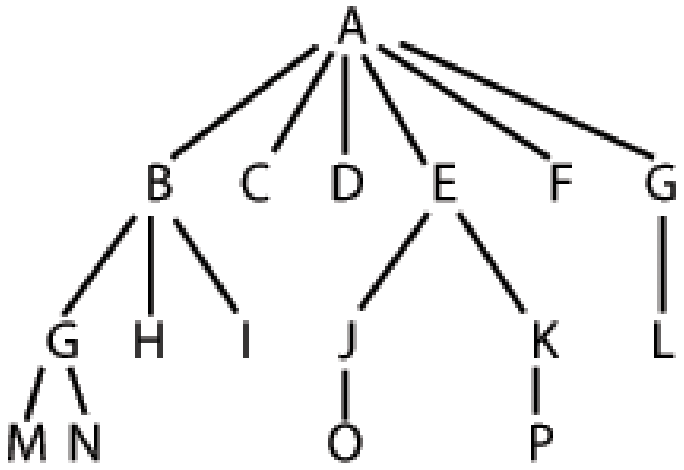
IB111 Programování a algoritmizace

Datové struktury:

Stromy

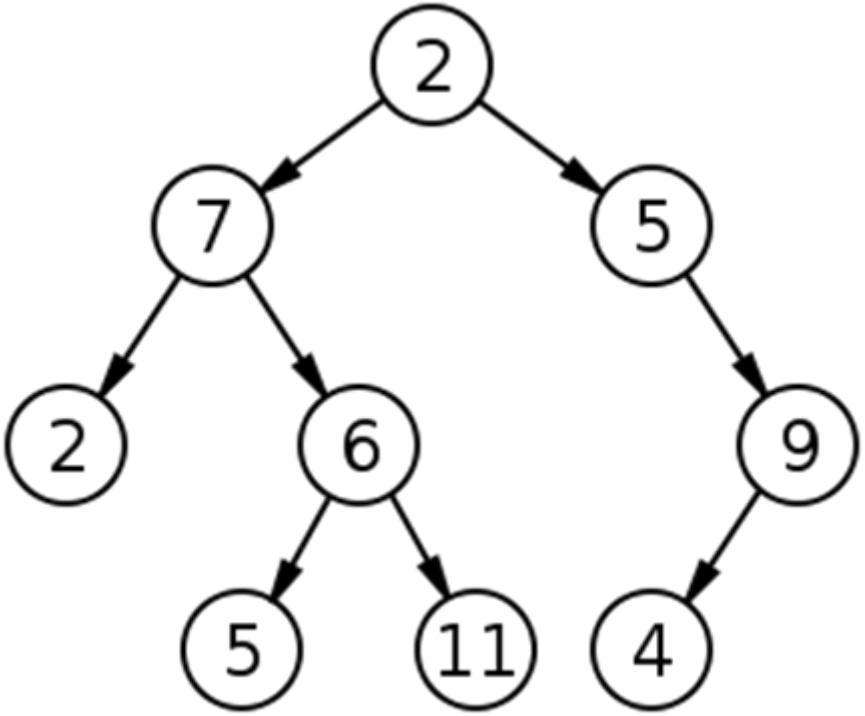
Struktura stromu

- Strom
  - Acyklická struktura, kde každý uzel kromě kořene má právě jednoho rodiče
  - Kořen, uzel, list
  - Rodič, dítě
  - Výška stromu
  - Hloubka uzlu
  - Vstupní stupeň uzlu
    - Pro kořen je nulová
  - Výstupní stupeň uzlu



