Problem A
XOR Pairs

XOR is a bitwise operator that evaluates the resulting bit into 1 if and only if their corresponding input bits differ (one of them is 1 while the other is 0). XOR operator is usually written with a symbol $\oplus$, or in most programming languages, the character ^(caret). For example, $(10 \oplus 6) = 12$.

```
10 =>   1010
 6 =>   0110
       ----- ⊕
        1100   => 12
```

In this problem, you are given an integer N and a set of integers $S_{1..M}$. Your task is to count how many pairs of integers $\langle A, B \rangle$ such that $1 \le A, B \le (A \oplus B) \le N$, and $(A \oplus B) \notin S$.

For example, let $N = 10$ and $S_{1..4} = \{4, 6, 7, 10\}$. There are 6 pairs of $\langle A, B \rangle$ that satisfy the condition.

- $\langle 1, 2 \rangle \rightarrow (1 \oplus 2) = 3$
- $\langle 1, 4 \rangle \rightarrow (1 \oplus 4) = 5$
- $\langle 1, 8 \rangle \rightarrow (1 \oplus 8) = 9$
- $\langle 2, 1 \rangle \rightarrow (2 \oplus 1) = 3$
- $\langle 4, 1 \rangle \rightarrow (4 \oplus 1) = 5$
- $\langle 8, 1 \rangle \rightarrow (8 \oplus 1) = 9$

Observe that a pair such as $\langle 2, 4 \rangle$ does not satisfy the condition for this example as $(2 \oplus 4) = 6$ but $6 \in S$. Another pair such as $\langle 5, 1 \rangle$ also does not satisfy the condition as it violates the requirement $A, B \le (A \oplus B)$.

Input

Input begins with a line containing two integers N M ($1 \le N \le 10^6$; $1 \le M \le 100\,000$) representing the given N and the size of the set of integers $S_{1..M}$. The next line contains M integers S_i ($1 \le S_i \le 10^6$) representing the set of integers $S_{1..M}$.

Output

Output contains an integer in a line representing the number of $\langle A, B \rangle$ such that $1 \le A, B \le (A \oplus B) \le N$ and $(A \oplus B) \notin S_{1..M}$.

Sample Input #1

```
10 4
4 6 7 10
```

Sample Output #1

```
6
```

Explanation for the sample input/output #1

This is the example from the problem description.

Sample Input #2

```
8 5
4 3 5 8 1
```

Sample Output #2

```
10
```

Explanation for the sample input/output #2

There are 10 pairs of $\langle A, B\rangle$ that satisfy the condition.

- $\langle 1, 6\rangle \to (1 \oplus 6) = 7$
- $\langle 2, 4\rangle \to (2 \oplus 4) = 6$
- $\langle 2, 5\rangle \to (2 \oplus 5) = 7$
- $\langle 3, 4\rangle \to (3 \oplus 4) = 7$
- $\langle 3, 5\rangle \to (3 \oplus 5) = 6$
- $\langle 4, 2\rangle \to (4 \oplus 2) = 6$
- $\langle 4, 3\rangle \to (4 \oplus 3) = 7$
- $\langle 5, 2\rangle \to (5 \oplus 2) = 7$
- $\langle 5, 3\rangle \to (5 \oplus 3) = 6$
- $\langle 6, 1\rangle \to (6 \oplus 1) = 7$

Sample Input #3

```
20 7
3 7 18 15 12 18 19
```

Sample Output #3

```
50
```

Sample Input #4

```
5 6
1 2 3 4 5 6
```

Sample Output #4

```
0
```

Problem B

Bicycle Tour

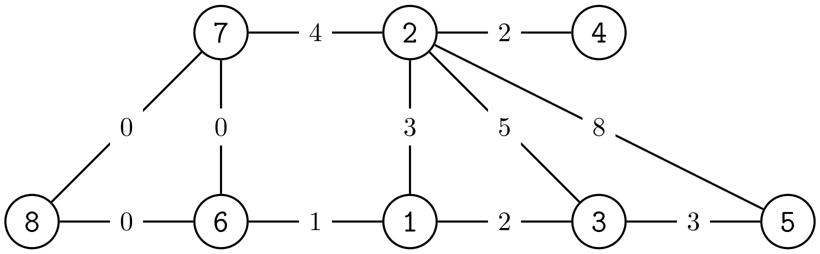
There are N junctions in Jakarta (numbered from 1 to N) that are connected by M bidirectional roads such that from any junction you can reach any other junctions by going through one or more roads. The i^{th} junction has a height of H_i.

Ahmad loves to cycle, especially on flat roads. He argues that you need more power to cycle if the road is uphill or downhill. Specifically, to cycle on a road connecting junction i and junction j, you will require a power of $|H_i - H_j|$. The required power to cycle a set of roads is equal to the highest required power to cycle each of those roads. For example, let there be 3 roads where each requires a power of 10, 12, and 7 to cycle, then the required power to cycle through all those roads is $\max(10, 12, 7) = 12$.

A cycling route from junction i is defined as a tour that starts at junction i, going to one or more other junctions while not using the same road more than once, and ends at the same junction i.

Ahmad would like to find a cycling route from each junction that has a minimum required power, and that is your task in this problem. For each junction, you need to output the required power of a cycling route from that junction that has the minimum required power. There might be a case where a cycling route from a junction is not possible; in such a case, the output should be -1 for the respective junction.

For example, let $N = 8$, $H_{1..8} = \{5, 2, 7, 0, 10, 6, 6, 6\}$, $M = 11$, and the roads be shown in the following figure. The label on each node indicates the junction number while the number on each edge indicates the required power to cycle through that road. Notice that the required power to cycle through each road can be obtained from the given $H_{1..8}$.



The following are the cycling routes with the minimum required power from each junction:

- Junction 1: $1 \rightarrow 2 \rightarrow 7 \rightarrow 6 \rightarrow 1$, with a required power of 4.
- Junction 2: $2 \rightarrow 1 \rightarrow 6 \rightarrow 7 \rightarrow 2$, with a required power of 4.
- Junction 3: $3 \rightarrow 2 \rightarrow 1 \rightarrow 3$, with a required power of 5.
- There is no possible cycling route from junction 4.
- Junction 5: $5 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 5$, with a required power of 8.
- Junction 6: $6 \rightarrow 7 \rightarrow 8 \rightarrow 6$, with a required power of 0.

- Junction 7: $7 \rightarrow 8 \rightarrow 6 \rightarrow 7$, with a required power of 0.
- Junction 8: $8 \rightarrow 6 \rightarrow 7 \rightarrow 8$, with a required power of 0.

