Sporočilo iz prihodnosti

Imamo dvoboj dveh igralcev. Vsaka partija se vedno konča z zmago enega od njiju.

Poznamo:

- končni rezultat: (a, b)
- trenutni rezultat: (x, y),

kjer velja $0 \leq x \leq a$ in $0 \leq y \leq b$.

Zanima nas, ali lahko enolično določimo zmagovalca naslednje partije.

Od trenutnega do končnega rezultata manjka še:

- Antoinu: ($A = a - x$) zmag,
- Evgeniju: ($B = b - y$) zmag.

Skupno število preostalih partij je:

$$A + B$$

Vsaka naslednja partija poveča bodisi x bodisi y za 1.

Imamo štiri možnosti:

1. Ni več nobene partije Če je $A = 0$ in $B = 0$, potem je trenutni rezultat že končni. Naslednje partije ne bo.
2. Odgovor: 0.
3. Antoine ne sme več zmagati Če je $A = 0$ in $B > 0$, pomeni:
4. Antoine je že dosegel vse svoje zmage,
5. vse preostale partije mora dobiti Evgenij.

Zato mora naslednjo partijo nujno zmagati Evgenij.

- Odgovor: 2.
- Evgenij ne sme več zmagati Če je $B = 0$ in $A > 0$, analogno:
- vse preostale partije mora dobiti Antoine.

Zato mora naslednjo partijo nujno zmagati Antoine.

- Odgovor: 1.
- Oba še lahko zmagujeta Če je $A > 0$ in $B > 0$, potem obstajata vsaj dve možni nadaljevanji:
- v enem zmaga naslednjo partijo Antoine,
- v drugem zmaga naslednjo partijo Evgenij,

in v obeh primerih je še vedno mogoče doseči končni rezultat (a, b).

Zato naslednje partije ne moremo enolično določiti.

- Odgovor: -1.

Povzetek v obliki pogojev

Naj bo:

$$A = a - x, \quad B = b - y$$

Tedaj:

- če $A = 0$ in $B = 0$ → izpiši 0,
- če $A = 0$ in $B > 0$ → izpiši 2,
- če $B = 0$ in $A > 0$ → izpiši 1,
- sicer → izpiši -1.

Zakaj to deluje?

Gre za zelo preprost problem dosegljivosti:

- Končni rezultat zahteva natanko (A) zmag Antoina in (B) zmag Evgenija.
- Če eden od igralcev nima več “prostora” za zmage, potem mora vsako naslednjo partijo zmagati drugi.
- Če imata oba še prostor, je izbira odprta in enoličnosti ni.
- Če prostora nima nihče, je dvoboj že končan.

Časovna zahtevnost

$$\mathcal{O}(1)$$

Beg pred paparaci

Damjan se po mreži križišč velikosti $N \times M$ premika od $(1, 1)$ do (N, M). Dovoljena sta samo dva premika:

- navzdol: $(i, j) \to (i + 1, j)$,
- desno: $(i, j) \to (i, j + 1)$.

Vsako križišče (i, j) ima ceno $a_{i,j}$, ki predstavlja število paparacev. Cena poti je vsota vseh $a_{i,j}$ na obiskanih poljih, vključno z začetnim in končnim.

V vsakem scenariju je natanko eno križišče (x, y) zaprto in ga ne smemo obiskati. Zagotovljeno je, da pot vedno obstaja.

Rešitve po podnalogah

1. podnaloga: $N = 2$

Mreža ima le dve vrstici. To pomeni, da se moramo natanko enkrat premakniti navzdol, vsi ostali premiki so v desno.

Naj bo M število stolpcev. Označimo z d_i ceno poti, če se navzdol premaknemo v stolpcu i:

$$(1, 1) \to (1, 2) \to \cdots \to (1, i) \to (2, i) \to (2, i + 1) \to \cdots \to (2, M)$$

$$d_i = \sum_{j=1}^{i} a_{1,j} + \sum_{j=i}^{M} a_{2,j}$$

Niz d_i lahko izračunamo s:

- prefiksnimi vsotami v prvi vrstici,
- sufiksnimi vsotami v drugi vrstici.

Če je zaprto polje:

- $(1, i)$: navzdol se moramo premakniti pred stolpcem i, zato je odgovor $\min(d_1, \ldots, d_{i-1})$,
- $(2, i)$: navzdol se moramo premakniti po stolpcu i, zato je odgovor $\min(d_{i+1}, \ldots, d_M)$.

Časovna zahtevnost:

$$\mathcal{O}(M + Q)$$

pomnilniška:

$$\mathcal{O}(M)$$

2. podnaloga: $Q \leq 100,; N, M \leq 7$

Uporabimo brute force.

Vsaka pot ima:

$$N + M - 2$$

korakov, od tega

$$N - 1$$

premikov navzdol. Skupno število poti je:

$$\binom{N + M - 2}{N - 1}$$

Za vsak scenarij:

- generiramo vse poti,

- zavržemo tiste, ki gredo čez zaprto polje,
- med ostalimi izberemo minimalni strošek.

Časovna zahtevnost:

$$\mathcal{O}\left(Q \cdot \binom{N+M-2}{N-1}\right)$$

pomnilniška:

$$\mathcal{O}(NM)$$

3. podnaloga: $Q \leq 2000$,; $N \cdot M \leq 2000$

Za vsak scenarij posebej izvedemo dinamično programiranje.

Naj bo:

$$dp_{i,j} = \text{minimalna cena poti od } (1,1) \text{ do } (i,j)$$

pri čemer polje (x, y) preskočimo.

Velja:

$$dp_{1,1} = a_{1,1}$$

$$dp_{i,j} = a_{i,j} + \min(dp_{i-1,j}, dp_{i,j-1})$$

če (i, j) ni zaprto.

Odgovor je $dp_{N,M}$.

Časovna zahtevnost:

$$\mathcal{O}(Q \cdot NM)$$

pomnilniška:

$$\mathcal{O}(NM)$$

4. podnaloga: $Q \leq 10^4$

Najprej izračunamo dve matriki:

- $A_{i,j}$: minimalna cena poti od $(1,1)$ do (i,j),
- $B_{i,j}$: minimalna cena poti od (i,j) do (N,M).

Velja:

$$A_{1,1} = a_{1,1}, \qquad A_{i,j} = a_{i,j} + \min(A_{i-1,j}, A_{i,j-1})$$

$$B_{N,M} = a_{N,M}, \qquad B_{i,j} = a_{i,j} + \min(B_{i+1,j}, B_{i,j+1})$$

Če pot gre skozi (i,j), je njen strošek:

$$A_{i,j} + B_{i,j} - a_{i,j}$$

Opazimo, da vsaka pot od $(1,1)$ do (N,M) na vsaki diagonali $i + j = \text{konst}$ obišče natanko eno polje.

Za zaprto polje (x, y) velja $d = x + y$. Rešitev je:

$$\min_{i+j=d\ (i,j)\neq(x,y)} (A_{i,j} + B_{i,j} - a_{i,j})$$

Ker ima vsaka diagonala največ $\min(N, M)$ polj:

Časovna zahtevnost:

$$\mathcal{O}(NM + Q \cdot \min(N, M))$$

pomnilniška:

$$\mathcal{O}(NM)$$

Celotna rešitev (5. podnaloga)

Vse naredimo enako kot prej, le da poizvedbe pospešimo na $\mathcal{O}(1)$.

Definiramo:

$$C_{i,j} = A_{i,j} + B_{i,j} - a_{i,j}$$

Za vsako diagonalo $i + j = d$ shranimo vrednosti $C_{i,j}$ v zaporedje in nad njim izračunamo:

- prefiksne minimume,
- sufiksne minimume.

Za zaprto polje (x, y):

- pogledamo minimum v prefiksu pred tem poljem,
- pogledamo minimum v sufiksu za tem poljem,
- odgovor je minimum izmed obeh.

Tako izločimo točno eno polje v konstantnem času.

Zakaj to deluje?

Vsako pot lahko razdelimo na dva dela:

- od $(1, 1)$ do neke točke na diagonali $i + j = d$,
- od te točke do (N, M).

Ker mora vsaka pot to diagonalo prečkati natanko enkrat, je dovolj, da izberemo najboljše možno polje na njej, razen tistega, ki je zaprto.

Končna zahtevnost

- izračun matrik A in B: $\mathcal{O}(NM)$,
- izgradnja prefiksnih in sufiksnih minimumov: $\mathcal{O}(NM)$,
- vsak scenarij: $\mathcal{O}(1)$.

Skupaj:

$$\mathcal{O}(NM + Q)$$

Pomnilniška zahtevnost:

$$\mathcal{O}(NM)$$

Pari s podano vsoto

Podani so:

- celo število S,
- celo število N,
- polje $a_1, a_2, \ldots, a_N$.

Prešteti moramo število parov indeksov (i, j), za katere velja:

- $1 \leq i < j \leq N$,
- $a_i + a_j = S$.

Rešitve po podnalogah

1. podnaloga: $1 \leq N \leq 2000$

Uporabimo brute force.

Pregledamo vse pare (i, j), kjer velja $i < j$, in preverimo, ali je:

$$a_i + a_j = S$$

Če je pogoj izpolnjen, povečamo odgovor za 1.

Časovna zahtevnost:

$$\mathcal{O}(N^2)$$

Pomnilniška zahtevnost:

$$\mathcal{O}(1)$$

2. podnaloga: $1 \leq N \leq 200000 \quad in \quad 0 \leq a_i \leq 10^5$

Ker so vse vrednosti majhne in nenegativne, lahko uporabimo frekvenčno tabelo.

Naj bo:

$$cnt[x] = \text{število pojavitev števila } x$$

Najprej preberemo vse elemente in napolnimo tabelo cnt.

Nato za vsak x pogledamo, koliko elementov $y = S - x$ obstaja.

Obravnavamo dva primera:

1. $x \neq y$

Prispevek k rezultatu je:

$$cnt[x] \cdot cnt[y]$$

1. $x = y$

Takrat štejemo pare znotraj iste vrednosti:

$$\binom{cnt[x]}{2} = \frac{cnt[x] \cdot (cnt[x] - 1)}{2}$$

Da ne štejemo parov dvakrat, upoštevamo le tiste x, za katere velja:

$$x \leq y$$

Časovna zahtevnost:

$$\mathcal{O}(N + A)$$

kjer je $A = 10^5$ največja možna vrednost elementa.

Pomnilniška zahtevnost:

$$\mathcal{O}(A)$$

3. podnaloga: $1 \leq N \leq 200000$, polje je urejeno

Uporabimo metodo dveh kazalcev.

Naj bo:

- $l = 1$,
- $r = N$.

Dokler velja $l < r$:

- če je $a_l + a_r < S$, povečamo l,
- če je $a_l + a_r > S$, zmanjšamo r,
- če je $a_l + a_r = S$, imamo veljaven par.

Takrat moramo paziti na ponovitve.

Naj bo:

- $x = a_l$,
- $y = a_r$.

Če je $x \neq y$:

- preštejemo, kolikokrat se x pojavi zaporedno od leve,
- preštejemo, kolikokrat se y pojavi zaporedno od desne,
- k rezultatu prištejemo:

$$cnt_x \cdot cnt_y$$

Če je $x = y$:

- vemo, da so vsi elementi med l in r enaki,
- število parov je:

$$\binom{r - l + 1}{2}$$

in algoritem lahko končamo.

Časovna zahtevnost:

$$\mathcal{O}(N)$$

Pomnilniška zahtevnost:

$$\mathcal{O}(1)$$

Celotna rešitev (4. podnaloga)

Uporabimo slovar oziroma zgoščeno tabelo.

Iteriramo po polju od leve proti desni. Naj bo:

$$cnt[x] = \text{kolikokrat se je število } x \text{ že pojavilo.}$$

Za trenutni element a_i potrebujemo število:

$$S - a_i$$

Vsak prejšnji pojav tega števila tvori veljaven par z a_i.

Zato:

- k rezultatu prištejemo:

$$cnt[S - a_i]$$

- nato povečamo:

$$cnt[a_i] \leftarrow cnt[a_i] + 1$$

S tem zagotovimo, da vedno štejemo le pare z $i < j$.

Zakaj to deluje?

Ko obdelujemo element a_i, so v slovarju zapisani vsi elementi:

$$a_1, a_2, \ldots, a_{i-1}$$

Vsak od njih, ki je enak $S - a_i$, skupaj z a_i tvori par s pravilno vsoto.

Ker vsak par štejemo natanko takrat, ko obdelamo večji indeks, se noben par ne prešteje dvakrat in noben ne ostane neupoštevan.

Končna zahtevnost

Časovna zahtevnost:

$$\mathcal{O}(N)$$

Pomnilniška zahtevnost:

$$\mathcal{O}(N)$$

Začarane sobane

Imamo graf z N sobanami in M prehodi. Vsak prehod ima tip $t \in 0, 1$:

- $t = 0$: prost prehod (uporaben, ko je stikalo izklopljeno),
- $t = 1$: trdno zaprt prehod (uporaben, ko je stikalo vklopljeno).

Zvitorepec začne v sobani u, stikalo je na začetku izklopljeno (stanje 0). V sobani lahko stikalo poljubno velikokrat preklopi; vsak preklop stane 1. Želimo najmanjše število preklopov, da pride v sobano v (končno stanje je poljubno).

Če to ni mogoče, izpišemo -1.

Rešitve po podnalogah

1. podnaloga: $M = N - 1$, iz vsake sobane vodita največ 2 prehoda

Ker je $M = N - 1$ in je stopnja vsakega vozlišča največ 2, je graf drevo in hkrati "veriga" (ena sama pot).

V drevesu je pot med u in v enolična. Naj bo zaporedje tipov prehodov na tej poti:

$$t_1, t_2, \ldots, t_k$$

Začnemo v stanju:

$$s = 0$$

Ko želimo prečkati prehod tipa t_i:

- če $t_i = s$, gremo naprej brez preklopa,
- če $t_i \neq s$, moramo preklopiti stikalo:

$$\text{odgovor} \leftarrow \text{odgovor} + 1, \qquad s \leftarrow t_i.$$

Ker je pot enolična, je to optimalno.

Časovna zahtevnost:

$$\mathcal{O}(N)$$

Pomnilniška zahtevnost:

$$\mathcal{O}(N)$$

2. podnaloga: vsi prehodi so tipa 1

Če je $u = v$, je odgovor:

$$0$$

Sicer je premikanje možno le v stanju 1, zato moramo stikalo vsaj enkrat preklopiti (pred prvim prehodom). Torej:

- če sta u in v povezana v grafu, je odgovor 1,
- sicer -1.

Časovna zahtevnost (npr. BFS/DFS):

$$\mathcal{O}(N + M)$$

Pomnilniška zahtevnost:

$$\mathcal{O}(N)\ (BFS) \quad oziroma \quad \mathcal{O}(N + M)\ (DFS)$$

3. podnaloga: $M = N - 1$ in graf je povezan (drevo)

Ker je graf drevo, je pot med u in v enolična. Rešitev je enaka kot pri 1. podnalogi:

- najdemo enolično pot $u \to v$,
- gremo po njej in štejemo, kolikokrat se tip naslednjega prehoda razlikuje od trenutnega stanja.

Časovna zahtevnost:

$$\mathcal{O}(N)$$

Pomnilniška zahtevnost:

$$\mathcal{O}(N)$$

4. podnaloga: $N, M \leq 5000$

Uporabimo splošno rešitev z razširjenim grafom (opisano spodaj). Ker so omejitve majhne, za to podnalogo delujejo tudi manj učinkoviti algoritmi (npr. kvadratni Dijkstra algoritem), vendar imamo uteži $0/1$, zato je naravna izbira 0–1 BFS (rešitev za 100 točk).

Časovna zahtevnost:

$$\mathcal{O}(N + M)$$

Pomnilniška zahtevnost:

$$\mathcal{O}(N + M)$$

Celotna rešitev (5. podnaloga)

Ključna ideja: stanje stikala je del stanja poti.

Zato naredimo razširjen graf z $2N$ vozlišči:

- vozlišče $(i, 0)$ pomeni: smo v sobani i in stikalo je izklopljeno,
- vozlišče $(i, 1)$ pomeni: smo v sobani i in stikalo je vklopljeno.

Dodamo dve vrsti povezav:

1. Premiki po prehodih (strošek 0)

Za vsak prehod (a, b, t):

- iz (a, t) lahko gremo v (b, t) s stroškom 0,
- iz (b, t) lahko gremo v (a, t) s stroškom 0.

- Preklop stikala v sobani (strošek 1)

Za vsako sobano i:

- $(i, 0) \leftrightarrow (i, 1)$ s stroškom 1.

Začetno vozlišče je:

$$(u, 0).$$

Cilj je doseči:

$$(v, 0) \text{ ali } (v, 1)$$

ker je končno stanje poljubno.

Ker so uteži samo 0 ali 1, najkrajšo pot izračunamo z **0–1 BFS**.

Naj bo $dist[i][s]$ najmanjše število preklopov do stanja (i, s). Algoritem uporablja deque:

- prehod s stroškom 0 potisnemo na začetek,
- prehod s stroškom 1 potisnemo na konec.

Odgovor je:

$$\min(dist[v][0], dist[v][1]),$$

če je dosegljivo, sicer -1.

Zakaj to deluje?

Vsaka pot v originalnem problemu je zaporedje dveh vrst korakov:

- prečkanje prehoda tipa t (dovoljeno samo, če je stanje stikala t),
- preklop stikala (spremeni stanje in stane 1).

V razširjenem grafu:

- prehod tipa t je premik med (a, t) in (b, t) s stroškom 0,
- preklop je premik med $(i, 0)$ in $(i, 1)$ s stroškom 1.

Zato vsaka veljavna pot v originalnem grafu ustreza poti v razširjenem grafu z enakim številom preklopov in obratno. Iskanje najmanjšega števila preklopov je torej natanko problem najkrajše poti v grafu z utežmi $0/1$, kar pravilno reši 0–1 BFS.

Veljavne so tudi implementacije, ki z DFS prehode tipa 0 in 1 sestavi v povezane skupine in nato z BFS iščemo najkrajšo pot.

Končna zahtevnost

Razširjeni graf ima:

- $2N$ vozlišč,
- približno $2M$ povezav za prehode (v obeh smereh),
- $2N$ povezav za preklop.

0–1 BFS teče v času:

$$\mathcal{O}(N + M)$$

Pomnilniška zahtevnost:

$$\mathcal{O}(N + M)$$