Question 22 Max. Marks 100.00

Jump Over This

You are given two arrays of size N. Array 1 is val denoting values, and Array 2 is h denoting heights. At every position of the array, we can either pick the value val_i or increase the current height by h_i. You are allowed to perform **exactly 1** move at each position of array.

You are provided with M indexes I_i. For each of these indexes, you are given with min_height_i, which denotes the minimum height required to reach that index. Our height must be minimum of min_height_i before reaching the position i . If our height is less than min_height_i at index I_i, we can not jump to index I_i.

Note: All the indices I_i are distinct.

Your aim is to **maximise the sum** of val_i picked, at the end of the array, subject to given constraints. You need to **print -1**, incase there is no way of reaching the end of the array.

Input

The first line of input contains a number T that denotes the total number of test cases.
The first line for each test case consists of a number N.
The next line contains N space separated integers corresponding to val_i.
The next line contains N space separated integers corresponding to h_i.
The next line contains a single number M.
The next line contains M space separated integers corresponding to I_i.
The next line contains M space separated integers corresponding to min_height_i,.

Output

Output a single integer corresponding to maximum possible sum of val_i picked, at the end of the array, subject to given constraints.

Constraints

$1 \leq T \leq 10$
$1 \leq N \leq 1000$
$0 \leq M \leq N$
$0 \leq val_i \leq 10^9$
$0 \leq h_i \leq 10^9$
$0 \leq I_i \leq N-1$
$0 \leq min\_height_i \leq 10^3$

Sample Input

```
1
4
1 4 3 2
4 0 2 3
2
1 2
3 2
```

Sample Output

```
9
```

Explanation

Initially height=0, sum=0.

At index 0, we dont have any constraint on height. We move up by 4 at this step. Therefore, height=4 and sum=0.
At index 1, we should have height of atleast 3. As we are at height 4, this constraint is satisfied. At this stage, we pick the value. Therefore, height=4 and sum=4.
At index 2, we should have height of atleast 2. As we are at height 4, this constraint is satisfied. At this stage, we pick the value. Therefore, height=4 and sum=7.
At index 3, we dont have any constraint on height. At this stage, we pick the value. Therefore, height=4 and sum=9.

Therefore, the maximum possible sum of values picked is 9.

Note: Your code should be able to convert the sample input into the sample output. However, this is not enough to pass the challenge, because the code will be run on multiple test cases. Therefore, your code must solve this problem statement.

Time Limit: 2.0 sec(s) for each input file
Memory Limit: 256 MB
Source Limit: 1024 KB
Marking Scheme: Marks are awarded if any testcase passes
Allowed Languages: Java, Java 8, JavaScript(Rhino), JavaScript(Node.js), Python, Python 3, Ruby, Scala

New Submission | All Submissions

Java (openjdk 1.7.0_95) Save

```java
import java.io.*;
import java.util.*;


public class TestClass {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        PrintWriter wr = new PrintWriter(System.out);
        int T = Integer.parseInt(br.readLine().trim());
        for(int t_i=0; t_i<T; t_i++)
        {
            int N = Integer.parseInt(br.readLine().trim());
            String[] arr_val = br.readLine().split(" ");
            int[] val = new int[N];
            for(int i_val=0; i_val<arr_val.length; i_val++)
            {
                val[i_val] = Integer.parseInt(arr_val[i_val]);
            }
            String[] arr_h = br.readLine().split(" ");
            int[] h = new int[N];
            for(int i_h=0; i_h<arr_h.length; i_h++)
            {
                h[i_h] = Integer.parseInt(arr_h[i_h]);
```

```
        }
        int M = Integer.parseInt(br.readLine().trim());
        String[] arr_I = br.readLine().split(" ");
        int[] I = new int[M];
        for(int i_I=0; i_I<arr_I.length; i_I++)
        {
            I[i_I] = Integer.parseInt(arr_I[i_I]);
        }
        String[] arr_min_height = br.readLine().split(" ");
        int[] min_height = new int[M];
        for(int i_min_height=0; i_min_height<arr_min_height.length; i_min_height++)
        {
            min_height[i_min_height] = Integer.parseInt(arr_min_height[i_min_height]);
        }

        int out_ = solve(I, h, min_height, val,N,M);
        System.out.println(out_);
```

1:1

Press Ctrl/Command+Spacebar for autocomplete suggestions (accuracy dependent on connection stability).

Provide custom input

COMPILE & TEST SUBMIT