B-stromy

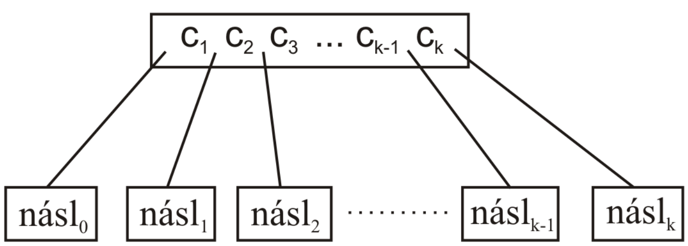
B-stromy jsou velmi významným typem vyhledávacích stromů. Na rozdíl od doposud uváděných stromů, kdy v každém uzlu byl uložen jeden prvek, B-stromy mají v uzlech uloženo více prvků. Struktura B-stromů je definována následujícími vlastnostmi:

- Kapacita uzlu (počet prvků, které lze do uzlu uložit) je u všech uzlů stromu stejná a volíme ji před začátkem vytváření stromu, kapacitu označme r ($r \geq 2$). Protože v B-stromech má významnou úlohu polovina z tohoto počtu, zaveďme pro ni samostatné označení $p = \frac{r}{2}$ pro r sudé a $p = \frac{r-1}{2}$ pro r liché.
- Důležité pro efektivní využití uzlů je jejich zaplnění. Všechny uzly vyjma kořene musí být aspoň z poloviny zaplněny prvky pro r sudé nebo p prvky pro r liché, tedy počet prvků v uzlech uložených musí být v rozmezí p až r prvků. Jedině u kořene stačí, aby obsahoval aspoň jeden prvek, tedy jeho zaplnění je v rozmezí 1 až r prvků.
- Prvky uložené v uzlu jsou v něm seřazeny vzestupně dle velikosti.

$$\boxed{c_1\ c_2\ c_3\ \dots\ c_{k-1}\ c_k}$$

$$c_1 < c_2 < c_3 < \dots < c_{k-1} < c_k$$

- Uzel je buďto list anebo má o jednoho následníka více, než je počet prvků v něm uložený.



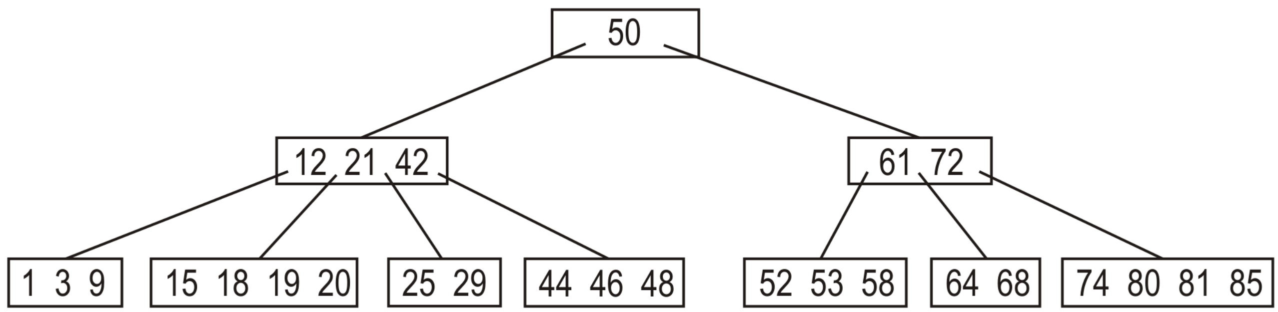
Přitom pro prvky v jednotlivých podstromech, které těmito následníky začínají, platí:

- Pro každý prvek d v podstromu začínající uzlem $násl_0$ platí $d < c_1$.
- Pro každý prvek d v podstromu začínající uzlem $násl_1$ platí $c_1 < d < c_2$.
- Pro každý prvek d v podstromu začínající uzlem $násl_2$ platí $c_2 < d < c_3$.
- …………
- Pro každý prvek d v podstromu začínající uzlem $násl_{k-1}$ platí $c_{k-1} < d < c_k$.
- Pro každý prvek d v podstromu začínající uzlem $násl_k$ platí $d > c_k$.

- A zbývá nám poslední vlastnost, jež určuje vyváženost B-stromu. Ta stanoví, že listy jsou v B-stromu jen v jeho poslední (nejspodnější) úrovni. Tj. vzdálenost všech listů od kořene je shodná a je rovna výšce stromu.

Podle maximálního počtu následníků označujeme i řád B-stromu. B-strom s kapacitou uzlu r má řád $r+1$, neboť nelistový uzel může mít až $r+1$ následníků.

