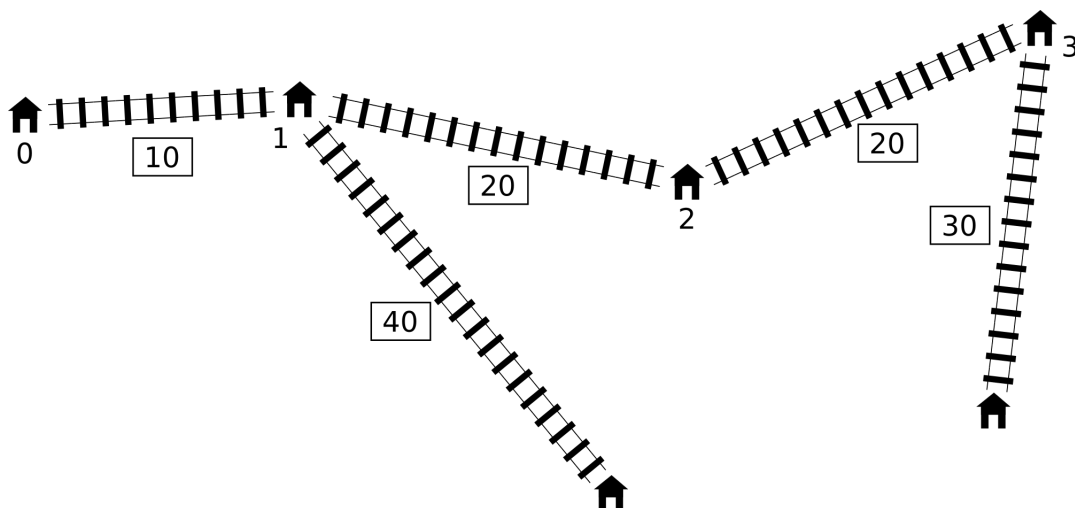


Skrót (Shortcut)

Paweł ma zabawkową kolejkę elektryczną. Nie jest ona skomplikowana: zawiera ona jedną główną linię torów łączącą n stacji ponumerowanych kolejno od 0 do $n - 1$ wzdłuż tej linii. Odległość między stacjami i oraz $i + 1$ wynosi l_i centymetrów ($0 \leq i < n - 1$).

Oprócz głównej linii, w kolejce mogą występować także poboczne linie. Każda poboczna linia łączy stację leżącą na głównej linii z pewną nową stacją, nieleżącą na głównej linii. (Te nowe stacje nie są numerowane). Z każdej stacji głównej linii może wychodzić co najwyżej jedna poboczna linia. Długość pobocznej linii, która zaczyna się na stacji i , wynosi d_i centymetrów. Jeżeli na stacji i nie zaczyna się żadna poboczna linia, to przyjmujemy, że $d_i = 0$.



Paweł chce teraz dodać jeden skrót: ekspresową linię pomiędzy dwiema różnymi (być może sąsiadującymi ze sobą) stacjami **głównej linii**. Ta ekspresowa linia będzie miała długość dokładnie c centymetrów, niezależnie od tego, które dwie stacje ona połączy.

Każdy odcinek torów, łącznie z nowo utworzoną ekspresową linią, kolejka może pokonywać w obie strony. *Odległość* pomiędzy dwiema stacjami rozumiemy jako długość najkrótszej drogi, która łączy te stacje poprzez sieć torów. Z kolei *średnicą* całej sieci torów jest największa wśród odległości wszystkich par stacji. Innymi słowy, jest to najmniejsza liczba t , taka że odległość pomiędzy każdymi dwiema stacjami wynosi co najwyżej t .

Paweł chce wybudować ekspresową linię tak, aby średnica powstałej sieci torów była minimalna.

Szczegóły implementacji

Powinieneś zaimplementować jedną funkcję

`int64 find_shortcut(int n, int[] l, int[] d, int c)`

- `n`: liczba stacji na głównej linii,
- `l`: odległości pomiędzy stacjami na głównej linii (tablica rozmiaru $n - 1$),
- `d`: długości pobocznych linii (tablica rozmiaru n),
- `c`: długość nowej, ekspresowej linii.
- Funkcja powinna zwracać najmniejszą możliwą średnicę sieci torów po dodaniu ekspresowej linii.

Szczegóły implementacji w Twoim języku programowania znajdują się w dostarczonych plikach z szablonami.

