HomeWork_2

과목 : :컴퓨터알고리즘
분반 : 061
학과 : 정보컴퓨터공학부
학번 : 201824520
이름 : 안주현

## 1. 방법

info class를 정의하고 입력된 대회들의 정보를 저장한다.

```
class info  //class for information about input
{
public:
          int starting_day;
          int ending_day;
          int reward;
          long long int max_to_ending; //optimal reward to ending_day
          info()
          {
                    starting_day = 0;
                    ending_day = 0;
                    reward = 0;
                    max_to_ending = 0;
          }
          bool operator <(info &info)
          {
                    return this->ending_day < info.ending_day;
          }
};
```

Starting_day : 대회가 시작하는 날짜

Ending_day : 대회가 끝나는 날짜

Reward: 대회의 상금

Max_to_ending : 끝나는 날까지의 최대 상금(optimal reward)

Bool operator<() : 끝나는 날(ending_day)을 기준으로 정렬하기위한 비교 연산자 오버로딩

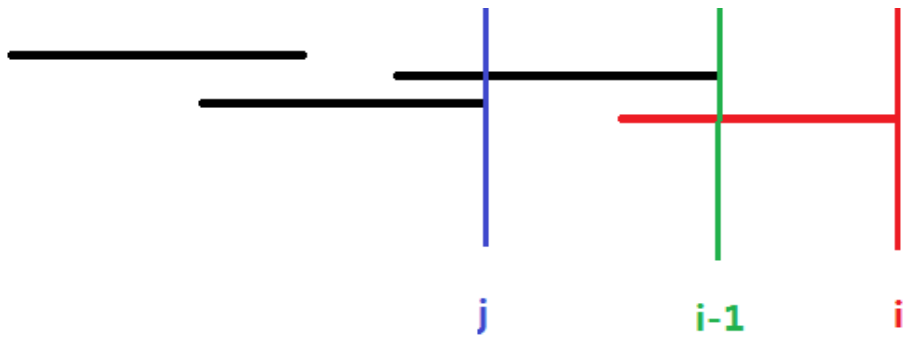
### 1. 먼저 Info 클래스배열을 ending_day를 기준으로 오름차순으로 정렬한다.

### 2. 차례로 max_to_ending 계산한다

#### 2-1 날짜가 겹치는 경우

i번째 max_to_ending를 계산한다고 가정할 때, i번째 max_to_ending은

**max( (j번째 max_to_ending + i번째 reward), (i-1번째 max_to_ending) )**로 구할 수 있다.



### 예외(1번째 max_to_ending)

단 i가 1일 때는 **max( (0번째 max_to_ending), (1번째 max_to_ending) )** 으로 구한다.


### 2-2 날짜가 겹치지 않는 경우

i번째 max_to_ending를 계산한다고 가정할 때, i번째 max_to_ending은

**i-1번째 max_to_ending + i번째 max_to_ending** 으로 구할 수 있다.

## 2. 코드

```
//201824520_안주현
#include <iostream>
#include <algorithm>
#include <time.h>

using namespace std;

class info  //class for information about input(대회 정보 저장을 위한 class)
{
public:
		int starting_day;
		int ending_day;
		int reward;
		long long int max_to_ending; //optimal reward to ending_day(끝나는 날까지의 최대 reward)

		info()
		{
			starting_day = 0;
			ending_day = 0;
			reward = 0;
			max_to_ending = 0;
		}
		bool operator <(info &info)
		{
			return this->ending_day < info.ending_day;
		}
};
int main(void)
{
		int line=0;
		info *inf = new info[200000];

		cin >> line;

		for(int i=0; i<line; i++) { //save data(정보 저장)
			cin >> inf[i].starting_day
				>> inf[i].ending_day
				>> inf[i].reward;
		}

		clock_t start = clock(); //start clock(실행 시간 측정 시작)

		sort(inf, inf+line); //Sort in ascending order based on ending_day
				//끝나는 날을 기준으로 오름차순 정렬
```

```
        int index=0;
        long long int max_reward=0;

        max_reward = inf[0].reward;
        inf[0].max_to_ending = inf[0].reward; //max reward to 0th ending day is 0th reward
                                              //0번째 ending_day까지의 최대 reward는 자신의 reward
        info *current, *previous; //pointer to current and previous info
                                  //현재와 이전의 info클래스에 대한 포인터
        for(int i=1; i<line; i++) { //start with index 1(인덱스 1부터 시작)
                index = 0; //index for info to get previous optimal reward
                           //이전의 겹치지 않는 대회에 대한 인덱스
                current = &inf[i]; //point current info(현재 info클래스에 대한 포인터)
                previous = &inf[i-1]; //point previous info(이전 info클래스에 대한 포인터)

                //If the dates of the competitions overlap(대회 날짜가 겹치는 경우, 2-1)
                if(current->starting_day <= previous->ending_day) {
                        //겹치지 않는 이전의 대회에 대한 인덱스의 최댓값 찾기
                        for(int j=i-1; j>=0; j--) {
                                if(inf[j].ending_day < current->starting_day) {
                                        index = j; break;
                                }
                        }

                        if(index == 0 && i == 1) { //exception for 1th(i가 1인 경우, 2-1 예외)
                                max_reward = max( inf[0].max_to_ending,
                                                  (long long int)current->reward);
                                current->max_to_ending = max_reward;
                        }
                        else{ //not an exception(예외가 아닌 경우, 2-1)
                                max_reward =
                                        max( inf[index].max_to_ending + current->reward ,
                                           previous->max_to_ending);
                                current->max_to_ending = max_reward;
                        }
                }
                 //If the dates of the competition do not overlap(대회 날짜가 겹치지 않는 경우, 2-2)
                else {
                        current->max_to_ending = previous->max_to_ending + current->reward;
                }
        }

        clock_t end = clock(); //stop clock(실행 시간 측정 종료)

        cout << endl << inf[line-1].max_to_ending; //print optimal reward(최대 reward 출력)

        //print time to get the optimal reward(최대 reward를 구하는데 걸린 시간 출력)
        cout << endl << "실행 시간 : " << (double)(end - start)/CLOCKS_PER_SEC;
        cout << "초(" << end - start << "ms)";

        delete inf;

        return 0;
}
```