Input

Input begins with a line containing two integers N M ($2 \leq N \leq 100\,000$; $N - 1 \leq M \leq 200\,000$) representing the number of junctions and the number of bidirectional roads, respectively. The second line contains N integers H_i ($0 \leq H_i \leq 10^9$) representing the height of the i^{th} junction. The next M lines, each contains two integers u_i v_i ($1 \leq u_i < v_i \leq N$) representing a bidirectional road connecting junction u_i and v_i. It is guaranteed that from any junction you can reach any other junctions by going through one or more roads. It is also guaranteed that for any pairs of junctions (u, v), there is at most one bidirectional road connecting junction u and junction v.

Output

Output contains N integers in a line each separated by a single space. The i^{th} integer represents the required power of a cycling route from junction i with the minimum required power. If there is no possible cycling route from junction i, then the i^{th} integer is -1.

Sample Input #1

```
8 11
5 2 7 0 10 6 6 6
1 2
1 3
2 3
2 4
2 5
2 7
3 5
1 6
6 7
6 8
7 8
```

Sample Output #1

```
4 4 5 -1 8 0 0 0
```

Explanation for the sample input/output #1

This is the example from the problem description.

Sample Input #2

```
4 5
10 20 30 40
1 2
1 3
1 4
2 3
3 4
```

Sample Output #2

```
20 20 20 30
```

Sample Input #3

```
5 4
72 35 22 49 108
1 2
2 3
3 4
4 5
```

Sample Output #3

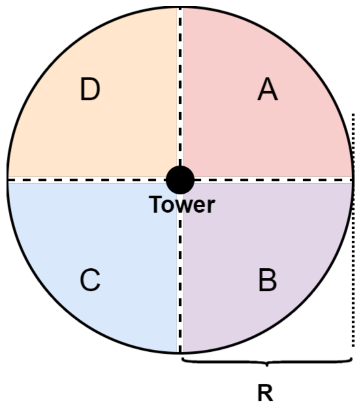
```
-1 -1 -1 -1 -1
```

Problem C
Energy Generation

You are given a particle collision energy generator. The generator is a two-dimensional field with N towers. The i^{th} tower is located at position (X_i, Y_i), and all towers have the same field of effect with a radius of R.

Each tower radiates 4 types of particles at a fixed configuration. Specifically, each tower radiates each particle $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ on its own quadrant (area separated by both its x-axis and y-axis) in a clockwise order, respectively. The tower does not radiate any particle at its x-axis or y-axis.

Initially, each tower is oriented at a multiple of 90° angle. Let $(\,\cdot\,,\,\cdot\,,\,\cdot\,,\,\cdot\,)$ represents the particles radiated by a tower on its upper right, lower right, lower left, and upper left quadrant, respectively.



- If the tower is oriented at 0°, then it radiates $(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D})$ as illustrated in the figure on the right.
- If the tower is oriented at 90° (rotated clockwise from 0°), then it radiates $(\mathcal{D}, \mathcal{A}, \mathcal{B}, \mathcal{C})$.
- If the tower is oriented at 180°, then it radiates $(\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{B})$.
- If the tower is oriented at 270°, then it radiates $(\mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{A})$.

An interesting phenomenon might occur on the i^{th} tower when there is a j^{th} tower such that the following conditions are satisfied.

(1) $X_i \neq X_j$,

(2) $Y_i \neq Y_j$, and

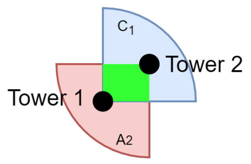
(3) their Euclidean distance is **no larger than** R.

The interesting phenomenon is as follow. Let p be the particle that is radiated by the i^{th} tower on the quadrant where the j^{th} tower is located, and q be the particle that is radiated by the j^{th} tower on the quadrant where the i^{th} tower is located. For simplicity, let's say $\langle p, q\rangle$ are the particles radiated by their facing sides.

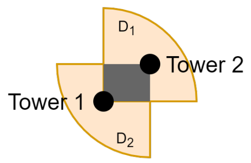
- If their facing sides are radiating either $\langle \mathcal{A}, \mathcal{C}\rangle$, $\langle \mathcal{C}, \mathcal{A}\rangle$, $\langle \mathcal{B}, \mathcal{D}\rangle$, or $\langle \mathcal{D}, \mathcal{B}\rangle$, then the i^{th} tower will generate an interaction energy of G.
- If their facing sides are radiating the same particles, $\langle \mathcal{A}, \mathcal{A}\rangle$, $\langle \mathcal{B}, \mathcal{B}\rangle$, $\langle \mathcal{C}, \mathcal{C}\rangle$, or $\langle \mathcal{D}, \mathcal{D}\rangle$, then the i^{th} tower will generate an interaction energy of $-G$; in other words, it consumes G energy.
- If their facing sides are radiating any one of the remaining 8 possible combination of particles that are not mentioned above, then the i^{th} tower is not interacting with the j^{th} tower. In other words, there is no energy generated from this pair of towers.

This phenomenon applies both ways (from each tower's perspective) and **stacks** indefinitely if there are multiple towers satisfying the conditions.

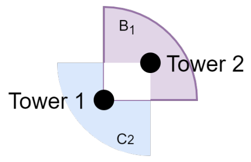
Here are some examples of the energy generated from a pair of towers.



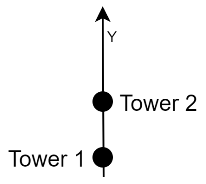
Radiates <A, C>
Each tower generates
+G energy



Radiates <D, D>
Each tower generates
-G energy



Radiates <B, C>
There is no interaction
between these towers



$X_1 = X_2$
There is no interesting
phenomenon that will occur
out of these towers

Each tower also passively generates its own energy. Initially, each tower generates P energy by itself.

You have the option to change the orientation of each tower by rotating it, possibly taking advantage of the interesting phenomenon and increasing the total amount of energy generated. Each $90°$ rotation in **any direction** (either CW/clockwise or CCW/counterclockwise) causes the tower to produce P **less energy** passively. A tower that is rotated $90°$ in any direction will produce 0 energy passively and a tower that is rotated $180°$ (rotated $90°$ twice in the same direction) will produce $-P$ energy (or in other words, consume P energy). Note that you can only rotate any tower by a multiple of $90°$.

Your task in this problem is to find the maximum amount of total energy that can be generated by changing the orientation of zero or more towers in any way. The total energy generated in a configuration is the sum of energy generated by each tower either passively or due to interaction with another tower in the configuration.

Input

Input begins with a line containing 4 integers N R G P ($1 \leq N \leq 50$; $1 \leq R, G, P \leq 1000$) representing the number of towers, the radius of effect of all towers, the tower interaction energy constant, and the initial energy passively generated by each tower, respectively. The following N lines contain 3 integers X_i Y_i O_i ($-1000 \leq X_i, Y_i \leq 1000$; $O_i \in \{0, 90, 180, 270\}$) representing the position and the initial orientation of the i^{th} tower. It is guaranteed that no two towers have the same position.

