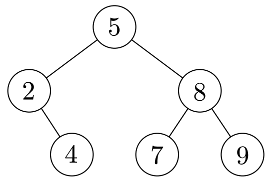
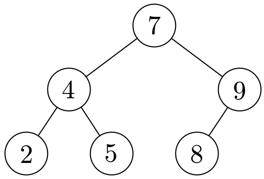
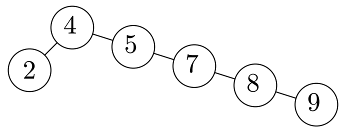
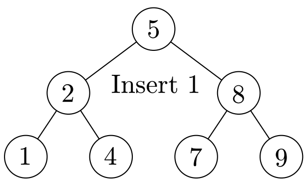
7. Vyhledávací stromy *zapsali M. Řezáč, Š. Masojídek, B. Urbancová*

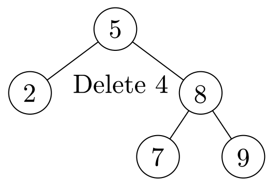
Pár obrázků, které by stály za přesun do předchozí kapitoly:

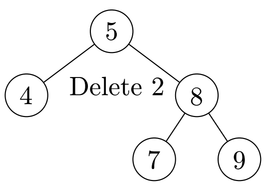


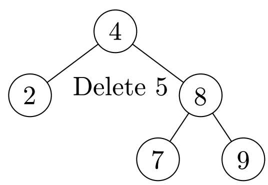












Vyvážené binární vyhledávací stromy

V minulé kapitole jsme se zabývali problematikou přidávání a ubírání prvků binárního vyhledávacího stromu a jeho složitostí a zjistili, že vše záleží na hloubce stromu. Víme, že chceme hloubku logaritmickou, ale jak ji můžeme udržet při operacích? Řešením je následující definice:

Definice: *Dokonale vyvážení* je takové vyvážení, ve kterém pro všechny vrcholy v platí $||L(v)| - |P(v)|| \leq 1$.

Toto nám jistě zajišťuje logaritmickou hloubku, ale je velmi pracné na udržování. Zkusíme proto slabší podmínku:

Definice: *Hloubkové vyvážení* je takové vyvážení, ve kterém pro všechny vrcholy v platí $|h(L(v)) - h(P(v))| \leq 1$. Stromům s hloubkovým vyvážením se říká *AVL stromy* a platí o nich následující lemma.

Lemma: AVL strom o n vrcholech má hloubku $\mathcal{O}(\log n)$.

Důkaz: Uvažme a_k = minimální počet vrcholů AVL stromu o hloubce k. Lehce spočteme:

$$\begin{aligned} a_0 &= 0 \\ a_1 &= 1 \\ a_2 &= 2 \\ &\vdots \\ a_k &= 1 + a_{k-1} + a_{k-2}. \end{aligned}$$

Rekurentní vzorec jsme dostali rekurzivním stavěním stromu hloubky k: nový kořen a 2 podstromy o hloubce $k-1$ a $k-2$.

Indukcí dokážeme, že $a_k \geq 2^{\frac{k}{2}}$. První indukční krok jsme si už ukázali, teď pro $k \geq 2$ platí: $a_k = 1 + a_{k-1} + a_{k-2} > 2^{\frac{k-1}{2}} + 2^{\frac{k-2}{2}} = 2^{\frac{k}{2}} \cdot (2^{-\frac{1}{2}} + 2^{-1}) \cong 2^{\frac{k}{2}} \cdot 1.21 > 2^{\frac{k}{2}}$

Tímto jsme dokázali, že na každé hladině je minimálně exponenciálně vrcholů, což nám zaručuje hloubku $\mathcal{O}(\log n)$ ♡

Operace s AVL stromy

Find se neliší od operace find v binárních stromech.

Důraz klademe na operace *Insert* a *Delete*, protože při nich musíme ošetřit udržení struktury AVL stromů.

První nutnou podmínkou je, že si musíme *pamatovat stav* v každém vrcholu tohoto stromu. A to *vyvážení* hloubky jeho podstromů.

