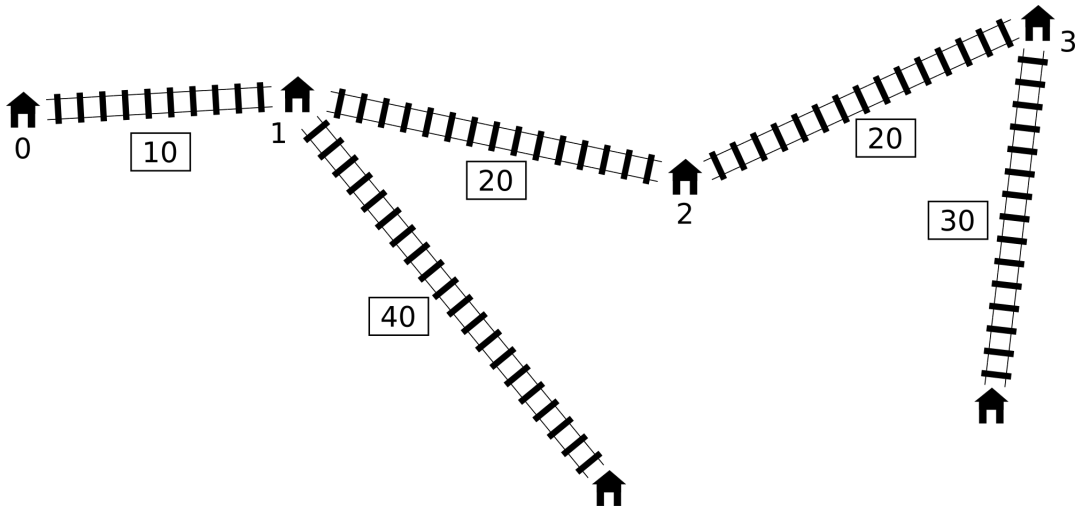
Shortcut

En Pavel té una xarxa ferroviaria de joguina, que és molt simple. Hi ha una única línia principal formada per n estacions. Aquestes estacions estan numerades del 0 al $n-1$ en l'ordre de la línia. La distància entre les estacions i i $i+1$ és de l_i centímetres ($0 \leq i < n-1$).

A part de la línia principal pot haver-hi algunes línies secundàries. Cada línia secundària uneix una estació de la línia principal i una nova estació que no es troba a la línia principal. (Aquestes estacions no estan numerades.) A cada estació de la línia principal hi comença com a molt una línea secundària. La longitud de la línia secundària que comença a l'estació i és de d_i centímetres. Si no hi ha línia secundària a l'estació i ho denotarem per $d_i = 0$.



En Pavel està plantejant-se construir una drecera: una línia exprés entre dues estacions diferents (possiblement veïnes) de la **línia principal**. El tram exprés té una longitud d'exactament c centímetres, independentment de quines dues estacions connecti.

Cada tram de la via, incluint el de la nova línia exprés, pot ser utilitzat en ambdues direccions. La *distància* entre dues estacions és la mínima longitud d'entre les rutes que vagin per les vies entre una estació i l'altra. El *diàmetre* de tota la xarxa de vies és el màxim de totes les distàncies entre parelles d'estacions. En altres paraules, és el mínim nombre t , tal que la distància entre cada parell d'estacions és com a molt t .

En Pavel vol construir la línia exprés de tal manera que el diàmetre de la xarxa

resultant sigui el mínim possible.

Detalls de la implementació

Se us demana que implementeu una funció (mètode):
`int64 find_shortcut(int n, int[] l, int[] d, int c)`

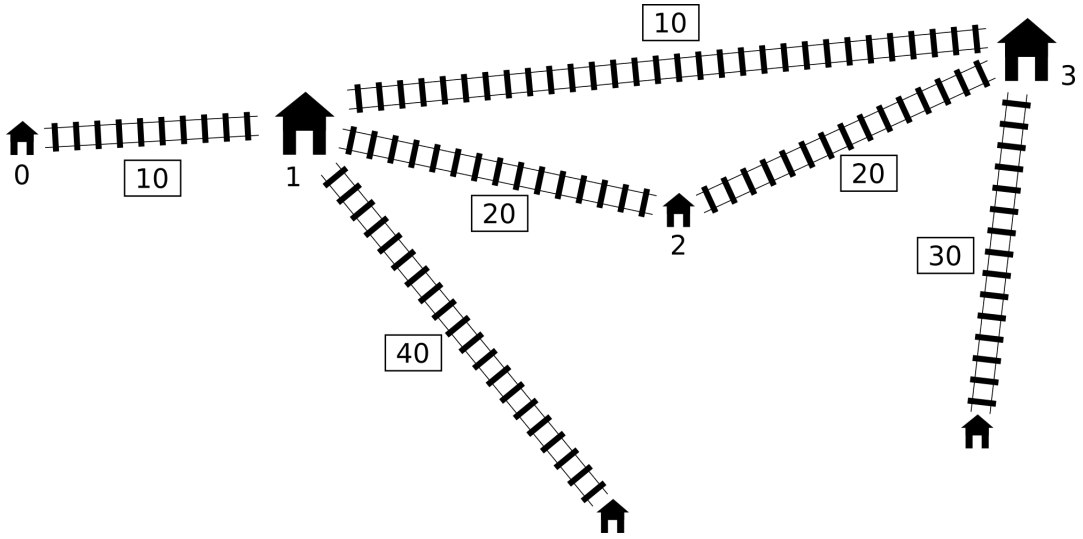
- `n`: nombre d'estacions de la línia principal,
- `l`: distàncies entre parells d'estacions de la línia principal (array de mida $n-1$),
- `d`: Longituds de les línies secundàries (array de longitud n),
- `c`: Longitud de la nova línia exprés.
- la funció ha de retornar el diàmetre mínim possible de la xarxa ferroviaria després d'afegir el tram exprés.

Si us plau, feu servir els arxius de mostra per veure els detalls de la implementació al vostre llenguatge de programació.

Exemples

Exemple 1

Si considerem la xarxa anterior, el grader faria la crida següent:
`find_shortcut(4, [10, 20, 20], [0, 40, 0, 30], 10)`
La solució òptima consisteix a construir la drecera entre les estacions 1 i 3, tal i com es mostra a continuació.



El diàmetre de la nova xarxa és de 80 centímetres, de manera que la funció ha de retornar 80.

Exemple 2

El grader fa la crida següent:

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

La solució òptima consisteix a connectar les estacions 2 i 7, cosa que fa que el diàmetre resultant sigui de 110.

Exemple 3

El grader fa la crida següent:

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

La solució òptima consisteix a connectar les estacions 1 i 2, cosa que redueix el diàmetre a 21.

Exemple 4

El grader fa la crida següent:

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

El fet de connectar qualsevol parell d'estacions amb la drecera de mida 3 no redueix el diàmetre inicial de la xarxa, que és de 4.

Subtasques

Per a totes les subtasques es compleix $2 \leq n \leq 1\,000\,000$, $1 \leq l_i \leq 10^9$, $0 \leq d_i \leq 10^9$, $1 \leq c \leq 10^9$.

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

Grader de mostra

El grader de mostra llegeix l'entrada en el format següent:

- línia 1: enters n i c,
- línia 2: enters $l_0, l_1, \ldots, l_{n-2}$,
- línia 3: enters $d_0, d_1, \ldots, d_{n-1}$.