Algorithm
- Dynamic programming -

제출일자	2020.10.08
과제 번호	05
이　　름	강인한
학　　번	201701969

실행결과 :

시간복잡도 설명 : 이중 반복문을 통해 table 만들면서 결과 값을 알아내었다. 첫 반복문에서는 item의 개수만큼 반복하였고, 두 번째 반복문에서는 최대 무게 한도 만큼 반복했다. item의 개수를 n, 최대 무게 한도를 w라고 할 때 해당 코드의 시간 복잡도는 O(nw)라고 볼 수 있다. 뒤에 while 문을 통해 결과에 포함된 item을 찾긴 했지만 while문에서 반복은 n만큼 되었지만 nw에 큰 영향을 끼치지 못한다.

풀이 과정 : value 값만 모아 배열에 저장하고 weight값만 또 다른 배열에 저장한다.그리고 table을 만들어 한 개씩 채워나가면 마지막 값에 max value를 찾아낼 수 있다. 한 개씩 채워 나갈 때의 기준을 점화식으로 표현하자면 다음과 같다.

recursive equation

if w_i>w then M(i,w) = M(i-1,w)

else M(i,w) = max(value(i)+M(i-1,w-w_i),M(i-1,w)

코드 설명 : 핵심적인 부분은 table을 어떻게 채울지에 대한 코드인 것 같다. table에는 i,j를 index로 갖는다고 가정하면 i,j에서는 i번째 item까지만 사용하고 j만큼의 최대 무게를 갖는 item의 조합 중에 가장 큰 value값을 갖는 조합을 찾는 것이다. 가장 큰 조합을 찾기 위해서는 우선 i번째 item의 weight가 j보다 큰지 비교를 해줘야 한다.

optimal substructure

i) j보다 크다면 조합에 포함될 수 없음을 의미하므로 테이블 상에서 i-1,j 의 index를 갖는 값이 그대로 들어오게 된다.

ii) j보다 작다면 추가로 한 가지의 조건을 걸어 분류해야 한다. 바로 value의 더 큰 값을 택해야 한다. i-1,j의 index에서 i의 weight만큼 뺀 index, i-1,j-weight(i)의 값에서 value(i)를 더 한 값과 테이블 상에서 윗 열의 값(i-1,j)을 비교하여 큰 값이 table의 i,j 값이 된다.

느낀 점 : 난이도에 대해서는 저번 주 과제보다 조금 더 생각할 부분이 많아 어려웠던 것 같습니다. 포함된 item을 찾는 과정에서 while문을 사용했는데 사실 for문도 가능할 것 같습니다. 저는 결국 못 찾았지만 table을 채우는 두 개의 반복문에서 결과 값에 포함된 item을 찾을 수 있지 않을까 생각도 듭니다. 마지막으로 수업시간에 배운 점화식에서 많은 힌트를 얻었습니다.