Greedy Algorithm

개념

- 매 선택마다 **현재 상태에서 최적**이라 생각되는 것을 선택해 나가는 방법
  - 현재 상태에서 최적이 전체 범위에서 최적이라고 보장하지는 못함
  - 그렇기에, 이 전략은 지역적으로 최적인 선택이 전체 범위에서 최적인 문제들에게만 적용 가능

문제 조건 (필수 요소)

- **Greedy Choice Property**
  - 앞의 선택이 이후 선택에 영향을 주지 않음
- **Optimal Substructure Property**
  - 문제 전체에 대한 최적 해가 부분 문제에 대해서도 최적임
- 즉, **"원 문제의 최적해 = Greedy Choice + 하위 문제의 최적해"** 임을 증명해야 함

귀류법

- 명제의 결론이 부정이라고 가정했을 때, 모순이 생김을 보여 원래 명제가 참임을 증명
- 모든 n에 대해 $P(n)$이 참임을 증명
    - $P(n)$이 거짓이 되는 n이 있다고 가정
    - $P(n)$이 거짓이 되는 n이 있으면 모순이 일어나는 것을 보임
    - $P(n)$이 거짓이 되는 n이 없으므로, 모든 n에 대해 $P(n)$은 참

귀류법 - 예시

- $\sqrt{2}$가 유리수가 아님을 증명
  - $\sqrt{2}$가 유리수라고 가정 → $\sqrt{2} = \frac{b}{a}$로 둘 수 있다. $(a, b$는 서로소인 자연수)
  - $2a^2 = b^2$이므로 b^2은 2의 배수이다. b^2이 2의 배수이므로, b도 2의 배수이다.
    - 따라서 $b = 2b'$으로 둘 수 있다. $(b'$은 자연수)
  - $a^2 = \frac{1}{2}b^2 = 2b'^2$이므로 a^2은 2의 배수이다. a^2이 2의 배수이므로, a도 2의 배수이다.
  - 이는 $\sqrt{2}$가 유리수$(a, b$가 서로소)라는 가정에 모순이다.

보물 BOJ 1026

- 길이가 N인 음이 아닌 정수 배열 A와 B가 있고 함수 S를 아래와 같이 정의한다
- $S = A[0] \times B[0] + \cdots + A[N-1] \times B[N-1]$
- S의 값을 가장 작게 만들기 위해 A의 수를 재배열하여라 (B는 불가)
- S의 최솟값을 출력한다.

- 어떤 수끼리 짝짓는다면 그 합이 작아질까?

보물 BOJ 1026

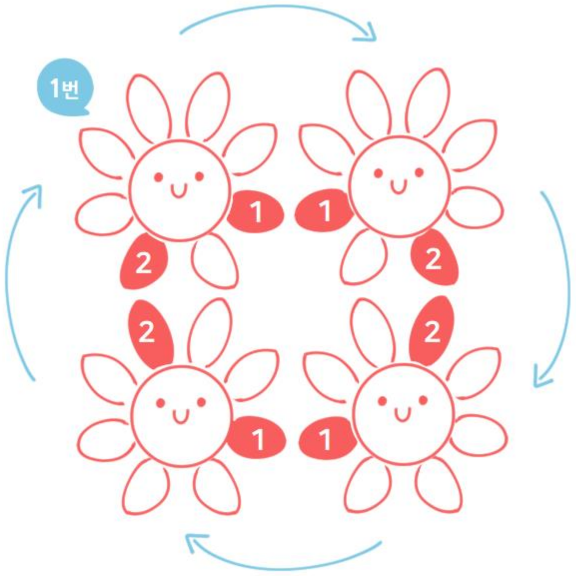
- A'과 B'이 각각 A, B가 오름차순, 내림차순 정렬되어 있는 배열이라 하자
  - $A'[0] < A'[1] < \cdots < \mathrm{A}'[\mathrm{N}-1]$
  - $B'[0] > B'[1] > \cdots > B'[N-1]$
- $S = \sum_{i=0}^{N-1} A'[i]B'[i]$가 최소일 것이다 (그리디)
- $N = 2$인 상황을 보자
  - $A'[0] \times B'[0] + A'[1] \times B'[1]$ (그리디)
  - $A'[0] \times B'[1] + A'[1] \times B'[0]$ (다른 최적해가 존재한다고 하자)