Příklad. Na následujícím obrázku je B-strom pro $r=4$. Prvky uložené v tomto stromu jsou celá čísla.



B-stromy vznikly v roce 1969, tedy 7 let po vzniku AVL stromů, jež pocházejí z roku 1962. B-stromy jsou hodně používány v databázových systémech.

Postup vyhledávání

1. Počáteční krok

Uzel, který je v daném okamžiku vyhledávání aktuální, budeme označovat u. Na začátku jím bude kořen stromu. Hledaná hodnota nechť je x.

2. Průběžný krok

Uděláme vyhledání hodnoty x mezi prvky uloženými v aktuálním uzlu u. Protože prvky jsou v uzlu setříděné, lze k tomu použít binární vyhledávání. To použijeme v případě, kdy kapacita uzlů je dostatečně velká, aby se to vyplatilo (např. $r \geq 10$). Vyhledání může skončit třemi způsoby:

- Prvek byl v aktuálním uzlu u nalezen, čímž vyhledávání úspěšně končí.
- Prvek nebyl v aktuálním uzlu u nalezen a tento uzel je list. Tím vyhledávání končí – hledaný prvek není ve stromu obsažen.
- Prvek nebyl v aktuálním uzlu u nalezen a tento uzel je nelistový. V tom případě vyhledávání skončilo v místě, kde je odkaz na následníka, ve kterém vyhledávání má pokračovat (tj. na následníka, kterým začíná podstrom, jež by hledaný prvek měl obsahovat). Tohoto následníka učiníme novým aktuální uzlem a opět se udělá krok 2.

Postup přidání prvku

Chceme-li do B-stromu přidat prvek, znamená to najít příslušný uzel, do kterého nový prvek patří, a následně ověřit, zda přitom nedošlo k přeplnění, a pokud ano, provést rozdělení uzlu.

1. Přidání prvku

Označme přidávaný prvek *x*. Uděláme jeho vyhledání ve stromu. Použijeme k tomu již popsaný algoritmus pro vyhledání. Ten může skončit dvěma způsoby:

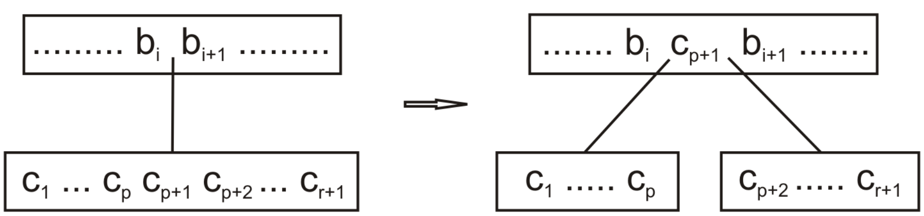
- Prvek *x* byl ve stromu nalezen. Tím přidávání končí, neboť prvek *x* už je ve stromu obsažen (u vyhledávacích stromů se nepředpokládá vícenásobný výskyt stejného prvku).
- Vyhledávání skončilo v listovém uzlu *u* v místě, kam nový prvek podle velikosti vzhledem k ostatním prvků patří. Prvek na toto místo vložíme. Pokud uzel *u* předtím nebyl zcela zaplněn, operace přidání tím končí. Jinak následuje rozdělení uzlu.

2. Rozdělení uzlu

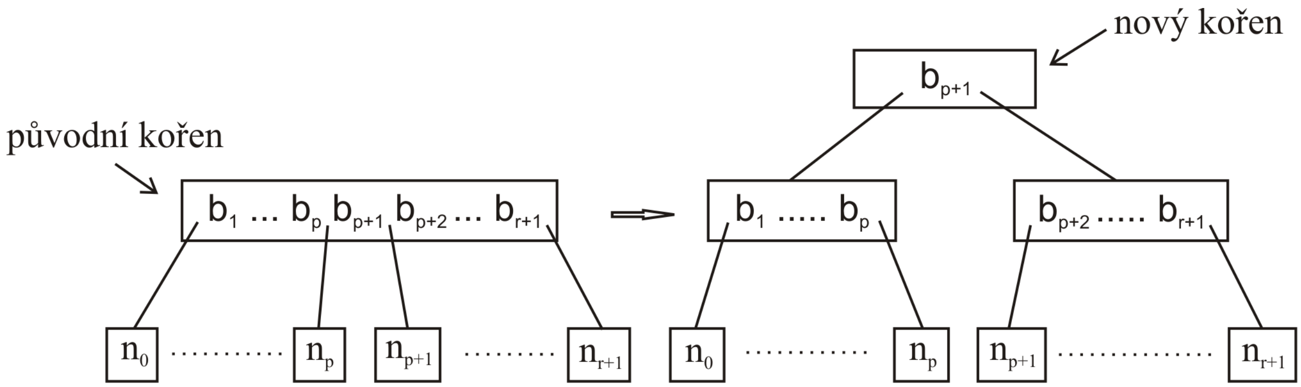
Jestliže uzel *u* má po přidání $r+1$ prvků, tedy došlo k jeho přeplnění, rozdělíme ho na tři části:

