Domáca úloha č. 2

1-AIN-105, Zima 2023

Termín: 30.10.2023, 22:00

Skôr ako sa pustíte do riešenia domácej úlohy, oboznámte sa so všeobecnými pokynmi, ktoré sú priložené na konci tohto dokumentu. Riešenia, ktoré odovzdáte, musia byť vaše vlastné. Neopisujte a nesnažte sa nájsť riešenia v literatúre alebo na internete!

1. [*20 bodov*] **Abstraktný dátový typ množina.** Abstraktný dátový typ množina ukladá množinu prvkov, pričom má operácie *Insert(x)* (vlož prvok do množiny), *Remove(x)* (odstráň prvok z množiny, ak sa v nej nachádza) a *isMember(x)* (zisti, či sa prvok nachádza v množine). Prvok sa môže nachádzať v množine len raz, preto ak už sa prvok x v množine nachádza, tak následný *Insert(x)* nemá žiaden efekt.

a) Navrhnite dátovú štruktúru, ktorá implementuje množinu, pričom prvky môžu byť celé čísla od 0 po n. Všetky tri operácie musia fungovať v čase $\Theta(1)$, máte však povolené použiť pamäť o veľkosti $O(n)$ a pri inicializácii dátovej štruktúry môžete použiť predspracovanie v čase $O(n)$.

b) Predstavte si, že máte pole $A[1..m]$, pričom v poli A sú uložené celé čísla medzi 0 a n a tieto čísla sú rôzne (logicky teda $m < n$). Uvažujte nasledujúci pseudokód:

```
pole B[0..n] je nainicializované náhodnými celými číslami
for i=1 to m
  B[A[i]] = i
```

Viete pre konkrétne k v konštantnom čase o hodnote $B[k]$ povedať, či bola modifikovaná v rámci pseudokódu, alebo či sa v nej nachádza pôvodná náhodná hodnota?

c) Ako by ste zmodifikovali implementáciu z časti a) aby nebola potrebná náročná inicializácia dátovej štruktúry? To znamená, že všetky operácie vrátane inicializácie by mali trvať $\Theta(1)$.

Hint: Môžete použiť pamäť o veľkosti $O(n)$ a súčasne môžete predpokladať, že programovací jazyk, v ktorom pracujete, umožňuje pole o veľkosti n naalokovať v konštantnom čase bez toho, aby ho bolo potrebné inicializovať.

2. [*20 bodov*] **Dvojsmerne triedené pole.** Dvojsmerne triedené pole (2STP) je pole $m \times n$ čísel, kde prvky v každom riadku a v každom stĺpci sú utriedené od najväčšieho po najmenšie. Aby sme vedeli v 2TSP ukladať aj menej ako mn čísel, tak niektoré prvky môžu byť $-\infty$. Takéto prvky však tiež musia spĺňať podmienky na utriedenie riadkov a stĺpcov, tzn. že musia byť buď na konci riadku alebo na spodku stĺpca. Príklad, ako môže vyzerať 2STP:

$$\begin{pmatrix} 15 & 12 & 10 & 9 \\ 14 & 11 & 8 & 7 \\ 13 & 6 & 2 & 1 \\ 5 & 3 & -\infty & -\infty \\ 4 & -\infty & -\infty & -\infty \end{pmatrix}$$

a) Ukážte, ako môžete použiť 2STP ako implementáciu abstraktného dátového typu prioritná fronta s operáciami `insert` a `extractMax`. Obe operácie (insert, extractMax) by ste mali vedieť implementovať v čase $O(m + n)$. Môžete predpokladať, že na začiatku je celá matica nainicializovaná hodnotami $-\infty$ a že v každom okamihu je v matici aspoň jeden prvok $-\infty$ (t.j. nedôjde k preplneniu dátovej štruktúry)

b) Pripomeňme si, že triediť pomocou prioritnej formy môžeme tak, že najprv vložíme do dátovej štruktúry všetkých N prvkov pomocou operácie `insert` a potom vyťahujeme zo štruktúry utriedené prvky pomocou

operácie `extractMax`. Ak by ste použili implementáciu z časti a) na takéto triedenie, akú časovú zložitosť triedenia by ste vedeli dosiahnuť?

Vaša odpoveď by mala závisieť len od čísla N, preto súčasťou vašej odpovede by malo byť, ako zvoliť rozmery m a n matice pre 2STP.

c) Implementácia prioritnej fronty pomocou 2STP má výhodu, že okrem operácií `insert` a `extractMax` viete implementovať aj nejaké ďalšie operácie v dobrej časovej zložitosti. Ukážte, ako môžete nájsť v čase $O(m+n)$ najmenší prvok uložený v prioritnej fronte (t.j. najmenší prvok, ktorý je uložený v 2STP matici, ktorý nie je $-\infty$).

