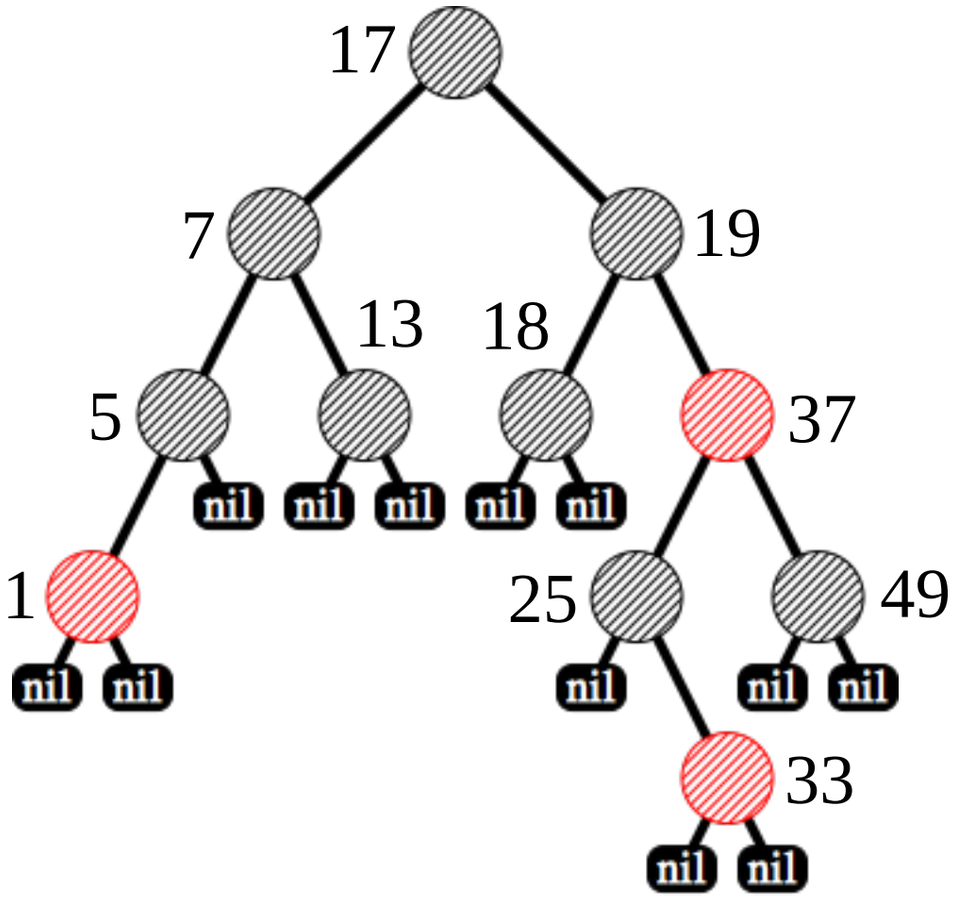



## Vlastnosti Red-Black stromů

- **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:  25

# Zrušení uzlu v Red-Black stromu

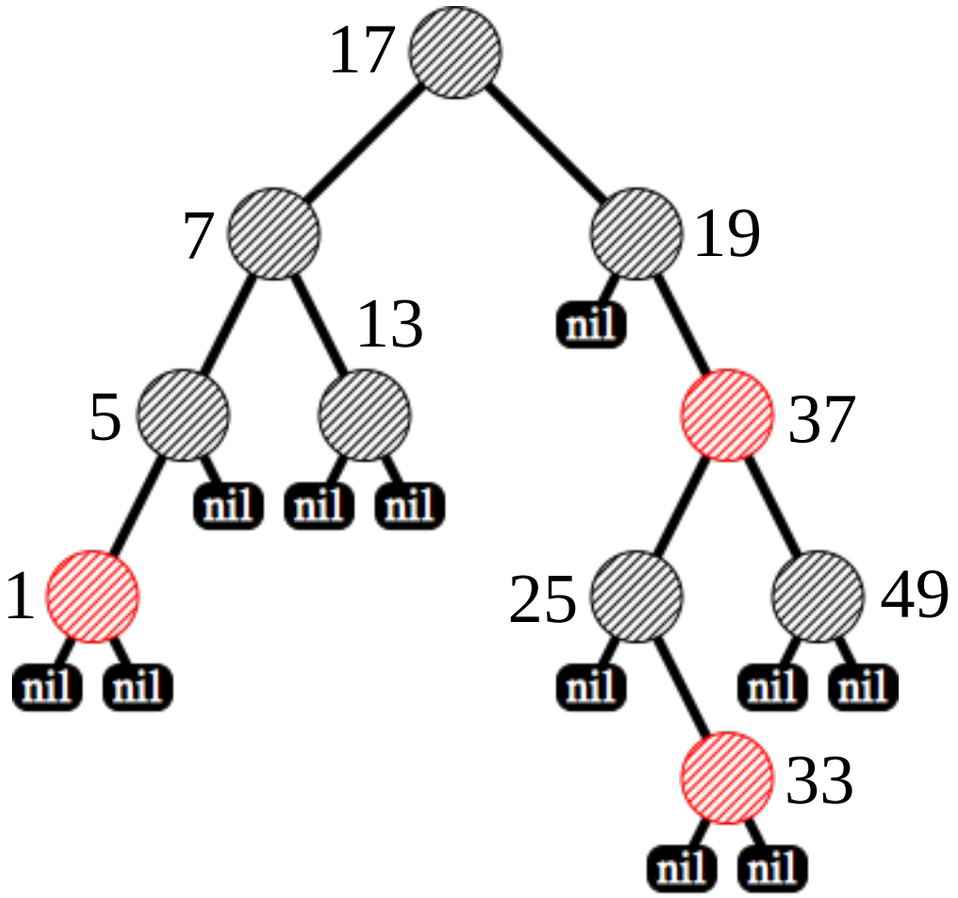



## Vlastnosti Red-Black stromů

- **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

