



Superman Celebrates Diwali



by vbegins

Problem

Submissions

Leaderboard

Discussions

Editorial

Superman has been invited to India to celebrate Diwali. Unfortunately, on his arrival he learns that he has been invited mainly to help rescue people from a fire accident that has happened in a posh residential locale of New Delhi, where rescue is proving to be especially difficult. As he reaches the place of the fire, before him there are N buildings, each of the same height H , which are on fire. Since it is Diwali, some floors of the buildings are empty as the occupants have gone elsewhere for celebrations. In his hurry to start the rescue Superman reaches the top of the building, but realizes that his jumping power is depleted and restricted due to change in his geographical setting. He soon understands the restrictions of his jumping power, and they are as follows:

- He can use the jumping power any number of times until he reaches the bottom floor, which means he can use the jumping power only until before he reaches the bottom (Ground floor), which means, once he reaches the bottom floor, he cannot move to the top floor again and try to save people. (In one single drop from the top to bottom)
- While switching buildings, he loses height I while jumping.

The second restriction is explained below with an example.

Assume $I = 2$. Now Superman is in the 2nd building 5th floor ($B = 2, F = 5$). If he wants to switch to the fifth building ($B = 5$), he will lose height ($I = 2$), which means he will be at floor 3 at building 5 ($B = 5, F = 3$). He can jump freely from the current floor to the floor below on the same building. That is, suppose if he is at ($B = 5, F = 4$), he can go to ($B = 5, F = 3$) without any restrictions. He cannot skip a floor while jumping in the same building. He can go to the floor below the current floor of the same building or use his jumping power, switch building, and lose height I .

Given the information about the occupied floors in each of the N buildings, help Superman to determine the maximum number of people he can save in one single drop from the top to the bottom floor with the given restrictions.

Input Format

Input starts with three values: the number of buildings N , the height of the buildings H , and the height Superman will lose when he switches buildings I .

These are followed by N lines. Each i^{th} line starts with a non negative integer u indicating how many people are in the i^{th} building. Each of the following u integers indicates that a person is at height u_i in the i^{th} building. Each of the following u integers are given and repetitions are allowed which means there can be more than one person in a floor.

i indicates building number and j indicates floor number. Building number will not be given; since N lines follow the first line, you can assume that the i^{th} line indicates the i^{th} building's specifications.

Constraints

$$1 \leq H, N \leq 1900$$

$$1 \leq I \leq 450$$

$$0 \leq u \leq 1900 \text{ (for each } i, \text{ which means the maximum number of people in a particular building will not exceed } 1900)$$

$$1 \leq u_{ij} \leq H$$

Output Format

Output the maximum number of people Superman can save.

Sample Input

```

4 15 2
5 1 1 1 4 10
8 9 5 7 7 3 9 8 8
5 9 5 6 4 3
0

```

Sample Output

12

Explanation

Input starts with $N = 4$, $H = 15$, $I = 2$.

N lines follow. Each line describes building i .

Each line begins with u , which denotes the number of persons in a particular building, followed by floor number, where each person resides. Floor number can repeat as any number of people can reside on a particular floor.

I've attached a figure here to explain the sample test case.

You can verify the first building's specifications with the figure.

$u = 5$ (Total number of persons in the first building), followed by 1 1 1 4 10 (Floor numbers).

1st floor = 3 persons.

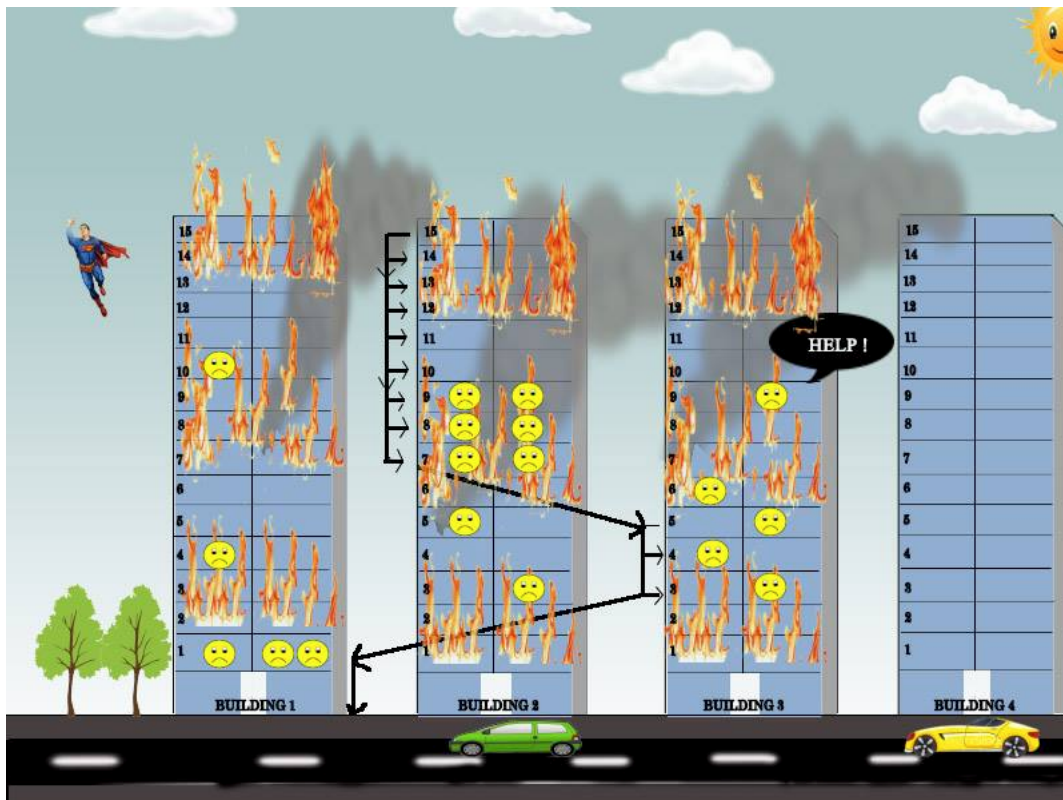
4th floor = 1 person.