보물 BOJ 1026

- 그리디한 방식이 아닌 다른 최적해가 최솟값을 가진다고 가정
  - $A'[0] \times B'[0] + A'[1] \times B'[1] > A'[0] \times B'[1] + A'[1] \times B'[0]$ (양변을 $A'[0]$으로 나눔)
  - $B'[0] + \frac{A'[1]}{A'[0]} \times B'[1] > B'[1] + \frac{A'[1]}{A'[0]} \times B'[0]$ ($\frac{A'[1]}{A'[0]}$을 k라 하면, $k > 1$)
  - $B'[0] + kB'[1] > kB'[0] + B'[1]$ (이항을 해줌)
  - $(k-1)B'[1] > (k-1)B'[0]$ ($(k-1)$로 나눔)
  - $B'[1] > B'[0]$, 이는 B'이 내림차순 정렬되어 있다는 조건에 모순

문어 BOJ 21313

- 문어에게 여덟 개의 팔이 있고, 1~8번의 번호를 붙임
- 문어는 양 옆의 서로 다른 두 문어와 손을 맞잡아 원을 만듦
  - 서로 같은 번호의 손을 잡아야 함
  - 한 문어와 둘 이상의 손을 잡을 수 없음
  - 한 손으로 여러 문어의 손을 잡을 수 없음
- 1번 문어를 기준으로 시계방향 순으로 번호를 붙임
- 맞잡은 손의 번호를 이용해 길이 N의 수열을 만듦
  - 이렇게 만들 수 있는 수열 중 사전순으로 제일 앞서는 수열은?



문어 BOJ 21313

- 인접한 문어와 공통으로 사용하지 않은 가장 작은 번호의 손 맞잡기 (그리디)
  - 해당 방식으로 만들어진 수열을 A라고 하자
- 공통으로 사용하지 않은 가장 작은 번호의 손이 아닌 다른 손을 맞잡기 (다른 최적해)
  - 해당 방식으로 만들어진 수열을 B라고 하자
- A와 B가 달라지는 지점 i가 존재
  - $A[1] = B[1],\ A[2] = B[2],\ \ldots,\ A[i-1] = B[i-1]$이고, $A[i] < B[i]$
  - 수열 A가 수열 B보다 사전 순으로 앞서서 모순이 발생

접근법?

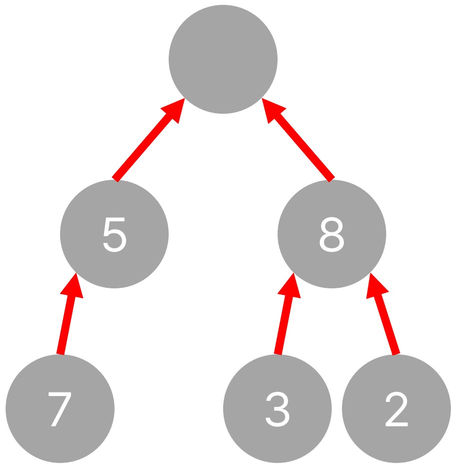
- 알고리즘의 이름과 동일한 방식의 접근
  - 어떤 상황이든 이렇게 선택하는 것이 최선이라는 믿음으로
  - 물론, 이렇게 나온 풀이에 대한 증명이 필요
- 전체 문제를 부분 문제로 쪼개었던 DP와 유사하다고 생각할 수 있음
  - 그리디는 DP와 달리 선택되지 않은 하위 문제들은 답에 영향을 주지 않음

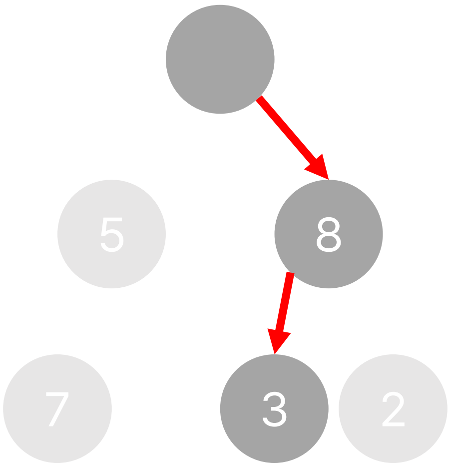
DP와 그리디

- 리프 노드까지 내려가면서 노드에 적힌 수의 합을 최대화 하는 경로를 구하려고 할 때
  - DP는 왼쪽과 같이 리프 노드에서부터 부분 문제를 해결하면서 진행됨
  - 그리디는 지역적으로 최적인 선택을 하였지만, 전역적으로 최적인 5 – 7 경로를 찾지 못함