p prvků na začátku uzlu + prvek uprostřed uzlu + p prvků na konci uzlu.

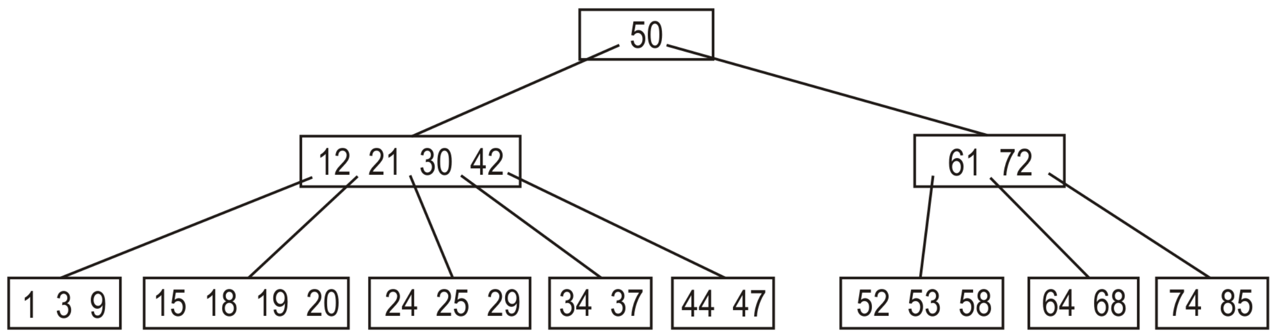
Části s p prvky budou tvořit nové listy. Prvek, jež je v uzlu *u* uprostřed, se přesune do předchůdce na místo, kde byl původní odkaz na list. Po přesunu vytvoříme nalevo a napravo od něho odkazy na nově vzniklé listy.



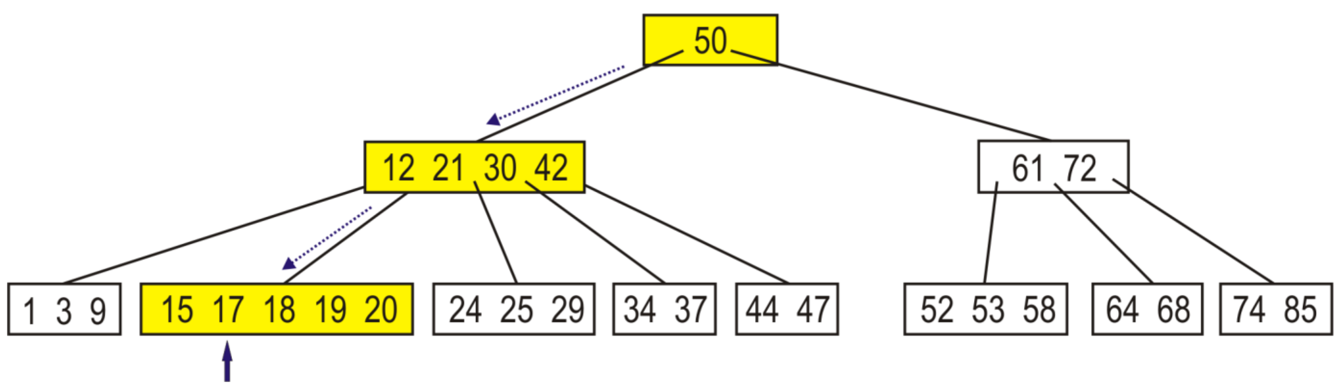
Zřejmě po přidání uzlu do předchůdce může v něm dojít rovněž k jeho přeplnění, pokud předtím byl zcela zaplněn. To se řeší stejným způsobem – rozdělením tohoto uzlu na tři části s počtem prvků $p+1+p$. Dvě jeho části s p prvky budou tvořit nové uzly a zbývající prostřední prvek vložíme do jeho předchůdce. Takto postupujeme směrem nahoru, až buďto narazíme na uzel, u kterého po vložení dalšího prvku nedojde k přeplnění, anebo se nakonec dostaneme až ke kořenu. Pokud i u něho dojde k přeplnění, rozdělí se opět na tři části, přičemž střední prvek bude umístěn do samostatného uzlu, který bude novým kořenem. V této situaci dojde ke zvětšení výšky stromu.



