

CEOI 2026



Dzień 2
(polski)

Day 2
(Polish)

Cięcie kwiatów

Limit czasu: 4 s Limit pamięci: 256 MiB

W ogrodzie społeczności CEOI hodujemy kolekcję specjalnych kwiatów, które ciasno splatają swoje korzenie. Jeśli je przetniemy, odrosną, o ile nie tniemy zbyt agresywnie i nie spowodujemy nieodwracalnych uszkodzeń.

Jeśli para kwiatów a i b splecie swoje korzenie, mówimy, że są „połączone”. W przeciwnym razie są „niepołączone”.

Korzenie rosną zgodnie z następującą regułą: Niech a i b będą dwoma niepołączonymi kwiatami. Jeśli istnieją co najmniej 2 inne kwiaty c i d , takie że zarówno a , jak i b są połączone zarówno z c , jak i z d , wtedy między a i b wyrosną korzenie i staną się one połączone.

Te kwiaty rosną już od pewnego czasu i wszystkie korzenie, które mogły wyrosnąć zgodnie z powyższą regułą, już wyrosły. Innymi słowy, jeśli dwa kwiaty a i b są oba połączone z pewnymi kwiatami c i d , to gwarantowane jest, że a i b są również połączone ze sobą.

Musimy teraz wykopać ten ogród i przenieść go do kolejnej lokalizacji CEOI. Aby uprościć przenoszenie, chcielibyśmy przeciąć jak najwięcej korzeni. Chcemy jednak, aby kwiaty odrosły do swojego obecnego stanu. Oblicz, jaka jest maksymalna liczba połączonych par kwiatów, których połączenia możemy przeciąć, tak aby mimo to odrosły one z powrotem do swojego obecnego stanu. Nie ma znaczenia, ile iteracji wzrostu będzie to wymagało.

Wejście

Pierwszy wiersz wejścia zawiera dwie liczby całkowite n i m oddzielone odstępem — liczbę kwiatów oraz liczbę istniejących połączeń między nimi. Następne m wierszy, zawiera po jednej parze liczb całkowitych a_i i b_i , oznaczających, że kwiaty a_i i b_i są połączone.

Kwiaty są oznaczone liczbami całkowitymi od 1 do n .

Dane wejściowe spełniają regułę opisaną w treści zadania.

Ograniczenia

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

Podzadania

- Podzadanie 1 (20 punktów): $n \leq 10$ i $m \leq 20$
- Podzadanie 2 (14 punktów): $m = \frac{n \cdot (n-1)}{2}$
- Podzadanie 3 (15 punktów): Każdy kwiat jest połączony z co najwyżej 7 innymi kwiatami.
- Podzadanie 4 (15 points): $n \leq 50$ and $m \leq 1000$
- Podzadanie 5 (36 punktów): Brak dodatkowych ograniczeń.

Wyjście

Twój program powinien wypisać na wyjście dokładnie jeden wiersz. W tym wierszu należy wypisać jedną liczbę całkowitą — maksymalną liczbę połączeń, które możemy przeciąć.

Przykład

Wejście

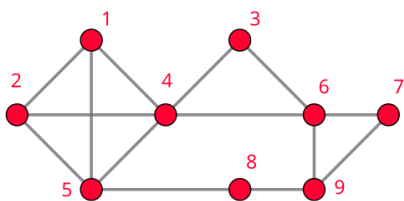
```
9 14
1 2
1 4
1 5
2 4
2 5
3 4
4 5
3 6
4 6
6 7
6 9
7 9
8 9
5 8
```

Wyjście

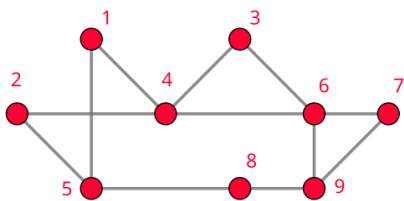
```
2
```

Wyjaśnienie

Kwiaty z przykładu przed jakimkolwiek cięciem. Można sprawdzić, że na podstawie opisanego procesu wzrostu nie mogą powstać żadne dodatkowe połączenia.