Output

Output contains an integer in a line representing the maximum amount of total energy that can be generated out of all possible configurations of the towers.

Sample Input #1

```
3 10 10 15
0 0 0
2 2 180
100 100 180
```

Sample Output #1

```
35
```

Explanation for the sample input/output #1

The maximum amount of total generated energy can be obtained by rotating the 2^{nd} tower for $180°$. No interesting phenomenon can happen on the 3^{th} tower as its Euclidean distance to any other towers is larger than R. By not rotating the 3^{th} tower, it generates 15 passive energy. Rotating the 2^{nd} tower for $180°$ will cause it to be oriented at $0°$. An interesting phenomenon occurs on the 1^{st} tower and the 2^{nd} tower as they satisfy all the conditions and their facing side radiates $\langle \mathcal{A}, \mathcal{C} \rangle$ (or $\langle \mathcal{C}, \mathcal{A} \rangle$ from the 2^{nd} tower's perspective). The 1^{st} tower will produce $15 + 10 = 25$ from its passive energy generation and interaction with the 2^{nd} tower, while the 2^{nd} tower will produce $-15 + 10 = -5$ energy. These 3 towers generate a total energy of $25 - 5 + 15 = 35$ in this configuration.

Sample Input #2

```
3 10 1 1000
0 0 0
2 2 0
-4 4 180
```

Sample Output #2

```
2998
```

Explanation for the sample input/output #2

The maximum amount of total generated energy can be obtained by not doing any rotation. These 3 towers are interacting with each other at their current orientation. The 1^{st} and 2^{nd} towers each generates $1+(-1) = 0$ interaction energy while the 3^{rd} tower generates $-1 + (-1) = -2$ interaction energy. Each tower also passively generates 1000 energy. The total energy generated in this configuration is 2998.

Sample Input #3

```
4 10 1000 1
0 0 0
0 2 90
2 0 180
2 2 270
```

Sample Output #3

```
4002
```

Explanation for the sample input/output #3

The maximum amount of total generated energy can be obtained by rotating the 1^{st} tower for 90° CCW and the 2^{nd} tower for 90° CW. With these rotations, there are only interactions between the 1^{st} and 4^{th} towers, and the 2^{nd} and 3^{rd} towers due to the interesting phenomenon. The 1^{st} tower generates $0 + 1000 = 1000$ energy, the 2^{nd} tower generates $0 + 1000 = 1000$ energy, the 3^{rd} tower generates $1 + 1000 = 1001$ energy, and the 4^{th} tower generates $1 + 1000 = 1001$ energy. The total energy generated in this configuration is 4002.

Problem D
Uniform Maker

The International Costumes and Props Company (ICPC) received an order from a client to produce N pennants each containing the same word. However, due to some miscommunication between the account manager and the client, not all the produced pennants have the same word although all of them have a word of the same length. Reproducing those pennants is very costly as the ICPC only uses a certain type of rare fabric in their production.

Fortunately, the client didn't specify the word that they want to be in the pennants. In fact, the client will be satisfied if and only if all the pennants have the same word.

The ICPC has a special technique to change one character in a word into some other character. It is expensive, albeit not as expensive as reproducing a new pennant. Therefore, the ICPC has to minimize the number of times they have to use such a technique. Your task in this problem is to help the ICPC to determine the minimum total number of characters that need to be changed so that the client will be satisfied.

For example, let there be $N = 6$ pennants with the following words: `calf`, `palm`, `book`, `icpc`, `ball`, and `room`. The total number of characters than need to be changed can be minimized if all the words are changed into `balm`.

- `calf` $\rightarrow 2$ characters: `b**m`
- `palm` $\rightarrow 1$ characters: `b***`
- `book` $\rightarrow 3$ characters: `*alm`
- `icpc` $\rightarrow 4$ characters: `balm`
- `ball` $\rightarrow 1$ characters: `***m`
- `room` $\rightarrow 3$ characters: `bal*`

The symbol `*` represents an unchanged character. There are a total of 14 characters that need to be changed in this example.

Input

Input begins with a line containing two integers N M ($2 \leq N \leq 100$; $1 \leq M \leq 100$) representing the number of pennants and the length of each word in the pennant, respectively. The next N line each contains a string S_i ($|S_i| = M$) representing the word on the i^{th} pennant. Each string only contains lowercase alphabetical characters.

Output

Output contains an integer in a line representing the minimum total number of characters that need to be changed so that the client will be satisfied.

Sample Input #1

```
6 4
calf
palm
book
icpc
ball
room
```

Sample Output #1

```
14
```

Explanation for the sample input/output #1

This is the example from the problem description.

Sample Input #2

```
3 11
goodluckfor
icpcjakarta
contestants
```

Sample Output #2

```
19
```

Sample Input #3

```
5 14
helpiamtrapped
inanincfactory
forthreemonths
withoutfoodand
drinkandshower
```

Sample Output #3

```
49
```

Problem E
Concerto de Pandemic

There are N cities numbered from 1 to N. There is a bi-directional road connecting city i to city $i + 1$ for $1 \le i < N$ and city N to city 1. Each road takes 1 day to travel.

Due to a pandemic situation, there are M cities that impose a quarantine order for any visitors to mitigate the pandemic spread in those cities. Specifically, whenever someone visits city C_i, they will be quarantined for a duration of exactly T_i days in a government-provided facility in that city. The order applies to any visitor including those who don't intend to stay in that city, e.g., only transiting.

Nawan is a rising young musician who already has K die-hard fans. The i^{th} fan lives in city D_i, and surprisingly enough, **none** of the fans live in a city that imposes a quarantine order for visitors. Nawan has just released an album and now he wants to hold concerts for his die-hard fans. Despite rejections from his team, Nawan insists that the concert must be held live and in person; he believes that he wouldn't be able to convey his "musical feeling" to his fans through a virtual concert.

After considering the budget and their resource, Nawan and his team agree to hold at most P concerts. Moreover, These concerts can only be held in cities that are **not** imposing any quarantine order for visitors. Nawan has contacted all of his fans and each of them agrees to attend only 1 concert. The only remaining issue is in choosing the cities where Nawan should have a concert.

Each of the fans will attend a concert in which the venue requires the minimum travel time from their city. Each concert venue has no maximum capacity. Nawan wishes to hold the concerts in at most P cities such that the longest travel time among all of his fans is as minimum as possible. Since Nawan needs to practice and prepare for the concerts, he asked you to choose the cities in which he should have a concert such that the longest required travel time by any fan is as minimum as possible; you only need to output the minimum longest travel time.

