BOJ 23559 밥

소프트웨어학과 201921017 이지우

Step1. 접근

문제

제주대 학생회관 식당에는 두 개의 메뉴가 있다. 코너 A로 가면 5,000원짜리 메뉴를 먹을 수 있고, 코너 B로 가면 1,000원짜리 메뉴를 먹을 수 있다.

준원이는 대면 수업이 시작되는 바람에 이제 남은 학기의 N일동안 매일 학식의 두 메뉴 중 정확히 하나를 골라서 먹어야 한다. N일간의 두 메뉴는 이미 공지되어 있고, 준원이는 이미 모든 날의 각 메뉴가 얼마나 맛있을지 수치를 매겨 두었다.

준원이는 N일간 학식에 총 X원 이하를 써야 한다.

여러분이 N일간 준원이의 메뉴를 잘 골라서, 고른 메뉴의 맛의 합을 최대화 해주자!

입력

첫째 줄에는 두 정수 N, X가 주어진다.

둘째 줄부터 N개의 줄에, 각 날에 먹을 수 있는 5,000원짜리 메뉴의 맛 A와 1,000원짜리 메뉴의 맛 B가 공백을 사이에 두고 주어진다.

출력

준원이가 고른 메뉴들의 맛의 합을 최대화했을 때의 값을 출력하라.

Step1. 접근

제한

- $1 \le N \le 100\,000$
- $1\,000N \le X \le 5\,000N$
- $1 \le A \le 10,000, 1 \le B \le 10,000$

- 어느 날의 A 코스와 B 코스의 만족도 차이가 크다면
- 비싸더라도 그 날은 해당 코스를 먹는 게 낫다!

- 반대로 만족도 차이가 크지 않다면
- 적당히 상황 봐서 골라 먹으면 된다!(즉, 별로 중요하지 않다)

- N일 간의 메뉴는 이미 공지되어 있고
- 얻을 수 있는 만족도의 최대치를 구해주면 되기 때문에
- 식단(menu)을 A코스 B코스의 만족도 차이가 큰 순으로 내림차순 정렬한다

Step1. 접근

- 준원이는 매일매일 A 코스 / B 코스 중에 먹어야 함

- 만약 오늘(today) 돈이 5000원 이상이고(==A 코스를 먹을 수 있고),
- A코스를 먹는 것이 더 만족도가 높다면,
- A코스를 먹는다

- 만약 그렇지 않다면
- B코스를 먹는다

Step1. 접근

- 준원이는 매일매일 A 코스 / B 코스 중에 먹어야 함

- 만약 오늘(today) 돈이 5000원 이상이고(==A 코스를 먹을 수 있고),
- A코스를 먹는 것이 더 만족도가 높다면,
- A코스를 먹는다

- 만약 그렇지 않다면
- B코스를 먹는다

Step1. 접근

- 여기서 잠깐!!!

- 준원이는 매일매일 학식을 먹어야 되기 때문에
- N일 간 반드시, 적어도! 1000 * N 원을 쓸 수밖에 없다(N일 간 B코스만 먹는 경우)

- 이를 간과하고
- 현재 남은 돈이 5000원보다 많다고 무작정 A코스를 먹어버리면 틀린 답이 나오게 된다

Step1. 접근

- N=5, X=5000
- 50 10
- 10 50
- 10 50
- 10 50
- 10 50

- 이 경우 첫날 5000원을 모두 써서 남은 4일간 아무것도 먹지 못하게 되는 일이 생긴다

- 따라서 1000 * N원을 이미 소비(cost)했다고 생각하고(반드시 쓸 수밖에 없는 돈이기 때문에)
- 남은 돈으로 A코스를 먹을 수 있는지를(-4000하여) 확인해야 한다

Step2. 구현 - main

```cpp
int main(void)
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);

    int n, x;

    cin >> n >> x;

    vector<vector<int>> menu(n, vector<int>(2));

    for (int i = 0; i < n; i++) {
        cin >> menu[i][0] >> menu[i][1];
    }

    sort(menu.begin(), menu.end(), compare);

    int result = 0;
    int cost = 1000 * n;
    for (int today = 0; today < n; today++) {
        if ((x >= cost + 4000) && (menu[today][0] > menu[today][1])) {
            cost += 4000;
            result += menu[today][0];
        }
        else {
            result += menu[today][1];
        }
    }

    cout << result << endl;

    return 0;
}
```

Step2. 구현 - compare

```
bool compare(vector<int>& a, vector<int>& b)
{
    return a[0] - a[1] > b[0] - b[1];
}
```

menu[a][0] – menu[a][1] > menu[b][0] - menu[b][1] 이면,

True을 반환하여 menu[a]가 더 우선되도록 함

END