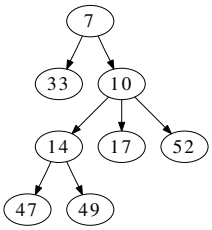
Haldy

Minimová halda je strom, v ktorom platí: prvok uložený v hociktorom vrchole je menší alebo rovný ako všetky prvky uložené v synoch daného vrcholu. Túto vlastnosť budeme volať *podmienka haldy*

(Analogicky existuje aj *maximová halda*, kde uvažujeme opačnú podmienku – prvok uložený v hociktorom vrchole musí byť väčší alebo rovný ako všetky prvky uložené v jeho synoch.)

Príklad haldy je na obrázku **??**.



Obr. 1.1: Halda. Každý vrchol obsahuje hodnotu menšiu alebo rovnú ako každý jeho syn.

Všimnite si, že z podmienky haldy vyplýva, že pre ľubovoľný vrchol v haldy platí, že hodnota v ňom je najmenšia z hodnôt v celom podstrome s koreňom v. (Totiž hodnota vo v je menšia alebo rovná ako hodnoty v jeho synoch, tie sú zase menšie alebo rovné ako hodnoty v ich synoch, a tak ďalej.)

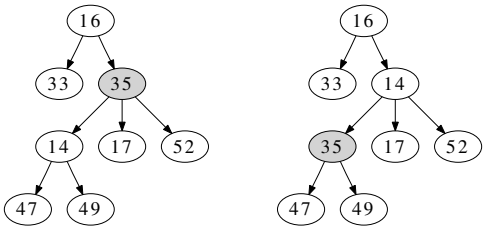
V koreni haldy je teda nutne vždy uložený najmenší zo všetkých prvkov v halde.

Cvičenie 1.0.1 *Majme haldu, o ktorej vieme len to, že obsahuje N navzájom rôznych prvkov, pričom $N \geq 3$. Kde všade sa v tejto halde môže nachádzať tretí najmenší prvok? Kde najväčší? A kde druhý najväčší?*

Základná operácia

Pri výrobe a udržiavaní haldy si vystačíme s jediným pravidlom, ktoré sa zvykne nazývať *bublanie dodola*. Princíp je jednoduchý:

Majme vrchol v, ktorý nespĺňa podmienku haldy. To znamená, že v aspoň jednom z jeho synov je menší prvok ako vo v. Nech u je ten syn, v ktorom je najmenší prvok zo všetkých synov vrcholu v. Vymeňme hodnoty v u a v.



Obr. 1.2: Príklad operácie bublania dodola. V strome naľavo zvýraznený vrchol nespĺňa podmienku haldy. Použitím bublania dodola dostávame strom napravo.

Čo sme tým dosiahli? Zjavne po tejto výmene už platí podmienka haldy pre vrchol v. Avšak mohla sa porušiť vo vrchole u (kde sa hodnota zväčšila) aj v otcovi vrcholu v (keďže vo v sa hodnota zmenšila).

Vidíme teda, že toto pravidlo budeme musieť používať nejako systematicky, aby sme si jeho používaním nenarobili viac problémov ako odstránime.

Cvičenie 1.0.2 *Majme ľubovoľný strom obsahujúci prvky, ktoré vieme porovnávať. Budeme dookola opakovať proces: "Nájdi vrchol, kde neplatí podmienka haldy. Ak už taký neexistuje, skonči. Ak si nejaký našiel, použi naň pravidlo bublania dodola."*
Musí byť tento postup nutne konečný?