For example, let $N = 10$, $M = 4$, $C_{1..4} = \{1, 4, 6, 7\}$, $T_{1..4} = \{2, 4, 2, 5\}$, $K = 3$, $D_{1..3} = \{2, 5, 8\}$, and $P = 2$. In this example, the concert venues should be in city 5 and city 10 with a longest travel time of only 4 days.

- The 1^{st} fan at city 2 will go to the concert at city 10, i.e. $2 \to 1$(quarantined for 2 days) $\to 10$, for a total travel time of 4 days.
- The 2^{nd} fan at city 5 will go to the concert at city 5 where no travel is needed.
- The 3^{rd} fan at city 8 will go to the concert at city 10. i.e. $8 \to 9 \to 10$, for a total travel time of 2 days.

Input

Input begins with a line containing four integers N M K P ($1 \le M < N \le 200\,000$; $1 \le K, P \le N - M$) representing the number of cities, the number of cities that impose a quarantine order, the number of Nawan's die-hard fans, and the maximum number of concerts to be held, respectively. The next M lines each contains two integers C_i T_i ($1 \le C_i \le N$; $1 \le T_i \le 200\,000$) representing the city that has a quarantine order and its quarantine duration, respectively. It is guaranteed that all C_i are unique. The last line contains K integers

D_i $(1 \leq D_i \leq N)$ representing the city in which the i^{th} fan lives in. It is guaranteed that no fan lives in a city that imposes a quarantine order for visitors, and all fans live in a different city.

Output

Output contains an integer in a line representing the minimum longest travel time needed by any fan to reach a concert venue.

Sample Input #1

```
10 4 3 2
1 2
4 4
6 2
7 5
2 5 8
```

Sample Output #1

```
4
```

Explanation for the sample input/output #1

This is the example from the problem description.

Sample Input #2

```
8 1 3 5
1 5
4 2 7
```

Sample Output #2

```
0
```

Explanation for the sample input/output #2

Nawan can hold a private concert for each of his fans, i.e. the concert venues should be in cities 2, 4, and 7.

Sample Input #3

```
5 2 2 1
1 14
2 14
3 5
```

Sample Output #3

```
1
```

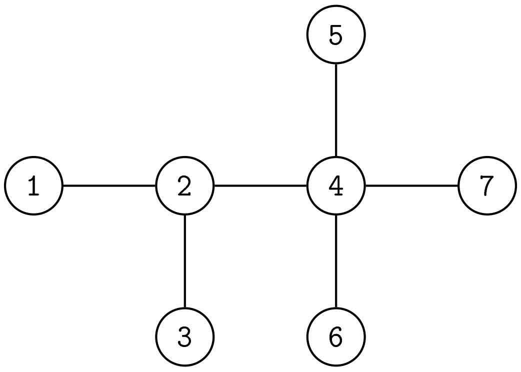
Problem F

Not One

The greatest common divisor (GCD) of a set of positive integers S is defined as the largest positive integer d such that d is a divisor for all elements in S, e.g., $\text{GCD}(10) = 10$, $\text{GCD}(6, 9) = 3$, $\text{GCD}(20, 12, 16, 36) = 4$.

In this problem, you are given a tree of N nodes where each node is numbered from 1 to N and has a positive integer A_i assigned to it. Your task is to find the size of the largest subtree such that the GCD of the weight of all nodes in that subtree is not 1, or output 0 if there is no such a subtree. A tree T' is a subtree of T if and only if T' is connected and is a subset of T. The size of a subtree is the number of nodes in that subtree.

For example, consider the following tree of $N = 7$ nodes where $A_{1..7} = \{10, 5, 8, 6, 10, 6, 4\}$.



In this example, there are 15 subtrees where the GCD of all its nodes' weight is not 1, i.e. seven subtrees of size 1, four subtrees of size 2, three subtrees of size 3, and one subtree of size 4 (the largest). The largest subtree contains nodes 4, 5, 6, and 7, and the GCD of their weights is $\text{GCD}(A_4, A_5, A_6, A_7) = \text{GCD}(6, 10, 6, 4) = 2$.

Input

Input begins with a line containing an integer N ($2 \le N \le 100\,000$) representing the number of nodes in the given tree. The next line contains N integers A_i ($1 \le A_i \le 10^6$) representing the weight of the i^{th} node. The next $N - 1$ line each contains two integers u_j v_j ($1 \le u_j < v_j \le N$) representing an edge connecting node u_j and node v_j. It is guaranteed that the given tree is connected.

Output

Output contains an integer in a line representing the size of the largest subtree such that the GCD of all its nodes' weight is not 1. If there is no such a subtree, output 0 in a line.

Sample Input #1

```
7
10 5 8 6 10 6 4
1 2
2 3
2 4
4 5
4 6
4 7
```

Sample Output #1

```
4
```

Explanation for the sample input/output #1

This is the example from the problem statement.

Sample Input #2

```
4
1 1 1 1
1 2
2 3
3 4
```

Sample Output #2

```
0
```

Explanation for the sample input/output #2

There is no subtree where the GCD of all its nodes' weight is not 1 in this case.

Sample Input #3

```
5
100 100 100 100 100
3 4
1 2
3 5
2 4
```

Sample Output #3

```
5
```

Problem G
Greedy Knapsack

Budi stumbled upon the classical 0-1 Knapsack Problem that he has learned from his class in university. There are N items numbered from 1 to N. The i^{th} item has a positive weight of W_i and a positive value of V_i. The objective is to take zero or more items such that their total weight does not exceed M while their total value is maximum.

It has been a while since the last time Budi works on this problem, and he couldn't remember how to solve it. So, Budi devises a greedy algorithm in an attempt to solve this problem. The algorithm goes like this. Each item is processed one by one from the 1^{st} item to the N^{th} item in sequential order. If the i^{th} item can be taken such that the total weight does not exceed M, then you should take that item; otherwise, ignore it. Output the total value of all taken items.

The greedy algorithm can be presented with the following pseudocode.

```
GreedyKnapsack(W[1..N], V[1..N], M):
   total_value = 0
   total_weight = 0
   for i = 1 to N:
      if total_weight + W[i] <= M:
         total_weight = total_weight + W[i]
         total_value = total_value + V[i]
   return total_value
```

Of course, Budi realized that such a greedy algorithm might not produce the optimum solution, but at this point, he doesn't care. Budi noticed that the output of such a greedy algorithm is sensitive to M. So, he decides to run the greedy algorithm with various M ranging from 1 to a given upper limit T and determines what is the largest output that he can get.

Your task in this problem is to help Budi to determine the largest output that he can get from the greedy algorithm by varying the input M from 1 to T.

For example, let $N = 5$, $T = 10$, $W_{1..5} = \{10, 1, 2, 3, 4\}$, and $V_{1..5} = \{1, 1, 1, 1, 1\}$. In this example, the largest output that can be obtained is 3, e.g., with $M = 6$; the greedy algorithm will take the 2^{nd}, 3^{rd}, and 4^{th} items with a total weight of $1 + 2 + 3 = 6$ and a total value of $1 + 1 + 1 = 3$. Notice that if we set $M = 10$, then the greedy algorithm will take only the 1^{st} item with a total weight of 10 and a total value of 1.