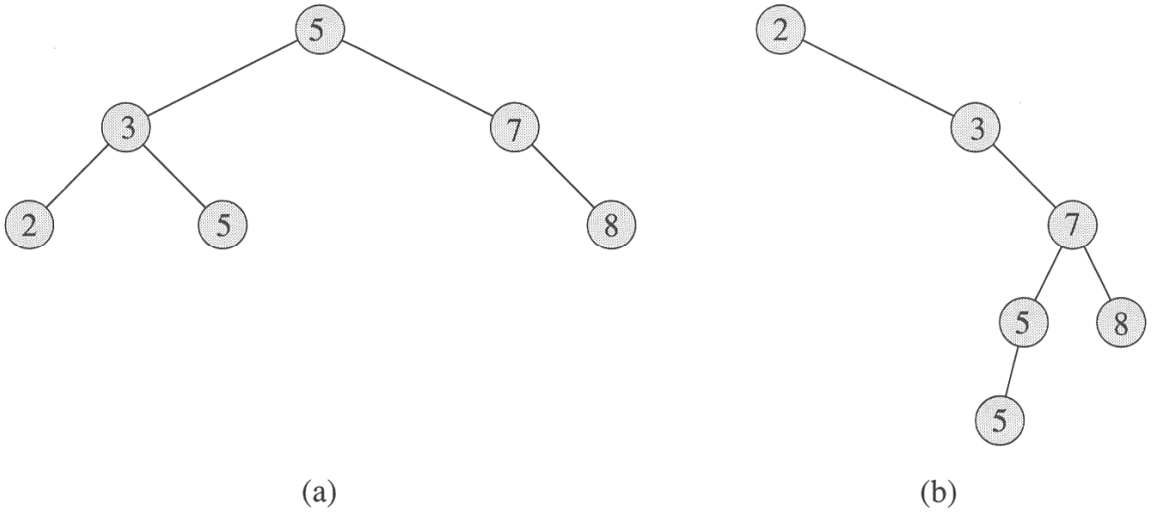
Binární stromy

- Binární strom je strom, kde každý uzel má maximálně 2 děti



Binární vyhledávací stromy

- Binární vyhledávací stromy
  - Pro každý uzel *x* platí, že uzly v levém podstromě uzlu *x* mají klíč menší nebo roven *x* a uzly v pravém podstromě mají klíč větší nebo roven uzlu *x*.



T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: Introduction to Algorithms.

Implementace

- Jak implementovat binární stromy?

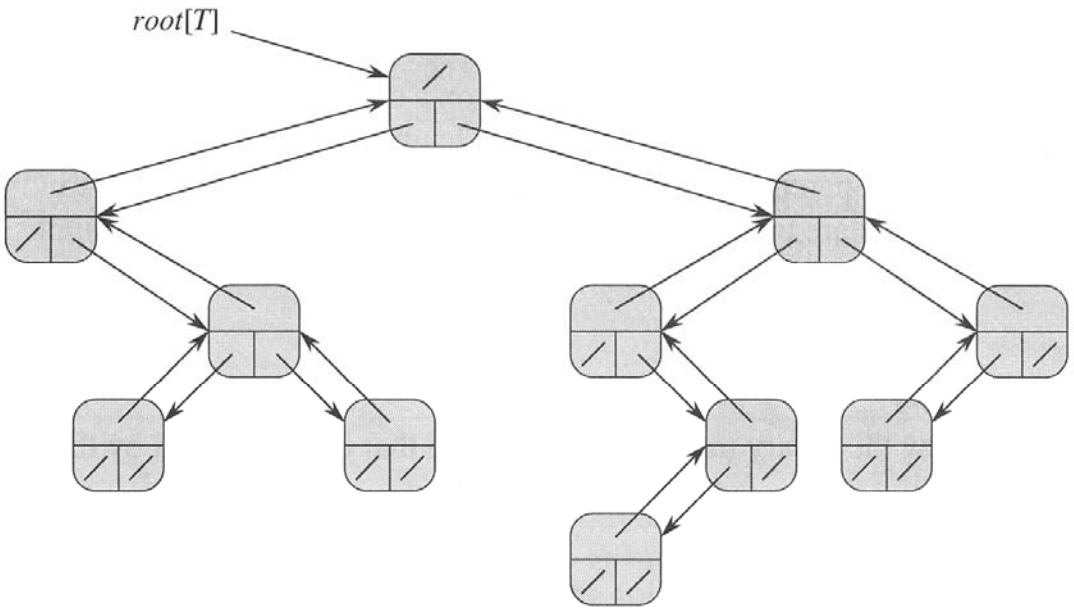


Figure 10.9 The representation of a binary tree T. Each node x has the fields $p[x]$ (top), $left[x]$ (lower left), and $right[x]$ (lower right). The key fields are not shown.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: Introduction to Algorithms.