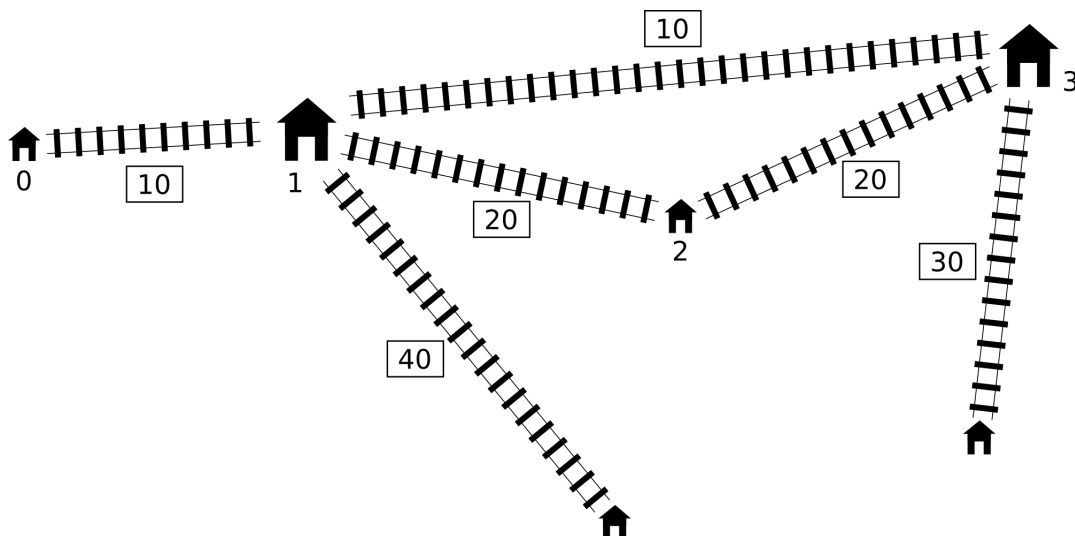
Przykłady

Przykład 1

Dla sieci torów pokazanej powyżej program sprawdzający wykonuje następujące wywołanie:

```
find_shortcut(4, [10, 20, 20], [0, 40, 0, 30], 10)
```

Optymalnym rozwiązaniem jest wybudowanie ekspresowej linii pomiędzy stacjami 1 oraz 3, jak pokazano poniżej.



Średnica nowej sieci torów wynosi 80 centymetrów, zatem funkcja powinna zwrócić 80.

Przykład 2

Program sprawdzający wykonuje następujące wywołanie:

```
find_shortcut(9, [10, 10, 10, 10, 10, 10, 10, 10],  
[20, 0, 30, 0, 0, 40, 0, 40, 0], 30)
```

Optymalne rozwiązanie polega na połączeniu stacji 2 oraz 7. Średnica sieci kolejowej wynosi wtedy 110.

Przykład 3

Program sprawdzający wykonuje następujące wywołanie:

```
find_shortcut(4, [2, 2, 2],  
              [1, 10, 10, 1], 1)
```

Połączenie stacji 1 oraz 2 jest optymalne i zmniejsza średnicę do 21.

Przykład 4

Program sprawdzający wykonuje następujące wywołanie:

```
find_shortcut(3, [1, 1],  
              [1, 1, 1], 3)
```

Połączenie żadnych dwóch stacji ekspresową linią o długości 3 nie poprawi początkowej średnicy sieci torów, wynoszącej 4.

Podzadania

We wszystkich podzadaniach $2 \leq n \leq 1\,000\,000$, $1 \leq l_i \leq 10^9$, $0 \leq d_i \leq 10^9$, $1 \leq c \leq 10^9$.

1. (9 punktów) $2 \leq n \leq 10$,
2. (14 punktów) $2 \leq n \leq 100$,
3. (8 punktów) $2 \leq n \leq 250$,
4. (7 punktów) $2 \leq n \leq 500$,
5. (33 punkty) $2 \leq n \leq 3000$,
6. (22 punkty) $2 \leq n \leq 100\,000$,
7. (4 punkty) $2 \leq n \leq 300\,000$,
8. (3 punkty) $2 \leq n \leq 1\,000\,000$.

Przykładowy program sprawdzający

Przykładowy program sprawdzający wczytuje dane w następującym formacie:

- o wiersz 1: liczby całkowite n i c ,
- o wiersz 2: liczby całkowite l_0, l_1, \dots, l_{n-2} ,
- o wiersz 3: liczby całkowite d_0, d_1, \dots, d_{n-1} .