Jam Tutup

Hungaria adalah sebuah negara dengan N kota, dinomori dari 0 sampai $N-1$.

Kota-kota tersebut dihubungkan oleh $N-1$ jalan *dua arah*, dinomori dari 0 sampai $N-2$. Untuk setiap j sedemikian sehingga $0 \leq j \leq N-2$, jalan j menghubungkan kota $U[j]$ dan kota $V[j]$ dan mempunyai panjang $W[j]$. Dengan kata lain, perjalanan antara kedua kota tersebut ditempuh dalam $W[j]$ unit waktu. Setiap jalan menghubungkan dua kota berbeda, dan setiap pasang kota terhubung oleh paling banyak satu jalan.

Sebuah **jalur** antara dua kota berbeda a dan b adalah sebuah barisan $p_0, p_1, \ldots, p_t$, sedemikian sehingga:

- $p_0 = a$
- $p_1 = b$
- untuk setiap i ($0 \leq i < t$), terdapat sebuah jalan yang menghubungkan kota p_i dan p_{i+1}.

Terdapat cara untuk berpindah dari kota mana pun ke kota lain mana pun menggunakan jalan-jalan tersebut. Dengan kata lain, terdapat sebuah jalur yang menghubungkan setiap dua kota berbeda. Dapat dibuktipkan bahwa jalur ini unik untuk setiap pasang kota berbeda.

Panjang dari jalur $p_0, p_1, \ldots, p_t$ adalah total panjang dari t jalan yang menghubungkan kota-kota konsekutif pada jalur.

Di Hungaria, banyak orang bepergian untuk menghadiri perayaan *Foundation Day* di suatu kota-kota besar. Saat perayaan berakhir, mereka akan kembali ke rumah masing-masing. Pemerintah mereka ingin mencegah agar kerumunan tidak mengganggun orang-orang lokal, sehingga mereka merencanakan penguncian pada kota-kota tersebut pada waktu-waktu tertentu. Setiap kota akan diberi **jam tutup** nonnegatif oleh pemerintahnya. Pemerintah sudah menetapkan bahwa jumlah dari semua jam tutup harus tidak melebihi K. Lebih tepatnya, untuk setiap i antara 0 dan $N-1$, inklusif, jam tutup yang diberikan ke kota i adalah bilangan bulat nonnegatif $c[i]$. Jumlah dari semua $c[i]$ tidak boleh melebihi K.

Perhatikan sebuah kota a dan suatu pemberian jam tutup. Kita katakan bahwa kota b **tercapai** dari kota a jika dan hanya jika $b = a$, atau jalur $p_0, \ldots, p_t$ antara kedua kota tersebut (lebih tepatnya $p_0 = a$ dan $p_t = b$) memenuhi kondisi-kondisi berikut:

- panjang dari jalur p_0, p_1 adalah paling panjang $c[p_1]$, dan
- panjang dari jalur p_0, p_1, p_2 adalah paling panjang $c[p_2]$, dan

- ...
- panjang dari lajur $p_0, p_1, p_2, \ldots, p_t$ adalah paling panjang $c[p_t]$.

Tahun ini, dua perayaan utama berada di kota X dan kota Y. Untuk setiap pemberian jam tutup, **nilai kenyamanan** didefinisikan sebagai jumlah dari dua nilai berikut:

- Banyaknya kota yang tercapai dari kota X.
- Banyaknya kota yang tercapai dari kota Y.

Perhatikan bahwa jika sebuah kota tercapai dari kota X dan tercapai dari kota Y, maka kota tersebut terhitung *dua kali* terhadap nilai kenyamanan.

Tugas Anda adalah menghitung nilai kenyamanan maksimum yang dapat dicapai dengan suatu pemberian jam tutup.

Detail Implementasi

Anda harus mengimplementasikan prosedur berikut.

```
int max_score(int N, int X, int Y, int64 K, int[] U, int[] V, int[] W)
```

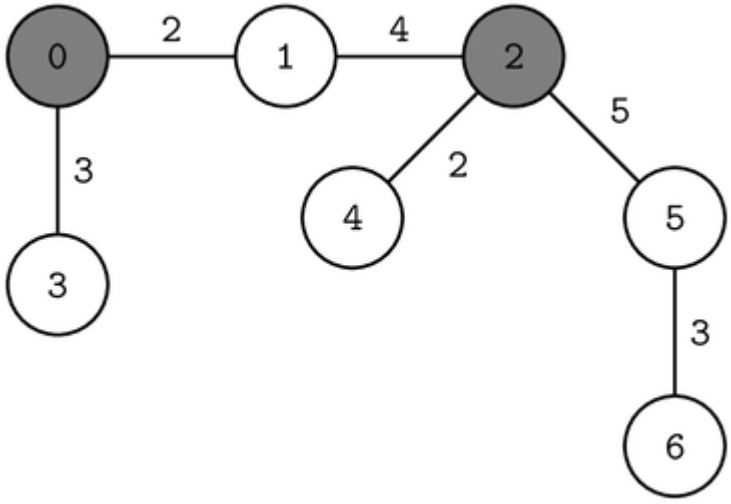
- N: banyaknya kota.
- X, Y: kota-kota dengan perayaan utama.
- K: batas atas dari jumlah jam tutup.
- U, V: *arrays* sepanjang $N-1$ yang mendeskripsikan describing hubungan jalan-jalan.
- W: *array* sepanjang $N-1$ yang mendeskripsikan panjang-panjang jalan.
- Prosedur ini harus mengembalikan nilai kenyamanan maksimum yang bisa dicapai dengan suatu pemberian jam tutup.
- Prosedur ini bisa dipanggil **berkali-kali** dalam satu kasus uji.

