그리디(Greedy) 알고리즘

발표자 주경연

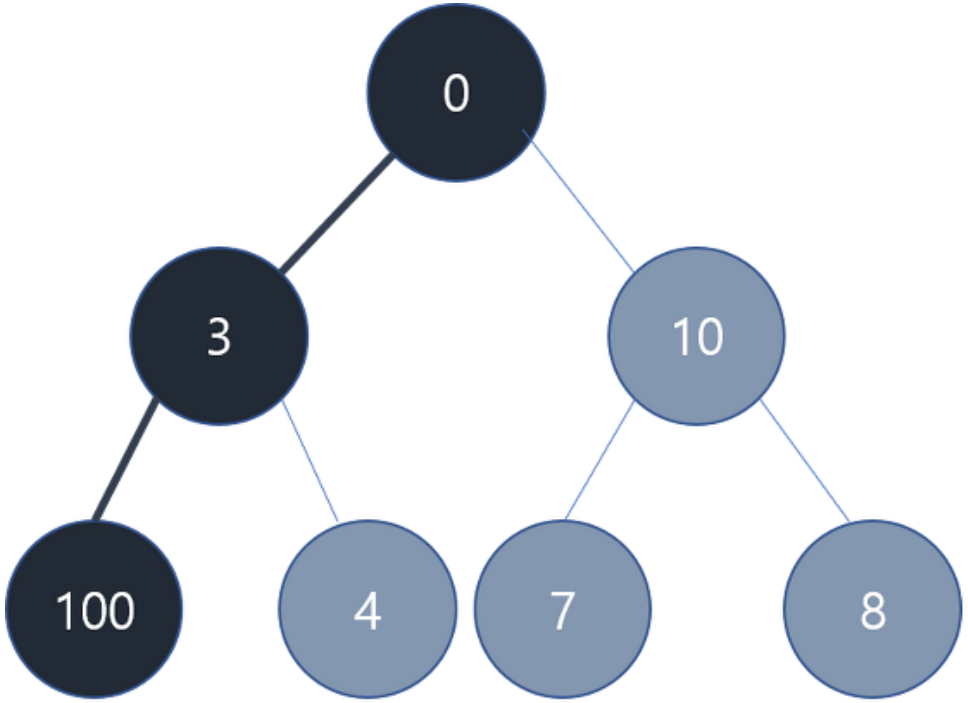
그리디 알고리즘(탐욕법)

