파이썬으로 배우는 알고리즘 기초

Chap 5. 되추적(백트래킹)

5.4 부분집합의 합 구하기

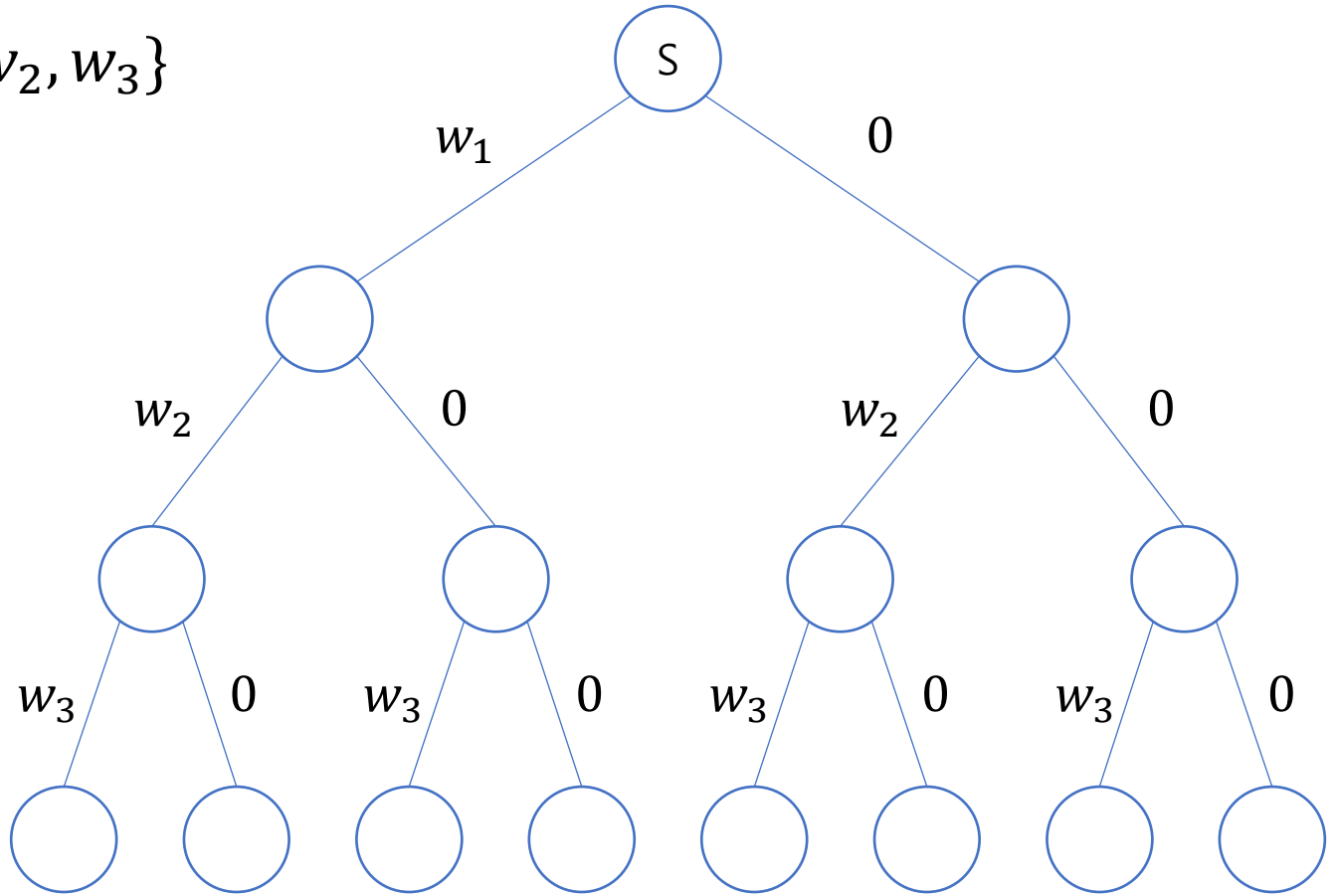
5.4 부분집합의 합 구하기

- **부분집합의 합** 구하기 문제 (Sum-of-Subset Problem)
  - 원소가 n개인 양의 정수 집합 w와 양의 정수 W가 주어졌을 때,
  - 합이 W가 되는 w의 부분집합을 모두 찾아라.

  - 입력 사례:
    - 문제: $n = 5, w = \{5,\ 6,\ 10,\ 11,\ 16\}, W = 21$
    - 해답: $\{5, 6, 10\},\ \{5, 16\},\ \{10, 11\}$

