



## Opisi algoritama

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

### Zadatak: Kraljevstvo

Pripremili: Tonko Sabolčec i Paula Vidas

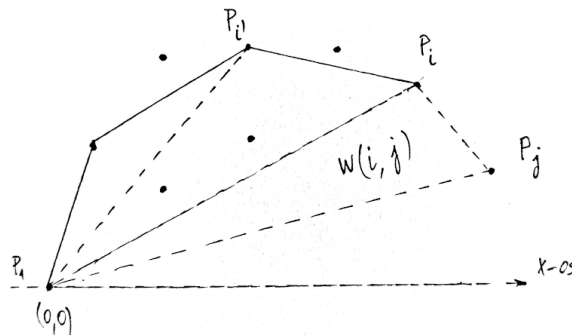
Potrebno znanje: geometrija, konveksna ljuska, dinamičko programiranje, metoda *podijeli pa vladaj*

Za prvi podzadatak dovoljno je za svaki  $K$ -člani podskup dvoraca koji sadrži najljeviji i najdesniji dvorac izračunati površinu konveksne ljuske pripadajućih točaka iz podskupa. Za određivanje konveksne ljuske određenog skupa točaka preporučuje se korištenje *monotone chain* algoritma, dok se površina konveksnog poligona može dobiti podjelom poligona na  $K - 2$  trokuta (npr. koji dijele jedan zajednički vrh) i zbrajanjem površina trokuta dobivenih preko analitičke formule:

$$P(\triangle ABC) = \frac{1}{2} \cdot |x_A(y_B - y_C) + x_B(y_C - y_A) + x_C(y_A - y_B)|.$$

Ostalim podzadacima pristupit ćemo tako da zadani skup točaka podijelimo na one iznad i ispod  $x$ -osi (pri čemu najljeviju i najdesniju točku uzimamo u oba skupa) te odredimo optimalne donje i gornje polovice konačne ljuske. Konkretnije, izračunat ćemo vrijednosti  $up(k)$  i  $down(k)$  za svaki  $2 \leq k \leq K$  gdje  $up(k)$  predstavlja najveću moguću površinu neke ljuske koja se sastoji od  $k$  vrhova gornjeg skupa točaka, dok  $down(k)$  predstavlja analogne vrijednosti za donji skup točaka. Također primijetite da nas zanimaju ljuske koje se sastoje od  $K$  ili *manje* vrhova. Konačno rješenje tada dobivamo pronalaskom najvećeg zbroja  $up(k_1) + down(k_2)$  pri čemu je  $k_1 + k_2 \leq K + 2$  (ova dvojka proizlazi iz činjenice što smo najljeviju i najdesniju točku koristili u oba skupa).

Vrijednosti  $up(k)$  i  $down(k)$  možemo dobiti primjenom dinamičkog programiranja. U nastavku ćemo se usredotočiti samo na gornju polovicu točaka (obrada za donju polovicu ide analogno). Neka su  $P_1, P_2, \dots, P_n$  točke gornje polovice uzlazno sortirane po  $x$ -koordinati ( $P_1$  je najljevija, a  $P_n$  najdesnija). Označimo s  $dp(k, i, j)$  najveću moguću površinu neke konveksne ljuske prvih  $j$  točaka ( $P_1, P_2, \dots, P_j$ ) koja se sastoji od  $k$  vrhova, pri čemu su  $P_i$  i  $P_j$  posljednja dva vrha. Stanje gradimo prijelazima u kojima dodajemo po jednu sljedeću točku gornje ljuske uz pribrajanje pripadajuće površine. U stanje  $dp(k, i, j)$  mogli smo doći iz stanja  $dp(k - 1, i', i)$  uz pribrajanje površine trokuta  $P(\triangle P_1 P_i P_j)$ , koju ćemo označiti s  $w(i, j)$ . Pritom je važno voditi računa o konveksnosti vrhova koje uzimamo, tj. da točke  $P'_i, P_i, P_j$  budu poredane u smjeru kazaljke na satu.



