Programming Assignment (Due Date: 7 Mar 2019)

For this programming assignment, you will not have to submit any written answers. Instead, you will submit all of them on CodeCrunch at `https://codecrunch.comp.nus.edu.sg/`. You will have to submit in Java or C++11 and the portal will stop accepting submissions after 7 Mar 2019 2359h so please start your assignment **early**.

To get the full score for each part, you need to pass **ALL** the test cases correctly and within the time limit. The time limit for each test case is different for each part and will be stated below. As this is a programming assignment, not only is the time complexity of your algorithm important, but also the efficiency of your implementation.

If you are able to solve more than half the test cases correctly but not all, your algorithm is most likely correct but too slow, so think about how to make the implementation more efficient. We may award partial marks to you after the assignment if your answer is very close to the correct answer. You are strongly encouraged to create your own test cases for your own testing.

Templates will be provided for all the problems. These templates provide a starting point for your implementation and also provide fast implementations of the input routine. You are strongly recommended to use all the templates given to you. **Do not change the file name or the class name of the template, or else your code will be marked as incorrect**. However, you have to submit your own work. Any form of plagiarism is subject to disciplinary action.

Background

Mr. Panda has started a new delivery company called Amazin which helps customers deliver packages across very long distances. The company houses all its packages inside a warehouse where they are arranged neatly so that they can be located for delivery later.

As the company grows, the number of packages it needs to manage in its warehouse has increased significantly and the human workers simply cannot keep up with the workload. Thus, the company has resorted to using automated robots to help them perform tasks around the warehouse.

There are many different tasks to be done around the warehouse and efficiency is of utmost importance in order to ensure the packages get delivered on time. Thus, Mr. Panda has hired you, his best programmer, to design and program the most efficient algorithms into the robots to perform several tasks around the warehouse.

Task A (3%)

The warehouse contains N charging points labelled 1 to N. These charging points can charge the robots almost instantly. Using this charging points as vertices, the warehouse can be modelled as an undirected, weighted graph with E edges between the charging points with weight equal to the amount of time taken to travel between them. Some pairs of charging points have no edge because of obstacles such as walls between them.

The storage unit is at charging point 1 and delivery unit is at charging point N. You need to find a route for the robots to follow to transfer packages from storage to delivery. When following the route, the battery of the robot must be able to last at least as long as the highest weight edge along the path, since that is the maximum amount of time it has to travel without charging.

You know you want the route to take at most T time units in total and charging time is negligible, however you do not need to choose the shortest path. Instead, you want to save costs by minimising how long the battery needs to last, since longer-lasting batteries are more expensive. Design an algorithm and write a program to calculate this minimum value.

Input

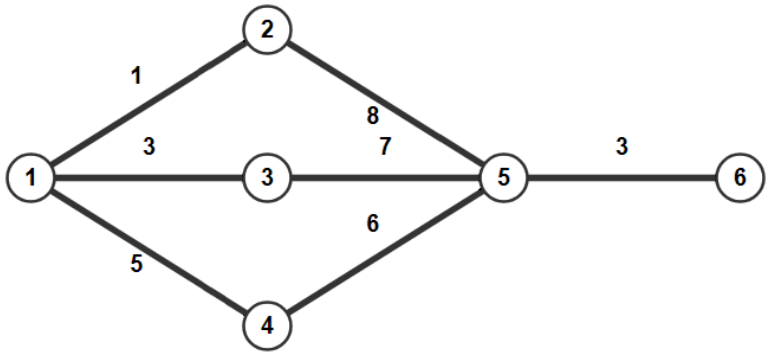
The first line of input will contain 3 integers, N, E and T as described above. The next E lines of input will each contain 3 integers, u, v and w representing an edge between vertices u and v with weight w. The edges in the input will be sorted in non-decreasing weight and the shortest path from 1 to N will be at most T time units.

Output

Your program simply needs to output one integer on a single line representing how long the battery needs to last in order to deliver the packages within T seconds.

Example Input

```
6 7 13
1 2 1
1 3 3
5 6 3
1 4 5
4 5 6
3 5 7
2 5 8
```



Example Output

7

Explanation

In this example, we have a graph with 5 vertices and 7 edges. We want to find the minimum battery needed to deliver packages within 13 time units. There are 3 possible routes. The route 1, 4, 5, 6 is not feasible since the total length if 14 which is more than 13. This leaves 1, 3, 5, 6 and 1, 2, 5, 6, both of which have total length at most 13. The maximum edge along the routes are 7 and 8 respectively, of which the lowest value is 7 which is the answer.

