구현 유형 푸는 방법

1. **규칙을 Pseudo 형식으로 작성한다.**
    - 경우에 : If문
    - 돌아간다 : 반복문
2. **한줄 한줄을 함수형식으로 작성한다.**
3. **함수 명칭을 직관적으로 네이밍한다.**
    - 값을 쓰거나 읽어오는 함수의 Prefix : Get/Set
    - Boolean 함수의 Prefix : Is/Check

알고리즘 풀이 방법

1. **문제를 완전히 이해하고 유형을 파악한다 (DFS,BFS 유형/구현 유형)**
2. **적합한 알고리즘/자료구조 선정**
3. **시간 복잡도 계산 (반복문/경우의 수)**
    - C++/Java 기준 1억 당 1초로 계산

구현 유형 풀이 예시

14503번: 로봇청소기

로봇 청소기가 작동을 시작한 후 작동을 멈출 때까지 청소하는 칸의 개수를 출력한다.

1. 구현

1. 현재 칸이 아직 청소되지 않은 경우, 현재 칸을 청소한다.
2. 현재 칸의 주변 4칸 중 청소되지 않은 빈 칸이 없는 경우,
    1. 바라보는 방향을 유지한 채로 한 칸 후진할 수 있다면 한 칸 후진하고 1번으로 돌아간다.
    2. 바라보는 방향의 뒤쪽 칸이 벽이라 후진할 수 없다면 작동을 멈춘다.
3. 현재 칸의 주변 4칸 중 청소되지 않은 빈 칸이 있는 경우,
    1. 반시계 방향으로 90° 회전한다.
    2. 바라보는 방향을 기준으로 앞쪽 칸이 청소되지 않은 빈 칸인 경우 한 칸 전진한다.
    3. 1번으로 돌아간다.

2. Peudo Code

```
while True:
    # 조건 1
    if 현재 칸이 아직 청소되지 않은 경우 :
        현재 칸을 청소한다.
    # 조건 2
    if 현재 칸의 주변 4칸 중 청소되지 않은 빈 칸이 없는 경우 :
        if 바라보는 방향을 유지한 채로 한 칸 후진할 수 있다면 :
            한 칸 후진한다.
        else:
            작동을 멈춘다.
    # 조건 3
    else:
        반시계 방향으로 90° 회전한다.
        if 바라보는 방향을 기준으로 앞쪽 칸이 청소되지 않은 빈 칸인 경우:
            한 칸 전진한다.
```

3. 각 줄의 함수 구현

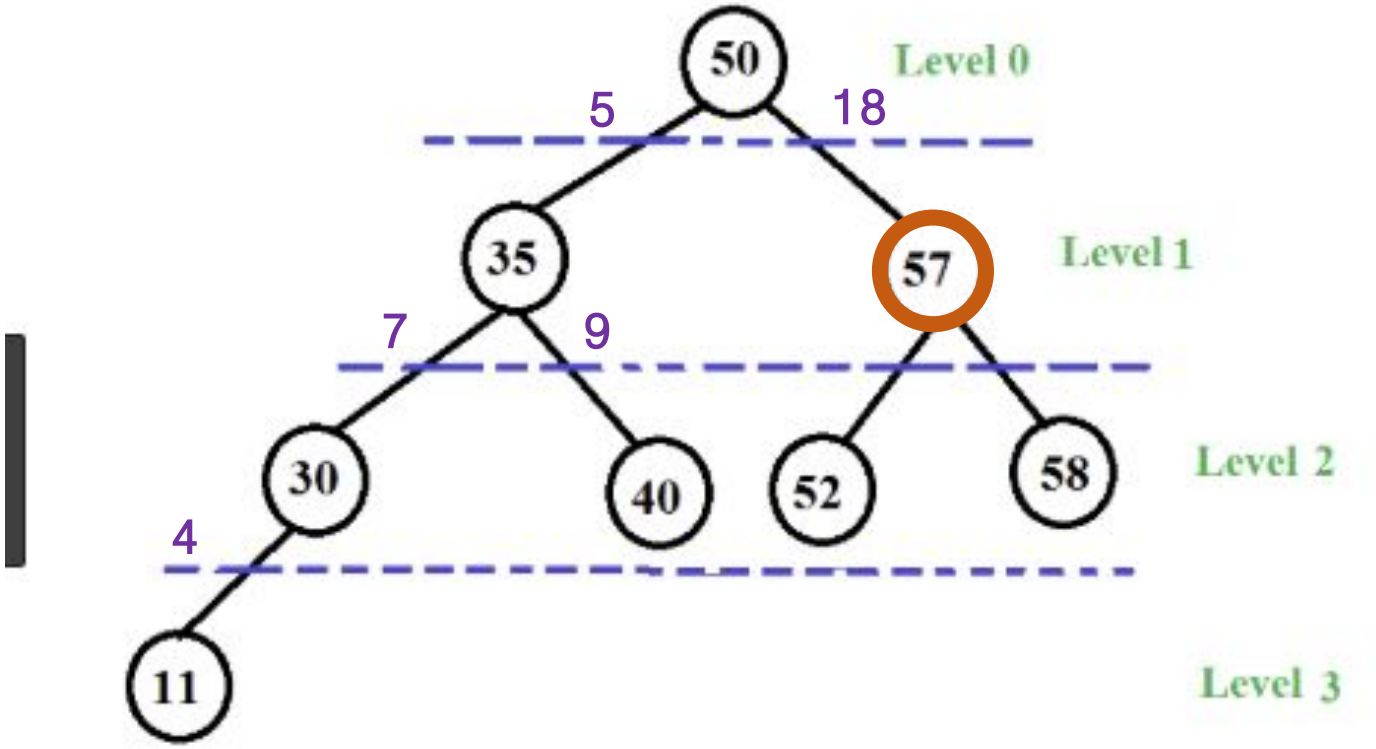
```
def check_position_is_not_clean(self, x, y):
    return (self.condition[x][y] == RoomConditionConst.ROOM) and (not self.is_cleaned[x][y])
```

