

chatGPT와 함께 하는

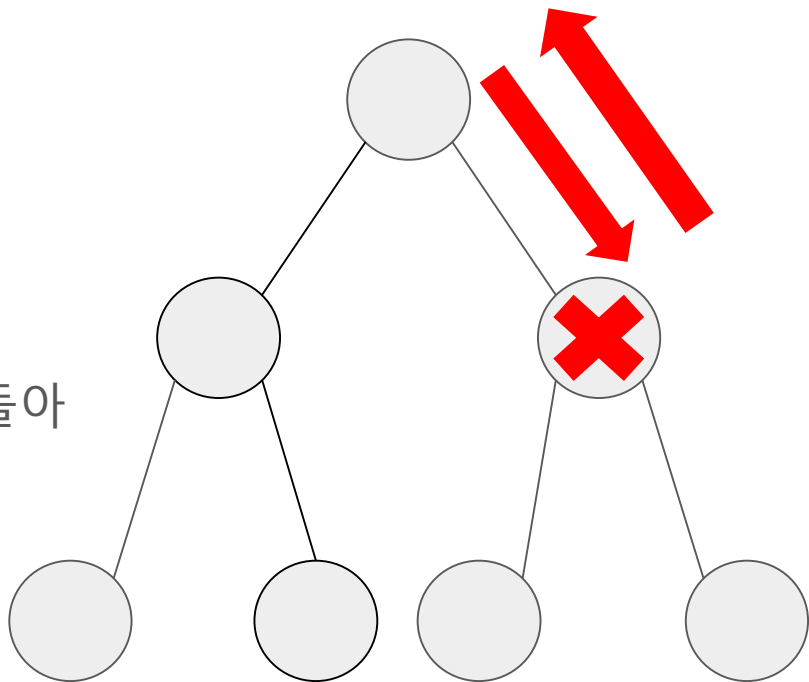
# 알고리즘 백트래킹

# 목차

- 백트래킹이란?
- 백트래킹 문제의 특징
- 백트래킹 문제 유형 (N-Queens | 스도쿠 | 조합 | 그래프 | 문자열)
- 백트래킹 대표 문제 풀이 (N-Queens)
- 참고 > 백트래킹 문제를 파악하는 방법 / 백트래킹의 기법 / DP와의 차이

# 백트래킹이란?

- 그래프/트리의 모든 원소 (노드)를  
완전 탐색하기 위한 목적으로 사용
- 해를 찾는 도중 해가 아니어서 막히면 되돌아  
가서 다시 해를 찾아가는 기법  
(최적화 문제와 결정 문제를 푸는 방법)



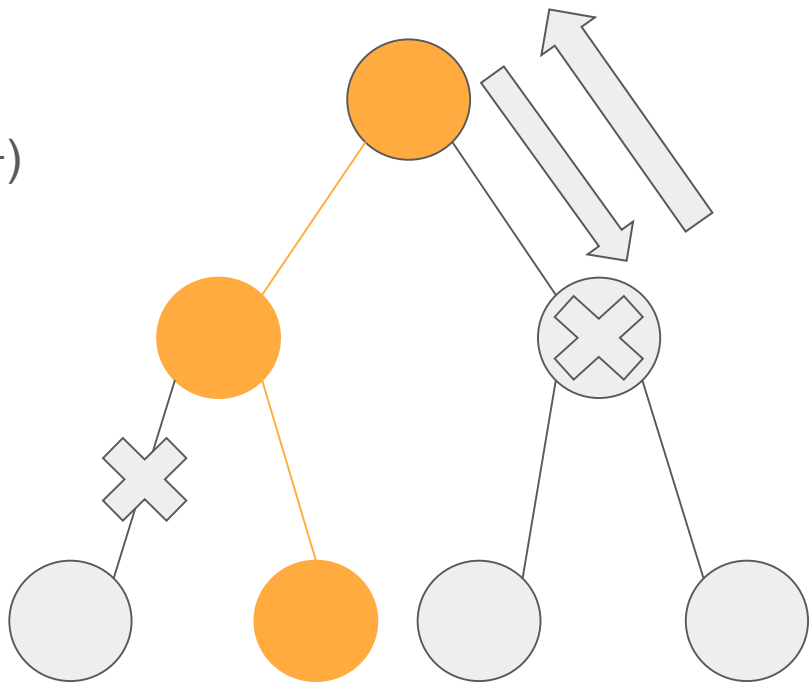
# 백트래킹이란?

