Google Kick Start 2020

Round B Solutions

Pb 1 : Bike Tour (5pts, 7pts)

Problem

Li has planned a bike tour through the mountains of Switzerland. His tour consists of N checkpoints, numbered from 1 to N in the order he will visit them. The i-th checkpoint has a height of Hi.

A checkpoint is a peak if:
It is not the 1st checkpoint or the N-th checkpoint, and
The height of the checkpoint is strictly greater than the checkpoint immediately before it and the checkpoint immediately after it.

Please help Li find out the number of peaks.

Input
The first line of the input gives the number of test cases, T. T test cases follow. Each test case begins with a line containing the integer N. The second line contains N integers. The i-th integer is Hi.

Output
For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is the number of peaks in Li's bike tour.

Limits
Time limit: 10 seconds per test set.
Memory limit: 1GB.
1 ≤ T ≤ 100.
1 ≤ Hi ≤ 100.

Test set 1
3 ≤ N ≤ 5.

Test set 2
3 ≤ N ≤ 100.

Sample

Input

Output

```
4
3
10 20 14
4
7 7 7 7
5
10 90 20 90 10
3
10 3 10
```

```
Case #1: 1
Case #2: 0
Case #3: 2
Case #4: 0
```

```
In sample case #1, the 2nd checkpoint is a peak.
In sample case #2, there are no peaks.
In sample case #3, the 2nd and 4th checkpoint are peaks.
In sample case #4, there are no peaks.
```

Solution:

```python
for t in range(int(input())):
    n = int(input())
    l = [int(x) for x in input().split()]
    peaks = 0
    for i in range(1,n):
        if (i+1)<n and (l[i-1]<l[i] and l[i]>l[i+1]):
            peaks+=1
    print("Case #{}: {}".format(t+1,peaks))
```

Pb 2 : Bus Routes (10pts, 13pts)

Problem

Bucket is planning to make a very long journey across the countryside by bus. Her journey consists of N bus routes, numbered from 1 to N in the order she must take them. The buses themselves are very fast, but do not run often. The i-th bus route only runs every Xi days.

More specifically, she can only take the i-th bus on day Xi, 2Xi, 3Xi and so on. Since the buses are very fast, she can take multiple buses on the same day.

Bucket must finish her journey by day D, but she would like to start the journey as late as possible. What is the latest day she could take the first bus, and still finish her journey by day D?

It is guaranteed that it is possible for Bucket to finish her journey by day D.

Input
The first line of the input gives the number of test cases, T. T test cases follow. Each test case begins with a line containing the two integers N and D. Then, another line follows containing N integers, the i-th one is Xi.

Output
For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is the latest day she could take the first bus, and still finish her journey by day D.

Limits
Time limit: 10 seconds per test set.
Memory limit: 1GB.
1 ≤ T ≤ 100.
1 ≤ Xi ≤ D.
1 ≤ N ≤ 1000.
It is guaranteed that it is possible for Bucket to finish her journey by day D.

Test set 1
1 ≤ D ≤ 100.

Test set 2
1 ≤ D ≤ 1012.

Sample

Input

Output

```
3
3 10
3 7 2
4 100
11 10 5 50
1 1
1
```

```
Case #1: 6
Case #2: 99
Case #3: 1
```

```
In Sample Case #1, there are N = 3 bus routes and Bucket must arrive by
day D = 10. She could:
Take the 1st bus on day 6 (X1 = 3),
Take the 2nd bus on day 7 (X2 = 7) and
Take the 3rd bus on day 8 (X3 = 2).

In Sample Case #2, there are N = 4 bus routes and Bucket must arrive by
day D = 100. She could:
Take the 1st bus on day 99 (X1 = 11),
Take the 2nd bus on day 100 (X2 = 10),
Take the 3rd bus on day 100 (X3 = 5) and
Take the 4th bus on day 100 (X4 = 50),

In Sample Case #3, there is N = 1 bus route and Bucket must arrive by day
D = 1. She could:
Take the 1st bus on day 1 (X1 = 1).
```

Solution:

```python
for t in range(int(input())):
    n = int(input())
    l = [int(x) for x in input().split()]
    peaks = 0
    for i in range(1,n):
        if (i+1)<n and (l[i-1]<l[i] and l[i]>l[i+1]):
            peaks+=1
    print("Case #{}: {}".format(t+1,peaks))
```

Pb 3 : Robot Path Decoding (11pts, 16pts)

Problem

Your country's space agency has just landed a rover on a new planet, which can be thought of as a grid of squares containing 109 columns (numbered starting from 1 from west to east) and 109 rows (numbered starting from 1 from north to south). Let (w, h) denote the square in the w-th column and the h-th row. The rover begins on the square (1, 1).

The rover can be maneuvered around on the surface of the planet by sending it a program, which is a string of characters representing movements in the four cardinal directions. The robot executes each character of the string in order:

N: Move one unit north.
S: Move one unit south.
E: Move one unit east.
W: Move one unit west.

There is also a special instruction X(Y), where X is a number between 2 and 9 inclusive and Y is a non-empty subprogram. This denotes that the robot should repeat the subprogram Y a total of X times. For example:
2(NWE) is equivalent to NWENWE.
3(S2(E)) is equivalent to SEESEESEE.
EEEE4(N)2(SS) is equivalent to EEEENNNNSSSS.

