#2457: 공주님의 정원

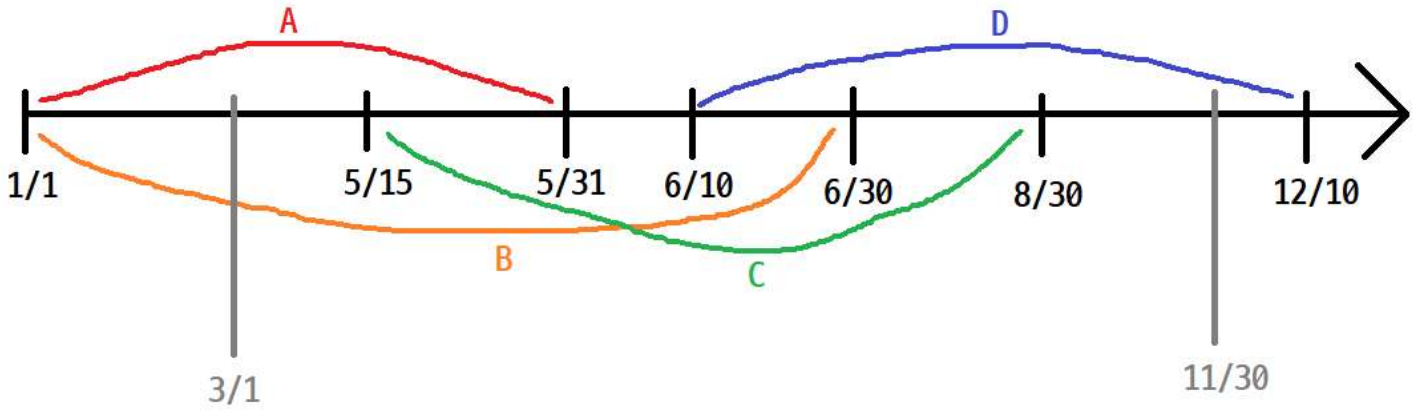
sianke1991

꽃 목록, 각 꽃의 피는 날짜와 지는 날짜를 입력 받아서, 3월 1일부터 11월 30일 까지 최소 하나의 꽃이 피어있도록 꽃을 선택하여 정원에 심는다. 이 때 심을 수 있는 꽃의 최소 개수를 구한다.

문제의 입력 예시로 주어진 값은 다음과 같다.

꽃 종류	피는 날짜	지는 날짜
A	1/1	5/31
B	1/1	6/30
C	5/15	8/31
D	6/10	12/10

이 꽃들의 피는 날짜와 지는 날짜의 전후 관계를 파악하기 쉽게 그림을 그려보면 다음과 같다.



각 꽃들의 피는 날짜와 지는 날짜
해당 꽃이 지는 날짜에는 꽃이 피어 있다고 간주하지 않는다.
예를 들어 12월 10일에는 모든 꽃이 져서 피지 않은 것으로 간주한다.

이 값으로 3월 1일부터 11월 30일까지 모든 날짜에 대해 최소 하나의 꽃이 필 수 있도록 꽃을 선택해 보자. 일단 3월 1일에는 꽃이 피어있어야 하므로 3월 1일 혹은 그 이전에 피기 시작하는 꽃을 선택해야 한다. 그 후보는 A(1/1 – 5/31), B(1/1 – 6/30)이 있는데 이 중 더 늦게 까지 피는 B를 선택해야 꽃이 피어있는 기간을 오래 유지할 수 있다. 꽃 B는 6월 30일에 지므로 이 날짜 혹은 이전에 피는 꽃을 선택해야 한다. 그 후보는 A, B, C(5/15 – 8/30), D(6/10 – 12/10)이 있는데 이 중 가장 늦게 까지 피는 D를 선택하는 것이 옳은 선택이다. (또한 이 시점에서 꽃 A와 B를 선택하면 안되는데 이 꽃들은 6월 30일을 넘기지 못하고 지기 때문이다.) 꽃 D가 지는 날짜는 12월 10일이고 이것은 목표치 11월 30일을 지난 것이므로 더 이상 꽃을 선택할 필요가 없다. 공주님의 정원에는 꽃 B와 D를 선택하여 심으면 된다.