3. [*20 bodov*] **Username do AISu (programátorská úloha).** Kvôli neustálym sťažnostiam na univerzitný AIS sa Študentský vývojový tím FMFI UK rozhodol naprogramovať vlastný Alternatívny Informačný Systém (skrátene AIS). Najpodstatnejšia vec, ktorú treba naprogramovať je výber používateľského mena pre nových používateľov. Ten (oproti starému AISu) bude zohľadňovať preferenciu používateľov na používateľské meno. Ak už je ich želané meno, napríklad `eads`, obsadené, portál im láskyplne ponúkne číslovanú náhradu, napríklad `eads123`.

Keďže ŠVT je aktuálne zaneprázdnený vyrábaním alternatívnych ikon bežca do nového AISu, potrebujú, aby ste túto funkcionalitu naimplementovali vy!

V tejto úlohe budeme predpokladať nasledovné skutočnosti:

1. Každé želané meno je buď tvorené len malými písmenami (anglickej abecedy), alebo najskôr malými písmenami (tvoriacimi tzv. základné meno) a potom kladným celým číslom (tvoriacim tzv. sufix).
2. Ak je želané meno k dispozícii, užívateľ ho dostane.
3. Ak nie, ponúkneme mu najlepšiu možnosť, ktorú ešte máme k dispozícii, v poradí: najskôr samotné základné meno a následne základné meno s číselnými sufixami 1, 2, 3, atď.
4. Užívateľ vždy ponúknuté meno akceptuje.

Vstup. Vstup obsahuje niekoľko riadkov (aspoň jeden, najviac stotisíc). V každom riadku je jedno želané meno. Dĺžka každého riadku je medzi 1 a 25, vrátane. Každé želané meno má vyššie popísaný tvar a obsahuje aspoň jedno písmeno. Posledný riadok vstupu obsahuje práve jeden znak `0`.

Výstup. Pre každé želané meno vypíšte jeden riadok s menom, ktoré dotyčný užívateľ naozaj dostane.

Hint: Už by ste mali poznať niektoré vhodné dátové štruktúry. Použitie vhodnej dátovej štruktúry vám vie výrazne uľahčiť život. Ak na veľkých vstupoch prekračujete časový limit, niečo robíte principiálne zle. A dotyčné "niečo", ktoré robíte zle, je skoro určite hľadanie mena, ktoré užívateľovi ponúknuť.

Príklad vstupu:

```
jano
baska
jano
jano
michal27
jano4
jano4
jano
jano1
0
```

Príklad výstupu:

```
jano
baska
jano1
jano2
michal27
jano4
jano3
jano5
jano6
```

Všimnite si obzvlášť, že predposledný `jano` *dostal meno* `jano5`*, keďže* `jano4` *už bolo obsadené skôr.*

Príklad vstupu:

```
jano2
jozo
jano2
0
```

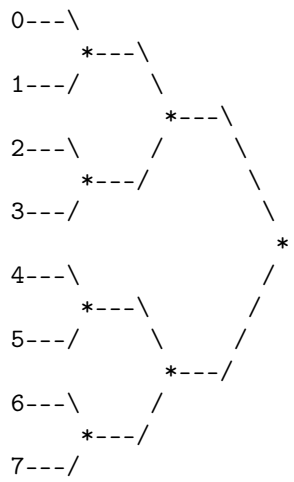
Príklad výstupu:

```
jano2
jozo
jano
```

Odovzdávanie: Odkaz na odovzdanie programátorskej úlohy : `https://testovac.ksp.sk/tasks/eads2023-du2a/`

3. [*20 bodov*] **Tenis (programátorská úloha).** V tenisovom turnaji sa stretlo 2^n hráčov. Turnaj vyzerá tak, že sa hráči očíslujú od 0 po $2^n - 1$ a následne sa spraví tzv. pavúk. V každom kole zoberieme všetkých hráčov, ktorí ešte nevypadli, usporiadame ich podľa čísla a popárujeme do dvojíc. Každá dvojica odohrá zápas a porazený z turnaja vypadne.

Pre 8 ľudí to teda vyzerá takto (každá hviezdička je zápas):



Pre jednoduchosť budeme predpokladať, že každý tenista má *zručnosť*, čo jecelé číslo, ktoré hovorí, aký je dobrý. Keď sa stretnú dvaja tenisti, vždy vyhráva ten s vyššou zručnosťou; v prípade rovnosti zučností vyhráva ten s menším číslom.

Táto úloha má dve fázy. V prvej fáze dostanete na vstupe zručnosti všetkých tenistov a vašou úlohou bude určiť víťaza turnaja. V druhej fáze potom tenisti budú piť rôzne "energetické nápoje" a vy budete sledovať, ako ktorý z nich ovplyvní výsledky turnaja.

Vstup. Vstup začína riadkom obsahujúcim celé číslo n ($1 \leq n \leq 20$), udávajúce binárny logaritmus počtu tenistov. Tenistov je teda presne $m = 2^n$. V druhom riadku je m celých čísel: začiatočné zručnosti všetkých tenistov v poradí od tenistu 0 po tenistu $m - 1$.

