[Baekjoon] 1202 - 보석도둑

태그	baekjoon
공개여부	☑
작성일자	@January 13, 2022

상덕이라는 친구가 보석을 터는 문제다. 가방 K개, 각 가방에 담을 수 있는 최대 무게 Ci, 가방에는 최대 한개의 보석만 넣을 수 있는 조건이 주어졌을 때, 최대 가격을 구하는 프로그램을 작성하면 된다.

배낭채우기 문제가 떠오르는 문제같다. 문제는 배낭채우기를 푼지 정말 오래되었다는점... 데이터구조 응용시간때 (2학년2학기) 한 뒤로는 안한것같기도..?

그래도 뭐, 그리디 아니면 DP임은 알 수 있었다. 가방무게들을 정렬하고, 보석은 무게는 작고 가격은 높은 기준으로 정렬한 뒤, 담으면 되지 않을까? 싶었다.

배낭채우기문제는 '내 기억으로는' 최대한 많이 채울 수 있을때, 가장 큰 가치를 지니는것으로 생각하고 문제를 풀었는데, 이 문제 또한 이를 베이스로 깔고간다고 판단했다.

어짜피 가방에는 최대 한개의 보석만 넣을 수 있기때문에, 가장 비싼 보석을 넣되, 그 보석은 가장 큰 무게를 담을 수 있는 가방에 넣으면 되지 않나? 싶다. 한마디로 그리디이다.

문제에 힌트가 나와있어 봤을때, “두 번째 예제의 경우 첫 번째 보석을 두 번째 가방에, 세 번째 보석을 첫 번째 가방에 넣으면 된다.” 라고 적었다. 이를 보고 내가 생각한것이 맞을 수 있겠다는 생각이 들었다. 그래서 일단 구현해보았다.

```
#include <iostream>
#include <algorithm>
#include <vector>

using namespace std;

bool cmp(pair<int, int> a, pair<int, int> b) {
    if (a.second < b.second)
        return false;
    else {
        if (a.second == b.second)
            if (a.first < b.first)
                return false;
        return true;
    }
}

int main(void)
{
    ios::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);

    int N, K, pos = 0, sum = 0;
    vector<int> C;
    vector<pair<int, int>> MV;

    cin >> N >> K;
    for (int i = 0; i < N; i++) {
        pair<int, int> tmp;
        cin >> tmp.first >> tmp.second;
        MV.push_back(tmp);
    }
    for (int i = 0; i < K; i++) {
        int tmp;
        cin >> tmp;
        C.push_back(tmp);
    }

    sort(MV.begin(), MV.end(), cmp);
    sort(C.begin(), C.end(), greater<int>());
    for (int i = 0; i < K && i < N; i++)
    {
        while (MV[pos].first > C[i]) {
            if (pos >= N)
                break;
            pos++;
        }
        if (pos >= N)
```

```
            break;
        sum += MV[pos].second;
    }

    cout << sum << '\n';

    return 0;
}
```

첫제출은 세그먼트 폴트가 떠서, 바로 인지하고 수정하였으나, 틀렸다. 아무래도 풀이가 문제인 것같다.

결국 그래서 다른사람이 푼 코드를 보았다.

결과적으론 같은 풀이기는한데, 다들 우선순위큐를 이용하였다. 왜 이용하였는가 하니, 가장 가치가 높은 순으로 보석을 정렬해두기 위함이다. 뺄때마다 가장 가치가 큰 보석만 나올 수 있도록.

그럼 여기서 질문이 생긴다면, 그냥 가치순으로 정렬하고, 그다음 무게순으로 이어서 정렬하면 되지 않나? 라고 생각할 수 있지만, 바로 이 생각은 사라졌다. 결국 가방에 넣어야되는 것은 해당 가방에 넣을 수 있는 가장 가치가 높은 보석이다. 따라서 무게에 따라 정렬을 해둔 뒤, 우선순위 큐에 가방에 넣을 수 있는 무게보다 작은 보석들을 전부 넣으면, 그 보석들은 자연스럽게 가치에따라 정렬이 되는것이다.

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <queue>

using namespace std;

int main(void)
{
    ios::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);

    int N, K, pos = 0;
    long long sum = 0;
    vector<int> C;
    vector<pair<int, int>> MV;
    priority_queue<int, vector<int>, less<int>> pq;

    cin >> N >> K;
    for (int i = 0; i < N; i++) {
        pair<int, int> tmp;
        cin >> tmp.first >> tmp.second;
```

```
        MV.push_back(tmp);
    }
    for (int i = 0; i < K; i++) {
        int tmp;
        cin >> tmp;
        C.push_back(tmp);
    }

    sort(MV.begin(), MV.end());
    sort(C.begin(), C.end());
    for (int i = 0; i < K; i++)
    {
        while (pos < N && C[i] >= MV[pos].first) {
            pq.push(MV[pos].second);
            pos++;
        }
        if (pq.empty() != true) {
            sum += pq.top();
            pq.pop();
        }
    }

    cout << sum << '\n';

    return 0;
}
```

구현에서 내가 신경썻던것은, 무게와 비용이 같이 붙어다녀야한다는 강박(?) 이 있었다. 그래서 이전 코드를 보았을때, 정렬도 같이 하는것을 볼 수 있는데, 이 문제같은 경우는 오히려 그것과 멀어져서, 가방에 넣을 수 있는 보석 가치만 정렬한 개념이기 때문에, 문제가 요구하는 바가 무엇인지, 어떻게 짤지 등을 더 세밀하게 생각해야될것같다.