10th floor = 1 person.

Similarly, the specifications for the other three buildings follow.

The connected line shows the path which Superman can use to save the maximum number of people. In this case, that number is 12.

You can also note in the figure that when he switches from Building 2 to Building 3, he loses height I ($I = 2$). Similarly, when he switches from Building 3 to Building 1, the same height loss happens as mentioned in the problem statement.




Submissions: 837

Max Score: 80

Difficulty: Hard

Rate This Challenge:

[More](#)Current Buffer (saved locally, editable)  

Java 8



```
1 import java.io.*;
2 import java.util.*;
3
4 public class Solution {
5
6     public static void main(String[] args) throws IOException{
7
8         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
9         String[] num = br.readLine().split("\\s");
10
11         int B = Integer.parseInt(num[0]);
12         int F = Integer.parseInt(num[1]);
13         int I = Integer.parseInt(num[2]);
14
15         int[][] peoples = new int[F][B];
16
17         for(int i = 0 ; i < B ; i++){
18
19             num = br.readLine().split("\\s");
20             int N = Integer.parseInt(num[0]);
21
22             for(int j = 1 ; j <= N ; j++){
23                 peoples[Integer.parseInt(num[j]) - 1][i]++;
24             }
25         }
26
27         for(int i = 0 ; i < B ; i++){
28             for(int j = 1 ; j < F ; j++){
29                 peoples[j][i] += peoples[j - 1][i];
30             }
31         }
32
33         int[] output = new int[F];
34
35         output[0] = Arrays.stream(peoples[0]).max().getAsInt();
36         HashSet<Integer> maxIndexSet[] = new HashSet[F];
37
38         for(int i = 0 ; i < F ; i++){
39             maxIndexSet[i] = new HashSet<Integer>();
40         }
41
42         storeAllIndex(output[0], maxIndexSet[0], peoples[0]);
43
44         for(int i = 1 ; i < F ; i++){
45
46             int tmp = Integer.MIN_VALUE;
47             int maxTillNow = Integer.MIN_VALUE;
48             int temp = Integer.MIN_VALUE;
49
50             for(int j = 0 ; j < B ; j++){
51
52                 if(i - I < 0){
53                     output[i] = Arrays.stream(peoples[i]).max().getAsInt();
54                     storeAllIndex(output[i], maxIndexSet[i], peoples[i]);
55                 }
56                 else{
57                     if(maxIndexSet[i - 1].contains(j)){
58                         temp = Math.max(Math.max(peoples[i][j] - peoples[i - 1][j] + output[i - 1], peoples[i][j] -
59                         peoples[i - 1][j] + output[i - 1]), Math.max(peoples[i][j], 0));
```

```
60     else{
61         temp = Math.max(Math.max(peoples[i][j] - peoples[i - 1][j] + output[i - 1],0),
Math.max(peoples[i][j],0));
62     }
63
64     if(maxTillNow < temp){
65         maxValueIndexSet[i] = new HashSet<Integer>();
66         maxValueIndexSet[i].add(j);
67         maxTillNow = temp;
68         output[i] = temp;
69     }
70     else if(temp == maxTillNow){
71         maxValueIndexSet[i].add(j);
72     }
73
74     tmp = temp;
75 }
76 }
77 }
78
79 System.out.println(output[F - 1]);
80
81 }
82
83 public static int bst(int[] arr, int key){
84
85     int left = 0;
86     int right = arr.length - 1;
87
88
89     while(left != right){
90
91         if(arr[left] > arr[right]){
92             right--;
93         }
94         else{
95             left++;
96         }
97     }
98     return arr[right];
99 }
100
101 public static void storeAllIndex(int val, HashSet<Integer> set, int[] arr ){
102
103     for(int i = 0 ; i < arr.length ; i++){
104         if(arr[i] == val){
105             set.add(i);
106         }
107     }
108 }
109 }
```

Line: 1 Col: 1

[Upload Code as File](#) ☐ Test against custom input

Run Code

Submit Code

Copyright © 2017 HackerRank. All Rights Reserved

Join us on IRC at [#hackerrank](#) on freenode for hugs or bugs.[Contest Calendar](#) | [Interview Prep](#) | [Blog](#) | [Scoring](#) | [Environment](#) | [FAQ](#) | [About Us](#) | [Support](#) | [Careers](#) | [Terms Of Service](#) | [Privacy Policy](#) | [Request a Feature](#)