V treťom riadku je počet z zmien, ktoré sa následne udejú ($0 \leq z \leq 100\,000$). Zvyšok vstupu tvorí z riadkov, v každom z nich je popis jednej zmeny: číslo tenistu, ktorý sa napil nápoja, a zmena zručnosti, ktorú mu to spôsobilo. Zmeny sú udané v chronologickom poradí.

Zručnosť žiadneho tenistu nikdy nebude mať viac ako 9 cifier (môže však byť aj záporná).

Výstup. V prvom riadku vypíšte číslo a zručnosť tenistu, ktorý by vyhral turnaj, ak by nik nič nepil. Následne postupne spracúvajte zmeny. Pre každú zmenu vypíšte, koľko zápasov by vyhral tenista, ktorý sa práve napil, ak by sa turnaj hral tesne po tom ako dopije.

Hint: *Zamyslite sa nad tým, ako si celého pavúka na začiatku vypočítať a uložiť.* ***Poriadne*** *si rozmyslite, ktoré zápasy treba prepočítať, keď sa zmení zručnosť jedného tenistu! (Niektoré veci sa optimalizovať naozaj neoplatí, obzvlášť ak tým vznikne chyba.)*

Príklad vstupu:

```
3
17332 39133 37242 14235 656 12265 20598 6471
4
0 1000
4 39000
7 15127
7 -1000
```

Príklad výstupu:

```
1 39133
0
3
1
0
```

- Máme $2^3 = 8$ tenistov. Ak nik nič nepije, turnaj vyhrá tenista číslo 1, lebo jeho zručnosť (39133) je najväčšia.
- Po prvej zmene bude mať tenista 0 zručnosť 18332, ale aj tak hneď svoj prvý zápas prehrá.
- Po druhej zmene bude mať tenista 4 zručnosť 39656, a teda vyhrá celý turnaj.
- Tretia zmena stačí tenistovi 7 na to, aby vyhral svoj prvý zápas, no v druhom zápase prehrá s tenistom 4 (ktorý má stále zručnosť 39656).
- Po štvrtej zmene majú tenisti 6 a 7 rovnakú zručnosť, vyhráva teda tenista 6.

Odovzdávanie: Odkaz na odovzdanie programátorskej úlohy : `https://testovac.ksp.sk/tasks/eads2023-du2b/`

Všeobecné pokyny

Písomné úlohy. Písomné úlohy odovzdávajte *do Google Classroom* ako PDF súbory v stanovenom termíne. **Každý príklad odovzdajte v osobitnom PDF súbore.** Na neskoro odovzdané riešenia sa nebude prihliadať. Píšte riešenia takým spôsobom, aby obsahovali všetku potrebnú informáciu na pochopenie vášho riešenia, ale súčasne aby boli stručné a ľahko pochopiteľné. Všetky tvrdenia je potrebné zdôvodniť (a to aj v prípade, že to nie je explicitne napísané v zadaní).

Ak sa v zadaní požaduje vyriešenie algoritmickej úlohy, odovzdajte najlepší algoritmus, aký viete navrhnúť. Základným kritériom na hodnotenie bude *správnosť algoritmu*, druhým kritériom bude jeho *časová, prípadne pamäťová zložitosť*. Správny ale pomalý algoritmus dostane podstatne viac bodov ako algoritmus, ktorý je síce rýchly, ale nedá správnu odpoveď na každý vstup. Neefektívne algoritmy spĺňajúce podmienky zadania dostanú cca 50% bodov. Súčasťou vášho riešenia musia byť nasledujúce časti:

- Najprv popíšte hlavnú myšlienku algoritmu.
- Vyjadrite algoritmus formou pseudokódu.
- Ak to nie je zrejmé na prvý pohľad, ukážte že váš algoritmus je správny.
- Nezabudnite na analýzu zložitosti algoritmu.

Ak nie je povedané inak, logaritmy majú základ 2.

Programátorské úlohy. Pri programátorských úlohách je Vašou úlohou odovzdať len funkčný program, nie je vyžadované písomné riešenie. Riešenie odovzdávate cez webové rozhranie `https://testovac.ksp.sk/tasks/`, kde bude okamžite otestované na niekoľkých vstupoch a dozviete sa, koľko bodov získalo (body získate, keď všetky vstupy z danej sady vyriešite správne v časovom limite). Riešenie môžete odovzdávať aj viackrát, hodnotí sa posledné riešenie odovzdané v stanovenom termíne. Na odovzdávanie riešení (keďže na testovač nefungujú univerzitné prihlasovacie údaje) je nutné sa na stránke zaregistrovať (vľavo na stránke testovača). Pri vytváraní účtu nastavte správne meno a priezvisko, a ako používateľské meno nastavte Váš univerzitný login. Nezabudnite tiež napísať Vaše používateľské meno do PDF súboru k ostatným úlohám, ktoré odovzdávate. Podrobnosti o tom, ako má váš program vyzerať (vrátane povolených programovacích jazykov), nájdete v sekcii "Čo odovzdávať?". Informácie o testovači nájdete v sekcii "Odpovede testovača".

Na zoznámenie sa s rozhraním testovača si môžete vyskúšať naprogramovať niektoré z úloh z časti "Úvod do programovania".