ILE3-020 다이나믹 프로그래밍

중복되는 문제는 한번만 계산 후 메모리에 저장하여 중복 계산을 피해 성능을 향상시키는 기법
직관적으로 풀이 방법을 판단하기 힘들어 실제로는 사용하기는 힘든 방식

재귀 함수로 해결한 피보나치 수열

```
fibonacci(4)
- fibonacci(2)
  - fibonacci(0)
  - fibonacci(1)
- fibonacci(3)
  - fibonacci(1)
  - fibonacci(2)
    - fibonacci(0)
    - fibonacci(1)
```

→ fibonacci(2), fibonacci(1), fibonacci(0)이 중복되어 호출이 되는 것을 알 수 있음

다이나믹 프로그래밍을 적용하여 보자

이미 연산된 값은 메모리에 저장해서 다시 연산하지 않도록 변경

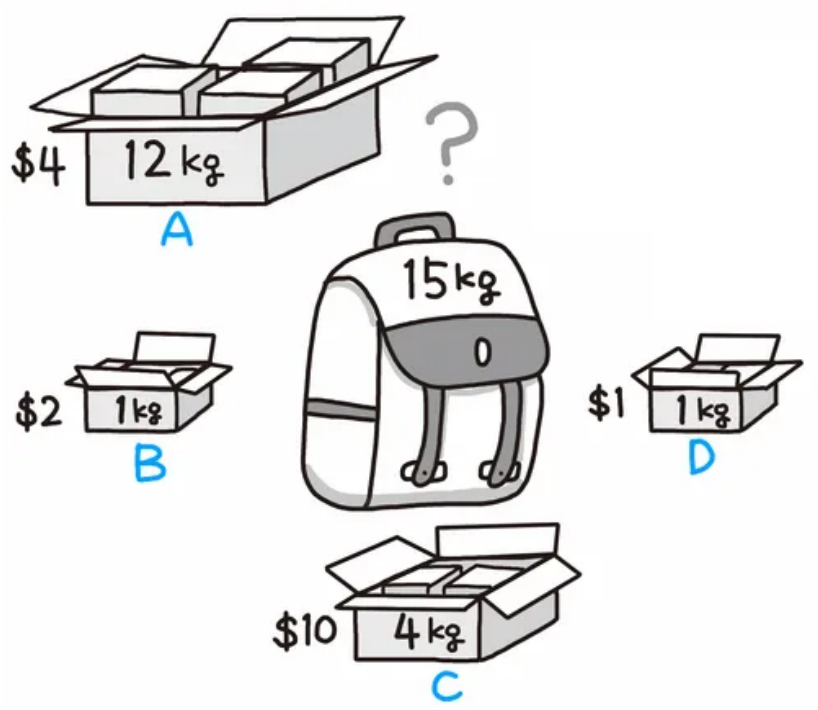
```python
def fibonacci_dynamic(n):
    fib = [0] * (n + 1)  # 결과를 저장할 배열

    fib[0] = 0
    fib[1] = 1

    for i in range(2, n + 1):
        fib[i] = fib[i - 1] + fib[i - 2]

    return fib[n]
```

피보나치 연산에 대한 각 방식에 대한 연산량의 차이

횟수	재귀 함수 `2^n-1`	다이나믹 프로그래밍 `n`
1	1	1
10	1023	10
100	1.26765E+30	100
1000	1.0715E+301	1000

과제

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



해결 방법

단순하게 해결하는 방법

모두 넣는 방법을 계산해 보고 그 때 최대 가치를 판단 → `2^n`

넣은 물건	무게	가치	넣은 물건	무게	가치
0000	0	0	1000	12	4
0001	1	1	1001	13	5
0010	4	10	1010	16	14
0011	5	11	1011	17	15
0100	1	2	1100	13	6
0101	2	3	1101	14	7
0110	5	12	1110	17	16
0111	6	13	1111	18	17

다이나믹 프로그래밍으로 해결하는 방법

무게를 기준으로 가장 무거운 용량을 관리 → 순서대로 각 항목을 넣었을 때를 판단

Table T

무게	Init	iter A	iter B	iter C	iter D
0	-1				

1	-1		2	2
				`if w+i < 15:`
				`t[w+i]=v+t[v]`
					2
					`if w+i < 15:`
					`t[w+i]=v+t[v]`

2	-1				3
3	-1				

4	-1			10	10
					`if w+i < 15:`
					`t[w+i]=v+t[v]`

5	-1			12	12
					`if w+i < 15:`
					`t[w+i]=v+t[v]`

6	-1				13
7	-1				
8	-1				
9	-1				
10	-1				
11	-1				

12	-1	4	4
			`if w+i < 15:`
			`t[w+i]=v+t[v]`
				12
				`if w+i < 15:`
				`t[w+i]=v+t[v]`
					13
					`if w+i < 15:`
					`t[w+i]=v+t[v]`

13	-1		6	12
				`if w+i < 15:`
				`t[w+i]=v+t[v]`
					13

14	-1				
15	-1				

과제 내용

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$