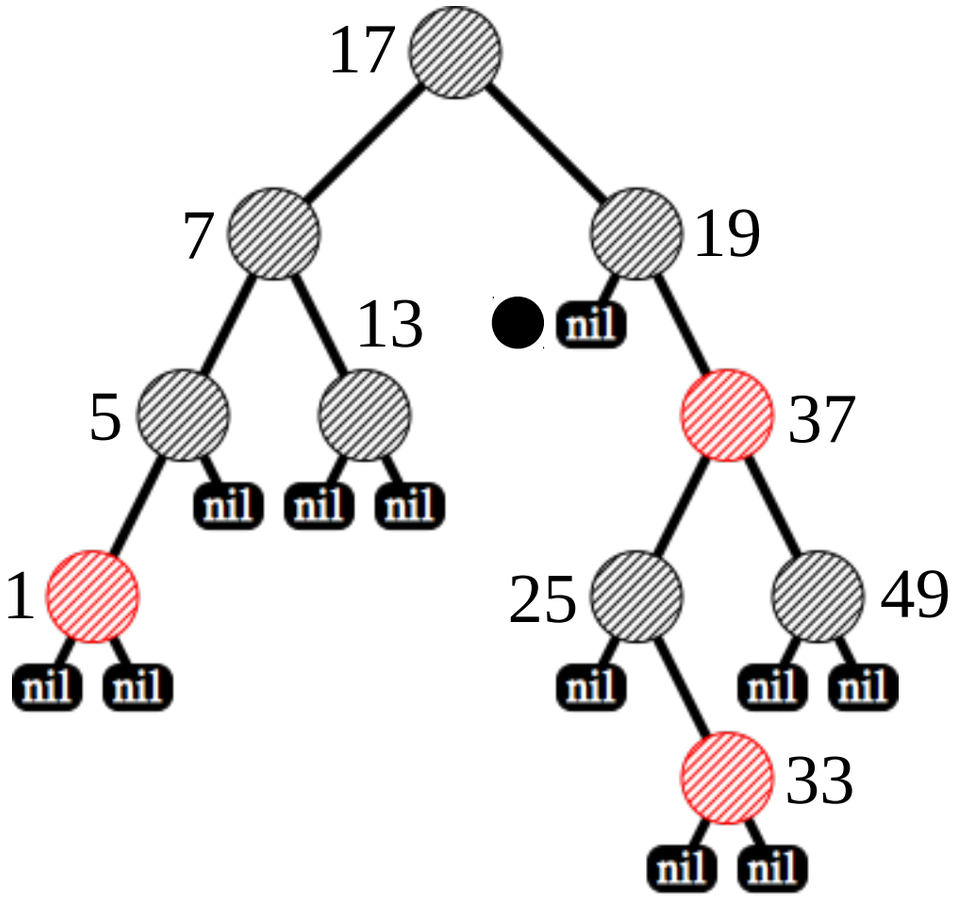



## Vlastnosti Red-Black stromů

- **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



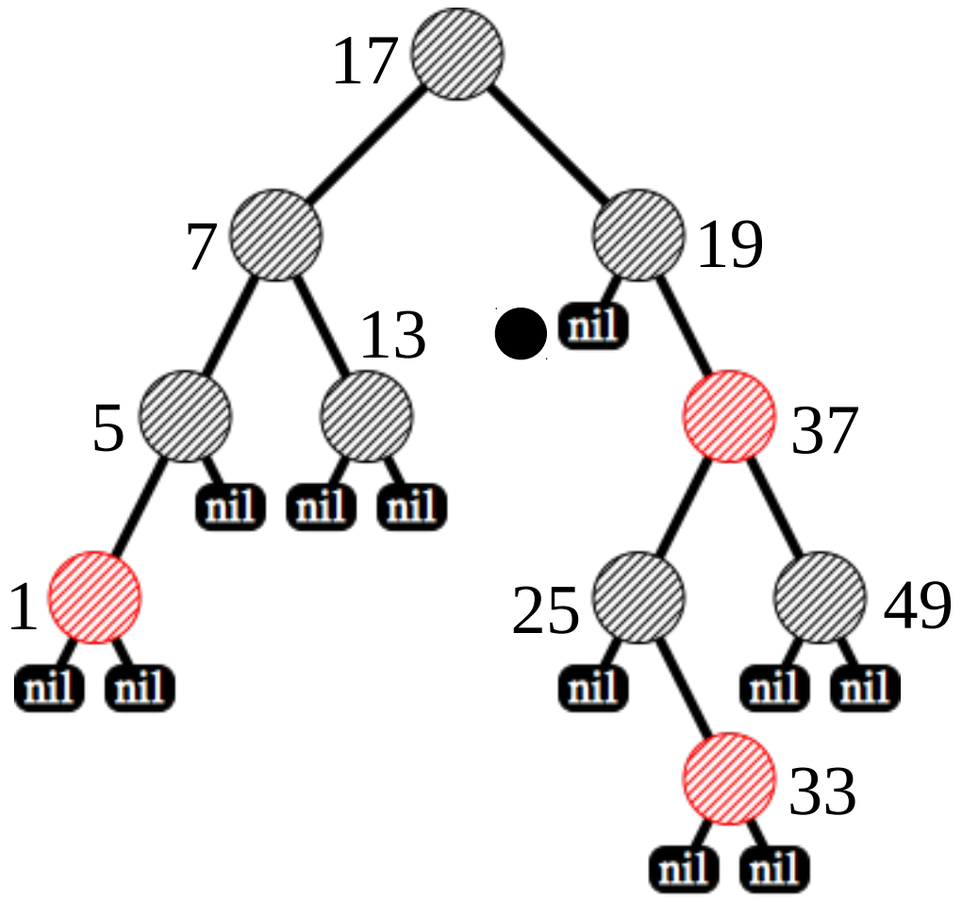

## Vlastnosti Red-Black stromů

- **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

