파이썬으로 배우는 알고리즘 기초

Chap 5. 되추적(백트래킹)

4.5 배낭 문제와 탐욕 알고리즘

4.5 배낭 문제와 탐욕 알고리즘

- **배낭 문제** (Knapsack Problem)
  - 도둑이 30kg까지 담을 수 있는 배낭을 메고 곡식 창고에 침투했다.
  - 창고 입구에는 보관중인 곡식의 전체 수량과 1kg당 가격이 적혀 있다.
  - 도둑의 목적은 이익이 최대가 되도록 배낭을 채우는 것이다.

곡식	쌀	보리	밀	옥수수	소금	밤	조
수량(kg)	40	8	5	24	2	13	7
1kg당 가격(원)	7	8	12	2	15	5	8

4.5 배낭 문제와 탐욕 알고리즘

- 배낭 채우기: 탐욕 알고리즘 (The Greedy Approach)
  - 탐욕적인 전략: 가장 값어치가 높은 아이템을 먼저 채우는 것
  - 1kg당 가격을 기준으로 내림차순으로 정렬
  - 배낭의 무게(=30kg)를 초과하지 않을 때까지 비싼 순으로 채우기

곡식	소금	밀	보리	조	쌀	밤	옥수수
수량(kg)	2	5	8	7	40	13	24
1kg당 가격(원)	15	12	8	8	7	5	2

4.5 배낭 문제와 탐욕 알고리즘

- **분할 가능한 배낭 채우기 문제: 탐욕 알고리즘**
  - Fractional Knapsack Problem: **아이템의 분할이 가능할 경우**
  - **탐욕 알고리즘으로 최적의 해를 얻을 수 있다.**

곡식	소금	밀	보리	조	쌀	밤	옥수수
수량(kg)	2	5	8	7	8	0	0
1kg당 가격(원)	15	12	8	8	7	-	-
총 가격	30	60	64	42	56	0	0

Algorithm 4.6: Greedy Algorithm for the Fractional Knapsack Problem

```
def knapsack1(W, w, p):
    n = len(w) - 1
    K = [0] * (n + 1)
    weight = 0
    for i in range(1, n + 1):
        weight += w[i]
        K[i] = w[i]
        if (weight > W):
            K[i] -= (weight - W)
            break;
    return K
```

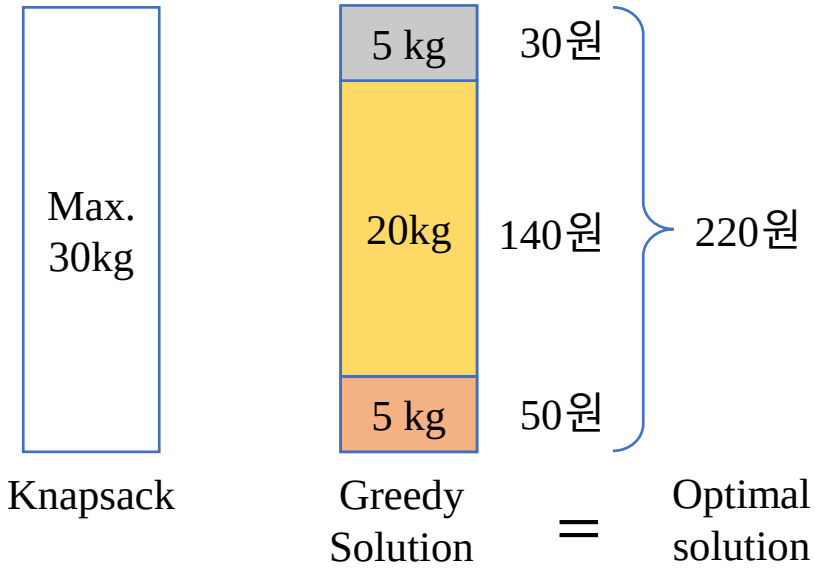
4.5 배낭 문제와 탐욕 알고리즘


```
w = [0, 2, 5, 8, 7, 40, 13, 24]
p = [0, 15, 12, 8, 8, 7, 5, 2]
W = 30
K = knapsack1(W, w, p)
print(K, sum(K))
price = 0
for i in range(1, len(K)):
    price += p[i] * K[i]
print("Total price is", price)
```

