Strom

- Datová struktura, která uchovává objekty hierarchicky ve vztahu předchůdce-následovník
- Stromy patří mezi často používané datové struktury v mnoha oblastech komputer science

Definice: Strom je graf, ve kterém existuje pouze jedna cesta z kořene, do kteréhokoliv dalšího uzlu grafu.

- **Kořen** - uzel který nemá předchůdce, ve stromu může být pouze jedniný kořen
- **Listy** (vnější uzly) – uzly které nemají žádného následníka
- **Vnitřní uzly** – uzly které mají alespoň jednoho následníka
- **Cesta** – je-li $n_1,n_2,\dots n_k$ množina uzlů ve stromu takových, že n_i je předchůdce n_{i+1}, pro $1 \leq i \leq k$, pak se tato množina nazývá cesta z uzlu n_1 do n_k
- **Délka cesty** – počet hran, které spojují uzly cesty
- **Hloubka uzlu** – délka cesty od kořene do uzlu
- **Výška stromu** – délka cesty od kořene k nejhlubšímu uzlu (největší hloubka)

Typy stromů a implementace

Obecný strom – vnitřní uzly stromu mohou mít libovolný počet následníků. Implementuje se výhradně jako dynamická struktura – seznam zřetězených prvků

n-ární strom – vnitřní uzly stromu obsahují maximálně n následníků. Implementuje se obvykle jako seznam zřetězených prvků, kde každý uzel stromu obsahuje pole ukazatelů na následníky (syny)

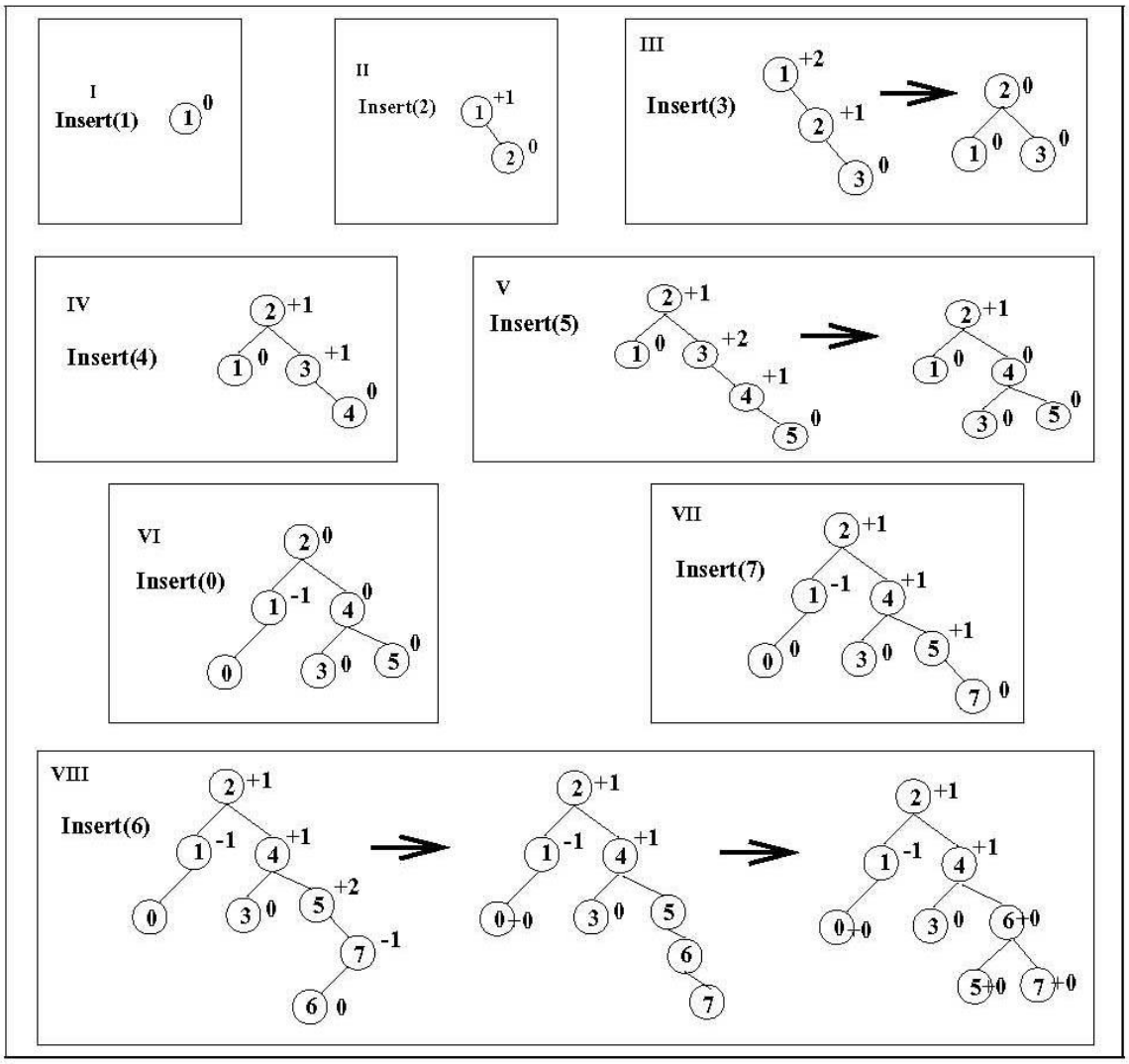
binární strom – speciální případ n-árního stromu, každý vnitřní uzel může mít nejvýše dva následníky (syny). Implementuje se jako seznam zřetězených prvků, někdy jako pole.