Implementace v C

- Jak implementovat binární stromy?
- struct BinTreeNode {
  int key;
  struct BinTreeNode *left, *right;
  // struct BinTreeNode *parent;
  }
- struct BinTreeNode *root = NULL;

Implementace v Pythonu

- class Node:

```
left, right, key = None, None, 0

def __init__(self, value):
    self.left = None
    self.right = None
    self.key = value
```

- class BinTree:

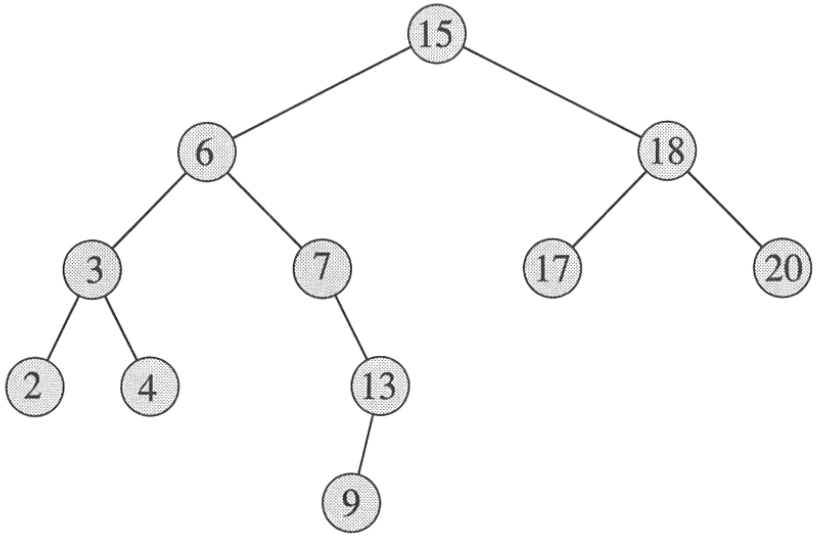
```
def __init__(self):
    self.root = None
```

Operace nad binárními stromy

- Nad binárními vyhledávacími stromy provádíme především vyhledávání
- Ostatní operace
  - Přidání prvku
  - Smazání prvku
  - Výpis prvků (inorder, preorder, postorder)
  - Nalezení minimálního a maximálního prvku

Vyhledání prvku

- Začneme v kořeni a pokračujeme dolů
  - Porovnáme klíč uzlu s hledanou hodnotou
    - Je roven – našli jsme
    - Hledaná hodnota je menší – pokračujeme levým podstromem
    - Hledaná hodnota je větší – pokračujeme pravým podstromem
  - Složitost je dána výškou (hloubkou) stromu



T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: Introduction to Algorithms.

Výpis prvků binárního stromu

- Inorder verze
  - Vypiš prvky v levém podstromu (existuje-li)
  - Vypiš procházený uzel
  - Vypiš prvky v pravém podstromu (existuje-li)
- Postorder (levý, pravý, aktuální)
- Preorder (aktuální, levý, pravý)

```
INORDER-TREE-WALK(x)
1   if x ≠ NIL
2       then INORDER-TREE-WALK(left[x])
3            print key[x]
4            INORDER-TREE-WALK(right[x])
```

Minumum a maximum

- Nalezení minimální hodnoty
  - sledujeme levé podstromy
- Nalezení maximální hodnoty
  - Sledujeme pravé podstromy

```
TREE-MINIMUM (x)
1  while left[x] ≠ NIL
2       do x ← left[x]
3  return x
```