Vkladanie do haldy

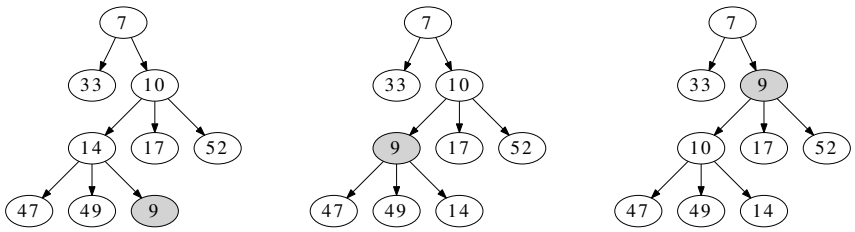
Keď chceme vložiť nový prvok do haldy, pridáme ho ako list pod niektorý prvok haldy. Označme nový vrchol v a jeho otca u. Vieme, že doteraz každý vrchol spĺňal podmienku haldy. Keď sme pridali v, jediné miesto, kde sme ju mohli pokaziť, je vrchol u.

Skúsime teda použiť vo vrchole u pravidlo bublania dodola. Sú dve možnosti. Ak sa nič nestane, máme korektnú haldu a môžeme skončiť.

V opačnom prípade sa vymenia hodnoty v u a v (keďže jedine hodnota vo v mohla byť menšia od hodnoty v u). V tomto okamihu je opäť nanajvýš jeden vrchol, v ktorom neplatí podmienka haldy – a to otec vrcholu u.

Pre tento vrchol zopakujeme celú úvahu. Takto pokračujeme v najhoršom prípade až dovtedy, kým sa nedostaneme do koreňa. Ten už otca nemá, takže v tomto okamihu už nemá kde byť problém – máme korektnú haldu a v nej o prvok viac.

(Tomuto postupu sa niekedy hovorí *bublanie dohora*. Dá sa naň totiž dívať aj tak, že novovložený prvok sa v halde presúva nahor, až kým nenájde miesto, kam podľa svojej hodnoty patrí. Všimnite si tiež, že krok bublania dohora vieme implementovať v konštantnom čase, bez ohľadu na stupeň vrchola u – vieme totiž, že aktuálny vrchol v je jediným synom u, ktorý od neho môže byť menší, preto sa nepotrebujeme pozerať na ostatných synov u.)



Obr. 1.3: Vloženie prvku 9 do haldy a jeho následné bublanie dohora.

Vyberanie najmenšieho prvku z haldy

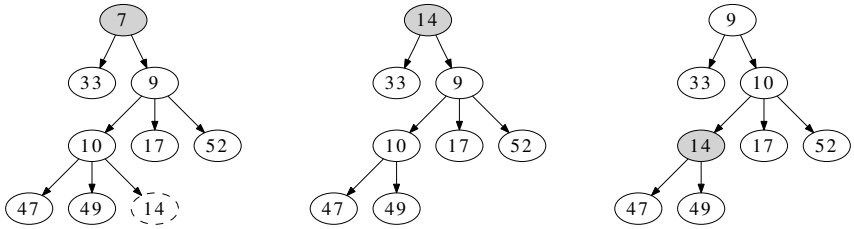
Vieme už, že najmenší prvok sa nachádza v koreni. Keď ho však chceme z haldy vybrať, potrebujeme nový koreň. Toto sa dá riešiť napríklad tak, že zoberieme ľubovoľný list, odstránime ho a hodnotu z neho dáme do koreňa namiesto práve vybranej hodnoty.

Čo sme takto dostali? Strom, v ktorom všetky vrcholy okrem koreňa spĺňajú podmienku haldy. A už je jasné, ako ďalej – použijeme na koreň pravidlo "bublanie dodola". Ak toto pravidlo nič nezmení, máme hotovú haldu.

Ak nastane výmena, pokračujeme použitím pravidla "bublania dodola" na vrchol, kam sme práve presunuli hodnotu z koreňa. Toto opakujeme až do okamihu, kedy už výmena nenastane.

Po každej iterácii platí, že jediný vrchol, ktorý môže porušovať podmienku haldy, je vrchol s prvkom, ktorým bubleme dodola. Na začiatku (keď je náš prvok v koreni) aj po prvej iterácii je to zjavné.

(Všimnite si, že na tento proces sa dá dívať ako na opak procesu, ktorý prebieha pri pridávaní prvku. V tomto prípade prvok, ktorý sme umiestnili do koreňa, klesá nadol, kým nenájde miesto, kam podľa svojej hodnoty patrí.)



