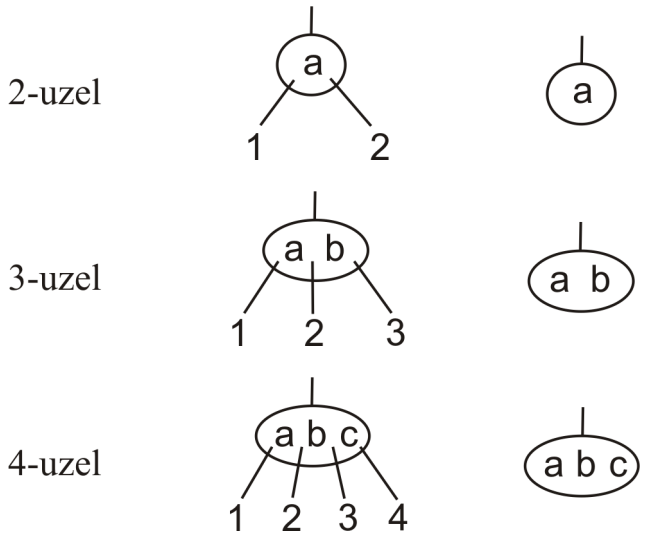
2-3-4 stromy

2-3-4 stromy jsou B-stromy řádu 4. Jejich název je odvozen od označení uzlů podle počtu, kolik má uzel následníků v případě, když je nelistový. Jsou tři druhy uzlů:



Operace vyhledávání v nich probíhá stejně jako v B-stromu. Rozdíl mezi 2-3-4 stromem a B-stromem je v operaci přidání prvku do stromu.

Přidání prvku do B-stromu řádu 4:

- Vyhledání uzlu, do kterého má být prvek vložen.
- Pokud je uzel již plný (obsahuje již 3 prvky) je provedeno rozdělení tohoto uzlu a dle potřeby i dělení dalších uzlů směrem nahoru.

Přidání prvku do 2-3-4 stromu:

- V průběhu hledání uzlu, do kterého má být prvek vložen, jsou plně obsazené uzly (4-uzly) na cestě od kořene k příslušnému listu preventivně štěpeny tak, aby při dosažení cílového listu tento nebyl plně obsazen (byl 2-uzel nebo 3-uzel).

Přidání prvku do B-stromu reprezentuje průchod dolů (hledání uzlu pro vložení) a následně případný postup nahoru (rozdělování plně obsazených uzlů).

Naproti tomu přidání prvku do 2-3-4 stromu reprezentuje jen průchod dolů, neboť rozdělováním plně obsazených uzlů při tomto průchodu se zajistí, že po přidání prvku do listu nedojde k překročení jeho kapacity.

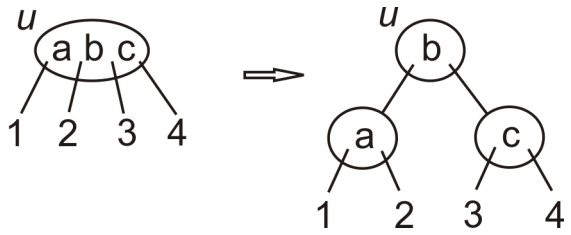
Přidání prvku

1. Počáteční krok

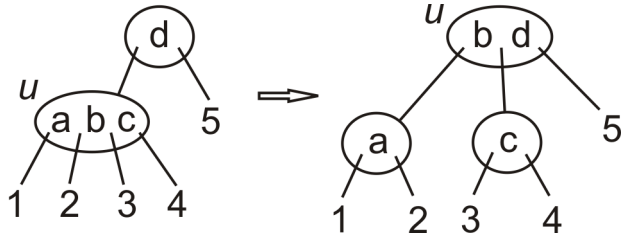
   Uzel, který je v daném okamžiku vyhledávání aktuální, budeme označovat *u*. Na začátku jím bude kořen stromu.

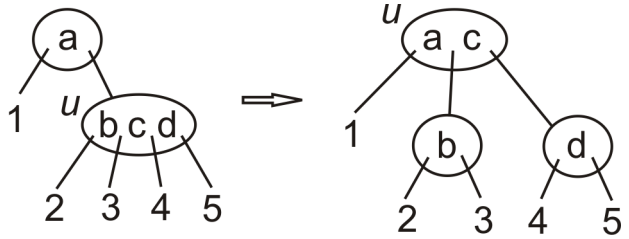
Přidávaný prvek nechť je *x*.