Přidání prvku

- Vytvoříme nový uzel a zařadíme jej tak, aby stále platily podmínky binárního vyhledávacích stromů

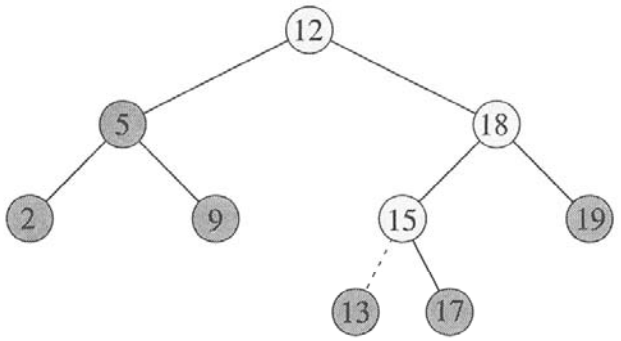


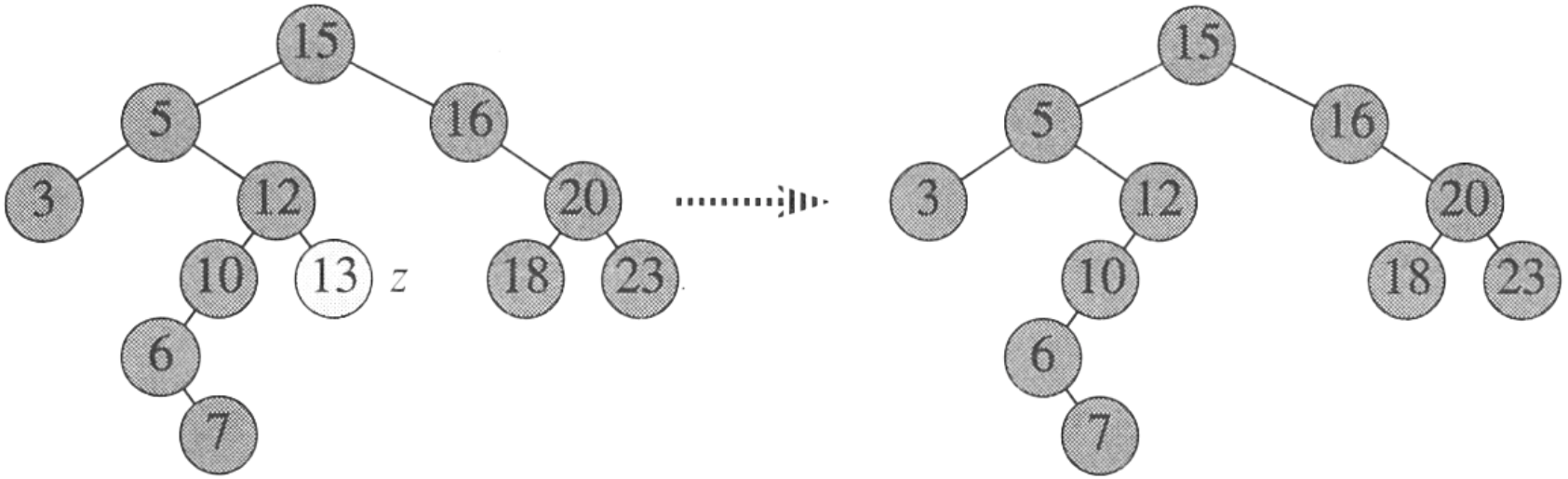
Figure 12.3 Inserting an item with key 13 into a binary search tree. Lightly shaded nodes indicate the path from the root down to the position where the item is inserted. The dashed line indicates the link in the tree that is added to insert the item.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: Introduction to Algorithms.