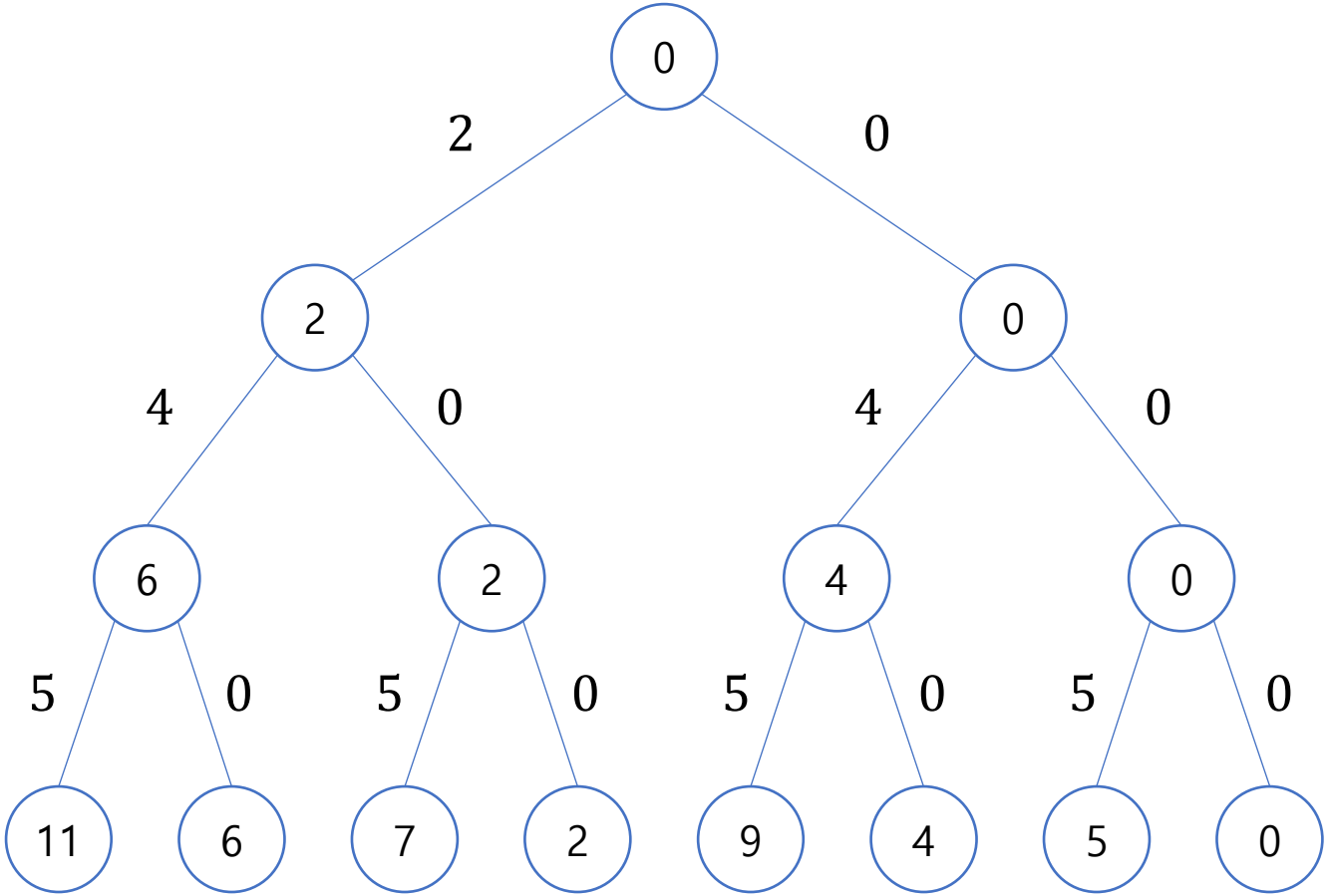
5.4 부분집합의 합 구하기

- **부분집합의 합 구하기: 백트래킹 (backtracking)**
  - 상태공간트리를 만들어 푸는 방법
  - 예) $w = \{w_1, w_2, w_3\}$