Input

Input begins with a line containing two integers N T ($1 \le N \le 100\,000$; $1 \le T \le 10^{10}$) representing the number of items and the upper limit, respectively. The second line contains N integers W_i ($1 \le W_i \le 100\,000$) representing the weight of the i^{th} item. The third line contains N integers V_i ($1 \le V_i \le 100\,000$) representing the value of the i^{th} item.

Output

Output contains an integer in a line representing the largest output that can be obtained by the presented greedy algorithm.

Sample Input #1

```
5 10
10 1 2 3 4
1 1 1 1 1
```

Sample Output #1

```
3
```

Explanation for the sample input/output #1

This is the example from the problem statement.

Sample Input #2

```
5 10000000000
10 1 2 3 4
30 2 15 7 11
```

Sample Output #2

```
65
```

Explanation for the sample input/output #2

You can set M to be large enough such that all items can be taken, i.e. $M \geq 20$.

Sample Input #3

```
5 20
4 9 5 1 3
203 175 131 218 304
```

Sample Output #3

```
900
```

Explanation for the sample input/output #3

The largest output can be obtained with $M = 17$. The 1^{st}, 2^{nd}, 4^{th}, and 5^{th} will be taken with a total weight of $4 + 9 + 1 + 3 = 17$ and a total value of $203 + 175 + 218 + 304 = 900$.

Problem H
Cell Game

Two brothers, Aldo and Bondan, are stuck in their home as their city is going into lockdown again due to the worsening situation of the COVID-19 pandemic. They have finished their semester and are on holiday, but what kind of holiday can you enjoy if you cannot get out of your house. However, boredom does spark creativity. They created a new game during their boring holiday.

The game is played on a board of R rows and C columns where each cell contains zero or one token. Each token is painted with one of the 26 colors that is represented by a character from 'a' to 'z'. There is at least one token on the board.

Two opposing players play alternatingly. In their turn, the player chooses one token and move it to a not necessarily empty adjacent cell (i.e. in north, south, west, or east direction). A move is a **good move** if and only if the player moves a token of color x to a cell that already contains a token of the same color x. The objective of the game is to have as many good moves as possible. The player with strictly more good moves wins the game. If both players have the same number of good moves, then it's a tie and no one wins.

This game can be played indefinitely. However, Aldo and Bondan agree to play it for a total of 10^{100} moves for both with Aldo having the first move.

Bondan thinks that this kind of game is boring, so he decides to play it lazily. Whenever it is Bondan's turn, he will choose the same token that has just been moved by Aldo in his last previous turn, and move that token to an adjacent cell uniformly at random.

Despite that, Bondan doesn't like to lose. Before the game starts, he might need to alter the board by changing the board size or moving the tokens' initial location such that there is exactly zero chance for Aldo to win the game even though Bondan plays lazily. Specifically, the board can be extended from $R \times C$ into $R' \times C'$ where $R' \geq R$ and $C' \geq C$, and each token's initial location can be moved to another cell while ensuring that each cell contains at most one token. No token can be discarded and no new token can be added to the board.

Your task in this problem is to find a new board setting such that there is zero chance for Aldo (the first player) to win the game with a total of 10^{100} played moves although Bondan plays it lazily. The new board setting should have a minimum total number of cells. If there is more than one solution, you only need to output (any) one of them.

Input

Input begins with a line containing two integers R C ($1 \leq R, C \leq 1000$) representing the initial number of rows and the number of columns in the board, respectively. It is guaranteed that there are at least 2 cells on the board. Each of the next R lines contains C characters representing the initial board condition. The character '`.`' represents an empty cell while a character from the lowercase alphabets ('`a-z`') represents a token with such a color. It is guaranteed that there is at least 1 token on the board.

Output

Output begins with a line containing two integers R' C' representing the number of rows and the number of columns in the new board, respectively. Each of the next R' lines contains C' characters representing the new board condition. The character '`.`' represents an empty cell while a character from the lowercase alphabets ('`a`–`z`') represents a token with such a color. The new board should guarantee a zero chance for Aldo (the first player) to win the game with a total of 10^{100} played moves although Bondan plays it lazily. It should have a minimum total number of cells and should have the same set of tokens as the initial board.

Sample Input #1

```
2 3
ab.
c.d
```

Sample Output #1

```
2 3
ab.
c.d
```

Explanation for the sample input/output #1

There are no two tokens with the same color, thus, it is not possible for any player to have a good move. Aldo cannot win this game.

Sample Input #2

```
2 3
aa.
c.d
```

Sample Output #2

```
2 3
a.a
c.d
```

Sample Input #3

```
1 2
oo
```

Sample Output #3

```
1 3
o.o
```

Problem I
Stable Planetary System

Emma, a young astronomer, has just received an award from the International Cosmic and Planetary Committee (the ICPC) for her discovery of a new planetary system along with her novel method in recording the planetary system.

The new planetary system consists of a single star with N planets orbiting it. All these planets are orbiting on the same plane such that they can be drawn easily on a 2-dimensional surface. Moreover, each of these planets has a perfectly circular orbit centered at the star. All planets revolve around the star in the same counter-clockwise direction.

To simplify this problem, the star and each planet are represented by a point and their sizes are negligible. The star is located at the origin.

Through her research, Emma managed to record the location of each planet along with their revolution period (the time needed for a planet to complete one full orbit around the star). Specifically, Emma records $\langle R_i, \theta_i, T_i \rangle$ for each planet where $\langle R_i, \theta_i \rangle$ is its polar coordinate and T_i is its revolution period. A polar coordinate $\langle R_i, \theta_i \rangle$ means that its distance to the star is R_i and its angle from the polar axis (positive x-axis) is θ_i. To have an integer input, the degree θ_i is multiplied by 1000, so the value is between 0 and $359\,999$ (inclusive) representing any degree between 0° and 359.999°.

Emma records all these planets' positions at the same time, i.e. at time $t = 0$. Observe that the position of each planet may differ on $t > 0$ depends on their revolution period, e.g., a planet at $\langle 3, 180\,000 \rangle$ at $t = 0$ with a revolution period of 4 unit will be at $\langle 3, 270\,000 \rangle$ when $t = 1$ and at $\langle 3, 315\,000 \rangle$ when $t = 1.5$; at $t = 4$ (its revolution period), this planet will complete its orbit and return to $\langle 3, 180\,000 \rangle$.

