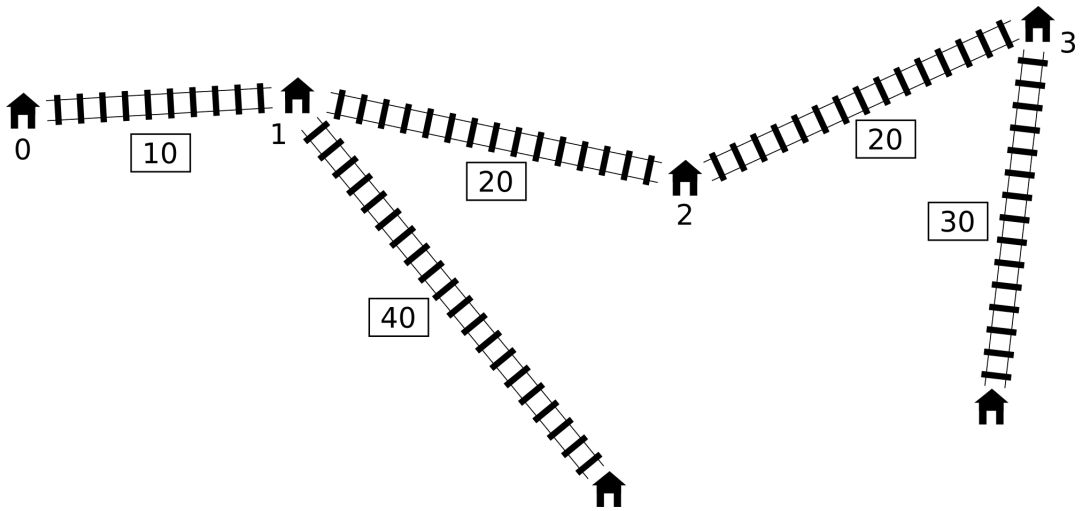
Scorciatoia

Pavel ha un trenino molto semplice. C'è una sola linea principale con n stazioni, numerate da 0 a $n-1$ in ordine lungo la linea. La distanza tra le stazioni i e $i+1$ è di l_i centimetri ($0 \le i < n-1$).

Oltre alla linea principale, ci possono essere alcune linee secondarie. Ogni linea secondaria collega una stazione della linea principale con una nuova stazione che non appartiene alla linea principale (e che non è numerata).

Al più una linea secondaria può partire da ogni stazione della linea principale. La lunghezza della linea secondaria che inizia dalla stazione i è di d_i centimetri. Usiamo $d_i = 0$ per denotare che dalla stazione i non parte alcuna linea secondaria.



Pavel sta pianificando di aggiungere una scorciatoia: una linea espresso tra due stazioni diverse **sulla linea principale** (potenzialmente anche consecutive). La linea espresso è lunga esattamente c centimetri, indipendentemente da quali stazioni collega.

Ogni segmento della ferrovia, compresa la nuova linea espresso, può essere usato in entrambe le direzioni. La *distanza* tra due stazioni è la lunghezza del percorso più breve che collega le due stazioni. Il *diametro* della ferrovia è la massima distanza tra tutte le coppie di stazioni. In altre parole, è il più piccolo numero t tale che la distanza tra due stazioni non supera mai t.

Pavel vuole aggiungere la linea espresso in modo tale che il diametro della ferrovia risultante sia il minimo possibile.

Dettagli di implementazione

Devi implementare la seguente funzione (metodo):
`int64 find_shortcut(int n, int[] l, int[] d, int c)`

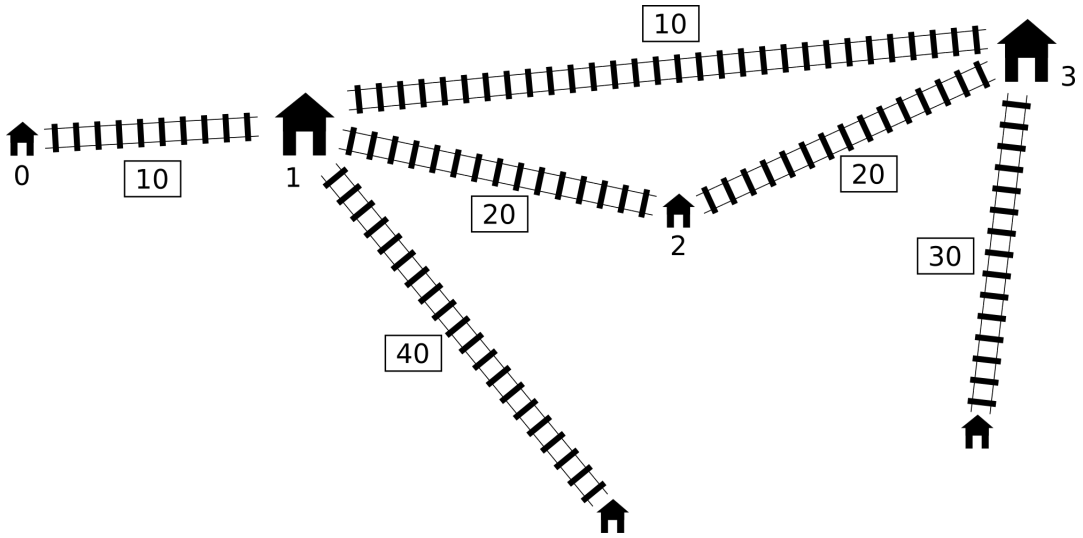
- `n`: numero di stazioni sulla linea principale,
- `l`: distanze tra le stazioni della linea principale (array di lunghezza $n - 1$),
- `d`: lunghezze delle linee secondarie (array di lunghezza n),
- `c`: lunghezza della nuova linea espresso.
- la funzione deve restituire il più piccolo diametro della ferrovia ottenibile dopo aver aggiunto la linea espresso.

Per ulteriori dettagli, vedi il template fornito per il tuo linguaggio di programmazione.

Esempi

Esempio 1

Per la ferrovia nella figura precedente, il grader chiamerà la funzione in questo modo:
`find_shortcut(4, [10, 20, 20], [0, 40, 0, 30], 10)`
La soluzione ottima si ottiene aggiungendo la linea espresso tra le stazioni 1 e 3, come mostrato qui di seguito.



Il diametro della ferrovia risultante è 80 cm, quindi la funzione deve restituire 80.

Esempio 2

Il grader chiamerà la funzione in questo modo:

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

La soluzione ottima consiste nel collegare le stazioni 2 e 7, ottenendo un diametro pari a 110.

Esempio 3

Il grader chiamerà la funzione in questo modo:

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

La soluzione ottima consiste nel collegare le stazioni 1 e 2, ottenendo un diametro pari a 21.

Esempio 4

Il grader chiamerà la funzione in questo modo:

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

Il diametro iniziale, pari a 4, non è ridotto dalla nuova linea espresso, di lunghezza 3, indipendentemente da quali stazioni la linea espresso collega.

Subtask

Per tutti i subtask, $2 \le n \le 1\,000\,000$, $1 \le l_i \le 10^9$, $0 \le d_i \le 10^9$, $1 \le c \le 10^9$.

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

Grader di esempio

Il grader di esempio legge l'input nel formato seguente:

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