CoconutsEasy

John is a coconut trader. He carries coconuts in a bag to all the shops. Initially bag contains **K** coconuts. If the bag carries more than **N-1** coconuts John starts to feel stress. If the number of coconuts becomes less than **N**, he starts feeling normal. Whenever he visits the i^{th} shop he either purchases S_i coconuts (adds S_i coconuts to the bag) or sells S_i coconuts to the shopkeeper (if he has less than S_i coconuts, he gives all coconuts and empties the bag), then moves to the next shop. There are **M** shops and he visits all the shops from 1 to **M** and he will not skip any shop.
If the number of coconuts in the bag becomes more than **N-1**, then in the next shop he will decrease the count to less than **N**. Given information about the shops, find the maximum number of times there will be change in John's mood (either stress -> normal or normal -> stress).

Constraints:
$1 \le N \le 3000$
$0 \le K < N$
$1 \le M \le 100$
$1 \le S_i \le 100$

Input Format:

First line contains three space separated integers *K,N, M*, denoting initial number of coconuts in the bag, minimum number of coconuts carrying which John starts feeling stressed and number of shops he visits.
Second line contains *M* space separated integers, i^{th} number denotes S_i of a shop.

Output Format:
Print a single integer denoting maximum number of instances John will change his mood

Sample Input #00:
1900 2100 5
100 200 100 1 1

Sample Output #00:

3

TeleportMedium

A graph with **n** nodes and **m** directed edges are given. If there is no edge between any two nodes, then there always exists a teleportation in between those nodes. It takes *1* unit time to reach from a to b, using teleportation. Teleportation can be used only when there is no edge between two nodes. Initially we start at node **1**. Find the minimum cost to reach node **n**.

Constraints:
$1 \le n \le 1000$
$1 \le m \le 10000$
Edges are non-negative.

Input Format:
First line contains two space separated integers *n* and *m* representing number of nodes and number of edges respectively.
Next *m* lines contain three integers *a*,*b* and *c* separated by space, representing an edge between *a* to *b* with edge cost *c*.

Output Format:
Print a single integer denoting minimum cost to reach from **1** to **n**.

Sample Input00:
2 1
1 2 3

Sample Output00:
3

Sample Input01:
3 1
1 2 3

Wall around the city

There are N watchtowers in the city located at (x[0],y[0]), (x[1],y[1]), ..., (x[N-1],y[N-1]), where x[i] denotes x-coordinate and y[i] denotes y-coordinate of i^{th} watchtower.

The king has asked his loyal masons to enclose the city by a stone wall, which will be following some constraints:

1. The stone wall should be in shape of square.
2. It should have vertices with integer coordinates.
3. The sides of the square should be parallel to the coordinate axes.
4. The masons have to ensure that K of the N watchtowers in the city lie strictly inside the walls, i.e., the watchtowers on the square's boundary are not counted as lying inside.

You have to complete a function int *minArea*(int x[], int y[], int k) which return the smallest possible area of a square that satisfies the above conditions.

If result doesn't fit in an integer, return result%1000000007 (result *mod* 1000000007).

Constraints
2 <= N <= 100
1 <= K <= N
-10^9 <= x_i <= 10^9

Sample Input #00:
x = [0,2]
y = [0,4]
k = 2

Sample Output #00:
36
Explanation: A square wall from (-1,-1) and (5,5) (Size = 6) will contain both the points