Limits

- $2 \leq N \leq 2^{16}, 1 \leq E \leq 2^{17}, 1 \leq u < v \leq N, 1 \leq w, T \leq 10^9$
- Your algorithm should have a time complexity of $O(E \log^2 E)$ and solve each test case within 3.0 seconds for Java and 1.0 seconds for C++.
- You should modify the template `Battery.java` and submit that file to CodeCrunch

Task B (4%)

In order to easily locate packages for delivery, the packages need to be sorted in some way. An overseas shipment of N packages have arrived that you need to distribute within your country.

To make it easier to locate the packages, you plan to use the robots to sort the packages in non-decreasing order of weight. However, as the robots are small, they are only able to swap 2 neighbouring packages. Furthermore, the robots are also light so if the 2 packages it swaps differ in weight by more than D grams, the robot will become imbalanced and may tip over.

Given this, you want to estimate the risk that the robot will tip over by calculating the minimum number of swaps needed where the boxes differ in weight by more than D grams.

Input

The first line of input will contain 2 integers, N and D as described above. The next line of input will contain N integers, representing the weight of the boxes in grams from left to right.

Output

Your program simply needs to output one integer on a single line representing the minimum number of swaps needed where the boxes differ in weight by more than D grams.

Example Input

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

Example Output

```
1
```

Explanation

There are 4 boxes with weights in the order (3,4,3,1) and we want to count swaps with weight difference more than 2. We swap the middle 2 boxes first with weight difference 1 giving (3,3,4,1). Then we swap the last box to the start, with weights (4,1), then (3,1) and (3,1) thus resulting in the order (1,3,3,4). Only the swap (4,1) has difference of more than 2 so the answer is 1.

Limits

There are 2 parts to this assignment

Part 1 (2%)

- $1 \leq N \leq 2^{16}, 1 \leq D,$ box weights $\leq 10^9$
- Your algorithm should have a time complexity of $O(N \log^2 N)$ and solve each test case within 0.5 seconds for Java and 0.33 seconds for C++.

Part 2 (2%)

- $1 \leq N \leq 2^{20}, 1 \leq D,$ box weights $\leq 10^9$
- Your algorithm should have a time complexity of $O(N \log N)$ and solve each test case within 0.5 seconds for Java and 0.33 seconds for C++.

Hints

You ask Mr. Panda for some hints on this task. He tells you that counting the number of swaps, regardless of their weight difference, is equivalent to counting the number of inversions. He asks you to look at the article `https://www.geeksforgeeks.org/counting-inversions/` and an efficient Java implementation of that algorithm called `CountingSwaps.java`. You are strongly encouraged to understand and modify the code to implement your algorithm to solve this task. Mr. Panda also tells you that to solve Part 1, you only need to change/add at most 10 lines of code from the implementation given. To solve Part 2, you only need to change/add at most 5 lines of code from the implementation given.

Challenge Task (1%, optional)

Instead of sorting the packages by weight, you realise that the packages also have a unique ID from 1 to N and sorting by ID instead will make locating the packages much easier. Thus, you plan to use the robots to sort these packages in increasing ID instead of by weight.

The robots have the same constraints as described in the previous task and you want to also want to calculate, in this case, the minimum number of swaps needed where the boxes differ in weight by more than D grams.

Input

The first line of input will contain 2 integers, N and D as described above. The next line of input will contain N integers, representing the weight of the boxes in grams from left to right. The next line of input will contain N integers, IDs of the boxes from left to right. The IDs will form a permutation of the integers from 1 to N.

Output

Your program simply needs to output one integer on a single line representing the minimum number of swaps needed where the boxes differ in weight by more than D grams.

Example Input

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

Example Output

```
3
```

Limits

- $1 \leq N \leq 2^{16}, 1 \leq D,$ box weights $\leq 10^9$
- Your algorithm should have a time complexity of $O(N \log^2 N)$ and solve each test case within 4.0 seconds for Java and 0.33 seconds for C++.
- You should modify the template `CountingSwaps.java` to and submit that file to CodeCrunch