Closing Time

Ungaria este o țară cu N orașe, numerotate de la 0 la $N-1$.

Orașele sunt conectate prin $N-1$ drumuri *bidirecționale*, numerotate de la 0 la $N-2$. Pentru fiecare j, $0 \leq j \leq N-2$, drumul j conectează orașul $U[j]$ cu orașul $V[j]$ și are lungimea $W[j]$, adică permite călătoria între orașele $U[j]$ și $V[j]$ în $W[j]$ unități de timp. Fiecare drum conectează două orașe diferite și fiecare pereche de orașe este conectată de cel mult un drum.

Un **itinerar** dintre două orașe distincte a și b este o secvență $p_0, p_1, \ldots, p_t$ de orașe distincte, astfel încât:

- $p_0 = a$,
- $p_t = b$,
- pentru fiecare i ($0 \leq i < t$), există un drum care conectează orașele p_i și p_{i+1}.

Există posibilitatea de a călători din orice oraș în orice alt oraș, adică există un itinerar între orice două orașe distincte.

Se poate demonstra că acest itinerar este unic pentru fiecare pereche de orașe distincte.

Lungimea unui itinerar $p_0, p_1, \ldots, p_t$ este suma lungimilor a t drumuri care conectează consecutiv orașele de-a lungul itinerarului.

În Ungaria, mulți oameni călătoresc pentru a participa la festivitățile de Ziua Fundației în două cele mai mari orașe. Odată ce festivitățile se încheie, oamenii revin la casele lor. Guvernul vrea să prevină deranjamentele provocate de călători localnicilor, astfel planifică să închidă toate orașele la anumite ore. Guvernul va atribui fiecărui oraș o valoare ne-negativă a **timpului de închidere**. Guvernul a decis ca suma tuturor timpurilor de închidere nu trebuie să fie mai mare decât K. Mai exact, pentru fiecare i între 0 și $N-1$, inclusiv, timpul de închidere atribuit orașului i este un întreg ne-negativ $c[i]$. Suma tuturor $c[i]$ nu trebuie să fie mai mare decât K.

Fie orașul a cu un oarecare timp de închidere atribuit. Spunem că un oraș b este **accesibil** din orașul a dacă și numai dacă itinerarul $p_0, \ldots, p_t$ dintre aceste două orașe (în particular $p_0 = a$ și $p_t = b$) satisface următoarea condiție:

- lungimea itinerarului p_0, p_1 este cel mult $c[p_1]$, și
- lungimea itinerarului p_0, p_1, p_2 este cel mult $c[p_2]$, și
- ...
- lungimea itinerarului $p_0, p_1, p_2, \ldots, p_t$ este cel mult $c[p_t]$.

În acest an, două festivaluri importante sunt organizate în orașul X și orașul Y. Pentru fiecare atribuire a timpului de închidere, **gradul de comoditate** se definește ca suma următoarelor două numere:

- Numărul orașelor accesibile din orașul X.
- Numărul orașelor accesibile din orașul Y.

De notat că dacă un oraș este accesibil din orașul X și accesibil din orașul Y, atunci acest oraș este calculat *dublu* la gradul de comoditate.

Sarcina ta este de a calcula gradul de comoditate maximal, care poate fi atins printr-o anumită atribuire a orelor de închidere.

Detalii de implementare

Trebuie să implementați următoarea procedură.

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

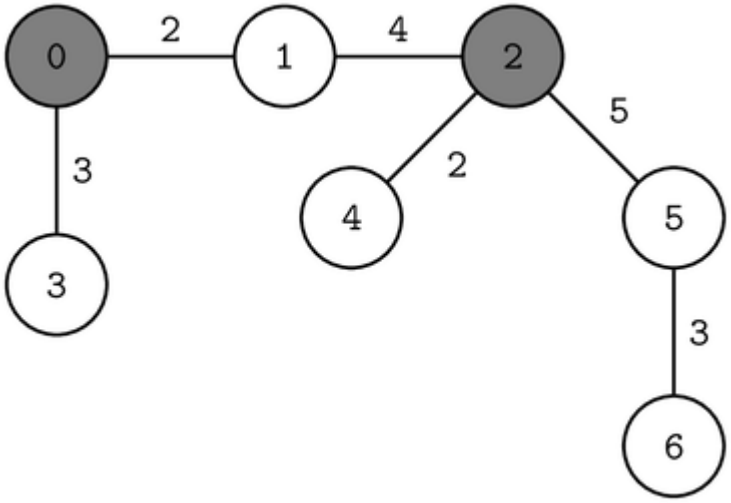
- N: numărul de orașe.
- X, Y: orașele unde se organizează festivalurile importante.
- K: limita superioară a sumei timpurilor de închidere.
- U, V: tablouri de lungime $N-1$ ce descriu conexiunile drumurilor.
- W: tablou de lungime $N-1$ ce descrie lungimile drumurilor.
- Procedura va returna gradul de comoditate maximal care poate fi obținut prin atribuirea anumitor timpi de închidere.
- Această procedură poate fi apelată de **mai multe** ori în fiecare test.

