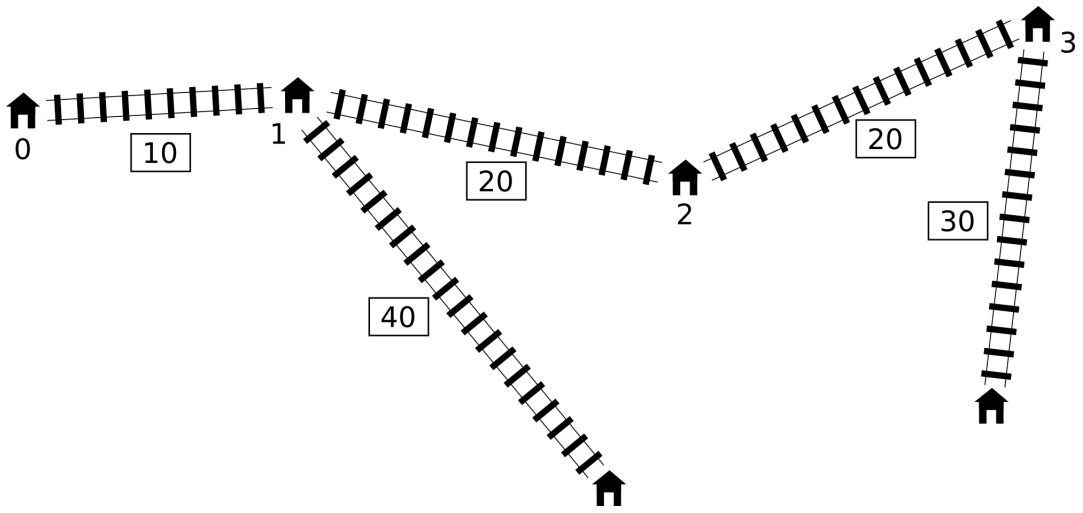
უმოკლესი გზა

პაველს აქვს სათამაშო რკინიგზა. იგი შედგება ერთი მთავარი მაგისტრალისაგან, რომელიც შეიცავს n სადგურს, გადანომრილს 0 -დან $(n-1)$ -მდე. სადგურები 0 და $n-1$ ნომრებით წარმოადგენენ მთავარი მაგისტრალის ორ ბოლოს. მანძილი i და $i+1$ სადგურებს შორის შეადგენს l_i სანტიმეტრს ($0 \leq i < n-1$).

გარდა მთავარი მაგისტრალისა არსებობენ კიდევ მეორადი ხაზებიც. ყოველი მეორადი ხაზი წარმოადგენს რკინიგზას მთავარი მაგისტრალის რომელიმე სადგურიდან ახალ სადგურამდე, რომელიც არ შედის მთავარ მაგისტრალში (ახალი სადგურები გადანომრილი არ არის). მთავარი მაგისტრალის ყოველი სადგური შეერთებულია არაუმეტეს ერთ ახალ სადგურთან. მთავარი მაგისტრალის i -ური სადგურიდან დაწყებული მეორადი ხაზის სიგრძე წარმოადგენს d_i სანტიმეტრს. მნიშვნელობა $d_i = 0$ აღნიშნავს, რომ მთავარი მაგისტრალის i -ური სადგურიდან მეორადი ხაზი არ გამოდის.



პაველი აპირებს ახალი უმოკლესი გზის აგებას: ეს იქნება ექსპრეს–ხაზი მთავარი მაგისტრალის ორ სადგურს (შესაძლოა მეზობელ) შორის. ექსპრეს–ხაზის სიგრძე იქნება ზუსტად c სანტიმეტრი, მიუხედავად იმისა თუ რომელ ორ სადგურს შეაერთებს ის.

მთლიანი რკინიგზის ქსელის ნებისმიერი სეგმენტი, ექსპრეს–ხაზის ჩათვლით, შეიძლება ორმხრივად იქნას გამოყენებული. "მანძილი" ორ სადგურს შორის წარმოადგენს იმ მარშრუტის უმოკლეს სიგრძეს, რომელიც მიემართება ერთი სადგურიდან მეორესაკენ რკინიგზის ხაზების გავლით. რკინიგზის ქსელის "დიამეტრს" უწოდებენ მაქსიმალურ მანძილს სადგურთა ყველა შესაძლო წყვილს შორის. სხვა სიტყვებით, ეს არის ისეთი უმცირესი რიცხვი t , რომ მანძილი სადგურთა ნებისმიერ წყვილს შორის არ აღემატება t -ს.

პაველს სურს ისეთი ექსპრეს–ხაზის აგება, რომ რკინიგზის საბოლოო ქსელის დიამეტრი იყოს მინიმალური.