Umluvíme se např. na tomto označení:

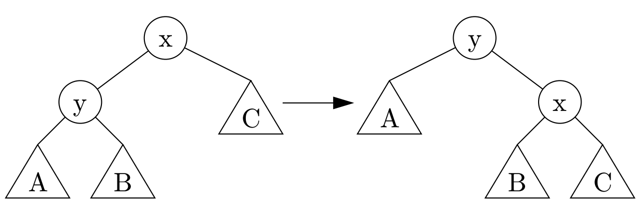
Dostaneme tři typy vrcholů, které se mohou v AVL stromu vyskytnout:

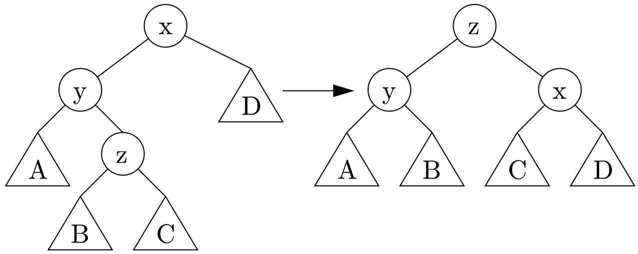
- *Vrchol typu* $\oplus$, pokud je pravý podstrom hlubší
- *Vrchol typu* $\ominus$, pokud je levý podstrom hlubší a
- *Vrchol typu* $\odot$ *(nulou)*, který má oba syny schodné hloubky.

Sestavení AVL stromu:

Postupujeme po struktuře binárního stromu od listů ke kořeni a kontrolujeme, zda jsou vrcholy v jednom ze tří uvedených stavů. Pokud ne, opravíme ho operací jménem rotace.

Rotace





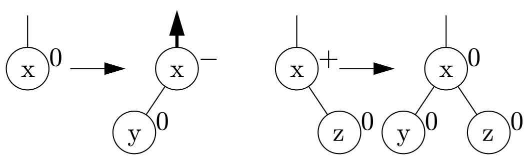
Jde o převrácení hrany mezi původním otcem (kořenem podstromu) a nevyváženým vrcholem tak, aby byli i po přeskupení synové vzhledem k otcům správně uspořádáni.

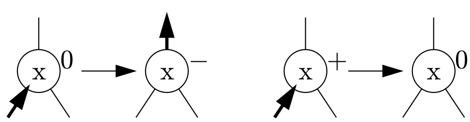
Insert - vložení vrcholu do AVL stromu.

Vložíme jej jako list. Nový list má vždy „znaménko“ nula $\odot$. Předpokládáme, že patří nalevo od posledního otce. Podíváme se na znaménko jeho otce:

- *měl* $\odot$ *(neměl syna)* $\rightarrow$ *teď má* $\ominus$, po struktuře stromu nahoru posíláme informaci, že se podstrom prohloubil o 1, což může mít samozřejmě vliv na znaménka vrcholů na cestě ke kořeni.
- *měl* $\oplus$ *(měl pravého syna, který je listem)* $\rightarrow$ *teď má* $\odot$, hloubka podstromu se nemění
- *měl* $\ominus$ – nenastane, protože v binární struktuře nemohou být dva leví synové

Připadne-li přidaný list napravo, řešíme zrcadlově.

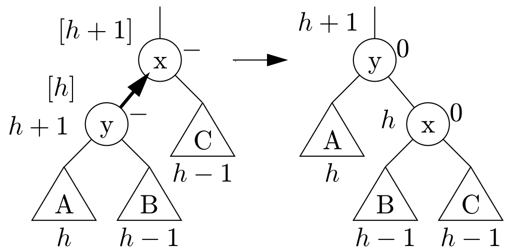




Prohloubil-li se strom vložením nového listu, musíme pracovat s vyvážením:

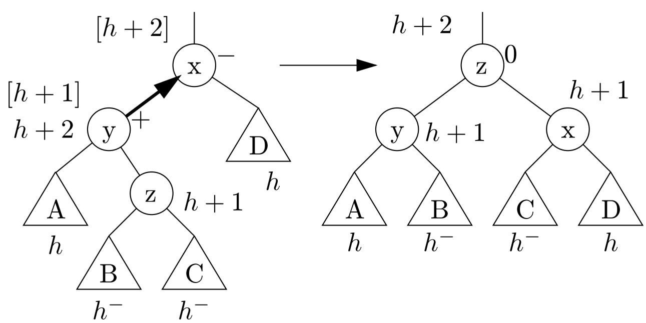
- Informace o prohloubení přišla zleva *do vrcholu typu* $\odot$ $\rightarrow$ mění jej na vrchol se znaménkem $\ominus$ a informace o prohloubení je třeba poslat o úroveň výš.
- Informace o prohloubení přišla zleva *do vrcholu typu* $\oplus$ $\rightarrow$ mění jej na vrchol se znaménkem $\odot$, hloubka je vyrovnána, dál nic neposíláme.
- Informace o prohloubení přišla zleva *do vrcholu s* $\ominus$ $\rightarrow$
  rozebereme na tři případy podle znaménka vrcholu, ze kterého přišla informace o prohloubení:

- Informace přišla *z vrcholu typu* $\ominus$ → provedeme rotaci doprava tak, že novým kořenem se stane vrchol y, ze kterého přišla informace o prohloubení.



Pozorování 1: znaménko vrcholů y a x je $\odot$
Pozorování 2: hloubka před vkládáním byla $h+1$ a nyní je také $h+1$, tedy nemusíme dále posílat informaci o prohloubení a můžeme skončit

- Informace přišla *z vrcholu typu* $\oplus$
    - uvažme ještě vrchol z jako pravého syna vrcholu y, ze kterého prišla informace o prohloubení, a jeho podstromy B a C
    - vrcholy B a C mají hloubku h nebo $h-1$ → označme ji tedy $h-$ (to zřejmě protože vrchol y má znaménko $\oplus$, tedy jeho pravý podstrom s kořenem z má hloubku $h+1$)
    - provedeme dvojrotaci tak, že novým kořenem se stane vrchol z



Pozorování 1: znaménko vrcholu z bude $\odot$
Pozorování 2: znaménka vrcholu x a y se dopočítají v závislosti na hloubce B a C
Pozorování 3: rozdíl hloubky pravého a levého podstromu bude u těchto vrcholů 0 nebo 1

Pozorování 4: hloubka před vkládáním byla $h + 2$ a nyní je také $h+2$, tedy nemusíme dále posílat informaci o prohloubení a můžeme skončit

- informace přišla *z vrcholu typu* $\odot$ – to nemůže nastat, protože v tom případě by nešlo o prohloubení

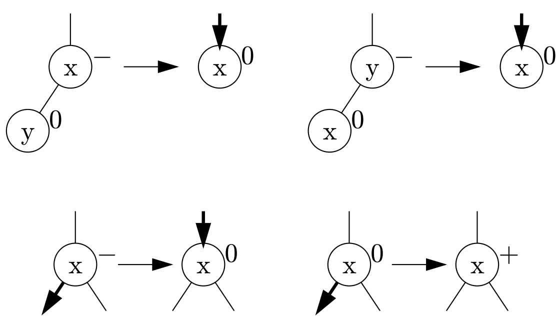
Delete – odebrání vrcholu z AVL stromu Buď mažeme list nebo mažeme vrchol, který měl nějaké syny.

- pokud mažeme list, podíváme se na typ otce. Předpokládáme mazání levého syna.
  - byl typu $\ominus$ (neměl pravého syna) $\rightarrow$ změní se na $\odot$ (vrchol teď nemá žádné syny)
  - byl typu $\odot$ (měl oba syny) $\rightarrow$ změní se na $\oplus$

  (mažeme-li pravý list, řešíme zrcadlově)
- mažeme vrchol s jedním (levým nebo pravým) synem $\rightarrow$ syn nastupuje na místo otce a získává typ $\odot$
  V obou případech posíláme informaci o změně hloubky stromu…
- mazaný vrchol měl oba syny (listy) $\rightarrow$ vybereme jednoho ze synů na místo smazaného otce. Hloubka se nemění.
- mazaný vrchol měl syny podstromy $\rightarrow$ na jeho místo vezmeme největší prvek levého podstromu (nebo nejmenší prvek pravého podstromu) a od odebraného (nahrazujícího) listu kontrolujeme vyvážení podstromu.

