

CEOI 2026



Ziua 2
(Română)

Day 2
(Romanian)

Tăierea florilor

Limită de timp: 4 s Limită de memorie: 256 MiB

În grădina comunității CEOI cultivăm o colecție de flori speciale care își împletesc strâns rădăcinile. Dacă le tăiem, ele cresc la loc, atâta timp cât nu tăiem prea agresiv și nu provocăm daune ireparabile.

Dacă o pereche de flori, a și b , își împletește rădăcinile, spunem că florile sunt „conectate”. În caz contrar, sunt „deconectate”.

Rădăcinile cresc conform următoarei reguli: Fie a și b două flori deconectate. Dacă există cel puțin alte 2 flori c și d , astfel încât atât a , cât și b sunt conectate atât cu c , cât și cu d , atunci între a și b vor crește rădăcini și ele vor deveni conectate.

Aceste flori cresc deja de ceva timp, iar toate rădăcinile care puteau crește urmând regula de mai sus au crescut deja. Cu alte cuvinte, dacă două flori a și b sunt ambele conectate cu anumite flori c și d , atunci este garantat că și a și b sunt conectate între ele.

Trebuie să dezrădăcinăm această grădină și să o mutăm la următoarea locație CEOI. Pentru a simplifica mutarea, am dori să tăiem cât mai multe rădăcini posibil. Totuși, dorim ca florile să crească din nou până la starea lor actuală. Care este numărul maxim de perechi de flori conectate ale căror conexiuni le putem tăia astfel încât ele să crească totuși la loc și să revină la starea lor actuală? Nu contează câte iterații de creștere ar fi necesare.

Intrare

Prima linie a intrării conține două numere întregi separate printr-un spațiu, n și m , reprezentând numărul de flori și numărul de conexiuni existente între ele. Urmează m linii, fiecare conținând o pereche de numere întregi a_i și b_i , indicând faptul că florile a_i și b_i sunt conectate.

Florile sunt numerotate cu numere întregi de la 1 la n .

Se garantează că datele de intrare respectă regula descrisă în enunț.

Restricții

- $1 \leq n \leq 1000$
- $1 \leq m \leq 10^5$

Subtaskuri

- Subtask 1 (20 puncte): $n \leq 10$ și $m \leq 20$
- Subtask 2 (14 puncte): $m = \frac{n \cdot (n-1)}{2}$
- Subtask 3 (15 puncte): Garantăm că fiecare floare este conectată cu cel mult 7 alte flori.
- Subtask 4 (15 puncte): $n \leq 50$ și $m \leq 1000$
- Subtask 5 (36 puncte): Fără restricții suplimentare.

Ieșire

Afișați un singur număr întreg — numărul maxim de conexiuni pe care le putem tăia.

Exemplu

Intrare

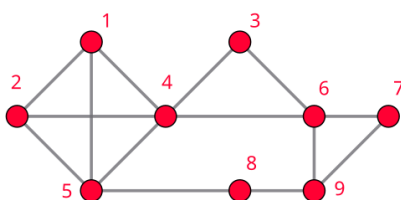
```
9 14
1 2
1 4
1 5
2 4
2 5
3 4
4 5
3 6
4 6
6 7
6 9
7 9
8 9
5 8
```

Ieșire

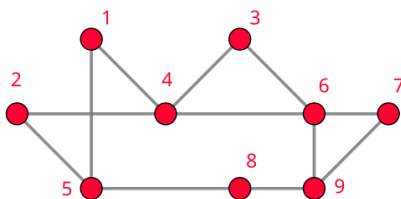
```
2
```

Comentariu

Florile din exemplu înainte de orice tăiere. Se poate verifica faptul că nu se pot forma conexiuni suplimentare pe baza procedurii de creștere descrise.



Florile din exemplu după tăiere au cu 2 conexiuni mai puțin. Conexiunea dintre 1 și 2 poate crește din nou deoarece florile 1 și 2 sunt ambele conectate la florile 4 și 5. În mod similar, conexiunea dintre 4 și 5 poate crește din nou deoarece florile 4 și 5 sunt ambele conectate la florile 1 și 2.



Turnuri

Limită de timp: 1 s Limită de memorie: 256 MiB

Aveți n calculatoare și m turnuri, toate aflate în poziții distincte de-a lungul unei drepte. Trebuie să împerecheați calculatoarele folosind cabluri astfel încât fiecare cablu să pornească de la un calculator, să viziteze anumite turnuri și să se termine la un alt calculator. Cablul poate vizita oricare turnuri (nu doar cele dintre calculatoarele de la capetele sale) în orice ordine. Poate sări peste turnuri trecând pe lângă ele fără să le viziteze. De asemenea, poate să nu viziteze niciun turn și să conecteze direct cele două calculatoare, dar nu poate conecta un calculator la el însuși. Numărul de calculatoare este par.

Fie a și b pozițiile a două calculatoare și fie $x_1, \dots, x_k$ pozițiile turnurilor pe care le vizitează cablul. Lungimea cablului este $|a - x_1| + |x_1 - x_2| + \dots + |x_{k-1} - x_k| + |x_k - b|$. Pentru un anumit cablu definim scorul său ca fiind $f \cdot u - l$, unde l este lungimea sa, f este o constantă fixă, iar u este numărul de turnuri distincte pe care cablul le vizitează dintre $x_1, \dots, x_k$. Mai multe cabluri pot vizita același turn, iar acel turn contribuie la scorul fiecăruia dintre acele cabluri.

Trebuie să calculați suma maximă posibilă a scorurilor cablurilor pentru o împerechere a tuturor calculatoarelor (adică fiecare calculator trebuie să aparțină exact unei perechi sau, echivalent, trebuie să fie conectat prin exact un cablu).