Emma hypothesizes that a planetary system will be much more stable if the planets are not too close to each other. As no analysis has been done on this new planetary system, Emma wants to know the minimum distance among all pairs of planets in this new planetary system. To measure the distance, Emma simply uses the Euclidean distance on a 2-dimensional plane.

The distance between two planets is defined as the closest distance that these two planets can ever achieve, i.e. for any $t \in [0, \infty)$. Note that t can be a real number. For example, let there be two planets $\langle 3, 180\,000, 4 \rangle$ and $\langle 4, 0, 2 \rangle$. The distance between these two planets at $t = 0$ is 7 unit; their positions are the opposite to each other from the star, i.e. the first planet is at 180° while the second planet is at 0°. When $t = 2$, the first planet will travel half of its orbit and its position becomes $\langle 3, 0 \rangle$, at the opposite side from its position when $t = 0$. On the other hand, the second planet will travel one complete orbit so that its position is the same as in when $t = 0$, i.e. $\langle 4, 0 \rangle$. In this situation, their distance is 1 unit. This is the closest distance these two planets will ever achieve, thus, the distance between these two planets is 1 unit.

Given the position at time $t = 0$ and the revolution period of each planet, your task is to determine the minimum distance among all pairs of planets. If there exist two planets colliding (being at the same position) for any $t > 0$, then simply output 0; such a planetary system is not stable.

Input

Input begins with an integer N ($2 \leq N \leq 200\,000$) representing the number of planets in the new planetary system. The next N lines, each contains three integers R_i θ_i T_i ($1 \leq R_i, T_i \leq 10^8$; $0 \leq \theta_i < 360\,000$) representing the i^{th} planet's position in a polar coordinate $\langle R_i, \theta_i \rangle$ at $t = 0$, and its revolution period, respectively. The polar coordinate $\langle R_i, \theta_i \rangle$ means that the planet's distance to the star is R_i and its angle from the polar axis is θ_i. It is guaranteed that there are no two planets with the same position at $t = 0$.

Output

Output contains a single real number representing the minimum distance among all pairs of planets as defined in the problem description. If there exist two planets colliding at a certain time, then output 0. Your answer will be accepted as long as its absolute or relative error does not exceed 10^{-6}.

Sample Input #1

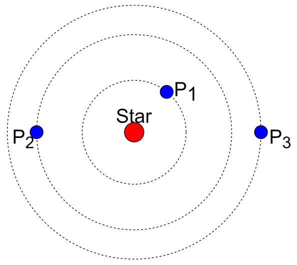
```
3
1 55555 4
3 180000 4
4 0 2
```

Sample Output #1

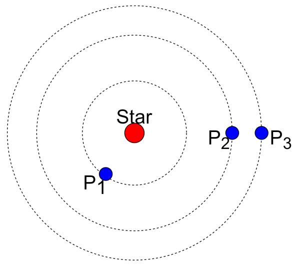
```
1
```

Explanation for the sample input/output #1

At $t = 0$, the planets' positions are shown as in the following figure.



The minimum distance can be obtained from the pair of the 2^{nd} and 3^{rd} planet at time $t = 2$. The following figure shows the planets' positions at $t = 2$.



Sample Input #2

```
3
100 0 2
100 270000 1
9 2000 3
```

Sample Output #2

```
0
```

Explanation for the sample input/output #2

The 1^{st} and 2^{nd} planet will collide at $t = 0.5$, i.e. when both are at $\langle 100, 90000 \rangle$.

Sample Input #3

```
2
10 0 2
1 90000 2
```

Sample Output #3

```
10.0498756211
```

Explanation for the sample input/output #2

The distance between the 1^{st} and 2^{nd} planet will always be the same all the time.

Problem J
Feeder Robot

Vincent switches his retirement plan from raising horses and goats into raising chickens. He procures N chickens and lays each of them into their own coop (hen house). The coops are placed on a single line numbered sequentially from 1 at the left-most to N at the right-most.

To make his retirement blissful (or at least he thought), Vincent buys a feeder robot. This feeder robot is to be loaded with M pellets and it will distribute them for the chickens to feed on. The feeder robot will move from one coop to an adjacent coop and distribute 1 pellet to each coop it visits. If a coop is visited x times by the robot including the robot's initial position, then it will get x pellets.

However, Vincent has just noticed that he cannot control how the robot moves. Let the robot be in front of coop p. If the robot still has pellets to distribute, it will move to an adjacent coop (coop $p-1$ or $p+1$) at random and distributes 1 pellet to that coop. This process repeats until the robot has no more pellets to distribute. Note that if $p = 1$, then the robot will move to coop 2; similarly, if $p = N$, then the robot will move to coop $N-1$.

Since Vincent dislikes you even though you are his only friend, he challenges you to a problem. The challenge is to count how many possible pellets distributions are there if the robot starts at coop A. A pellets distribution is defined as a tuple $\langle R, S_{1..N} \rangle$ where R is the final position of the robot and S_i is the number of pellets coop i gets. Two distributions are different if and only if the final position of the robot differs or there is a coop that gets a different number of pellets. The robot's movement or the order when the coop gets a pellet does not matter.

Since the output can be quite big, Vincent requires you to give him the non-negative remainder when the output is divided by $998\,244\,353$. Believing that you cannot solve it, Vincent agrees to award you with a hefty reward if you managed to solve his challenge.

For example, let $N = 4$, $M = 3$, and $A = 2$. In this example, there are 3 different pellets distributions.

- $\langle 2, \{1, 2, 0, 0\} \rangle$: The robot ends at coop 2 and the number of pellets each coop gets is $\{1, 2, 0, 0\}$. The robot's movement that causes this distribution is: The robot starts at coop 2 and distributes 1 pellet to coop 2; it moves to coop 1 and distributes 1 pellet to coop 1; it moves to coop 2 and distributes 1 pellet to coop 2. In a simple notation, the robot's movement is $2 \to 1 \to 2$.
- $\langle 2, \{0, 2, 1, 0\} \rangle$: The robot ends at coop 2 and the number of pellets each coop gets is $\{0, 2, 1, 0\}$. The robot's movement is $2 \to 3 \to 2$.
- $\langle 4, \{0, 1, 1, 1\} \rangle$: The robot ends at coop 4 and the number of pellets each coop gets is $\{0, 1, 1, 1\}$. The robot's movement is $2 \to 3 \to 4$.

Input

Input contains three integers N M A ($2 \le N \le 100\,000$; $1 \le M \le 200\,000$; $1 \le A \le N$) representing the number of coops, the number of pellets, and the starting position of the feeder robot, respectively.

Output

Output contains an integer in a line representing the non-negative remainder when the number of different pellets distributions is divided by $998\,244\,353$.

Sample Input #1

```
4 3 2
```

Sample Output #1

```
3
```

Explanation for the sample input/output #1

This is the example from the problem statement.

