HALDY

Obecně: Datové struktury pro dynamické udržování minima v lineárně uspořádané množině (nebo maxima)
↳ porovnávací model

Rozhraní: Insert
Min, ExtractMin
(Increase), Decrease, Delete ← dostane ukazatel na prvek! (identifikátor vrácený Insertem)
Build, (Merge)

BVS umí vše v $O(\log n)$, Build v $O(n \log n)$ … ale půjde to rychleji

Příklad: Dijkstrův alg.: $O(n) \times$ Insert, $O(n) \times$ ExtractMin, $O(m) \times$ Decrease
⇒ chceme rychlý Decrease, i za cenu zpomalení ostatních operací
↳ binární halda $O((m+n)\log n)$

Příklad: HeapSort … z toho dolní odhad na Build / ExtractMin
$\Omega(n \log n)$ $\Omega(\log n)$

d-regulární halda ($d \geq 2$) – detaily viz Průvodce, kap. 4.2

- Insert bublá nahoru → $O(\log_d n)$ … $O\left(\frac{\log n}{\log d}\right)$
- ExtractMin bublá dolů → $O(d \cdot \log_d n)$ … $O\left(\frac{d}{\log d} \cdot \log n\right)$
- Decrease nahoru, Increase dolů ← je třeba udržovat polohy prvků
- Delete je Increase / Decrease (nebo Decrease + ExtractMin) → $O\left(\frac{d}{\log d} \cdot \log n\right)$

Volba d: S rostoucím d se zrychluje Insert/Decrease a zpomalují ostatní operace
↳ u Dijkstry: $O(n \cdot d \cdot \log_d n + m \cdot \log_d n)$
⇒ optimum pro $d \approx m/n$ … pak $O(m \cdot \log_{m/n} n)$

Build: zespoda zabublávám dolů (jako u binární haldy)
trvá to $O\left(\sum_{i=0}^{h} d^i \cdot d^{h-i}\right) = O\left(dn \cdot \underbrace{\sum_{i=0}^{h} \frac{i}{d^i}}\right) = O(n)$. ← konstanta v O by mohla záviset na d, ale podrobnější výpočet ukáže, že nezávisí
$h = \lceil \log_d n \rceil$
pro každé d konverguje (podílové kritérium)
[pro $d=2$ se sečte na 2, pro $d>2$ na méně]

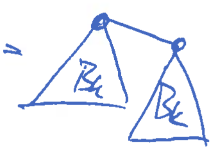
Merge neumíme efektivně

Binomiální halda – viz Průvodce, kap. 18

Df: Binomiální strom B_k řádu k: $B_0 = \bullet$ $B_{k+1} =$

& jakýkoli isomorfní zakořeněný (pěstovaný) strom

Ekvivalentně: $B_0 = \bullet$ $B_{k+1} =$



Příklady: B_0 B_1 B_2 B_3

Lemma: B_k má 2^k vrcholů v $k+1$ hladinách.

Df: Binomiální halda je posloupnost binom. stromů $\mathcal{T} = T_1, T_2, \dots, T_\ell$ t.ž.:

1. ~~řády stromů jsou~~ $r(T_i) < r(T_{i+1})$ ... speciálně: řády jsou různé
   (r = řád stromu)
2. V každém vrcholu $v \in T_i$ je uložen prvek $h(v)$.
3. Platí haldové uspořádání: je-li s synem v, pak $h(v) \leq h(s)$.

Reprezentace: Vrchol stromu si pamatuje:
- prvního syna
- následujícího bratra ← v kořeni používáme pro další strom v haldě
- hodnotu
- řád (používáme jen v kořeni)
- otce (chceme-li Decrease)

Řády stromů odpovídají jedničkám ve dvojkovém zápisu n (počtu prvků)

Lemma: # stromů i jejich řády jsou $O(\log n)$ [dokonce $\lceil \log n \rceil + 1$]

Min ... projitím všech kořenů, pak ukážeme, že ho lze kešovat

Merge (to je pro BH základní operace) ... $O(\log n)$ sléváním posloupností & slučováním stromů stejného řádu → jako sčítání dvojkových čísel

Insert → Merge s 1-prvkovou haldou

Build → $n \times$ Insert, celkem $O(n)$ ... amortizace bin. počítadla
↑ pozor na detaily Merge: dojde-li 1 seznam, musíme 2. připojit vcelku

ExtractMin ... Min, pak odebereme kořen → strom se rozpadne na podstromy různých řádů → mohu udělat Merge ... celkem $O(\log n)$

Decrease ... bublání nahoru, potřebuji udržovat pointery na otce → $O(\log n)$

Delete ... Decrease + ExtractMin → $O(\log n)$

Increase ... bublání dolů by trvalo $O(\log^2 n)$, takže raději Delete + Insert → $O(\log n)$

Min v $O(1)$ ... Merge aktualizuje keš, ExtractMin ji spočítá znovu
($O(1)$; $O(\log n)$)

Líná BH – horší worst case, lepší amortizovaně

Princip: upravíme definici, aby nevyžadovala rozdílné řády

Merge pak pouze spojí seznamy stromů ← potřebujeme obousměrné seznamy stromů/synů
→ $O(1)$ w.c.
↑ cyklické, ale pak umíme ptr na začátek odvodit ptr na konec
→ 3 ptr/vrchol (bez otce)

ExtractMin konsoliduje haldu (přihrádkově roztřídí stromy a pak je páruje)
↑ převede na std. tvar s různými řády
→ $O(n)$ w.c., ale ukážeme, že se zamortizuje.

