Opisi algoritama

Zadatke, testne primjere i rješenja pripremili: Adrian Beker, Marin Kišić, Daniel Paleka, Ivan Paljak, Tonko Sabolčec, Bojan Štetić i Paula Vidas. Primjeri implementiranih rješenja su dani u priloženim izvornim kodovima.

Zadatak: Ogledalo

Pripremio: Marin Kišić
Potrebno znanje: naredbe učitavanja i ispisivanja, rad s riječima

Možemo primijetiti da će se uvijek unositi četiri riječi i da je potrebno ispisati zadnju riječ bez zadnjeg znaka. Pola bodova moglo se osvojiti ispisivanjem podriječi koja se proteže od 18. do 22. znaka rečenice iz ulaza.

Programski kod (pisan u ***Python 3****):*

```
print(input().split(' ')[-1][:-1])
```

Zadatak: Pastiri

Pripremili: Daniel Paleka, Adrian Beker
Potrebno znanje: grafovi, pretraga u širinu/dubinu (BFS/DFS)

Neka je V skup čvorova stabla. U opisu rješenja poistovjećivat ćemo ovcu/pastira i čvor u kojem se nalazi. Primijetimo najprije da zadatak možemo promatrati kao instancu takozvanog set cover problema. Zaista, ako svakom čvoru $v \in V$ pridružimo skup S_v ovaca koje čuva pastir u v, tada je potrebno naći najmanji podskup $P \subseteq V$ takav da je $\bigcup_{v \in P} S_v$ cijeli skup ovaca. Iako je generalna verzija ovog problema NP-potpuna, specifična struktura dotičnog slučaja omogućit će nam da ga riješimo u odgovarajućoj složenosti.

Krenimo od prvog podzadatka, odnosno slučaja kada je zadano stablo lanac. Tada pastir može čuvati samo prvu ovcu lijevo/desno od sebe, stoga se familija $\{S_v \mid v \in V\}$ sastoji od sljedećih skupova:

- $\{x\}$ za svaku ovcu x;
- $\{x, y\}$ za susjedne ovce x, y takve da su x, y iste parnosti.

Nameće se sljedeći pohlepni algoritam – pogledamo prvu ovcu, u slučaju da je druga ovca iste parnosti, postavimo pastira na pola puta između njih, u suprotnom ga postavimo u čvor prve ovce. Nakon toga uklonimo pokrivene ovce te ponavljamo algoritam. Opisano rješenje ima složenost $\mathcal{O}(N + K)$.

U drugom podzadatku, podskupove ovaca predstavljat ćemo bitmaskama, odnosno brojevima iz skupa $\{0, 1, \ldots, 2^K - 1\}$. Na početku pustimo BFS/DFS iz svake ovce te na taj način u složenosti $\mathcal{O}(K \cdot N)$ odredimo skup S_v za svaki čvor v. Dalje zadatak rješavamo dinamičkim programiranjem. Za svaku bitmasku $mask$ neka $f(mask)$ označava minimalan broj skupova S_v čija unija sadrži $mask$. Iteriramo kroz stanja $mask$ u rastućem poretku. U prijelazu iteriramo kroz sve podmaske $submask$ te ukoliko $submask$ odgovara nekom od skupova S_v, osvježavamo $f(mask)$ vrijednošću $f(mask \wedge submask) + 1$ (ovdje $\wedge$ označava bitovno isključivo ili). Nakon toga preostaje za sve podmaske $submask$ osvježiti $f(submask)$ vrijednošću $f(mask)$. Memorijska je složenost $\mathcal{O}(N + 2^K)$, a vremenska $\mathcal{O}(K \cdot N + 3^K)$ (dokaz ove standardne činjenice ostavljamo čitateljici za vježbu).