იმპლემენტაციის დეტალები

თქვენ უნდა მოახდინოთ შემდეგი ფუნქციის იმპლემენტაცია
`int64 find_shortcut(int n, int[] l, int[] d, int c)`

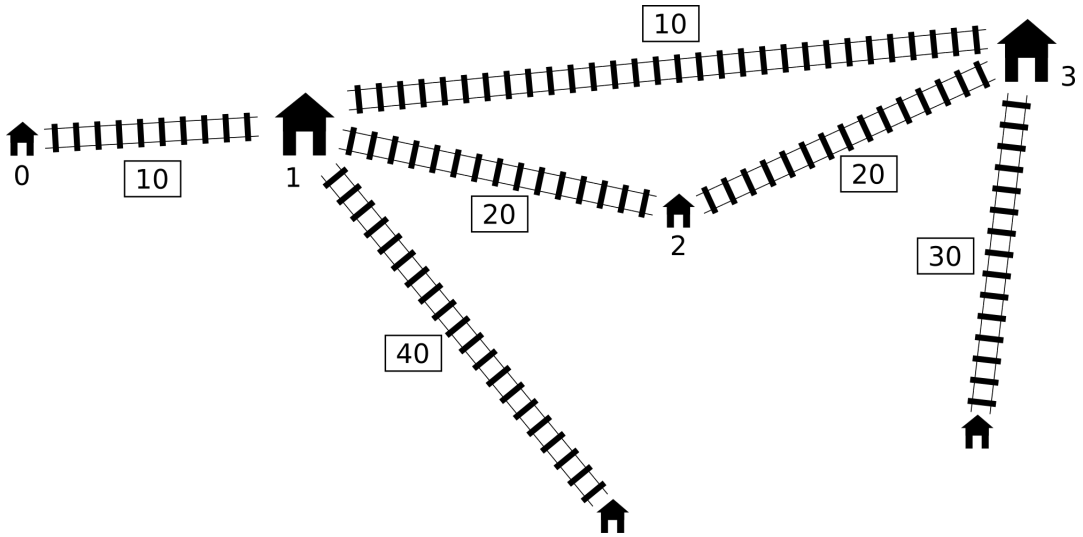
- `n`: სადგურების რაოდენობა მთავრ მაგისტრალზე,
- `l`: მანძილები მთავარი მაგისტრალის სადგურებს შორის ($n-1$ სიგრძის მასივი),
- `d`: მეორადი ხაზების სიგრძეები (n სიგრძის მასივი),
- `c`: ექსპრეს–ხაზის სიგრძე.
- ფუნქციამ უნდა დააბრუნოს უმცირესი შესაძლო დიამეტრი რკინიგზის ქსელზე ექსპრეს–ხაზის დამატების შემდეგ.

Please use the provided template files for details of implementation in your programming language.

მაგალითები

მაგალითი 1

ზემოთ ნაჩვენები რკინიგზის ქსელისათვის გრადერი გააკეთებს ასეთ გამოძახებას:
`find_shortcut(4, [10, 20, 20], [0, 40, 0, 30], 10)`
ოპტიმალური ამონახსნია ექსპრეს–ხაზის დამატება პირველ და მესამე სადგურებს შორის, როგორც ეს ქვედა ნახაზზეა ნაჩვენები.



რკინიგზის ახალი ქსელის დიამეტრია 80 სანტიმეტრი, მაშასადამე ფუნქციამ უნდა დააბრუნოს 80 .

Example 2

გრადერი გააკეთებს შემდეგი ფუნქციის გამოძახებას:

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

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

ოპტიმალური ამონახსნია 1 და 6 სადგურების შეერთება, რის შემდეგაც დიამეტრი იქნება 110.

Example 3

გრადერი გააკეთებს შემდეგი ფუნქციის გამოძახებას:

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

ოპტიმალური ამონახსნია 2 და 3, სადგურების შეერთება, რის შემდეგაც დიამეტრი იქნება 21.

Example 4

გრადერი გააკეთებს შემდეგი ფუნქციის გამოძახებას:

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

სადგურთა არცერთი წყვილის შეერთება 3 სიგრძის ექსპრეს–ხაზით არ აუმჯობესებს რკინიგზის საწყისი ქსელის დიამეტრს, რომლის სიგრძეა 4.

Subtasks

In all Subtasks $2 \leq n \leq 1\,000\,000$, $1 \leq l_i \leq 10^9$, $0 \leq d_i \leq 10^9$, $1 \leq c \leq 10^9$.

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

Sample grader

The sample grader reads the input in the following format:

- line 1: integers n and c,
- line 2: integers $l_0, l_1, \ldots, l_{n-2}$,
- line 3: integers $d_0, d_1, \ldots, d_{n-1}$.