

## HIPERPROSTOR

Nacrtajmo usmjereni težinski graf kojemu su čvorovi planete, a bridovi veze između planeta. Za riješiti podzadatak s manjim test primjerima bio je dovoljan slijedeći algoritam:

Primjetimo da će najkraći put sadržavati najviše 29 bridova jer je broj čvorova  $\leq 30$ . Kada bismo prošli više od 29 bridova onda bi neki čvor obišli više puta, a to sigurno nije optimalno. Nadalje, budući da je cijena svakog brida manja od 50, onda je dužina najkraćeg puta koji ne koristi hiperveze manja od  $50 \cdot 29 (=1450)$ . Sada možemo isprobati za hipervezu sve težine do 1450 te pronaći najkraći put Dijkstrinim algoritmom.

Kako bi dobili još 25% bodova možemo pronaći broj različitih vrijednosti koji može poprimiti najkraći put i u većim test primjerima. Binarnim pretraživanjima nađimo broj  $K$ , najveću cijenu trajanja prolaska hipervezom za koju je najkraći put manji nego najkraći put koji ne koristi hiperveze. Sada je broj različitih vrijednosti koje može poprimiti put jednak  $K + 1$ .

Za rješenje koje dobiva sve bodove konstruirajmo drukčiji graf. Svaki čvor je opisan s planetom i brojem hiperveza koje smo do sad prošli na putu do te planete (u daljnjem opisu rješenja prikazivat ćemo svaki čvor kao uređeni par). Za svaku običnu vezu koja povezuje planetu  $A$  i  $B$  dodajmo u grafu vezu između čvorova  $(A, i)$  i  $(B, i)$  za  $i$  od 0 do  $N-1$ , a za svaku hipervezu dodajmo vezu između čvorova  $(A, i)$  i  $(B, i+1)$  za  $i$  od 0 do  $N-2$ .

Označimo s  $d(b, k)$  najkraći put do čvora  $(b, k)$  ne računajući trajanje hiperveza. Primijetimo da smo dobili skup pravaca  $k \cdot x + d(b, k)$ , za  $k$  od 0 do  $N-1$ , gdje je  $x$  duljina trajanja hiperveze. Za svaki pravac nađimo interval na kojemu taj pravac daje najkraći put. To možemo riješiti tako da pronađemo presjeke svih pravaca i zatim konveksnu ljusku koju čine. Konveksna ljuska bit će sačinjena od dijelova pravaca s kojima je pojedinačno lako računati razne podatke koji se od nas traže.

## ROTACIJE

Svedimo zadatak na zasebne podprobleme koji će nam pomoći pri rješavanju zadatka.

a) Ako je  $R \cdot S \leq 9$

Broj mogućih stanja ploče je 9! te možemo BFS ili DFS algoritmom obići sva stanja pamteći ih pomoću hash tablice.

Rekurzivna implementacija DFS algoritma relativno je jednostavna, ali se mora paziti na dubinu rekurzije radi memorije. Stoga, dobro je ograničiti maksimalnu dubinu DFSa (recimo na 1000). Ako bismo time slučajno izgubili put do konačnog stanja, vrlo lako možemo pronaći neki drugi put u nekoj drugoj grani ili povećati maksimalnu dubinu.

b) Ako je neka od dimenzija jednaka 2

Valja primjetiti da se svaka  $2 \times 4$  tablica može riješiti. To možemo vidjeti, primjerice, iz prethodno spomenutog DFSa. Uz pomoć te opservacije možemo riješiti  $2 \times N$  tablicu tako da odgovarajuća dva broja stavimo na kraj tablice te postupak ponavljamo dok ne svedemo problem na  $2 \times 4$  tablicu.

Postavljanje ta dva broja možemo također riješiti primjenom DFS algoritma kao u a) dijelu. Nužno je smanjiti broj stanja u DFSu tako da ne razlikujemo brojeve koji nisu ta dva koja trenutno postavljamo.

c)  $R, S \leq 25$

Tablicu možemo rješavati red po red prema dolje sve dok nam ne ostane  $2 \times N$  tablica koju znamo riješiti. Da bismo složili trenutni red, možemo primjeniti DFS algoritam kao i u b) dijelu ili napraviti to ad-hoc. Primjerice, možemo traženi broj prvo odvući do prvog retka u kojem možemo proizvoljno manevrirati bez da remetimo već složeni dio tablice. Zatim broj odvučemo do traženog stupca, a zatim ga spustimo u traženi redak. Međutim, tim pristupom valja biti izrazito oprezan pri postavljanju zadnja dva broja u svakom redu.

## SNJEGULJICA

Potrebno je održavati niz brojeva koji ćemo nazvati gdje: gdje[p] nam govori gdje se u nizu nalazi patuljak p.

Kada patuljci x i y zamjene mjesta dovoljno je zamjeniti gdje[x] i gdje[y].

Za 50 bodova je  $B - A \leq 50$ , zato je za drugu operaciju dovoljno je pogledati gdje se nalaze svi patuljci s oznakama od A do B. To nam govori niz gdje[p].

Konkretnije, dovoljno je pronaći najlijeviju i najdesniju poziciju među svim patuljcima od A do B i pogledati jesu li one udaljene za točan broj pozicija tako da nema "stranih" patuljaka između.

Dakle, potrebno je odrediti je li  $\max(\text{gdje}[p], \text{za } A \leq p \leq B) - \min(\text{gdje}[p], \text{za } A \leq p \leq B)$  jednako  $B - A$ .

Sada vidimo da nas zanima najveći i najmanji broj niza gdje u intervalu od A do B.

Jedna od struktura koja može brzo odgovoriti na takvo pitanje je Tournament (Interval) stablo.

Naime, u svakom čvoru možemo pamtit minimum i maksimum na odgovarajućem intervalu te dovoljno brzo odrediti promjene.

## ŠPIJUNI

Pretpostavimo najprije da u misiju valja poslati samo jednoga špijuna. Povežemo li sve špijune u stablo, razgovore možemo nanizati tako da sve informacije dođu do jednoga (bilo kojeg) od špijuna. Dovoljno je dakle u misiju poslati najjeftinijeg špijuna, a za pronalazak stabla s najmanjom ukupnom cijenom bridova (razgovora) možemo koristiti jedan od poznatih algoritama za pronalazak minimalnog razapinjućeg stabla -- Primov algoritam ili Kruskalov algoritam.

Ako u misiju šaljemo više špijuna, treba nam više stabala. Međutim, oba algoritma izvorno daju jedno stablo.

Primov algoritam izvorno započinje s jednim, proizvoljnim čvorom u jezgri koju onda iterativno širi, ali možemo ga modificirati tako da kreće s više čvorova u jezgri -- špijunima koje planiramo poslati u misiju. Za 40 bodova, znamo da je u misiju dovoljno poslati do četiri špijuna pa možemo isprobati sve njihove kombinacije i za svaku pokrenuti ovaj modificirani Primov algoritam.

Kruskalov algoritam kreće sa  $N$  stabala i u svakom koraku spaja dva stabla u jedno na najjeftiniji način. Ako su cijene slanja u misiju međusobno jednake, algoritam možemo modificirati tako da prestanemo spajati stabla kada cijena spajanja postane veća od cijene slanja špijuna u misiju. Ovo rješenje nosi 50 bodova.

Za 100 bodova, uočimo da možemo u graf dodati novi čvor -- misiju, koja je sa svakim špijunom povezana bridom čija je težina -- cijena slanja tog špijuna u misiju. Na takvom grafu potrebno je pronaći samo jedno minimalno razapinjuće stablo što možemo izravno Primom ili Kruskalom.