Možemo pisati:

$$dp(k, i, j) = \max_{1 \leq i' < i} \{ dp(k - 1, i', i) + w(i, j) \} \quad \text{t.d.} \quad ccw(P'_i, P_i, P_j) < 0,$$

pri čemu je  $ccw(A, B, C)$  negativna ako su točke  $A, B, C$  poredane u smjeru kazaljke na satu. Konkretna vrijednost  $ccw$  funkcije može se dobiti vektorskim množenjem vektora  $\overrightarrow{AB}$  i  $\overrightarrow{AC}$ :

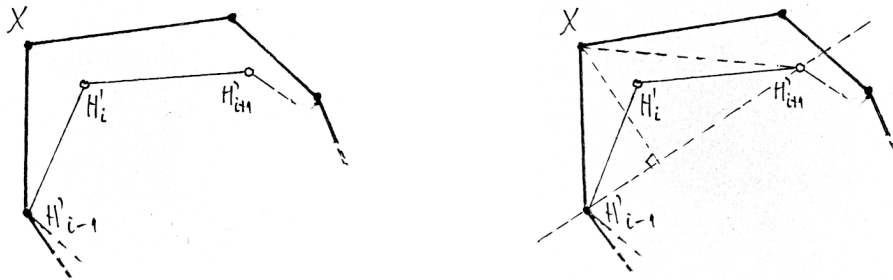
$$ccw(A, B, C) = \overrightarrow{AB} \times \overrightarrow{AC} = (x_B - x_A)(y_C - y_A) - (x_C - x_A)(y_B - y_A).$$



Vrijednosti početnih stanja  $dp(2, 1, i)$  jednake su nuli, a konačna vrijednost  $up(k)$  dobiva se kao  $up(k) = \max_{1 \leq i \leq n} dp(k, i, n)$ . Stanja ima  $\mathcal{O}(K \cdot N^2)$ , a složenost prijelaza je  $\mathcal{O}(N)$  pa je ukupna složenost takvog algoritma  $\mathcal{O}(K \cdot N^3)$ , dovoljna za osvajanje bodova na drugom podzadatku.

**Zamjedba 1.** *Vrhovi optimalne konveksne ljuske bit će podskup vrhova konveksne ljuske svih ulaznih točaka.*

*Skica dokaza.* Neka je  $H$  skup vrhova konveksne ljuske svih ulaznih točaka (točke tog skupa nazvat ćemo *vanjskim* točkama). Pretpostavimo suprotno, tj. da je za određenu veličinu ljuske, optimalna konveksna ljuska  $H' \not\subseteq H$ . Drugim riječima  $H'$  sadrži točku koja nije *vanjska*, neka je to točka  $H'_i$ . (Na slici ispod vanjske su točke popunjene.) Provucimo pravac kroz vrhove susjedne vrhu  $H'_i$  odabrane konveksne ljuske, tj. kroz točke  $H'_{i-1}$  i  $H'_{i+1}$ . Tražimo najudaljeniju točku (nazovimo je  $X$ ) od promatranog pravca s iste strane pravca kao točka  $H'_i$ . Najudaljenija točka  $X$  sigurno je različita od  $H'_i$  te pripada vanjskoj ljusci  $H$  (kad bi  $H'_i$  bila najudaljenija točka, tada bi  $H'_i$  ujedno bila i dio vanjske ljuske, što je suprotno pretpostavci). Primjetite da se zamjenom točke  $H'$  točkom  $X$  povećava površina odabrane konveksne ljuske (što proizlazi iz formule za trokut  $\frac{1}{2} \cdot \text{baza} \cdot \text{visina}$ ). No, to je u kontradikciji s pretpostavkom da je polazna konveksna ljuska bila optimalne površine.  $\square$



Razlog zbog kojeg smo u prethodno opisanoj dinamici pamtili posljednje dvije točke u stanju je taj da omogućimo postupnu izgradnju konveksne ljuske na *proizvoljnom* skupu točaka (treću točku u prijelazu uvijek smo birali tako da se zadovolji smjer kazaljke na satu). No, sada možemo zanemariti sve točke koje nisu vrhovi konveksne ljuske ulaznih točaka, umjesto zadnje dvije točke u stanju možemo pamtili samo posljednju točku. Drugim riječima, stanje dinamike  $dp(k, i)$  predstavlja najveću moguću površinu neke konveksne ljuske koja se sastoji od  $k$  vrhova pri čemu je posljednji vrh točka  $P_i$ . Prijelaz se tada može zapisati kao:

$$dp(k, i) = \max_{1 \leq j < i} \{dp(k-1, j) + w(j, i)\}.$$

Pritom se pretpostavlja da točke  $P_1, P_2, \dots, P_n$  čine vrhove gornje konveksne ljuske. Postoji  $\mathcal{O}(K \cdot N)$  stanja, a složenost prijelaza je  $\mathcal{O}(N)$ , što daje ukupnu vremensku složenost od  $\mathcal{O}(K \cdot N^2)$ , što je dovoljno za treći podzadatak.