- **BVS:** v levym podstromu jsou prvky mensi, nez dany vrchol, v pravym vetsi, nez vrchol, tento strom se nevyvazuje (muze degenerovat) INSERT jednoduchy, DELETE nahradime nejvyssim nizsim nebo nejnizsim vyssim. Levý podstrom uzlu obsahuje pouze klíče menší než je klíč tohoto uzlu. Pravý podstrom uzlu obsahuje pouze klíče větší než je klíč tohoto uzlu.

AVL strom (Adelson-Velskii, Landys) vyvážený strom

AVL strom je výškově vyvážený binární vyhledávací strom, pro který platí, že pro libovolný vnitřní uzel stromu se výška levého a pravého syna liší nejvýše o 1.

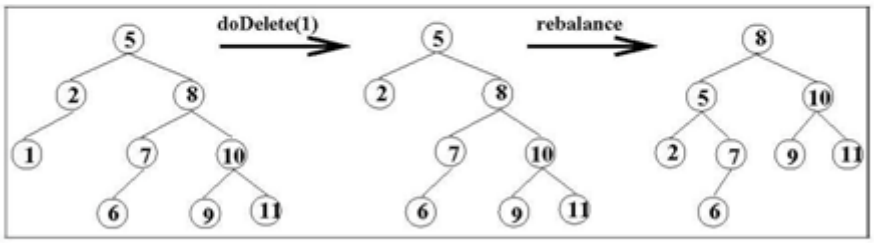
- Vyváženost AVL stromu se kontroluje po každé operaci vložení a zrušení prvku, v případě že je vyváženost porušena, provádí se opětovné vyvážení pomocí jedné popř. několika rotací v jednotlivých částech stromu.
- Implementace je obdobná jako u BVS, datová struktura pro uzel stromu je doplněna o celočíselnou proměnnou reprezentující stupeň vyváženosti uzlu, který může nabývat následujících hodnot:
    - 0 – oba podstromy jsou stejně vysoké
    - 1 – pravý podstrom je o 1 vyšší
    - 2 – pravý podstrom je o 2 vyšší
    - -1 – levý podstrom je o 1 vyšší
    - -2 – levý podstrom je o 2 vyšší
- Rotace :
    - Jednoduchá pravá (levá) – používáme pokud vyvažujeme přímou větev, tj. jsou-li znaménka stupně vyváženosti stejná
    - Dvojitá pravá (levá) – používá se tehdy pokud nejde použít jednoduchá rotace – vyvažujeme-li „zalomenou" větev.



