

Županijsko natjecanje iz informatike

Srednja škola

Prva podskupina (1. i 2. razred)

15. veljače 2019.

OPISI ALGORITAMA

Autori zadataka: Mislav Bradač (*Parcela, Spust*), Adrian Satja Kurdija (*Stranice*)

Zadatke, rješenja i test podatke pripremili: Mislav Bradač, Domagoj Bradač i Adrian Satja Kurdija



Ministarstvo
znanosti i
obrazovanja



Agencija za odgoj i obrazovanje



**HRVATSKI SAVEZ
INFORMATIČARA**

Trokut možemo spremiti kao matricu čiji redak po redak popunjavamo redovima trokuta:

```
1
2 3
4 5 6
7 8 9 ...
```

Brisanje stranice C svodi se na uklanjanje posljednjeg retka, što možemo efikasno izvesti. Brisanje stranice B svodi se na uklanjanje zadnjeg broja iz svakog retka, što također efikasno izvodimo jednim prolazom po redcima.

Brisanje stranice A malo je veći problem jer uklanjanjem prvog broja iz retka moramo pomaknuti sve ostale brojeve za jedno mjesto naprijed, čime bi ukupna vremenska složenost postala prevelika (kubna) jer bismo u najgorem slučaju približno N puta prolazili po cijelom trokutu.

Da bismo izbjegli pomicanje ostalih brojeva, dovoljno je u pomoćnom nizu za svaki red održavati dvije kazaljke - indekse lijevog i desnog kraja onog dijela reda koji je još uvijek "živ" tj. neobrisan, čime se brisanje svodi na jednostavan pomak (povećanje ili smanjenje za jedan) odgovarajuće kazaljke.

Ako bismo nakon svakog brisanja prošli po cijelom trokutu da bismo izračunali zbroj preostalih brojeva, dobivena vremenska složenost bila bi prevelika (kubna). Efikasnije je u pomoćnoj varijabli cijelo vrijeme održavati zbroj brojeva u trokutu i od nje oduzimati brojeve koje brišemo.

Da bismo riješili ovaj zadatak, bilo je potrebno primijetiti da će postojati tražena minimalna parcela takva da svaki pravokutnik koji predstavlja tlocrt zgrade ima barem jedan zajednički vrh s nekim drugim pravokutnikom. Koristeći tu opservaciju generirat ćemo sve moguće rasporede zgrada u kojima svaka zgrada dijeli vrh s nekom drugom zgradom.

Svaki pravokutnik možemo rotirati za 90 stupnjeva ili ne rotirati. Rasporede generiramo tako da za sve kombinacije rotacija pravokutnika (ukupno njih 8), isprobamo sve redoslijede njihovog postavljanja (tzv. permutacije, ukupno njih 6). Za svaku rotaciju i redoslijed postavljanja smjestit ćemo zgrade u koordinatni sustav te izračunati površinu pripadne minimalne parcele. Krećemo od prvog pravokutnika te ga postavljamo tako da mu je donji lijevi vrh u ishodištu. Zatim za drugi pravokutnik odabiremo jedan njegov vrh (na 4 načina) te jedan vrh prvog pravokutnika (na 4 načina) te postavljamo drugi pravokutnik tako da se njegov odabrani vrh poklapa s odabranim vrhom prvog pravokutnika. Nastavljamo slično s trećim pravokutnikom: odabiremo jedan njegov vrh (na 4 načina) te jedan od vrhova prvih dvaju pravokutnika (ukupno 8 načina) te postavljamo pravokutnik tako da se njegov odabrani vrh poklapa s odabranim vrhom jednog od postavljenih pravokutnika. Prije računanja površine parcele potrebno je iz koordinata vrhova pravokutnika provjeriti sijeku li se, te u slučaju da se neka dva pravokutnika sijeku, odbacujemo taj raspored. Inače pronalazimo najdesniji, najljeviiji, najgornji i najdonji vrh te iz njih računamo površinu parcele u koju stanu tako raspoređene zgrade. Površina najmanje parcele tako generiranih rasporeda zgrada tada je rješenje zadatka.

Alternativno se moglo primijetiti da je većina rasporeda koje bismo generirali na prethodno opisan način ili besmisleno (uvijek će se preklapati zgrade, ako se razmjestite na taj način) ili istovjetno nekom drugom rasporedu. Tada je moguće ručno popisati smislene rasporede (nema ih puno!) te provjeriti koji od njih zahtijeva parcelu najmanje površine.

Ako se ne uoči glavna opservacija potrebna za rješavanje cijelog zadatka, moglo se doći do rješenja čija složenost polinomijalno ovisi o dimenzijama zgrada. Jedno od njih je unutar pravokutnika neke dimenzije probati postaviti zgrade na sve moguće koordinate i tako za svaki raspored izračunati najmanju parcelu.

Ovaj zadatak riješit ćemo tako da vrijeme promatramo unatrag, tj. krenut ćemo od dana u kojem se snijeg otopio na cijeloj planini i računati duljinu najduljeg spusta za svaki dan od tog dana do prvog dana skijanja. Duljina najduljeg spusta na i -ti dan jednaka je duljini najduljeg spusta koji završava na jednom od kvadratića čija je visina veća ili jednaka i .

S obzirom da za svaki kvadratić u spustu mora vrijediti da je na većoj visini od idućeg, tada zaključujemo da je duljina najduljeg spusta koji završava nekim kvadratićem jednaka duljini puta u acikličnom grafu od tog kvadratića do njemu najudaljenijeg. Taj problem može se lagano riješiti DFS obilaskom grafa s pomoću rekurzivne funkcije.

Duljinu najduljeg spusta koji završava na nekom kvadratiću visine veće od i ne moramo ponovno računati jer smo ga već izračunali, tj. ono je jednako duljini najduljeg spusta na dan $i+1$. Sada slijedi da je traženo rješenje za i -ti dan jednako maksimumu rješenja za dan $i+1$ te duljine spustova izračunatih na dan i . Ovaj algoritam ima složenost $O(N^2M^2)$ te je osvajao 50% bodova.

Za sve bodove bilo je potrebno primijetiti da nije potrebno raditi DFS obilazak svaki put kada nas zanima najdulji spust do nekog kvadratića, nego možemo prethodno izračunate duljine spustova koristiti u računanju novih. U polju $dp[y][x]$ pamtimo duljinu najduljeg spusta koji završava na pripadnom kvadratiću. Prilikom računanja vrijednosti duljine spusta za kvadratić visine i već ćemo imati zapamćene duljine najduljih spustova koji završavaju na kvadratićima veće visine te je dovoljno od trenutnog kvadratića pogledati vrijednosti polja dp njegovih susjeda veće visine, uzeti najveću vrijednost, uvećati ju za 1 i spremiti izračunatu vrijednost u polje dp . Takvo rješenje osvajalo je sve bodove, složenosti je $O(NM \log NM + NM) = O(NM \log NM)$ te je najsporiji dio algoritma početno sortiranje visina kvadratića.