Opisi algoritama

IBT 2011. MASTERS LIGA - 3. kolo

Komentare i pitanja uputite na askurdija@gmail.com *ili na* frane.kurtovic@gmail.com.

Monoton (P1-1)

Prolazom po nizu *for* petljom utrdit ćemo je li se negdje dogodio *rast* (element veći od prethodnog), je li se negdje dogodio *pad* (element manji od prethodnog) i je li se negdje dogodila *stagnacija* (element jednak prethodnom).

Ako se dogodilo i rast i pad, niz nije monoton. Inače, ako se dogodio rast i stagnacija, niz je rastući. Inače, ako se dogodio pad i stagnacija, niz je padajući. Inače se dogodio ili samo rast (pa je niz strogo rastući) ili samo pad (pa je strogo padajući).

Palindrom (P1-2, P2-1)

Prolaskom po danom stringu, izbrojimo za svako slovo od a do z koliko puta se pojavilo.

Da bi riječ mogla biti palindrom, svako slovo u njoj mora se pojaviti paran broj puta, osim možda jednog (onog u sredini). Ako dakle imamo dva ili više slova koji se javljaju neparan broj puta, sigurno ne možemo složiti palindrom, a u suprotnom možemo (dokaz je jednostavan).

Povratak (P1-3, P2-2)

Treba zaključiti da su sumnjivi brojevi zapravo duljine ciklusa u grafu koji je implicitno zadan (iz svakog vrha izlazi jedan usmjereni brid u neki drugi vrh). Algoritam nalaženja ciklusa, a da radi dovoljno brzo - u složenosti *O(N)*, ide kako slijedi.

For petljom prolazimo po vrhovima grafa (bilo kojim redoslijedom - recimo, od 1 do N) i od svakog vrha, ako je još neposjećen, započinjemo *putovanje*: krećemo iz njega dok se ne dogodi jedna od sljedećih stvari:

- Došli smo do vrha posjećenog u nekom ranijem putovanju. To znači da vrhovi kojima smo prolazili u ovom putovanju ne pripadaju ciklusu.
- Došli smo do vrha posjećenog u ovom putovanju. Tada on pripada ciklusu, pa još jednim prolaskom od tog vrha do njega samoga nalazimo duljinu tog ciklusa.

Za implementaciju ovog algoritma ključan je pomoćni niz koji za svaki vrh pamti redni broj posljednjeg putovanja u kojem je taj vrh posjećen.

Reakcija (P1-4, P2-3)

Podijelimo danu podlogu na N ploča dimenzija 2 x 2. Pokazat ćemo da je u svakoj takvoj ploči moguće upisati sva četiri slova A, B, C, D što znači da će ukupan broj pojavljivanja svakog slova biti točno N, što se i traži.

Ploče 2 x 2 popunjavamo redom slijeva na desno. U svakoj su gornja dva slova već upisana (i ona su različita) - neka su to recimo AB. Tada za donja dva stavljamo CD ili DC, već prema tome koje je bilo posljednje slovo u prethodnoj ploči (ako je primjerice bilo C, ne možemo sada staviti CD nego moramo DC).

Wordplay (P2-4)

Izdvojimo sve riječi duljine $K-1$ i primijetimo da svaka od danih riječi duljine K povezuje dvije riječi duljine $K-1$. Zamislimo sada graf u kojem su vrhovi riječi duljine $K-1$, a dane riječi duljine K su usmjereni bridovi u tom grafu.

Ideja za takav prikaz problema dolazi iz činjenice da dvije riječi duljine K, koje u orginalnoj velikoj riječi slijede jedna za drugom, imaju zajednički dio duljine $K-1$. To znači da dva brida u grafu koji pripadaju tim dvama riječima dijele zajednički vrh, tj. tvore put duljine 2. Analogno, svih $N-K+1$ danih riječi duljine K, ako su u ispravnom poretku s obzirom na orginalnu veliku riječ, tvore u opisanom grafu put duljine $N-K+1$ koji točno jednom prolazi svakim bridom. (Pritom mogu postojati i višestruki bridovi, ako se neke riječi duljine K u inputu javljaju više puta.)

Treba dakle naći Eulerov put u dobivenom grafu, a za to možemo koristiti neki od poznatih algoritama za nalaženje takvog puta.