Vložení prvků {1,2,3,4,5,0,7,6} do AVL stromu

Zrušení prvků v AVL stromu

- Rušení prvků je podobné jako u BVS stromu, po zrušení prvku je nutné provést kontrolu stupně vyváženosti uzlů směrem ke kořeni a v případě že je to nutné znovu vyvážit strom pomocí jednoduchých popř. dvojnásobných rotací

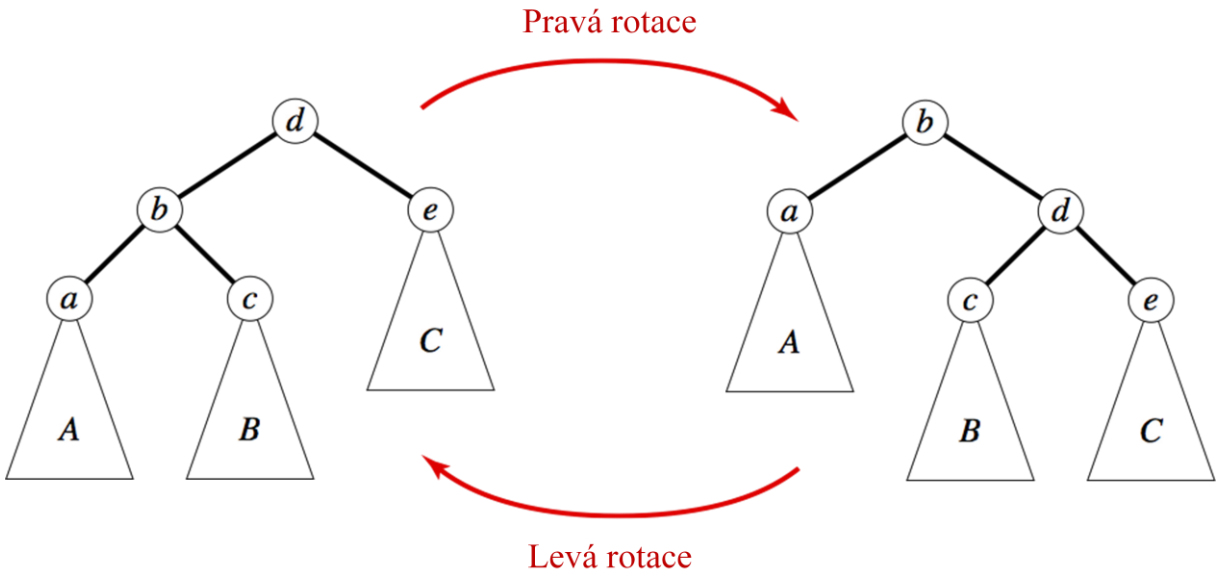


Red-Black Stromy

Binární vyhledávací stromy, u kterých je časová složitost operací v nejhorším případě rovná O(log n)

Vlastnosti:

- 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



B-Strom

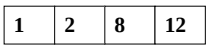
B-stromy jsou vyvážené stromy, které jsou optimalizovány pro případ, kdy je část stromu, popř. celý strom uložen na vnější paměti (např. magnetický disk). Vzhledem k tomu, že přístup na diskovou paměť je časově náročný, B-strom je navržen tak, aby optimalizoval (a minimalizoval) počet přístupů do vnější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

- 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 (*m*/2) následníků ((*m*/2)-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)

Příklad

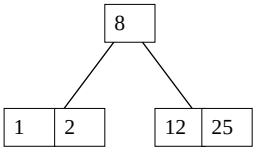
Vytvoření B-stromu

- 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 :

