1. cvičení

Datové struktury I, 2. 10. 2025 https://iuuk.mff.cuni.cz/~chmel/2526/ds1/

Úloha 1 (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.

Řešení
Od nejmenšího k největšímu

- $\log n, \log_e n, \log(n^2)$
- $(\log n)^2$
- $\sqrt{n}$
- $n, 42n + 7$
- $n^2, 2^{2 \log n}$
- 2^n
- $2^{2n}, 4^n$
- $n!$
- $(n + 1)!$
- $n^n, 2^{n \log n}$

Úloha 2 (Iterace následníka)
Najdeme v BVS (s n vrcholy) vrchol s minimálním klíčem, a poté $(n - 1)$-krát provedeme operaci nalezení následníka. Jaká bude celková časová složitost?

Řešení
Je to DFS, takže lineární.

Úloha 3 (A teď oboustranně a se smrštěním)
Zatím jsme přidávali jenom na konec pole. Šlo by konstrukci upravit tak, abychom mohli přidávat i na začátek a zachovali jsme přitom složitost?
A co kdybychom chtěli prvky z konce (či začátku) mazat?

Řešení
Oboustrannost: možnost je například cyklické pole, pak se to snadno uamortizuje.
Mazání: počkáme, až budeme vyplňovat jen čtvrtinu maximální kapacity, a pak zmenšíme na polovinu.

Úloha 4 (Natahujeme pole)
Ukázali jsme si, jak implementovat natahovací pole, které funguje v amortizovaně konstantním čase, s tím, že vždycky, když je pole plné, a potřebujeme přidat další prvek, všechno překopírujeme do pole dvojnásobné velikosti.
Co by se stalo, kdybychom místo zdvojnásobování kapacitu C zvyšovali jinak?

a) $C \rightsquigarrow C + k$, kde $k \geq 1$ je konstanta,

b) $C \rightsquigarrow C^2$,

c) $C \rightsquigarrow k \cdot C$ pro konstantu $k > 1$.

Řešení
Zvyšování o konstantu: ukážeme, že řešení neprojde. Nechť ℓ je počáteční kapacita pole, a proveďme $n \gg \ell$ operací, kde pro jednoduchost $n = \ell + \alpha \cdot k$, kde $\alpha \in \mathbb{N}$. Spočtěme jenom cenu na kopírování prvků: celkem provedeme $\sum_{i=0}^{\alpha} \ell + i \cdot k$ operací, což je součet prvních α členů aritmetické posloupnosti, kde α můžeme vyjádřit jako $\frac{n-\ell}{k}$. Pro součet máme vzoreček, který zní $\alpha \cdot \frac{\ell+(\ell+\alpha k)}{2} = \frac{n-\ell}{k} \cdot \frac{\ell+n}{2} \in \Omega(n^2)$ (protože k, ℓ jsou konstanty). Z toho plyne, že součet n operací má cenu alespoň $\Omega(n^2)$, a proto amortizovaně nemůžeme mít cenu lepší než $\Omega(n)$ na operaci.
Zvětšení na druhou mocninu: zde záleží na našich předpokladech, konkrétně na tom, jak funguje alokace pole. Dopadne to jinak, pokud alokace je v konstantním čase, a jinak pokud je v čase lineárním.
Pokud alokujeme v konstantním čase, pak použijeme penízkový argument jako u zdvojnásobení. Těsně po zvětšení pole máme pole s kapacitou C^2 obsahující C prvků. Máme tedy zbývajících $C^2 - C$ insertů, během nichž potřebujeme ušetřit C^2 penízků. To uděláme prostě tak, že při každém insertu budeme mít tři penízky: jeden použijeme na vložení prvku do pole, a dva si schováme na překopírování. Protože alokace je konstantní, stačí nám jenom zaplatit překopírování, na což máme $2(C^2 - C) \geq C^2$ penízků (pro C alespoň 3).
Pokud ovšem alokujeme v lineárním čase, pak si musíme uvědomit, že musíme zaplatit celou délku pole předem. Speciálně může tedy dojít k tomu, že provedeme přesně tolik insertů, že zvětšíme pole, a hned potom se zastavíme. Celková cena by pak byla $\mathcal{O}(n^2)$, ale my bychom provedli pouze $\Theta(n)$ operací, což znamená, že amortizovaně nemůžeme mít lepší než lineární cenu.
Zvětšení na konstantní násobek: použijeme podobný penízkový argument jako u zdvojnásobování. Obecně po zvětšení máme pole velikosti C, v němž je zaplněno $\frac{1}{k} \cdot C$ položek, a k dalšímu zkopírování dojde po $(1 - \frac{1}{k})C = \frac{k-1}{k}C$ vloženích. Celkem potřebujeme mít našetřeno C, ale máme na to pouze $\frac{k-1}{k}C$ vložení. To znamená, že každé přidání prvku musí zaplatit $\frac{C}{\frac{k-1}{k}C} = \frac{k}{k-1}$ zkopírování prvku, a k tomu ještě navíc své vlastní vložení. To tedy můžeme udělat tak, že každý insert zaplatí $1 + \frac{k}{k-1} \in \mathcal{O}(1)$ penízků, kde první penízek zaplatí vložení prvku do pole, a další penízky si pošetříme na zaplacení kopírování.

Bonusové úlohy

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

Řešení
$=$ by mělo být $\in$

Úloha 6 (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.)

Řešení
Rozděl a panuj: vezmeme prostředek (délku pole víme předem, nebo si ji jedním průchodem spočítáme) a zarekurzíme se na levé podpole a pravé podpole (už s informací o délce).

Úloha 7 (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.

Řešení
TODO