1	
2	
3	
4	
Σ	

Podatkovne strukture in algoritmi 1: Izpit 1

20. januar 2023

Čas reševanja je 120 minut. Pojasnite vse odgovore.

Sedež (2.03)

Ime in priimek

Vpisna številka

1. naloga (20 točk)

Na zabavo pride n oseb. Oseba je *superzvezda*, če jo poznajo vsi ostali na zabavi, medtem ko superzvezda ne pozna nobenega.

a) (3 točke) Pokaži, da na zabavi obstaja največ ena superzvezda.

Protislovje.

b) (17 točk) Na voljo imamo metodo `pozna(a,b)`, ki vrne `true`, v kolikor oseba a pozna osebo b, in `false` sicer. Opiši algoritem, ki ugotovi, če na zabavi obstaja superzvezda. Za vse točke mora algoritem delovati v $O(n)$

$O(n^2)$: preverimo za vsakega, ali je superzvezda.
Dve for zanki $\Rightarrow O(n^2)$

$O(n\log)$: Razdelimo na pol in rekurzivno rešimo.
Dobimo 2 kandidata (en levi, en desni)
in še preverimo ali sta superzvezdi v $O(n)$.
$\Rightarrow T(n) = 2T(n/2) + O(n)$
$\Rightarrow T(n) = O(n\log n)$

$O(n)$: Damo vse osebe na sklad S.
Dokler $|S| \geq 2$ jemlemo iz sklada po dve osebi in preverjamo ali se poznata.
V vsakem primeru lahko vsaj eno osebo zavržemo kot super zvezdo.
Če na skladu ostane ena oseba, preverimo v $O(n)$, če je res super zvezda.

2. naloga (20 točk)

V nekem podjetju so zadolženi za testiranje posebnih čipov. V pomoč jim je posebno vezje, kamor priklopijo dva izmed čipov ter vsak čip pove za drugega, ali je dober ali slab. Dobri čipi vedno vrnejo pravilen odgovor, medtem ko slabim čipom ne moremo zaupati (lahko povejo bodisi pravilno bodisi narobe).

a) (2 točki) Vsi možno rezultati enega testa čipov A in B so zapisani v spodnji tabeli. Izpolni zadnji stolpec.

	čip A vrne	čip B vrne	Kaj lahko sklepamo o A in B?
(D,D)	B je dober	A je dober	Oba dobra ali oba slaba
(D,S)	B je dober	A je slab	A slab
(S,D)	B je slab	A je dober	B slab
(S,S)	B je slab	A je slab	vsaj en slab

} vedno vsaj en slab

b) (15 točk) Izmed n čipov za katere vemo, da je vsaj $\frac{n}{2}$ dobrih želimo najti enega, ki je zagotovo dober. Opiši postopek, ki najde dober čip.

Namig: Z uporabno največ $\lfloor \frac{n}{2} \rfloor$ prevedi zgornji problem na problem velikosti približno $\frac{n}{2}$. Za začetek lahko tudi predpostaviš $n = 2^k$ ter nato posplošiš na poljuben n.

Razdelimo čipe v pare ($\lfloor \frac{n}{2} \rfloor$ parov, če n lih ostane en brez para) in jih testiramo.

Če je rezultat:

1) (D,D) odstranimo enega iz para

2) sicer odstranimo oba (lahko tudi specificiraš bolj natančno ampak ni treba)!

Če je n sod nam je ostalo zagotovo več dobrih kot slabih čipov in lahko nadaljujemo.

Če je n lih moramo biti pozorni na tisti čip, ki ostane netestiran.

Če je sodo število rezultatov tipa (D, D) moramo netestiran čip nesti naprej, drugače ga lahko pustimo.

c) (3 točke) Izmed n čipov za katere vemo, da je vsaj $\frac{n}{2}$ dobrih poišči vse tiste čipe, ki so dobri v času $\Theta(n)$.

Po točki b) imamo $T(n) = T(n/2) + O(n)$

$\Rightarrow T(n) = O(n)$.

ko enkrat imamo dober čip lahko potestiramo vse ostale v $\Theta(n)$.

Zakaj tega niste reševali?

3. naloga (20 točk)

Imamo n letališč oštevilčenih z $\{0, 1, \ldots, n-1\}$ ter metodo `odhodni_leti(letalisce, cas)`. Vsaka povezava je oblike `(začetno_letališče, čas_odhoda, končno_letališče, čas_prihoda)`. Poleg tega so nekatera letališča večja ter zato potrebujemo več časa, da pridemo iz enega konca letališča na drugega. Za vsako letališče `i` imamo torej podan še najmanjši čas postanka na letališču i. Opiši algoritem, ki za podano začetno in končno letališče izračuna najkrajši čas potovanja od začetnega do končnega vozlišča. Kakšna je njegova časovna zahtevnost?

Modificiramo Djikstra, da upošteva še postanke na vozliščih/letališčih.

To najlažje storimo tako, da ceni/času povezave $a \longrightarrow b$ prištejemo postanek[b].

POMEMBNO: Grafa ne moremo zgraditi vnaprej, ker ne vemo izhodnih povezav iz letališča dokler tja ne pridemo v nekem času t!!

Ker nekaj vas je delalo to napako, drugače ste pa to nalogo reševali dobro.

4. naloga (20 točk)

Opiši algoritem, ki izračuna najmanjšo dolžino particije niza S v palindromske podnize. Na primer:

- S = 'aabab' lahko razdelimo v 'aa|bab'. Ker S ni palindrom je torej najkrajša dolžina 2.
- S = 'ababbbabbababa' lahko razdelimo v 'a|babbbbab|b|ababa'. Dolžina je torej 4.

Bonus: Če algoritem deluje v $O(n^2)$, dobiš dodatnih 8 točk.

$\underline{O(n^3)}$: $n = \text{len}(S)$

$dp[i][j]$ = najmanjša dolžina particije niza $S[i:j]$ v palindromske podnize.

$$dp[i][j] = \begin{cases} 1 \ ; & S[i:j] \text{ palindrom} \\ \min\limits_{k=i,\dots,j-1} \{dp[i][k] + dp[k+1][j]\} \end{cases}$$

razdelimo nekje vmes in vzamemo min

$dp[i][i] = 1$

Imamo n^2 stanj in $O(n)$ za izračun stanja $\Rightarrow O(n^3)$.

$O(n^2)$:

Najprej izračunamo $P[i][j] = \begin{cases} \text{true} \ ; & S[i:j] \text{ palindrom} \\ \text{false} \ ; & \text{sicer} \end{cases}$

v $O(n^2)$ to naredimo recimo tako:

$P[i][i] = \text{true}$

$P[i][j] = (S[i] == S[j]) \text{ AND } P[i+1][j-1]$

n^2 stanj, $O(1)$ za stanje $\Rightarrow O(n^2)$.

Sedaj:

$dp[i]$ = najmanjša dolžina particije niza $S[:i]$ v palindromske podnize.

$$dp[i] = \begin{cases} 1 & ; \ P[0][i] = \text{true} \\ \min\limits_{\substack{j=0,\dots,i-1 \\ \text{and } P[j+1][i] = \text{true}}} dp[j] + 1 \end{cases}$$