Odebrání prvku

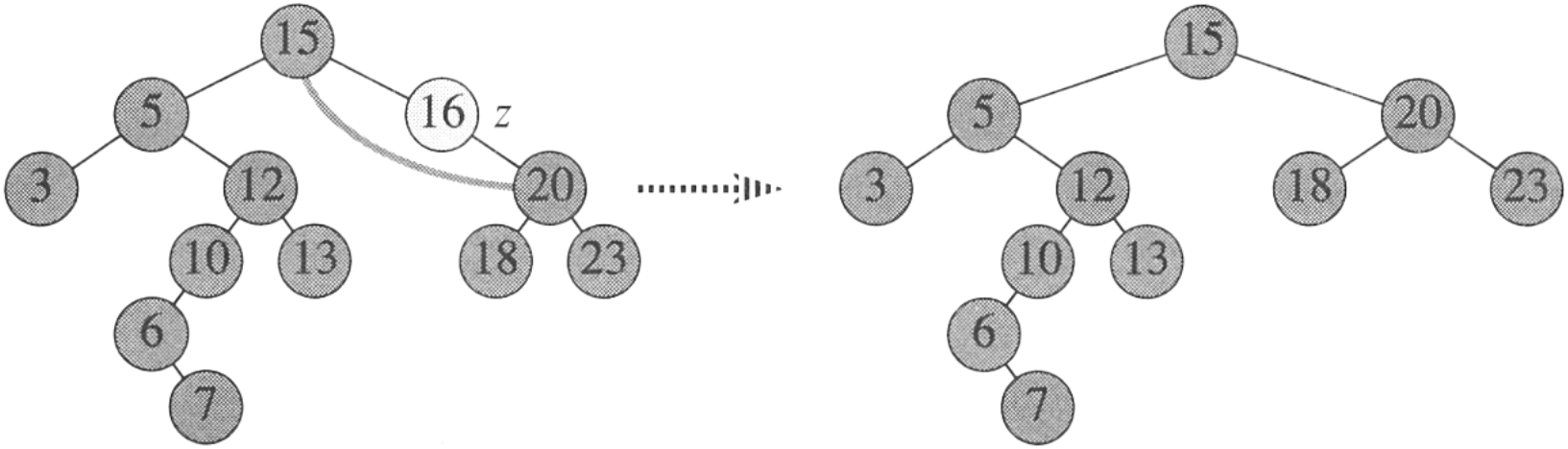
- 3 případy:
  - Uzel nemá děti → rovnou smažeme
  - Uzel má právě jedno dítě (levé nebo pravé) → nahradíme uzel, které máme smazat tímto dítětem
  - Uzel má dvě děti → najdeme minimální hodnotu v pravém podstromu, tento uzel smažeme a jeho hodnotou přepíšeme uzel, který jsme chtěli odstranit.

Odebrání prvku – ilustrace 1



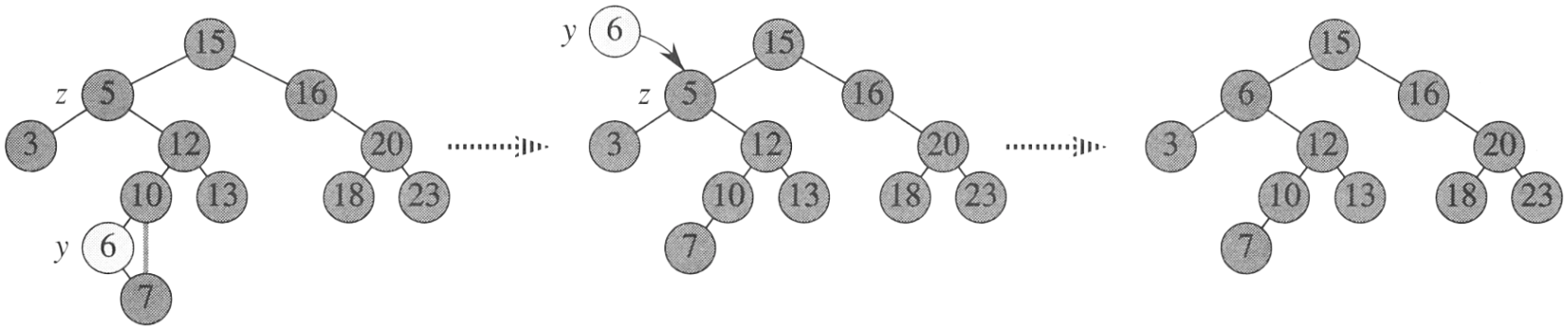
(a)