5.4 부분집합의 합 구하기

- 상태공간트리의 구축 사례
  - 예) $w_1 = 2, w_2 = 4, w_3 = 5$



5.4 부분집합의 합 구하기

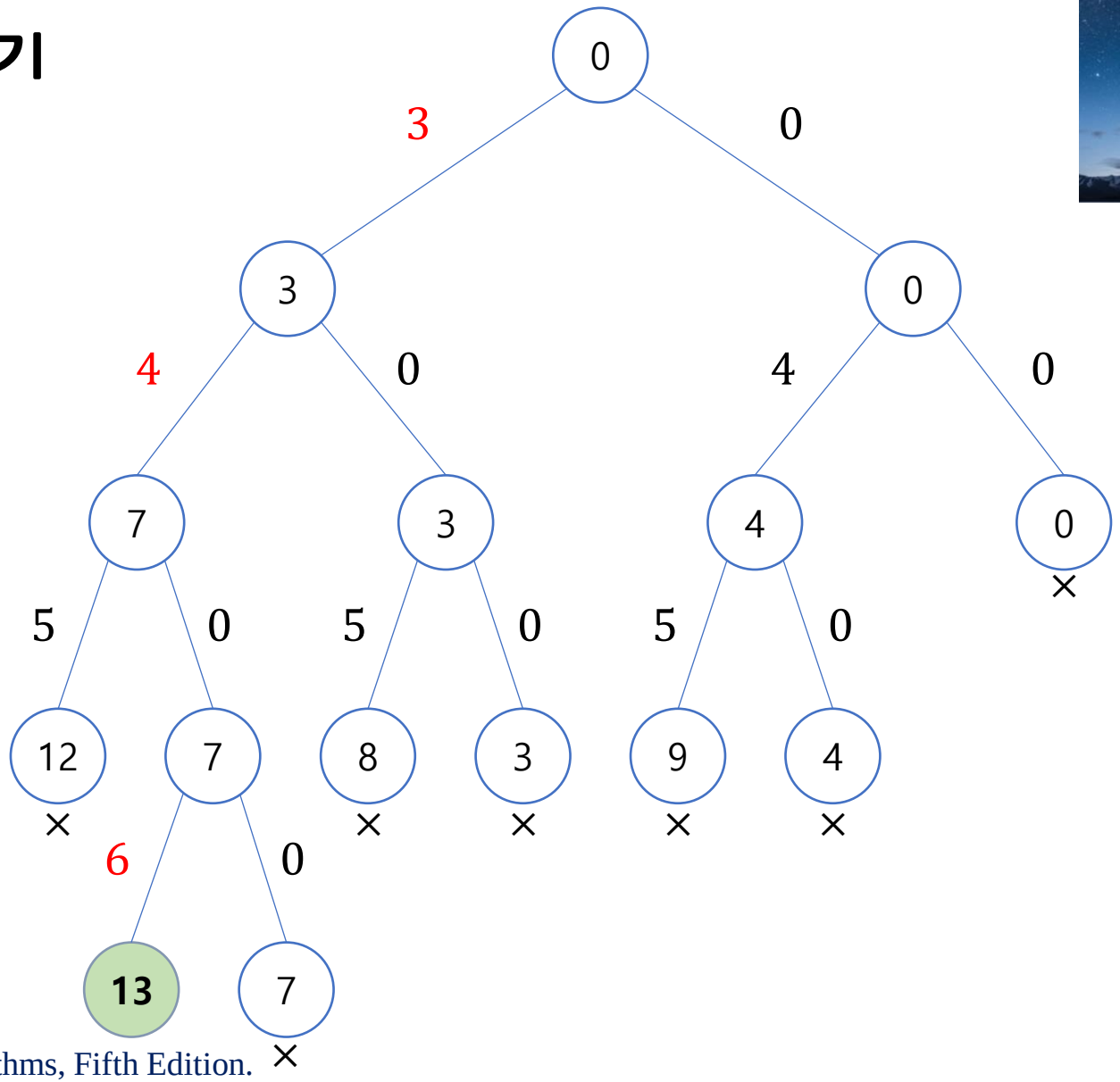
- **가지치기 전략** (pruning strategy)
  - 검색하기 전에 정수 원소를 비내림차순으로 정렬하면,
    - 각 노드가 유망하지 않음을 알 수 있게 된다.
  - 가지치기를 어떻게 할까?
    - i번째 레벨에서 w_{i+1}는 남아 있는 정수 원소 중에서 가장 작은 값이다.
    - 어떤 노드에서 남아 있는 노드 원소의 합이 W의 값보다 작다면, 그 노드의 하위 노드는 더이상 방문할 필요가 없다.

5.4 부분집합의 합 구하기

- 유망 함수의 구현
  - $weight$를 레벨 i까지의 모든 정수 원소의 합이라고 하자.
  - $weight$가 W와 같지 않으면(그 노드에 해답이 있다는 의미),
    다음 조건을 만족하면 레벨 i의 노드는 유망하지 않다.
    - $weight + w_{i+1} > W$.
  - total을 남은 정수 원소의 합이라고 하자.
  - 남은 정수 원소의 합에다 $weight$를 더해도 최소한 W와 같아지지 않으므로,
    다음 조건을 만족하면 이 노드는 유망하지 않다.
    - $weight + total < W$.

5.4 부분집합의 합 구하기

- 유망 함수로 가지치기
  - $n = 4, W = 13$
  - $w = \{3, 4, 5, 6\}$



5.4 부분집합의 합 구하기

Algorithm 5.4: Backtracking for the Sum-of-Subsets Problem

```
def sum_of_subsets (i, weight, total):
    n = len(w) - 1
    if (promising(i, weight, total)):
        if (weight == W):
            print(include[1: n + 1])
        else:
            include[i + 1] = True
            sum_of_subsets(i + 1, weight + w[i+1], total - w[i+1])
            include[i + 1] = False
            sum_of_subsets(i + 1, weight, total - w[i+1])
```

5.4 부분집합의 합 구하기

Algorithm 5.4: Backtracking for the Sum-of-Subsets Problem

```
def promising (i, weight, total):
    if ((weight + total >= W) and
        (weight == W or weight + w[i+1] <= W)):
        return True
    else:
        return False
```

- total = $\sum_{j=1}^{n} w[j]$;
- sum_of_subsets(0, 0, total);

5.4 부분집합의 합 구하기


```
n = 5
W = 21
w = [0, 5, 6, 10, 11, 16]
total = sum(w)
include = [False] * (n + 1)
print(total)
sum_of_subsets(0, 0, total)
```

```
48
[True, True, True, False, False]
[True, False, False, False, True]
[False, False, True, True, False]
```

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