Intrare

Prima linie conține numărul de cazuri de test, T . Cazurile de test urmează unul după altul. Fiecare caz de test este alcătuit din trei linii. Prima linie conține trei numere întregi n , m și f — numărul de calculatoare, numărul de turnuri și constanta f . A doua linie conține n numere întregi $a_1, a_2, \dots, a_n$ — pozițiile calculatoarelor. A treia linie conține m numere întregi $b_1, b_2, \dots, b_m$ — pozițiile turnurilor.

Restricții

Fie N și M sumele valorilor lui n , respectiv m , peste toate cazurile de test.

- $1 \leq T \leq 10^4$
- $1 \leq N, M \leq 2 \cdot 10^5$
- $0 \leq f \leq 10^9$
- n este par
- $1 \leq a_i, b_i \leq 10^9$
- Toate pozițiile calculatoarelor și turnurilor sunt distincte (în cadrul unui singur caz de test).

Subtaskuri

- Subtaskul 1 (5 puncte): $N \leq 5000$, $m = 1$
- Subtaskul 2 (10 puncte): $T \leq 20$, $n \leq 10$, $m \leq 100$

- Subtaskul 3 (27 puncte): $N, M \leq 5000$
- Subtaskul 4 (21 puncte): $N \leq 5000$
- Subtaskul 5 (37 puncte): Fără restricții suplimentare.

Ieșire

Afișați T numere întregi, fiecare pe câte o linie — suma maximă posibilă a scorurilor cablurilor pentru fiecare caz de test.

Exemplu

Intrare

```
4
2 1 100
1 10
11
4 1 10
2 4 6 8
20
4 1 10
2 4 6 8
5
6 3 10
2 13 4 8 6 10
5 1 9
```

Ieșire

```
89
-4
12
51
```

Vim

Limită de timp: 1 s Limită de memorie: 256 MiB

E greu să găsim ceva mai *geeky* decât vechiul Vim — un editor în care putem face lucruri la care în editoarele „obișnuite” am putea doar să visăm. Desigur, dacă suntem dispuși să investim o săptămână pentru învățare și o lună de practică pentru a ne intra în reflex comenzile neobișnuite, dar puternice, pe care acesta le oferă. Ei bine, la fel ca în alte editoare, în Vim avem un *cursor*, care este mereu poziționat pe unul dintre caracterele textului. Inițial, acesta este poziționat pe primul caracter al textului, și, la fel ca în alte editoare, în Vim există și un *clipboard*, destinat stocării (și indirect copierii și mutării) fragmentelor de text. Clipboard-ul este inițial gol.

În această problemă vom presupune că la început există exact un caracter '-' (minus) în editor, iar scopul nostru este să obținem exact n semne minus consecutive. Vom folosi patru comenzi pentru a ne ajuta:

- **h**: Dacă cursorul este pe primul caracter, atunci această comandă nu face nimic; în caz contrar, mută cursorul cu un caracter spre stânga.
- **l**: Dacă cursorul este pe ultimul caracter, atunci această comandă nu face nimic; în caz contrar, mută cursorul cu un caracter spre dreapta.
- **Y**: Secvența de caractere care se întinde de la poziția cursorului până la sfârșitul textului este copiată în clipboard, „suprascriind” orice conținut anterior al clipboard-ului. (Dacă cursorul este pe primul caracter al textului, întregul text va fi copiat în clipboard.)
- **P**: O copie a textului stocat în clipboard este inserată înaintea caracterului pe care se află cursorul, iar cursorul este mutat pe ultimul caracter inserat. Conținutul clipboard-ului nu este modificat. Dacă clipboard-ul este gol, nu se întâmplă nimic.

Scrieți un program care citește numărul n și afișează numărul minim de comenzi necesare pentru a obține exact n caractere '-' consecutive în Vim în condițiile descrise. Programul trebuie, de asemenea, să afișeze un exemplu de secvență cu numărul minim de comenzi.

Intrare

Prima linie conține numărul de cazuri de test t . Următoarele t linii conțin cazurile de test, fiecare cu câte o valoare n , reprezentând lungimea dorită a șirului.

Restricții

- $1 \leq t \leq 100$
- $1 \leq n \leq 10^7$

Subtask-uri

Dacă afișați doar numărul corect de comenzi, dar nu și un exemplu de secvență de comenzi sau secvența afișată este incorectă, veți primi jumătate din punctajul pentru acel subtask.

- Subtask 1 (20 puncte): $n \leq 100$
- Subtask 2 (8 puncte): $n \leq 1000$
- Subtask 3 (18 puncte): $n \leq 10^4$
- Subtask 4 (18 puncte): $n \leq 10^5$
- Subtask 5 (18 puncte): $n \leq 10^6$
- Subtask 6 (18 puncte): Fără restricții suplimentare.

Ieșire

Pentru fiecare caz de test, afișați numărul minim necesar de comenzi și un exemplu al unei asemenea secvențe de comenzi, fiecare pe propria linie.

Exemplu

Intrare

```
2
21
2
```

Ieșire

```
10 YPYPhPYPPP
2 YP
```

Comentariu

După cum putem vedea în tabelul următor, în primul caz, secvența YPYPhPYPPP produce rezultatul corect. Caracterul '=' din coloana Screen reprezintă caracterul '-' pe care se află cursorul.

Step	Command	Screen	Clipboard
0		=	(empty)
1	Y	=	-
2	P	= -	-
3	Y	= -	--
4	P	= - -	--
5	h	= - - -	--
6	P	= - - - -	--
7	Y	= - - - -	-----
8	P	-----=-----	-----
9	P	-----=-----	-----
10	P	-----=-----	-----