Odebrání prvku – ilustrace 2



(b)

Odebrání prvku – ilustrace 3

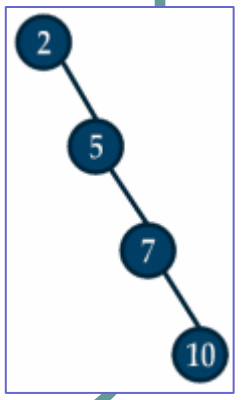


(c)

T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: Introduction to Algorithms.

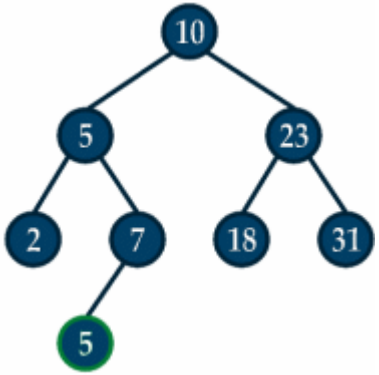
Časová náročnost

- Vyhledávání, přidávání i odebírání prvků má časovou náročnost O(h), kde h je výška stromu
- V ideálním případě je výška stromu rovna $\log_2 n$, kde n je počet uzlů
- V nejhorším případě je výška stromu rovna n, kde n je počet uzlů
  - Degenerovaný strom, vlastně zřetězený seznam

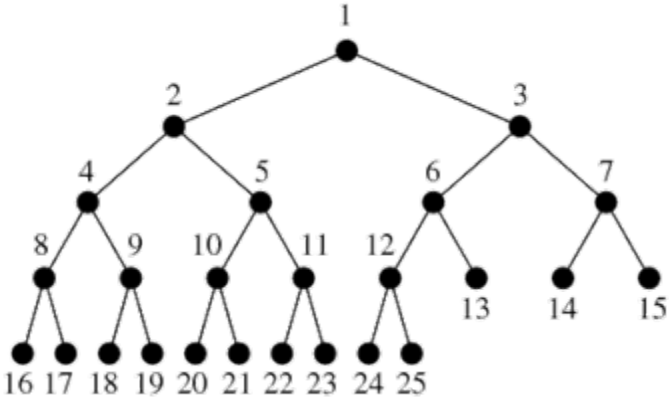


Výška stromu

- Strom je vyvážený, pokud až na poslední úroveň je strom „zaplněný“.

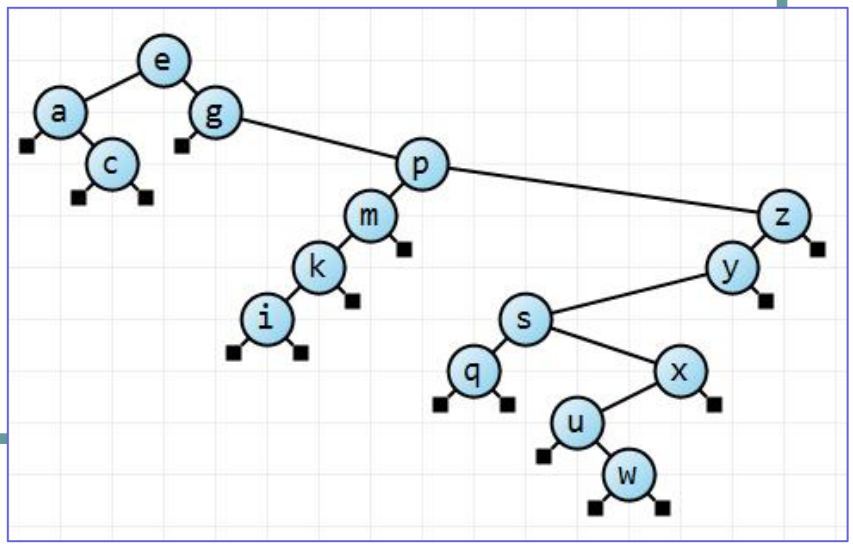


- Strom je kompletní, pokud je vyvážený a poslední úroveň se plní postupně zleva doprava.