- 현재 상황에서 지금 당장 좋은 것만 고르는 알고리즘
- 문제의 성격을 잘 파악하여 대입하고 정당성 분석이 중요

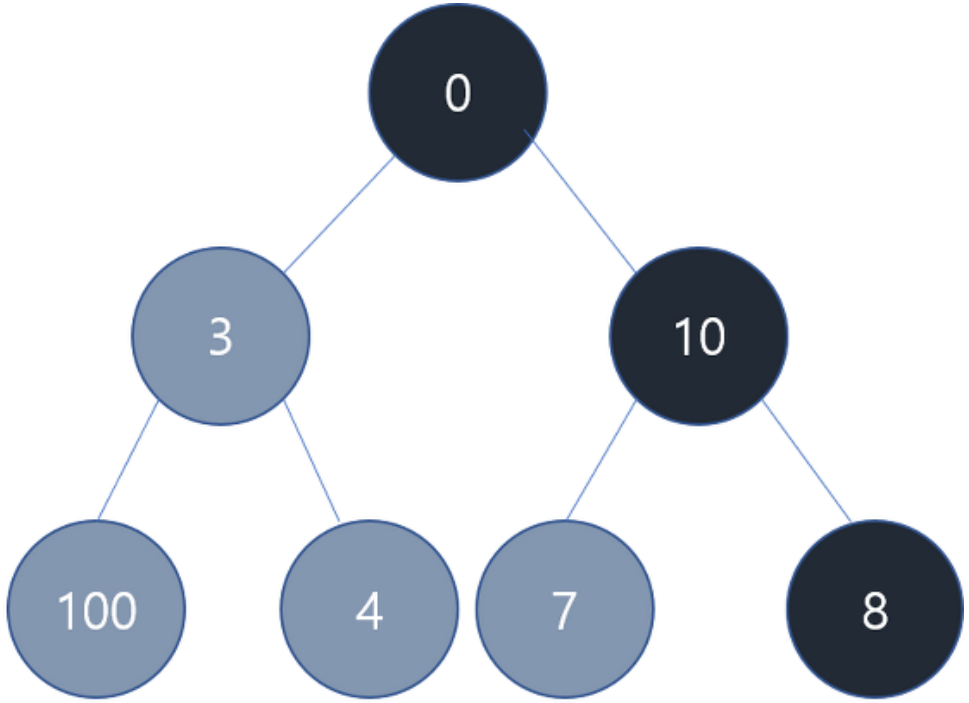
  -> 지금 당장 좋은 것이 반복적으로 실행되어서

  종합적으로 최적의 해를 구할 수 있는지는 보장 X

[문제 상황] 루트 노드부터 시작하여 거쳐 가는 노드 값의 합을 최대로 만들고 싶을 경우



최적의 선택



그리디한 선택

그리디 알고리즘 수행절차

1. 해 선택 (Selection Procedure)

   : 현재 상태에서의 최적의 해를 선택

2. 적합성 검사 (Feasibility Check)

   : 선택된 해가 문제 조건을 만족하는지 검사

3. 해답 검사 (Solution Check)

   : 원래의 문제가 해결되었는지 검사하고,

   해결되지 않았다면 1단계로 돌아가서 과정을 반복

거스름돈 문제

- 거스름돈으로 사용할 500원, 100원, 50원, 10원 짜리 동전이 무한히 존재한다고 가정
- 손님에게 거슬러 주어야 할 돈이 n원일 때 거슬러 주어야 할 동전의 최소 개수 구하기 단, n원은 항상 10의 배수

거스름돈 문제

- 동전의 최소 개수를 구하는 최적의 해를 빠르게 구하기 위해서는 가장 큰 동전 단위부터 거슬러 주면 된다!

```python
n = 1260  # 거스름돈
count = 0  # 동전의 수

# 큰 단위의 동전부터 차례대로 확인
coins = [500, 100, 50, 10]

for coin in coins:
    count += n // coin  # 해당 동전으로 거슬러 줄 동전의 개수
    n %= coin  # 해당 동전으로 거슬러주고 남은 거스름돈

print(count)
```

거스름돈 문제 - 정당성 분석

- 가장 큰 동전 단위부터 거슬러 주는 것이 최적의 해를 보장하는 이유는?

  -> 거스름돈으로 사용할 동전은 500원, 100원, 50원, 10원

  큰 단위가 항상 작은 단위의 배수이므로 작은 단위의 동전들을 종합해서 다른 해가 나올 수 없기 때문

- 그렇다면 800원을 거슬러 주는데, 단위가 500원, 400원, 100원이면?

그리디 알고리즘 - 분석

- 일반적인 상황에서 그리디 알고리즘이 최적의 해를 보장할 수 없을 때가 많다
- 그리디 알고리즘으로 얻은 해가 최적의 해가 되는 상황을 파악하고 이 알고리즘을 적절하게 적용할 수 있어야 한다.

그리디 알고리즘의 최적해 조건

- 탐욕적 선택 속성 (Greedy Choice Property)

  : 앞의 선택이 이후의 선택에 영향을 주지 않을 것

- 최적 부분 구조 (Optimal Substructure)

  : 문제에 대한 최종 해결 방안은 부분 문제에 대한 최적 문제 해결 방법으로 구성

그리디 알고리즘 정리

- 최적해 조건이 성립하지 않는다면 최적해에 가까운 근사 알고리즘으로 사용가능
- 계산 속도가 빠르기 때문에 실용적으로 사용
- 지역적(local)으로 최적인 해답이면서 최종적(global)으로도 최적인 해답을 찾는 해결 방식

그리디 알고리즘 백준 1931번 문제

회의실 배정

https://www.acmicpc.net/problem/1931

그리디 알고리즘 백준 1931번 문제

```python
import sys
input = sys.stdin.readline

n = int(input())  # 회의의 수
time = []
for i in range(n):
    time.append(list(map(int, input().split())))

# 최대 회의 수를 구하기 위해, 빨리 끝나는 순서대로 먼저 오름차순 정렬하고,
# 시작하는 시간으로 또 오름차순 정렬한다.
time.sort(key = lambda x: (x[1],x[0]))
```

```python
count = 0  # 사용할 수 있는 회의 최대 개수
end_t = 0 # 회의 끝나는 시간
for start, end in time:
    if start >= end_t:
        end_t = end
        count += 1

print(count)
```