2. Průběžný krok
   - Nejprve zjistíme, zda procházený uzel *u* není 4-uzel. Jestliže ano, rozštěpíme ho:
     - uzel *u* je kořen

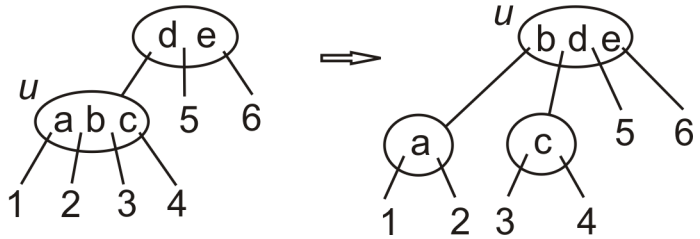


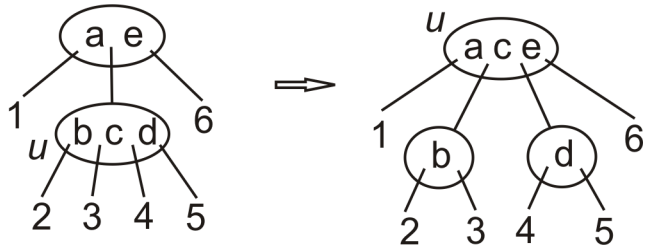
     - předchůdce uzlu *u* je 2-uzel

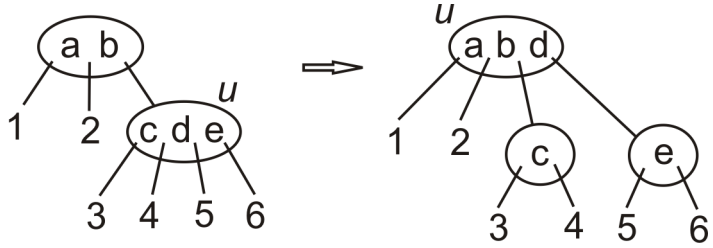




     - předchůdce uzlu *u* je 3-uzel



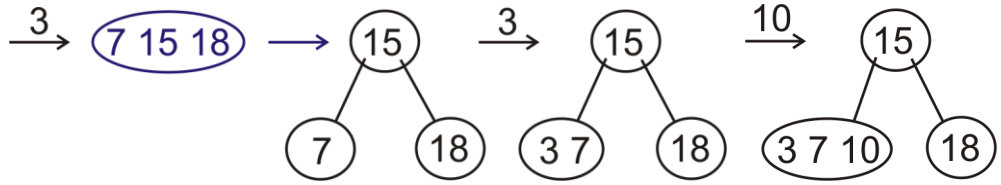


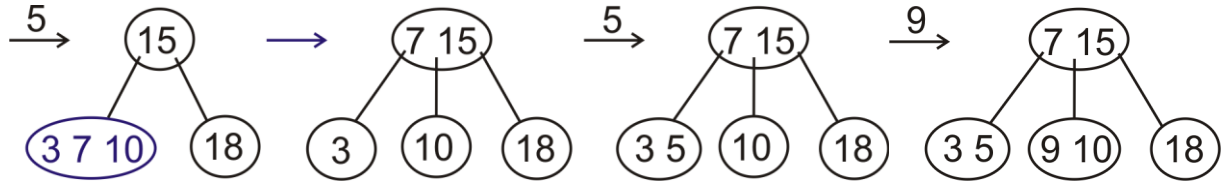


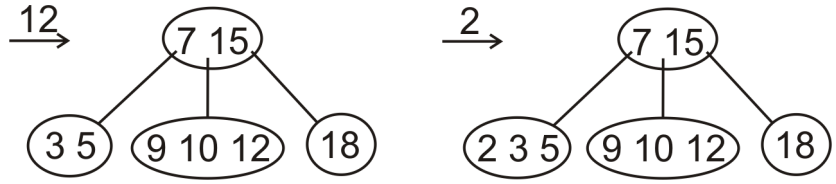
Po rozštěpení se aktuálním uzlem *u* stává uzel, do kterého byl vložen střední prvek štěpeného uzlu.

