

## Zbieranie diamentów

 3 sek.  4 MB

W Bajtockich górach odkryto kopalnię diamentów. Dla uproszczenia można założyć, że kopalnia ma  $N$  komnat ponumerowanych liczbami całkowitymi od 0 do  $N - 1$ . Komnaty połączone są  $M$  jednokierunkowymi korytarzami w taki sposób, że z każdej komnaty wychodzi przynajmniej jeden korytarz. W każdym korytarzu znajduje się pewna liczba diamentów, które można zebrać, jeżeli przejdzie się danym korytarzem. Ta liczba diamentów **nie zmienia się** po przejściu danym korytarzem – pozostaje taka sama dla kolejnych przejść.

Możliwe jest, że korytarz łączy komnatę z nią samą lub, że wiele korytarzy łączy tę samą parę komnat (być może również w tym samym kierunku). Ponadto, nie jest gwarantowane, że komnaty są spójne – może istnieć para komnat  $(x, y)$ , że nie można dostać się z  $x$  do  $y$ .

Piotr przejdzie  $K$  korytarzami, aby zebrać diamenty. Wybierze pewną komnatę  $s$ , w której zacznie. Następnie przejdzie do kolejnej komnaty korytarzem wychodzącym z komnaty  $s$  i będzie to powtarzał, dopóki nie przejdzie  $K$  korytarzy. Zwróć uwagę, że komnaty i korytarze, którymi chodzi, mogą się powtarzać oraz liczba diamentów zebranych w korytarzu nie zmienia się mimo wielokrotnego przechodzenia tym korytarzem. Zwróć uwagę, że zawsze będzie mógł przejść przez  $K$  korytarzy.

Piotr wybierze  $s$  oraz ścieżkę korytarzy w następujący sposób: Najpierw chce zmaksymalizować liczbę diamentów, które zbierze w pierwszym korytarzu, który przejdzie. Wśród wszystkich tych opcji wybierze tę, która maksymalizuje liczbę diamentów, które zbierze w drugim korytarzu. I tak powtórzy  $K$  razy. Innymi słowy, Piotr chce wybrać leksykograficznie największą ścieżkę. Pomóż mu i policz ile diamentów zbierze, jeżeli wybierze ścieżkę zgodnie z opisem.



### Szczegóły implementacji

Należy zaimplementować funkcję `calculate_diamonds`:

```
long long int calculate_diamonds(int N, int M, int K,  
    std::vector<int> u, std::vector<int> v, std::vector<int> d)
```

- $N$ : liczba komnat w kopalni diamentów;
- $M$ : liczba korytarzy między komnatami;
- $K$ : liczba korytarzy, które przejdzie Piotr;
- $u, v, d$ : wektory mające  $M$  liczb całkowitych, opisujące komnaty startowe, komnaty końcowe i liczby diamentów kolejnych korytarzy.

Ta funkcja zostanie wywołana raz dla każdego testu i powinna zwrócić jedną liczbę –



sumaryczną liczbę diamentów, które zbierze Piotr używając jego strategii.



## Ograniczenia

- $1 \leq N \leq 2\,000$
- $1 \leq M \leq 4\,000$
- $1 \leq K \leq 10^9$
- $0 \leq u[i], v[i] < N$
- $1 \leq d[i] \leq 10^9$  dla każdego  $0 \leq i < M$
- Gwarantowane jest, że z każdej komnaty wychodzi przynajmniej jeden korytarz.
- **Zwróć uwagę na mały limit pamięci: 4 MB.**



## Podzadania

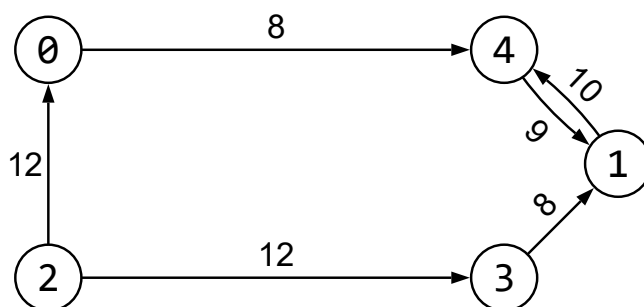
Podzadanie	Punkty	Wymagane podzadania	$N$	$M$	$K$	Dodatkowe ograniczenia
0	0	—	—	—	—	Przykłady.
1	11	0	$\leq 10$	$\leq 20$	$\leq 10$	—
2	10	0 — 1	$\leq 100$	$\leq 1\,000$	$\leq 1000$	—
3	26	0 — 2	$\leq 100$	$\leq 1\,000$	$\leq 10^9$	—
4	11	—	$\leq 2\,000$	$= N$	$\leq 10^9$	Każda komnata ma dokładnie jeden korytarz zaczynający się w niej oraz dokładnie jeden korytarz kończący się w niej.
5	10	—	$\leq 2\,000$	$\leq 4\,000$	$\leq 10^9$	Wszystkie $d[i]$ są różne.
6	11	—	$\leq 2\,000$	$\leq 4\,000$	$\leq 10^9$	Jest dokładnie jedno $d[i] = 2$ ( $0 \leq i < M$ ) oraz wszystkie pozostałe wartości $d$ wynoszą 1.
7	21	0 — 6	$\leq 2\,000$	$\leq 4\,000$	$\leq 10^9$	—