DP와 그리디

- 리프 노드까지 내려가면서 노드에 적힌 수의 합을 최대화 하는 경로를 구하려고 할 때
  - DP와 비교하였을 때, 그리디의 연산 횟수가 훨씬 적음
  - DP는 모든 경우를 고려해야 하지만, 그리디는 그 순간 최선의 선택만 진행하면 됨





그리디 알고리즘

- Minimum Coin Change Problem
- Fractional Knapsack Problem
- Shortest Job First (SJF) Scheduling
  - Exchange Argument
- Activity Selection Problem

동전 0 BOJ 11047

- 동전의 종류가 N개이고, 동전들의 합을 K로 만들려고 할 때, 동전을 최소로 사용
  - (조건) 동전은 서로 **약수 / 배수** 관계여야 함
- **Minimum Coin Change Problem**
- 어떤 가격의 동전부터 사용해야 할까?

동전 0 BOJ 11047

- 비싼 동전부터 최대한 많이 사용
  - 가격 내림차순으로 동전을 정렬 → $C[i]$ (i번째로 비싼 동전의 금액)
  - 위 가격의 동전을 사용한 개수를 저장한 배열 중 그리디한 방식을 A, 다른 최적해 방식을 B라 하자

  - A와 B가 사용한 동전의 개수가 달라지는 지점 i가 존재
    - $A[i] > B[i]$ (비싼 동전부터 사용하는 전략에 의해)
    - $C[i] \times (A[i] - B[i])$ 만큼의 금액 차이를 $j(i < j)$지점에서 채워야 함
    - $C[i] \times (A[i] - B[i]) = C[j] \times (B[j] - A[j])$에서 $C[i] > C[j]$이므로 $(A[i] - B[i]) < (B[j] - A[j])$
      - 즉, j지점에서 동전 개수의 차이가 더 커야 함(B에서 더 많은 동전을 사용해야 함)

동전 0 BOJ 11047

- 동전의 종류가 N개이고, 동전들의 합을 K로 만들려고 할 때, 동전을 최소로 사용
  - (조건) 동전은 서로 **약수 / 배수** 관계여야 함
  - 조건이 어긋난다면?
    - 50, 30, 5원 짜리로 60원을 만들어보자
    - (그리디) 50, 5, 5 → 3개
    - (최적해) 30, 30 → 2개

동전 0 BOJ 11047

- 동전의 종류가 N개이고, 동전들의 합을 K로 만들려고 할 때, 동전을 최소로 사용
  - (조건) 동전은 서로 **약수 / 배수** 관계여야 함
  - 조건이 어긋난다면?
    - 50, 30, 5원 짜리로 60원을 만들어보자
    - (그리디) 50, 5, 5 → 3개
    - (최적해) 30, 30 → 2개
    - **BOJ 2293 동전 1 (DP)**

Fractional Knapsack Problem

- Knapsack Problem (배낭 문제)
  - 물건의 무게와 가치가 주어지고, 담을 수 있는 배낭의 최대 용량이 주어지면 배낭에 넣을 수 있는 물건의 가치합의 최댓값을 구하는 문제
- Fractional Knapsack Problem (부분 배낭 문제)
  - 물건을 잘라서 담을 수 있게 됨
  - 가치가 K인 물건의 무게를 W라고 할 때, 무게가 $w, W - w$인 물건으로 분할 가능
    - 가치는 $\frac{K}{W} \times w, \frac{K}{W} \times (W - w)$로 분할

Fractional Knapsack Problem

- 비싸면서 가벼운, 무게 당 가치가 높은 물건부터 넣기
- 무게 당 가치가 높은 순으로 정렬
  - 물건의 무게가 배낭의 용량보다 작거나 같은 경우, 모두 담으면 됨
  - 크다면, 남은 배낭의 용량만큼 분할하여 담으면 됨