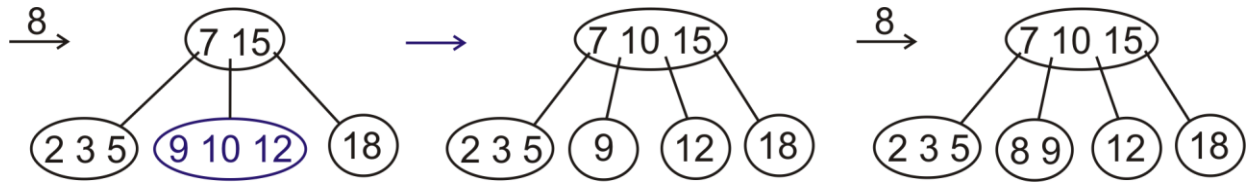
- Vyhledáme prvek *x* v aktuálním uzlu *u*.
  - Je-li prvek *x* v uzlu *u* nalezen, operace přidání prvku končí.
  - Není-li prvek v uzlu *u* nalezen:
    - Není-li uzel *u* list, dalším aktuálním uzlem *u* se stává jeho příslušný následník.
    - Je-li uzel *u* list, prvek *x* vložíme do tohoto uzlu.

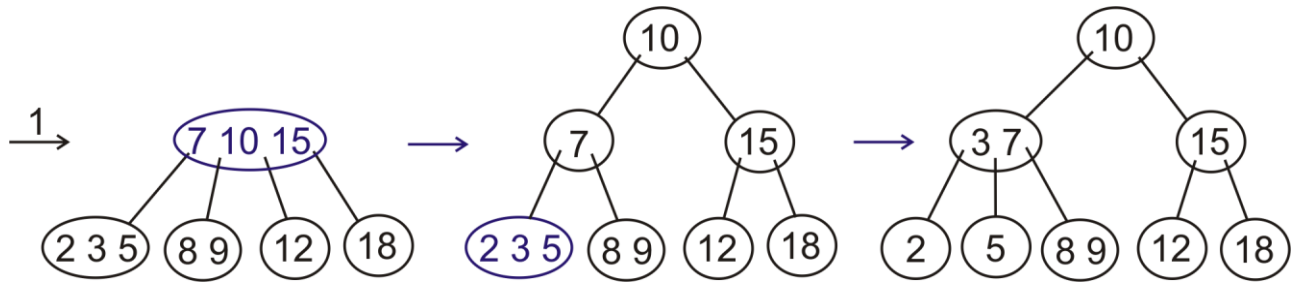
Příklad.

