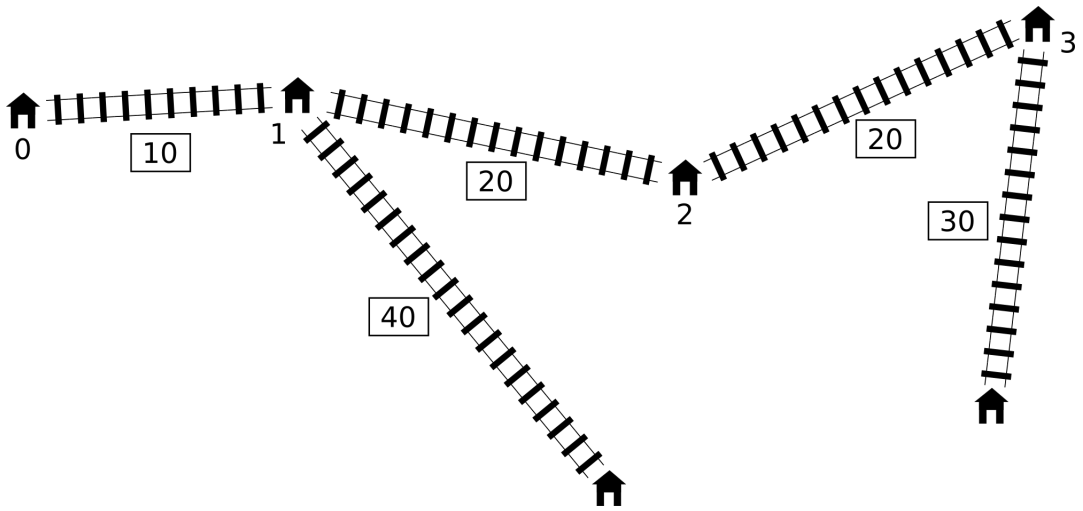
Shortcut

Pavel tiene una red férrea de juguete. Esta es muy simple. Hay una única línea principal que consiste en n estaciones numeradas de forma consecutiva desde el 0 al $n-1$. Las estaciones 0 y $n-1$ están ubicadas en los extremos de la línea principal. La distancia entre las estaciones i y $i+1$ es l_i centímetros ($0 \leq i < n-1$).

Además de la línea principal puede haber algunas líneas secundarias. Cada línea secundaria es una vía férrea entre una estación en la línea principal y una nueva estación que no está ubicada en la línea principal. (Estas nuevas estaciones no están numeradas.) A lo más una línea secundaria puede partir de cada estación en la línea principal. El largo de la línea secundaria que parte en la estación i es d_i centímetros. Un valor de $d_i = 0$ denota que no hay línea secundaria que salga de la estación i.



Pavel está planeando construir un atajo: una línea express entre dos estaciones diferentes de **la línea principal** (posiblemente contiguas). La línea express tendrá un largo de exactamente c centímetros, sin importar qué estaciones conecte.

Cada segmento de vía férrea, incluyendo el de la nueva línea express, puede ser utilizado en ambas direcciones. La *distancia* entre dos estaciones es el largo de la ruta más corta que va desde una estación a la otra utilizando las vías férreas. El *diámetro* de toda la red de vías es la máxima distancia entre todos los pares de estaciones. En otras palabras, es el número t más pequeño, tal que la distancia entre cada par de estaciones es a lo más t.

Pavel quiere construir una línea express de tal forma que el diámetro resultante sea el mínimo posible.

Detalles de implementación

Debes implementar la siguiente función

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

- `n`: número de estaciones en la línea principal,
- `l`: distancia entre las estaciones en la línea principal (arreglo de tamaño $n-1$).
- `d`: largos de las líneas secundarias (arreglo de tamaño n),
- `c`: largo de la nueva línea express,
- la función debe retornar el menor diámetro posible de la red de vías después de añadir la línea express.

Por favor utiliza los archivos con la plantillas para obtener detalles de implementación en tu lenguaje de programación.

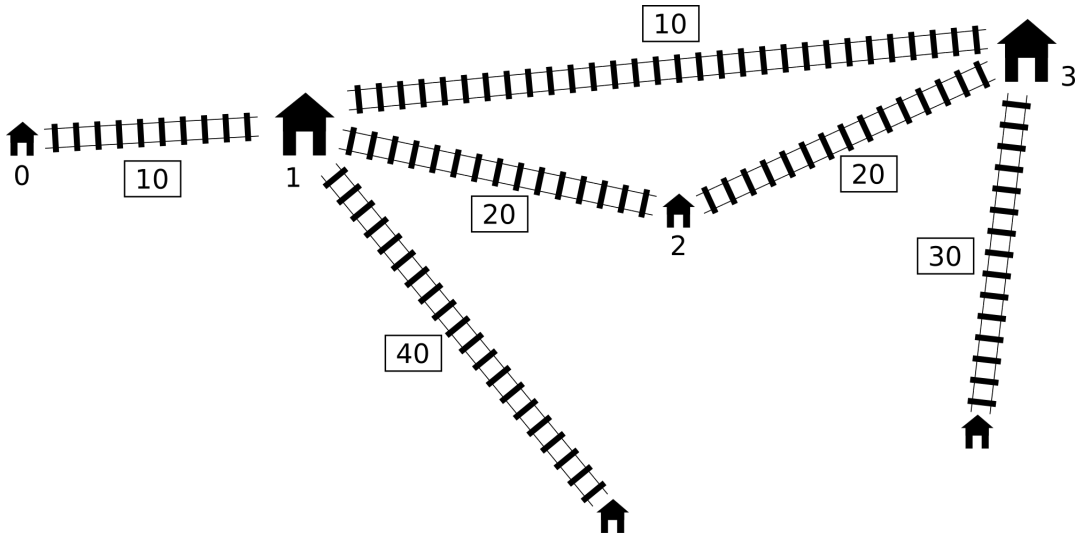
Ejemplos

Ejemplo 1

Para la red de vías mostrada más arriba, el grader haría la siguiente llamada de función:

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

La solución óptima es construir una línea express entre las estaciones 1 y 3, como se muestra más abajo.



El diámetro de la red férrea es 80 centímetros, por lo que la función debe retornar 80.

Ejemplo 2

El grader hace la siguiente llamada de función:

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

La solución óptima es conectar las estaciones 2 y 7, en cuyo caso el diámetro es 110.

Ejemplo 3

El grader hace la siguiente llamada de función:

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

La solución óptima es conectar las estaciones 1 y 2, reduciendo el diámetro a 21.

Ejemplo 4

El grader hace la siguiente llamada de función:

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

Conectar cualquier par de estaciones con una línea express de largo 3 no mejora el diámetro inicial de la red férrea que es igual a 4.

Subtareas

En todas las subtareas $2 \le n \le 1\,000\,000$, $1 \le l_i \le 10^9$, $0 \le d_i \le 10^9$, $1 \le c \le 10^9$.

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

Grader de ejemplo

El grader de ejemplo lee de la entrada en el siguiente formato:

- línea 1: enteros n y c,
- línea 2: enteros $l_0, l_1, \ldots, l_{n-2}$,
- línea 3: enteros $d_0, d_1, \ldots, d_{n-1}$.