Fractional Knapsack Problem

- 물건을 무게 당 가치(효율)가 높은 순으로 정렬
  - 해당 방식으로 정렬한 가치 배열을 C라고 하자
  - 각 물건을 가져간 무게를 저장한 배열을 그리디한 방식은 A, 다른 최적해의 방식은 B라고 하자

Fractional Knapsack Problem

- A와 B가 달라지는 지점 i가 존재
  - $A[i] > B[i]$ (효율이 높은 것부터 최대한 많이 가져가는 전략에 의해)
  - 남은 공간인 $A[i] - B[i]$만큼을 $j(i < j)$지점에서 채운다고 하자
  - $A[i] - B[i] = B[j] - A[j]$이고, 가치는 $C[i](A[i] - B[i]) > C[j](B[j] - A[j])$가 된다
  - $C[i] \times A[i] + C[j] \times A[j] > C[i] \times B[i] + C[j] \times B[j]$
  - A보다 비싼 최적해는 존재하지 않음

Shortest Job First (SJF) Scheduling

- 프로세서가 사용 가능할 때 실행 시간이 가장 짧은 작업부터 할당
  - 평균 대기 시간을 최소화 → 대기 시간 합 최소

ATM BOJ 11399

- i번째 사람은 P_i시간 동안 점유
- 대기 시간의 합이 최소가 되도록 하려면?
  - 앞의 SJF Scheduling을 이용
  - 실행 시간이 가장 짧은 것부터 처리

ATM BOJ 11399

- P_i가 가장 작은 사람부터 처리하는 것이 최적
  - P는 오름차순으로 정렬된 상태라고 하자 $(P_i < P_j(i < j))$
  - 두 사람 P_k, P_{k+1}중 먼저 처리해야 하는 사람을 결정하자 (P_{k-1}까지는 처리된 상태)

  - P_k, P_{k+1}순서로 처리
    - P_k 대기시간: $\sum_{i=1}^{k-1} P_i$
    - P_{k+1} 대기시간: $\sum_{i=1}^{k-1} P_i + P_k$

$P_1, \dots, P_{k-1}$	P_k	P_{k+1}	$P_{k+2}, \dots, P_n$

  - P_{k+1}, P_k순서로 처리
    - P_{k+1} 대기시간: $\sum_{i=1}^{k-1} P_i$
    - P_k 대기시간: $\sum_{i=1}^{k-1} P_i + P_{k+1}$

$P_1, \dots, P_{k-1}$	P_{k+1}	P_k	$P_{k+2}, \dots, P_n$

  - 공통된 부분인 $\sum_{i=1}^{k-1} P_i$ 를 제외하고 비교했을 때, $P_k < P_{k+1}$이므로 P_k, P_{k+1}순서로 처리하는 것이 최적

Exchange Argument

- Shortest Job First (SJF) Scheduling의 증명 방법을 일반화 한 것
  - 가상의 최적해를 설정하고, 이 최적해를 수정하며 결과가 나빠지지 않음을 보임
    - 결국 그 결과가 최적해가 되는 것
  - 인접한 두 원소의 순서를 결정할 수 있으면, 버블 정렬을 이용해 전체 원소의 순서 결정 가능

- 버블 정렬
  - 서로 인접한 두 원소를 검사하여 정렬하는 알고리즘
    - 인접한 2개의 원소를 비교하여 정렬되어 있지 않으면 서로 교환

Activity Selection Problem

- N개의 활동이 주어졌을 때, 한 사람이 수행할 수 있는 최대 활동 수를 선택하는 문제

회의실 배정 BOJ 1931

- 한 개의 회의실을 사용하고자 하는 N개의 회의가 있음
- 각 회의는 시작 시간과 종료 시간이 주어지고, 해당 기간 동안 회의실을 점유
- 진행 가능한 회의의 최대 개수를 구하는 문제

- 어떤 회의부터 배정해야 할까?

회의실 배정 BOJ 1931

- 한 개의 회의실을 사용하고자 하는 N개의 회의가 있음
- 각 회의는 시작 시간과 종료 시간이 주어지고, 해당 기간 동안 회의실을 점유
- 진행 가능한 회의의 최대 개수를 구하는 문제

- 어떤 회의부터 배정해야 할까?
  - 시작 시간이 빠른 것부터? → 0부터 끝까지 차지하는 회의가 있으면?
  - 시작 시간이 빠른 것 중에서 종료 시간이 가장 빠른 것부터? → 위의 경우와 마찬가지
  - 종료 시간이 빠른 것부터? → Why?