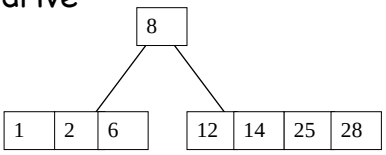


- Vložení páté položky (klíč 25) poruší podmínku 5 (dojde k tzv. přeplnění stránky)
- Přeplněná stránka se rozdělí na dvě, prostřední prvek se přesune do nadřazené stránky

Vytvoření B-stromu (pokračování)

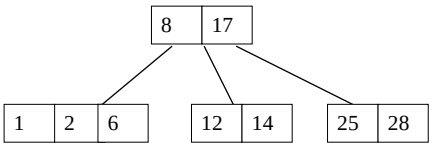


Další položky 6, 14, 28 budou vloženy do listů (listy se obsazují nejdříve

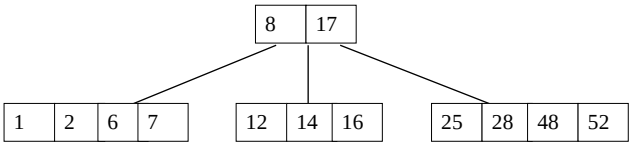


Vytváření B-stromu

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

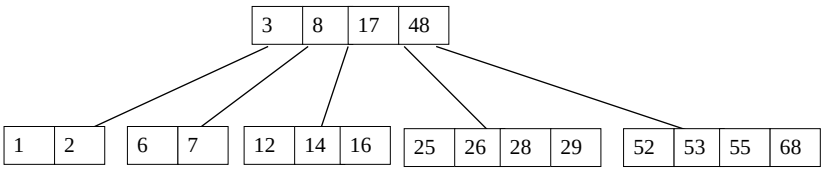


7, 52, 16, 48 se opět přidají do listů

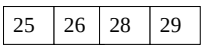


Vytváření B-stromu

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

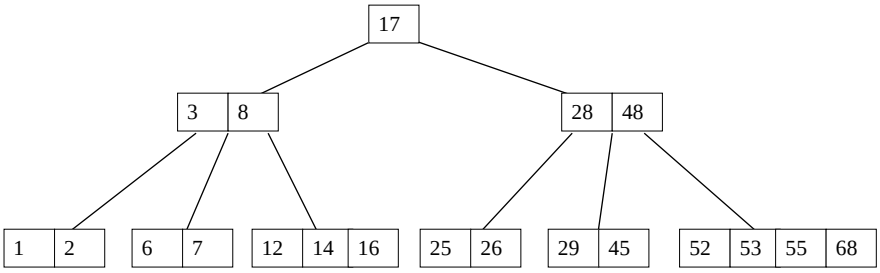


45 přeplní stránku



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

Vytváření B-stromu

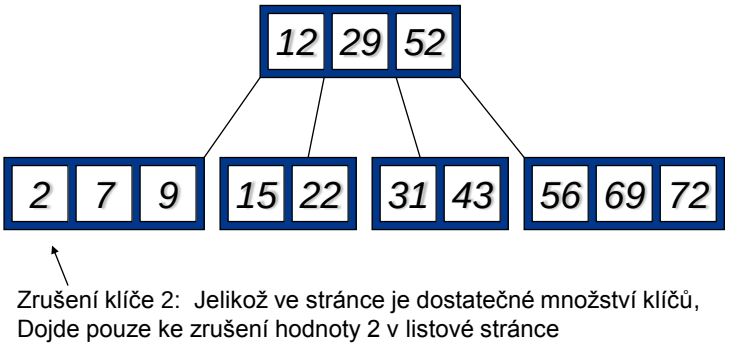


Shrnutí vkládání prvku:

- Nový prvek se vždy vkládá do listrové 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

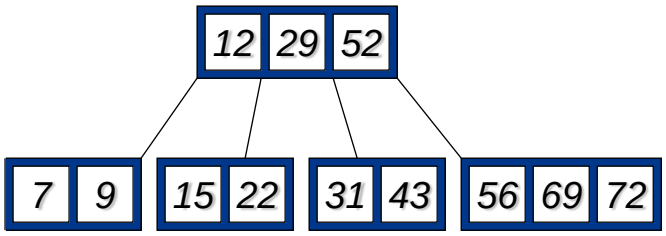
Předpokládejme B-strom 5 řádu...



