과제

장유선

2023.08.29

1. 문제 정의

넣을 수 있는 무게가 정해져 있는 가방에 무게와 가치가 정해져 있는 짐을 넣을려고 한다. 가장 높은 가치를 출력하라

위 알고리즘을 2가지로 구현하세요

- 무식하게 모든 Case를 다 계산하는 방법
- 다이나믹 프로그래밍을 이용하는 방법

입력값

무게: 7KG 물건 - 6KG, 13$ - 4KG, 8$ - 3KG, 6$ - 5KG, 12$

결과

14$

2. 개념 설명

<초기 상태>

weight	A	B	C	D
0	0			
1	0			
2	0			
3	0			

4	0			
5	0			
6	0			
7	0			

<A 를 넣었을 때>

weight	A	B	C	D
0	0			
1	0			
2	0			
3	0			
4	0			
5	0			
6	13			
7	0			

<B 를 넣었을 때>

weight	A	B	C	D
0	0	0		
1	0	0		
2	0	0		
3	0	0		
4	0	8		

5	0	0		
6	13	13		
7	0	0		

<C 를 넣었을 때>

weight	A	B	C	D
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	6	
4	0	8	8	
5	0	0	0	
6	13	13	13	
7	0	0	14	

<D 를 넣었을 때>

weight	A	B	C	D
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	6	6
4	0	8	8	8
5	0	0	0	12

6	13	13	13	13
7	0	0	14	14

3. Python Code Hard Copy

```python
# 1. 모든 Case를 다 계산
def solve1(weight_limit, weights, values):
    n = len(weights)
    max_value = 0

    for i in range(2 ** n):
        combination = []
        total_weight = 0
        total_value = 0

        for j in range(n):
            if (i >> j) & 1:
                combination.append(j)
                total_weight += weights[j]
                total_value += values[j]

        if total_weight <= weight_limit and total_value > max_value:
            max_value = total_value

    print("1번 - 최대 가치:", max_value)

# 2. 다이나믹 프로그래밍
def solve2(limit, w, v):
    n = len(w)
    table = [0] * (limit+1)

    for i in range(n):

        if w[i] > limit:
            continue

        for j in range(limit, 0, -1):
            if j + w[i] <= limit and table[j] != 0:
                table[j+w[i]] = max(table[j+w[i]], table[j] + v[i])

        table[w[i]] = max(table[w[i]], v[i])

    print("2번 - 최대 가치:", max(table))
```

```
# 입력값
limit = 7
w = [6, 4, 3, 5]
v = [13, 8, 6, 12]
solve1(limit, w, v)
solve2(limit, w, v)
```

4. Code 설명

```
# 1. 모든 Case를 다 계산
def solve1(weight_limit, weights, values):
    n = len(weights)  # 물건의 개수
    max_value = 0  # 최대 가치 초기화

    # 모든 조합을 생성 => 2의 n제곱
    for i in range(2 ** n):
        combination = []  # 현재 조합에 선택된 물건의 인덱스를 저장
        total_weight = 0  # 현재 조합의 물건 무게 합
        total_value = 0  # 현재 조합의 물건 가치 합

        # 이진수의 각 비트가 물건의 선택 여부를 나타냄
        for j in range(n):
            if (i >> j) & 1: # j번째 물건을 선택했으면
                combination.append(j)  # 인덱스를 추가
                total_weight += weights[j]
                total_value += values[j]

        # 현재 조합의 물건을 선택했을 때, 무게 제한을 넘지 않고 최대 가치인 경우
        if total_weight <= weight_limit and total_value > max_value:
            max_value = total_value  # 최대 가치 갱신

    print("1번 - 최대 가치:", max_value)

# 2. 다이나믹 프로그래밍
def solve2(limit, w, v):
    n = len(w) # 물건의 개수

    # 테이블을 초기화
    # i는 배낭의 무게
    # 테이블[i]는 해당 무게에서의 최대 가치
    table = [0] * (limit+1)

    for i in range(n):
```

```
        # 무게 제한보다 크면 넘어감
        if w[i] > limit:
            continue

        # 배낭의 무게 제한부터 1 까지 역순으로 반복
        for j in range(limit, 0, -1):
            # 현재 물건을 배낭에 넣을 수 있는 경우를 고려
            if j + w[i] <= limit and table[j] != 0:
                # 물건을 넣는 경우와 넣지 않는 경우 중 더 큰 가치를 선택하여 업데이트
                table[j+w[i]] = max(table[j+w[i]], table[j] + v[i])

        # 0 번째
        table[w[i]] = max(table[w[i]], v[i])

    print("2 번 - 최대 가치:", max(table))

# 입력값
limit = 7  # 배낭의 무게 제한
w = [6, 4, 3, 5]  # 각 물건의 무게 리스트
v = [13, 8, 6, 12]  # 각 물건의 가치 리스트
solve1(limit, w, v)
solve2(limit, w, v)
```

5. 결과

```
1번 - 최대 가치: 14
2번 - 최대 가치: 14
```

6. 결과 화면