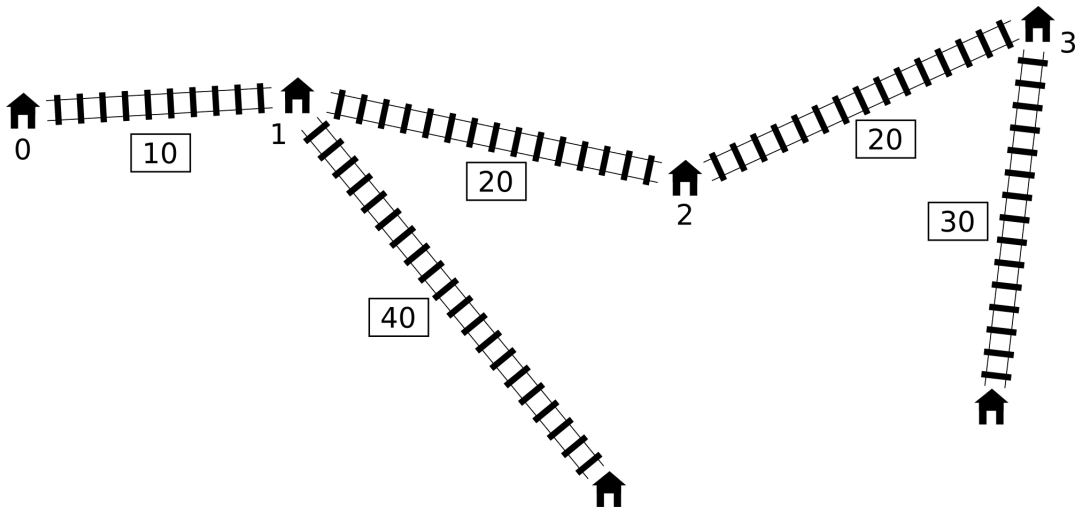
Abkürzung

Pavel hat eine Spielzeugeisenbahn. Sie ist sehr simpel. Es gibt eine Hauptlinie, die aus n Stationen besteht, welche der Reihe nach von 0 bis $n-1$ nummeriert sind. Die Distanz zwischen den Stationen i und $i+1$ ist l_i Zentimeter ($0 \leq i < n-1$).

Ausser der Hauptlinie kann es einige Nebenstrecken geben. Jede Nebenstrecke führt von einer Station an der Hauptlinie zu einer neuen Station, die nicht an der Hauptlinie liegt. (Diese neuen Stationen sind nicht nummeriert.) Bei jeder Station der Hauptlinie startet höchstens eine Nebenstrecke. Die Länge der Nebenstrecke, die bei Station i beginnt, ist d_i Zentimeter. Dabei bedeutet $d_i = 0$, dass es keine Nebenstrecke bei Station i gibt.



Pavel möchte nun eine Abkürzung einbauen: eine Expressstrecke zwischen zwei verschiedenen (möglicherweise benachbarten) Stationen **der Hauptlinie**. Die Expressstrecke wird exakt c Zentimeter lang sein, egal welche zwei Stationen sie verbindet.

Jeder Abschnitt des Eisenbahnnetzes, inklusive der neuen Expressstrecke, kann in beide Richtungen befahren werden. Die *Distanz* zwischen zwei Stationen ist die Länge der kürzesten Route entlang der Eisenbahnschienen von der einen Station zur anderen. Der *Durchmesser* des gesamten Eisenbahnnetzes ist die maximale Distanz zwischen allen Stationspaaren. In anderen Worten, das ist die kleinste Zahl t, sodass die Distanz zwischen allen Stationspaaren höchstens t ist.

Pavel möchte die Expressstrecke so bauen, dass der Durchmesser des resultierenden

Eisenbahnnetzes minimiert wird.