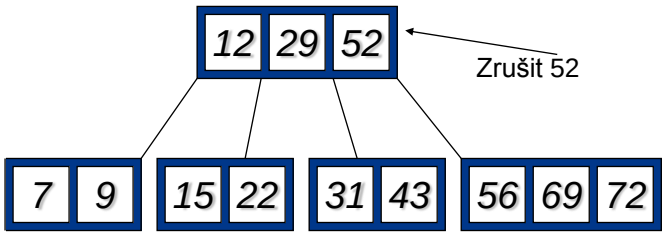
Zrušení klíče 2: Jelikož ve stránce je dostatečné množství klíčů,
Dojde pouze ke zrušení hodnoty 2 v listové stránce

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

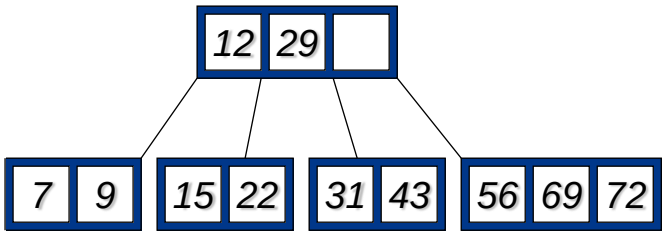
Předpokládejme B-strom 5 řádu...



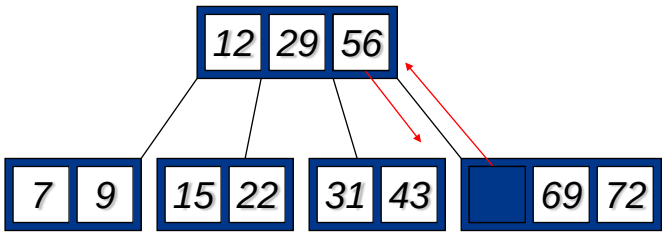
Rušení prvků B-stromu - rušení pvku v ostatních stránkách (kromě listů) bez podtečení velikosti stránky



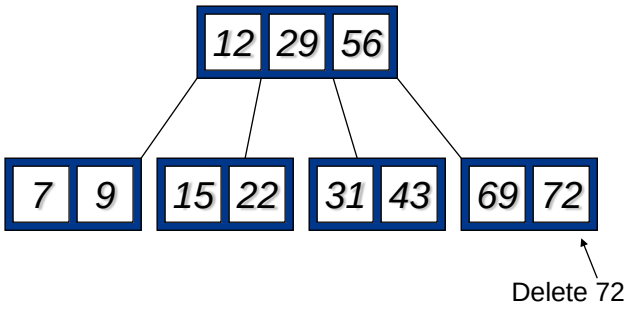
Rušení prvků B-stromu - rušení pvku v ostatních stránkách (kromě listů) bez podtečení velikosti stránky



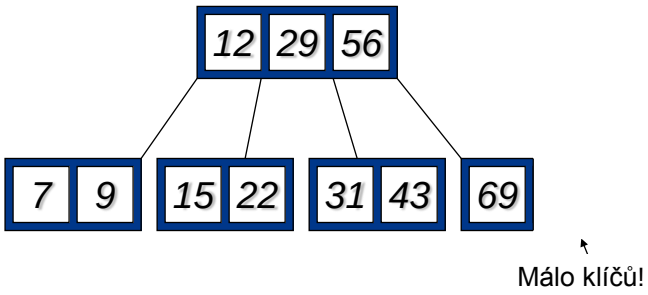
Rušení prvků B-stromu - rušení pvku v ostatních stránkách (kromě listů) bez podtečení velikosti stránky