Kwiaty z przykładu po cięciu mają o 2 połączenia mniej. Połączenie między 1 i 2 może odrosnąć, ponieważ kwiaty 1 i 2 są oba połączone z kwiatami 4 i 5. Podobnie połączenie między 4 i 5 może odrosnąć, ponieważ kwiaty 4 i 5 są oba połączone z kwiatami 1 i 2.



Wieże

Limit czasu: 1 s Limit pamięci: 256 MiB

Masz n komputerów i m wież, wszystkie znajdujące się w różnych pozycjach na prostej. Musisz sparować komputery za pomocą kabli w taki sposób, aby każdy kabel zaczynał się przy pewnym komputerze, odwiedzał pewne wieże i kończył się przy innym komputerze. Kabel może odwiedzać dowolne wieże (nie tylko te znajdujące się między łączonymi komputerami) w dowolnej kolejności. Może pomijać wieże, przechodząc obok nich bez ich odwiedzania. Może również nie odwiedzać żadnej wieży i połączyć oba komputery bezpośrednio, ale nie może połączyć komputera z samym sobą. Liczba komputerów jest parzysta.

Niech a i b będą pozycjami dwóch komputerów, a $x_1, \dots, x_k$ pozycjami wież odwiedzanych przez kabel. Długość kabla wynosi $|a - x_1| + |x_1 - x_2| + \dots + |x_{k-1} - x_k| + |x_k - b|$. Dla danego kabla definiujemy jego rentowność jako $f \cdot u - l$, gdzie l jest jego długością, f jest ustaloną stałą, a u jest liczbą unikalnych wież odwiedzonych przez kabel spośród $x_1, \dots, x_k$. Wiele kabli może odwiedzać tę samą wieżę i taka wieża wnosi wkład do rentowności każdego z tych kabli.

Musisz obliczyć maksymalną możliwą sumę rentowności kabli użytych do sparowania wszystkich komputerów (tzn. każdy komputer musi należeć do dokładnie jednej pary lub równoważnie musi być połączony dokładnie jednym kablem).

Wejście

Pierwszy wiersz wejścia zawiera liczbę zestawów testowych T . Następnie opisane są kolejne zestawy testowe. Każdy zestaw testowy składa się z trzech wierszy. Pierwszy wiersz zawiera trzy liczby całkowite n , m i f — liczbę komputerów, liczbę wież oraz stałą f . Drugi wiersz zawiera n liczb całkowitych $a_1, a_2, \dots, a_n$ — pozycje komputerów. Trzeci wiersz zawiera m liczb całkowitych $b_1, b_2, \dots, b_m$ — pozycje wież.

Ograniczenia

Niech N i M oznaczają odpowiednio sumy wartości n i m we wszystkich zestawach testowych.

- $1 \leq T \leq 10^4$
- $1 \leq N, M \leq 2 \cdot 10^5$
- $0 \leq f \leq 10^9$
- n jest parzyste.
- $1 \leq a_i, b_i \leq 10^9$
- Wszystkie pozycje komputerów i wież są unikalne (w obrębie pojedynczego zestawu testowego).

Podzadania

- Podzadanie 1 (5 punktów): $N \leq 5000$, $m = 1$

- Podzadanie 2 (10 punktów): $T \leq 20$, $n \leq 10$, $m \leq 100$
- Podzadanie 3 (27 punktów): $N, M \leq 5000$
- Podzadanie 4 (21 punktów): $N \leq 5000$
- Podzadanie 5 (37 punktów): Brak dodatkowych ograniczeń.

Wyjście

Twój program powinien wypisać na wyjście dokładnie T wierszy. W i -tym z tych wierszy należy wypisać jedną liczbę całkowitą — maksymalną możliwą sumę rentowności kabli dla i -tego zestawu testowego.

