2. cvičení

Datové struktury I, 11. 10. 2022 https://iuuk.mff.cuni.cz/~chmel/2223/ds1/

Úloha 1 (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?

1. $C \rightsquigarrow C + k$, kde $k \geq 1$ je konstanta,
2. $C \rightsquigarrow C^2$,
3. $C \rightsquigarrow k \cdot C$ pro konstantu $k > 1$.

Úloha 2 (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 3 (Binární čítač)
Připomeňte si důkaz, že binární počítadlo (na začátku nastavené na 0) má amortizovanou konstantní složitost přičítání jedničky.
Jak se změní amortizovaná složitost, pokud povolíme přičítání i odčítání jedničky? Lze upravit počítadlo, aby zůstala konstantní?

Řešení
Bude to $\Theta(\ell)$, protože pro ℓ operací máme hodnotu všech operací přibližně $\sum_{k \geq 1} \frac{\ell}{2^k} = 2\ell$.
Odečtení jedničky: nepůjde to takhle snadno, může se stát že dojdu na nějaké číslo 2^k, a pak budu opakovat operaci -1, +1 až do konce.
Úprava může vypadat například tak, že budeme mít dva čítače K, Z (kladný a záporný) s vlastností, že každý bit má maximálně jeden z těchto čítačů nastavený na jedničku – výsledná hodnota pak bude $K - Z$. Když tedy budu přičítat, přičtu +1 ke K, když budu odečítat, přičtu +1 k Z. Stačí jen zařídit invariant s tím, že na stejném bitu oba čítače nebudou mít jedničku. K tomu stačí jen rychlá úprava přičítacích operací - když máme posloupnost i jedniček na nejnižších bitech v jednom z čítačů a přičteme jedničku, pak nám vznikne jednička na bitu $i + 1$ - pokud je jednička i v druhém čítači, pak ji odečteme, a do našeho původního čítače dáme na bit $i + 1$ taky nulu. Pokud je v druhém čítači ovšem nula, pak si jedničku zapíšeme.
Amortizaci pak jde snadno analyzovat pomocí potenciálu, který zadefinujeme jako počet jedniček v obou čítačích. Pak reálná cena je počet změněných bitů z 0 na 1 + počet změněných bitů z 1 na 0, změna potenciálu je počet změněných bitů z 0 na 1 - počet změněných bitů z 1 na 0, a tedy amortizovaná cena je dvakrát počet změněných bitů z 0 na 1. Při každém přičtení ale můžeme nejvýše jeden bit změnit z nuly na jedničku, a tedy amortizovaná cena je konstantní.

Bonusové úlohy

Úloha 4 (Binární čítač II)
Jsme ve stejné situaci jako v předchozí úloze a chceme jenom přičítat. Ukažte, že když za přehození k-tého bitu platíme 2^k (a nultý bit je nejméně významný bit), pak posloupnost ℓ přičtení jedničky stojí $\mathcal{O}(\ell \cdot n)$.

Řešení
Pro ℓ operací máme $\sum_{i=1}^{\lceil \log \ell \rceil} \frac{2^i \cdot \lfloor \ell \rfloor}{2^i} \leq \ell \cdot \lceil \log \ell \rceil$.

Úloha 5 (Sublineární množina)
Máme následující implementaci množiny (celých čísel), která podporuje operace Insert, Lookup.

Množina `M` bude implementovaná jako pole polí, kde pole `M[i]` má délku 2^i. Každé pole `M[i]` je buď prázdné, nebo plné, a každé pole je seřazené (mezi jednotlivými poli nemusí být žádný vztah).
Konkrétně, máme-li v naší množině 9 prvků, tak pole `M[0], M[3]` budou plná a ostatní budou prázdná (vlastně to odpovídá binární reprezentaci čísla 10). Příkladem může být například:

```
M[0] = {5}
M[1] = {}
M[2] = {}
M[3] = {3, 4, 8, 10, 11, 15, 19, 22}
```

a) Jaká je složitost operace LOOKUP, kterou lze implementujeme tak, že na každém plném poli provedeme binární vyhledávání?
b) Operaci INSERT budeme implementovat následovně: začneme s tím, že vytvoříme prázdné pole s jedním prvkem, který vkládáme. Dále se podíváme na pole `M[0]`. Pokud je prázdné, vložíme naše nové pole tam. Pokud je plné, pak slijeme naše nové pole s polem `M[0]` a pokračujeme dále s polem `M[1]`, dokud nenajdeme pole `M[i]`, které je prázdné. Jaká je (amortizovaná) složitost operace INSERT?
Hint: může se vám hodit předchozí úloha.

Řešení
a) Worstcase máme složitost $\sum_{i=0}^{\log n} \log(2^i) = \sum_{i=0}^{\log n} i = \frac{\log(n)\cdot(\log(n)+1)}{2} \in \Theta(\log^2 n)$.
b) Worstcase to může být lineární, třeba přidání pro $n = 2^k - 1$ znamená, že musíme slít všechna pole do jednoho velkého, ale můžeme to uamortizovat to díky čítači II: jedno slití trvá řádově 2^{k+1}, takže máme (až na konstantu) přičítání u čítače II, a díky tomu, že ten je amortizovaně $\mathcal{O}$(délka čítače), což je v našem případě $\log(n)$, jsme hotovi.