Optimální binární vyhledávací stromy

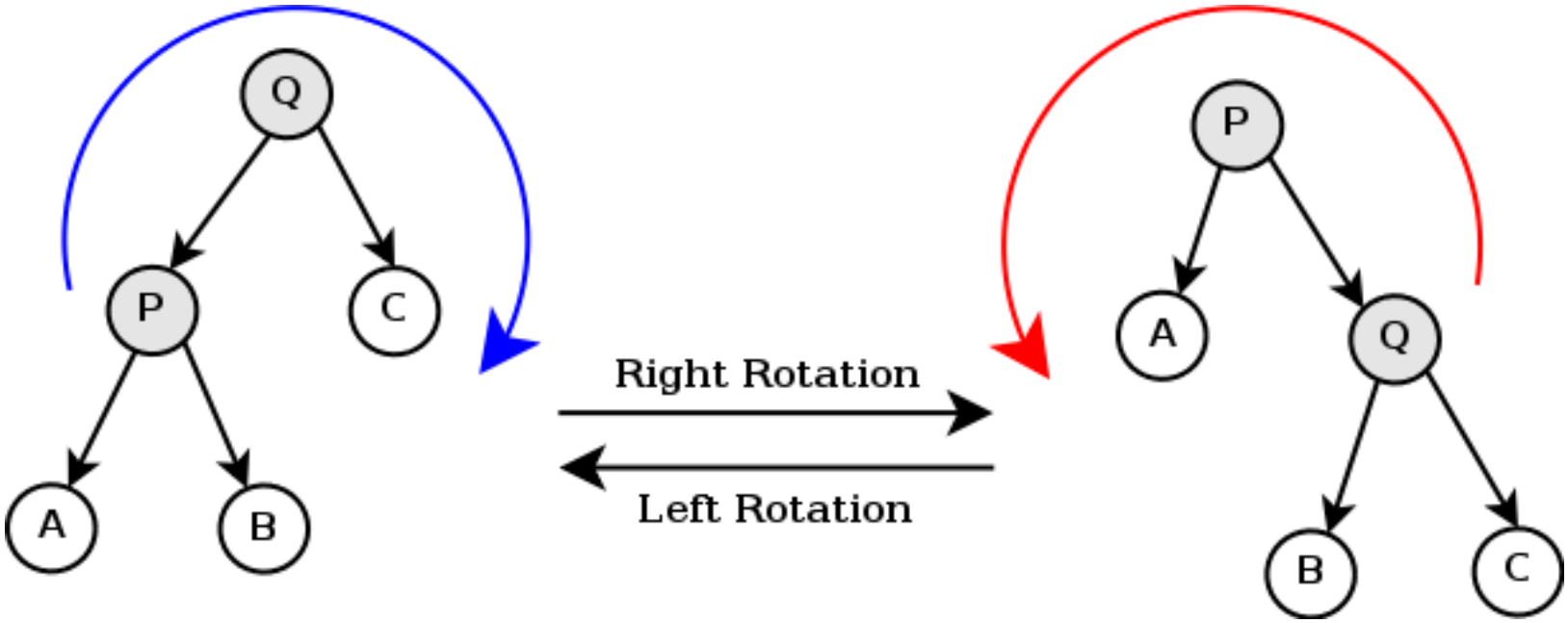
- Optimální binární vyhledávací stromy – průměrná doby vyhledání je minimalizována
  - Uzly jsou uspořádány tak, aby u často vyhledávaných položek byla cesta z kořene krátká
  - Potřebujeme znát četnosti vyhledávání jednotlivých prvků



AVL stromy

- Vyvážené binární vyhledávací stromy, toto vyvážení se neustále udržuje i při operacích přidávání a odebírání prvků
- 1962: G.M. Adelson-Velskii a E.M. Landis
- Faktor vyvážení
  - Výška jednoho podstromu MINUS výška druhého podstromu
  - Uzel je vyvážený, pokud má faktor -1, 0, 1
- Operace na AVL stromech jsou stejné jako u normálních binárních vyhledávacích stromů, ale pokud po operaci nejsou všechny uzly vyváženy je potřeba provést dodatečné vyvážení pomocí rotací stromů.