**První krok:** Označíme syna zrušeného uzlu jako extra černý (“doubly black”)

# Zrušení uzlu v  Red-Black stromu



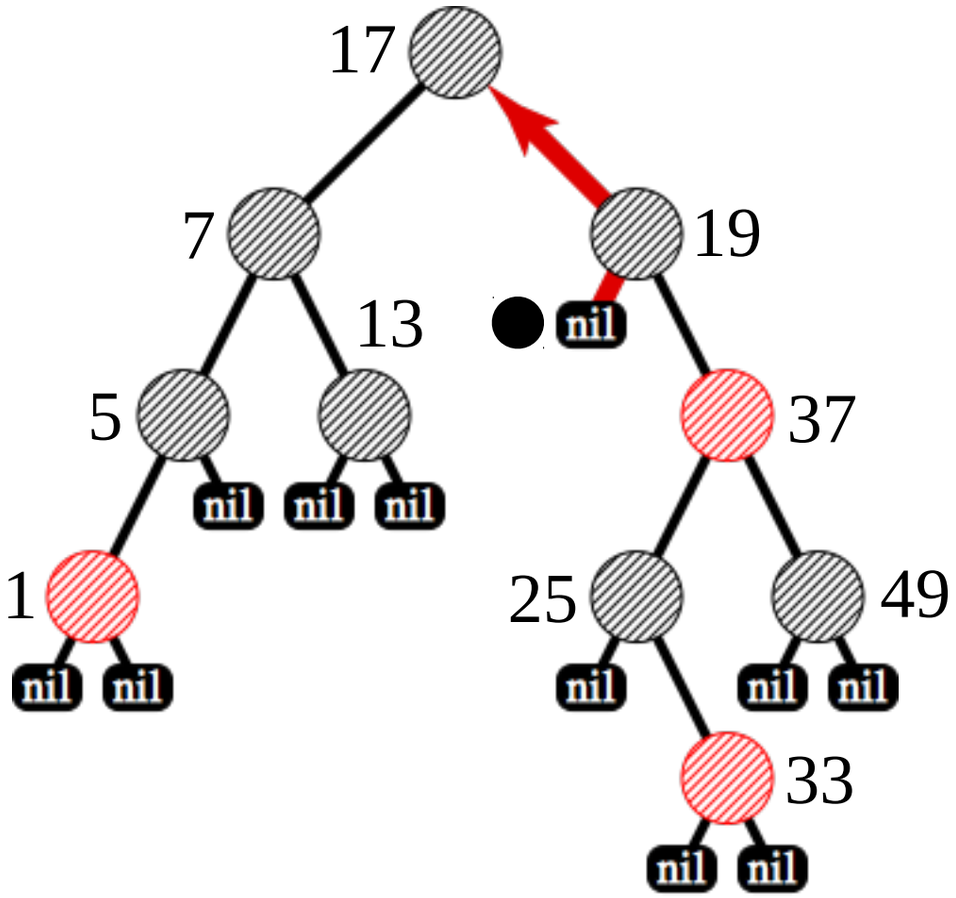

Rušený prvek:  18

## 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

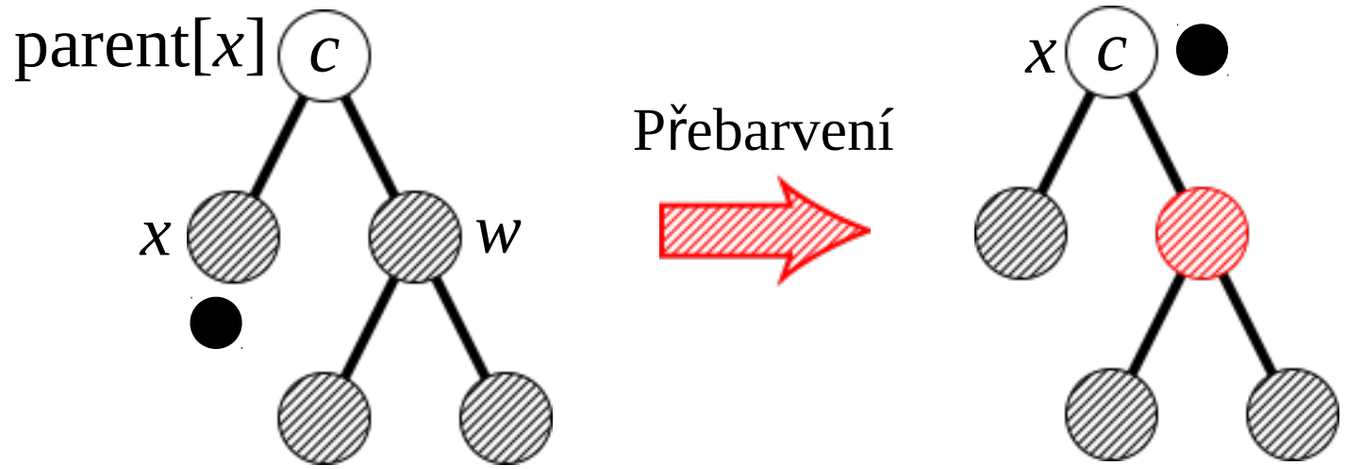



Rušený prvek:  18

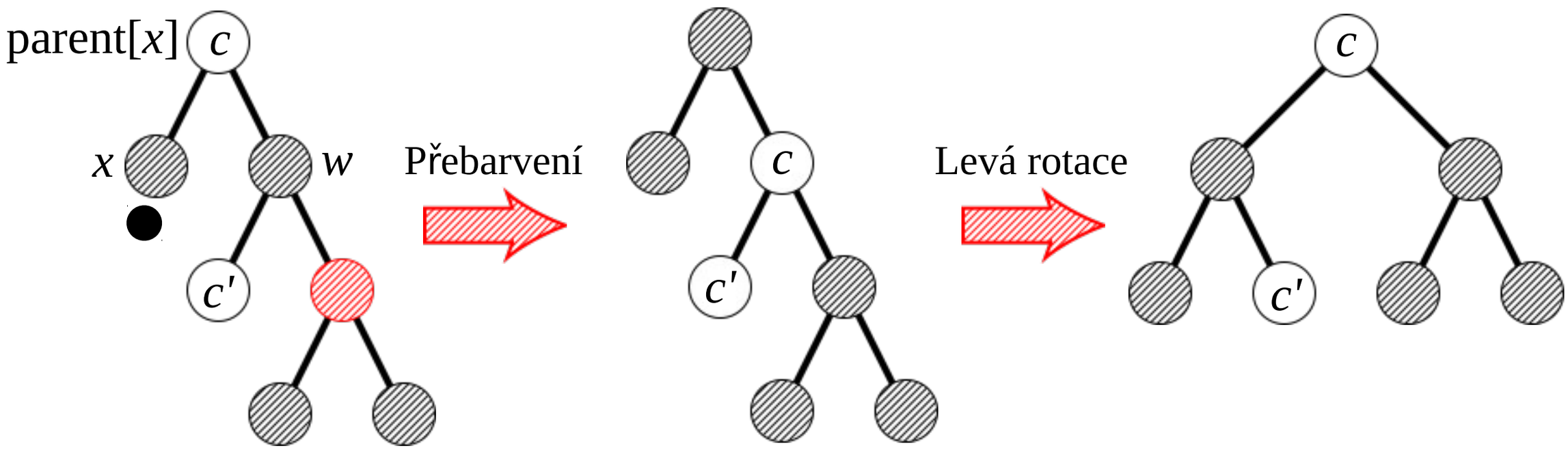