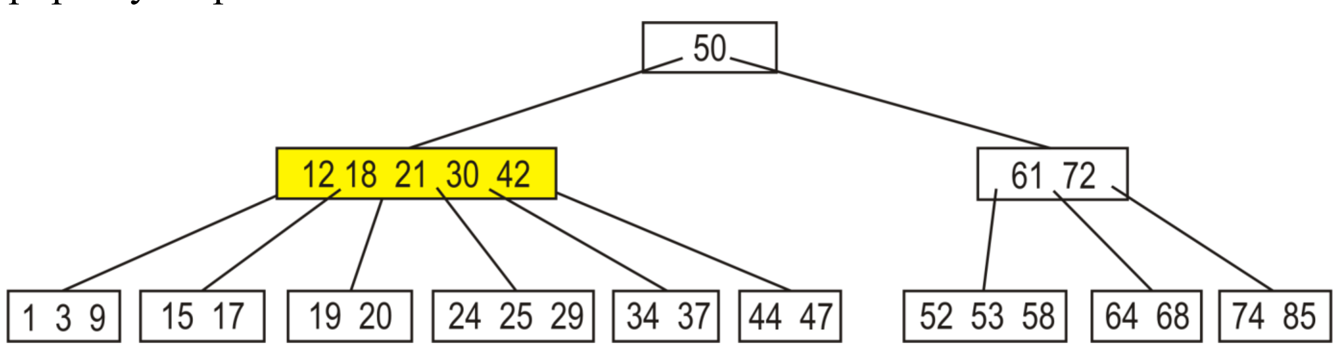
Příklad. K následujícímu B-stromu ($r = 4$)



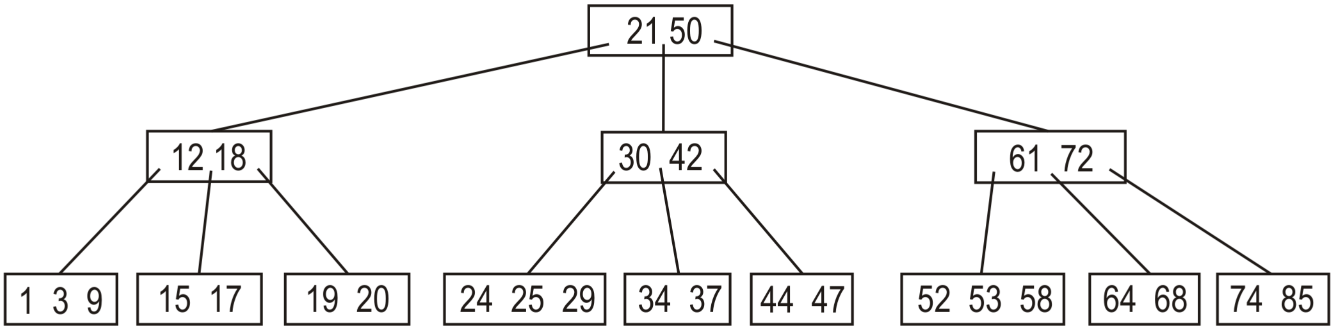
přidáme prvek 17. Následující obrázek ukazuje postup vyhledání příslušného místa, kam prvek patří, a vložení prvku.



Protože list, do kterého byl prvek 17 vložen, nyní obsahuje 5 prvků, je zapotřebí ho popsaným způsobem rozdělit.



Je zřejmé, že nyní je přeplněn předchozí uzel, do něhož byl přesunut střední prvek z rozděleného listu. Je zapotřebí rozdělit i tento uzel.



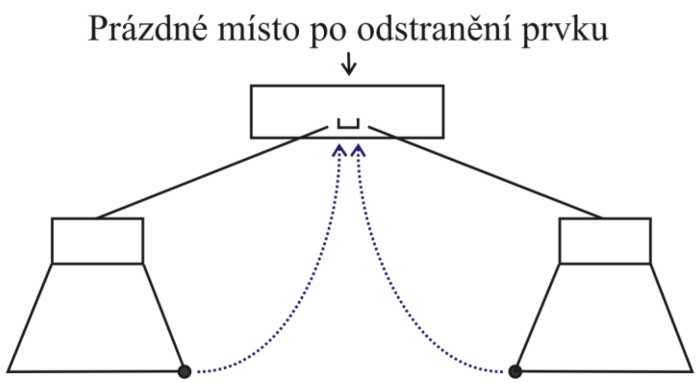
Postup odebrání prvku

Chceme-li z B-stromu odebrat prvek, znamená to vyhledat uzel, ve kterém se prvek nachází, prvek z něho odstranit a následně ověřit, zda odebráním prvku nepoklesl počet prvků v daném uzlu pod přípustnou mez, a pokud ano, je nutné to vyřešit.