Úprava vyvážení stromu po odebrání listu z podstromu

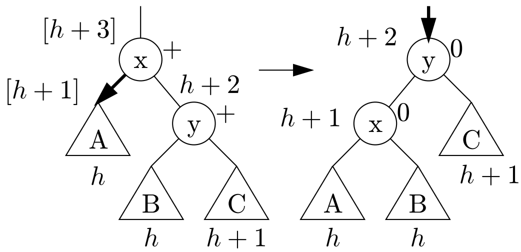
- informace o změně hloubky přišla z levého podstromu do vrcholu typu $\odot$ $\rightarrow$ vrchol se změní na $\oplus$ a dál se hloubka nemění
- informace přišla zleva do vrcholu s $\ominus$ $\rightarrow$ mění se na $\odot$ a posíláme informaci o změně hloubky.



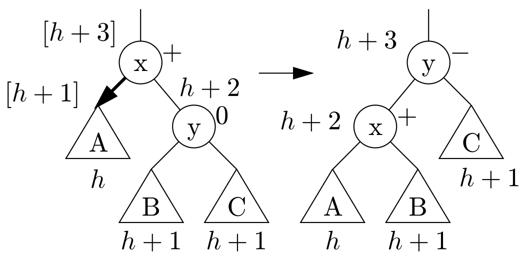
- problémová situace nastává, když informace o změně přišla zleva do vrcholu se znaménkem $\oplus$

Rozebereme na tři případy podle znaménka pravého syna nevyváženého vrcholu

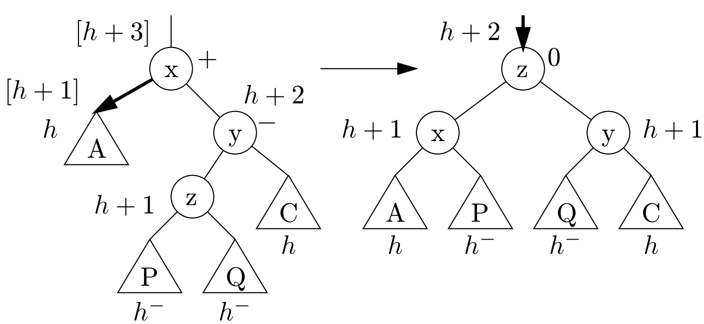
- *pravý syn je typu* $\oplus \rightarrow$ provedeme rotaci vlevo, novým kořenem se stává y (pravý syn), oba vrcholy změní typ na $\odot$ a posíláme informaci o změně hloubky.



- *pravý syn je typu* $\odot \rightarrow$ provedeme opět rotaci vlevo, kořenem se stává y, následně se u y změní typ na $\ominus$, u vrcholu x se typ nemění. Hloubka stromu se nemění, tudíž není třeba posílat informaci.



- *pravý syn je typu* $\ominus \rightarrow$ v tomto případě uvažujeme ještě vrchol z jako levého syna vrcholu y, s podstromy B a C, podstromy B a C mají hloubku h nebo $h-1$. Provedeme dvojrotaci, napřed vpravo rotujeme vrcholy z a y, potom vlevo vrcholy x a z tak, že se z stane novým kořenem, typ vecholu x bude potom $\ominus$ nebo $\odot$, typ y $\oplus$ nebo $\odot$ (podle toho, jaké znaménko měl původně vrchol z), typ z bude $\odot$ a opět posíláme informaci o změně hloubky stromu.



