

Magiczne Miasto

Burmistrz Taszkientu chce przebudować słynny park rozrywki Magic City. Otrzymujesz dodatnią liczbę całkowitą K i Twoim zadaniem jest zaprojektowanie parku w następujący sposób:

- Wybierz liczbę całkowitą N oznaczającą liczbę atrakcji. Atrakcje są oznaczone liczbami całkowitymi od 0 do $N - 1$.
- Dodaj dwukierunkowe przejścia, każda łącząca parę **różnych** atrakcji. Dozwolone jest istnienie więcej niż jednego przejścia pomiędzy tymi samymi atrakcjami. **Nie jest wymagana możliwość przemieszczania się między każdą parą atrakcji za pomocą przejść.**
- Dla każdego i takiego, że $0 \leq i < N$, przypisz atrakcji i typ $T[i]$. Istnieje $2K$ typów ponumerowanych od 0 do $2K - 1$. Każdy typ powinien być przypisany do co najmniej jednej atrakcji. Różne atrakcje mogą mieć ten sam typ.

Badania rynku wykazały, że:

- Każdy turysta chce odwiedzić trzy atrakcje unikając odwiedzenia dwóch atrakcji tego samego typu jednej po drugiej.
- Turyści nie lubią atrakcji, które stanowią jeden z końców wielu przejść.

Aby zadowolić **wszystkich** potencjalnych gości, burmistrz stawia projektowi dwa warunki.

Nazywamy uporządkowaną trójkę **typów** (t_1, t_2, t_3) dla $0 \leq t_1, t_2, t_3 < 2K$ **ciekawą**, jeśli $t_1 \neq t_2$ i $t_2 \neq t_3$. Zauważ, że t_1 może być równe t_3 . Istnieje zatem $2K \cdot (2K - 1)^2$ interesujących trójek.

Warunek 1: Dla każdej interesującej trójki (t_1, t_2, t_3) muszą istnieć trzy atrakcje a_1, a_2, a_3 (gdzie $0 \leq a_1, a_2, a_3 < N$) takie, że:

- Typy a_1, a_2, a_3 są zgodne z t_1, t_2, t_3 . Oznacza to, że $T[a_1] = t_1$, $T[a_2] = t_2$ i $T[a_3] = t_3$.
- Istnieje przejście pomiędzy a_1 i a_2 .
- Istnieje przejście pomiędzy a_2 i a_3 .

Nie ma znaczenia, czy między a_1 i a_3 istnieje przejście, czy nie. Należy również zauważyć, że gdy $t_1 = t_3$, atrakcje a_1 i a_3 mogą być takie same.

Warunek 2: Każda atrakcja może być jednym z końców **maksymalnie** K ścieżek.

To zadanie składa się z 50 podzadań **typu output-only z częściową punktacją**. Każde podzadanie odpowiada określonej wartości K i Twoim zadaniem jest zaprojektowanie parku spełniającego wszystkie powyższe warunki dla tej wartości K . Twój wynik zależy od liczby atrakcji w rozwiązaniu: mniej atrakcji odpowiada wyższemu lub równemu wynikowi.

Szczegóły implementacji

Istnieją dwie metody przesyłania rozwiązań. Dla każdego podzadania możesz użyć dowolnej z nich:

- **Wywołanie procedury**
- **Plik wyjściowy**

Aby przesłać swoje rozwiązanie za pośrednictwem **wywołania procedury**, należy zaimplementować następującą procedurę:

```
std::pair<std::vector<int>, std::vector<std::pair<int, int>>>>  
construct(int K)
```

- K : połowa liczby typów atrakcji, a także maksymalna liczba ścieżek, których jednym z końców może być każda atrakcja.
- Ta procedura jest wywoływana dokładnie raz dla każdego podzadania.

Procedura powinna zwrócić parę (T, E) opisującą park rozrywki. Niech M będzie liczbą ścieżek w Twoim parku.

