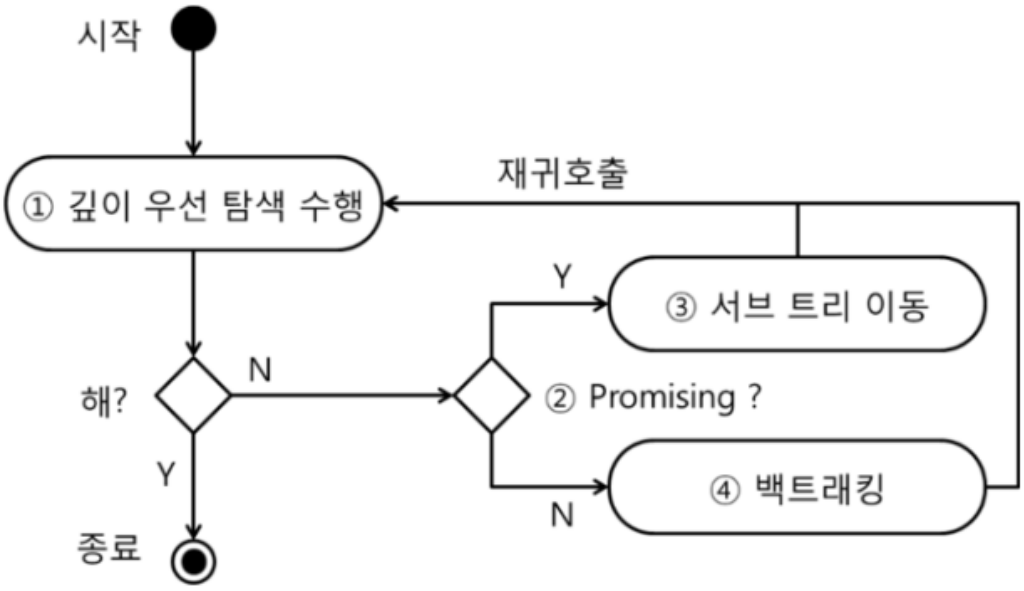
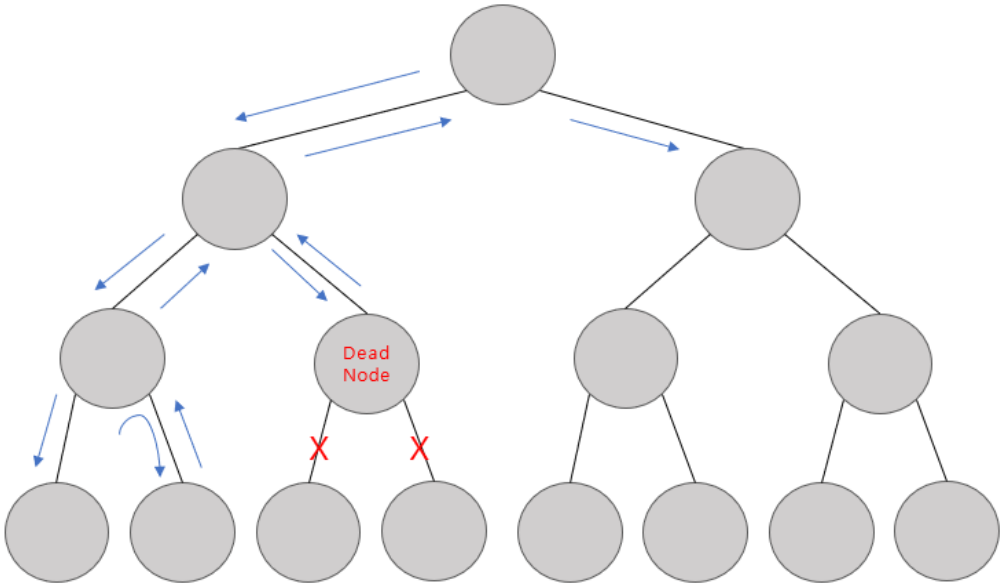
3주차 : 백트래킹(Back Tracking)

백트래킹 알고리즘

> **백트래킹** : 해를 찾는 도중 해가 아니어서 막히면 되돌아가서 다시 해를 찾아가는 기법.
> **완전 탐색의 아이디어에서 불필요한 분기를 가지치기** 하는 것으로, 정답을 도출하기 전 탐색 과정 중에 정답이 될 수 없는 조건에 해당된다면 가지치기 하여 효율을 높일 수 있음.

- 깊이우선탐색(DFS)을 진행하면서 조건을 확인하여 해당 노드가 유망하지 않으면 가지치기(더 탐색하지 않음)
- 일반적으로 재귀의 형태로 구현, 아래 3가지 내용이 중요.
  - Depth - 재귀 진행동안 사용될 깊이를 매개변수로 넣기
  - 가지치기 - 유망하지 않은 노드 더 탐색 하지 않음
  - 종료조건 - 재귀호출을 종료하는 조건
- 백트래킹 알고리즘 Flow