회의실 배정 BOJ 1931

- 그리디한 방식에 의하면 종료 시간이 가장 빠른 회의를 포함해야 함
  - 종료 시간이 가장 빠른 회의 m을 포함하지 않는 최적해 O'이 있다고 하자
  - O'에서 종료 시간이 가장 빠른 회의 m'를 제거하고 m을 추가한 해를 O라고 하자
  - O에서 겹치는 회의는 존재하지 않는다
  - O에서 m 다음 회의까지의 간격은 O'에서 m' 다음 회의까지의 간격보다 크다
  - 모든 최적해 O'은 m을 포함하도록 바꿀 수 있다.

회의실 배정 BOJ 1931

- 종료 시간이 가장 빠른 회의 m을 포함하지 않는 최적해 O'이 있다고 하자
- O'에서 종료 시간이 가장 빠른 회의 m'를 제거하고 m을 추가한 해를 O라고 하자

MEXchange BOJ 30462

- 길이가 N인 순열 A가 주어졌을 때, 수열 B를 다음과 같이 정의하자
  - $B_i = MEX(\{A_1, A_2, \dots, A_i\})\ (1 \le i \le N)$
- 길이가 N인 수열 B가 주어질 때, 순열 A를 구해보자
- $MEX(S)$는 집합 S에 포함되지 않는 가장 작은 양의 정수이다.
  - $MEX(\{1, 2, 5\}) = 3$
  - $MEX(\{2, 3, 4\}) = 1$
  - 이 문제에서 정의한 MEX는 그 값으로 0이 나올 수 없음에 주의하자

MEXchange BOJ 30462

- 수열 B의 i번째 원소가 k이다
  - 어떤 제약 조건이 생길까?

MEXchange BOJ 30462

- 수열 B의 i번째 원소가 k이다
  - 어떤 제약 조건이 생길까?
    - 순열 A의 1번째 원소부터 $i-1$번째 원소에 1부터 $k-1$까지 나왔으며, k는 나오지 않았다

MEXchange BOJ 30462

- 수열 B의 i번째 원소랑 $i+1$번째 원소랑 다르다
  - 어떤 제약 조건이 생길까?

MEXchange BOJ 30462

- 수열 B의 i번째 원소랑 $i+1$번째 원소랑 다르다
  - 어떤 제약 조건이 생길까?
    - 순열 A의 $i+1$번째 원소는 수열 B의 i번째 원소와 같은 수이다

MEXchange BOJ 30462

- 확정되지 않은 자리는 어떻게 채워야 할까?
    - $B_i \neq B_{i+1}$인 경우, $A_{i+1} = B_i$로 확정

MEXchange BOJ 30462

- 확정되지 않은 자리는 어떻게 채워야 할까?
  - $B_i \neq B_{i+1}$인 경우, $A_{i+1} = B_i$로 확정
  - 남은 수들을 오름차순으로 배치?

MEXchange BOJ 30462

- 확정되지 않은 자리는 어떻게 채워야 할까?
  - $B_i \neq B_{i+1}$인 경우, $A_{i+1} = B_i$로 확정
  - 남은 수들을 오름차순으로 배치?
- 제약 조건을 만족하는가?
  - $B_i = k$인 경우, 순열 A의 i번째 앞의 원소들에서 1부터 $k-1$까지 모두 나오고, k는 나오지 않았다

숫자의 신 BOJ 1422

- K개의 자연수 중 N개의 수를 붙여서 만들 수 있는 가장 큰 수를 구하여라
  - 같은 수를 여러 번 이용해도 된다
  - 단, 모든 수는 적어도 한 번은 이용되어야 한다
  - 자연수는 중복되어 들어올 수 있고 이 경우는 각 수가 적어도 입력으로 주어진만큼 사용되어야 한다
- 2, 3, 7을 가지고 있고 4개의 수를 뽑아야 한다면 7732가 가장 큰 수이다

