벡준 문제풀이 12865 배낭 문제

문제 설명:

이 문제는 아주 평범한 배낭에 관한 문제이다.

한 달 후면 국가의 부름을 받게 되는 준서는 여행을 가려고 한다. 세상과의 단절을 슬퍼하며 최대한 즐기기 위한 여행이기 때문에, 가지고 다닐 배낭 또한 최대한 가치 있게 싸려고 한다.

준서가 여행에 필요하다고 생각하는 N개의 물건이 있다. 각 물건은 무게 W와 가치 V를 가지는데, 해당 물건을 배낭에 넣어서 가면 준서가 V만큼 즐길 수 있다. 아직 행군을 해본 적이 없는 준서는 최대 K만큼의 무게만을 넣을 수 있는 배낭만 들고 다닐 수 있다. 준서가 최대한 즐거운 여행을 하기 위해 배낭에 넣을 수 있는 물건들의 가치의 최댓값을 알려주자.

문제 입력:

첫 줄에 물품의 수 N(1 ≤ N ≤ 100)과 준서가 버틸 수 있는 무게 K(1 ≤ K ≤ 100,000)가 주어진다. 두 번째 줄부터 N개의 줄에 거쳐 각 물건의 무게 W(1 ≤ W ≤ 100,000)와 해당 물건의 가치 V(0 ≤ V ≤ 1,000)가 주어진다.

입력으로 주어지는 모든 수는 정수이다.

문제 출력:

예제 입력 1 복사

```
4 7
6 13
4 8
3 6
5 12
```

예제 출력 1 복사

```
14
```

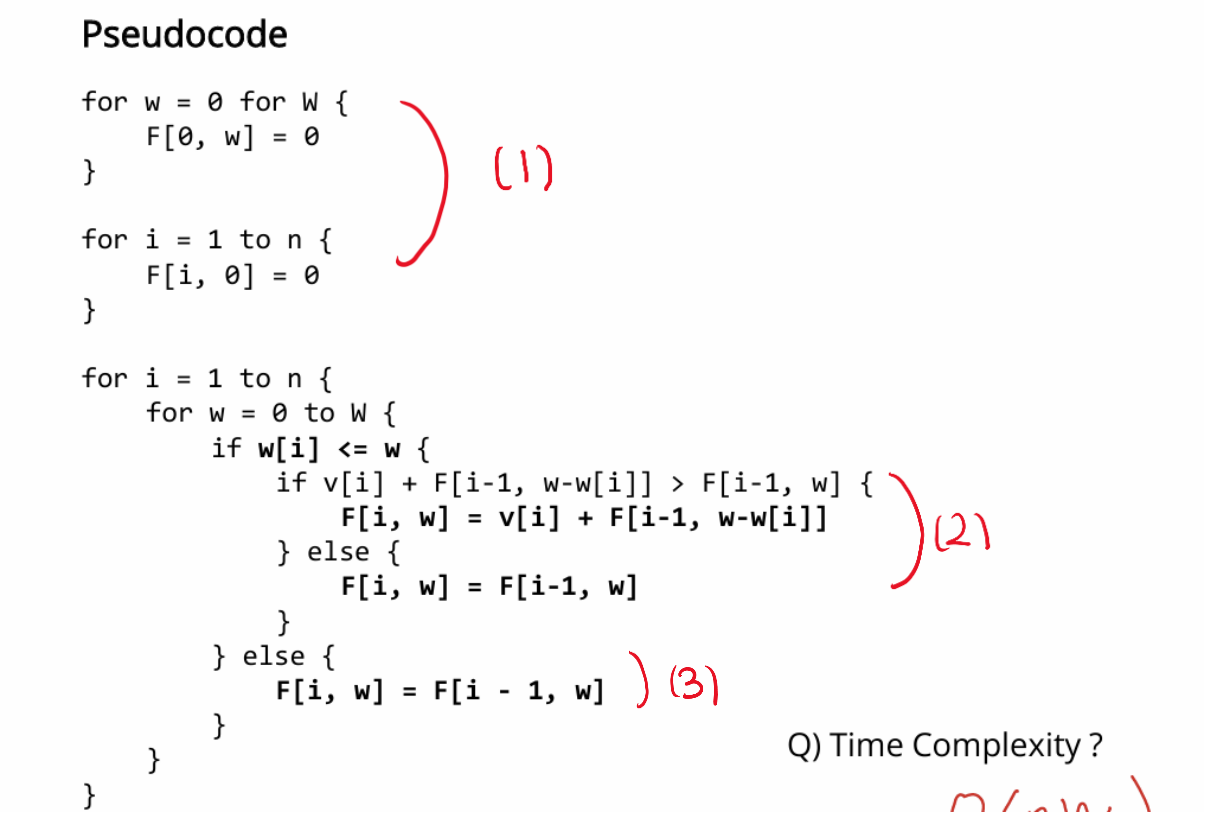
문제 개인 설명:

이 문제는 배낭 문제로 DP문제로 유명하다. 아이템 개수와 배낭 무게가 주어졌을 때 가장 가치가 높게 가방에 담는 최대값을 구하는 문제이다.

즉 4 7을 입력했을 때 무게 7로 가장 max로 담을 수 있는 가치는 4 8, 3 6으로 14가 최대값이다.

문제 풀이 방법:

DP 점화식이 떠오르지 않아 대학교 알고리즘 강의 자료를 보아 다음과 같은 수도 코드를 얻었다.



먼저, 배낭의 무게 w가 0이거나 아이템을 선택할 수 없는 경우에는 최대 가치가 0이 되므로, 이를 초기화한 구문이 (1)이다.

그 후, 배낭 용량 w가 i번째 아이템의 무게 w[i]보다 클 경우, 해당 아이템을 배낭에 담을 수 있는 경우이다. 이때, 아이템을 넣었을 때의 가치는 i-1번째까지 고려한 최적 값에 현재 아이템의 가치를 더한 값과, 아이템을 넣지 않았을 때의 최적 값 중 더 큰 값을 선택하여 배열에 갱신한다. 이를 (2)에서 처리한다.

배낭에 남은 용량이 i번째 아이템의 무게 w[i]보다 작을 경우, 해당 아이템은 담을 수 없

으므로 이전 값을 그대로 사용한다. 이 과정은 (3)에서 처리된다.

최종적으로, dp[N][V]는 N개의 아이템을 고려하고 배낭 용량 V에 담을 수 있는 최대 가치를 저장하므로 이를 출력하여 문제를 해결한다.

실제 구현코드 DP:

```
for(int i=1; i<=N; i++){
    for(int j=1; j<=V; j++){
        if(w[i]<=j){
            //선택한 경우
            if(v[i]+dp[i-1][j-w[i]]>dp[i-1][j]){
                dp[i][j]=dp[i-1][j-w[i]]+v[i];
            }
            //선택하지 않은경우
            else{
                dp[i][j]=dp[i-1][j];
            }
        }
        else{
            dp[i][j]=dp[i-1][j];
        }
    }
}
System.out.println(dp[N][V]);
/*
```

느낀점:

알고리즘 시간에 배운 알고리즘이라 영감은 알았지만 DP식을 구하기 힘들었다. 사실 DP식을 봐도 진행하지 못하여 수도코드를 참조해서 구현하였다.

하지만 이 경험을 얻어 추후에 이런 문제가 다시 나온다면 까먹지 않고 구현할 수 있을 것이라고 생각했다.