

Vyhledávací stromy

Binární vyhledávání popsané v předchozí části má příznivou časovou složitost. Pokud bychom měli množinu prvků, v níž velmi často hledáme, pak by se nám viditelně vyplatilo je na začátku setřídít. Problém ovšem nastane, když se tato množina v průběhu času mění, tj. jsou k ní přidávány nové prvky nebo z ní naopak některé prvky jsou odebírány. Vkládání prvků na jiné místo, než je konec pole, nebo odebírání z jiného místa, než je konec pole, je poměrně neefektivní operace, neboť je spojena s přesuny prvků v poli. Časová složitost těchto operací je lineární, což už je nepříznivá časová složitost, pokud tyto operace probíhají častěji. Zde už je výhodnější použít vyhledávací stromy.

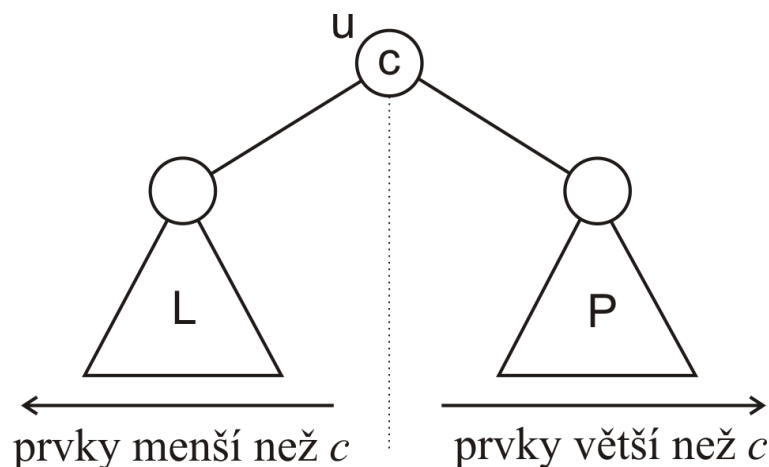
Vyhledávací stromy jsou velmi významnou a širokou skupinou vyhledávacích metod. Lze je rozdělit na

- Binární vyhledávací stromy
- Vícecestné vyhledávací stromy

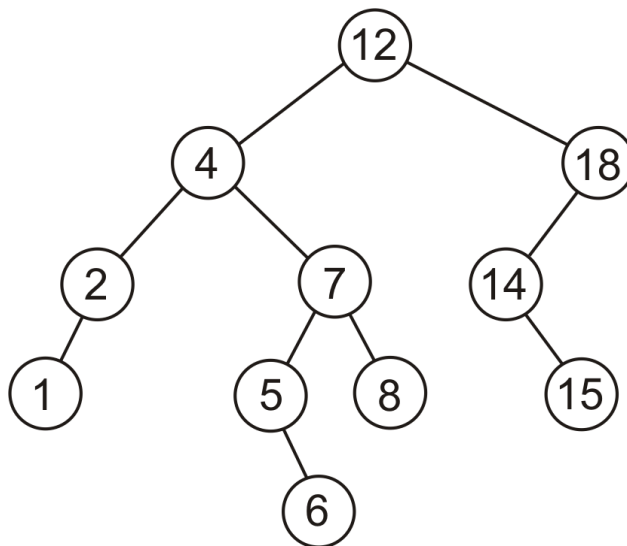
Binární vyhledávací stromy

Jsou binární stromy s vlastnostmi:

- V každém uzlu stromu je uložen jeden datový prvek.
- Pro každý uzel u a prvek v něm uložený c platí, že prvky uložené v levém podstromu uzlu u (má-li uzel u levý podstrom) jsou menší než prvek c a prvky uložené v pravém podstromu uzlu u (má-li uzel u pravý podstrom) jsou větší než prvek c .



Příklad.



Vyhledání prvku

1. Počáteční krok

Uzel, který je v daném okamžiku vyhledávání aktuální, budeme označovat u .

Na začátku jím bude kořen stromu.

Nechť hledaný prvek má hodnotu x .

2. Průběžný krok

Vezmeme prvek obsažený v aktuálním uzlu u , označme ho c , a srovnáme ho s hledanou hodnotou x :

- Nejprve srovnáme, zda je $x < c$:

Pokud ano, pak je nutné v hledání pokračovat v levém podstromu. Jako nový aktuální uzel u stanovíme levého následníka současného aktuálního uzlu a znovu uděláme krok 2.

Pokud současný aktuální uzel u levého následníka nemá, vyhledávání končí - hledaný prvek není ve stromu obsažen.