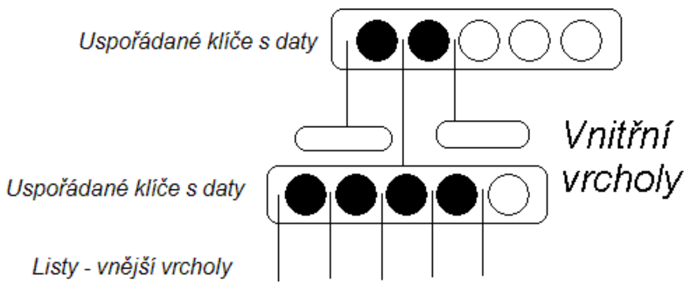
Obecné vyhledávací stromy

Při uložení dat na disku se snažíme, aby se čtení z disku provádělo pokud možno co nejméněkrát a nezáleží nám tolik na tom, kolik operací se vykoná v jednom uzlu. (Časově je operace porovnávání zanedbatelná oproti čtení z disku.)

Definice: (a, b)-*strom* pro parametry a, b, $a \geq 2$, $b \geq 2a - 1$ je zakořeněný strom s uspořádanými syny a vnějšími vrcholy, pro který platí následující axiomy:

1) Data jsou uložena ve vnitřních vrcholech a každý vrchol obsahuje o 1 méně klíčů než má synů.
2) Platí stromové uspořádání, tedy že $A < x_1 < B < x_2 < C < x_3 < D$.
3) Kořen má 2 až b synů, ostatní vnitřní vrcholy a až b synů.
4) Všechny vnější vrcholy jsou ve stejné hloubce (vnější vrchol=list).

Poznámka: kdekoli by mohl být syn a není, připojíme vrchol, kterému říkáme vnější vrchol)



Lemma: (a, b)-strom na n vrcholech má hloubku $O(\log_a n)$.

Důkaz: Zjistíme jeho minimální počet listů (označme jej m): každý vrchol až na kořen má alespoň a synů, hloubku si označíme $d \rightarrow$

$$m \geq \ a^{(d-1)}$$

$$\log_a m \geq d - 1$$

$$d \leq 1 + \log_a m$$

což je řádově $O(\log_a n)$, kde n je počet vrcholů.

Operace s (a,b)-stromy:

Find

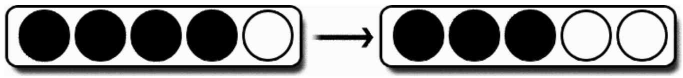
- Vždy zjistíme, mezi které 2 klíče hledaný vrchol patří a potom se zanoříme hlouběji.

Časová složitost nalezení prvku v (a, b)-stromu je $O(\log b \cdot \log_a n)$, kde $\log b$ je čas strávený na jednom vrcholu pro zjištění, mezi které 2 vrcholy hledaný patří, $\log_a n$ je hloubka stromu.

Insert

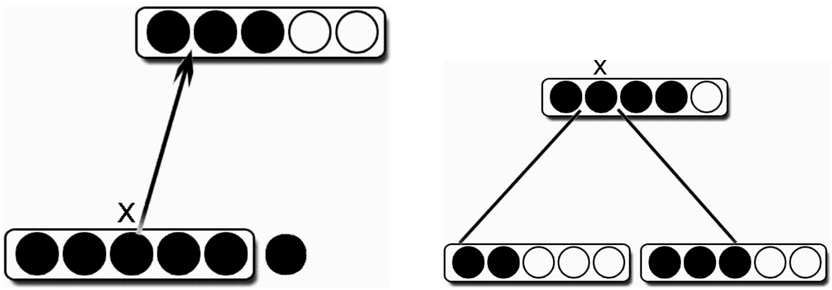
Jako Find, přičemž jestliže nenašel, skončí na posledním patře a přidáme klíč

- pokud přidáním nepřesáhneme maximální počet klíčů můžeme skončit



- pokud přidáním přesáhneme maximální počet klíčů

  1. rozdělíme vrchol na 3 části: L,x,P
  2. L a P jsou nové vrcholy
  3. x je hodnota mezi L a P, kterou vložíme o patro výš jako klíč oddělující nově vzniklé vrcholy L a P
  4. tím jsme převedli problém o patro výš a opakujeme algoritmus



Poznámka: Jestliže se dostaneme až do kořene, rozdělí se kořen na dvě části, vznikne nám nový kořen se dvěma syny (což je povoleno) a celému stromu vzroste hloubka o jedna.

