# 1202 - 보석 도둑

| 난이도 | 골드2 |
| --- | --- |

## 문제

| 시간 제한 | 메모리 제한 | 제출 | 정답 | 맞힌 사람 | 정답 비율 |
| --- | --- | --- | --- | --- | --- |
| 1 초 | 256 MB | 27971 | 6515 | 4582 | 22.304% |

### 문제

세계적인 도둑 상덕이는 보석점을 털기로 결심했다.

상덕이가 털 보석점에는 보석이 총 N개 있다. 각 보석은 무게 $M_i$와 가격 $V_i$를 가지고 있다. 상덕이는 가방을 K개 가지고 있고, 각 가방에 담을 수 있는 최대 무게는 $C_i$이다. 가방에는 최대 한 개의 보석만 넣을 수 있다.

상덕이가 훔칠 수 있는 보석의 최대 가격을 구하는 프로그램을 작성하시오.

### 입력

첫째 줄에 N과 K가 주어진다. ($1 \leq N, K \leq 300,000$)

다음 N개 줄에는 각 보석의 정보 $M_i$와 $V_i$가 주어진다. ($0 \leq M_i, V_i \leq 1,000,000$)

다음 K개 줄에는 가방에 담을 수 있는 최대 무게 $C_i$가 주어진다. ($1 \leq C_i \leq 100,000,000$)

모든 숫자는 양의 정수이다.

## 출력

첫째 줄에 상덕이가 훔칠 수 있는 보석 가격의 합의 최댓값을 출력한다.

## 예제 입력 1 복사

```
2 1
5 10
100 100
11
```

## 예제 출력 1 복사

```
10
```

## 예제 입력 2 복사

```
3 2
1 65
5 23
2 99
10
2
```

## 예제 출력 2 복사

```
164
```

## 힌트

두 번째 예제의 경우 첫 번째 보석을 두 번째 가방에, 세 번째 보석을 첫 번째 가방에 넣으면 된다.

# 풀이과정

처음엔 가방을 기준으로 보석 하나하나 탐색하며 넣어주는 완전탐색으로 했었으나 O(N^2)로 시간초과가 나서, **우선순위 큐**를 활용하였다.

## 💡 주요 포인트

### 1. 작은 가방에 들어갈 수 있는 보석은 그보다 큰 가방에도 들어갈 수 있다.

그냥 보면 당연한 얘기지만 코드로 구현할 때 제일 크게 간과했던 부분이다. 보석들은 배열에 무게를 기준으로 오름차순 정렬했고, 가방들을 크기를 기준으로 오름차순 정렬했다. 그 다음 특정 가방에 들어갈 수 있는 보석을 우선순위 큐에 넣어줬는데, 이때 힙 조건은 **가격 내림차순**이었다.

즉, 첫번째 가방에 들어갈 수 있는 보석들을 뽑아 우선순위 큐에 넣어주고, 가격 내림차순으로 정렬된 우선순위 큐에서 top을 뽑아 총 가격에 더해주었다. 그럼 **현재 우선순위 큐에 남아있는 보석들은 그 이후에 첫번째 가방보다 큰 두번째 가방에도 들어갈 수 있다.**

## 📌 조건

- 그 가방에 담을 수 있는 가능한 보석 중 가장 비싼 것

## 🎈절차

1. 가방 선택
2. 현재 가방에 넣을 수 있는 보석들을 우선순위 큐에 넣어준다.
3. 우선순위 큐의 top을 뽑아 총 가격에 더해준다.
4. 다음 가방으로 넘어가, 모든 가방을 다 볼 때까지 1~3번을 반복한다.

## 소스코드

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.*;

class Jewelry {
    int weight;
    int cost;

    public Jewelry(int weight, int cost) {
        this.weight = weight;
        this.cost = cost;
    }

    public int getWeight() {
        return weight;
    }

    public int getCost() {
        return cost;
    }
}

public class Main {

    static int N, K, M, V;
    static long C;
    static List<Jewelry> jewel;
    static List<Long> bag;

    public static void main(String[] args) throws Exception{
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st = new StringTokenizer(br.readLine());
```

```
        // 힙 조건: 보석들의 가격 내림차순
        PriorityQueue<Jewelry> pq = new PriorityQueue<>(Comparator.comparingInt(Jewelry::getCost).reversed());

        N = Integer.parseInt(st.nextToken());
        K = Integer.parseInt(st.nextToken());

        jewel = new ArrayList<>();
        for(int i=0 ; i<N ; i++){
            st = new StringTokenizer(br.readLine());
            M = Integer.parseInt(st.nextToken());
            V = Integer.parseInt(st.nextToken());
            jewel.add(new Jewelry(M, V));
        }
        // 보석들 - 무게 기준 오름차순 정렬
        Collections.sort(jewel, Comparator.comparingInt(Jewelry::getWeight));

        bag = new ArrayList<>();
        for(int i=0 ; i<K ; i++){
            C = Long.parseLong(br.readLine());
            bag.add(C);
        }
        // 가방들 - 크기 기준 오름차순
        Collections.sort(bag);

        int curJ = 0;
        long total = 0;
        for(int i=0 ; i<K ; i++) {
            long curBag = bag.get(i);

            while(curJ < N && jewel.get(curJ).weight <= curBag) {
                pq.offer(jewel.get(curJ++));
            }

            if(!pq.isEmpty()){
                total += pq.poll().cost;
            }
        }
        System.out.println(total);
    }
}
```