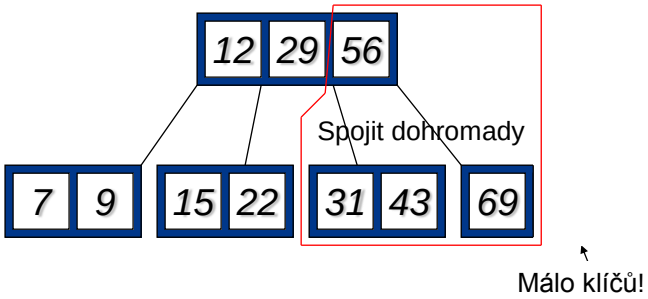
Rušení prvků ve stránce s minimálním počtem klíčů – sousední stránky obsahují také minimální počet klíčů



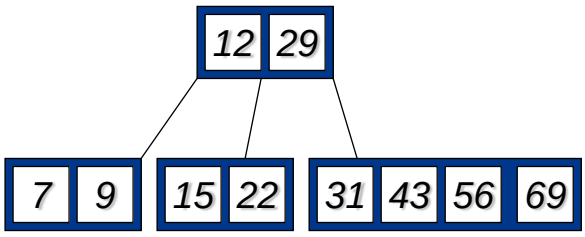
Rušení prvků ve stránce s minimálním počtem klíčů – sousední stránky obsahují také minimální počet klíčů



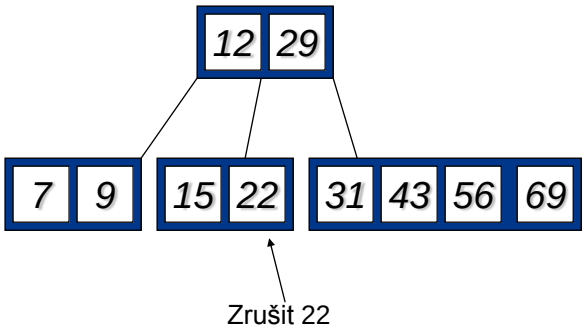
Rušení prvků ve stránce s minimálním počtem klíčů – sousední stránky obsahují také minimální počet klíčů



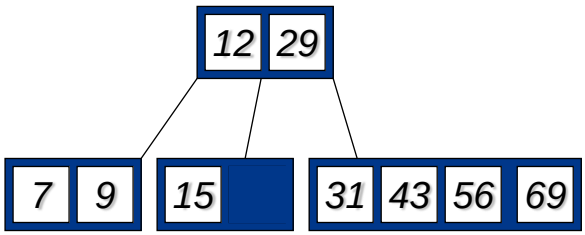
Rušení prvků ve stránce s minimálním počtem klíčů – sousední stránky obsahují také minimální počet klíčů



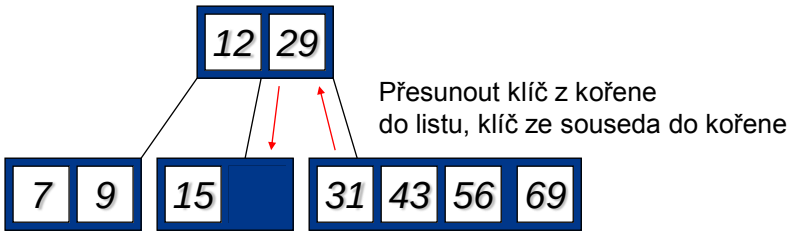
Rušení prvků ve stránce s minimálním počtem klíčů – sousední stránky obsahují dostatek klíčů



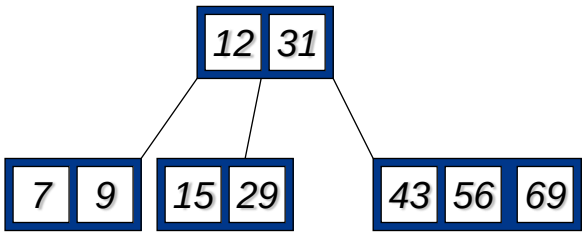
Rušení prvků ve stránce s minimálním počtem klíčů – sousední stránky obsahují dostatek klíčů



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



B-Tree-Search(x, k)

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