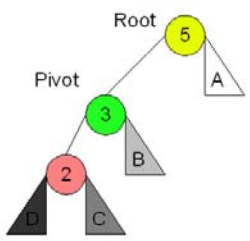
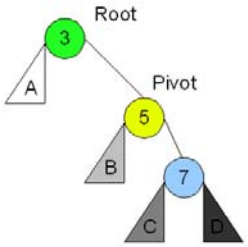
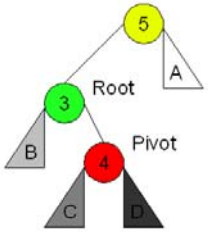
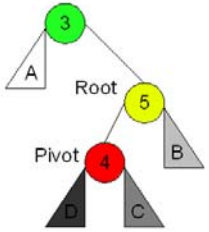
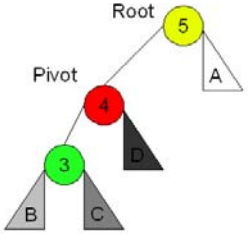
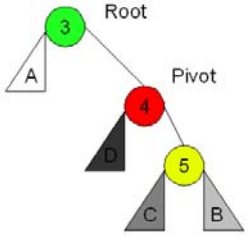
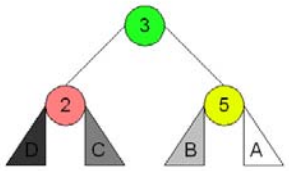
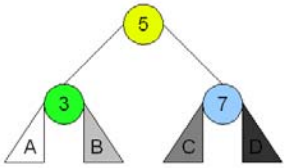
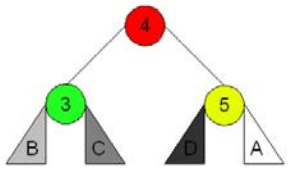
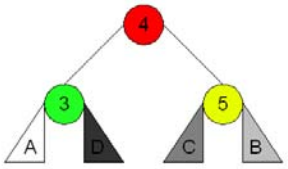
Rotace stromů



Vkládání

- Vložíme uzel a kontrolujeme faktor vyváženosti všech předchůdců
  - Pokud -1, 0, +1 pak OK
  - Pokud +2 pak kontrolujeme faktor pravého dítěte
    - Pokud +1 → levá rotace
    - Pokud -1 → dvojitá levá rotace (pravá, pak levá)
  - Pokud -2 pak kontrolujeme faktor levého dítěte
    - Pokud -1 → pravá rotace
    - Pokud +1 → dvojitá pravá rotace (levá, pak pravá)

Rotace stromu

Left Left Case	Right Right Case	Left Right Case	Right Left Case
			
Right Rotation	**Left Rotation**	**Left Rotation**	**Right Rotation**
			
		Right Rotation	**Left Rotation**
			

Odstranění uzlu

- Obdobné jako při vkládání
- Navíc další případy
- Faktor vyváženosti uzlu je 2
  - Faktor vyváženosti pravého dítěte 0 → levá rotace
- Faktor vyváženosti uzlu je -2
  - Faktor vyváženosti levého dítěte je 0 → pravá rotace

AVL stromy

- Časová náročnost operací je stejná jako u operací s binárními vyhledávacími stromy, tj. O(h), kde h je výška stromu
  - Vyvažování stromů tedy asymptotickou časovou náročnost nezvyšuje
- ALE výška h stromu je u AVL stromů vždy logaritmická, tedy i náročnost operací je logaritmická