Okreni (P1-5)

Zahvaljujemo Matiji Buciću za pomoć pri izradi i rješenju ovog zadatka.

Za $N = 2$ rješavamo ručno, a za $N \geq 3$ provodimo sljedeći algoritam. Ako je N paran, zamjenimo N sa svima ostalima i rješavamo problem za $N-1$, koji je neparan, na sljedeći način.

Neparan $N = 2k+1$ rješavamo tako da najprije napravimo operaciju $(2, k+1, 2k+1)$. Potom ponavljamo operacije kojima brojeve i i $2k+1-i$ zajedno selimo između $i-1$ i $2k+2-i$ operacijom $(i+1, i+k, i+k+2)$. Na taj način, jednom operacijom stavljamo dva broja na mjesto, pa za N brojeva možemo riješiti problem pomoću $N\text{div } 2+1$ operacija. Moguće je (ne tako lako) dokazati da je ovo i najmanji mogući broj operacija.

Za ilustraciju pogledajmo kako se niz transformira za $N = 7$:
7 6 5 4 3 2 1
7 3 2 1 6 5 4
1 6 7 3 2 5 4
1 2 5 6 7 3 4
1 2 3 4 5 6 7

Lego (P2-5)

Ključna ideja je pronaći karakteristiku svih pobjedničkih odnosno gubitničkih pozicija. (Pobjednička pozicija je ona za koju igrač na potezu ima pobjedničku strategiju; inače se radi o gubitničkoj poziciji.) To je pak gotovo nemoguće pronaći ako se ne prouči dovoljan broj malih primjera; za njihovo generiranje dovoljan je rekurzivni program koji radi za 30% službenih test podataka.

A zaključak je sljedeći: ako je broj najviših stupaca neparan, pozicija je pobjednička, a inače je gubitnička. To je moguće i dokazati, na sljedeći način. Primijetimo najprije da iz pozicije sa parno mnogo najviših stupaca nužno prelazimo u poziciju sa neparno mnogo najviših stupaca (točnije, jedan manje): to znači da igrač koji se nalazi u gubitničkoj poziciji nužno ostavlja svome protivniku pobjedničku poziciju. Još treba dokazati da iz pobjedničke pozicije uvijek možemo prijeći u gubitničku, a dokaz će slijediti iz algoritma koji ćemo opisati.

Za paran broj stupaca treba dakle odmah ispisati 0. Ako je broj najviših stupaca neparan, a veći od 1, onda smanjivanjem bilo kojeg od njih na bilo koju visinu ostavljamo protivnika u

gubitničkoj poziciji jer preostaje paran broj najviših stupaca. Rješenje je u tom slučaju dakle jednako broju najviših stupaca pomnoženom s brojem mogućih visina na koje možemo spustiti takav stupac (a taj broj jednak je upravo visini stupca).

Preostaje nam slučaj kad postoji samo jedan najviši stupac. Neka je k broj sljedećih najviših stupaca (tj. onih koji postaju najviši ako maknemo promatrani stupac) i neka je h njihova visina. Spuštanjem najvišeg stupca na visinu veću od h ostavljamo protivnika u pobjedničkoj poziciji jer mu ostaje opet neparno mnogo (tj. opet jedan) najviših stupaca, pa takve poteze odbacujemo.

Ako je k paran, spuštanjem najvišeg stupca na visinu h ostavljamo protivnika u pobjedničkoj poziciji jer mu ostaje $k+1$ najviših stupaca, što je neparan broj. Moramo dakle smanjiti najviši stupac na neku od visina od 0 do $h-1$, što će biti dobro jer će tada protivnik ostati u gubitničkoj poziciji (imat će k, dakle parno mnogo, najviših stupaca). Za ovaj slučaj stoga ispisujemo h načina.

Ako je k neparan, spuštanjem najvišeg stupca na visinu manju od h ostavlja protivnika u pobjedničkoj poziciji (sa neparno mnogo, točnije k, najviših stupaca), pa moramo spustiti najviši stupac na visinu h, što ostavlja protivnika u gubitničkoj poziciji (sa $k+1$, što je parno, najviših stupaca). Za ovaj slučaj stoga ispisujemo 1 način.