백트래킹 (DFS, BFS)

조건

문제

Leaf 노드에 도달했을 때의 Cost를 계산하고, Minimum Cost를 구하라



Node	Cost
50	0
35	5
30	12
40	14
11	16
57	18
52	
58	

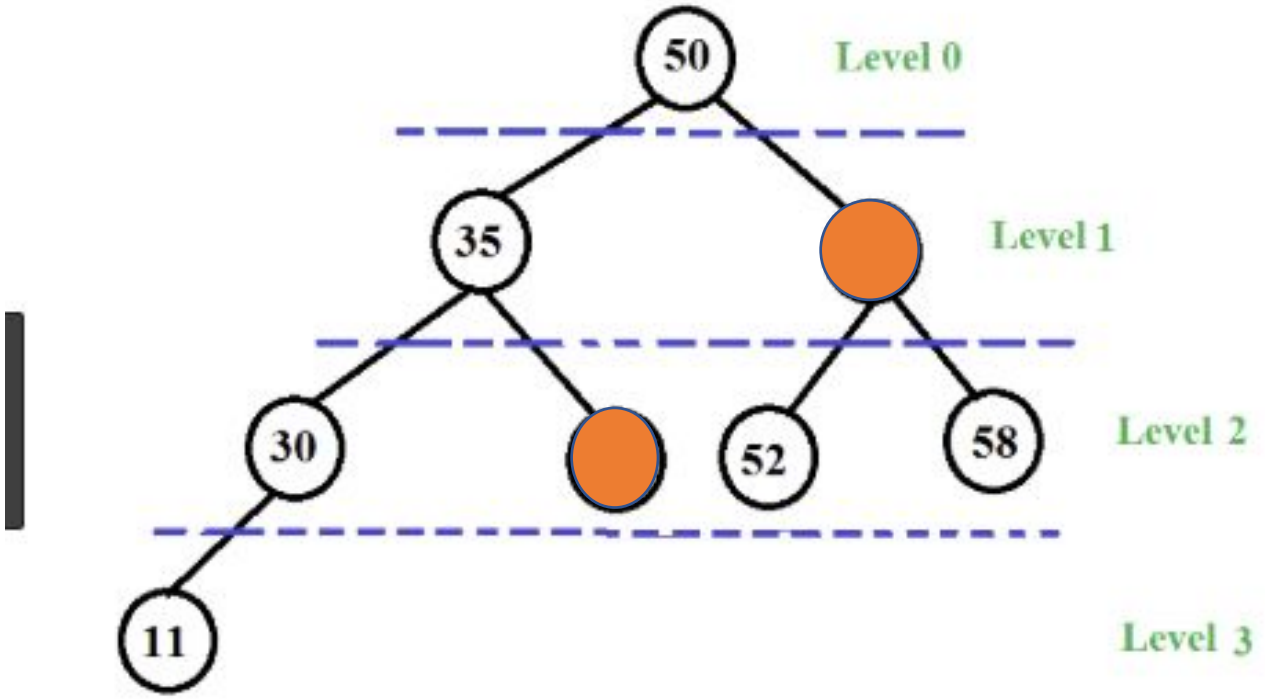
Minimum cost : 16

하위 Level의 노드들이 해조건이 되지 않음을 알고 가지치기를 할 수 있을 때에만 완전탐색 대신에 백트래킹을 사용한다.

BFS의 최소 조건

13460번: 구슬탈출2

최소 몇 번 만에 빨간 구슬을 구멍을 통해 빼낼 수 있는지 출력한다. 만약, 10번 이하로 움직여서 빨간 구슬을 구멍을 통해 빼낼 수 없으면 -1을 출력한다.



- **해조건** : 빨간 구슬을 구멍을 통해 빼낼 수 있다
- **최소조건** : 최소 몇 번 만에

최소조건을 만족하는 경우에는 DFS보다 BFS를 사용해야 한다