숫자의 신 BOJ 1422

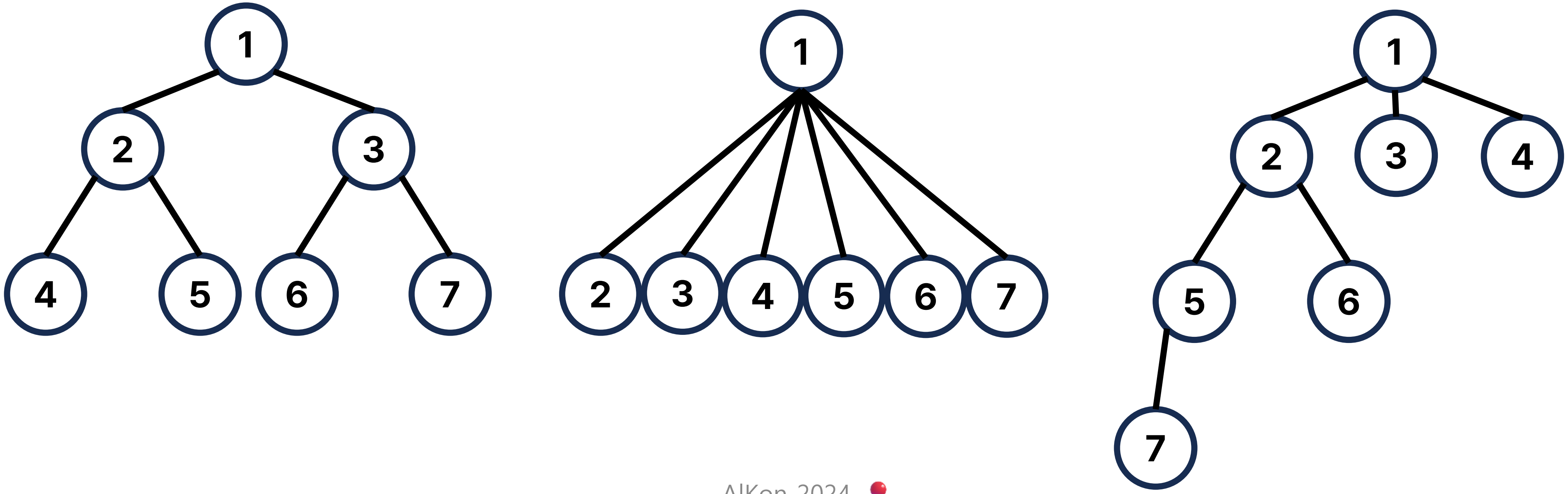
- Exchange Argument를 이용하자!
- 두 수가 주어졌을 때, 어떻게 나열해야 더 큰 수인지 확인하자
    - 991, 99에서 어떤 수가 먼저 나와야 하는가?
        - **991**99
        - 99**991**
    - 인접한 두 수 중 무엇이 먼저 나와야 더 큰 수를 만들 수 있을 지에 대한 순서 관계를 정할 수 있다.
        - Exchange Argument에 따라 증명할 수 있다
- 즉, 이에 따라 정렬한 후에
    - 맨 앞에 나와야 하는 수를 $K-(N-1)$번
    - 나머지를 $(N-1)$번 사용하면 된다

우정은 BFS처럼, 사랑은 DFS처럼 BOJ 30466

- 1부터 N까지 번호가 매겨진 N개의 정점으로 이루어진 트리를 구하고자 한다
- 1번 정점부터 깊이 우선 탐색과 너비 우선 탐색을 각각 시행했을 때
  - d_i = 깊이 우선 탐색(DFS)에서 i번 정점을 방문한 순서
  - b_i = 너비 우선 탐색(BFS)에서 i번 정점을 방문한 순서
- $\sum_{i=1}^{N} |d_i - b_i|$ 를 최대로 하는 트리를 만들어 보자
- 순회 후보가 여럿인 경우에는 번호가 더 작은 정점을 먼저 방문한다