Sample Input #2

```
3 5 2
```

Sample Output #2

```
3
```

Explanation for the sample input/output #2

There are 3 different pellets distributions in this case.

- $\langle 2, \{2, 3, 0\}\rangle$: The robot ends at coop 2 and the number of pellets each coop gets is $\{2, 3, 0\}$. The robot's movement is $2 \rightarrow 1 \rightarrow 2 \rightarrow 1 \rightarrow 2$.
- $\langle 2, \{0, 3, 2\}\rangle$: The robot ends at coop 2 and the number of pellets each coop gets is $\{0, 3, 2\}$. The robot's movement is $2 \rightarrow 3 \rightarrow 2 \rightarrow 3 \rightarrow 2$.
- $\langle 2, \{1, 3, 1\}\rangle$: The robot ends at coop 2 and the number of pellets each coop gets is $\{1, 3, 1\}$. The robot's movement is $2 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 2$ or $2 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 2$; both of them give the same distribution, i.e. the robot ends at the same coop and each coop gets the same number of pellets in both distributions.

Sample Input #3

```
3 6 2
```

Sample Output #3

```
6
```

Explanation for the sample input/output #3

There are 6 different pellets distributions in this case: $\langle 1, \{3, 3, 0\}\rangle$, $\langle 1, \{2, 3, 1\}\rangle$, $\langle 3, \{2, 3, 1\}\rangle$, $\langle 1, \{1, 3, 2\}\rangle$, $\langle 3, \{1, 3, 2\}\rangle$, and $\langle 3, \{0, 3, 3\}\rangle$.

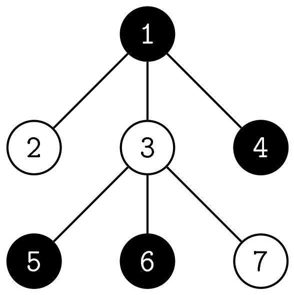
Problem K
White-Black Tree

The White-Black Tree is a game played by two players on a rooted tree of N nodes. The nodes are numbered from 1 to N and node 1 is the root. Each node in the tree has a color C_i of either 0 (representing black) or 1 (representing white) that can be changed according to the rule of the game.

Two opposing players play alternatingly. In their turn, the player chooses a **white** node in the tree; let the chosen node be x. First, the color of node x (C_x) is changed from white to black. Then, in the same turn, the player is allowed to change the color of zero or more nodes that are a descendant of node x from white to black or black to white. A node y is a descendant of a node x if and only if the parent of node y is either node x or a descendant of node x. The player who cannot make any move loses and the opposing player wins the game.

Your task in this problem is to determine who will win the game assuming both players play optimally; it means that if there exists a move that guarantees their win, then they will surely play that move.

For example, consider the following game with a rooted tree of $N = 7$ nodes and let the initial colors be $C_{1..7} = \{0, 1, 1, 0, 0, 0, 1\}$.



There are three white nodes (node 2, 3, and 7) in which the first player can choose for their first move. In this example, there is a strategy for the first player to win the game. One of the optimal plays for the first player is to choose node 3, change C_3 into 0 (black), and then change the color of node 6 and node 7 (both are node 3's descendant), i.e. C_6 becomes 1 (white) and C_7 becomes 0 (black). When it's the second player's turn, there are only two white nodes to choose from (node 2 and 6) and both of them do not have any descendants. No matter what the second player chooses for their turn, the first player will simply choose the remaining one white node so that the second player will not be able to make any move. Therefore, the second player loses and the first player wins this example game.

Input

Input begins with a line containing an integer N ($2 \leq N \leq 100\,000$) representing the number of nodes in the given tree. The second line contains $N - 1$ integers P_i ($1 \leq P_i < i$) for $i = 2..N$ representing the parent of node i. The third line contains N integers C_i ($C_i \in \{0, 1\}$) representing the initial color of node i. The color black is represented by the integer 0 while the color white is represented by the integer 1.

Output

Output a string “First” in a line if the first player will win the game assuming both players play the game optimally. Otherwise, output a string “Second” in a line.

Sample Input #1

```
7
1 1 1 3 3 3
0 1 1 0 0 0 1
```

Sample Output #1

```
First
```

Explanation for the sample input/output #1

This is the example from the problem description.

Sample Input #2

```
5
1 1 2 3
0 1 1 0 0
```

Sample Output #2

```
Second
```

Explanation for the sample input/output #2

The black root has two white children with the same subtree structure and colors. Whatever move the first player makes on one child, the second player simply mimics them on the other child.

Sample Input #3

```
4
1 1 1
1 1 0 1
```

Sample Output #3

```
First
```

Explanation for the sample input/output #3

The first player can make all nodes to be black in one move by choosing node 1.

Problem L

Happy Travelling

A holiday is coming and Sam plans to visit his parents. There are N cities numbered from 1 to N. Sam lives in a campus dormitory in city 1 while his parents live in city N.

Sam already did some research on the cities. He assigns an integer H_i to each city i representing the "happiness value" he will get if he visits city i, including city 1 and city N.

As for the transportation, he found out that for every city $1 \leq i < N$, there is a one-way bus that starts at city i and stops at each city from city $i+1$ until city $i+T_i$; in other words, the bus will stop at each of the next T_i cities. It is guaranteed that $i + T_i \leq N$. When Sam is at city $i < N$, he will board a bus that starts at city i and alight at city j where $j \in (i+1, i+T_i)$ while skipping all the stops between city i and city j (exclusive). Note that Sam cannot board a bus that does not start at city i if he is at city i.

Sam likes to be happy but he doesn't like being on a bus for a long time (he gets bored easily). Specifically, if he boards a bus that starts at city i and alights at city j (where $i < j$), then his happiness value will be decreased by the following.

$$\left\lfloor \frac{j-i}{K} \right\rfloor \times D$$

Sam is busy preparing souvenirs for his parents and figuring out what to do when he gets there. So he needs your help computing the total maximum happiness value that he can get by carefully planning his journey from city 1 to city N.

For example, let $N = 6$, $K = 2$, $D = 1$, $H_{1..6} = \{8, -7, -8, 9, 0, 2\}$, and $T_{1..5} = \{5, 3, 3, 2, 1\}$. The maximum total happines value that can be obtained from this example is 18 as shown in the following plan.

- Sam starts at city 1. He obtains a happiness value of $H_1 = 8$ at city 1.
- At city 1, Sam boards the bus that can stop at any city in $[2, 6]$ and alights at city 4. His happiness value is decreased by $\lfloor \frac{4-1}{2} \rfloor \times 1 = 1$ due to this bus trip. He obtains a happiness value of $H_4 = 9$ at city 4.
- At city 4, Sam boards the bus that can stop at any city in $[5, 6]$ and alights at city 5. His happiness value is decreased by $\lfloor \frac{5-4}{2} \rfloor \times 1 = 0$ due to this bus trip. He obtains a happiness value of $H_5 = 0$ at city 5.
- At city 5, Sam boards the bus that can stop at any city in $[6, 6]$ and alights at city 6. His happiness value is decreased by $\lfloor \frac{6-5}{2} \rfloor \times 1 = 0$ due to this bus trip. He obtains a happiness value of $H_6 = 2$ at city 6.