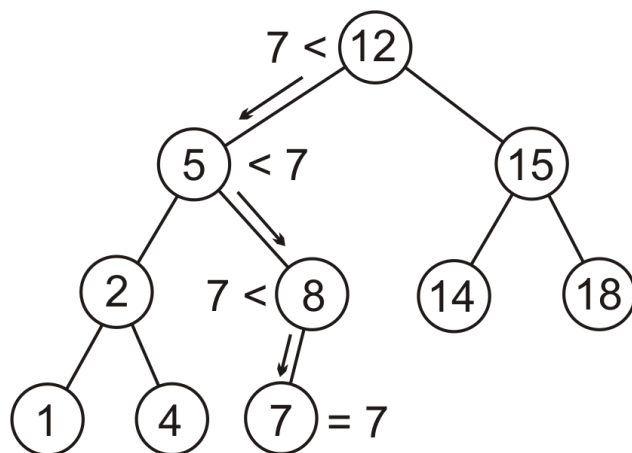
- Pokud není $x < c$, uděláme druhé srovnání, zda je $x > c$:

Pokud ano, je nutné v hledání pokračovat v pravém podstromu. Jako nový aktuální uzel u stanovíme pravého následníka současného aktuálního uzlu a opět uděláme krok 2.

Pokud současný aktuální uzel pravého následníka nemá, vyhledávání končí, hledaný prvek není ve stromu obsažen.

- Pokud není ani $x > c$, zbývá už jen případ, že platí $x = c$, čímž jsme u konce hledání a prvek c obsažený v současném aktuálním uzlu u je hledaným prvkem.

Příklad. V následujícím stromu máme vyhledat číslo 7.



Časová složitost: $\Theta(h)$, kde h je výška vyhledávacího stromu.

Pseudokód:

```
Search(T, x)
    u ← T.root
    while u ≠ NIL
        if x < u.item
            u ← u.left
        else
            if x > u.item
                u ← u.right
            else
                return u
    return NIL
```

Přidání prvku

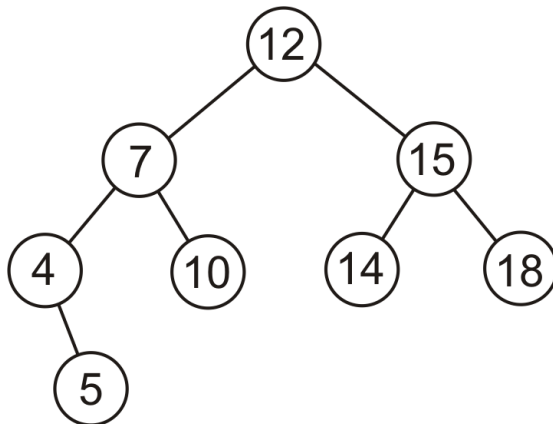
Operace přidání prvku do binárního vyhledávacího stromu znamená na příslušném místě přidat do stromu uzel, do kterého nový prvek vložíme.

Označme přidávaný prvek x . Nejprve proběhne jeho vyhledání ve stromu. Použijeme k tomu již popsany algoritmus vyhledávání. Vyhledání prvku x ve stromu může skončit třemi způsoby:

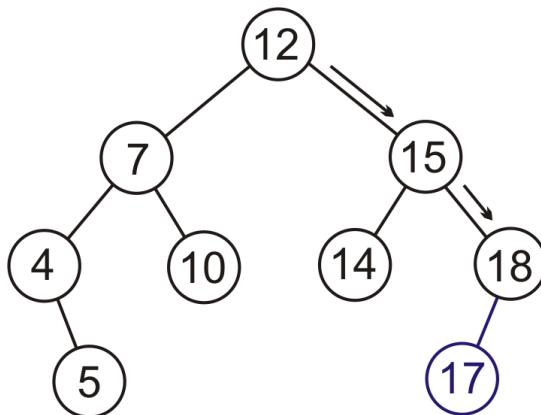
- Prvek x byl ve stromu nalezen. Tím přidávání končí, neboť prvek x už je ve stromu obsažen a u standardních vyhledávacích stromů nelze mít ve stromu stejný prvek vícekrát.

- Vyhledávání skončilo v uzlu u s prvkem c , přičemž je $x < c$ a uzel u nemá levého následníka. V tom případě přidáme ke stromu nový uzel jako levého následníka uzlu u a do něho nový prvek x vložíme.
- Vyhledávání skončilo v uzlu u s prvkem c , přičemž je $x > c$ a uzel u nemá pravého následníka. V tom případě přidáme ke stromu nový uzel jako pravého následníka uzlu u a do něho nový prvek x vložíme.

