VK Cup 2015 - Round 2 (unofficial online mirror, Div. 1 only)

A. Berland Miners

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The biggest gold mine in Berland consists of n caves, connected by $n - 1$ transitions. The entrance to the mine leads to the cave number 1, it is possible to go from it to any remaining cave of the mine by moving along the transitions.

The mine is being developed by the InMine Inc., k miners work for it. Each day the corporation sorts miners into caves so that each cave has at most one miner working there.

For each cave we know the height of its ceiling h_i in meters, and for each miner we know his height s_j, also in meters. If a miner's height doesn't exceed the height of the cave ceiling where he is, then he can stand there comfortably, otherwise, he has to stoop and that makes him unhappy.

Unfortunately, miners typically go on strike in Berland, so InMine makes all the possible effort to make miners happy about their work conditions. To ensure that no miner goes on strike, you need make sure that no miner has to stoop at any moment on his way from the entrance to the mine to his cave (in particular, he must be able to stand comfortably in the cave where he works).

To reach this goal, you can choose exactly one cave and increase the height of its ceiling by several meters. However enlarging a cave is an expensive and complex procedure. That's why InMine Inc. asks you either to determine the minimum number of meters you should raise the ceiling of some cave so that it is be possible to sort the miners into the caves and keep all miners happy with their working conditions or to determine that it is impossible to achieve by raising ceiling in exactly one cave.

Input

The first line contains integer n $(1 \le n \le 5\cdot 10^5)$ — the number of caves in the mine.

Then follows a line consisting of n positive integers $h_1, h_2, ..., h_n$ $(1 \le h_i \le 10^9)$, where h_i is the height of the ceiling in the i-th cave.

Next $n - 1$ lines contain the descriptions of transitions between the caves. Each line has the form a_i, b_i $(1 \le a_i, b_i \le n, a_i \ne b_i)$, where a_i and b_i are the numbers of the caves connected by a path.

The next line contains integer k $(1 \le k \le n)$.

The last line contains k integers $s_1, s_2, ..., s_k$ $(1 \le s_j \le 10^9)$, where s_j is the j-th miner's height.

Output

In the single line print the minimum number of meters that you need to raise the ceiling by in some cave so that all miners could be sorted into caves and be happy about the work conditions. If it is impossible to do, print -1. If it is initially possible and there's no need to raise any ceiling, print 0.

Sample test(s)

input

```
6
5 8 4 6 3 12
1 2
1 3
4 2
2 5
6 3
6
7 4 2 5 3 11
```

output

```
6
```

input

```
7
10 14 7 12 4 50 1
1 2
2 3
2 4
5 1
6 5
1 7
6
7 3 4 8 8 10
```

output

```
0
```

input

```
3
4 2 8
1 2
1 3
2
17 15
```

output

```
-1
```

Note

In the first sample test we should increase ceiling height in the first cave from 5 to 11. After that we can distribute miners as following (first goes index of a miner, then index of a cave): $1 \to 2, 2 \to 3, 3 \to 5, 4 \to 4, 5 \to 6, 6 \to 1$.

In the second sample test there is no need to do anything since it is already possible to distribute miners as following: $1 \to 3, 2 \to 5, 3 \to 6, 4 \to 1, 5 \to 2, 6 \to 4$.

In the third sample test it is impossible.

B. Work Group

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

One Big Software Company has n employees numbered from 1 to n. The director is assigned number 1. Every employee of the company except the director has exactly one immediate superior. The director, of course, doesn't have a superior.

We will call person a a subordinates of another person b, if either b is an immediate supervisor of a, or the immediate supervisor of a is a subordinate to person b. In particular, subordinates of the head are all other employees of the company.

To solve achieve an Important Goal we need to form a workgroup. Every person has some efficiency, expressed by a positive integer a_i, where i is the person's number. The efficiency of the workgroup is defined as the total efficiency of all the people included in it.

The employees of the big software company are obsessed with modern ways of work process organization. Today pair programming is at the peak of popularity, so the workgroup should be formed with the following condition. Each person entering the workgroup should be able to sort all of his subordinates who are also in the workgroup into pairs. In other words, for each of the members of the workgroup the number of his subordinates within the workgroup should be even.

Your task is to determine the maximum possible efficiency of the workgroup formed at observing the given condition. Any person including the director of company can enter the workgroup.

