Algoritmi in podatkovne strukture 1

27. januar 2025

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 zelo dolg osnovni niz s, iz katerega smo s konkatenacijo različnih podnizov sestavili niza a in b. Ker sta niza predolga, da bi ju shranili eksplicitno, ju bomo predstavili s seznamom podnizov oz. kosov, iz katerih sta sestavljena. Vsak izmed njiju je sestavljen iz največ n kosov, vsak kos pa je predstavljen z začetnim in končnim indeksom podniza v s-ju. S parom (i, j) torej predstavimo podniz $s_i s_{i+1} \ldots s_{j-1}$ (začetni indeks je vključen, končni pa ne).

   Na voljo je funkcija `compare(i1,j1,i2,j2)`, ki zna učinkovito (leksikografsko) primerjati med seboj poljubna dva podniza s-ja $(i1, j1)$ in $(i2, j2)$, recimo da v $O(t)$ časa. Funkcija prejme dva podniza v obliki para indeksov (kjer je začetni indeks vključen, končni pa ne) in vrne -1, 0 ali 1, glede na to, ali je prvi podniz manjši, enak ali večji od drugega. Radi bi jo uporabili za izgradnjo funkcije `compareMulti`, ki bo znala učinkovito primerjati dve konkatenaciji podnizov in prav tako vrnila rezultat -1, 0 ali 1.

[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 `compareMulti`, ki bo sprejela osnovni niz s (preko reference) in dva niza a in b, ki sta predstavljena s seznamom začetnih in končnih indeksov ter vrnila rezultat primerjave.
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 string s = "ABAABABCABACBA";
2 vector<pair<int,int>> a = {{3,6}, {1,4}, {7,8}, {10,12}, {2,6}};
3 // a = ABA + BAA + C + AC + AABA = ABABAACACAABA
4 vector<pair<int,int>> b = {{10,11}, {9,10}, {5,7}, {13,14}, {10,13}, {7,10}};
5 // b = A + B + AB +A + ACB + CAB = ABABAACBCAB
6 int x = compareMulti(s,a,b);  // -1
```

2. Kruskalov algoritem za splošne grafe bomo izvajali na različnih družinah grafov z uteženimi povezavami. Za vsako izmed spodnjih družin navedite in utemeljite časovno zahtevnost izvajanja algoritma.

[5] (a) pot oz. linearen seznam z n vozlišči (P_n)

[5] (b) cikel z n vozlišči (C_n)

[5] (c) drevo z n vozlišči stopnje največ d

[5] (d) poln graf z n vozlišči (K_n)

[5] (e) mrežni graf, ki predstavlja pravokotno mrežo z v vrsticami in s stolpci, kjer je vsako izmed $v \times s$ vozlišč v mreži povezano s svojima sosedoma v isti vrstici in stolpcu (z izjemo tistih na robu mreže)

3. Odgovorite na vprašanja o preskočnem seznamu (*skip list*). Utemeljite svoje odgovore.

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

[6] (b) Na kratko opišite delovanje preskočnega seznama. Kakšni sta časovni zahtevnosti dodajanja ter iskanja elementa?

[6] (c) Kako bi v preskočnem seznamu izvedli operacijo odstranjevanja elementa in kakšno časovno zahtevnost bi imela vaša rešitev?

[8] (d) Kako bi preskočnemu seznamu dodali možnost učinkovitega indeksiranja oz. dostopa do i-tega elementa, pri čemer želimo ohraniti učinkovitost operacij dodajanja, iskanja in odstranjevanja elementa. Kako učinkovita je vaša rešitev?

4. Podan imamo acikličen neusmerjen graf z n vozlišči (V) stopnje največ d. Poiskati želimo dominacijsko množico vozlišč $S \subseteq V$ (dominating set), za katero velja, da dominira vsa vozlišča. Vozlišče x je dominirano z množico S, če je vozlišče prisotno v njej ali pa ima v njej vsaj enega soseda. Če z $N(x)$ označimo množico sosedov vozlišča x, je S dominacijska množica, če velja $\forall x \in V : x \in S \vee N(x) \cap S \neq \emptyset$. Utemeljite (ne)pravilnost sledečih algoritmov za iskanje najmanjše dominacijske množice.

[5] (a) V S bomo dodali vozlišče x, ki dominira čim več drugih vozlišč, izbrisali dominirana vozlišča (x in $N(x)$), ter postopek ponavljali, dokler niso dominirana vsa vozlišča.

[5] (b) V S bomo dodali vozlišče x, ki dominira čim več še nedominiranih vozlišč, in ta postopek ponavljali, dokler niso dominirana vsa vozlišča. V primerjavi s prejšnjim postopkom, je izbrano vozlišče x lahko že dominirano.

[5] (c) Izberemo poljuben list in njegovega soseda označimo z x (če imamo opravka z osamljenim vozliščem, naj bo to vozlišče x). Vozlišče x bomo dodali v S, izbrisali x in $N(x)$, ter postopek ponavljali, dokler niso dominirana vsa vozlišča.

[10] (d) Dokažite pravilnost kateregakoli od zgornjih algoritmov, ki je pravilen, ali predlagajte in dokažite svoj algoritem (navedite tudi časovno in prostorsko zahtevnost).