- 백트래킹은 단순한 완전탐색 X  
(가능한 모든 경로를 탐색하는 DFS와 다름)

- 조건**에 따라 유망한 노드로 이동  
(탐색하지 않아도 되는 곳을 미리 막음)

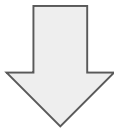
=> **“가지치기”**

가지치기를 얼마나 잘하느냐가 효율성 결정



# 백트래킹이란?

- (1) 모든 가능한 경우의 수 중
- (2) **특정한 조건**을 만족하는 경우만 살펴보는 것



- (1) **DFS / BFS**를 활용
- (2) **조건문(if-else)**으로 절대로 답이 될 수 없는 상황 정의  
가지치기 (Pruning)
- (1) 답이 될 수 없는 상황 => **이전으로 돌아가 다른 경우 탐색**

Back Tracking

# 백트래킹

- 일반적으로 백트래킹은 DFS와 결합된 방법으로 많이 풀  
(DFS의 구현 방식이 백트래킹과 잘 어울리기 때문)
- 탐색에서 순환이 발생할 조건이라면 BFS방식과 결합해서 사용

# 백트래킹



일반적으로 백트래킹 문제는 DFS와 함께 사용되며, DFS의 구현 방식이 백트래킹과 잘 어울리기 때문에 DFS가 백트래킹 문제에 더 자주 사용됩니다. 하지만 순환 구조를 탐지하는 것이 중요한 문제에서는 BFS 구조를 사용하는 것이 더 효율적입니다.



순환 구조를 탐지하는 방법 중 하나는 DFS를 사용하는 것입니다. DFS는 스택을 이용하여 구현되며, 현재 상태에서 가능한 모든 경우의 수를 탐색하면서 해를 찾아가는 알고리즘입니다. 그러나 순환 구조가 발생하면 DFS는 무한히 탐색을 반복하게 되어 시간 초과나 메모리 초과 문제가 발생할 수 있습니다.

반면 BFS는 큐를 이용하여 구현되며, 모든 경로를 탐색하는 것이 아니라 최단 경로를 탐색하기 때문에, 순환 구조를 탐지할 수 있습니다. 따라서 순환 구조를 탐지하는 것이 중요한 문제에서는 BFS 구조를 사용하는 것이 더 효율적입니다.

# 백트래킹의 일반적인 코드 형태

def function():

if (종료 조건을 만족한다면) :  
    처리

재귀함수의 종료조건  
(없을 경우 무한 루프에 빠짐)  
일반적으로 마지막 노드 접근 시 종료

for (자식 노드 확인) :  
    if (임의 조건 만족한다면):  
        처리

유망한 경우에 대해서만 재귀함수 호출  
=> 가능한 노드에 대해서 재귀적으로 함수 호출

자식노드 방문

재귀함수 호출



자식노드 방문

처리 해제



# 백트래킹 문제 유형



백트래킹은 여러 종류의 문제에 적용될 수 있습니다. 이번에는 대표적인 백트래킹 문제 유형 몇 가지를 살펴보겠습니다.  

## 1. N-Queens 문제

N-Queens 문제는  $N \times N$  크기의 체스판 위에  $N$ 개의 퀸을 서로 공격할 수 없도록 놓는 문제입니다. 이 문제는 백트래킹을 이용하여 해결할 수 있습니다.

## 2. 스도쿠 문제

스도쿠 문제는  $9 \times 9$  크기의 격자 판 위에 숫자를 채워 넣는 문제입니다. 각 행, 각 열, 그리고  $3 \times 3$  크기의 작은 사각형 안에는 중복된 숫자가 없어야 합니다. 이 문제도 백트래킹을 이용하여 해결할 수 있습니다.

## 3. 조합 문제

조합 문제는 주어진 집합에서 특정 수의 원소를 선택하는 경우의 수를 구하는 문제입니다. 이 문제도 백트래킹을 이용하여 해결할 수 있습니다.

## 4. 그래프 문제

그래프 문제에서 백트래킹을 사용하는 경우가 있습니다. 예를 들어, 한 노드에서 시작하여 모든 노드를 방문하는 문제나, 두 노드 사이의 경로를 찾는 문제 등이 있습니다.

## 5. 문자열 문제

문자열 문제에서도 백트래킹을 사용하는 경우가 있습니다. 예를 들어, 문자열에서 주어진 조건을 만족하는 모든 순열을 찾는 문제나, 문자열에서 주어진 조건을 만족하는 모든 부분 문자열을 찾는 문제 등이 있습니다.

이 외에도 다양한 문제에서 백트래킹을 사용할 수 있습니다. 백트래킹은 일반적으로 완전탐색 알고리즘보다 효율적이며, DFS(Depth-First Search) 알고리즘을 기반으로 구현됩니다.