Input

The first line contains integer n $(1 \le n \le 2 \cdot 10^5)$ — the number of workers of the Big Software Company.

Then n lines follow, describing the company employees. The i-th line contains two integers p_i, a_i $(1 \le a_i \le 10^5)$ — the number of the person who is the i-th employee's immediate superior and i-th employee's efficiency. For the director $p_1 = -1$, for all other people the condition $1 \le p_i < i$ is fulfilled.

Output

Print a single integer — the maximum possible efficiency of the workgroup.

Sample test(s)

input

```
7
-1 3
1 2
1 1
1 4
4 5
4 3
5 2
```

output

```
17
```

Note

In the sample test the most effective way is to make a workgroup from employees number $1, 2, 4, 5, 6$.

C. Board Game

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp and Vasiliy love simple logical games. Today they play a game with infinite chessboard and one pawn for each player. Polycarp and Vasiliy move in turns, Polycarp starts. In each turn Polycarp can move his pawn from cell (x, y) to $(x - 1, y)$ or $(x, y - 1)$. Vasiliy can move his pawn from (x, y) to one of cells: $(x - 1, y)$, $(x - 1, y - 1)$ and $(x, y - 1)$. **Both players** are also allowed to skip move.

There are some additional restrictions — a player is forbidden to move his pawn to a cell with negative x-coordinate or y-coordinate or to the cell containing opponent's pawn The winner is the first person to reach cell $(0, 0)$.

You are given the starting coordinates of both pawns. Determine who will win if both of them play optimally well.

Input

The first line contains four integers: x_p, y_p, x_v, y_v $(0 \le x_p, y_p, x_v, y_v \le 10^5)$ — Polycarp's and Vasiliy's starting coordinates.

It is guaranteed that in the beginning the pawns are in different cells and none of them is in the cell $(0, 0)$.

Output

Output the name of the winner: "`Polycarp`" or "`Vasiliy`".

Sample test(s)

input
```
2 1 2 2
```
output
```
Polycarp
```

input
```
4 7 7 4
```
output
```
Vasiliy
```

Note

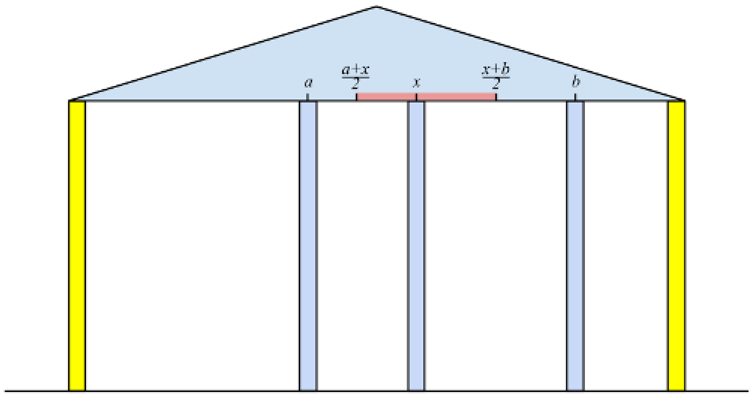
In the first sample test Polycarp starts in $(2, 1)$ and will move to $(1, 1)$ in the first turn. No matter what his opponent is doing, in the second turn Polycarp can move to $(1, 0)$ and finally to $(0, 0)$ in the third turn.

D. Landmarks

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

We have an old building with $n + 2$ columns in a row. These columns support the ceiling. These columns are located in points with coordinates $0 = x_0 < x_1 < ... < x_n < x_{n+1}$. The leftmost and the rightmost columns are special, we will call them *bearing*, the other columns are *ordinary*.

For each column we know its durability d_i. Let's consider an ordinary column with coordinate x. Let's assume that the coordinate of the closest to it column to the left (bearing or ordinary) is a and the coordinate of the closest to it column to the right (also, bearing or ordinary) is b. In this task let's assume that this column supports the segment of the ceiling from point $\frac{a+x}{2}$ to point $\frac{x+b}{2}$ (here both fractions are considered as real division). If the length of the segment of the ceiling supported by the column exceeds d_i, then the column cannot support it and it crashes after a while, and after that the load is being redistributeed between the neighbouring columns according to the same principle.



