

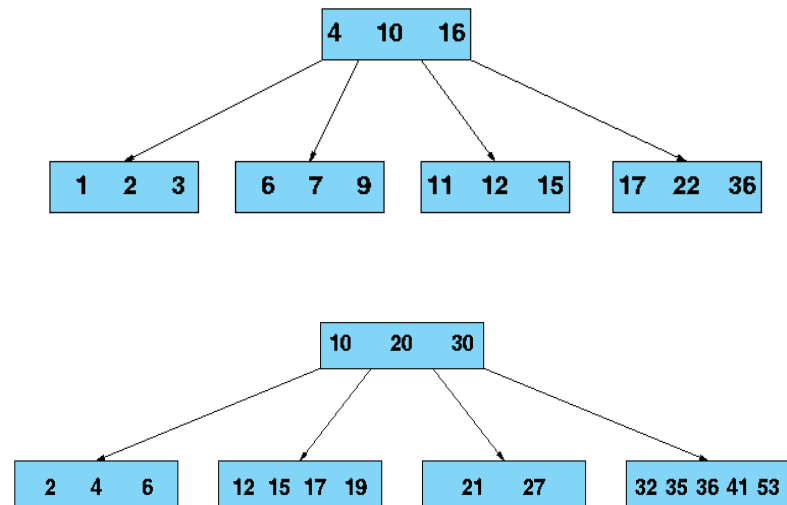
Definice B-stromu

- B-strom řádu m je strom, kde každý uzel má maximálně m následníků a ve kterém platí:
 1. Počet klíčů v každém vnitřním uzlu, je o jednu menší než je počet následníků (synů)
 2. Všechny listy jsou na stejné úrovni (mají stejnou hloubku)
 3. Všechny uzly kromě kořene mají nejméně $\lceil \frac{m}{2} \rceil$ následníků ($\lceil \frac{m}{2} \rceil - 1$ klíčů)
 4. Kořen je buďto list, nebo má od 2 do m následníků
 5. Žádný uzel neobsahuje více než m následníků ($m-1$ klíčů)

B-stromy

B-stromy jsou vyvážené stromy, které jsou optimalizovány pro případ, kdy je část stromu, popř. celý strom uložen na vnější paměti (např. magnetický disk). Vzhledem k tomu, že přístup na diskovou paměť je časově náročný, B-strom je navržen tak, aby optimalizoval (a minimalizoval) počet přístupů do vnější paměti.

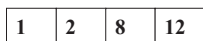
Příklad B-Stromu



Search(21)

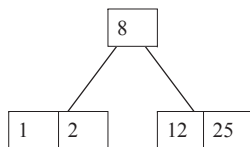
Vytvoření B-stromu

- Předpokládejme, že začínáme s prázdným B-stromem a ukládáme klíče v následujícím pořadí: 1 12 8 2 25 5 14 28 17 7 52 16 48 68 3 26 29 53 55 45
- Chceme vytvořit B-strom řádu $m=5$, tj. každý uzel (kromě kořene) obsahuje nejméně 2 klíče a nejvýše 4 klíče.
- První 4 klíče :

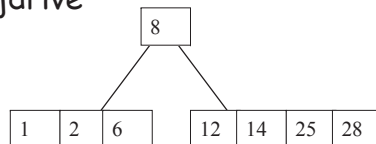


- Vložení páté položky (klíč 25) poruší podmínku 5 (dojde k tzv. přeplnění stránky)
- Přeplněná stránka se rozdělí na dvě, prostřední prvek se přesune do nadřazené stránky

Vytvoření B-stromu (pokračování)

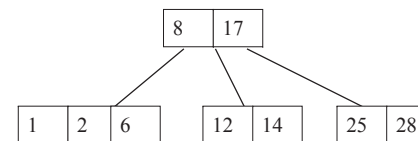


Další položky 6, 14, 28 budou vloženy do listů (listy se obsazují nejdříve)

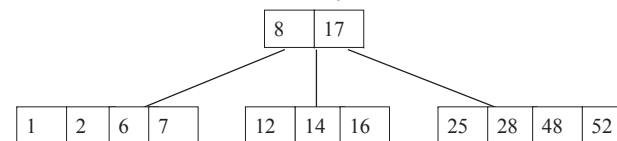


Vytváření B-stromu

Vložení 17 do pravé stránky způsobí přeplnění, stránka se rozdělí podle prostředního klíče a ten se přesune do kořene