Příklad. Do binárního vyhledávacího stromu



máme vložit prvek 17. Následující obrázek ukazuje postup.



Časová složitost: $\Theta(h)$, kde h je výška vyhledávacího stromu.

Pseudokód:

NewNode (x)

$u \leftarrow \text{new Node}$

$u.\text{item} \leftarrow x$

$u.\text{left} \leftarrow u.\text{right} \leftarrow \text{NIL}$

return u

Insert(T, x)

if $T.\text{root} = \text{NIL}$

$T.\text{root} \leftarrow \text{NewNode}(x)$

```

    return true
u ← T.root
while true
    if x < u.item
        if u.left = NIL
            u.left ← NewNode(x)
            return true
        u ← u.left
    else
        if x > u.item
            if u.right = NIL
                u.right ← NewNode(x)
                return true
            u ← u.right
        else
            return false

```

Odebrání prvku

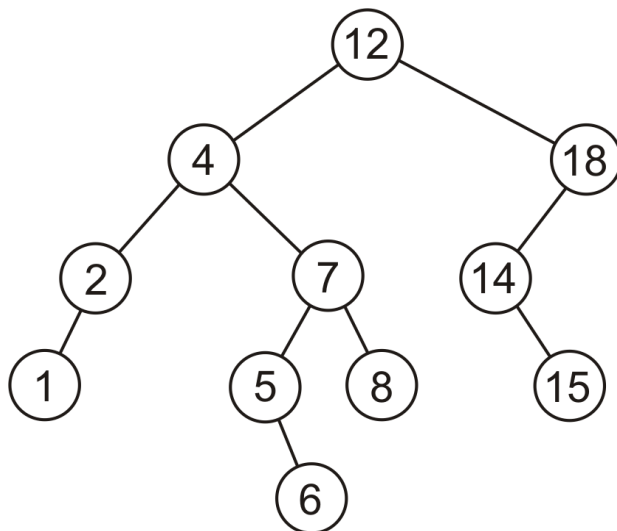
Operace odebrání prvku z binárního stromu znamená zrušení uzlu ve stromu. Nemusí to ale být uzel, ve kterém je odebíraný prvek, jak ukazuje následující popis postupu odebrání.

Označme odebíraný prvek x .

Vyhledáme prvek x ve stromu. Vyhledání může skončit třemi způsoby:

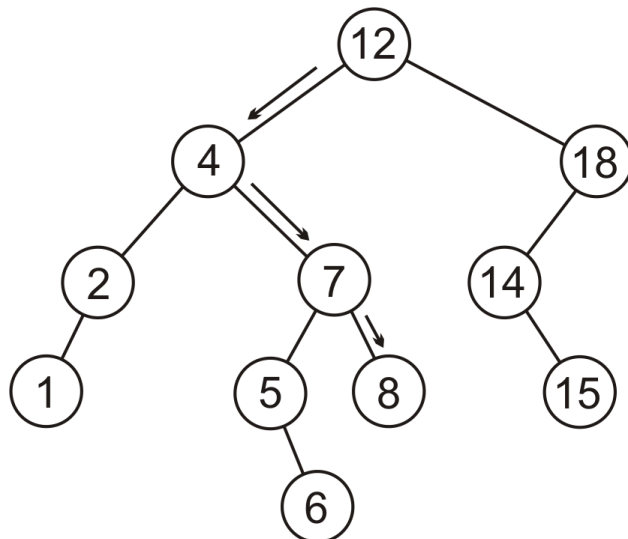
- Prvek x nebyl ve stromu nalezen – není co odebrat.
- Prvek byl nalezen v uzlu u , který má nejvýše jednoho následníka. Tento uzel zrušíme. Pokud rušený uzel u měl (jednoho) následníka, jeho následník nyní bude následníkem předchůdce rušeného uzlu u . Pokud rušený uzel u neměl předchůdce (byl kořenem), jeho následník bude novým kořenem.
- Prvek byl nalezen v uzlu u , který má dva následníky. V tomto případě do uzlu u přesuneme buďto nejpravější (největší) prvek z jeho levého podstromu anebo nejlevější (nejmenší) prvek z jeho pravého podstromu a uzel, z kterého byl prvek přesunut, zrušíme.