Thus, ordinary columns will be crashing for some time until the process stops at some state. One can prove that the set of the remaining columns doesn't depend on the order in which columns crash. If there are only two bearing columns left in the end, then we assume that the whole construction crashes under the weight of the roof. But if at least one ordinary column stays in addition to the bearing ones, then the building doesn't crash.

To make the building stronger, we can add one extra ordinary column of arbitrary durability d' at any (not necessarily integer) point $0 < x' < x_{n+1}$. If point x' is already occupied by an ordinary column, it is replaced by a new one.

Your task is to find out: what minimal durability can the added column have so that the building doesn't crash?

Input

The first line contains integer n $(1 \le n \le 10^5)$ — the number of ordinary columns.

The second line contains $n + 2$ integers $x_0, x_1, ..., x_n, x_{n+1}$ $(x_0 = 0, x_i < x_{i+1}$ for $0 \le i \le n, x_{n+1} \le 10^9)$ — the coordinates of the columns.

The third line contains n integers $d_1, d_2, ..., d_n$ $(1 \le d_i \le 10^9)$.

Output

Print a single number — the minimum possible durability of the column that you need to add in order to make the building stay. If you do not have to add the column, please print 0. Your answer will be checked with the relative or absolute error 10^{-4}.

Sample test(s)

input

```
2
0 20 40 100
15 40
```

output

```
10
```

input

```
3
0 4 10 28 30
9 13 5
```

output

```
0
```

E. Correcting Mistakes

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Analyzing the mistakes people make while typing search queries is a complex and an interesting work. As there is no guaranteed way to determine what the user originally meant by typing some query, we have to use different sorts of heuristics.

Polycarp needed to write a code that could, given two words, check whether they could have been obtained from the same word as a result of typos. Polycarpus suggested that the most common typo is skipping exactly one letter as you type a word.

Implement a program that can, given two distinct words S and T of the same length n determine how many words W of length $n+1$ are there with such property that you can transform W into both S, and T by deleting exactly one character. Words S and T consist of lowercase English letters. Word W also should consist of lowercase English letters.

Input

The first line contains integer n $(1 \le n \le 100\,000)$ — the length of words S and T.

The second line contains word S.

The third line contains word T.

Words S and T consist of lowercase English letters. It is guaranteed that S and T are distinct words.

Output

Print a single integer — the number of distinct words W that can be transformed to S and T due to a typo.

Sample test(s)

input
```
7
reading
trading
```
output
```
1
```

input
```
5
sweet
sheep
```
output
```
0
```

input
```
3
toy
try
```
output
```
2
```

Note

In the first sample test the two given words could be obtained only from word "**t**r**e**ading" (the deleted letters are marked in bold).

In the second sample test the two given words couldn't be obtained from the same word by removing one letter.

In the third sample test the two given words could be obtained from either word "t**or**y" or word "t**ro**y".

F. Encoding

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp invented a new way to encode strings. Let's assume that we have string T, consisting of lowercase English letters. Let's choose several pairs of letters of the English alphabet in such a way that each letter occurs in at most one pair. Then let's replace each letter in T with its pair letter if there is a pair letter for it. For example, if you chose pairs (`l`, `r`), (`p`, `q`) and (`a`, `o`), then word "`parallelogram`" according to the given encoding principle transforms to word "`qolorreraglom`".

Polycarpus already has two strings, S and T. He suspects that string T was obtained after applying the given encoding method from some substring of string S. Find all positions m_i in S ($1 \le m_i \le |S| - |T| + 1$), such that T can be obtained fro substring $S_{m_i}S_{m_i+1}\dots S_{m_i+|T|-1}$ by applying the described encoding operation by using some set of pairs of English alphabet letters

Input

The first line of the input contains two integers, $|S|$ and $|T|$ ($1 \le |T| \le |S| \le 2\cdot 10^5$) — the lengths of string S and string T, respectively.

The second and third line of the input contain strings S and T, respectively. Both strings consist only of lowercase English letters.

Output

Print number k — the number of suitable positions in string S.

In the next line print k integers $m_1, m_2, \dots, m_k$ — the numbers of the suitable positions in the increasing order.

Sample test(s)

input

```
11 5
abacabadaba
acaba
```

output

```
3
1 3 7
```

input

```
21 13
paraparallelogramgram
qolorreraglom
```

output

```
1
5
```