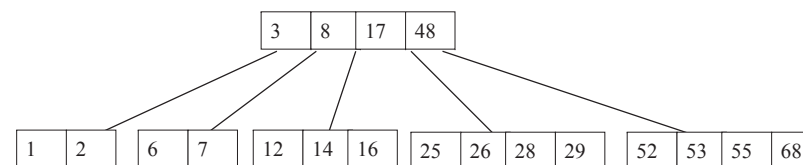


7, 52, 16, 48 se opět přidají do listů

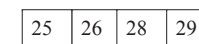


Vytváření B-stromu

Vložení 68 opět způsobí přeplnění stránky vpravo, klíč 48 se přesune do kořene, 3 přeplní levou stránku a po rozdělení přechází do kořene; 26, 29, 53, 55 jsou vloženy do listů

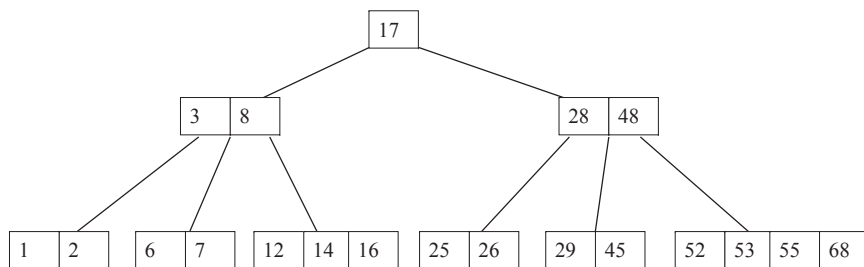


45 přeplní stránku



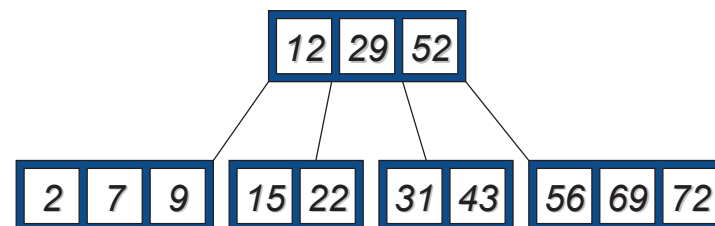
a klíč 28 se přesune do kořene, kde způsobí přeplnění a rozdělení kořenové stránky

Vytváření B-stromu



Rušení prvků B-stromu - rušení v listech bez podtečení velikosti stránky

Předpokládejme B-strom 5 řádu...



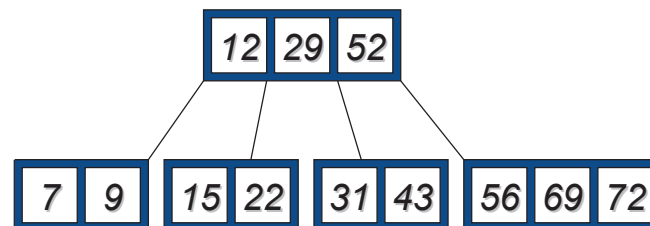
Zrušení klíče 2: Jelikož ve stránce je dostatečné množství klíčů, Dojde pouze ke zrušení hodnoty 2 v listové stránce

Vložení prvku do B-stromu (shrnutí)

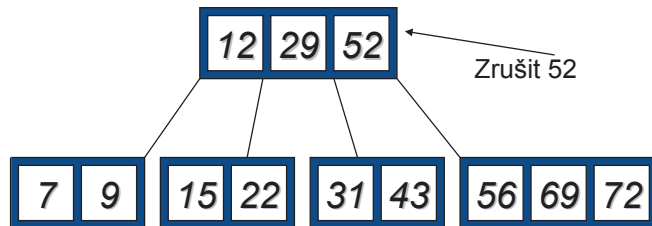
- Nový prvek se vždy vkládá do listové stránky, ve stránce se klíče řadí podle velikosti.
- Pokud dojde k přeplnění listové stránky, stránka se rozdělí na dvě a prostřední klíč se přesune do nadřazené stránky (pokud nadřazená stránka neexistuje, tak se vytvoří)
- Pokud dojde k přeplnění nadřazené stránky předchozí postup se opakuje dokud nedojde k zařazení nebo k vytvoření nového kořene.