1. Odebrání prvku

Označme odstraňovaný prvek *x*. Uděláme jeho vyhledání ve stromu. Použijeme k tomu již uvedený algoritmus pro vyhledání. Ten může skončit třemi způsoby:

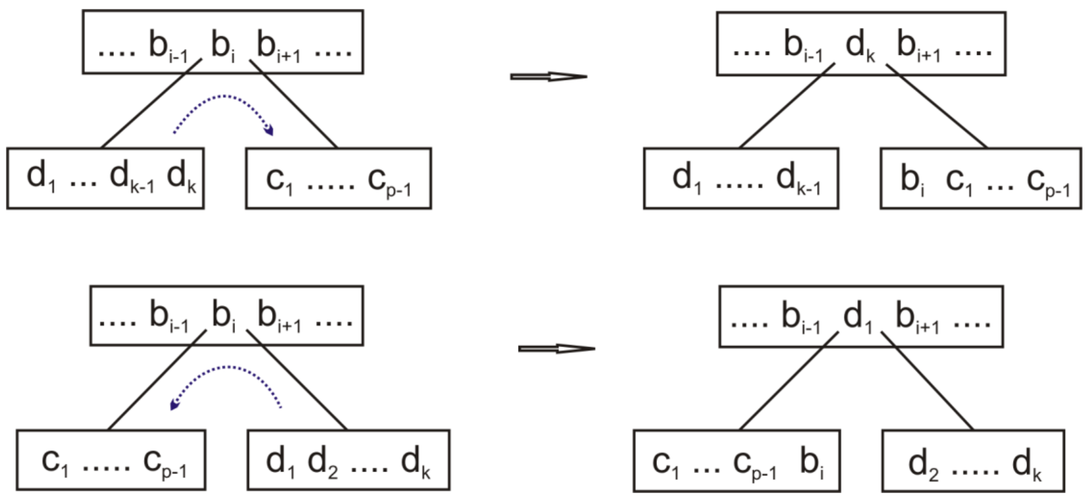
- Prvek *x* nebyl ve stromu nalezen – není co odebrat.
- Prvek *x* byl nalezen v listovém uzlu. Prvek z uzlu odstraníme. Pokud list je po zrušení prvku aspoň z poloviny zaplněn, operace odebrání končí. Jinak přejdeme ke kroku 2.
- Prvek *x* byl nalezen v uzlu *u*, který není listem. Prvek *x* z uzlu odstraníme a na volné místo v uzlu *u* přesuneme buďto největší prvek z jeho levého podstromu, což je poslední prvek v nejpravějším listu levého podstromu, anebo na volné místo přesuneme nejmenší prvek z jeho pravého podstromu, což je první prvek v nejlevějším listu pravého podstromu. Pokud list, odkud jsme prvek přesunuli, je stále aspoň z poloviny zaplněn, operace odebrání prvku končí. Jinak přejdeme ke kroku 2.



2. Nedostatečný počet prvků v uzlu

Sem se dostáváme v situaci, kdy po odstranění prvku ze stromu je nyní ve stromu list *v*, který má jen *p*-1 prvků. Jak se tento stav řeší, závisí na zaplnění sourozenců listu. (Sourozenec uzlu *v* je uzel, který s ním sousedí a má i stejného předchůdce – rodiče.) Jsou dvě možnosti:

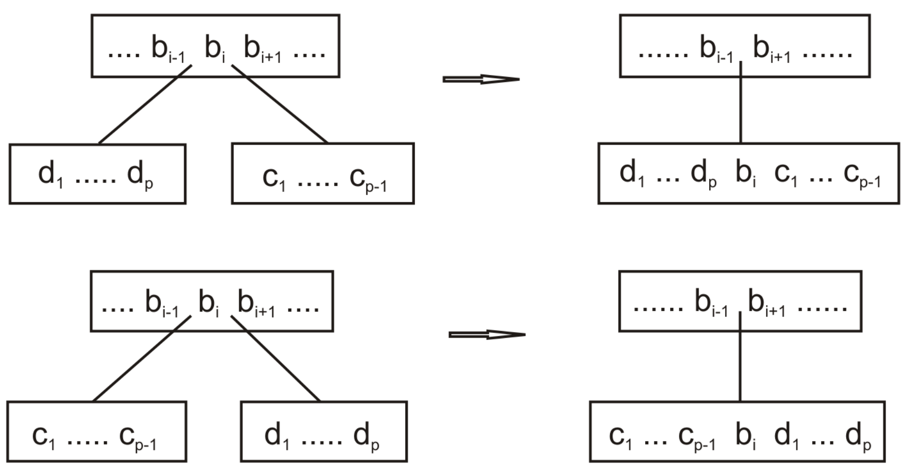
- List *v* má aspoň jednoho sourozence, který má více než *p* prvků. Pak do listu *v* přesuneme prvek z předchůdce a na prázdné místo v předchůdci přesuneme příslušný prvek ze souseda. Následující obrázek ukazuje tento přesun pro oba možné sourozence, nejdříve pro levého sourozence, pak pro pravého sourozence (prvku v uzlu *v* jsou značeny písmenem *c*).



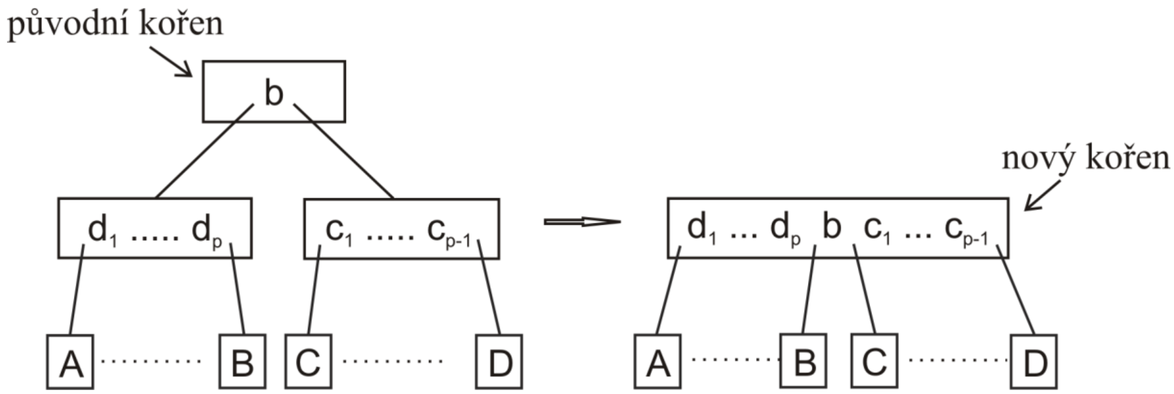
- List *v* má jen sourozence, které mají právě *p* prvků. Pak vytvoříme nový list s *r* prvky tak, že sloučíme

  prvky z listu *v* + prvek z předchůdce + prvky ze sourozence.

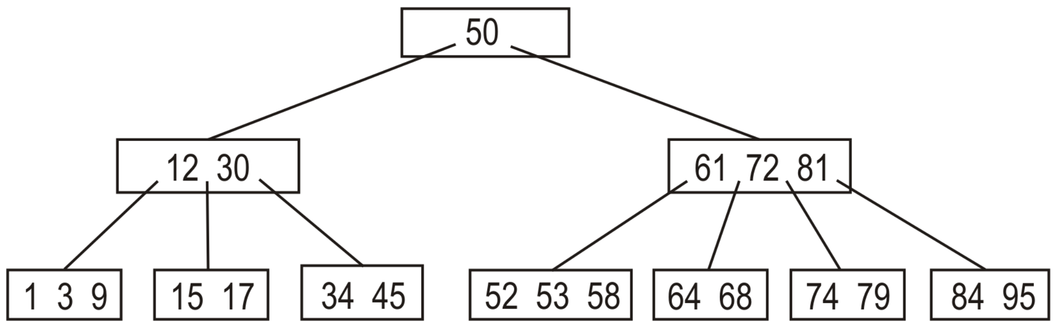
  Následující obrázek ukazuje toto sloučení opět pro oba možné sourozence – levého i pravého (prvku uzlu *v* jsou značeny písmenem *c*).