Exemplu

Considerăm următorul apel:

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

Aceasta corespunde următoarei rețele de drumuri:



Presupunem că timpii de închidere sunt atribuite astfel:

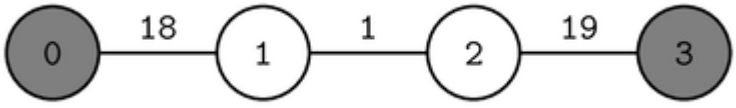
Oraș	0	1	2	3	4	5	6
Timp de închidere	0	4	0	3	2	0	0

De notat că suma timpurilor de închidere este 9, care nu este mai mare decât $K = 10$. Orașele 0,1, și 3 sunt accesibile din orașul X ($X = 0$), pe când orașele 1,2, și 4 sunt accesibile din orașul Y ($Y = 2$). Astfel, gradul de comoditate este $3 + 3 = 6$. Nu există atribuiri a timpurilor de închidere cu un grad de comoditate mai mare decât 6, respectiv funcția va returna 6.

De asemenea considerați următorul apel:

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

Aceasta corespunde următoarei rețele de drumuri:



Presupunem că timpii de închidere sunt atribuite astfel:

Oraș	0	1	2	3
Timp de închidere	0	1	19	0

Orașul 0 este accesibil din orașul X ($X = 0$), pe când orașele 2 și 3 sunt accesibile din orașul Y ($Y = 3$). Astfel, gradul de comoditate este $1 + 2 = 3$.

Nu există atribuiri a timpurilor de închidere cu un grad de comoditate mai mare decât 3, respectiv funcția va returna 3.

Restricții

- $2 \leq N \leq 200\,000$
- $0 \leq X < Y < N$
- $0 \leq K \leq 10^{18}$
- $0 \leq U[j] < V[j] < N$ (pentru fiecare j astfel încât $0 \leq j \leq N - 2$)
- $1 \leq W[j] \leq 10^6$ (pentru fiecare j astfel încât $0 \leq j \leq N - 2$)
- Este posibil de călătorit din orice oraș în orice alt oraș utilizând drumurile.
- $S_N \leq 200\,000$, unde S_N este suma valorilor lui N pentru toate apelurile procedurii `max_score` pentru fiecare test.

Subtask-uri

Vom spune că un drum este **liniar** dacă drumul i conectează orașele i și $i + 1$ (pentru fiecare i astfel încât $0 \leq i \leq N - 2$).

1. (8 puncte) Lungimea itinerarului din orașul X în orașul Y este mai mare decât $2K$.
2. (9 puncte) $S_N \leq 50$, rețeaua de drumuri este liniară.
3. (12 puncte) $S_N \leq 500$, rețeaua de drumuri este liniară.
4. (14 puncte) $S_N \leq 3\,000$, rețeaua de drumuri este liniară.
5. (9 puncte) $S_N \leq 20$
6. (11 puncte) $S_N \leq 100$
7. (10 puncte) $S_N \leq 500$
8. (10 puncte) $S_N \leq 3\,000$
9. (17 puncte) Fără restricții adiționale.

Exemplul de Grader

Fie C este numărul de scenarii, adică numărul de apeluri a `max_score`. Grader-ul citește input-ul în următorul format:

- linia 1: C

Descrierea a C scenarii este următoarea.

Grader-ul citește descrierea fiecărui scenariu în următorul format:

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

Grader-ul tipărește o singură linie pentru fiecare scenariu, în următorul format::

- linia 1: valoarea returnată de `max_score`