Obr. 1.4: Výber najmenšieho prvku 7 z haldy. Na jeho miesto dáme prvok 14 z listu. Tento prvok následne prebuble dodola, čím opäť dostávame haldu.

Aké haldy sú nanič?

Haldy rôznych degenerovaných tvarov nám príliš nepomôžu. Napr. je nanič halda, ktorá je tvorená jedným koreňom a $N-1$ listami. A takisto je nanič halda-had, v ktorej má každý vrchol najviac jedného syna.

V čom je problém?

Pozrime sa najskôr na prvú spomínanú haldu. Keby sme v našom programe používali haldu, ktorá vyzerá takto, príliš by nám to nepomohlo. Zjavne do takejto haldy vieme ľahko vkladať ďalšie prvky – stačí pridať nový list a ak treba, tak hodnotu v ňom vymeniť s hodnotou v koreni. Avšak do problémov sa dostávame v okamihu, keď by sme chceli najmenší prvok z haldy vybrať. Vtedy by sme totiž na to, aby sme zistili, ktorú hodnotu presunúť do koreňa, museli prezrieť úplne všetky prvky v halde.

Predstavme si, že sme do takejto haldy najskôr vložili N prvkov a potom N-krát vybrali najmenší z nich. Každé vloženie prebehlo v konštantnom čase. Potom sme však museli po výbere prvého prvku nájsť najmenší spomedzi zvyšných $N-1$, po výbere druhého prvku najmenší zo zvyšných $N-2$, a tak ďalej. Keď toto všetko sčítame, zistíme, že dokopy vykonáme $\Theta(N^2)$ krokov.

A o nič lepšie na tom nie je ani druhá spomínaná halda.

Pozrime sa na to, ako by tá istá postupnosť operácií (najskôr vložiť N prvkov a potom N-krát vybrať najmenší z nich) prebiehala tu. Vždy, keď vložíme nový prvok, ho pridáme ako nový list do ešte hlbšej úrovne, a následne s nim bubleme dohora, až kým sa nezastaví. V najhoršom možnom prípade (ktorý nastane, ak budeme vkladať prvky v poradí od najväčšieho po najmenší) teda každý prvok postupne prebuble hore celou reťazou až do koreňa. Teda druhý vložený prvok bude bublať dohora 1-krát, tretí 2-krát, ..., a posledný N-tý bude bublať až $(N-1)$-krát. V tomto prípade si teda už samotná postupnosť vložení N prvkov môže vyžiadať až $\Theta(N^2)$ krokov.

K týmto dvom patologickým príkladom pridáme ešte poznámku: Prvý z nich zodpovedá situácii, kedy by sme si prvky pamätali v obyčajnom poli a navyše mali premennú, v ktorej máme index najmenšieho z nich. Druhý z nich zodpovedá tomu, že si prvky pamätáme v utriedenom poli, resp. v utriedenom spájanom zozname.

Ako teda musí vyzerať dobrá halda?

V predchádzajúcej časti sme si ukázali dva patologické prípady, v ktorých nám princíp haldy nijako nepomohol. Všimnime si, čo presne v nich spôsobovalo, že práca s nimi bola pomalá.

V prvom z nich to bol veľký stupeň koreňa. Ak máme v halde vrcholy s veľkým stupňom, tieto nás budú spomaľovať – lebo vždy, keď pre tento vrchol vykonáme bublanie dodola, musíme prezrieť všetkých jeho synov.

V druhom to zase bola veľká hĺbka haldy. Keď pridáme nový prvok a bubleme s ním dohora, môže sa nám stať, že prebuble až do koreňa. A v takomto prípade je počet krokov priamo úmerný hĺbke, v ktorej prvok začínal.

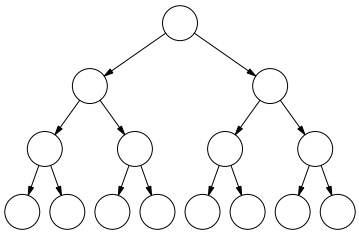
Dobrá halda by teda mala mať vrcholy malého stupňa a zároveň by nemala byť príliš hlboká.