Za preostale podzadatke, ukorijenimo stablo u proizvoljnom čvoru. Za svaku ovcu x, njenim teritorijem zvat ćemo skup $\{v \in V \mid x \in S_v\}$. Za dvije ovce x, y reći ćemo da su prijatelji ako njihovi teritoriji imaju neprazan presjek. Ideja je pohlepno postavljati pastira koji čuva neku ovcu i sve njene dosad nepokrivene prijatelje. To će nam omogućiti sljedeća tvrdnja:

Tvrdnja 1. Za neku ovcu x, neka je $a(x)$ njen najviši predak koji se nalazi u njenom teritoriju. Tada $S_{a(x)}$ sadrži sve prijatelje od x koji nisu dublji od x.

Dokaz. Neka je ovca y prijatelj od x koji nije dublji od x. Tada možemo pretpostaviti da se y ne nalazi u podstablu od $a(x)$ jer u suprotnom je tvrdnja očita. Uzmimo čvor z koji se nalazi u teritorijima od x i y te neka je čvor w polovište puta od x do y. Tada je

$$d(z, x) = d(z, y) = d(z, w) + d(w, x) = d(z, w) + d(w, y),$$

gdje $d(u, v)$ označava udaljenost čvorova u, v. Također, za bilo koju ovcu t vrijedi

$$d(z, w) + d(w, x) = d(z, x) \leq d(z, t) \leq d(z, w) + d(w, t),$$

odakle slijedi $d(w, x) \leq d(w, t)$, a analogno se pokazuje $d(w, y) \leq d(w, t)$. Dakle, w se nalazi u presjeku teritorija od x i y. Štoviše, budući da y nije dublja od x, w je predak od x, stoga se nalazi na putu od x do $a(x)$. Budući da y nije u podstablu od $a(x)$, vrijedi

$$d(a(x), y) \leq d(w, y) = d(w, x) \leq d(a(x), x),$$

stoga pastir u $a(x)$ čuva y, što je i trebalo dokazati.

Prema Tvrdnji 1, sljedeći je algoritam ispravan:

- Dok god nisu sve ovce pokrivene ponavljaj:
  - Postavi pastira u $a(x)$, gdje je x najdublja dosad nepokrivena ovca.

Direktna implementacija ovog algoritma ima složenost $\mathcal{O}(N(N+K))$ te je dovoljna za ostvariti sve bodove na trećem podzadatku. Za četvrti podzadatak potrebno je ubrzati opisani algoritam. Za svaki čvor v, neka $dep(v)$, $dist(v)$ redom označavaju njegovu dubinu i udaljenost do najbliže ovce. Na početku možemo izračunati $dist(v)$ pomoću BFS-a iz svih ovaca. Alternativno, možemo zamisliti da smo stablu dodali dummy čvor povezan sa svim ovcama i iz njega pustili BFS. Sljedeća obzervacija sada nam omogućuje da efikasno odredimo čvor $a(x)$ za svaku ovcu x:

Opservacija 2. Ako je čvor v predak ovce x, tada je $dist(v) \leq dep(x) - dep(v)$, i jednakost vrijedi ako i samo ako je v u teritoriju ovce x.

Prema Obzervaciji 2, $a(x)$ je pozicija prvog pojavljivanja maksimuma od $dist(v) + dep(v)$ po svim čvorovima v na putu od korijena do x. Stoga je $a(x)$ za svaku ovcu x moguće izračunati jednostavnim DFS-om iz korijena. Konačno, preostaje efikasno održavati najdublju dosad nepokrivenu ovcu. Ako sortiramo ovce padajuće po dubini, taj se problem svodi na održavanje pokrivenih ovaca. U tu svrhu za početni BFS promotrimo pripadajući graf najkraćih putova. To je usmjeren graf G sa skupom vrhova V te bridovima (u, v) gdje je $\{u, v\}$ brid stabla takav da je $dist(v) = dist(u) + 1$.

Opservacija 3. Za svaki čvor $v \in V$, S_v je skup ovaca x takvih da postoji put od x do v u G.

