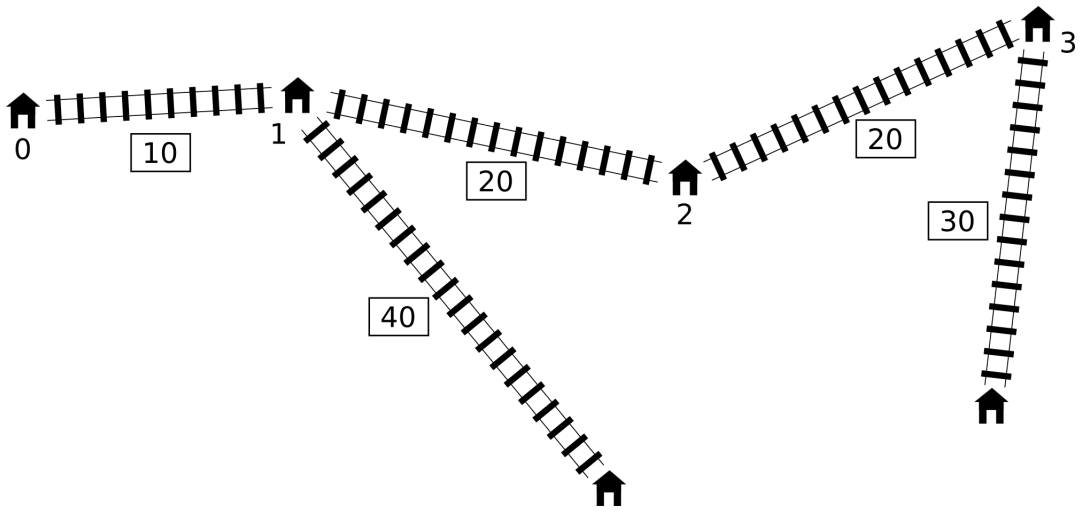
Atalho

Pavel tem um comboio de brincar. É muito simples. Há uma única linha principal que consiste de n estações. Estas estações são numeradas de 0 a $n-1$, em ordem, ao longo da linha. A distância entre as estações i e $i+1$ é igual a l_i centímetros ($0 \leq i < n-1$).

Além da linha principal, podem existir algumas algumas linhas secundárias. Cada linha secundária é uma linha de estrada de ferro entre uma estação na linha principal e uma nova estação que não se encontra na linha principal. (Estas estações novas não são numeradas.) No máximo uma linha secundária se inicia em cada em cada estação da linha principal. O comprimento da linha secundária que se inicia na estação i é d_i centímetros. Usamos $d_i = 0$ para indicar que não existe linha secundária que se inicia na estação i.



Pavel agora está planejando um atalho: uma linha expresso entre duas estações distintas (possivelmente vizinhas) da **linha principal**. O comprimento da linha expresso será exatamente de c centímetros, independentemente de quais duas estações serão ligadas.

Cada segmento da estrada de ferro, inclusive a nova linha expresso, pode ser usado em ambos os sentidos. A *distância* entre duas estações é o menor comprimento de uma rota que vai de uma estação à outra. O *diâmetro* de todo sistema ferroviário é a distância máxima entre todos os pares de estações. Em outras palavras, o diâmetro é o menor número t, tal que a distância entre quaisquer duas estações é no máximo t.

Pavel quer construir a linha expresso de tal forma que o diâmetro resultante seja minimizado.

Detalhes de implementação

Você deve implementar a função
`int64 find_shortcut(int n, int[] l, int[] d, int c)`

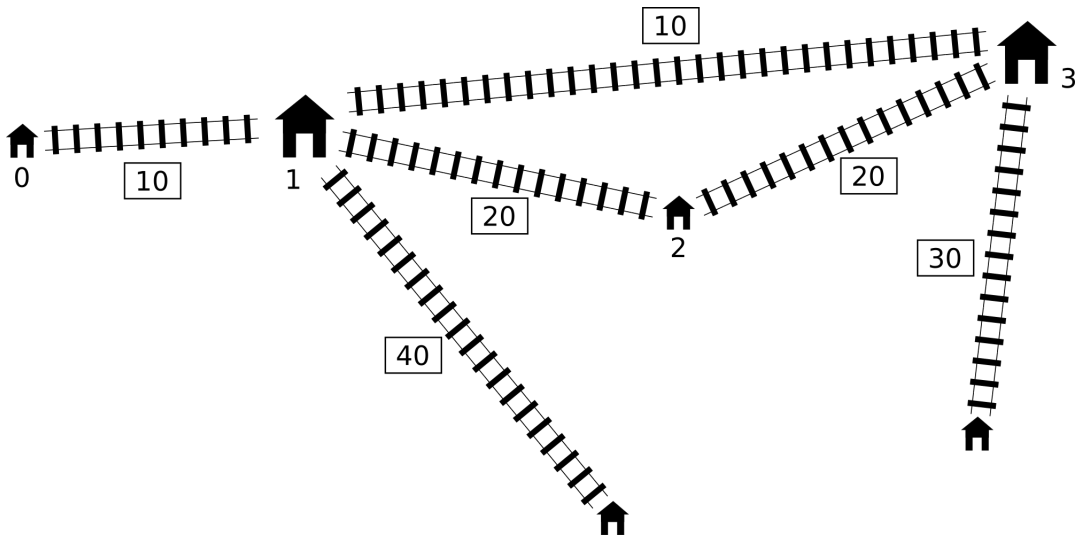
- `n`: o número de estações na linha principal,
- `l`: as distâncias entre estações na linha principal (vetor de comprimento $n-1$),
- `d`: os comprimentos das linhas secundárias (vetor de comprimento n),
- `c`: o comprimento da nova linha expresso.
- a função deve retornar o menor diâmetro possível da rede ferroviária, após a adição da linha expresso.

Por favor, use os ficheiros modelo fornecidos para os detalhes de implementação na sua linguagem de programação.

Exemplos

Exemplo 1

No caso da ferrovia dada acima, o corretor executa a seguinte chamada de função:
`find_shortcut(4, [10, 20, 20], [0, 40, 0, 30], 10)`
A solução ótima é construir a linha expresso ligando as estações 1 e 3, como indicado abaixo.



O diâmetro da nova rede ferroviária é igual a 80 centímetros, assim a função deve retornar 80.

Exemplo 2

O corretor executa a seguinte chamada da função:

```
find_shortcut(9, [10, 10, 10, 10, 10, 10, 10, 10],
```

```
[20, 0, 30, 0, 0, 40, 0, 40, 0], 30)
```

A solução ótima consiste em conectar as estações 2 e 7, e nesse caso o diâmetro é 110.

Exemplo 3

O corretor executa a seguinte chamada da função:

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

A solução ótima consiste em conectar as estações 1 e 2, reduzindo o diâmetro para 21.

Exemplo 4

O corretor executa a seguinte chamada da função:

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

A conexão entre duas estações através da linha expresso de comprimento 3 não diminui o diâmetro inicial da rede ferroviária, que é igual a 4.

Subtarefas

Em todas as subtarefas, $2 \leq n \leq 1\,000\,000$, $1 \leq l_i \leq 10^9$, $0 \leq d_i \leq 10^9$, $1 \leq c \leq 10^9$.

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

Corretor exemplo

O corretor exemplo lê a entrada no formato abaixo:

- linha 1: inteiros n e c,
- linha 2: inteiros $l_0, l_1, \ldots, l_{n-2}$,
- linha 3: inteiros $d_0, d_1, \ldots, d_{n-1}$.