Takmer úplné binárne stromy

Pripomeňme si, že v binárnom strome môže každý vrchol mať nanajvýš dvoch synov – jedného ľavého a jedného pravého. A tiež si pripomeňme, že hĺbka vrcholu je definovaná ako počet hrán na ceste z neho do koreňa.

Binárny strom môžeme rozdeliť na *úrovne*: úroveň k tvoria všetky vrcholy, ktoré majú hĺbku k. Všimnime si, že na úrovni k môžeme mať nanajvýš 2^k vrcholov. (Na úrovni 0 je len koreň, ten má nanajvýš 2 synov ktorí tvoria úroveň 1, každý z nich má nanajvýš 2 synov ktorí spolu tvoria úroveň 2, atď.)

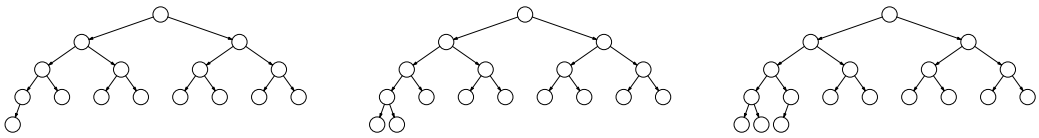
Binárny strom voláme *úplný*, ak je každá jeho úroveň buď prázdna alebo úplne plná – inými slovami, ak má každý vrchol v každej úrovni okrem poslednej práve dvoch synov. Úplný binárny strom hĺbky k má teda presne $1 + 2^1 + \cdots + 2^{k-1} + 2^k = 2^{k+1} - 1$ vrcholov, a z nich 2^k sú listy.



Obr. 1.5: Úplný binárny strom hĺbky 3. Má $2^4 - 1 = 15$ vrcholov, $2^3 = 8$ z nich sú listy.

Zjavne pre skoro žiadne N neexistuje úplný binárny strom s N vrcholmi. Budeme sa teda musieť uspokojiť so stromami, ktoré sa na úplné dostatočne podobajú.

Takmer úplný binárny strom je taký binárny strom, ktorý má všetky úrovne okrem poslednej plné, a v poslednej sú všetky vrcholy natoľko vľavo, nakoľko sa to pri ich počte dá.



Obr. 1.6: Takmer úplné binárne stromy so 16, 17 a 18 vrcholmi.

Zjavne pre každé N existuje práve jeden takmer úplný binárny strom s N vrcholmi. Ich konštrukciu môžeme popísať aj takto: Takmer úplný strom s $N > 1$ vrcholmi zostrojíme z takmer úplného stromu s $N - 1$ vrcholmi tak, že nájdeme najmenšiu úroveň, do ktorej sa dá pridať vrchol, a pridáme ho do nej na najľavejšie možné miesto.

Tiež si všimnite, že pre každé N z množiny $\{2^k, 2^k + 1, \ldots, 2^{k+1} - 1\}$ má N-vrcholový takmer úplný binárny strom hĺbku k. Toto isté môžeme povedať aj nasledovne: Takmer úplný binárny strom s N vrcholmi má hĺbku $\lfloor \log_2 k \rfloor$.

Binárna halda

Najčastejšie používaným typom haldy je *binárna halda*. Tá bude spĺňať obe požiadavky, ktoré sme sformulovali v časti **??** – nebude mať veľké stupne vrcholov a taktiež nebude mať veľkú hĺbku. Binárna halda je totiž strom, ktorý má vlastnosť haldy a je takmer úplný.

Obr. 1.7: Binárna halda.

Veľkou výhodou binárnej haldy je, že ju vieme ľahko implementovať bez použitia ukazovateľov. Stačí nám jedno dostatočne veľké pole.[1] Haldu doň uložíme "po riadkoch". Teda na začiatku poľa bude uložený koreň, za ním zľava doprava jeho synovia, za nimi zľava doprava ich synovia, a tak ďalej.

[1] Na inej prednáške sme si ukazovali abstraktnú dátovú štruktúru vektor, ktorú môžeme použiť, ak vopred nevieme potrebnú veľkosť poľa.

index	0	1	2	3	4	5	6	7	8	9
prvok	7	12	10	23	29	11	47	28	27	35

Obr. 1.8: Binárna halda z obrázka **??** uložená v poli.

Všimnite si, že N-prvková binárna halda nám zaberá presne prvých N pozícii v poli.

Ak majú prvky v poli indexy 0 až $N-1$, tak platia nasledovné vlastnosti: Otec vrcholu s indexom x má index $\lfloor (x-1)/2 \rfloor$. Synovia tohoto vrcholu majú indexy $2x+1$ a $2x+2$. Ku každému vrcholu teda vieme jeho otca aj prípadných synov nájsť v konštantnom čase. Vďaka tomuto vieme priamo s poľom pracovať tak, ako by sme pracovali s haldou reprezentovanou ako strom.

Výber minima z binárnej haldy

Pozrime sa podrobnejšie na výber najmenšieho prvku z binárnej haldy. Ako u každej haldy, tento nájdeme v koreni – čiže v našom poli, v ktorom máme uloženú haldu, na pozícii 0.

Po tom, ako zistíme jeho hodnotu, by sme ho potrebovali odstrániť. Pripomeňme si, že vo všeobecnej halde sa to robilo tak, že sme namiesto neho odstránili ľubovoľný list a hodnotu z neho sme presunuli do koreňa.

V binárnej halde si môžeme dovoliť odstrániť vždy práve jeden list – najpravejší list na najspodnejšej úrovni. Teda ten, ktorý je v našom poli uložený na pozícii $N-1$.

Keď hodnotu z neho presunieme do koreňa, dostávame už takmer hotovú binárnu haldu s $N-1$ vrcholmi. Jediný problém, ktorý môže nastať, je, že hodnota, ktorú sme práve presunuli do koreňa, tam nemusí spĺňať podmienku haldy. To samozrejme ľahko napravíme prebublaním tejto hodnoty dodola.

Efektívnosť binárnej haldy

Keďže v binárnej halde má každý vrchol najviac dvoch synov, vieme pravidlo bublania dodola použiť na ľubovoľný vrchol v konštantnom čase.

Pri vkladaní prvku nový prvok pridáme do najspodnejšej vrstvy a následne ním prebubleme dohora. Čas potrebný na vloženie prvku je teda v najhoršom prípade priamo úmerný hĺbke haldy.

Pri výbere prvku najskôr v konštantnom čase odstránime posledný list a presunieme jeho hodnotu do koreňa, a potom touto hodnotou prebubleme niekam dodola. Teda aj u tejto operácie je časová zložitosť v najhoršom prípade priamo úmerná hĺbke haldy.

A tu sa ukazuje, prečo bolo dobré zvoliť si práve binárnu haldu. Binárna halda s N prvkami má tvar takmer úplného binárneho stromu, a teda má hĺbku rovnú $\lfloor \log_2 N \rfloor$.

Preto majú obe operácie "vloženie prvku do binárnej haldy" a "výber najmenšieho prvku z nej" časovú zložitosť $O(\log N)$.

Tvorba binárnej haldy z neutriedeného poľa

Vyrobiť haldu obsahujúcu daných N prvkov vieme aj efektívnejšie ako postupným vkladaním prvkov do pôvodne prázdnej haldy. Použitím jednoduchého triku dokážeme binárnu haldu vyrobiť v lineárnom čase.

Majme pole obsahujúce našich N prvkov. Na toto pole sa môžeme dívať ako na takmer úplný binárny strom, v ktorom len potrebujeme preusporiadať prvky tak, aby všade platila podmienka haldy. Ukážeme, že na to, aby sme ju zabezpečili, stačí v správnom poradí bublať jej prvkami dodola.