- T : tablica o długości N opisująca typy atrakcji.
- E : tablica o długości M opisująca przejścia. Dla każdego $0 \leq j < M$, $E[j] = (U[j], V[j])$ reprezentuje dwukierunkowe przejście między różnymi atrakcjami $U[j]$ i $V[j]$.

Aby przesłać swoje rozwiązanie za pośrednictwem **pliku wyjściowego**, utwórz i prześlij plik tekstowy w następującym formacie:

```
N M  
T[0] T[1] ... T[N-1]  
U[0] V[0]  
U[1] V[1]  
...  
U[M-1] V[M-1]
```

Należy pamiętać, że aby Twoje rozwiązanie zostało uznane za **prawidłowe**, musi ono spełniać następujące ograniczenia:

- $N \leq 2000$
- $0 \leq T[i] < 2K$ dla każdego $0 \leq i < N$, a każdy typ powinien być przypisany do co najmniej jednej atrakcji.
- $0 \leq U[j], V[j] < N$ i $U[j] \neq V[j]$ dla każdego $0 \leq j < M$.
- Warunki 1 i 2 muszą być spełnione.

Ograniczenia

- $1 \leq K \leq 50$

Punktacja

Istnieje 50 podzadań, po jednym dla każdej liczby całkowitej K od 1 do 50. Dla podzadania i ($1 \leq i \leq 50$) wartość K wynosi i .

Każde podzadanie ma punktację S i docelową liczbę atrakcji P zgodnie z poniższą tabelą.

Podzadanie	S	P
1	1	2
2	8	12
3	9	24
4	9	40
5	9	50
6 - 10	4	$12 \cdot K$
11 - 12	3	$12 \cdot K$
13 - 50	1	$12 \cdot K$

Jeśli w przypadku każdego podzadania Twoje rozwiązanie nie opisuje prawidłowego parku rozrywki, wówczas wynik rozwiązania będzie wynosić 0 (w CMS będzie wyświetlany komunikat `Output isn't correct`).

W przeciwnym wypadku Twój wynik obliczany jest na podstawie N oraz parametrów S i P w następujący sposób:

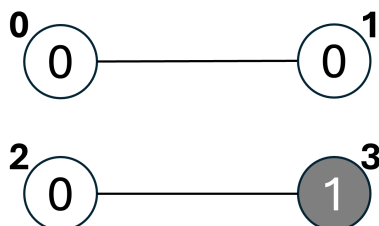
Warunek	Punkty
$N \leq P$	S
$P < N \leq 2P$	$(0.4 + 0.3 \cdot \frac{2P-N}{P}) \cdot S$
$2P < N \leq 2000$	$(0.1 + 0.3 \cdot \frac{2P}{N}) \cdot S$

Przykład

Rozważmy następujące wywołanie:

```
construct (1)
```

W tym przykładzie $K = 1$, więc istnieją $2K = 2$ typy atrakcji. Poniższy rysunek przedstawia prawidłowe rozwiązanie dla $N = 4$ atrakcji i $M = 2$ ścieżek. Atrakcje 0, 1, 2 są typu 0, a atrakcja 3 jest typu 1.



Istnieją dwie interesujące trójki:

- Dla trójki typów $(0, 1, 0)$ możemy wybrać $(a_1, a_2, a_3) = (2, 3, 2)$.
- Dla trójki typów $(1, 0, 1)$ możemy wybrać $(a_1, a_2, a_3) = (3, 2, 3)$. Oznacza to, że Warunek 1 jest spełniony.

Procedura może zwrócić parę $([0, 0, 0, 1], [(0, 1), (2, 3)])$. Należy pamiętać, że podane rozwiązanie dla tego przykładu może nie być optymalne dla $K = 1$.

Przykładowy program sprawdzający

Format wejścia:

```
K
```

Format wyjścia:

```
N M
T[0] T[1] ... T[N-1]
U[0] V[0]
U[1] V[1]
...
U[M-1] V[M-1]
```

Zwróć uwagę, że dane wyjściowe przykładowego programu sprawdzającego są zgodne z wymaganym formatem pliku wyjściowego.