

Opisi algoritama

Primjeri implementiranih rješenja dani su u priloženim izvornim kodovima koji ne odgovaraju nužno u svim detaljima ovdje opisanim algoritmima.

Eventualna pitanja, mišljenja ili prijedloge vezane uz zadatke uputite na jklepec@gmail.com.

Zadatak: Magija

Pokušajmo analizirati u kojem slučaju David i Filip mogu spasiti sve jelene bez teleportacija. Primjećujemo da ako postoje dva jelena koja na livadu dolaze u istom trenutku barem jednog (onog koji odlazi prije) neće biti moguće spasiti. Dakle, da bi David i Filip imali šansu spasiti sve jelene, svi moraju dolaziti na livadu u različitim trenucima.

Srećom, ispada da obrnuta tvrdnja također vrijedi. Tj. ako svi jeleni dolaze u različitim trenucima, sve ih možemo spasiti (možemo ih sve spasiti odmah u trenutku njihovog pristizanja).

Dakle, potrebno je odgonetnuti koliko jelena treba teleportirati tako da nakon toga svi preostali pristižu u različitim vremenima. Jedan od načina je za svako početno vrijeme prebrojati koliko jelena pristiže u tom trenutku. Recimo da je taj broj x , sigurno moramo teleportirati nekih $x - 1$ jelena (ispada da nije bitno kojih). Ako to napravimo za svako početno vrijeme dobijamo traženo rješenje.

Implementaciju je moguće napraviti pomoću polja u kojem za svako početno vrijeme pamtimo broj jelena koji pristižu u tom trenutku. Moguće su i druge implementacije.

Zadatak: Bitflip

Pokušajmo osmisлити algoritam koji može odgonetnuti može li se pločica počistiti u **jednom** koraku. Pretpostavimo za sada da znamo kako to napraviti (biti će opisano kasnije).

Recimo sada da smo uspjeli odgonetnuti možemo li pločicu počistiti u jednom koraku. U slučaju da je odgovor negativan, trebamo još odgonetnuti možemo li pločicu počistiti u dva koraka, ako je odgovor i tada negativan onda je rješenje tri.

Da odgonetnemo možemo li pločicu počistiti u dva koraka za prvi korak možemo isprobati sve mogućnosti te na pločici koju dobijemo nakon tog koraka pokrenemo algoritam koji provjerava počistivost u jednom koraku.

Broj bodova koje dobijamo ovisi o efikasnosti algoritma koji provjerava možemo li pločicu počistiti u jednom koraku.

Preostaje opisati kako implementirati algoritam u kojem provjeravamo možemo li počistiti pločicu u jednom koraku. Možemo isprobati sve moguće poteze slično kao i u prvom koraku prethodnog algoritma, ali u tom slučaju nećemo dobiti bodove za maksimalna ograničenja jer je složenost tog postupka $\mathcal{O}((N * M)^2)$.

Algoritama za računanje navedenog u jednom prolazu matrice ima puno. Opisati ćemo jedan od njih. Možemo primjerice pronaći najlijeviji, najgornji bit te najdesniji najdonji bit i provjeriti jesu li svi bitovi između upaljeni te jesu li to jedini bitovi u matrici.

Zadatak je također moguće riješiti rekurzijom (pogledajte opis zadatka tablica). Kod koji je priložen koristi rekurziju jer je takav pristup najkorisniji za naučiti nešto iz nejga.

Zadatak: Predrasude

Primjetimo da ako neko osoba govori istinu, obje osobe pored nje moraju lagati. Također, ako osoba laže barem jedna osoba kraj nje mora govoriti istinu.

To nas prvo dovodi to zaključka da ne možemo imati dvije osobe koje govore istinu jednu do druge. Ako

to nije slučaj uvijek možemo konstruirati rješenje. Primjerice, $I??...??I$ možemo ispuniti kao **ILILIL...ILI** ili **ILILIL...ILLI**, ovisno o parnosti.

Dakle dovoljno je ulazni podatak razbiti na segmente u kojima su sami upitnici te na rješenje dodati maksimalan broj istina koji možemo staviti ($(\text{veličina} - 1)/2$ zaokruženo na dolje).

Za vježbu: zadatak se može riješiti i kada bi u ulaznim podacima bili i znakovi 'L'. Pogledaj zadatak Izjave ili priloženi kod.

Zadatak: Krivolov

Zadatak je zapravo efikasno izvesti simulaciju. Očito je da kada god možemo spasiti nekog jelena, isplati se učiniti isto.

U dizajnu algoritma pomaže prvo razmišljati u jeziku apstraktnih struktura, a o implementaciji kasnije. Ono što želimo je imati skup svih jelena koji će dolaziti na livadu te potom minutu po minutu raditi provjeru. Provjera će biti sljedećeg oblika: "Je li u ovoj minuti u našem skupu samo jedan jelen koji bi tada pristigao na livadu?". Ako je odgovor potvrđan, izbacujemo tog jelena iz skupa te nastavljamo dalje.

Navedeni algoritam može se implementirati na mnogo načina, no u ovom zadatku nije se tražila efikasnost. Bilo kakva implementacija bit će dovoljno dobra. Možemo naš skup simulirati sa poljem, sa stl::set strukturom, vektorom te mnogim drugim načinima.

Zadatak: Izjave

Primjetimo da ako neko osoba govori istinu, obje osobe pored nje moraju lagati. Također, ako osoba laže barem jedna osoba kraj nje mora govoriti istinu.

To nas dovodi do zaključka da oko osobe koja govori istinu moraju biti lažovi te da ne smijemo imati tri lažova zaredom.

Možemo ponavljati sljedeće sve dok ne dođemo do kontradikcija ili ne dobijamo nove informacije:

- Ako se negdje nalazi 'I' oko njega moraju biti 'L' (u slučaju kontradikcije prekidamo)
- Ako se negdje nalaze dva 'L' zaredom ("LL") oko njih moraju biti istine.

Kada navedeni postupak završi ostatak će nam odvojeni uzastopni nizovi znakova 'I' koji na oba kraja imaju znakove 'L' i to takve da nisu dva 'L' zaredom (jer u suprotnom proces ne bi još završio). No sada je lako odgonetnuti koliko najviše istina možemo staviti u "L???...???L". Jednostavno pohlepno stavljamo što je više moguće istina s lijeva na desno. (Odgovor je $(\text{veličina} + 1)/2$ zaokruženo na dolje)

Treba biti pažljiv kako odgonetnuti uzastopne nizove upitnika jer se radi o cikličnom nizu.

Zadatak: Tablica

Zadatak ćemo riješiti rekurzijom. Naivno isprobavanje svih mogućih postavljanja 5 poteza rezultira programom koji se izvršava duže od 5 sekundi, no možemo i na taj način dobiti neke bodove.

Poboljšanje koje možemo napraviti je to da nekako ne isprobavamo bespotrebne poteze. Primjerice, ako promatramo najlijeviye najgornje polje koje je obojano u crno kroz njega mora prolaziti barem jedan potez. Također možemo pretpostaviti da je to polje početak nekog poteza jer u suprotnom možemo sve poteze koji idu lijevo-gore od tog polja izbrisati ili skratiti.

Možemo se rekurzivno proširiti probavši napraviti sve moguće poteze s tim poljem kao početkom i tako ponavljati sve do dubine 5 kada samo treba provjeriti je li tablica prazna ili nije.