**Zamjedba 2.** *Za oznake pozicija  $a < b < c < d$  vrijedi  $w(a, d) + w(b, c) < w(a, c) + w(b, d)$ .*

*Skica dokaza.* Spomenuta nejednakost popularno se naziva *nejednakost četverokuta*, a u našem će slučaju poslužiti kao trik za optimizaciju dinamike. No, obrazložimo za početak tu tvrdnju. Vrijednost  $w(i, j)$  jednaka je:

$$w(i, j) = P(\Delta P_1 P_i P_j) = P((0, 0), P_i, P_j) = \frac{1}{2}(x_j y_i - x_i y_j),$$

gdje je  $P_i = (x_i, y_i)$ . Raspisivanjem, preslagivanjem i sređivanjem dobiva se:

$$w(a, d) + w(b, c) - w(a, c) - w(b, d) = \frac{1}{2}((x_b - x_a)(y_d - y_c) - (y_b - y_a)(x_d - x_c)).$$



Izraz s desne strane odgovara polovici vektorskog umnoška  $\overrightarrow{P_a P_b} \times \overrightarrow{P_c P_d}$ , a budući da su točke  $P_a, P_b, P_c, P_d$  poredane u smjeru kazaljke na satu, ta je vrijednost negativna, tj. vrijedi:

$$w(a, d) + w(b, c) - w(a, c) - w(b, d) < 0 \Rightarrow w(a, d) + w(b, x) < w(a, c) + w(b, d).$$

□

**Zamjedba 3.** Neka je  $p_{k,i}$  oznaka najmanje vrijednosti  $j$  optimalnog prijelaza u  $dp(k, i) = dp(k-1, j) + w(j, i)$ . Tada vrijedi  $p_{k,i+1} \geq p_{k,i}$ .

*Skica dokaza.* Pretpostavimo suprotno, tj. da za neke  $k, i$  vrijedi  $p_{k,i+1} < p_{k,i}$ . Budući da je  $p_{k,i}$  optimalni prijelaz za  $dp(k, i)$ , odnosno  $p_{k,i+1}$  optimalni prijelaz za  $dp(k, i+1)$ , vrijede nejednakosti:

$$\begin{aligned} dp(k-1, p_{k,i}) + w(p_{k,i}, i) &\geq dp(k-1, p_{k,i+1}) + w(p_{k,i+1}, i) \\ dp(k-1, p_{k,i+1}) + w(p_{k,i+1}, i+1) &\geq dp(k-1, p_{k,i}) + w(p_{k,i}, i+1) \end{aligned}$$

Zbrajanjem nejednakosti i sređivanjem dolazi se do:

$$w(p_{k,i}, i) + w(p_{k,i+1}, i+1) \geq w(p_{k,i+1}, i) + w(p_{k,i}, i+1).$$

Po pretpostavci vrijedi  $p(k, i+1) < p(k, i) < i < i+1$ , pa primjenom *Zamjedbe 2* dobivamo:

$$w(p_{k,i}, i) + w(p_{k,i+1}, i+1) < w(p_{k,i+1}, i) + w(p_{k,i}, i+1).$$

što je u kontradikciji s prije dobivenom nejednakosti. Zaključujemo da vrijedi  $p(k, i+1) \geq p(k, i)$ . □