Implementierungsdetails

Du sollst folgende Funktion implementieren:
`int64 find_shortcut(int n, int[] l, int[] d, int c)`

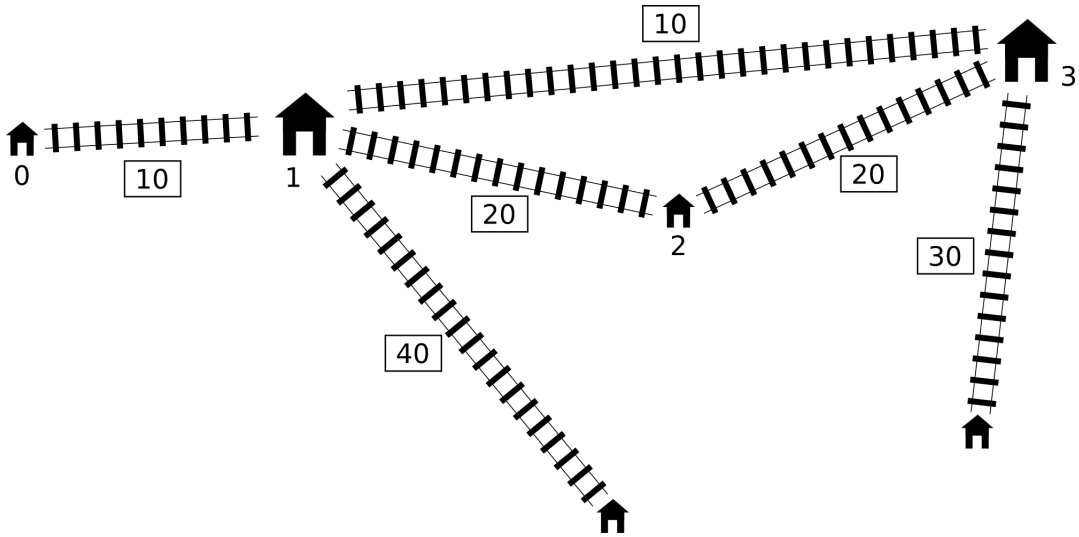
- `n`: die Anzahl der Stationen entlang der Hauptlinie,
- `l`: Distanzen zwischen den Stationen entlang der Hauptlinie (Array der Länge $n-1$),
- `d`: Länge der Nebenstrecken (Array der Länge n),
- `c`: Länge der neuen Expressstrecke.
- Die Funktion soll den kleinstmöglichen Durchmesser nach dem Hinzufügen der Expressstrecke zurückgeben.

Die zur Verfügung gestellten Vorlagedateien zeigen dir die Details in deiner Programmiersprache.

Beispiele

Beispiel 1

Für das oben abgebildete Eisenbahnnetz würde der Grader folgenden Funktionsaufruf machen:
`find_shortcut(4, [10, 20, 20], [0, 40, 0, 30], 10)`
Die optimale Lösung ist, die Expressstrecke zwischen den Stationen 1 und 3 zu bauen, wie unten abgebildet.



Der Durchmesser des neuen Eisenbahnnetzes ist 80 Zentimeter, also soll die Funktion 80 zurückgeben.

Beispiel 2

Der Grader macht folgenden Funktionsaufruf:

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

Die optimale Lösung verbindet die Stationen 2 und 7. In diesem Fall ist der Durchmesser 110.

Beispiel 3

Der Grader macht folgenden Funktionsaufruf:

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

Die optimale Lösung verbindet die Stationen 1 and 2, was den Durchmesser auf 21 verringert.

Beispiel 4

Der Grader macht folgenden Funktionsaufruf:

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

Das Verbinden jeglicher zwei Stationen mit der Expressstrecke der Länge 3 verbessert den ursprünglichen Durchmesser des Eisenbahnnetzes von 4 nicht.

Teilaufgaben

In allen Teilaufgaben gilt $2 \leq n \leq 1\,000\,000$, $1 \leq l_i \leq 10^9$, $0 \leq d_i \leq 10^9$, $1 \leq c \leq 10^9$.

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

Beispielgrader

Der Beispielgrader liest die Eingabe in folgendem Format:

- Zeile 1: Ganzzahlen n und c,
- Zeile 2: Ganzzahlen $l_0, l_1, \ldots, l_{n-2}$,
- Zeile 3: Ganzzahlen $d_0, d_1, \ldots, d_{n-1}$.