Terminator - Hackerearth

Thursday, March 26, 2020 10:35 PM

The Problem:

War is raging between Humans and Cyberdyne Systems. Our intel has shown that Cyberdyne has prepared a large army of terminators to exterminate humans from the face of Earth. We need to prepare for battle!

We sent some spies to the Terminator camp to gather more information about their tactics. Our spies
observed the following:

1. There are a total of *n* terminators in the camp. They are numbered from *1* to *n*. All terminators have a unique strength level.
2. They are divided into groups. Each group has a certain number of terminators in it. The size of each group may be different.
3. Every day the terminators stand in a queue for inspection. The spies observed that the terminators stood in such a way that a terminator never stands ahead of a stronger terminator of his group. A terminator may stand ahead of a stronger terminator of a different group.

The spies observed the camp for two days. They noted down the position of each terminator in the queue for each day. The supreme Commander of the Human Army has appointed you. Your job is to tell the maximum possible number of terminators in one group.

Constraints:
1≤t≤100
1≤n≤1000
No two terminators occupy the same position in the queue.

Input:
The first line of the input contains the number of test cases, *t*. The first line of each test case has the n, the number of terminators. *N* lines follow. The ith line has two integers, p1 and p2. P1 is the position of the ith terminator in the inspection line as observed on the first day. P2 is the position of the ith terminator in the inspection line as observed on the second day.

Output:
For each test case, output a single line containing the maximum number of terminators that may belong to the same group.

SAMPLE INPUT

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

1 4
2 2
3 3
4 1
SAMPLE OUTPUT

3
2

Explanation
For the first test case, the largest possible group is 1,2,3. For the second test case, the largest possible group is 2,3.

The Code:

```
/* Simple. Just find the Largest Common Subsequence between the two days.
* Afternote: I do suspect that it should've been something like Longest
Increasing
     subsequence instead, because of all the strength conditions. But the test
     cases passed, so maybe I'm overthinking this.
*/
#include <stdio.h>
#define max(a,b) (a > b ? a : b)
int main(){
    int caseCount, N, curr, prev, input, i, j;
    scanf("%d", &caseCount);

    int** lcs = (int**)malloc(2 * sizeof(int));
    lcs[0] = (int*)malloc(1001 * sizeof(int));
    lcs[1] = (int*)malloc(1001 * sizeof(int));
    int* day1 = (int*)malloc(1001 * sizeof(int));
    int* day2 = (int*)malloc(1001 * sizeof(int));
    while(caseCount > 0) {
        scanf("%d", &N);
        lcs[0][0] = lcs[1][0] = 0;
        for(i=1; i<=N; i++) {
            scanf("%d", &input);
            day1[input] = i;
            scanf("%d", &input);
            day2[input] = i;
            lcs[0][i] = lcs[1][i] = 0;
        }

        for(i=1; i<=N; i++) {
            curr = (i % 2);
            prev = abs(curr - 1);
            for(j=1; j<=N; j++) {
                if(day2[i] == day1[j]) {
                    lcs[curr][j] = 1 + lcs[prev][j-1];
                } else {
                    lcs[curr][j] = max(lcs[prev][j], lcs[curr][j-1]);
                }
            }
        }
        printf("%d\n", lcs[curr][N]);
        caseCount--;
    }
}
```

The Stats:

Score
30.0

Time (sec)
0.60777

Memory (KiB)
316

Language
C