2. CVIČENÍ Z DATOVEK 1, ZS24/25

Amortizujeme!

1. *Nafukování pole jinak.* Na přednášce bylo nafukovací pole (zásobník), u něhož při zaplnění zdvojnásobíme velikost a tím dosáhneme amortizované složitosti $O(1)$ na vložení. Co kdybychom při zaplnění místo toho:

a) zvětšili velikost pole o konstantní počet prvků?
b) zvětšili velikost pole z m na m^2?
c) zvětšili velikost pole k krát pro parametr $k > 1$?

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?

3. *Fronta pomocí dvou zásobníků.* Mějme dva zásobníky neomezené kapacity (vkládání/odebírání probíhá jen z vrchu zásobníku). Jak pomocí nich implementovat frontu (pouze s konstantní pomocnou pamětí)? Jakou amortizovanou složitost mají operace s frontou?

4. *Binární počítadlo s odčítáním.* Na přednášce byl důkaz, že binární počítadlo přičítá +1 (operace INC) v amortizovaně konstantním čase.

a) Ukažte, že pokud bychom povolili odčítání -1 (DEC), amortizovaná časová složitost se zhorší.
b) Navrhněte jinou reprezentaci celých čísel, v níž bude možné provádět operace INC, DEC a TESTZERO (vrátí, jestli je číslo nulové nebo ne) v amortizovaně konstantním čase.

5. *Víc než jen inkrement.* Uvažujme místo INC operaci ADD(k), která k počítadlu přičte číslo k. Dokažte, že amortizovaná složitost této operace je $O(\log k)$.

6. *Okénková minima amortizovaně konstantně:* Na vstupu postupně přicházejí čísla v nekonečném proudu. Kdykoliv přijde další, vypište minimum z posledních k čísel. Cílem je dostat amortizovaně konstantní čas na jedno číslo.

7. *Dražší binární počítadlo.* Uvažme binární počítadlo, které umí přičítat jenom jedničku, ale cena závisí na pozici bitu. Ukažte, že když za přehození k-tého bitu platíme 2^k (a nultý bit je nejméně významný bit), pak posloupnost ℓ přičtení jedničky stojí $\mathcal{O}(\ell \cdot \log \ell)$.

8. *Sublineární množina.* Máme následující implementaci množiny (celých čísel), která podporuje operace INSERT a 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.