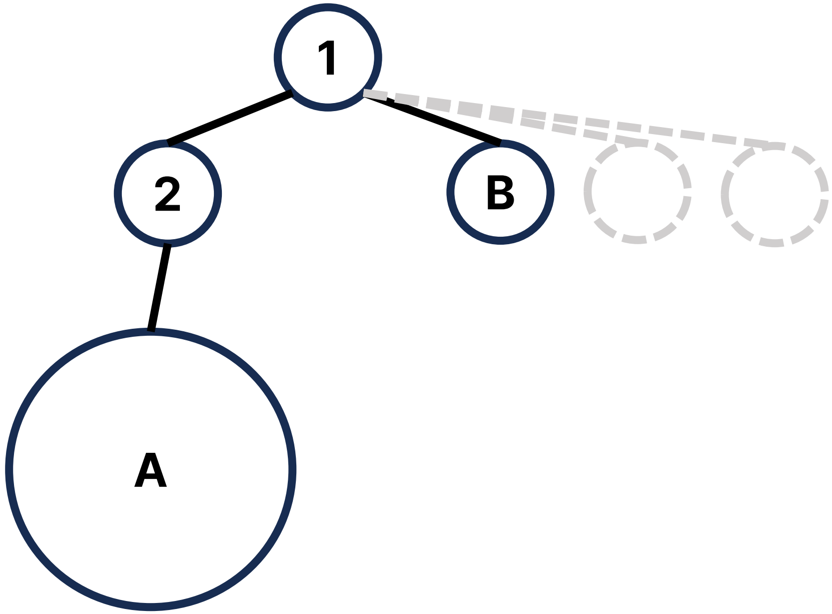
우정은 BFS처럼, 사랑은 DFS처럼 BOJ 30466

- 우선 7개의 정점으로 트리를 만들어보자
- bfs 순서대로 정점의 번호를 붙인다고 하자 (왼쪽부터)



