GREEDY(탐욕) 알고리즘

19기 분석 우아라

<그래서 님 티어가?>

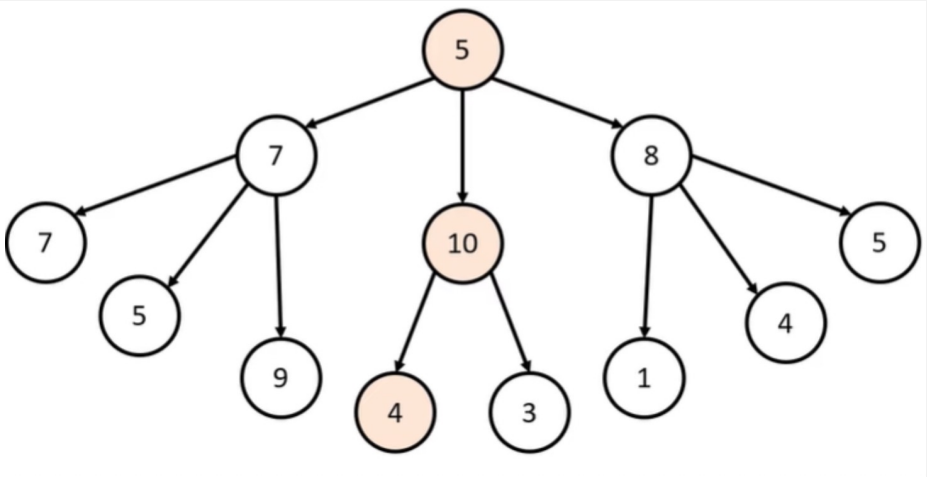
목차

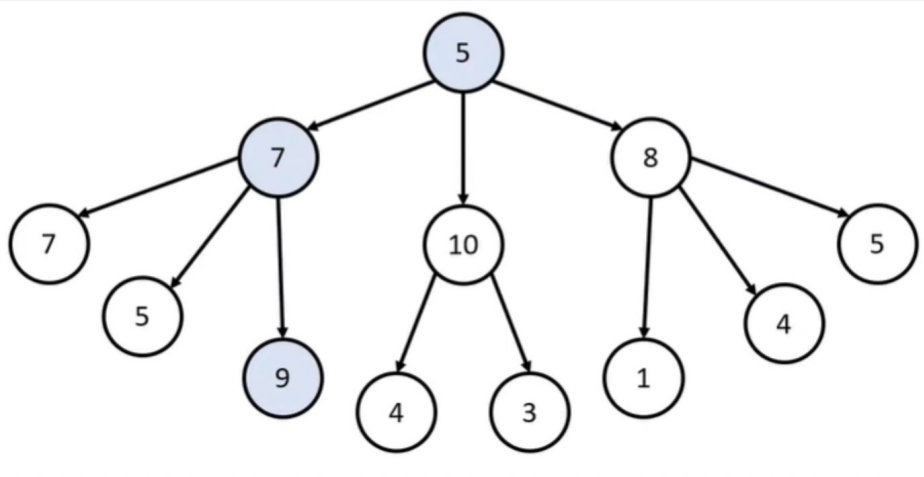
GREEDY 알고리즘이란?

- **Greedy** : '탐욕스러운, 욕심 많은'
- 현재 상황에서 **지금 당장의 최적의 상황**을 고르는 방법을 의미
- 최소한의 아이디어를 떠올릴 수 있는 능력을 요구
- **정당성 분석**이 중요

(단순히 가장 좋아 보이는 것을 반복적으로 선택해도 **최적의 해를 구할 수 있는지 검토**)

- 일반적인 상황에서 Greedy 알고리즘은 **최적의 해를 보장할 수 없을 때가 많음** (지역적)





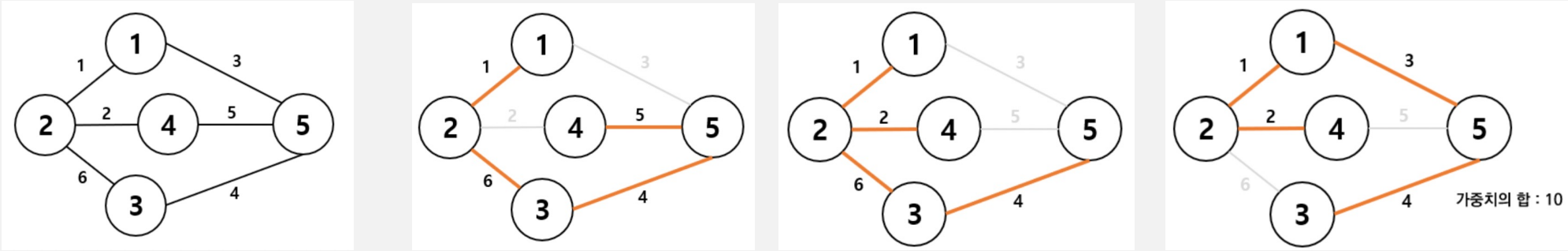
- 하지만 코딩테스트에서 대부분의 Greedy문제는 탐욕법으로 얻은 해가 최적의 해가 되는 상황에서 이를 추론할 수 있어야 풀리도록 출제

  (greedy 알고리즘을 적용할 수 있는 문제들은 **지역적으로 최적**이면서 **전역적으로 최적**인 문제들임)

- 이러한 **Greedy 알고리즘은** 기본적으로 (무조건 큰 경우대로, 무조건 작은 경우대로, 무조건 긴 경우대로, 무조건 짧은 경우대로.. 등등) 극단적으로 문제에 접근한다는 점에서 **정렬(Sort) 기법이 함께 사용**되는 경우가 많음
- 그 대표적인 예시가 **크루스칼(Kruskal) 알고리즘**으로, 모든 간선을 정렬한 이후에 짧은 간선부터 연결하는 최소 비용 신장 트리 알고리즘이 있음.