Konačno ćemo dobivene spoznaje iskoristiti za optimiziranje naše dinamike! Pretpostavimo da smo odredili sve vrijednosti  $dp(k-1, i)$  i da na temelju njih želimo izračunati vrijednosti  $dp(k, i)$  za svaki  $1 \leq i \leq n$ . Najprije ćemo izračunati  $dp(k, n/2)$ , tako da prođemo po svim  $1 \leq j < n/2$  i odredimo optimalni prijelaz  $p_{k,n/2}$ . Zbog  $p_{k,i+1} \geq p_{k,i}$  optimalni prijelaz za  $dp_{k,i < n/2}$  dobiva se u intervalu  $p_{k,i < n/2} \in [1, p_{i,n/2}]$ , dok je za  $i > n/2$  ta vrijednost u intervalu  $p_{k,i > n/2} \in [p_{i,n/2}, n]$ . Izračun svih vrijednosti  $dp(k, i)$  stoga se može obaviti rekursivnom metodom *podijeli pa vladaj* u kojoj pamtimo dva intervala. Prvi interval,  $[lo, hi]$ , odnosi se na vrijednosti  $i$  za koje želimo izračunati  $dp(k, i)$ , a drugi interval  $[p_{lo}, p_{hi}]$  odnosi se na granice u kojima se traži optimalni prijelaz  $p_{k,i}$ . Pseudokod rekursivne funkcije dan je u nastavku:

```
izracunaj(lo, hi, p_lo, p_hi):
    mid = (lo + hi) / 2
    p_opt = 0
    dp(k, i) = 0
    za svaki i := p_lo do min(mid-1, p_hi):
        ako dp(k-1, i) + w(i, mid) > dp(k, i):
            dp(k, i) = dp(k-1, i) + w(i, mid)
            p_opt = i
    izracunaj(lo, mid-1, p_lo, p_opt)
    izracunaj(mid+1, hi, p_opt, p_hi)
```

Rekurziju je potrebno pozvati s parametrima  $izracunaj(1, n, 1, n)$ , a može se pokazati da je složenost takvog poziva  $\mathcal{O}(N \log N)$ . Budući da funkciju moramo pozvati  $K$  puta (za svaki prijelaz s  $k-1$  na  $k$ ), ukupna složenost algoritma iznosi  $\mathcal{O}(K \cdot N \log N)$ .



## Zadatak: Redoslijed

Pripremio: Adrian Beker

Potrebno znanje: tournament stablo, topološko sortiranje

Za početak, opisat ćemo rješenja drugog i trećeg podzadatka, u kojima su sve boje u Davorovim potezima međusobno različite. Za  $1 \leq i \leq N$ , neka je  $P_i$  skup poteza čiji interval prekriva  $i$ -ti metar daske te neka  $f_i$  označava njegovu boju, odnosno neka je  $f_i = 0$  ako je on neobojan. Ukoliko je  $f_i = 0$  i  $P_i$  je neprazan, traženi redoslijed ne postoji, stoga ispisujemo "NE". Također, ako je  $f_i > 0$  te  $P_i$  ne sadrži potez boje  $f_i$ , odgovor je "NE". U suprotnom, kako bi  $i$ -ti metar na kraju bio obojan bojom  $f_i$ , nužan i dovoljan uvjet na redoslijed jest sljedeći: jedinstveni potez iz  $P_i$  boje  $f_i$  (nazovimo ga  $z_i$ ) dolazi poslije svih ostalih poteza iz  $P_i$ . Primijetimo sada da uvjete ovog oblika možemo prikazati pomoću usmjerenog grafa  $G$  u kojemu čvorovi predstavljaju poteze, a usmjereni brid  $pq$  označava da se potez  $p$  u redoslijedu nalazi prije poteza  $q$ . Ukoliko  $G$  ima ciklus, odgovor je "NE", a u suprotnom je traženi redoslijed moguće naći topološkim sortiranjem ovog grafa. Naivna implementacija ovog rješenja ima složenost  $\mathcal{O}(N \cdot M)$  te je dovoljna za ostvariti sve bodove na drugom podzadatku.

