Algoritmi in podatkovne strukture 1

2. september 2024

Ime in priimek: _____

Vpisna številka: _____

Pri reševanju si lahko pomagate z enim A4 listom zapiskov.

Svoje odgovore pišite jasno in jih utemeljite.

Ocenjuje se odgovore na izpitni poli. Dodatni listi so namenjeni pomoči pri reševanju in jih ne oddate.

Čas reševanja: 90 min.

Naloga:	1	2	3	4	Skupaj
Točke:	25	25	25	25	100
Ocena:					

1. Podan je seznam n števil, ki ga bomo sešteli na inovativen način. Vsakič bomo izbrali najmanjši dve števili, ju odstranili iz seznama in vanj dodali njuno vsoto. Če to ponavljamo, nam na koncu ostane eno samo število, ki je vsota seznama. Kateri dve števili sta bili zadnji dve prisotni v seznamu (iz katerih smo v zadnjem koraku izračunali vsoto)?

[15] (a) Opišite čim bolj učinkovit algoritem, ki izračuna odgovor na zastavljeno vprašanje. Navedite in utemeljite tudi njegovo časovno in prostorsko zahtevnost.

[10] (b) V C++ implementirajte funkcijo `vsota`, ki bo sprejela seznam števil in vrnila par iskanih števil.

Implementacija lahko ustreza manj učinkovitemu postopku od prej opisanega, vendar bo zato prejela manj točk (v tem primeru jo tudi na kratko opišite). Pri implementaciji lahko uporabljate vso funkcionalnost standardne knjižnice C++.

```
1 vector<int> stevila = {3,5,3,1,6};
2 pair<int,int> zadnji = vsota(stevila);
3 // {7,11}
```

2. Podan je bil urejen seznam n števil $x_1, x_2, \ldots, x_n$. Nekdo je v tem seznamu izbral indeksa i in j ter med seboj zamenjal števili na indeksih i in j. Tako je iz seznama

$$x_1, \ldots, x_{i-1}, \boldsymbol{x_i}, x_{i+1}, \ldots, x_{j-1}, \boldsymbol{x_j}, x_{j+1} \ldots, x_n$$

nastal nov seznam

$$x_1, \ldots, x_{i-1}, \boldsymbol{x_j}, x_{i+1}, \ldots, x_{j-1}, \boldsymbol{x_i}, x_{j+1} \ldots, x_n.$$

Kako se na takih skoraj urejenih seznamih, ki so rezultat medsebojne zamenjave dveh elementov, v najslabšem primeru obnesejo spodnji algoritmi za urejanje? Indeksov zamenjav i in j ne poznamo. Za vsak algoritem napišite in utemeljite njegovo časovno zahtevnost v odvisnosti od velikosti seznama n.

[5] (a) urejanje z izbiranjem (*selection sort*)

[5] (b) urejanje z vstavljanjem (*insertion sort*)

[5] (c) urejanje z zamenjavami (*bubble sort*) z zgodnjo ustavitvijo, ko ni več zamenjav

[4] (d) hitro urejanje (*quick sort*) z izbiro prvega elementa za pivot

[3] (e) urejanje z zlivanjem (*merge sort*)

[3] (f) urejanje s kopico (*heap sort*)

3. Odgovorite na vprašanja o Dijkstrovem algoritmu. Utemeljite svoje odgovore.

[5] (a) Opišite problem, ki ga algoritem rešuje.

[10] (b) Poznamo več različic Dijkstrovega algoritma. V čem se razlikujejo med seboj? Opišite dve različici in njuno časovno ter prostorsko zahtevnost.

[5] (c) Kdaj bi uporabili katero od različic?

[5] (d) Razložite, zakaj ne moremo uporabiti Dijkstrovega algoritma v primeru negativnih uteži povezav, tudi če ni negativnih ciklov.

4. Pri analizi potenciranja s kvadriranjem za izračun potence x^n smo predpostavili, da lahko poljubno veliki števili zmnožimo v konstantnem času. To predpostavko bomo zamenjali z bolj realističnim množenjem dveh celih števil s časovno zahtevnostjo $O(d \log d)$, kjer je d dolžina njunega produkta.

[10] (a) Predpostavimo, da so vsa števila v množenjih vedno krajša od m. Kakšna je časovna zahtevnost potenciranja s kvadriranjem v odvisnosti od x, n, m in zakaj?

[10] (b) Namesto prejšnje predpostavke o omejitvi dolžine števil bomo dolžino števila a bolj natančno merili v desetiškem zapisu z $d(a) = \log_{10} a$. Predpostavite lahko, da je n ravno potenca števila 2. Kakšna je časovna zahtevnost potenciranja s kvadriranjem v odvisnosti od x in n?

[5] (c) Kaj je krovni izrek (*master theorem*) v kontekstu algoritmov tipa deli in vladaj in zakaj si v nobenem od zgornjih dveh vprašanj ne morete prav dosti pomagati z njim?