Przykład

Wejście

```
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
```

Wyjście

```
89
-4
12
51
```

Vim

Limit czasu: 1 s Limit pamięci: 256 MiB

Trudno byłoby znaleźć coś bardziej *mózgożernego* niż stary dobry Vim — edytor, w którym możemy robić rzeczy, o których w „zwykłych” edytorach możemy tylko marzyć. Oczywiście pod warunkiem, że jesteśmy gotowi poświęcić tydzień na naukę i miesiąc na poligonie, aby nietypowe, ale potężne polecenia, które oferuje, stały się częścią naszej „pamięci mięśniowej”. Cóż, podobnie jak w innych edytorach, w Vimie mamy *kursor*, który zawsze znajduje się na jakimś znaku. Początkowo kursor znajduje się na pierwszym znaku tekstu. Vim ma także *schowek* (clipboard), przeznaczony do przechowywania (i pośrednio kopiowania oraz przenoszenia) fragmentów tekstu. Schowek jest początkowo pusty.

W tym zadaniu założymy, że na początku w edytorze znajduje się dokładnie jeden znak '-' (minus), a naszym celem jest uzyskanie dokładnie n kolejnych znaków minus. Pomogą nam w tym cztery polecenia:

- **h**: Jeśli kursor znajduje się na pierwszym znaku, to polecenie nic nie robi, w przeciwnym razie przesuwa kursor o jeden znak w lewo.
- **l**: Jeśli kursor znajduje się na ostatnim znaku, to polecenie nic nie robi, w przeciwnym razie przesuwa kursor o jeden znak w prawo.
- **Y**: Sekwencja znaków od pozycji kursora do końca tekstu jest kopiowana do schowka, „nadpisując” jego poprzednią zawartość. (Jeśli kursor znajduje się na pierwszym znaku tekstu, do schowka kopiowany jest cały tekst.)
- **P**: Kopia tekstu przechowywanego w schowku jest wstawiana przed znak, na którym znajduje się kursor, a kursor zostaje przesunięty na ostatni wstawiony znak. Zawartość schowka nie ulega zmianie. Jeśli schowek jest pusty, nic się nie dzieje.

Napisz program, który wczytuje liczbę n i wypisuje minimalną liczbę poleceń potrzebnych do uzyskania dokładnie n kolejnych znaków '-' w Vimie przy opisanych warunkach. Program powinien również wypisać przykład sekwencji poleceń o minimalnej długości.

Wejście

Pierwszy wiersz wejścia zawiera liczbę zestawów testowych t . Kolejne t wierszy zawiera zestawy testowe z żądaną długością ciągu n .

Ograniczenia

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

Podzadania

Jeśli wypiszesz tylko poprawną liczbę poleceń, ale nie przykład sekwencji poleceń lub wypisany przykład będzie błędny, otrzymasz połowę punktów za dane podzadanie.

- Podzadanie 1 (20 punktów): $n \leq 100$
- Podzadanie 2 (8 punktów): $n \leq 1000$
- Podzadanie 3 (18 punktów): $n \leq 10^4$
- Podzadanie 4 (18 punktów): $n \leq 10^5$
- Podzadanie 5 (18 punktów): $n \leq 10^6$
- Podzadanie 6 (18 punktów): Brak dodatkowych ograniczeń.

Wyjście

Twój program powinien wypisać na wyjście dokładnie t wierszy. W i -tym z tych wierszy należy wypisać wymaganą minimalną liczbę poleceń dla i -tego zestawu oraz przykład takiej sekwencji poleceń oddzielony pojedynczym odstępem.

Przykład

Wejście

```
2
21
2
```

Wyjście

```
10 YPYPhPYPPP
2 YP
```

Komentarz

Jak możemy zobaczyć w poniższej tabeli, w pierwszym przypadku sekwencja YPYPhPYPPP daje poprawny wynik. Znak '=' w kolumnie Screen reprezentuje znak '-', na którym znajduje się kursor.

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	-----=-----	-----