Contoh

Perhatikan pemanggilan berikut:

```
max_score(7, 0, 2, 10,
          [0, 0, 1, 2, 2, 5], [1, 3, 2, 4, 5, 6], [2, 3, 4, 2, 5, 3])
```

Ini sesuai dengan jaringan jalan berikut:



Misalkan jam tutup diberikan sebagai berikut:

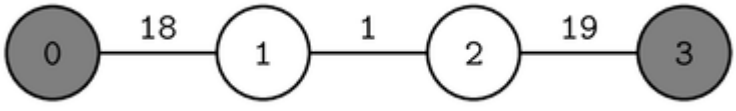
Kota	0	1	2	3	4	5	6
Jam tutup	0	4	0	3	2	0	0

Perhatikan bahwa jumlah dari semua jam tutup adalah 9, yang tidak lebih dari $K = 10$. Kota 0, 1, dan 3 tercapai dari kota X ($X = 0$), dan kota-kota 1, 2, dan 4 tercapai dari kota Y ($Y = 2$). Maka dari itu, nilai kenyamanan adalah $3 + 3 = 6$. Tidak terdapat pemberian jam tutup dengan nilai kenyamanan lebih dari 6, sehingga prosedur harus mengembalikan 6.

Perhatikan juga pemanggilan berikut:

```
max_score(4, 0, 3, 20, [0, 1, 2], [1, 2, 3], [18, 1, 19])
```

Ini sesuai dengan jaringan jalan berikut:



Misalkan jam tutup diberikan sebagai berikut:

Kota	0	1	2	3
Jam tutup	0	1	19	0

Kota 0 tercapai dari kota X ($X = 0$), dan kota 2 dan 3 tercapai dari kota Y ($Y = 3$). Maka dari itu, nilai kenyamanan adalah $1 + 2 = 3$. Tidak terdapat pemberian jam tutup dengan nilai

kenyamanan yang lebih dari 3, sehingga prosedur harus mengembalikan 3.

Batasan

- $2 \leq N \leq 200\,000$
- $0 \leq X < Y < N$
- $0 \leq K \leq 10^{18}$
- $0 \leq U[j] < V[j] < N$ (untuk setiap j sedemikian sehingga $0 \leq j \leq N-2$)
- $1 \leq W[j] \leq 10^6$ (untuk setiap j sedemikian sehingga $0 \leq j \leq N-2$)
- Terdapat cara untuk berpindah dari kota mana pun ke kota mana pun lainnya menggunakan jalan.
- $S_N \leq 200\,000$, dengan S_N adalah jumlah dari N dari semua pemanggilan `max_score`.

Subsoal

Kita katakan sebuah jaringan jalan sebagai **linear** jika kota i menghubungkan i dan $i+1$ (untuk setiap i sedemikian sehingga $0 \leq i \leq N-2$).

1. (8 poin) The length of the path from city X to city Y is greater than $2K$.
2. (9 poin) $S_N \leq 50$, jaringan jalan adalah linear.
3. (12 poin) $S_N \leq 500$, jaringan jalan adalah linear.
4. (14 poin) $S_N \leq 3\,000$, jaringan jalan adalah linear.
5. (9 poin) $S_N \leq 20$
6. (11 poin) $S_N \leq 100$
7. (10 poin) $S_N \leq 500$
8. (10 poin) $S_N \leq 3\,000$
9. (17 poin) Tidak ada batasan tambahan.

Contoh *Grader*

Misalkan C menyatakan banyaknya skenario, yaitu banyaknya pemanggilan `max_score`. Contoh *grader* membaca masukan dengan format berikut:

- baris 1: C

Deskripsi dari C skenario adalah sebagai berikut.

Contoh *grader* membaca deskripsi dari setiap skenario dengan format berikut:

- baris 1: $N\ X\ Y\ K$
- baris $2+j$ ($0 \leq j \leq N-2$): $U[j]\ V[j]\ W[j]$

Contoh *grader* mencetak sebuah baris untuk setiap skenario, dengan format berikut:

- baris 1: nilai kembali dari `max_score`