Příklad. Ze stromu

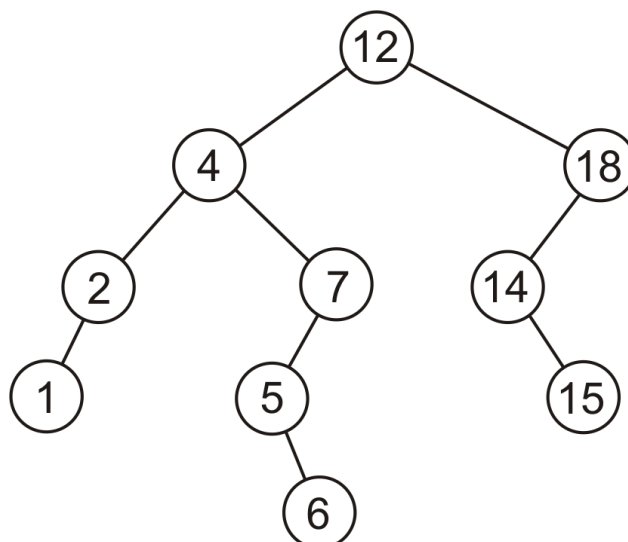


máme odebrat prvek 8.

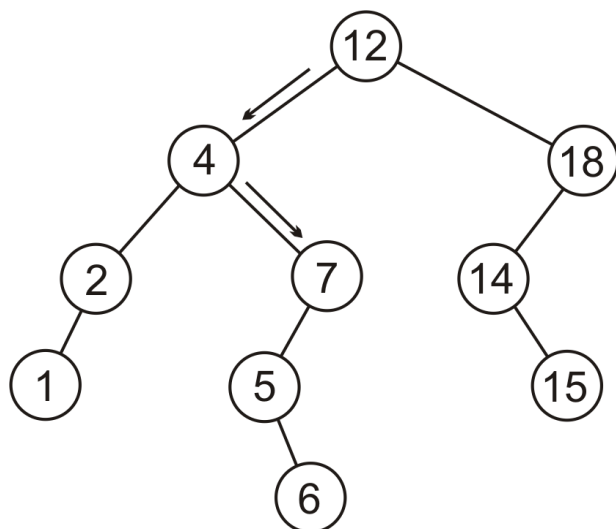
Nejprve prvek 8 ve stromu vyhledáme.



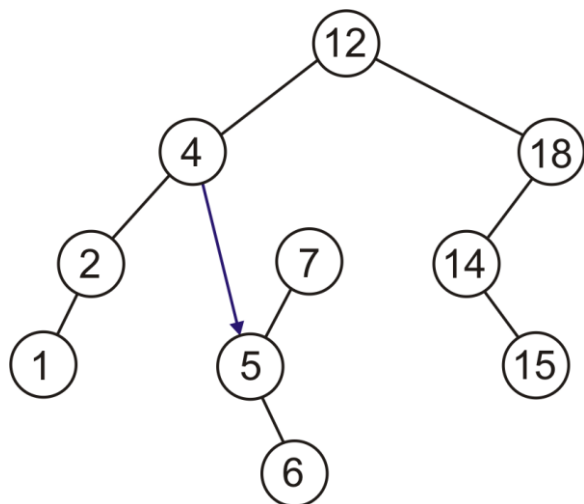
Protože prvek 8 je v uzlu, který nemá více než jednoho následníka, uzel s tímto prvkem můžeme zrušit.



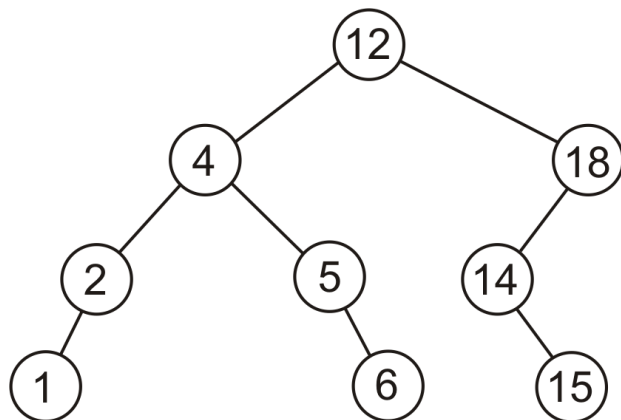
Dále máme odebrat prvek 7. Nejprve ho ve stromu vyhledáme.



Protože prvek 7 je v uzlu, který nemá více než jednoho následníka, uzel s tímto prvkem můžeme zrušit. Nejprve ale přesměrujeme odkaz v předchůdci rušeného uzlu na následníka rušeného uzlu.