Kad god postavimo novog pastira, proširimo se DFS-om iz njega unatrag po grafu G te pritom pazimo da obilazimo samo dosad neposjećene čvorove. Prema Obzervaciji 3, ovca je pokrivena ako i samo smo ju posjetili u DFS-u. Budući da svaki čvor posjećujemo najviše jednom, održavanje pokrivenih ovaca ima ukupno linearnu složenost.

Na kraju, ukupna je složenost $\mathcal{O}(N + K \log K)$. Primijetimo još da se faktor $\log K$ pojavljuje isključivo zbog sortiranja ovaca po dubini, što je naravno moguće izvesti i u složenosti $\mathcal{O}(N + K)$ jer dubine ne prelaze N. Stoga postoji rješenje i u linearnoj složenosti. Za implementacijske detalje pogledajte službene kodove.

Zadatak: Ulica

Pripremio: Marin Kišić
Potrebno znanje: petlje, provjera parnosti

Napravit ćemo pomoćno polje `moze` i za svaki broj x iz ulaza ćemo postaviti `moze[x]=1`. Nakon toga, prebrojimo ima li više parnih ili neparnih brojeva u u ulazu. Na kraju, ako ima više parnih, krenemo od 2 i idemo tako dugo dok je `moze[trenutni_broj]=1` krećući se po parnim brojevima. Analogno za neparne, samo počinjemo od 1.

Zadatak: Datum

Pripremio: Karlo Franić
Potrebno znanje: provjera palindromičnosti, ad-hoc

Za prvi podzadatak potrebno je uzeti prva dva znaka datuma i taj broj povećavati za 1 dok ne nađemo palindrom.

Za drugi podzadatak koristimo znanje prvog podzadatka, uz to svaki put kada dan dođe do zadnjeg u mjesecu (4. i 5. znak), postavimo dan na 1, a mjesec povećamo za 1.

Za treći podzadatak potrebno je povećavati i godinu svaki put kad dođemo do 31. dana u 12. mjesecu.

Za sve bodove bilo je potrebno primijetiti da palindromičnih datuma postoji 366. Svaki datum ima točno jednu godinu s kojom tvori palindrom. Te datume je moguće staviti u polje i nakon toga za svaki zadani datum iterativno po polju naći prvi datum koji je veći od zadanog.

Vremenska složenost je $\mathcal{O}(NK)$, gdje K predstavlja broj palindromičnih datuma. Zadatak se može riješiti i u složenosti $\mathcal{O}(N \log K)$, a to rješenje ostavljamo čitateljici za vježbu.

Zadatak: Zagrade

Pripremila: Paula Vidas
Potrebno znanje: stog ???

Prvo ćemo opisati rješenje u slučaju da je cijela lozinka validan niz. U prvom podzadatku je $Q = \frac{N^2}{4}$, pa možemo postaviti upit za svaki interval parne duljine (primjetimo da nema smisla pitati za intervale neparne duljine). Jedno moguće rješenje je sljedeće: pitamo za prva dva znaka, prva četiri znaka, itd. sve dok ne dobijemo pozitivan odgovor. Tada znamo na kojem je mjestu zatvorena zagrada koja je uparena s otvorenom zagradom na prvom mjestu. Sada rekurzivno riješimo isti zadatak na intervalu između tih zagrada i na intervalu desno od zatvorene zagrade.

U trećem podzadatku možemo postaviti $Q = N - 1$ upita. Koristit ćemo stog, koji je na početku prazan. Idemo redom po pozicijama lozinke. Ako je stog prazan, stavimo trenutnu poziciju na stog. Inače, postavimo upit za interval između pozicije koja je na vrhu stoga i trenutne pozicije. Ako je odgovor pozitivan, zagrade na tim pozicijama su uparene, te obrišemo vrh stoga. U suprotnom stavimo trenutnu poziciju na stog.

Što kada cijela lozinka nije validan niz? Algoritam je isti, ali na kraju stog ne mora ostati prazan. Od pozicija koje su ostale na stogu, na prvih (manjih) pola mora biti zatvorena, a na drugih pola otvorena zagrada.