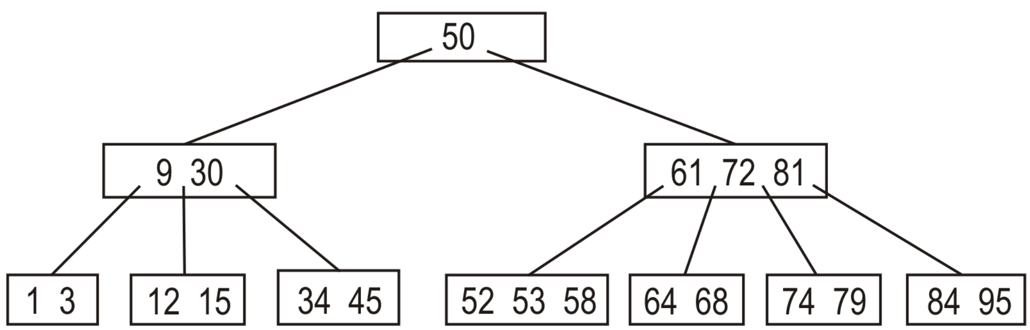
Je zřejmé, že tímto ubyl jeden prvek v předchůdci. Pokud tento má nyní jen *p*-1 prvků, řeší se to analogicky v závislosti na tom, kolik prvků mají jeho sourozenci. Takto se můžeme případně dostat až k uzlu, nad kterým už je jen kořen. Pokud dojde k jeho sloučení se sourozencem a prvkem z kořenu a v kořenu přitom byl jen tento jeden prvek, stane se uzel vytvořený sloučením novým kořenem a dojde zároveň ke snížení výšky stromu. Na následujícím obrázku je tato situace pro případ, kdy je nedostatečně obsazený uzel sloučen s levým sourozencem.



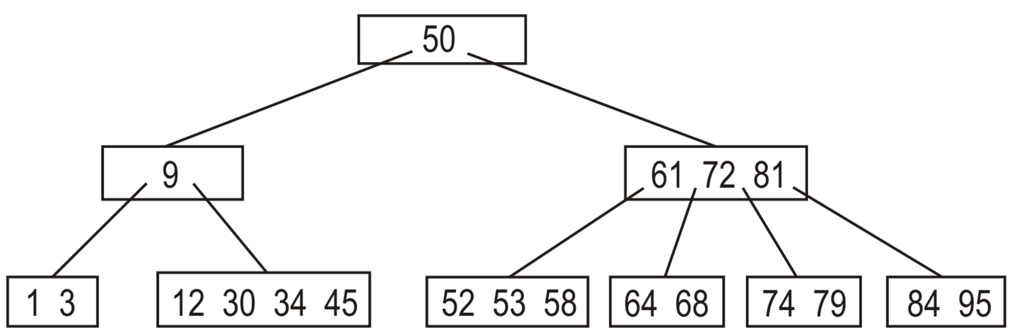
Příklad. Z následujícího B-stromu ($r = 4$)



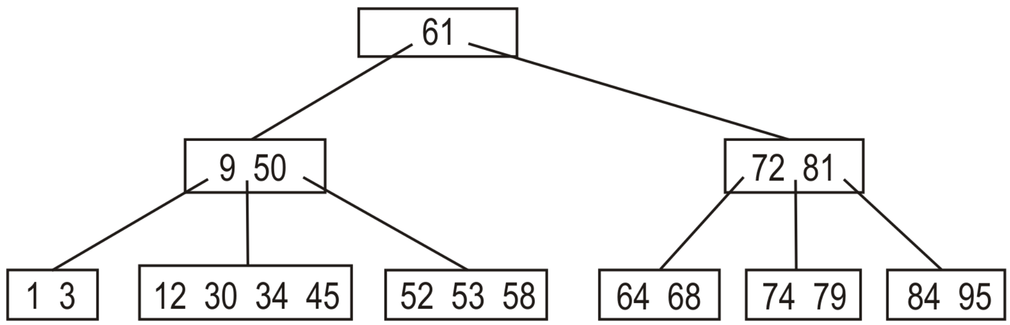
odebereme prvek 17. Po odstranění prvku z daného listu zůstane jen prvek 15. Tento list má ale levého sourozence, který má více než 2 prvky. Proto provedeme přesun prvku 9 z jeho levého sourozence do předchůdce a prvku 12 z předchůdce do listu s prvkem 15.