Rušení prvků B-stromu - rušení v listech bez podtečení velikosti stránky

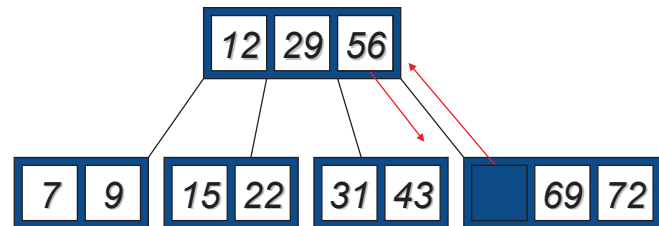
Předpokládejme B-strom 5 řádu...



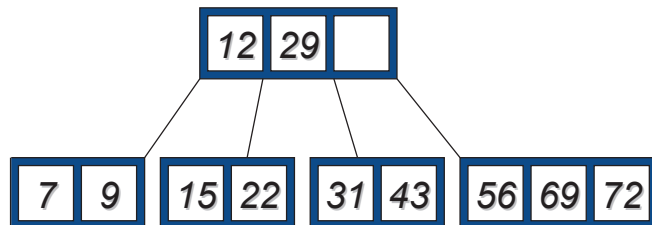
Rušení prvků B-stromu - rušení prvku v ostatních stránkách (kromě listů) bez podtečení velikosti stránky



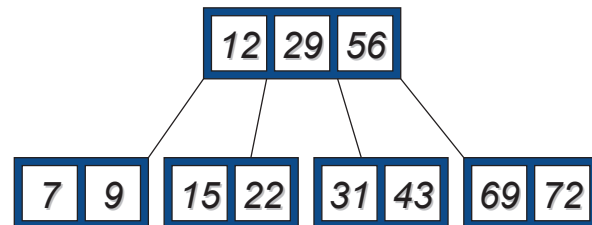
Rušení prvků B-stromu - rušení prvku v ostatních stránkách (kromě listů) bez podtečení velikosti stránky



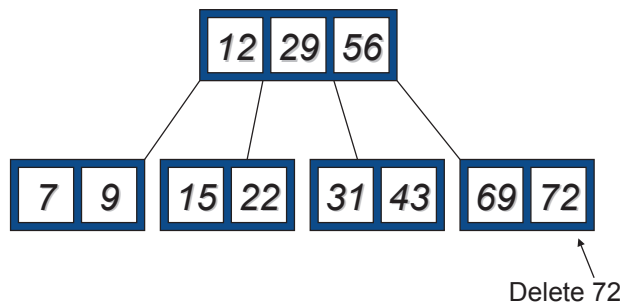
Rušení prvků B-stromu - rušení prvku v ostatních stránkách (kromě listů) bez podtečení velikosti stránky



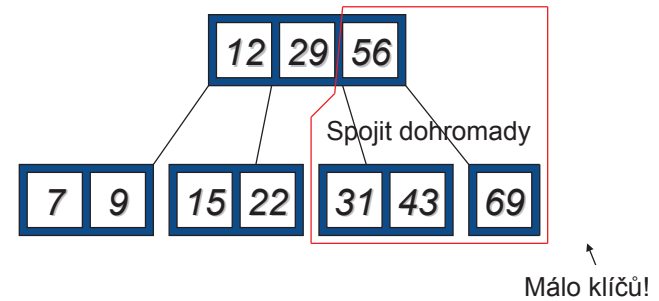
Rušení prvků B-stromu - rušení prvku v ostatních stránkách (kromě listů) bez podtečení velikosti stránky



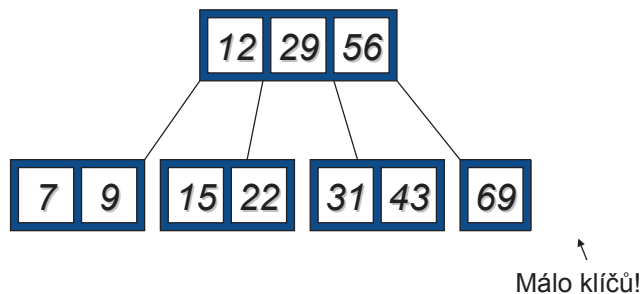
Rušení prvků ve stránce s minimálním počtem klíčů - sousední stránky obsahují také minimální počet klíčů



