Task 3: Relay Marathon (`relaymarathon`)

You are Nilan, the marathon organiser of the country Berapur. There are N cities in the country, connected by M roads. The cities are indexed from $1, 2, \ldots, N$ each road is indexed from $1, 2, \ldots, M$. The i_{th} road directly connects city u_i with city v_i, and it takes w_i seconds to travel on this road. The roads are bidirectional in nature and it is also ensured that there are no self-loops or multi-edges in the road network. Out of the N cities, there is a list of K distinct cities $A_1, A_2, \ldots, A_k$ which are **special**.

As the marathon organiser, you are going to try to organise a **relay marathon**. A **relay marathon** is defined as follows: 2 groups of people are present at the starting cities start_1 and start_2 respectively. Firstly the people from start_1 travel to finish_1. Once the first group reaches finish_1, then immediately (i.e without any delay), the second group of people start travelling from start_2 to finish_2. Once the second group reaches finish_2, the **relay marathon** ends. Do note that a **relay marathon** is **valid** only if $\text{start}_1, \text{finish}_1, \text{start}_2$ and finish_2 all are **special** and distinct from one another.

Let $D(a, b)$ denote the shortest time to travel from city a to city b in the above road network. In case there is no path from city a to city b, then let us define $D(a, b) = \infty$. Then the total time taken in such a **valid relay race** is defined as $D(\text{start}_1, \text{finish}_1) + D(\text{start}_2, \text{finish}_2)$.

Given this, your job is to find the minimum possible value of $D(\text{start}_1, \text{finish}_1)+D(\text{start}_2, \text{finish}_2)$ amongst all **valid** tuples $(\text{start}_1, \text{finish}_1, \text{start}_2, \text{finish}_2)$.

Note: In the input network of roads, it will always be ensured that there are exists at least one **valid** tuple of four distinct cities, (a, b, c, d), such that a, b, c, d all are **special** and $D(a, b) + D(c, d) < \infty$ (i.e there exists a path from city a to city b and another path from city c to city d.)

Input

Your program must read from standard input.

The first line of the input contains 3 integers, N M K denoting the number of cities, the number of roads and the number of special cities respectively.

M lines will follow. Each consisting of 3 integers u_i v_i w_i, meaning that there is a road between city u_i and city v_i which takes w_i seconds to travel in either direction.

The next line contains K distinct integers $A_1, A_2, \ldots, A_k$ denoting the list of **special** cities.

Output

Your program must print to standard output.

The output should contain a single integer on a single line, the optimal minimum value of $D(\text{start}_1, \text{finish}_1) + D(\text{start}_2, \text{finish}_2)$ (in seconds).

Implementation Note

As the input lengths for subtasks 2, 3, and 4 may be very large, you are recommended to use C++ with fast input routines to solve this problem. The scientific committee does not have a solution written in Python that can fully solve this problem.

C++ and Java source files containing fast input/output templates have been provided in the attachment. You are strongly recommended to use these templates.

If you are implementing your solution in Java, please name your file `RelayMarathon.java` and place your main function inside `class RelayMarathon`.

Subtasks

The maximum execution time on each instance is 6.0s, and the maximum memory usage on each instance is 1GiB. For all testcases, the input will satisfy the following bounds:

- $4 \leq K \leq N \leq 10^5$
- $2 \leq M \leq \min(\frac{N(N-1)}{2}, 3 \times 10^6)$
- $1 \leq w_i \leq 1000$ for all $1 \leq i \leq M$
- $1 \leq u_i \neq v_i \leq N$ for all $1 \leq i \leq M$

Your program will be tested on input instances that satisfy the following restrictions:

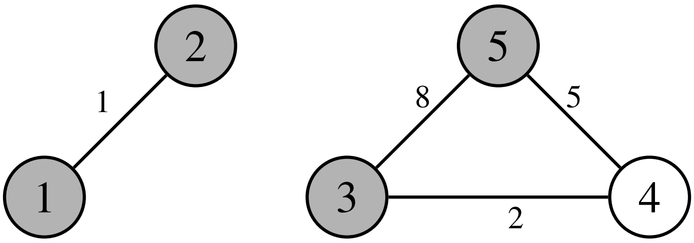
Subtask	Marks	Additional Constraints
1	5	$4 \leq K \leq N \leq 50$
2	12	$4 \leq K \leq N \leq 500$
3	25	- City 1 and City 2 both are **special** and directly connected with one another by an edge that takes 1 second to travel. - City 1 is NOT connected to any other city except City 2. - City 2 is NOT connected to any other city except City 1.
4	58	-

Sample Testcase 1

This testcase is valid for all subtasks.

Input	Output
5 4 4 1 2 1 3 4 2 4 5 5 5 3 8 3 1 5 2	8

Sample Testcase 1 Explanation



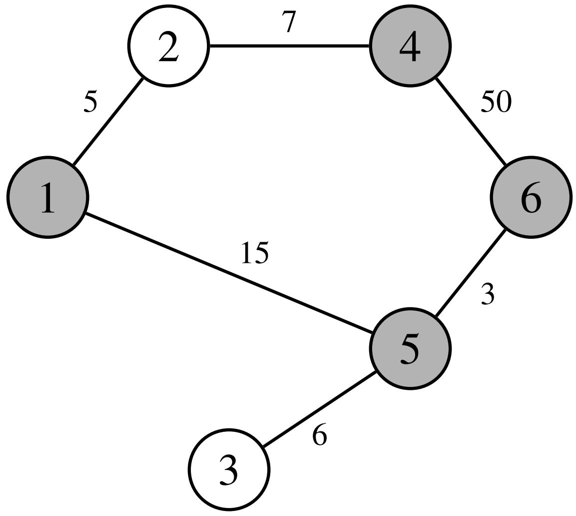
The cities in grey denote the **special** cities. We can observe that $D(1,2) = 1$ and $D(3,5) = \min(8, 2+5) = 7$. The optimal pairing here is $D(1,2) + D(3,5) = 1 + 7 = 8$ (i.e start$_1 = 1$, finish$_1 = 2$, start$_2 = 3$ and finish$_2 = 5$), any other kind of pairing configuration of $\{3, 1, 5, 2\}$ will not be better than this.

Sample Testcase 2

This testcase is valid for subtasks 1, 2, and 4.

Input	Output
6 6 4 1 2 5 2 4 7 4 6 50 6 5 3 1 5 15 3 5 6 1 5 4 6	15

Sample Testcase 2 Explanation



The cities in grey denote the **special** cities. We can observe that $D(1, 4) = 5 + 7 = 12$ and $D(5, 6) = 3$. The optimal pairing here is $D(1, 4) + D(5, 6) = 12 + 3 = 15$ (i.e start$_1 = 1$, finish$_1 = 4$, start$_2 = 5$ and finish$_2 = 6$), any other kind of pairing configuration of $\{1, 4, 5, 6\}$ will not be better than this.