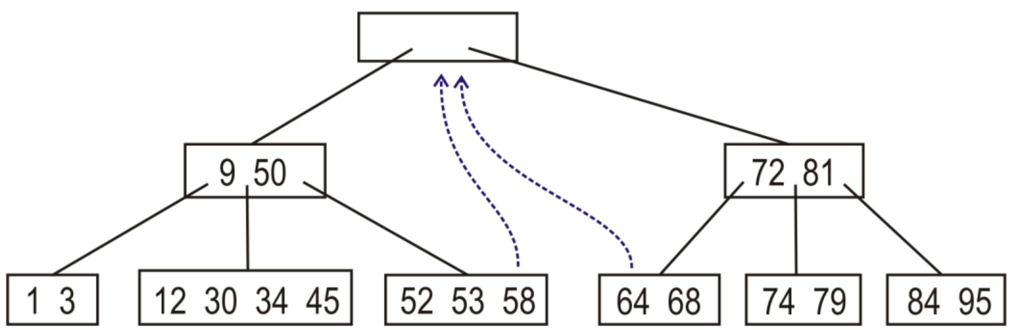
Odebereme dále prvek 15. V listu zůstane opět jen jeden prvek 12. List už ale nemá žádného sourozence, z kterého by se dal přesunout prvek, proto uděláme sloučení se sousedním listem. Vybereme si k tomu třeba pravého souseda.



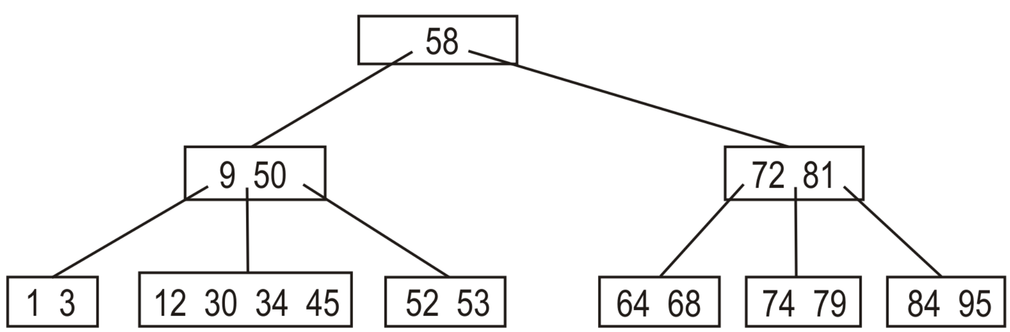
Při slučování byl odebrán prvek z předchůdce, ve kterém tímto zůstal jen jeden prvek 9. Tento uzel má ale pravého sourozence, který má více než dva prvky, čímž můžeme z něho prvek přesunout. Celý přesun se provede tak, že do uzlu s prvkem 9 se přesune prvek z jeho předchůdce (v tomto případě kořene) a na volné místo v předchůdci se přesune onen zmíněný prvek ze souseda. Přitom se příslušně přesune i ukazatel na jeho následníka – na list s prvky 52,53,58.



Odebereme prvek 61. Tento prvek je v nelistovém uzlu, jsou dvě možné náhrady, jak ukazuje následující obrázek.



Nahradíme ho třeba prvkem 58 z levého podstromu.



Časová složitost operací

Vezměme operaci vyhledávání. Ta jednak zahrnuje vyhledávání v uzlu. Pro ně je použito binární vyhledávání, které má logaritmickou složitost. A dále operace vyhledávání znamená procházení uzlů od kořene k listu. Ověříme, že výška stromu logaritmicky závisí na počtu prvků v něm. Vezmeme strom výšky *h* s kapacitou uzlu *r*. Minimální počet prvků v jednotlivých úrovních ukazuje následující tabulka $\left(p = \frac{r}{2}\right)$.

Úroveň	**Počet uzlů**	**Počet prvků**
0	1 (kořen)	1
1	2	$2p$
2	$2(p+1)$	$2p(p+1)$
3	$2(p+1)^2$	$2p(p+1)^2$
4	$2(p+1)^3$	$2p(p+1)^3$
….	…..	…..
h	$2(p+1)^{h-1}$	$2p(p+1)^{h-1}$

$$n = 1 + 2p\left(1 + (p+1) + (p+1)^2 + \ldots + (p+1)^{h-1}\right) = 1 + 2p\left(\frac{(p+1)^h - 1}{(p+1) - 1}\right)$$

$$n = 1 + 2\left((p+1)^h - 1\right) = 2(p+1)^h - 1$$

$$n + 1 = 2(p+1)^h$$

$$h = \log_{p+1}\left(\frac{n+1}{2}\right) \Rightarrow h = \Theta(\log(n))$$

Celkově tedy vyhledávání má logaritmickou složitost.

Základem zbývajících operací (přidání prvku, odebrání prvku) je vyhledávání a dále průchod stromem směrem nahoru. Složitost průchodu směrem nahoru je závislá na výšce stromu, která má logaritmickou složitost. Z čehož plyne, že i tyto operace mají logaritmickou složitost.