Korektnost: Potřebujeme, aby

$$|L| \geq a - 1$$

$$|P| \geq a - 1$$

po sečtení obou nerovností a pričtení 1 na obě strany rovnice:

$$|L| + |P| + 1 \geq 2a - 2 + 1 = 2a - 1$$

pravá strana je rovna b a to podle definice $\geq 2a - 1$.

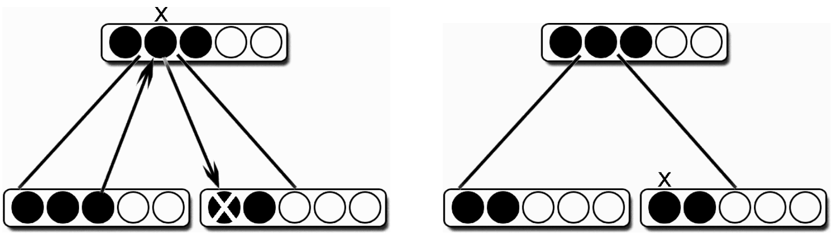
Časová složitost: vkládání prvku do (a, b)-stromu je $O(b \cdot \log_a n)$.

Delete

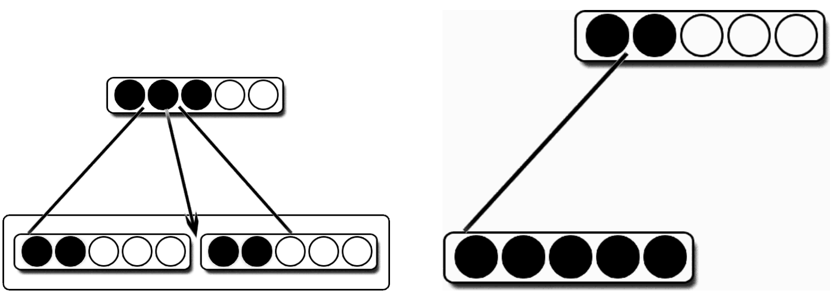
- převedeme na delete z listu (stejný postup jako u stromu: jestliže to není list, prohodíme tuto hodnotu s nejmenší hodnotou podstromu jeho pravého syna) – v tomto případě na klíč posledního vnitřního vrcholu, protože listy jsou vnější vrcholy bez dat.

- pokud má vrchol, ze kterého odebíráme stále $a - 1$ klíčů, můžeme skončit

- pokud má vrchol(V), ze kterého odebíráme $a-2$ klíčů a jeho levý sousední vrchol(L) alespoň a klíčů (klíč otce oddělující tyto vrcholy označme x):
  1. smažeme největší klíč levého sousedního vrcholu (L) a nahradíme tím klíč otce obou vrcholů (nahradíme x za tuto hodnotu)
  2. původní klíč otce(x) přidáme jako nejmenší klíč odebíranému vrcholu (V)
  3. tím mají oba tyto vrcholy $a-1$ klíčů a můžeme skončit



- pokud má vrchol, z kterého odebíráme(V) $a-2$ klíčů a jeho levý sousední vrchol(L) $a-1$ klíčů (klíč otce oddělující tyto vrcholy označme x):
  1. sloučíme V,x,L do jednoho vrcholu
  2. tím jsme problém převedli o patro výš a opakujeme algoritmus



Poznámka: Dojdeme-li takto až do kořene, na místo klíče odebraného z kořene lze použít nejmenší nebo největší klíč nově sloučeného podstromu. Ten odebrat lze, protože po sloučení (které bylo příčinou této situace), je v nejnižším vrcholu $2a-2$ klíčů.

Časová složitost:

$$O(b \cdot \log_a n)$$