# 백트래킹 문제 유형

## 1. N-Queens 문제

$N * N$  크기의 체스판 위에  $N$ 개의 퀸을 서로 공격할 수 없도록 놓는 문제  
퀸은 가로, 세로, 대각선으로 움직이기에  
이 조건을 이용하면 퀸을 놓을 수 있는 공간이 줄어듦

## 1. 스도쿠 문제

$9 * 9$  크기의 격자판 위에 숫자를 채워넣는 문제  
가로, 세로,  $3 * 3$  크기의 사각형 내 중복 숫자 없어야 한다는 조건 이용

# 백트래킹 문제 유형

## 4. 조합 문제

주어진 집합에서 특정 수의 원소를 선택하는 경우의 수.  
한번 선택한 원소는 중복해서 선택이 불가능하다는 조건 이용.

## 5. 그래프 문제

한 노드에서 시작해 모든 노드를 방문하는 문제 / 두 노드 사이의 경로를 찾는 문제 등

## 6. 문자열 문제

문자열 내 주어진 조건을 만족하는 모든 순열을 찾는 문제  
문자열에서 주어진 조건을 만족하는 모든 부분 문자열을 찾는 문제 등

백 트래킹 대표 문제 풀이 : N-Queens

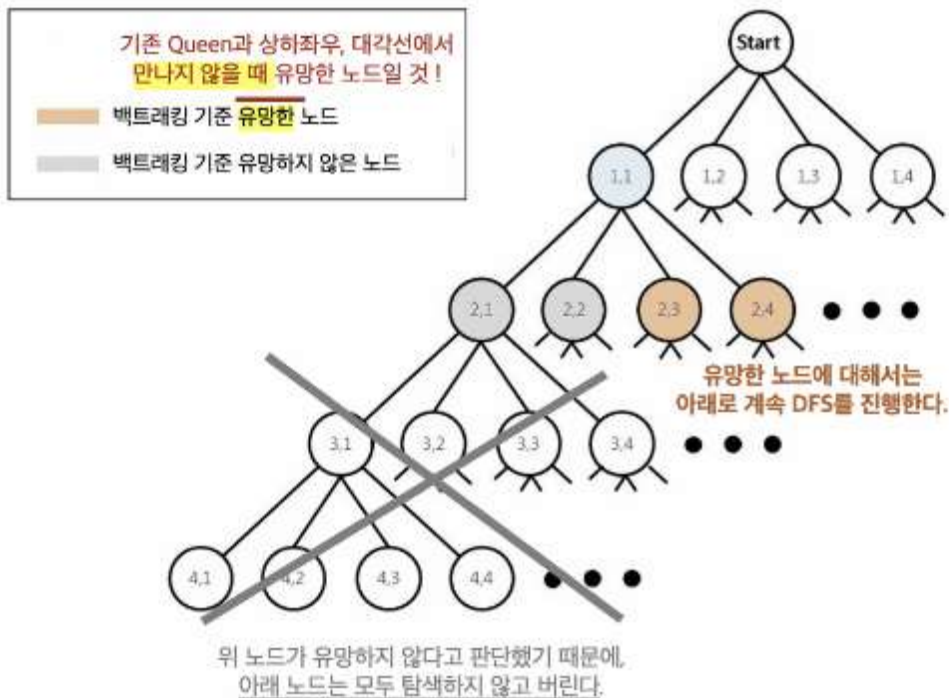
[illegible]

## 백 트래킹 대표 문제 풀이 : N-Queens

		Q					
V				V	V	V	V

완전 탐색을 하더라도  
Queen 이 놓일 수 없는 자리는 제외하고 탐색하자,  
=> 각 행에 한개, 각 열에 한개만  
=> 대각선 범위에서도 겹치지 않게

# 백 트래킹 대표 문제 풀이 : N-Queens



## Depth (N)



Row 단위로 노드를 뺌으므로  
DFS의 depth 또한 N이 될 것이다.

# 참고. 백트래킹 문제임을 파악하는 방법

민지

그럼 어떤 알고리즘 문제가 백트래킹 문제라는 것을 어떻게 파악할 수 있어?



백트래킹 문제를 파악하기 위해서는 문제의 특성과 해결 방법을 이해하는 것이 중요합니다.    
반적으로 백트래킹 문제는 다음과 같은 특징을 가지고 있습니다.