Since the planet is a torus, the first and last columns are adjacent, so moving east from column 109 will move the rover to column 1 and moving south from row 109 will move the rover to row 1. Similarly, moving west from column 1 will move the rover to column 109 and moving north from row 1 will move the rover to row 109. Given a program that the robot will execute, determine the final position of the robot after it has finished all its movements.

Input
The first line of the input gives the number of test cases, T. T lines follow. Each line contains a single string: the program sent to the rover.

Output
For each test case, output one line containing Case #x: w h, where x is the test case number (starting from 1) and w h is the final square (w, h) the rover finishes in.

Limits
Time limit: 10 seconds per test set.
Memory limit: 1GB.
1 ≤ T ≤ 100.
The string represents a valid program.
The length of each program is between 1 and 2000 characters inclusive.

Test set 1
The total number of moves the robot will make in a single test case is at most 104.

Test set 2
No additional constraints.

Sample

```
Input

Output

4
SSSEEE
N
N3(S)N2(E)N
2(3(NW)2(W2(EE)W))


Case #1: 4 4
Case #2: 1 1000000000
Case #3: 3 1
Case #4: 3 999999995
```

In Sample Case #1, the rover moves three units south, then three units east.

In Sample Case #2, the rover moves one unit north. Since the planet is a torus, this moves it into row 109.

In Sample Case #3, the program given to the rover is equivalent to NSSSNEEN.

In Sample Case #4, the program given to the rover is equivalent to NWNWNWWEEEEWWEEEEWNWNWNWWEEEEWWEEEEW.

Solution:

```python
T = int(input())
nums = set([str(_) for _ in range(10)])
N = 10**9

for t in range(1, T+1):
    program = input()

    cur = [0, 0]

    stack = []

    for char in program:
        if char == 'N':
            cur[0] -= 1
        elif char == 'S':
            cur[0] += 1
        elif char == 'W':
            cur[1] -= 1
        elif char == 'E':
            cur[1] += 1
        elif char in nums:
            stack.append((cur[0], cur[1], int(char)))

        elif char == '(':
            cur = [0, 0]

        elif char == ')':
            pop = stack.pop()
            cur = [pop[i] + pop[2]*cur[i] for i in [0, 1]]

    final_row = (1+cur[0])%N
    if final_row == 0:
        final_row = N

    final_column = (1+cur[1])%N
    if final_column == 0:
        final_column = N

    print("Case #%d: %d %d" % (t, final_column, final_row))
```

Pb 4 : Wandering Robot (14pts, 24pts)

Problem

Jemma is competing in a robotics competition. The challenge for today is to build a robot that can navigate around a hole in the arena.

The arena is a grid of squares containing W columns (numbered 1 to W from left to right) and H rows (numbered 1 to H from top to bottom). The square in the x-th column and y-th row is denoted (x, y). The robot begins in the top left square (1,1) and must navigate to the bottom right square (W, H).

A rectangular subgrid of squares has been cut out of the grid. More specifically, all the squares that are in the rectangle with top-left square (L, U) and bottom-right square (R, D) have been removed.

Jemma did not have much time to program her robot, so it follows a very simple algorithm:
If the robot is in the rightmost column, it will always move to the square directly below it. Otherwise,
If the robot is in the bottommost row, it will always move to the square directly right of it. Otherwise,
The robot will randomly choose to either move to the square directly to the right, or to the square directly below it with equal probability.

Jemma passes the challenge if her robot avoids falling into the hole and makes it to the square (W, H). What is the probability she passes the challenge?

Input
The first line of the input gives the number of test cases, T. T test cases follow. Each test case consists of a single line containing W, H, L, U, R and D.

Output
For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is a real number between 0 and 1 inclusive, the probability that Jemma passes the challenge.

y will be considered correct if it is within an absolute or relative error of 10-5 of the correct answer. See the FAQ for an explanation of what that means, and what formats of real numbers we accept.

Limits
Time limit: 15 seconds per test set.
Memory limit: 1GB.
1 ≤ T ≤ 100.
1 ≤ U ≤ D ≤ H.
1 ≤ L ≤ R ≤ W.
Neither the top-left nor bottom-right squares will be missing.

Test set 1
1 ≤ W ≤ 300.
1 ≤ H ≤ 300.

Test set 2
1 ≤ W ≤ 105.
1 ≤ H ≤ 105.

```
Sample

Input

Output

4
3 3 2 2 2 2
5 3 1 2 4 2
1 10 1 3 1 5
6 4 1 3 3 4


Case #1: 0.5
Case #2: 0.0625
Case #3: 0.0
Case #4: 0.3125
```

Solution:

```python
import scipy.stats

T = int(input())

for t in range(1, T+1):
    W, H, L, U, R, D = [int(_) for _ in input().split()]

    ans = 0.
    if D != H:
        #D downs before L-1 rights
        #D downs within D+L-2 tosses
        #>= D downs in D+L-2 tosses
        #<= L-2 rights in D+L-2 tosses

        ans += scipy.stats.binom.cdf(L-2, D+L-2, 0.5)

    if R != W:
        #R rights before U-1 downs
        #R rights within R+U-2 tosses
        #>= R rights in R+U-2 tosses
        #<= U-2 downs in R+U-2 tosses

        ans += scipy.stats.binom.cdf(U-2, R+U-2, 0.5)

    print("Case #%d: %f" % (t, ans))
```