4.5 배낭 문제와 탐욕 알고리즘

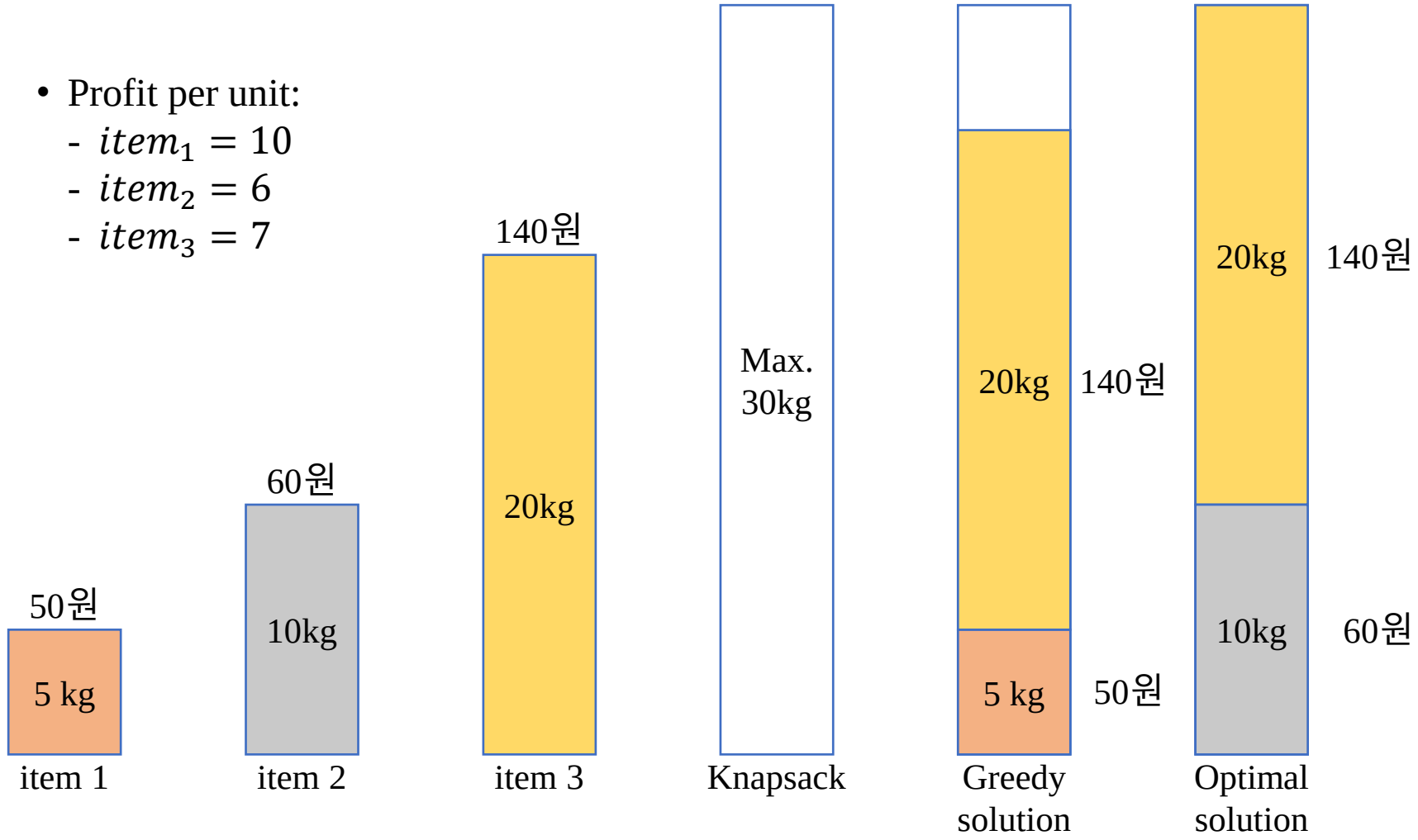
- 탐욕 알고리즘은 최적해를 보장하는가?
  - 아이템의 분할이 가능하면 Greedy가 최적해를 찾아줌

곡식	아이템1	아이템2	아이템3
수량(kg)	5	10	20
1kg당 가격(원)	10	6	7



4.5 배낭 문제와 탐욕 알고리즘

- **만약, 아이템의 분할이 불가능하다면?**



4.5 배낭 문제와 탐욕 알고리즘

- 0-1 **배낭 문제**: 0-1 Knapsack Problem
  - **분할이 불가능한** 0-1 **배낭 문제는 최적화 문제이며,**
    - **탐욕 알고리즘은 최적해를 보장하지 않는다.**
  - **동적 계획법** or **백트래킹** or **분기한정법**

주니온TV@Youtube

자세히 보면 유익한 코딩 채널

https://bit.ly/2JXXGqz

- 여러분의 **구독**과 **좋아요**는 강의제작에 큰 힘이 됩니다.
- 강의자료 및 소스코드: **구글 드라이브**에서 다운로드
  (다운로드 주소는 영상 하단 설명란 참고)

  https://bit.ly/3fN0q8t