Algoritmizace NPRG062 – úkol č. 3

Datové struktury & rekurze
pátek 9:50

Řešení úloh vypracujte ručně (na papír, případně ve Wordu / LaTeXu)[1] a odevzdejte do ReCodExu nejlépe v jednom PDF souboru. Svůj postup řešení podrobně popište – odpovědi *ano*, *lineární*, apod. nestačí. Pokud používáte papír a nemáte přístup ke skeneru, můžete využít některou mobilní aplikaci pro skenování pomocí fotoaparátu.

1 Dva zásobníky [4 body]

Máte k dispozici dva zásobníky s operacemi `push()` a `pop()`.

a) Jak je možné pouze s těmito zásobníky implementovat frontu s operacemi `enqueue()` a `dequeue()`?

b) Předpokládejte, že operace se zásobníky mají konstantní složitost. Jakou složitost budou mít operace s frontou vytvořenou pomocí zásobníků?

2 Zvýšení hodnoty klíče [5 bodů]

Zapište pseudokódem operaci IncreaseKey(A, i, n), která zvýší hodnotu prvku v haldě A na pozici i o hodnotu $n > 0$. Pracujeme s maximovou haldou uloženou v poli indexovaném od nuly.

Zdůvodněte, že operace skončí a že po dokončení operace bude halda v konzistentním stavu. Jakou bude mít operace asymptotickou složitost?

3 Pošta [6 bodů]

Na poště to funguje následovně:

- Okénko Dopisy umožňuje výdej a odesílání dopisů. Výdej dopisů je rychlejší, proto by každý člověk čekající na *výdej dopisu* měl být odbavený před lidmi, co chtějí *odeslat dopis*.
- Okénko Balíky umožňuje výdej a odesílání balíků. Výdej balíků je rychlejší, proto by každý člověk čekající na *výdej balíku* měl být odbavený před lidmi, co chtějí *odeslat balík*.
- Okénko Ostatní umožňuje tisk dokumentů, ověřování certifikátů a prodej losů do loterie.

Každý člověk po příchodu na poštu přijde k automatu, zvolí účel svojí návštěvy a dostane číslo, podle kterého je vyvolán k okénku. Předpokládá se, že zákazníci se stejným účelem návštěvy budou obsluhováni v pořadí, ve kterém přišli. Na poště (i kvůli vládním nařízením) nebude více než několik desítek zákazníků současně.

[1] Grafickou úpravu nehodnotím ;)

Hardware na poště je bohužel žalostně zastaralý. Pokud jde o paměť, máte k dispozici pouze pole na 1000 integerů a k tomu malé množství paměti pro několik globálních proměnných. Systém je navíc velmi pomalý a při operacích s vyšší než logaritmickou asymptotickou složitostí zamrzá.

Jak *co nejefektivněji* využít tento systém pro přiřazování čísel zákazníkům, aby plnil všechny požadavky? Popište slovně, jaké datové struktury / proměnné / algoritmy použijete a jak se bude systém chovat v různých situacích.

4 Binární stringy [6 bodů]

Zapište pseudokódem *rekurzivní* funkci, která vygeneruje všechny binární stringy (sekvence nul a jedniček) délky $N > 0$, které mají stejný součet číslic v levé i pravé polovině. Pokud je N liché, prostřední číslice může být jak 0, tak 1.

Příklady:

- $N = 3$: 000, 010, 101, 111.
- $N = 4$: 0000, 0101, 0110, 1001, 1010, 1111.

Popište (obrázkem nebo slovně), jak se funkce bude chovat.

5 Padovanova posloupnost [4 body]

Padovanova posloupnost čísel $P(n)$ je definována následujícím předpisem:

$$
\begin{aligned}
P(0) &= 1 \\
P(1) &= 1 \\
P(2) &= 1 \\
P(n) &= P(n-2) + P(n-3)
\end{aligned}
$$

První členy posloupnosti jsou: $1, 1, 1, 2, 2, 3, 4, 5, 7, 9, 12, \ldots$

V čem spočívá neefektivita následujícího programu pro výpočet n-tého členu posloupnosti? Jak program upravit, aby byl efektivnější?

```
def padovan(n):
  if n in [0, 1, 2]:
    return 1

  return padovan(n−2) + padovan(n−3)
```