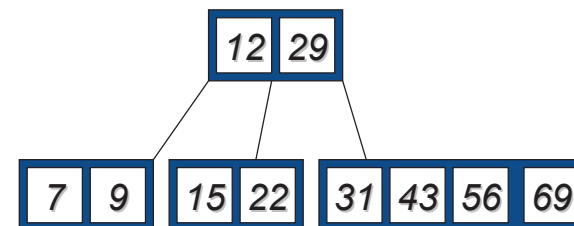
Rušení prvků ve stránce s minimálním počtem klíčů - sousední stránky obsahují také minimální počet klíčů



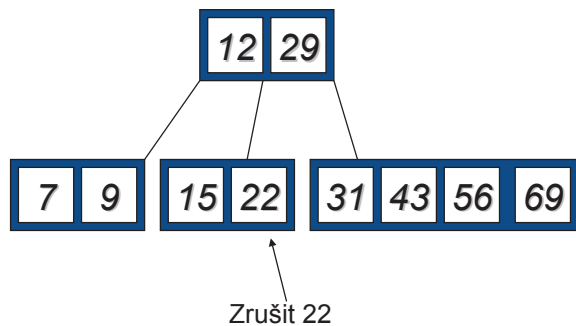
Rušení prvků ve stránce s minimálním počtem klíčů - sousední stránky obsahují také minimální počet klíčů



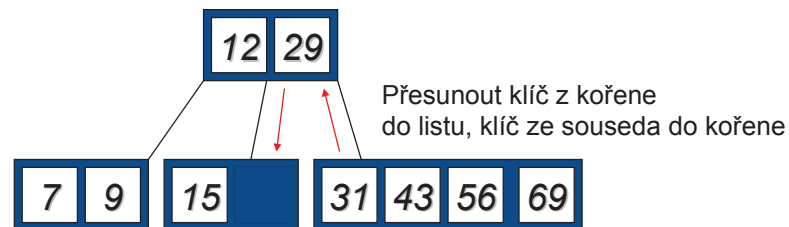
Rušení prvků ve stránce s minimálním počtem klíčů - sousední stránky obsahují také minimální počet klíčů



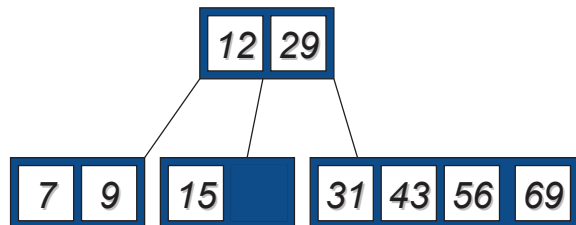
Rušení prvků ve stránce s minimálním počtem klíčů - sousední stránky obsahují dostatek klíčů



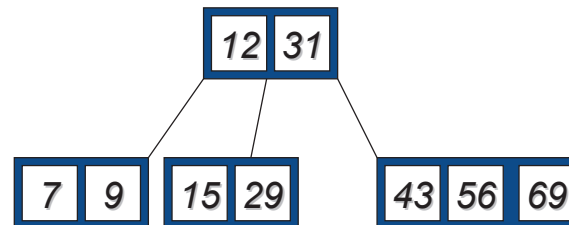
Rušení prvků ve stránce s minimálním počtem klíčů - sousední stránky obsahují dostatek klíčů



Rušení prvků ve stránce s minimálním počtem klíčů - sousední stránky obsahují dostatek klíčů



Rušení prvků ve stránce s minimálním počtem klíčů - sousední stránky obsahují dostatek klíčů