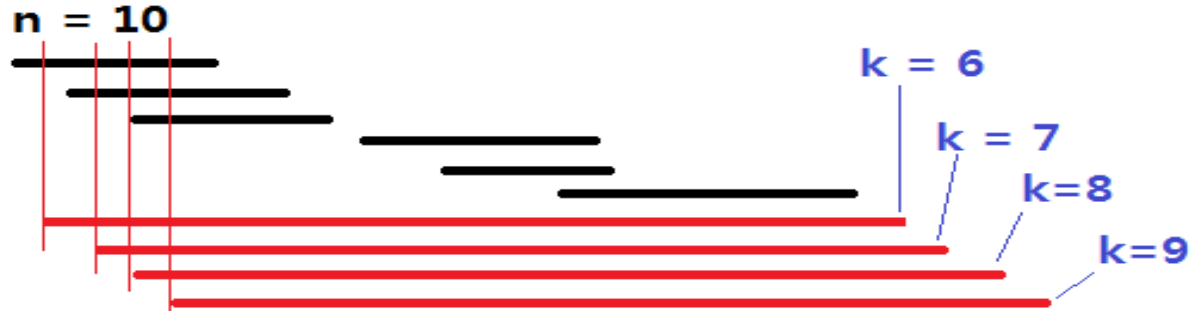
## 3. 시간복잡도

대회를 위한 정보를 저장하는 배열 info를 정렬하는데 걸리는 시간은 O(nlogn)이다.
각 info의 max_to_exding을 계산하는데 필요한 시간은 첫번째 for문을 수행하는 횟수를 n(데이터 개수)이라 하고, 두번째 for문을 수행하는 횟수를 k라고 가정하자.

- 일반적인 경우

2번째 for문의 k가 n보다 현저히 작기 때문에 max_to_exding을 계산하는데 필요한 시간은 O(n)이 되고, 전체 시간 복잡도는 O(nlogn + n) = O(nlogn)이다.

- 최악의 경우




위의 그림과 같이 2 번째 for 문의 k 가 n 에 가까워지는 경우에 max_to_exding 을 계산하는데 필요한 시간은 $O(n^2)$에 가까워 지고, 전체 시간복잡도는 $O(nlogn + n^2) = O(n^2)$에 가까워질 수 있다.

## 4. 출력

- Test_input 1

```
C:₩Users₩LG₩Desktop₩201824520_안주현.exe
10
14 14 98
76 76 58
94 94 57
92 92 45
82 82 14
86 86 41
87 87 72
14 14 26
27 27 85
48 48 52

522
실행 시간 : 0초(0ms)
```

- Test_input 2

```
C:₩Users₩LG₩Desktop₩201824520_안주현.exe
10
86 94 24
88 94 35
50 86 23
15 29 53
66 72 82
61 84 93
16 40 22
92 99 70
41 93 24
78 86 19

224
실행 시간 : 0초(0ms)
```

- Test_input 3

```
C:₩Users₩LG₩Desktop₩201824520_안주현
76385310 76385316 859011456
676119827 676119827 753970315
543769234 543769239 557664384
422011540 422011540 892396176
660671510 660671518 35413507
997821040 997821046 970555606
517343401 517343407 162516366
709705403 709705406 21448030
247956608 247956612 333183511
392795199 392795199 130420019
934533809 934533810 198802919
9664521 9664523 699919538
226146851 226146856 184708842
276156577 276156584 32063283
671468306 671468309 809254896
183410029 183410037 278596497
273217073 273217082 726707088
249454454 249454460 569033766
954699140 954699140 215770359

99858041144116
실행 시간 : 0.087초(87ms)
```