## Przykład 1

Rozważmy następujące wywołanie oraz rysunek dla  $N = 5$ ,  $M = 6$  i  $K = 4$ :

```
calculate_diamonds(5, 6, 4,
    {2, 0, 4, 2, 3, 1}, {0, 4, 1, 3, 1, 4}, {12, 8, 9, 12, 8, 10})
```

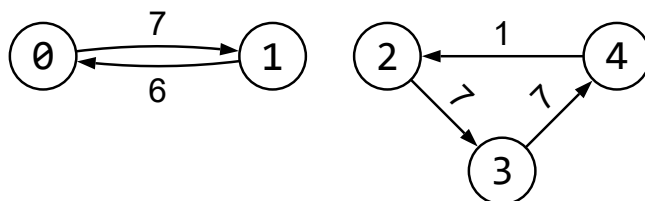


Piotr wybierze do przejścia następujące korytarze:  $2 \xrightarrow{12} 3 \xrightarrow{8} 1 \xrightarrow{10} 4 \xrightarrow{9} 1$ . Sumaryczna liczba diamentów, które zbierze to 39, co powinno być wynikiem wywołania funkcji.

## Przykład 2

Rozważmy następujące wywołanie oraz rysunek dla  $N = 5$ ,  $M = 5$  i  $K = 4$ :

```
calculate_diamonds(5, 5, 4,
    {0, 1, 2, 3, 4}, {1, 0, 3, 4, 2}, {7, 6, 7, 7, 1})
```



Jest 5 opcji, aby przejść 4 korytarze:

- (1)  $0 \xrightarrow{7} 1 \xrightarrow{6} 0 \xrightarrow{7} 1 \xrightarrow{6} 0$ ;
- (2)  $1 \xrightarrow{6} 0 \xrightarrow{7} 1 \xrightarrow{6} 0 \xrightarrow{7} 1$ ;
- (3)  $2 \xrightarrow{7} 3 \xrightarrow{1} 4 \xrightarrow{7} 2 \xrightarrow{7} 3$ ;
- (4)  $3 \xrightarrow{7} 4 \xrightarrow{1} 2 \xrightarrow{7} 3 \xrightarrow{7} 4$ ;
- (5)  $4 \xrightarrow{1} 2 \xrightarrow{7} 3 \xrightarrow{7} 4 \xrightarrow{1} 2$ .

Opcje (2) i (5) nie maksymalizują liczby diamentów w pierwszym korytarzu. Wśród opcji (1), (3) i (4), tylko (3) maksymalizuje liczbę diamentów w drugim korytarzu, zatem jest to najlepsza opcja dla Piotra. Zauważ, że opcja (3) nie maksymalizuje liczby diamentów w trzecim korytarzu ani nie maksymalizuje sumarycznej liczby diamentów, jednak jest to jedyna największa leksykograficznie ścieżka. Sumaryczna liczba diamentów, które zbierze Piotr to 22, co powinno być wynikiem wywołania funkcji.

## Przykładowa biblioteczka

Format wejścia jest następujący:

- wiersz 1: trzy liczby całkowite – wartości  $N$ ,  $M$  oraz  $K$ .
- wiersz  $1 + i$ : trzy liczby całkowite  $u[i]$ ,  $v[i]$ ,  $d[i]$  – opisujące korytarz zaczynający się w



komnacie  $u[i]$  i kończący się w komnacie  $v[i]$  mający  $d[i]$  diamentów.

Format wyjścia jest następujący:

- wiersz 1: jedna liczba całkowita – zwrócona wartość przez twoją funkcję.