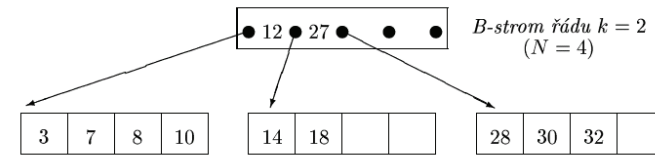
Analyza B-Stromů

- Maximální počet položek v B-stromu řádu m a výšky h :
 - kořen $m - 1$
 - úroveň 1 $m(m - 1)$
 - úroveň 2 $m^2(m - 1)$
 - ...
 - úroveň h $m^h(m - 1)$
- Celkový počet položek je

$$(1 + m + m^2 + m^3 + \dots + m^h)(m - 1) =$$

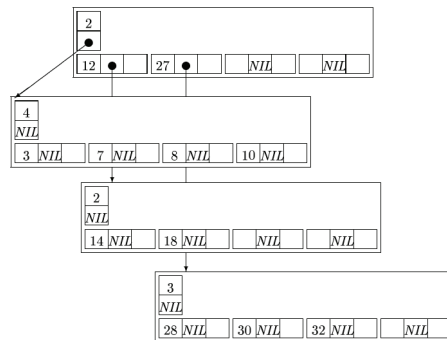
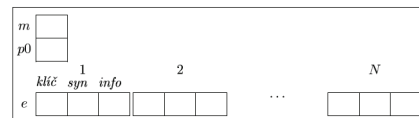
$$[(m^{h+1} - 1) / (m - 1)] (m - 1) = m^{h+1} - 1$$
- Pokud $m = 5$ a $h = 2$ tak je celkem $5^3 - 1 = 124$ položek

Implementace B-stromu -statická



kořen	aktuální počet klíčů	KLIC	SYN	INFO
3		1 ... N	1 ... N + 1	1 ...
1	3	1 28 30 32	1 NILNILNILNILNIL	1
2	4	2 3 7 8 10	2 NILNILNILNILNIL	2
3	2	3 12 27	3 2 5 1 NILNIL	3
4		4	4	4
5	2	5 14 18	5 NILNILNILNILNIL	5
volné				
4				

Implementace B-stromu - dynamická



Vytvoření B-stromu

B-Tree-Create(T)

```

x <- Allocate-Node()
leaf[x] <- TRUE
n[x] <- 0
Disk-Write(x)
root[T] <- x
    
```

Vložení prvku do B-Stromu

B-Tree-Insert(T, k)

```
r <- root[T]
if n[r] = 2t - 1
  then s <- Allocate-Node()
  root[T] <- s
  leaf[s] <- FALSE
  n[s] <- 0
  c1 <- r
  B-Tree-Split-Child(s, 1, r)
  B-Tree-Insert-Nonfull(s, k)
else B-Tree-Insert-Nonfull(r, k)
```

Vložení prvku do B-stromu

B-Tree-Insert-Nonfull(x, k)

```
i <- n[x]
if leaf[x]
  then while i >= 1 and k < keyi[x]
    do keyi+1[x] <- keyi[x]
    i <- i - 1
    keyi+1[x] <- k
    n[x] <- n[x] + 1
    Disk-Write(x)
  else while i >= 1 and k < keyi[x]
    do i <- i - 1
    i <- i + 1
    Disk-Read(ci[x])
    if n[ci[x]] = 2t - 1
      then B-Tree-Split-Child(x, i, ci[x])
      if k > keyi[x]
        then i <- i + 1
    B-Tree-Insert-Nonfull(ci[x], k)
```

Rozdělení stránky

B-Tree-Split-Child(x, i, y)

```
z <- Allocate-Node()
leaf[z] <- leaf[y]
n[z] <- t - 1
for j <- 1 to t - 1
  do keyj[z] <- keyj+t[y]
if not leaf[y]
  then for j <- 1 to t
    do cj[z] <- cj+t[y]
n[y] <- t - 1
for j <- n[x] + 1 downto i + 1
  do cj+1[x] <- cj[x]
ci+1 <- z
for j <- n[x] downto i
  do keyj+1[x] <- keyj[x]
keyi[x] <- keyt[y]
n[x] <- n[x] + 1
Disk-Write(y)
Disk-Write(z)
Disk-Write(x)
```

Vyhledávání v B-Stromu

B-Tree-Search(x, k)

```
i <- 1
while i <= n[x] and k > keyi[x]
  do i <- i + 1
if i <= n[x] and k = keyi[x]
  then return (x, i)
if leaf[x]
  then return NIL
else Disk-Read(ci[x])
  return B-Tree-Search(ci[x], k)
```