- **크루스칼 알고리즘?**

  : 그래프 내의 **모든 정점들을 가장 적은 비용으로 연결**하기 위해 사용



가중치 : 1+5+6+4 = **16**　　1+2+6+4 = **13**　　1+2+3+4 = **10**

→ **최소 신장 트리**(최소 비용 신장 트리)

GREEDY 알고리즘 문제 해결 방법

1. 선택 절차(Selection Procedure)

: 현재 상태에서의 최적의 해답을 선택한다.

2. 적절성 검사(Feasibility Check)

: 선택된 해가 문제의 조건을 만족하는지 검사한다.

3. 해답 검사(Solution Check)

: 원래의 문제가 해결되었는지 검사하고, 해결되지 않았다면 선택 절차로 돌아가 위의 과정을 반복한다.

GREEDY 알고리즘을 적용하려면

1. **탐욕적 선택 속성(Greedy Choice Property)** : 앞의 선택이 이후의 선택에 영향을 주지 않는다.
2. **최적 부분 구조(Optimal Substructure)** : 문제에 대한 최종 해결 방법은 부분 문제에 대한 최적 문제 해결 방법으로 구성된다.

- 이러한 조건이 성립하지 않는 경우에는 탐욕 알고리즘은 최적해를 구하지 못한다.
- 하지만, 이러한 경우에도 탐욕 알고리즘은 **근사 알고리즘으로 사용이 가능**할 수 있으며, 대부분의 경우 **계산 속도가 빠르기** 때문에 실용적으로 사용할 수 있다.

결론

- 항상 최적의 결과를 도출하는 것은 아니지만, 어느 정도 최적에 근사한 값을 빠르게 도출

→ 근사 알고리즘으로 사용 가능

- 언제나 최적해를 구할 수 있는 문제(매트로이드)가 있고, 이러한 문제에 탐욕 알고리즘을 사용해서 빠른 계산 속도로 답을 구할 수 있음 → 실용적

문제 1

- 500원, 100원, 50원, 10원짜리 동전으로 960원을 만들 때, 최적의 동전 개수는?

풀이 1

1. **선택 절차**

   거스름돈의 동전 개수를 줄이기 위해 현재 가장 가치가 높은 동전을 우선 선택한다.

2. **적절성 검사**

   1번 과정을 통해 선택된 동전들의 합이 거슬러 줄 금액을 초과하는지 검사한다.

   초과하면 가장 마지막에 선택한 동전을 삭제하고, 1번으로 돌아가 한 단계 작은 동전을 선택한다.

3. **해답 검사**

   선택된 동전들의 합이 거슬러 줄 금액과 일치하는지 검사한다.

   액수가 부족하면 1번 과정부터 다시 반복한다.

풀이 1

- 가장 가치가 높은 동전인 500원 1개를 먼저 거슬러 주고 잔액을 확인한 뒤, 이후 100원 4개, 50원 1개, 10원 1개의 순서대로 거슬러 준다. (답 : 7)

  → 매트로이드 문제

- 그런데 만약 800원을 거슬러 주어야 하는데 화폐 단위가 500원, 400원, 100원 이라면?

  → greedy 알고리즘에서는 500원 x 1, 100원 x 3 개가 최적의 해 이지만, 실제로는 400원 x 2 개가 최적의 해

코드 1

```
n = 960
count = 0

# 큰 단위의 화폐부터 차례대로 확인하기
array = [500, 100, 50, 10]

for coin in array:
    count += n//coin # 해당 화폐로 거슬러 줄 수 있는 동전의 개수 세기
    n %= coin # n = n % coin
print(count)
```

문제 2

- https://www.acmicpc.net/problem/11047
- 백준 11047 - 동전 0

동전 0

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	256 MB	96207	50360	39094	51.845%

문제

준규가 가지고 있는 동전은 총 N종류이고, 각각의 동전을 매우 많이 가지고 있다.

동전을 적절히 사용해서 그 가치의 합을 K로 만들려고 한다. 이때 필요한 동전 개수의 최솟값을 구하는 프로그램을 작성하시오.

입력

첫째 줄에 N과 K가 주어진다. ($1 \le N \le 10$, $1 \le K \le 100,000,000$)

둘째 줄부터 N개의 줄에 동전의 가치 A_i가 오름차순으로 주어진다. ($1 \le A_i \le 1,000,000$, $A_1 = 1$, $i \ge 2$인 경우에 A_i는 A_{i-1}의 배수)

출력

첫째 줄에 K원을 만드는데 필요한 동전 개수의 최솟값을 출력한다.

참고자료

- https://hanamon.kr/%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98-%ED%83%90%EC%9A%95%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98-greedy-algorithm/
- https://velog.io/@kyunghwan1207/%EA%B7%B8%EB%A6%AC%EB%94%94-%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98Greedy-Algorithm-%ED%83%90%EC%9A%95%EB%B2%95

감사합니다