Pro amort. analýzu zvolme potenciál $\Phi :=$ # stromů ve všech haldách

operace	skut. cena	$\Delta\Phi$	amort. cena
Merge	$O(1)$	0	$O(1)$
Insert	$O(1)$	1	$O(1)$
Build	$O(n)$	n	$O(n)$
ExtractMin	$O(\log n + s)$	$s' - s$ $\left.\le \log n - s\right\}$ za slučování $+ \left.\le \log n\right\}$ nové stromy po odtržení kořene	$O(\log n)$

$s = $ #stromů

Min ... kešujeme $\Rightarrow O(1)$

Decrease, Increase, Delete ... jako u pilné haldy (nemění Φ)

Fibonacciho halda

← správněji by asi měla být fibonacciovská (nevymyslel ji Fibonacci)

Chceme rychlý Decrease ...

Dec(x, α) ← nový klíč

pokud $\alpha > h(p)$, pouze přepíšeme $h(x)$

jinak useknemе podstrom zakořeněný v x

Důsledky: ① přišli jsme o pravidelnou strukturu stromů ... ale to nevadí

② řád stromu může být až n → pomalý bucketsort při konsolidaci!

... omezíme, jak moc smíme stromy "otrhat"

Df: Fibonacciho halda je les pěstovaných stromů t. ž.:

~~① každý vrchol obsahuje hodnotu $h(x)$~~

~~② $h(\text{otec}) \le h(\text{syn})$~~

~~③ vrcholy si pamatují: řád = # synů, ukazatel na otce, 1. syna, L a P bratra (cyklicky)~~

① Ve vrcholech je uloženo:

- hodnota $h(x)$
- řád = #synů
- ukazatele na otce, 1. syna, předchozího/následujícího bratra (cyklicky)
- značka, zda vrchol už přišel o syna (kořen není nikdy označen)

značka(new) ← 0

② platí haldové uspořádání: $h(\text{otec}) \le h(\text{syn})$

Insert, Merge, Min, Build: stejně jako u líné BH

ExtractMin: stejně, jen osamostatněným stromům maže značky v kořeni

- konsolidace pracuje podle řádů

Decrease → Cut(v): $p \leftarrow$ otec(v) ... je-li $p = \emptyset$ (v je kořen), skončíme.

Odpojíme v od p a přidáme v do seznamu stromů v haldě.

značka(v) ← 0

Pokud značka(p) = 0: značka(p) ← 1

else: Cut(p)

Lemma: $1 + \sum_{i=0}^{d} F_i = F_{d+2}$ Dk: indukcí

Invariant: Je-li v vrchol řádu k, pak $|T_v| \ge F_{k+2}$

Důsledek: řády $\in O(\log n)$.

Dk invariantu: Indukcí podle výšky T_v ... pro list v triviálně platí: $F_2 = 1$.

Krok: • Nechť $x_1 - x_k$ jsou synové vrcholu v od nejstaršího k nejmladšímu.

• Když jsme připojovali x_i ...

• $x_1 \_ x_{i-1}$ už byli připojeni $\Rightarrow$ řád(v) byl aspoň $i-1$ ← mohl být více, neb mohli existovat další, později ~~~~ odseknutí synové

• řád(x_i) se rovnal řád$(v) \geq i-1$

lépe značit r_i

• ~~~~ od té doby mohl x_i přijít o 1 syna $\Rightarrow$ teď je řád$(x_i) \geq i-2$

$\Downarrow$ IP

• $|T_{x_i}| \geq F_{i-2+2} = F_i$

• $|T_v| \geq 1 + \sum_{i=1}^{k} F_i = 1 + \sum_{i=0}^{k} F_i \overset{\text{Lemma}}{=\!=} F_{k+2}.$

Amortizace: $\Phi := \#$stromů $+ 2 \cdot \#$označ. vrcholů (ve všech haldách dohromady)

operace	skut. cena	$\Delta\Phi$	amort. cena
Merge	$O(1)$	0	$O(1)$
Insert	$O(1)$	1	$O(1)$
Build	$O(n)$	n	$O(n)$
ExtractMin	$O(\log n + s)$	$+ \leq \log n$ ... nové stromy $-$ ... ... odznačeny jejich kořeny $+ s' - s \leq \log n - s$	$O(\log n)$
Cut	$O(s)$	za 1 strom: -2 (mizí zn.) $+1$ (nový strom) $\}\ -s$ $+2$ nová značka	$O(1)$
Decrease	$O(s)$		$O(1)$

$s = \#$stromů před
$s' = \#$stromů po

~~~~ $s = \#$ odsek. stromů
~~~~