Presnejšie, budeme prvky spracúvať od konca poľa, t. j. od listov haldy. Pre každý z nich zoberieme hodnotu, ktorá v ňom aktuálne je, a prebubleme ňou dodola, kým to ide.

Prečo to funguje?

Stačí si všimnúť, že vždy v okamihu, kedy spracujeme niektorý vrchol, platí, že celý podstrom daného vrcholu už spĺňa podmienku haldy. (Keď spracúvame nejaký vrchol, znamená to, že už sme predtým spracovali oboch jeho synov, a teda ich podstromy už oba spĺňajú podmienku haldy. Máme teda starú známu situáciu, kedy

jedine hodnota v koreni aktuálneho podstromu kazí podmienku haldy v ňom, no a bublaním tejto hodnoty dodola tento kaz odstránime.)

A zaujímavejšia otázka: Ako rýchlo to funguje?

Celkovú časovú zložitosť vieme odhadnúť ako počet spracovaných vrcholov plus celkový počet krokov bublania dodola. Počet spracovaných vrcholov je N. Celkový počet krokov bublania dodola odhadneme nasledujúcou úvahou:

Predstavme si, že sme nechali prebehnúť náš algoritmus. Všimnime si teraz ľubovoľnú konkrétnu úroveň x v našej halde a položme si otázku: Koľkokrát sa stalo, že na túto úroveň klesol niektorý prvok?

Zjavne keď pre každú úroveň spočítame počet prvkov, ktoré na ňu niekedy klesli, dostaneme presne celkový počet krokov bublania dodola.

No a na druhej strane na úroveň x mohli klesnúť len prvky, ktoré pôvodne boli na vyšších úrovniach. A tých bolo presne $2^x - 1$.

Preto celkový počet krokov bublania dodola v halde s hĺbkou k je nanajvýš rovný

$$(2^k - 1) + (2^{k-1} - 1) + \cdots + (2^1 - 1) = 2^{k+1} - k - 2$$

No a pre počet vrcholov N haldy s hĺbkou k platí $N \geq 2^k$. A teda celkový počet krokov bublania dodola je menší ako $2N$.

*Zložitejšie haldy

Existujú aj zložitejšie haldovité dátové štruktúry, ktoré vedia efektívne robiť všetky operácie, ktoré vie "klasická" binárna halda, a niečo naviac. Prvým príkladom je *binomická halda*. Nejde vlastne o haldu podľa našej definície, ale o množinu obsahujúcu niekoľko háld špeciálneho tvaru, pričom počty vrcholov v nich sú navzájom rôzne mocniny dvoch. Oproti binárnej halde vie binomická halda navyše operáciu merge: vieme v čase $O(\log N)$ spojiť dve binomické haldy do jednej.

Ešte komplikovanejšou dátovou štruktúrou je *Fibonacciho halda*, ktorá vie robiť niektoré operácie (vloženie prvku, spojenie dvoch háld a zvýšenie priority danému prvku) dokonca v amortizovanom konštantnom čase. Implementácia Fibonacciho haldy je komplikovaná. V praxi sa len zriedka stretneme s natoľko veľkými vstupmi, aby sa lepšia asymptotická časová zložitosť Fibonacciho haldy prejavila. Táto dátová štruktúra však má veľký význam v teoretickej oblasti pri návrhu nových efektívnych algoritmov.

Cvičenie 1.0.3 *Binárnu haldu sa dá upraviť tak, aby vedela v logaritmickom čase vracať nie len najmenší, ale aj najväčší prvok. Jedna možnosť, ako to dosiahnuť, je pozmeniť podmienku haldy nasledovne: Každý vrchol, ktorý je na párnej úrovni (vrátane koreňa) musí obsahovať najmenší prvok vo svojom podstrome. Každý vrchol, ktorý je na nepárnej úrovni, musí obsahovať najväčší prvok vo svojom podstrome.*
Popíšte, ako vyzerajú jednotlivé operácie s takouto min-max haldou.