Za treći podzadatak potrebno je efikasno izgraditi spomenuti graf. U tu svrhu, izgradimo tournament stablo  $T$  nad nizom  $f_i$ , a graf  $G$  proširimo čvorovima stabla  $T$  (ali ne i bridovima). Ovdje ćemo čvorove stabla  $T$  poistovjećivati s pripadajućim intervalima u nizu  $f_i$ . Interval svakog poteza  $p$  podijelimo na čvorove stabla  $T$  (kao što to činimo u upitima na  $T$ ), nazovimo taj skup čvorova  $C_p$ . Za svaki  $x \in C_p$  dodamo brid od  $p$  prema  $x$ . Nadalje, za svaki  $i$  takav da je  $f_i > 0$ , neka  $S_i$  označava skup čvorova stabla  $T$  koji sadrže  $f_i$  te neka je  $y_i$  jedinstveni element u  $C_{z_i} \cap S_i$ . Tada za svaki  $x \in S_i \setminus \{y_i\}$  dodamo brid od  $x$  prema  $z_i$ . Ako za neki  $j$  vrijedi  $z_i \neq z_j$  i  $y_i = y_j$ , odgovor je "NE", a u suprotnom dodamo bridove od svih poteza  $p \neq z_i$  takvih da  $y_i \in C_p$  prema  $z_i$ . Nije teško vidjeti da su valjani redoslijedi inducirani upravo topološkim poretcima ovako izgrađenog grafa  $G$ . Budući da svaki od skupova  $C_p$ ,  $S_i$  ima veličinu  $\mathcal{O}(\log N)$ , graf  $G$  ima  $\mathcal{O}(N + M)$  čvorova i  $\mathcal{O}((N + M) \log N)$  bridova, stoga opisano rješenje ostvaruje sve bodove na trećem podzadatku.

Iako nije jasno kako modificirati ovaj pristup da radi u općenitom slučaju, potpuno rješenje zadatka koristit će neke slične ideje. Najprije za svaki  $1 \leq i \leq N$  provjerimo da vrijedi  $f_i > 0$  ako i samo ako se  $f_i$  nalazi u uniji intervala svih poteza – ako taj uvjet nije ispunjen, odmah znamo da je odgovor "NE". Dalje, traženi redoslijed pohlepno gradimo unatrag. Reći ćemo da je neki potez dobar ako još nije iskorišten, a trenutno se u nizu  $f_i$  na njegovom intervalu pojavljuje samo njegova boja (i eventualno nule). Nije teško tzv. exchange argumentom dokazati da je sljedeći algoritam točan:

Dok nisu svi potezi iskorišteni ponavljaj:

- Ako ne postoji dobar potez, odgovor je "NE";
- Inače uzmi bilo koji dobar potez  $p$ , postavi sve vrijednost u nizu  $f_i$  na njegovom intervalu na 0, te stavi  $p$  na početak redoslijeda.

Primijetimo da se postavljanje elemenata na intervalu na 0 lako svodi na postavljanje jednog elementa na 0 jer je svaki element potrebno najviše jednom postaviti na 0 (npr. ne-nul elemente možemo držati u setu). Naivna implementacija ovog algoritma ima složenost  $\mathcal{O}(N \cdot M)$  te je dovoljna za četvrti podzadatak.

Za sve bodove, preostaje efikasno održavati dobre poteze. Poteze koji su trenutno dobri držat ćemo u redu (queueu)  $Q$ . Nad nizom  $f_i$  izgradimo tournament stablo čiji svaki čvor pamti minimalnu i maksimalnu boju na svojem intervalu (odnosno redom  $\infty$ ,  $-\infty$  ako takva boja ne postoji), nazovimo ih *mini* i *maks*. Tijekom algoritma, za svaki čvor razlikujemo tri faze, ovisno o tome vrijedi li  $mini < maks$ ,  $mini = maks$  ili  $mini > maks$ , odnosno redom pojavljuju li se barem dvije, točno jedna ili niti jedna boja na tom intervalu.

Kao i u rješenju trećeg podzadatka, na početku interval svakog poteza  $p$  podijelimo na čvorove u stablu te



označimo dobiveni skup čvorova s  $C_p$ . Također, održavamo brojač koji broji za koliko čvorova iz skupa  $C_p$  vrijedi  $mini < maks$  ili  $mini = maks \neq c_p$  ( $c_p$  je boja poteza  $p$ ). Kada vrijednost tog brojača padne na 0, potez postaje dobar i stavljamo ga u red  $Q$ . Osvježavanje vrijednosti  $mini$  i  $maks$  u tournamentu radimo na uobičajen način, a odgovarajuće brojače nije teško osvježiti na samom početku te prilikom prijelaza između faza. Ukupna je složenost  $\mathcal{O}((N + M) \log N)$ . Za implementacijske detalje pogledajte službene kodove.



## Zadatak: Sadnice

Pripremila: Paula Vidas

Potrebno znanje: