

# CEOI 2026



1. dan  
(crnogorski)

Day 1  
(Montenegrin)



## Posmatrači ptica

Ograničenje vremena: 10 s    Ograničenje memorije: 512 MiB

Društvo posmatrača ptica San Serriffe ima neobično složenu unutrašnju strukturu, koja se stalno mijenja. Društvo se sastoji od  $n$  ogranaka, a svaki član Društva pripada tačno jednom ogranku. Ogranci su numerisani od 1 do  $n$ , a  $i$ -ti ogranak ima  $m_i$  članova. Dakle, Društvo ima ukupno  $M = m_1 + m_2 + \dots + m_n$  članova.

Svaki ogranak vodi jedan od njegovih članova, koji se u toj ulozi naziva *službenik* ogranaka. Službenici su numerisani isto kao i ogranci, tako da je (za svako  $i = 1, \dots, n$ ) službenik  $i$  zadužen za ogranak  $i$ .

Štaviše, službenici su hijerarhijski organizovani kroz sistem mentorstva: svaki službenik osim jednog ima *mentora*, koji je službenik nekog drugog ogranaka. Jedini službenik bez mentora je *Predsjednik* Društva. Ako je službenik  $a$  mentor službeniku  $b$ , takođe kažemo da je službenik  $b$  *učenik* službenika  $a$ .

Nijedan službenik nije, direktno ili indirektno, mentor samom sebi; stoga, prateći niz od službenika do njegovog mentora, zatim od mentora mentorovog mentora i tako dalje, na kraju se uvijek stiže do Predsjednika.

Definišemo *uticaj* službenika kao zbir broja članova u njegovom ogranku i uticaja svih njegovih učenika (ako ih ima). Lako se može vidjeti da je službenik sa najvećim uticajem Predsjednik, čiji je uticaj uvijek jednak  $M$ . Službenik se naziva *stariji službenik* ako je njegov uticaj  $\geq M/2$ .

Statut Društva propisuje da stariji službenik sa najmanjim uticajem (među svim starijim službenicima) obavlja funkciju *Blagajnika* Društva.

S vremena na vrijeme, službenik (osim Predsjednika) može *promijeniti svoju pripadnost*, tako da od tada postaje učenik drugog mentora (pod uslovom da njegov novi mentor nije jedan od njegovih učenika, učenika njegovih učenika itd.). Zbog toga se može desiti da se uticaj nekih službenika promijeni i da uloga Blagajnika pripadne drugom službeniku.

### Zadatak

Napišite program koji učitava opis početnog stanja Društva i niz promjena pripadnosti. Vaš program mora ispisati ko je Blagajnik u početnom stanju Društva, kao i nakon svake promjene pripadnosti.

### Ulaz

Prvi red sadrži dva cijela broja,  $n$  i  $q$ , odvojena razmakom;  $n$  je broj ogranaka, a  $q$  broj promjena pripadnosti.

Narednih  $n$  redova opisuju početno stanje Društva.  $i$ -ti od tih redova sadrži dva cijela broja  $s_i$  i  $m_i$ , odvojena razmakom;  $s_i$  je mentor službenika  $i$  (tj. službenika zaduženog za ogranak  $i$ ), dok je  $m_i$  broj članova ogranaka  $i$ . Vrijednost  $s_i = 0$  označava da je službenik  $i$  Predsjednik Društva i stoga nema mentora.

Preostalih  $q$  redova opisuju promjene pripadnosti.  $j$ -ti od tih redova sadrži dva cijela broja  $\hat{x}_j$  i  $\hat{z}_j$ , odvojena razmakom. Značenje ovih brojeva je sljedeće.

Neka je  $t_j$  (za  $j = 0, \dots, q$ ) Blagajnik nakon prvih  $j$  promjena pripadnosti (dakle,  $t_0$  je početni Blagajnik prije prve promjene pripadnosti). Tada se  $j$ -ta promjena pripadnosti sastoji u tome da službenik  $z_j$  postaje novi mentor službenika  $x_j$ , gdje je

$$x_j = 1 + ((t_{j-1} + \hat{x}_j) \bmod n) \text{ i } z_j = 1 + ((t_{j-1} + \hat{z}_j) \bmod n).$$

Svrha ovakvog predstavljanja vrijednosti  $x_j$  i  $z_j$  jeste da primora vaš program da obrađuje promjene pripadnosti redoslijedom kojim se pojavljuju.

Promjene pripadnosti u ulaznim podacima će uvijek biti ispravne, tj.  $z_j$  neće biti jednako  $x_j$ , niti će  $z_j$  biti učenik od  $x_j$ , učenik učenika od  $x_j$ , itd. Međutim, moguće je da je  $z_j$  već bio mentor od  $x_j$  neposredno prije  $j$ -te promjene (tako da se u tom trenutku zapravo ništa ne mijenja).

Napomena: ako vaš program u nekom trenutku izračuna pogrešan rezultat  $t_j$ , tada će i naredne ulazne vrijednosti  $\hat{x}_{j+1}, \hat{z}_{j+1}$  itd. pogrešno dekodirati, pa može dobiti kao ezltat RTE (greška pri izvršavanju) umjesto WA (pogrešan odgovor), jer pogrešno dekodirani ulazi mogu postati neispravni (npr. može pogrešno dobiti  $z_{j+1}$  koji je učenik od  $x_{j+1}$ ).

### Ograničenja

- $1 \leq n \leq 1\,000\,000$
- $1 \leq q \leq 30\,000$
- $1 \leq m_i$  za svako  $i = 1, \dots, n$
- $m_1 + m_2 + \dots + m_n \leq 10^9$
- $1 \leq \hat{x}_j \leq n$  i  $1 \leq \hat{z}_j \leq n$  za svako  $j = 1, \dots, q$ .

### Podzadaci

- Podzadatak 1 (15 poena):  $n \leq 100$
- Podzadatak 2 (10 poena):  $n \leq 1000$
- Podzadatak 3 (50 poena):  $n \leq 300\,000$
- Podzadatak 4 (25 poena): Nema dodatnih ograničenja.

### Izlaz

Ispišite brojeve  $t_0, t_1, \dots, t_q$ , svaki u posebnom redu, gdje je  $t_j$  Blagajnik nakon prvih  $j$  promjena pripadnosti. Naravno, svaki  $t_j$  mora biti cijeli broj iz opsega  $1 \leq t_j \leq n$ .

### Primjer

Ulaz	Izlaz
7 2	2
0 1	2
1 3	3
1 3	
2 3	
2 1	
5 2	
5 1	
3 7	
2 7	

### Komentar

Na početku je službenik 2 Blagajnik (otuda  $t_0 = 2$ ). U prvoj promjeni pripadnosti, čitamo  $\hat{x}_1 = 3$  i  $\hat{z}_1 = 7$  i računamo  $x_1 = 1 + ((2 + 3) \bmod 7) = 6$  i  $z_1 = 1 + ((2 + 7) \bmod 7)$ , tako da službenik 3 postaje novi mentor službenika 6; službenik 2 ostaje Blagajnik (tada je  $t_1 = 2$ ). U drugoj promjeni pripadnosti, čitamo  $\hat{x}_2 = 2$  i  $\hat{z}_2 = 7$  i računamo  $x_2 = 1 + ((2 + 2) \bmod 7) = 5$  i  $z_2 = 1 + ((2 + 7) \bmod 7) = 3$ , tako da službenik 3 postaje novi mentor službenika 5 i takođe postaje novi Blagajnik (pa je  $t_2 = 3$ ).



## DFS

Ograničenje vremena: 1 s    Ograničenje memorije: 256 MiB

Možda ste već upoznati sa poznatim DFS (depth-first search) algoritmom za obilazak grafa. U ovom zadatku razmatraćemo samo povezane neusmjerene proste grafove (bez petlji i višestrukih grana) sa čvorovima numerisanim brojevima  $0, 1, \dots, n - 1$ , a DFS algoritam će ispisivati dubine i čvorove na sljedeći način:

```
DFS(d, v):
  output d/v
  mark vertex v as visited
  W = the list of neighbors of v ordered by increasing numbers
  for each w in W:
    if vertex w has not yet been visited:
      DFS(d + 1, w)
```

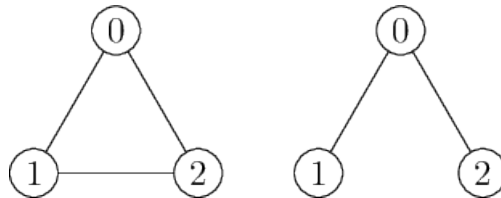
Napišite program koji ispisuje broj grafova za koje poziv  $\text{DFS}(0, n - 1)$  proizvodi isti ispis kao onaj dat na ulazu. Na primjer, ispis

0/2

1/0

2/1

dobija se pozivom  $\text{DFS}(0, 2)$  na bilo kojem od sljedeća dva povezana neusmjerena prosta grafa sa 3 čvora:



### Ulaz

Ulaz je ispis poziva  $\text{DFS}(0, n - 1)$  na nepoznatom povezanom neusmjerenom prostom grafu sa  $n$  čvorova. Ulaz se stoga sastoji od  $n$  linija u formatu  $d/v$ , pri čemu je prva linija  $0/n-1$ .

### Ograničenja

- $1 \leq n \leq 2 \cdot 10^5$

### Podzadaci

- Podzadatak 1 (10 poena):  $n \leq 6$ .
- Podzadatak 2 (20 poena):  $n \leq 500$ .
- Podzadatak 3 (20 poena):  $n \leq 10^4$ .

- Podzadatak 4 (10 poena): Za svaki  $i \in \{2, \dots, n\}$ ,  $i$ -ta ulazna linija je  $i-1/i-2$ .
- Podzadatak 5 (20 poena): Za svaki  $i \in \{2, \dots, n\}$ ,  $i$ -ta ulazna linija je  $i-1/v$  za neki  $v \in \{0, \dots, n-2\}$ .
- Podzadatak 6 (20 poena): Nema dodatnih ograničenja.

## Izlaz

Izračunajte traženi broj različitih grafova sa traženim svojstvom. Pošto taj broj može biti veoma velik, ispišite rezultat modulo 1 000 000 007.

## Primjer

Ulaz	Izlaz
0/2	2
1/0	
2/1	

# Potruga za blagom

Ograničenje vremena: 8 s    Ograničenje memorije: 256 MiB

Tražiš davno izgubljeno blago na ogromnom polju ćelija veličine  $N \times N$  uz pomoć magičnog kompasa. Ukupno postoji  $K$  kovčega s blagom skrivenih u različitim ćelijama. Tvoj cilj je da pronadeš sve kovčege sa blagom!

Kada postaviš kompas na neku ćeliju, on će pronaći najkraći put do kovčega s blagom koristeći samo pomjeranja ulijevo, nagore, udesno i nadolje (tj. do najbližeg kovčega prema Menhetn rastojanju). Zatim će pokazati smjer prvog koraka. Ako postoji više ispravnih prvih koraka koji vode do najbližeg kovčega (ili više njih), kompas će vratiti sve te smjerove. Ako ćelija sadrži blago, kompas će to naznačiti.

Kad god pronadeš kovčeg s blagom, možeš isprazniti njegov sadržaj, ali ne možeš ukloniti kovčeg pošto je pretežak. Tvoj kompas ne zna da li je kovčeg pun ili prazan pa će pokazivati put ka najbližem kovčegu, bilo da je prazan ili pun.

Pokušaj da pronadeš lokacije svih kovčega s blagom koristeći što manje upita kompasu.

## Zadatak

Ovo je interaktivni zadatak. U svakom test slučaju (tj. pri svakom izvršavanju tvog programa), program će morati da riješi nekoliko potraga na blago. Sa sistemom za ocjenjivanje treba da komuniciraš pomoću biblioteke koju su obezbijedili organizatori. Biblioteka sadrži sljedeće deklaracije:

- **void** *NextHunt*(**int** &*N*, **int** &*K*) — pozovi ovu funkciju da započneš sljedeću potragu za blagom. Ona će postaviti parametar  $N$  na veličinu mreže a parametar  $K$  na broj kovčega blaga. Ako nema više potraga na blago koje treba riješiti u trenutnom izvršavanju, funkcija će postaviti  $N$  i  $K$  na  $-1$ ; u tom slučaju treba da završiš program sa izlaznim kodom 0. Napomena: ovu funkciju možeš pozvati i prije nego što pronadeš sva blaga u trenutnoj potrazi, npr. ako pokušavaš da osvojiš samo djelimičan broj poena.
- **enum** { *TREASURE* = 0, *DIR\_RIGHT* = 1, *DIR\_UP* = 2, *DIR\_LEFT* = 4, *DIR\_DOWN* = 8 }; — ovo su konstante koje se koriste u povratnim vrijednostima funkcije *Query* (vidi ispod).
- **int** *Query*(**int** *x*, **int** *y*) — ako ćelija na koordinatama  $(x, y)$  sadrži blago, ova funkcija vraća *TREASURE*. U suprotnom vraća zbir jedne ili više konstanti *DIR\_RIGHT*, *DIR\_UP*, *DIR\_LEFT* i *DIR\_DOWN*, označavajući koja pomjeranja kompas vraća kada se postavi na ćeliju  $(x, y)$ . Koordinate  $x$  i  $y$  moraju biti cijeli brojevi od 0 do  $N - 1$ . (Napomena: u ovom zadatku  $y$ -koordinate rastu od vrha ka dnu.)

Nakon što *NextHunt* vrati  $N = K = -1$ , tvoj program više ne smije pozivati ni *NextHunt* ni *Query*. Takođe ne smije pozvati *Query* prije prvog poziva funkcije *NextHunt*. Ako program ne ispoštuje ova ograničenja, ili ako napravi više od 1000 upita tokom jedne potrage za blagom, ili ako funkciji *Query* proslijedi vrijednosti  $x$  i/ili  $y$  van dozvoljenog opsega, biblioteka će prekinuti program i vratiti run-time-error (RTE) za trenutni test slučaj.

Blago se smatra pronađenim ako si makar jednom postavio upit za ćeliju koja ga sadrži. Ako program pozove *NextHunt* prije nego što pronađe sva blaga iz trenutne pretrage, to se ne smatra greškom, ali će uticati na rezultat (više o bodovanju u nastavku).

Za pristup biblioteci, program treba da uključi heder fajl `treasurehuntlib.h`:

```
#include "treasurehuntlib.h"
```

Heder fajl možeš preuzeti ovdje: `treasurehuntlib.h`.

Kako bi ti pomogli u razvoju rješenja, dostupna je jednostavna implementacija biblioteke ovdje: `treasurehuntlib-public.cpp`. Da bi je kompajlirao zajedno sa svojim programom, jednostavno dodaj njeno ime kao parametar kompajleru, na primjer:

```
g++ foo.cpp treasurehuntlib-public.cpp
```

ako je `foo.cpp` ime fajla koji sadrži tvoje rješenje.

Implementacija fajla `treasurehuntlib-public.cpp` podržava, dodatno u odnosu na funkcije odozgo, i funkciju `void InitFromFile(const char *fileName)`, koja čita listu potraga za blagom iz fajla i omogućava vam da se igrate sa njima umjesto da se lista pretraga na blago generiše random. Za više detalja o ovoj funkciji pogledaj fajl `treasurehuntlib-public.cpp`.

Na evaluacionom serveru koristiće se drugačija implementacija biblioteke, tako da ne treba da praviš nikakve pretpostavke o tome kako je tačno implementirana biblioteka. Ipak, možeš pretpostaviti da biblioteka ne zagađuje globalni prostor imena nikakvim deklaracijama osim navedenih (*NextHunt*, *Query* i pet konstanti).

Tvoj program ne smije čitati sa standardnog ulaza niti pisati na standardni izlaz, jer će ih naša implementacija biblioteke na evaluacionom serveru koristiti za komunikaciju sa ostatkom evaluacionog okruženja.

## Ulaz

### Ograničenja

- $1 \leq N \leq 10^6$
- $1 \leq K \leq 3$
- Tokom jednog izvršavanja programa biće najviše 100 000 potraga za blagom.
- Tokom jedne potrage za blagom možeš napraviti najviše 1000 upita.

### Podzadaci

- Podzadatak 1 (10 poena)  $K = 1$
- Podzadatak 2 (30 poena)  $K = 2$
- Podzadatak 3 (60 poena)  $K = 3$ .

## Bodovanje

Jedan podzadatak može se sastojati od nekoliko test slučajeva (nekoliko izvršavanja tvog programa), a svaki test slučaj može sadržati nekoliko potraga za blagom. Za potrebe bodovanja, sve potrage za blagom datog podzadatka ocjenjuju se zajedno, bez obzira na to kako su prvobitno bile raspoređene među test slučajevima. Za  $i$ -tu potragu za blagom veličinu mreže označavamo sa  $N_i \times N_i$ , broj kovčega blaga sa  $K_i$ , broj upita koje je napravio program sa  $Q_i$ , i broj pronađenih blaga sa  $F_i$ . Neka takođe  $S$  označava ukupan broj poena dostupan za taj podzadatak. Tada je broj poena dodijeljen rješenju za taj podzadatak:

- Ako je program uvijek pronašao sve kovčege sa blagom (tj. ako je  $F_i = K_i$  za sva  $i$ ), rezultat zavisi od  $t_i = \frac{Q_i}{\lceil \log_2 N_i \rceil}$ :

$$\frac{S}{2} + \frac{S}{2} \cdot \min_i f(t_i) \text{ poena, gdje je } f(t) = \begin{cases} 1, & t \leq 11 \\ 1 - (t - 11)/9, & 11 \leq t \leq 20 \\ 0, & t \geq 20. \end{cases}$$

- Ako program nije uvijek pronašao sve kovčege sa blagom (tj. postoji  $i$  takav da je  $F_i < K_i$ ), dobija

$$\frac{S}{2} \cdot \min_i \frac{F_i}{K_i} \text{ poena.}$$

Drugim riječima, dobijaš polovinu poena za pronalaženje svih kovčega sa blagom, a drugu polovinu za njihovo pronalaženje sa što manje upita. Za maksimalan broj poena, tvoje rješenje treba da pronađe sva blaga u najviše  $11 \lceil \log_2 N_i \rceil$  upita. Između  $11 \lceil \log_2 N_i \rceil$  i  $20 \lceil \log_2 N_i \rceil$  upita rezultat opada linearno; a ako tvoje rješenje napravi više od  $20 \lceil \log_2 N_i \rceil$  upita, dobiće samo prvu polovinu poena za pronalaženje svih blaga. (Simboli  $\lceil \cdot \rceil$  znače da se vrijednost  $\log_2 N_i$  zaokružuje naviše na najbliži cijeli broj.)

Ako gornje formule daju necijeli broj poena za podzadatak, rezultat će biti zaokružen na najbliži cijeli broj.

Ako program ima grešku tokom izvršavanja ili se ne pridržava prethodno opisanog protokola za korišćenje biblioteke, dobiće 0 poena za cio podzadatak. Da bi dobio djelimične poene za pronalaženje samo podskupa kovčega sa blagom, program mora pravilno završiti pretragu pozivom funkcije *NextHunt*.

## Primjer

Poziv	Povratna vrijednost
NextHunt(N, K)	$N = 4, K = 1$
Query(2, 0)	$DIR\_DOWN + DIR\_RIGHT = 9$
Query(3, 1)	$DIR\_DOWN$
Query(3, 2)	$TREASURE$
NextHunt(N, K)	$N = -1, K = -1$