1. cvičení

Datové struktury I, 2. 10. 2024 https://iuuk.mff.cuni.cz/~vesely/vyuka/

Úloha 1 (Haldy a Dijkstra)
Připomeňte si Dijkstrův algoritmus a d-regulární haldy. Jaká je asymptotická složitost Dijkstrova algoritmu s d-regulární haldou? Nalezněte nejvhodnější d pro složitost Dijkstrova algoritmu s d-regulární haldou.

Úloha 2 (Asymptotika)
Roztřiďte následující funkce do skupin stejně rychle rostoucích funkcí (tj. pro všechny f, g v jedné skupině platí $f = \Theta(g)$) a následně porovnejte tyto skupiny pomocí o a ω: n, $42n + 7$, n^2, $\log n$, $\log_e n$, $\log(n^2)$, $(\log n)^2$, $\sqrt{n}$, 2^n, 2^{2n}, 4^n, $2^{n \log n}$, $2^{2 \log n}$, n^n, $n!$, $(n+1)!$.
Všechny logaritmy bez explicitního základu mají dvojkový základ.

Úloha 3 (Následník a jeho iterace)
Najdeme v BVS vrchol s minimálním klíčem, a poté $(n-1)$-krát provedeme operaci nalezení následníka. Jaká bude celková časová složitost?

Úloha 4 (Perfectly balanced, as all things should be)
Navrhněte algoritmus, který ze seřazeného pole v lineárním čase vytvoří dokonale vyvážený BVS. (Tedy pro každý vrchol musí platit, že počet vrcholů v levém podstromu se od počtu vrcholů v pravém podstromu smí lišit maximálně o 1.)

Bonusové úlohy

Úloha 5 (Pozorné čtení)
Najdete v zadání cvičení 2 formální chybku?

Úloha 6 (Intervalový update)
Mějme BVS jako slovník dvojic (klíč, hodnota) s číselnými hodnotami. Upravte jej, aby podporoval operaci $\textsc{Add}(x, y, \delta)$, která k hodnotám všech klíčů v intervalu $[x, y]$ přičte δ.
Tato operace má běžet v $\mathcal{O}(h)$, kde h je hloubka stromu. To znamená, že nemusíme hned aktualizovat hodnoty všech klíčů v intervalu. Stačí, když operace $\textsc{Find}(k)$ vrátí správnou hodnotu klíče k.

Definice (Halda jako obecná datová struktura)**.** Halda (v našem případě minimová) je datová struktura, jež ukládá množinu prvků opatřených klíči a nabízí následující operace:
Insert(x): vloží prvek x do množiny
Min(): najde prvek s nejmenším klíčem
ExtractMin(): odebere prvek s nejmenším klíčem a vrátí ho jako výsledek

Také můžeme přidat operace Increase, Decrease - pro prvek (nám daný ukazatelem) zvýší, resp. sníží, jeho klíč. Je zvykem, že klíč prvku x značíme jako $k(x)$.

Definice (Minimová binární halda)**.** Minimová binární halda je datová struktura tvaru binárního stromu, v jehož každém vrcholu je uložen jeden prvek, a který zároveň splňuje:

1. *Tvar haldy*: Všechny hladiny kromě poslední jsou plně obsazené. Poslední hladina je zaplněna zleva.
2. *Haldové uspořádání*: Je-li v vrchol a s jeho syn, pak $k(v) \leq k(s)$.

Operace se provádějí následovně:
Insert(x): vloží x do haldy na první volné místo v poslední hladině, a poté prvek vybubláme nahoru, aby bylo splněno haldové uspořádání
Min(): vrátí kořen stromu
ExtractMin(): hodnotu v kořeni nahradíme hodnotou nejpravějšího vrcholu v v poslední hladině, vrchol v smažeme, a kořen zabubláme dolů, aby bylo splněno haldové uspořádání
Pomocí bublání můžeme také přidat operace Increase, Decrease.

Algoritmus (Dijkstrův s haldou (náčrt))**.** Vstup: graf $G = (V, E)$ s kladnými délkami hran $\ell : E \to \mathbb{R}^+$ a počáteční vrchol $v_0 \in V$.

1. všechny vrcholy přidejme do haldy s klíči $+\infty$
2. Decrease(v_0, 0) - snížení klíče v_0 na nulu
3. Dokud je halda neprázdná:
   (a) v = ExtractMin()
   (b) Pro všechny sousedy u vrcholu v: pokud $k(u) > k(v) + \ell(v, u)$, pak Decrease(u, $k(v) + \ell(v, u)$)

Věta (Složitost Dijkstry)**.** Dijkstrův algoritmus s haldou běží v čase $\mathcal{O}(n \cdot T_i + n \cdot T_x + m \cdot T_d)$, kde T_i, T_x, T_d jsou po řadě složitosti operací Insert, ExtractMin, Decrease.