1. *Změna nafukování:*

   (a) *Uvažujme, že bychom pole namísto zdvojnásobování zvětšovali o konstantní počet prvků. Dokažte, se tím pokazí časová složitost.*

   (b) *Jak by to dopadlo, kdybychom m-prvkové pole rovnou zvětšovali na m^2-prvkové? Počáteční velikost musíme samozřejmě zvýšit na konstantu větší než 1.*

   V případě (a) uvažme zvyšování kapacity pole o k prvků. Pokud začneme s prázdným polem a uděláme celkem n Insertů, realokujeme postupně pole s jedním prvkem, $1+k$ prvky, $1+2k$ prvky, $\ldots$, $1+\lfloor \frac{n}{k} \rfloor k$ prvky. Tedy jenom realokací musíme strávit čas rovný součtu těchto velikostí:

   $$\sum_{i=1}^{\lfloor \frac{n}{k} \rfloor} 1 + ik \geq \sum_{i=1}^{\lfloor \frac{n}{k} \rfloor} ik = k \sum_{i=1}^{\lfloor \frac{n}{k} \rfloor} i \in \Omega\left(\frac{n^2}{k}\right)$$

   V případě (b) závisí na uvažovaném výpočetním modelu, konkrétně na tom jestli pro pole velikosti c strávíme $O(c)$ času alokací nebo ho dostaneme v čase $O(1)$ pokud nevyžadujeme vynulování. V prvním případě uvažme posloupnost $m+1$ operací Insert pro m takové, že m je velikostí pole po nějakém nafouknutí. Potom musíme naalokovat pole velikosti m^2 za což zaplatíme $\Theta(m^2)$ a tedy jedna operace Insert nemůže amortizovaně vyjít na složitost $O(1)$. V případě pokud nás alokace pole nic nestojí nám bude amortizace penízkovou metodou fungovat stejně jako při obvyklém zdvojnásobování. Jenom je tenhle přístup paměťově velmi neefektivní.

2. *Zásobníková fronta: K dispozici jsou dva zásobníky, které podporují pouze operace* Push *(přidej na vrchol zásobníku) a* Pop *(odeber z vrcholu zásobníku). Navrhněte algoritmus, který bude pomocí těchto dvou zásobníků simulovat frontu s operacemi* Enqueue *(přidej na konec fronty) a* Dequeue *(odeber z počátku fronty). Kromě zásobníků máte k dispozici pouze konstantní množství paměti. Ukažte, že operace s frontou budou mít amortizovaně konstantní časovou složitost.*

   Označíme zásobníky S_1 a S_2. Operace Enqueue provede operaci Push na S_1, tedy vloží prvek do zásobníku S_1. Operace Dequeue provede operaci Pop na S_2 pokud je S_2 neprázdný. Jinak přemístí celý obsah S_1 do S_2 a poté provede Pop na S_2. Prvky dostaneme ve FIFO pořadí, stačí tedy vyargumentovat amortizovanou složitost.

   Každý prvek dostane k dispozici 4 penízky. Jeden zaplatí za přidání do S_1, dva za přesun mezi zásobníky, a poslední za Pop z S_2.

3. *(Od)čítač:*

   (a) *Rozmyslete, že kdyby mělo obyčejné binární počítadlo (z přednášky) podporovat zároveň operace* Inc *a* Dec *(tedy zvyšení a snížení o 1), operace rozhodně nebudou mít konstantní amortizovanou složitost.*

   (b) *Navrhněte jinou reprezentaci čísel, v níž bude možné provádět operace* Inc, Dec *a* TestZero *(zjisti, zda číslo je nulové) v amortizovaně konstantním čase. Nápověda: povolte krom 0,1 i -1.*

   Pro (a) uvažte střídání operací Inc a Dec na čísle $2^n - 1$.

   Pro (b) zavedeme reprezentaci binárních čísel pomocí O,1,-1. Operace Inc zvýší -1 nebo 0 a skončí, nebo změní 1 na 0 a pokračuje ve vyšším řádu. Podobně operace Dec sníží 1

nebo 0 a skončí, nebo změní -1 na 0 a pokračuje ve vyšším řádu. Operace tedy pokračují pouze přes hodnoty ± 1 a vždy vyrobí nejvýše jednu ± 1. Stačí tedy opět dát každé operaci dva penízky, jeden zaplatí za vyrobení ± 1 a jeden nechá na místě pro budoucí vynulování.

Implementace TestZero je jednoduchá pokud si uvědomíme, že 0 lze reprezentovat pouze jedním způsobem - samými nulami. Pro spor uvažte, že by existovala jiná reprezentace 0 - nechť x je číslo odpovídající sečtení kladných řádů, a y sečtení záporných řádů. Potom $x - y = 0$ neboli $x = y$ a tedy jsme dostali dvě různé binární reprezentace čísla x, což není možné. Pro TestZero si tedy například můžeme udržovat značku na nejmenším řádu i takovém, že $2^i > |x|$ a pak jen kontrolovat jestli je značka na nultém řádu.

4. *Pokročilé počítadlo: Uvažujme místo* Inc *operaci* Add(k)*, která k počítadlu přičte číslo k. Dokažte, že amortizovaná složitost této operace je $O(\log k)$.*

   Číslo k má $\log k$ číslic v binárním zápisu. Každou z nich tedy přičteme individuálně jako při operaci Inc, jen začneme od odpovídajícího řádu. Rozmyslete si, že nám to nic nepokazí na amortizaci penízkovou metodou.