Algoritmi in podatkovne strukture 1

1. februar 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. V računalniški igrici želimo svojemu junaku kar čim bolj povečati moč z uporabo n čarobnih napojev. Na začetku je moč junaka enaka z. Vemo tudi, da junaku i-ti napoj poveča moč za a_i, vendar ga junak lahko odpre in uporabi samo, če za njegovo trenutno moč x velja, da je $x \geq a_i$. Istega napoja seveda ne more uporabiti večkrat. Kakšna je največja moč, ki jo lahko junak doseže z uporabo napojev?

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

[15] (b) V C++ implementirajte funkcijo `moc`, ki bo sprejela začetno moč z in seznam moči napojev a_i. Vrne naj iskano največjo moč.
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<double> napoji = {4, 20.7, 1, 5, 2.1};
2 double x = moc(2.5, napoji);
3 // 14.6
```

2. Podan je bil urejen seznam n števil $x_1, x_2, \ldots, x_n$. Nekdo je izbral zelo majhno območje od indeksa i do j ($1 \leq i \leq j \leq n$) dolžine $k = j - i + 1$ ($k \ll n$) in obrnil vrstni red števil na tem območju. Tako je iz urejenega seznama

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

nastal nov seznam

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

Kako se na takih skoraj urejenih seznamih, ki so rezultat obrnjenega vrstnega reda manjšega območja, v najslabšem primeru obnesejo spodnji algoritmi za urejanje? Za vsak algoritem napišite in utemeljite njegovo časovno zahtevnost v odvisnosti od parametrov n in k.

[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 sledeča vprašanja o AVL drevesih.

[5] (a) AVL drevo spada med dvojiška iskalna drevesa. Razložite, kaj to pomeni.

[5] (b) V katerem primeru bi za iskanje elementov uporabili AVL drevo namesto dvojiškega iskanja v urejenem seznamu?

[5] (c) Kaj so značilnosti AVL dreves (v čem se razlikujejo od drugih dvojiških iskalnih dreves)?

[10] (d) Podani imamo dve AVL drevesi, ki ju želimo združiti v novo AVL drevo (ki bo vsebovalo elemente obeh dreves). Opišite čim bolj učinkovit algoritem in njegovo časovno ter prostorsko zahtevnost.

4. Pri reševanju problema nahrbtnika smo se pri načrtovanju učinkovitega algoritma z dinamičnim programiranjem zanašali na majhne celoštevilske vrednosti.

[5] (a) Na kratko predstavite problem 0-1 nahrbtnika.

[5] (b) Razložite, kaj je tehnika dinamičnega programiranja.

[5] (c) Predstavite zgoraj omenjeno rešitev problema nahrbtnika z dinamičnim programiranjem.

[10] (d) Kako bi rešili problem nahrbtnika, v katerem so vrednosti predmetov majhna cela števila, ki so navzgor omejena s parametrom m, nosilnost in teže predmetov pa so realna števila? Predstavite čim bolj učinkovit algoritem in njegovo časovno ter prostorsko zahtevnost.
Namig: razmislite, ali lahko s predmeti kako dosežete vrednost x in kakšne so te vrednosti lahko.