In this plan, Sam boards a bus 3 times with the total happiness value of $8 + (-1+9) + (0+0) + (0+2) = 18$. No other plan has a better total happiness value than this in this example.

Input

Input begins with a line containing three integers N K D ($2 \leq N \leq 100\,000$; $1 \leq K \leq N$; $0 \leq D \leq 10\,000$) representing the number of cities, and the parameter K and D as defined in the problem description, respectively. The second line contains N integers H_i ($-10\,000 \leq H_i \leq 10\,000$) representing the happiness value Sam will get if he visits city i. The third line contains $N-1$ integers T_i $(1 \le T_i; i+T_i \le N)$ for $1 \le i < N$ representing the number of next contiguous cities in which the bus that starts at city i will stop at.

Output

Output contains an integer in a line representing the maximum total happiness value that can be obtained.

Sample Input #1

```
6 2 1
8 -7 -8 9 0 2
5 3 3 2 1
```

Sample Output #1

```
18
```

Explanation for the sample input/output #1

This is the example from the problem description.

Sample Input #2

```
8 8 8
10 -5 -5 -5 -5 -5 -5 10
5 2 5 3 2 1 1
```

Sample Output #2

```
15
```

Explanation for the sample input/output #2

At city 1, Sam boards the bus that can stop at any city in $[2, 6]$ and alights at city 3. At city 3, Sam boards the bus that can stop at any city in $[4, 8]$ and alights at city 8. The total happiness value of this plan is $10 + (\lfloor \frac{3-1}{8} \rfloor \times 8 - 5) + (\lfloor \frac{8-3}{8} \rfloor \times 8 + 10) = 8 + (0 - 5) + (0 + 10) = 15$.

Sample Input #3

```
13 2 2
-5 -4 -4 -1 7 -6 -5 -4 -3 -2 -1 5 -7
3 10 9 8 7 6 5 4 3 2 1 1
```

Sample Output #3

```
-9
```

Problem M
Maxdifficent Group

Given an array of integers $A_{1..N}$ where $N \geq 2$. Each element in A should be assigned into a group while satisfying the following rules.

- Each element belongs to exactly one group.
- If A_i and A_j where $i < j$ belongs to the same group, then A_k where $i \leq k \leq j$ also belongs to the same group as A_i and A_j.
- There is at least one pair of elements that belong to a different group.

Let G_i denotes the group ID of element A_i. The cost of a group is equal to the sum of all elements in A that belong to that group.

$$\text{cost}(x) = \sum_{i \text{ s.t. } G_i = x} A_i$$

Two different group IDs, G_i and G_j (where $G_i \neq G_j$), are **adjacent** if and only if G_k is either G_i or G_j for every $i \leq k \leq j$. Finally, the $\text{diff}()$ value of two group IDs x and y is defined as the absolute difference between $\text{cost}(x)$ and $\text{cost}(y)$.

$$\text{diff}(x, y) = |\text{cost}(x) - \text{cost}(y)|$$

Your task in this problem is to find a group assignment such that the largest $\text{diff}()$ value between any pair of adjacent group IDs is maximized; you only need to output the largest $\text{diff}()$ value.

For example, let $A_{1..4} = \{100, -30, -20, 70\}$. There are 8 ways to assign each element in A into a group in this example; some of them are shown as follows.

- $G_{1..4} = \{1, 2, 3, 4\}$. There are 3 pairs of group IDs that are adjacent and their $\text{diff}()$ values are:
  - $\text{diff}(1, 2) = |\text{cost}(1) - \text{cost}(2)| = |(100) - (-30)| = 130$,
  - $\text{diff}(2, 3) = |\text{cost}(2) - \text{cost}(3)| = |(-30) - (-20)| = 10$, and
  - $\text{diff}(3, 4) = |\text{cost}(3) - \text{cost}(4)| = |(-20) - (70)| = 90$.

  The largest $\text{diff}()$ value in this group assignment is 130.
- $G_{1..4} = \{1, 2, 2, 3\}$. There are 2 pairs of group IDs that are adjacent and their $\text{diff}()$ values are:
  - $\text{diff}(1, 2) = |\text{cost}(1) - \text{cost}(2)| = |(100) - (-30 + (-20))| = 150$, and
  - $\text{diff}(2, 3) = |\text{cost}(2) - \text{cost}(3)| = |(-30 + (-20)) - (-20)| = 70$.

  The largest $\text{diff}()$ value in this group assignment is 150.

The other 6 group assignments are: $G_{1..4} = \{1, 1, 1, 2\}$, $G_{1..4} = \{1, 1, 2, 2\}$, $G_{1..4} = \{1, 2, 2, 2\}$, $G_{1..4} = \{1, 1, 2, 2\}$, $G_{1..4} = \{1, 1, 2, 3\}$, and $G_{1..4} = \{1, 2, 3, 3\}$. Among all possible group assignments in this example, the maximum largest $\text{diff}()$ that can be obtained is 150 from the group assignment $G_{1..4} = \{1, 2, 2, 3\}$.

Input

Input begins with a line containing an integer N ($2 \leq N \leq 100\,000$) representing the number of elements in array A. The next line contains N integers A_i ($-10^6 \leq A_i \leq 10^6$) representing the array A.

Output

Output contains an integer in a line representing the maximum possible largest $\text{diff}()$ that can be obtained from a group assignment.

Sample Input #1

```
4
100 -30 -20 50
```

Sample Output #1

```
150
```

Explanation for the sample input/output #1

This is the example from the problem statement.

Sample Input #2

```
5
12 7 4 32 9
```

Sample Output #2

```
46
```

Explanation for the sample input/output #2

The maximum possible largest $\text{diff}()$ of 45 can be obtained from the group assignment $G_{1..5} = \{1, 1, 1, 1, 2\}$. The $\text{diff}()$ value of the only adjacent group IDs is: $\text{diff}(1, 2) = 45$.

Sample Input #3

```
6
-5 10 -5 45 -20 15
```

Sample Output #3

```
70
```

Explanation for the sample input/output #3

The maximum possible largest $\text{diff}()$ of 70 can be obtained from the group assignment $G_{1..6} = \{1, 2, 2, 2, 3, 4\}$. The $\text{diff}()$ values of any two adjacent group IDs are: $\text{diff}(1, 2) = 55$, $\text{diff}(2, 3) = 70$, and $\text{diff}(3, 4) = 35$.