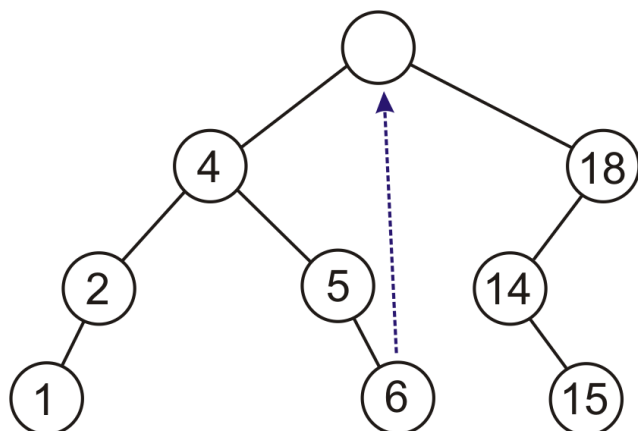


A uzel s prvkem 7 zrušíme.

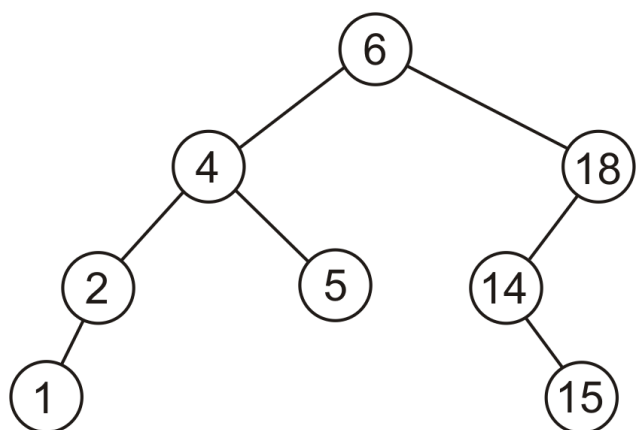


Jako další máme odebrat prvek 12. Jeho vyhledání je zde rychlé, neboť prvek je v kořenu. Uzel s tímto prvkem má dva následníky, nemůžeme ho přímo odebrat.

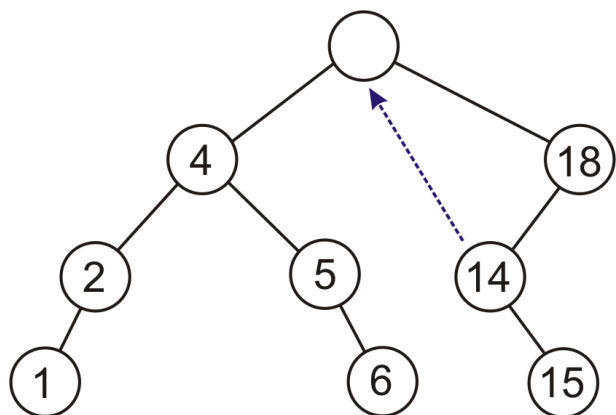
První možnost je nahradit odebíraný prvek 12 nejpravějším prvkem z levého podstromu, což je prvek 6.



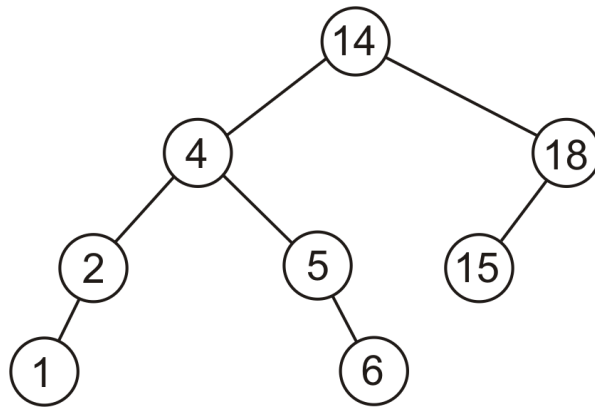
A uzel, ve kterém byl prvek 6, zrušit.



Druhá možnost zde byla nahradit odebíraný prvek 12 nejlevějším prvkem z pravého podstromu, což je prvek 14.



A uzel, ve kterém byl prvek 14, zrušit.



Časová složitost: $\Theta(h)$, kde h je výška vyhledávacího stromu.

Pseudokód:

```
Delete(T, x)
    u ← T.root
    if u = NIL
        return false
    if u.item = x
        T.root ← DeleteNode(u)
        return true
    while true
        if x < u.item
            if u.left = NIL
                return false
            if u.left.item = x
                u.left ← DeleteNode(u.left)
                return true
            u ← u.left
        else
            if u.right = NIL
                return false
            if u.right.item = x
                u.right ← DeleteNode(u.right)
                return true
            u ← u.right
DeleteNode(u)
    if u.left = NIL
```

```
    return u.right
if u.right = NIL
    return u.left
v ← u.right
if v.left = NIL
    u.item ← v.item
    u.right ← v.right
    return u
w ← v.left
while w.left ≠ NIL
    v ← w
    w ← w.left
u.item ← w.item
v.left ← w.right
return u
```