6756 Increasing Shortest Path

We all love short and direct problems, it is easier to write, read and understand the problem statement. Here is one of these problems. "*Life is too short to make a story*", said *Ahmed Aly*.

You are given a weighted directed graph of N nodes (the nodes are numbered from 1 to N), where the weights of the edges are distinct and positive. For each graph, you are also given a list of queries to answer.

Each query will be represented by 3 integers A B C, which means you need to find the shortest path (the path with minimum sum of weights of its edges) which goes from node A to node B and uses at most C edges, such that the weights of the edges in that path are in increasing order along the path, which means the weight of each edge in that path should be greater than the weight of the edge before it (unless it is the first edge in the path).

Your task is to write a program which answers these queries.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T, the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, the first line of each test case contains 3 integers separated by a single space N M Q ($2 \leq N \leq 150$), ($0 \leq M \leq 3,000$) and ($1 \leq Q \leq 1,000$) representing the number of nodes, the number of edges and the number of queries, respectively. Followed by M lines, each line contains 3 integers separated by a single space X Y Z ($1 \leq X, Y \leq N$) ($1 \leq Z \leq 3,000$) which represent an edge going from the node X to the node Y with cost Z (X and Y will be different). Followed by Q lines, each line contains 3 integers separated by a single space A B C ($1 \leq A, B \leq N$) ($0 \leq C \leq M$) which represent a query as described above (A and B will be different).

Note that there might multiple edges between the same pair of nodes.

Output

For each test case, print a single line for each query which contains a single integer, the minimum sum of weights for a path between the given pair of nodes which satisfies the given constraints, or '`-1`' if there is no valid path between the given nodes which satisfies the given constraints. The output must not contain empty lines between the cases.

Sample Input

```
1
8 9 3
1 2 1
2 3 2
3 4 3
4 5 12
5 8 7
1 6 8
6 4 9
1 7 5
7 4 4
1 4 2
```

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

Sample Output

```
17
6
-1
```