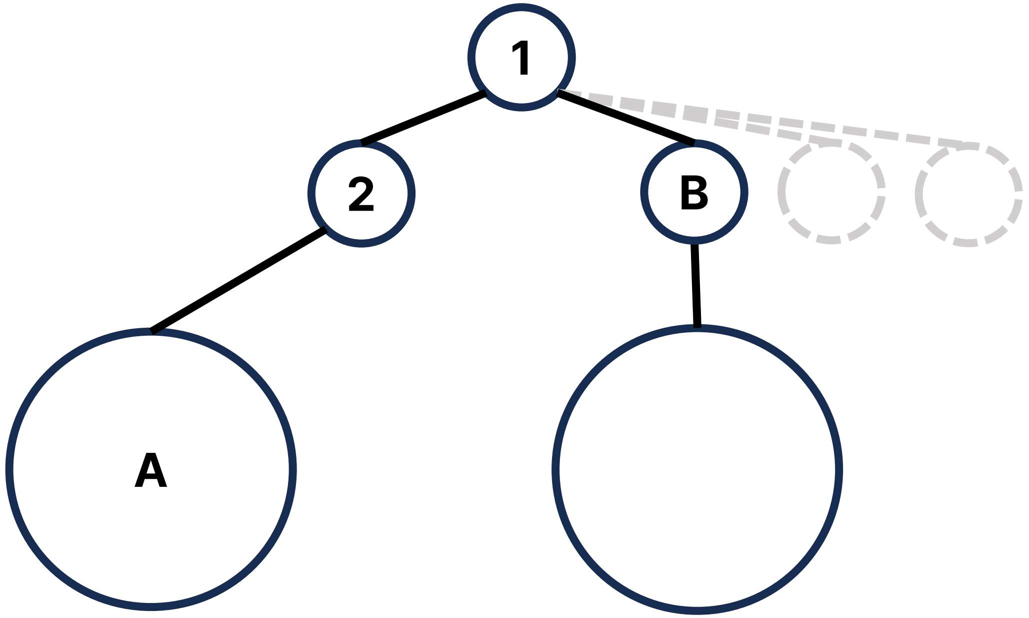
우정은 BFS처럼, 사랑은 DFS처럼 BOJ 30466

- 1, 2번 정점까지는 bfs, dfs의 차이가 존재하지 않는다
- bfs는 B를 다 거치고 A로 가고, dfs는 A를 다 거치고 B로 간다



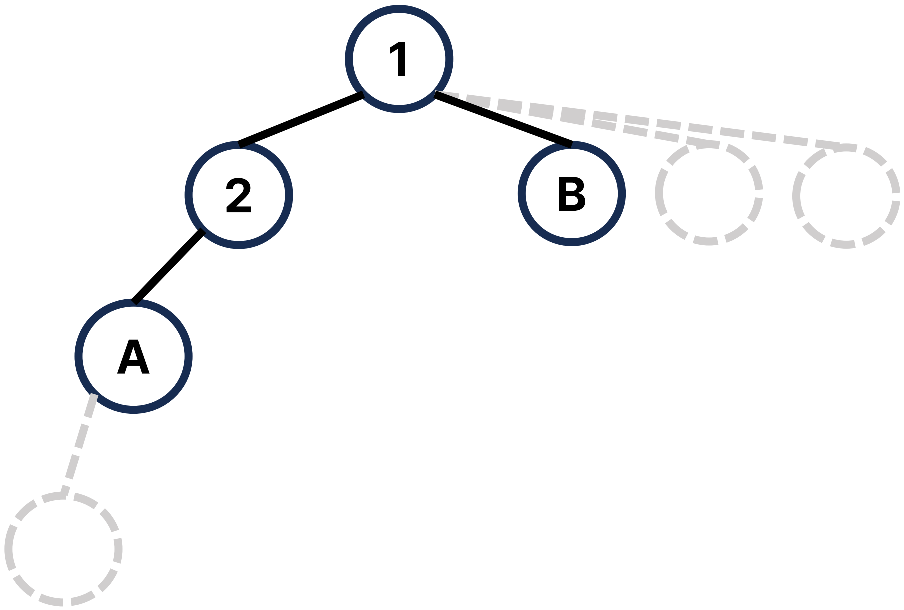
우정은 BFS처럼, 사랑은 DFS처럼 BOJ 30466

- bfs는 2, B, A 순으로, dfs는 2, A, B 순으로 간다
  - A에 X개의 정점이 있으면 B의 첫 번째 정점은 dfs로 탐색 시 X+3번째에 탐색하게 된다
  - B에 Y개의 정점이 있으면 A의 첫 번째 정점은 bfs로 탐색 시 Y+3번째에 탐색하게 된다



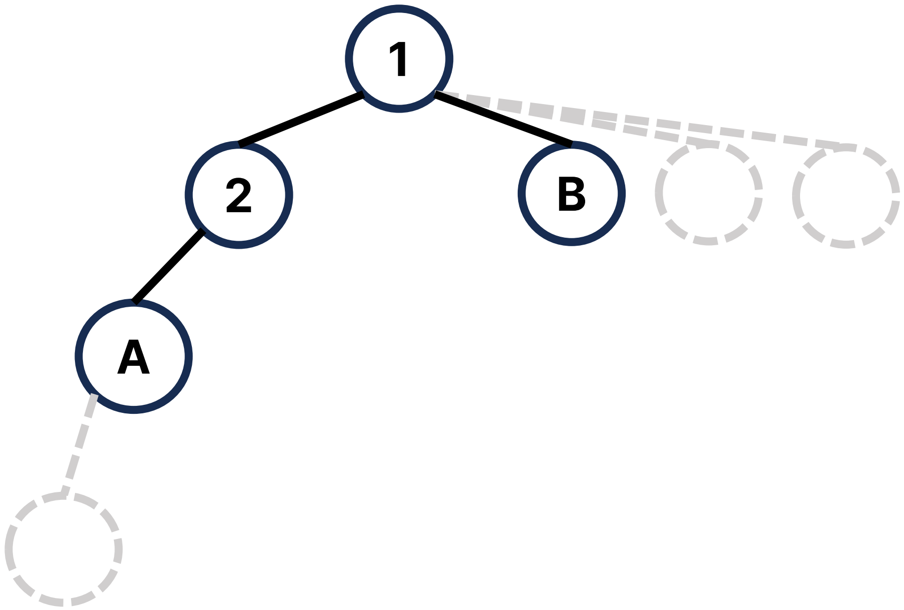
우정은 BFS처럼, 사랑은 DFS처럼 BOJ 30466

- 대강 이런 식으로 배치하면 차이가 최대가 될 것 같음



우정은 BFS처럼, 사랑은 DFS처럼 BOJ 30466

- 한 번 직접 증명해보자
  - https://solved.ac/arena/10/editorial M번



연습 문제

1026	: 보물
21313	: 문어
11047	: 동전 0
11399	: ATM
1931	: 회의실 배정
30462	: MEXchange
1422	: 숫자의 신
30466	: 우정은 BFS처럼, 사랑은 DFS처럼

References

- https://github.com/justiceHui/SSU-SCCC-Study
- https://www.cs.cornell.edu/courses/cs482/2007su/exchange.pdf