1. 모든 경우의 수를 탐색해야 할 때: 백트래킹은 모든 경우의 수를 탐색하는 알고리즘입니다. 따라서, "모든 경우의 수를 다 해보고 최적해를 찾아내야 한다"는 문제는 백트래킹으로 해결할 수 있는 문제일 가능성이 높습니다.
2. 일정 조건을 만족하는 해를 찾아야 할 때: 백트래킹은 조건에 맞지 않는 경우에는 탐색하지 않고 되돌아가는(backtracking) 기법을 사용하여 탐색 범위를 줄이는 것입니다. 따라서, "특정한 조건을 만족하는 해를 찾아야 한다"는 문제는 백트래킹으로 해결할 수 있는 문제일 가능성이 높습니다.
3. 재귀적인 구조를 가지고 있을 때: 백트래킹은 주로 재귀적인 구조로 구현됩니다. 따라서, "재귀적으로 문제를 해결해야 한다"는 문제는 백트래킹으로 해결할 수 있는 문제일 가능성이 높습니다.
4. 백트래킹의 기법을 사용할 수 있는 문제일 때: 백트래킹은 불필요한 탐색을 줄이기 위한 가지치기 기법 등을 사용하여 효율적으로 탐색 범위를 줄입니다. 따라서, "백트래킹 기법을 사용하여 문제를 해결해야 한다"는 문제는 백트래킹으로 해결할 수 있는 문제일 가능성이 높습니다.

이러한 특징을 파악하고, 문제를 분석하여 백트래킹을 적용할 수 있는지를 고려해보면 됩니다. 물론, 백트래킹이 항상 최적의 해를 찾아낼 수 있는 것은 아니기 때문에, 문제의 특성과 조건에 따라 다른 알고리즘을 적용해야 할 수도 있습니다.

# 참고. 백트래킹의 기법

민지

백트래킹의 기법은 뭐가 있어?



백트래킹은 다양한 기법을 사용하여 구현될 수 있습니다. 이 중에서 대표적인 백트래킹 기법으로는 다음과 같은 것들이 있습니다.



1. 상태 공간 트리: 문제의 가능한 모든 상태를 트리 구조로 표현하는 기법입니다. 각 노드는 상태를 나타내며, 각 간선은 가능한 상태 전이를 의미합니다. 상태 공간 트리를 만들고, 깊이 우선 탐색(DFS)을 사용하여 트리를 탐색하며 최적해를 찾아냅니다.
2. 가지치기: 상태 공간 트리에서 불필요한 탐색을 줄이기 위한 기법입니다. 각 노드를 탐색하기 전에, 해당 노드에서 이어지는 가지를 모두 탐색해본 후에 그 중에서 유망한 가지만을 선택하여 탐색을 진행합니다.
3. 제약 충족: 백트래킹 기법을 사용하여 제약 조건을 만족하는 해를 찾아내는 기법입니다. 각 변수의 도메인(domain)을 정하고, 각 변수의 도메인에서 값을 선택해가면서 제약 조건을 만족하는 값을 찾아냅니다.
4. 순열과 조합: 순열과 조합을 사용하여 가능한 모든 경우를 탐색하는 기법입니다. 순열은  $n$ 개의 원소 중에서  $r$ 개를 선택하여 나열하는 경우의 수를 탐색하며, 조합은  $n$ 개의 원소 중에서  $r$ 개를 선택하는 경우의 수를 탐색합니다.
5. 부분 집합: 주어진 집합의 부분 집합을 찾는 문제를 해결하는 기법입니다. 각 원소를 선택하거나 선택하지 않는 2가지 선택지를 가지고 모든 경우를 탐색하여 부분 집합을 찾아냅니다.

이러한 백트래킹 기법들은 서로 조합하여 사용할 수 있으며, 문제의 특성에 맞게 적절한 기법을 선택하여 구현할 수 있습니다.



# 백트래킹과 DP의 차이점

민지

백트래킹과 DP의 차이는 뭐야?



백트래킹과 DP(Dynamic Programming)는 모두 최적화 문제를 해결하는 알고리즘 기법 중 하나입니다. 하지만 두 기법의 접근 방식과 목적은 다릅니다.



## 1. 접근 방식

- 백트래킹: 상태 공간 트리를 생성하여 깊이 우선 탐색(DFS)을 수행하면서 최적해를 찾아냅니다. 모든 가능한 경우를 탐색하지만, 가지치기를 통해 불필요한 탐색을 줄입니다.
- DP: 작은 부분 문제(subproblem)의 최적해를 구하고, 이를 이용하여 큰 문제의 최적해를 구합니다. 작은 부분 문제들을 중복 계산하지 않도록 메모이제이션(Memoization)이나 Bottom-up 방식으로 구현합니다.

## 2. 목적

- 백트래킹: 가능한 모든 해를 찾는 것이 목적입니다. 최적해를 찾기 위해 모든 가능한 경우를 탐색합니다.
- DP: 최적해를 찾는 것이 목적입니다. 작은 부분 문제들의 최적해를 이용하여 큰 문제의 최적해를 구합니다.

따라서, 백트래킹은 가능한 모든 경우를 탐색하여 최적해를 찾지만, DP는 최적 부분 구조(optimal substructure)와 중복 부분 문제(overlapping subproblems)를 이용하여 최적해를 구합니다.