## 과제 #11 계단함수 - 201921438 조현태

### 1) 소스코드

```
#include <stdio.h>
#include <iostream>
#include <string>
#include <string.h>
#include <algorithm>
#include <vector>
#include <math.h>
#include <stdlib.h>
#include <map>
#include <stack>
#include <queue>
using namespace std;

// k1 = f(x), k2 = g(x)
// 1 <= k1, k2 <= 100000
int k1, k2;
// -20억 <= a <= b <= 20억
int a, b;
int answer = 0;

// mapset (x, f(x))
map <int, vector<int>> mapset;
// x좌표 벡터
vector <int> key_vec;

int main()
{
	// f(x) 입력받기
	cin >> k1;
	for (int i = 0; i < k1; i++)
	{
		int x, y;
		cin >> x >> y;
		mapset[x].push_back(y);
		key_vec.push_back(x);
	}

	// g(x) 입력받기
	cin >> k2;
```

```
	for (int i = 0; i < k2; i++)
	{
		int x, y;
		cin >> x >> y;
		mapset[x].push_back(y);
		key_vec.push_back(x);
	}

	// 시작값, 끝값 입력받기
	cin >> a >> b;

	// x좌표 정렬 + 중복제거
	sort(key_vec.begin(), key_vec.end());
	key_vec.erase(unique(key_vec.begin(), key_vec.end()), key_vec.end());

	// x1 < x2 -> f(x1) <= f(x2)
	for (int i = 0; i < key_vec.size()-1; i++)
	{
		int cur = key_vec[i]; // f(x)
		int next = key_vec[i+1]; // f(x+1)
		int cur_val = *max_element(mapset[cur].begin(), mapset[cur].end());
		int next_val = *max_element(mapset[next].begin(), mapset[next].end());
		// 각 key의 첫번째 value에 최댓값 대입
		mapset[cur][0] = cur_val;
		mapset[next][0] = next_val;
		// f(x) > f(x+1)이면
		if (cur_val > next_val)
			mapset[next][0] = cur_val; // f(x) = f(x+1)
	}

	// 현재 위치 (x값)
	int start = a;
	while (1)
	{
		// 끝 변수
		int end = upper_bound(key_vec.begin(), key_vec.end(), start) -
key_vec.begin();
		// end값이 첫 인덱스을 경우 -> val = 0이므로 통과
		if (end == 0)
		{
			start = key_vec[end];
			continue;
		}
```

```
                // start보다 큰 값이 없을 경우
                else if (end >= key_vec.size())
                {
                        // 이전 계단 값 x (끝값 - 현재위치 +1)
                        int pre_val = mapset[start][0] % 10007;
                        answer += (((b - start + 1) % 10007) * (pre_val % 10007)) %
10007;
                        break;
                }

                // 현재 위치 == 끝 변수
                if (key_vec[end] == start)
                {
                        answer += (mapset[key_vec[end]][0] % 10007);
                        start = key_vec[end+1];
                }
                // 현재 위치 < 끝 변수
                else
                {
                        // 이전 계단 값 x (끝 변수 - 현재위치)
                        int pre_val = mapset[key_vec[end-1]][0] % 10007;
                        answer += (((key_vec[end] - start) % 10007) * (pre_val %
10007)) % 10007;
                        start = key_vec[end];
                }
        }

        // 정답 출력
        cout << answer % 10007 << endl;

        return 0;
}
```

## 2) 문제 설명

문제는 계단함수 f(x)와 g(x)가 존재하고 [a, b]의 구간에서의 최댓값을 구하는 문제입니다.
(x,f(x))의 관계이므로 map을 사용해서 key-value의 관계로 구현했습니다.
f(x)와 g(x)를 비교해서 구간마다 최댓값을 가지는 함수 h(x)를 만들어서
해결하려고 생각했습니다.

여기서 문제는 map에서는 유일한 key값을 가지기 때문에 두 함수를 합칠 때 문제가 생깁니다.
하나의 x좌표에 여러 개의 함숫값이 생기므로 이 함숫값을 비교하기 위해서
따라서 map <int, int>가 아닌 map <int, vector<int>>로 구현해서
key값에서의 최대의 value를 처음 value에 저장하도록 했습니다.

두 번째 문제는 [a, b]의 범위가 최대 40억까지 입력될 수 있기 때문에
모든 x좌표에 대한 함숫값을 구하는 것이 아니라 계단함수임을 이용해서
구간의 길이를 분할한 후, "구간의 길이 x 구간의 함숫값"로 계산했습니다.

설명을 위해서 간단한 예시를 들자면,
f(x)
(~, 3) = 0
[-3, 2] = 2
[3] = 3
[4] = 4
[5] = 5
[6, ~] = 7

g(x)
(~, 2) = 0
[2, 4] = 1
[5] = 5
[6, ~] = 6

h(x) -> f(x), g(x)를 비교해서 함숫값이 더 큰 값만 취한 함수
(~, 3) = 0
[-3, 2] = 2
[3] = 3
[4] = 4
[5] = 5
[6, ~] = 7

[-10, -2] = 7 * 0 = 0
[-3, 2] = 6 * 2 = 12
[3] = 3
[4] = 4
[5] = 5
[6, 10] = 5 * 7 = 35

answer = 0 + 12 + 3 + 4 + 5 + 35 = 59입니다.