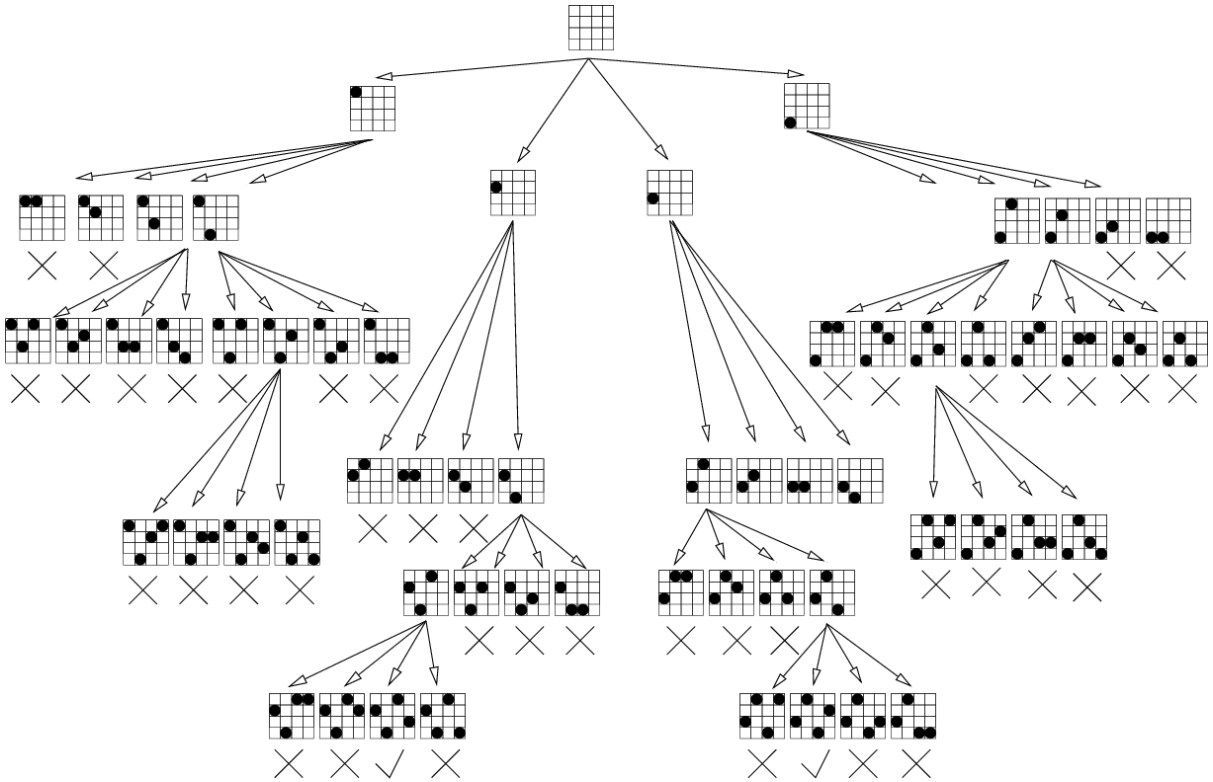


예제 및 풀이

N-Queen (Gold5) - https://www.acmicpc.net/problem/9663

- 크기가 N × N인 체스판 위에 퀸 N개를 서로 공격할 수 없게 놓는 방법의 수를 구하는 대표적인 백트래킹 문제
- 가지치기 조건
  - 같은 열에 있을 수 없음

- 같은 행에 있을 수 없음
- 두 대각선 방향에 있을 수 없음



▼ 코드

```
n = int(input())
total = 0
col = [0] * n

def chk(level):
    for i in range(level):
        # 대각선 or 같은 라인 여부 체크 / col[i]가 X좌표, i가 Y좌표를 의미.
        if col[i] == col[level] or abs(col[level] - col[i]) == level - i:
            return False
    return True


def nqueen(x):
    global total

    if x == n:
        total += 1
    else:
        for i in range(n):
            col[x] = i  # 해당 위치에 퀸을 배치
            if chk(x):  # 참이면 다음 행의 퀸 배치, 아니면 다른 위치로 퀸 배치 변경
                nqueen(x + 1)
```

```
nqueen(0)
print(total)
```

추가 문제

- 스도쿠 (**Gold 4**) - https://www.acmicpc.net/problem/2580

  ▼ 코드

```
sudoku = [list(map(int, input().split())) for _ in range(9)]
chk_row = [[False] * 10 for _ in range(10)]  # 행에 숫자(1~9)가 존재하는지 여부 확인
chk_col = [[False] * 10 for _ in range(10)]  # 열에 숫자(1~9)가 존재하는지 여부 확인
chk_square = [[False] * 10 for _ in range(10)]  # 정사각형(3x3)에 숫자(1~9)가 존재
하는지 여부 확인


def square(x, y):
    return (x // 3) * 3 + (y // 3)


def make_sudoku(z):
    if z == 81:
        for i in range(9):
            print(*sudoku[i])
        exit(0)
    x = z // 9
    y = z % 9

    if sudoku[x][y]:
        make_sudoku(z + 1)
    else:
        for i in range(1, 10):  # 0인 위치에 들어갈 숫자 찾기
            #  해당 열, 행, 정사각형 안에 i가 없으면 i를 넣는다
            if chk_row[x][i] == 0 and chk_col[y][i] == 0 and chk_square[squar
e(x, y)][i] == 0:
                chk_row[x][i] = chk_col[y][i] = chk_square[square(x, y)][i] =
True
                sudoku[x][y] = i
                make_sudoku(z + 1)
                # 숫자 i가 아니면 백트래킹
                sudoku[x][y] = 0
                chk_row[x][i] = chk_col[y][i] = chk_square[square(x, y)][i] =
False


for i in range(9):
    for j in range(9):
        if sudoku[i][j]:
            chk_row[i][sudoku[i][j]] = True
            chk_col[j][sudoku[i][j]] = True
            chk_square[square(i, j)][sudoku[i][j]] = True
```

```
make_sudoku(0)
```

- 비숍 (**Gold 1**) - https://www.acmicpc.net/problem/1799

참고

- https://chanhuiseok.github.io/posts/algo-23/
- https://covenant.tistory.com/123
- https://veggie-garden.tistory.com/24
- http://blog.skby.net/585/
- http://sooyoung32.github.io/dev/2016/03/14/n-queen-algorithm.html

➡️ 2주차 : 위상 정렬(Topological sorting)