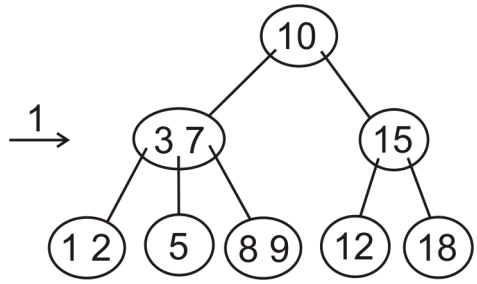






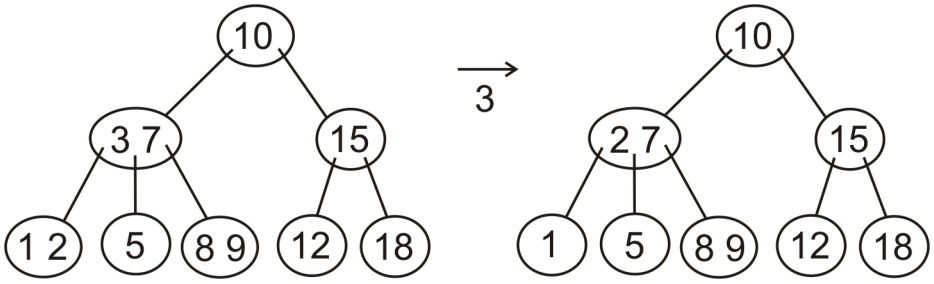


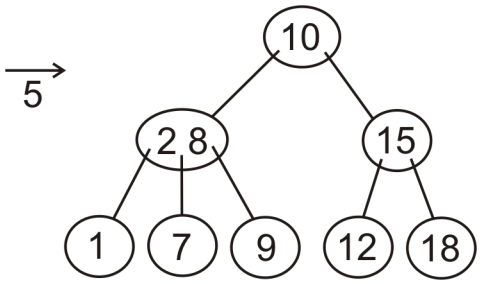


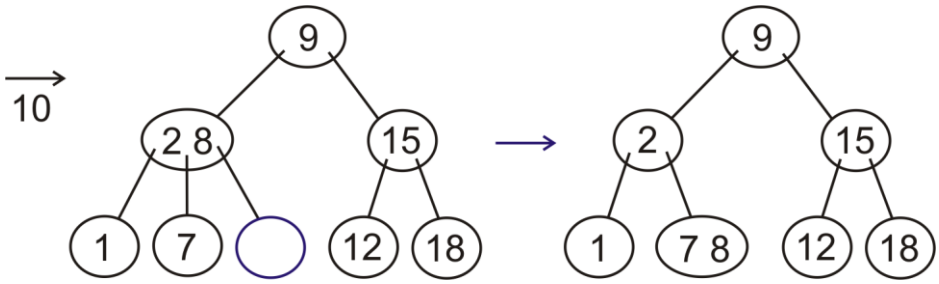


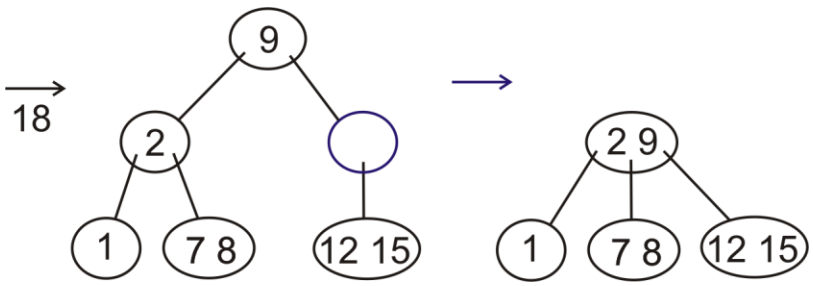
Operace odebrání probíhá ve 2-3-4 stromu stejně jako v B-stromu.

Příklad.









Časová složitost: 2-3-4 strom je B-strom 4. řádu, odtud $\Theta(\ln(n))$.

Pseudokód vyhledání:

```
Search(T, x)
  u ← T.root
  while u ≠ NIL
    i ← 0
    while i < u.order-1 and x >= u.item[i]
      if x = u.item[i]
        return u
      i ← i+1
    u ← u.child[i]
  return NIL
```

Pseudokód vložení:

```
CreateNode(x, v, w)
  u ← new Node
  u.order ← 2
  u.item[0] ← x
  u.child[0] ← v
  u.child[1] ← w
  u.child[2] ← u.child[3] ← NIL
  return u

SplitNode(u, v)   // v je předchůdce aktuálního uzlu u
  if v = NIL      // u je kořen
    u.child[0] ←
        CreateNode(u.item[0],u.child[0],u.child[1])
    u.child[1] ←
        CreateNode(u.item[2],u.child[2],u.child[3])
    u.order ← 2
    u.item[0] ← u.item[1]
    return u
  j ← v.order-1
  while j>0 and u.item[1] < v.item[j-1]
```

```
    v.item[j] ← v.item[j-1]
    v.child[j+1] ← v.child[j]
    j ← j - 1
  v.item[j] ← u.item[1]
  v.child[j] ← u
  u.order ← 2
  v.child[j+1] ←
      CreateNode(u.item[2],u.child[2],u.child[3])
  v.order ← v.order+1
  return v

Insert(T, x)
  u ← T.root
  if u = NIL        // strom zatím nemá žádný kořen
    T.root ← CreateNode(x,NIL,NIL)
    return true
  v ← NIL           // zatím není žádný předchůdce aktuálního uzlu u
  while true
    if u.order=4
      u ← SplitNode(u,v)    // nejprve rozštěpení uzlu u
    i ← 0
    while i < u.order-1 and x >= u.item[i]
      if x = u.item[i]
        return false
      i ← i+1
    if u.child[i] ≠ NIL
      v ← u
      u ← u.child[i]
    else
      j ← u.order-1
      while j>i
        u.item[j] ← u.item[j-1]
        j ← j-1
```

```
u.item[i] ← x
u.order ← u.order+1
return true
```