# Korekce barvy uzlů v RB stromu

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*

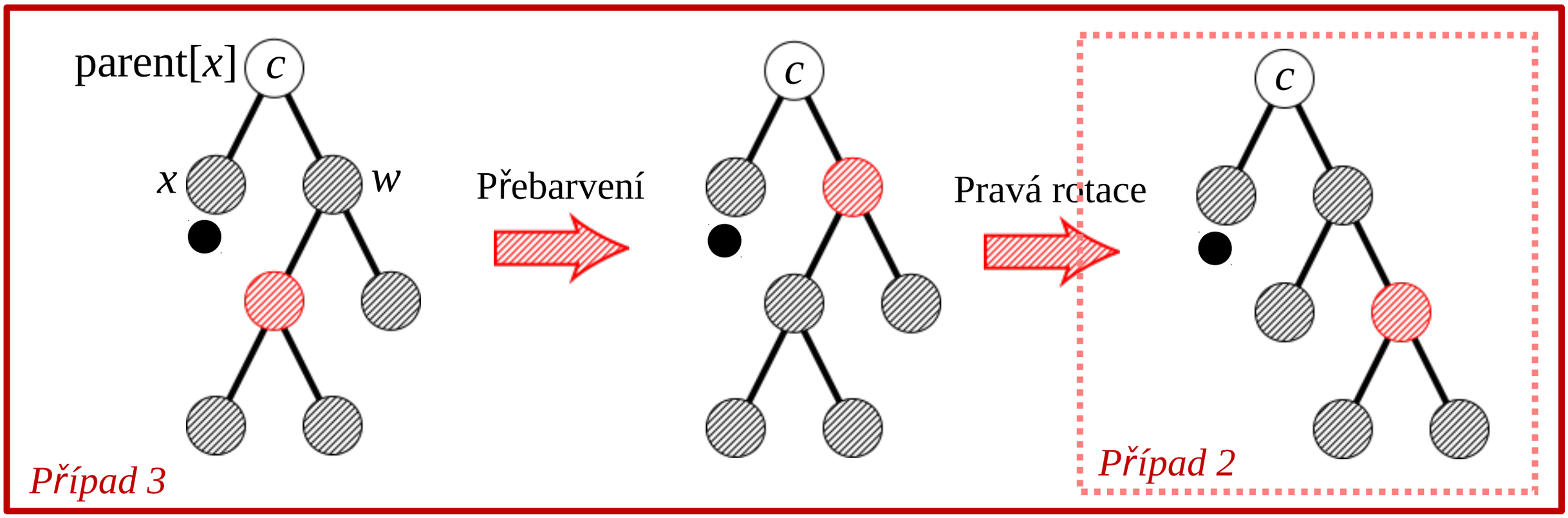



*Případ 2*

# Korekce barvy uzlů v RB stromu

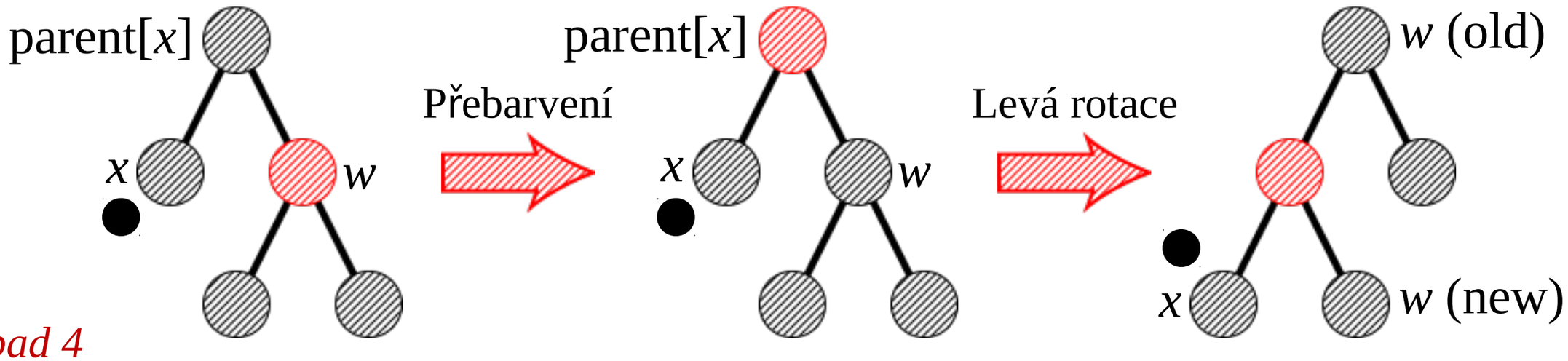
Předpokládejme že *x* je levý syn svého rodiče a jeho bratr *w* je černý.

- **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)
 1  if left[z] = nil[T] or right[z] = nil[T]
 2     then y ← z
 3     else y ← TREE-SUCCESSOR(z)
 4  if left[y] ≠ nil[T]
 5     then x ← left[y]
 6     else x ← right[y]
 7  p[x] ← p[y]
 8  if p[y] = nil[T]
 9     then root[T] ← x
10     else if y = left[p[y]]
11             then left[p[y]] ← x
12             else right[p[y]] ← x
13  if y ≠ z
14     then key[z] ← key[y]
15          copy y's satellite data into z
16  if color[y] = BLACK
17     then RB-DELETE-FIXUP(T, x)
18  return y
```

# Algoritmus zrušení prvku RB stromu

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