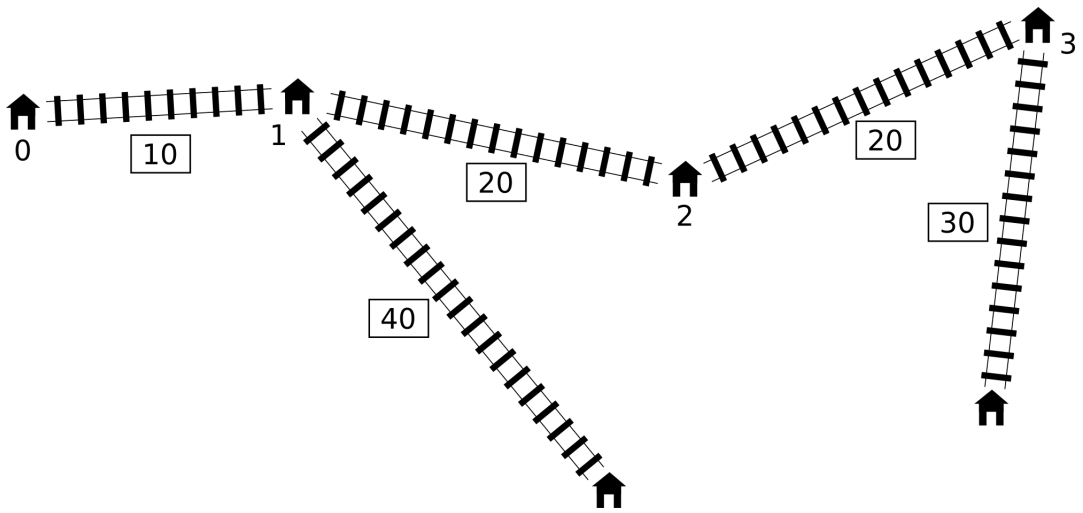
Shortcut

Šemso se voli igrati sa vozićima. Kupio je jednostavnu prugu koja se sastoji od n stanica koje su uzastopno označene od 0 do $n-1$. Stanice 0 i $n-1$ leže na krajevima glavne pruge. Udaljenost stanica i, te $i+1$ iznosi l_i centimetara ($0 < l_i < n-1$).

Osim glavne pruge mogu postojati i sporedne pruge. Svaka sporedna pruga povezuje stanicu na glavnoj pruzi i novu stanicu koja ne leži na glavnoj pruzi. (Ove nove stanice nisu numerisane). Najviše jedna sporedna pruga može početi na svakoj stanici glavne pruge. Dužina sporedne pruge koja počinje na stanici i iznosi d_i centimetara. Stavljamo $d_i = 0$ ako na stanici i nema sporedne pruge.



Šemso želi izgraditi kraticu: brzu prugu između dvije (možda i susjednih) stanice **glavne pruge**. Brza pruga će biti dužine tačno c centimetara, bez obzira koje dvije stanice povezuje.

Sve su pruge dvosmjerne. *Udaljenost* dvaju stanica je najmanja dužina dionice koja prugama ide od jedne do druge stanice. *Prečnik* cijele mreže je maksimalna udaljenost među svim parovima stanica. Drugim riječima, to je najmanji broj t takav da je udaljenost svakih dvaju stanica najviše t.

Šemso želi sagraditi brzu prugu tako da minimizira prečnik dobivene mreže.

Implementacijski detalji

Implementirajte funkciju

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

- `n`: broj stanica na glavnoj pruzi,

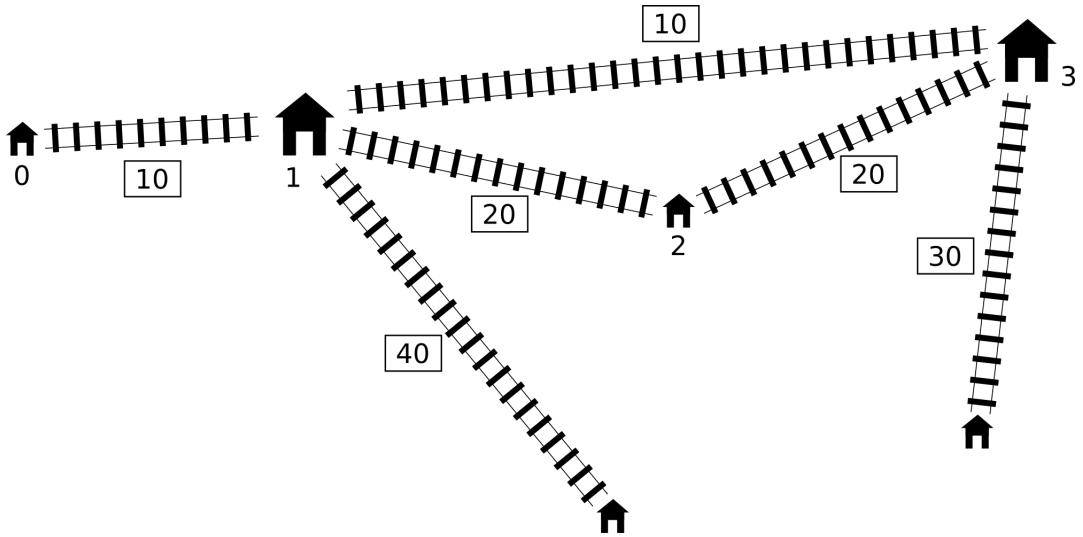
- l: udaljenosti između uzastopnih stanica na glavnoj pruzi (niz dužine *n*-*1*),
- d: dužine sporednih pruga (niz dužine *n*),
- c: dužina nove brze pruge.
- Funkcija treba vratiti najmanji mogući prečnik mreže nakon dodavanja brze pruge.

Za implementacijske detalje koristite date model datoteke(*template files*).

Primjeri

Primjer 1

Za mrežu s gornje slike grader će pozvati:
`find_shortcut(4, [10, 20, 20], [0, 40, 0, 30], 10)`
Optimalno je rješenje izgraditi brzu prugu između stanica *1* i *3*, kao na slici ispod.



Prečnik nove mreže iznosi *80* centimetara, tj. funkcija treba vratiti *80*.

Primjer 2

Grader poziva:

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

Optimalno rješenje povezuje postaje *2* i *7*, a prečnik u tom slučaju iznosi *110*.

Primjer 3

Grader poziva:

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

Optimalno rješenje povezuje postaje *1* i *2*, smanjujući dijametar na *21*.

Primjer 4

Grader poziva:

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

Povezivanje bilo kojih dvaju stanica brzom prugom dužine *3* ne poboljšava početni prečnik mreže koji iznosi *4*.

Podzadaci

U svim podzadacima, $2 \leq n \leq 1\,000\,000$, $1 \leq l_i \leq 10^9$, $0 \leq d_i \leq 10^9$, $1 \leq c \leq 10^9$.

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

Ogledni grader

Ogledni (sample) grader učitava ulaz u sljedećem obliku:

- red 1: cijeli brojevi n i c,
- red 2: cijeli brojevi l_0, l_1, $\ldots$ l_{n-2}
- red 3: cijeli brojevi d_0, d_1, $\ldots$, d_{n-1}.