This problem is a straightforward application of Dijkstra' algorithm implemented using heaps. You might find such implementations on the net, but may be better if you do it yourself. Be careful with memory usage.

You are given a directed graph with n vertices and m edges, with each edge having a non-negative integer cost. You also have k free passes with you. A free pass allows you to use any edge without incurring any cost, but it can be used only once. The problem is to find how to use the k free passes to minimize the cost of going from a node s to a node t. More formally, find a subset of at most k edges such that, after setting their cost to 0, the cost of the shortest path from s to t is minimized.

Input Format
The first line specifies the 5 integers n, m, k, s, t. The vertices are assumed to be numbered from 1 to n. Each of the next m lines contains 3 numbers i, j, c, indicating there is an edge from node i to node j of cost c.

Output
Output the minimum cost for going from s to t using at most k free passes.

Constraints
$1 \leq n, m \leq 10^6$
$0 \leq k \leq 20$
$0 \leq c \leq 10^9$
It can be assumed that there are no multiple edges or self loops in the graph, and there exists at least one path from s to t.

Sample Input

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

Sample Output
```
1
```