Orari di chiusura

L'Ungheria può essere rappresentata come un albero di N città, numerate da 0 a $N-1$. Le città sono collegate da $N-1$ strade *bidirezionali*, numerate da 0 a $N-2$. Per ogni $0 \leq j \leq N-2$, la strada j permette di viaggiare tra le città $U[j]$ e $V[j]$, e ha lunghezza $W[j]$. Ogni strada collega due città diverse e non esistono archi duplicati.

Un percorso tra due città a e b è una sequenza di città distinte $p_0, p_1, \ldots, p_t$, con $p_0 = a$ e $p_t = b$, tale che ogni coppia di città consecutive è collegata da un arco. Esiste un unico percorso tra ogni coppia di città.

Il Giorno della Fondazione è un grande evento in Ungheria, che si svolgerà nelle città X e Y, a cui accorreranno molte persone. Finito l'evento, tutti i viaggiatori devono tornare a casa.

Per evitare di disturbare i residenti, il governo vuole chiudere le città a partire da determinati orari: ad ogni città i viene assegnato un **orario di chiusura** $c[i]$ intero non negativo. È stato deciso che la somma di tutti gli orari di chiusura $c[i]$ non deve essere maggiore di K.

Considera una città a e un qualche assegnamento valido di orari di chiusura. Diciamo che la città b è **raggiungibile** da a se e solo se $b = a$ oppure se il percorso $p_0, \ldots, p_t$ tra le due città (quindi con $p_0 = a$ e $p_t = b$) soddisfa tutte le seguenti condizioni:

- La lunghezza del percorso p_0, p_1 è al più $c[p_1]$;
- La lunghezza del percorso p_0, p_1, p_2 è al più $c[p_2]$;
- $\ldots$
- La lunghezza del percorso $p_0, p_1, p_2, \ldots, p_t$ è al più $c[p_t]$.

Per una qualche assegnazione di orari di chiusura, la **convenienza** è definita come la somma di due valori:

- Il numero di città raggiungibili dalla città X;
- Il numero di città raggiungibili dalla città Y.

Se una città è raggiungibile sia da X che da Y, va contata *due volte* nel calcolo della convenienza.

Il tuo compito è quello di calcolare la massima possibile convenienza che può essere ottenuta con una qualche assegnazione di orari di chiusura.

Note di implementazione

Devi implementare la seguente funzione:

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

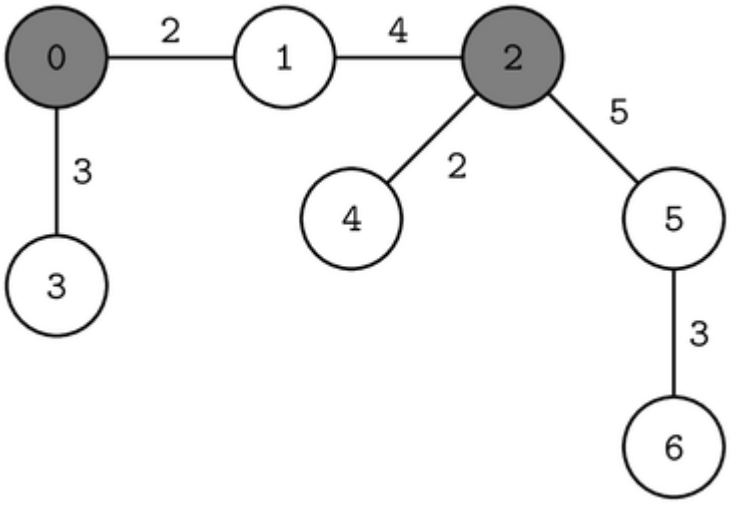
- N: il numero di città.
- X, Y: le città dove si svolgeranno i due eventi principali.
- K: la massima somma di orari di chiusura permessa.
- U, V: array di lunghezza $N-1$ che contengono le due città collegate da ogni strada.
- W: array di lunghezza $N-1$ che contiene le lunghezze di ogni strada.
- Questa funzione deve restituire la massima convenienza che può essere ottenuta per qualche assegnazione degli orari di chiusura.
- Questa funzione può essere chiamata **più volte** nello stesso testcase.

Esempi

Considera la seguente chiamata:

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

che corrisponde alla seguente rete stradale:



Assumi che questi siano gli orari di chiusura:

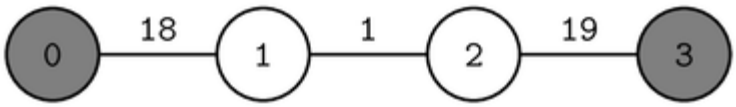
Città	0	1	2	3	4	5	6
Orario di chiusura	0	4	0	3	2	0	0

La somma di tutti gli orari di chiusura è 9, che non è maggiore di $K = 10$. Le città 0, 1 e 3 sono raggiungibili da $X = 0$, mentre le città 1, 2 e 4 sono raggiungibili da $Y = 2$. La convenienza totale è quindi $3 + 3 = 6$. Non esiste un'assegnazione di orari di chiusura con convenienza maggiore di 6, quindi la funzione deve restituire 6.

Considera ora la seguente chiamata:

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

che corrisponde alla seguente rete stradale:



Assumi che questi siano gli orari di chiusura:

Città	0	1	2	3
Orario di chiusura	0	1	19	0

La città 0 è raggiungibile da $X = 0$, mentre le città 2 e 3 sono raggiungibili da $Y = 3$. La convenienza totale è quindi $1 + 2 = 3$. Non esiste un'assegnazione di orari di chiusura con convenienza maggiore di 3, quindi la funzione deve restituire 3.

Assunzioni

- $2 \leq N \leq 200\,000$.
- $0 \leq X < Y < N$.
- $0 \leq K \leq 10^{18}$.
- $0 \leq U[j] < V[j] < N$ (per ogni $0 \leq j \leq N - 2$).
- $1 \leq W[j] \leq 10^6$ (per ogni $0 \leq j \leq N - 2$).
- È possibile viaggiare tra qualsiasi coppia di città usando le strade fornite.
- $S_N \leq 200\,000$, dove S_N è la somma di N tra tutte le chiamate a `max_score` in un testcase.

Subtask

Definiamo una rete stradale una **linea** se la strada i collega le città i e $i+1$, per ogni $0 \leq i \leq N-2$.

1. (8 punti) La distanza tra X e Y è maggiore di $2K$.
2. (9 punti) $S_N \leq 50$ e la rete stradale è una linea.
3. (12 punti) $S_N \leq 500$ e la rete stradale è una linea.
4. (14 punti) $S_N \leq 3\,000$ e la rete stradale è una linea.
5. (9 punti) $S_N \leq 20$.
6. (11 punti) $S_N \leq 100$.
7. (10 punti) $S_N \leq 500$.
8. (10 punti) $S_N \leq 3\,000$.
9. (17 punti) Nessuna limitazione aggiuntiva.

Grader di esempio

Sia C il numero di scenari, ovvero il numero di chiamate a `max_score`. Il grader di esempio legge l'input nel seguente formato:

- riga 1: C

Segue la descrizione di C scenari. Ogni scenario è composto da:

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

Il grader di esempio stampa una riga per ogni scenario, nel seguente formato:

- riga 1: il valore restituito da `max_score`