이 방법을 구현하기 위해 여러 가지 방법 중 선택할 수 있다.
그 중 첫 번째는 꽃을 지는 시기에 대해 정렬하는 것이다. 꽃을 늦게까지 유지되는 정도에 대해 정렬하면, 특정 시기에 피어있는 꽃을 선택할 때 가장 오래 남는 꽃을 바로 선택할 수 있다.

```
#include <bits/stdc++.h>
using namespace std;

struct Flower {
    int beginDate; //피는 날짜와 지는 날짜는 (100*달+날)의 형식으로 저장한다.
    int endDate;
}
```

```
/*
 * 꽃 배열을 정렬할 때 적용할 기준.
 * 꽃이 늦게 질 수록, (같은 날짜에 진다면) 일찍 필 수록 먼저 배치되는 정렬을 사용한다.
 */
bool compareFlower(Flower flower0, Flower flower1) {
    if (flower0.endDate != flower1.endDate) {
        return flower0.endDate>flower1.endDate;
    return flower0.beginDate < flower1.beginDate;
}

Flower fromInts(int beginMonth, int beginDate, int endMonth, int endDate) {
    Flower result;
    result.beginDate = beginMonth*100+beginDate;
    result.endDate = endMonth*100+endDate;
    return result;
}

… 중략 …

sort(flowers, flowers+numFlowers, compareFlower);
int numSelectedFlowers = 0; //(공주님의 정원에 심기로) 선택된 꽃의 개수
int now = 301;
int targetDate = 1130;
//recentFlowerIndex: 바로 직전에 선택된 꽃의 인덱스 값
//바로 직전에 선택한 꽃이 져서 새로운 꽃을 선택해야 하는 경우, 바로 직전에 선택한 꽃보다 먼저 지는 꽃을 선택하면 안되므로, flowerIndex는 recentFlowerIndex 미만의 범위에서 loop를 수행하게 된다.
int flowerIndex, recentFlowerIndex = numFlowers;
while (now<=targetDate) {
    for (flowerIndex = 0; flowerIndex<recentFlowerIndex; flowerIndex++) {
        if (now>=flowers[flowerIndex].beginDate) break; //현재 기준으로 피어있는 꽃을 발견한 경우 loop를 종료한다.
        /*
            꽃을 지는 날짜에 대해 내림차순으로 정렬했으므로 위에서 찾은 꽃은
            해당 날짜에 피어있는 꽃 중 가장 오래 가는 꽃이다.
        */
    } //flowerIndex loop
    if (flowerIndex==recentFlowerIndex) {
        /*
           해당 날짜에 피어있는 꽃 중 가장 오래가는 꽃을 선택했는데, 그것이 바로 직전에 선택한 꽃인 경우,
           (최소 하나의) 꽃이 피어있는 기간을 targetDate까지 연장하지 못함을 의미한다.
        */
        cout << 0;
        return 0;
    }
    numSelectedFlowers++;
    recentFlowerIndex = flowerIndex;
    now = flowers[flowerIndex].endDate;
} //while loop
```

또 다른 방법은 조사하는 시각 기준 피어있는 꽃에 대해 가장 늦게 까지 피어있는 꽃을 조사하는 것이다. (현재 피어있는 꽃 중에서 지는 시기의 최대치를 구한다.)

```
...생략...
int now = 301; //현재 날짜
int ans = 0; //선택한 꽃의 개수
while (now<1201) {
    int nxt_now = now; //이번에 추가할 꽃으로 인해 변경될 날짜 (이번에 추가할 꽃의 지는 날짜)
    for (int i=0; i<n; i++) {
        /*
            현재 날짜에 피어있고, 현재 선택한 꽃 보다 더 늦게 지는 꽃일 경우 해당 꽃의 지는 날짜로 nxt_now를 갱신한다.
            이 loop는 현재 날짜에 피어있는 꽃 중에서 지는 날짜의 최대치를 구하는 것이다.
        */
        if (flower[i].beginDate<=now && flower[i].endDate>nxt_now)
            nxt_now = flower[i].endDate;
    } //i loop
    if (nxt_now ==now) {
        /*
        시간 t에서 전진이 불가능 (꽃을 더 신택해도 꽃이 피어있는 날짜를 연장하지 못하는 경우)
        */
        cout << 0;
        return 0;
    }
    ans++;
    now = nxt_now;
} //while loop
```