

자료: 03\_search 개요 질문: 인공지능기초 과목에서 대표 탐색 문제 6문제: 8-puzzle, 8-queens, 최단 경로, 순회 외판원, 배낭, 집합 커버링 문제 해당 탐색 문제들의 해결에서 각 연산자 시퀀스들과 목적 함수의 수학적 모델링에 대한 이론을 공부하고 싶은데 uninformed search와 informed search에 대한 분류와 정리는 했지만 알고리즘 기반으로 탐색 문제를 해결하고 응용하는 방법을 모르겠다. 문제들을 해결하는 이론으로 경우의 수와 제약을 통해 해를 구하는 과정을 초심자에게 알려주세요. 생각시간: 17s 탐색 문제 이해하기 사용자의 요청을 보니, 인공지능 기초 과목에서 8-puzzle, 8-queens, 최단 경로, 순회 외판원, 배낭, 집합 커버링 같은 대표 탐색 문제를 다루고 계시네요. 답변: 이 문제들을 해결할 때 연산자 시퀀스와 목적 함수의 수학적 모델링 이론을 공부하고 싶어 하시는데, uninformed search와 informed search는 이미 분류하셨지만 알고리즘으로 적용하는 부분이 어렵다고 하시네요.

## 8-Puzzle 문제

8-Puzzle은 3x3 격자에 1부터 8까지의 타일과 하나의 빈 공간(공백)이 있는 퍼즐로, 타일을 이동시켜 목표 상태(예: 1-2-3 / 4-5-6 / 7-8-공백)로 만드는 문제입니다. 이는 상태 공간 탐색의 대표적인 예로, AI에서 검색 알고리즘을 테스트하는 데 자주 사용됩니다.

### 수학적 모델링

- **상태 공간(State Space):** 각 상태는 3x3 격자의 타일 배치로 표현됩니다. 총 가능한 상태 수는  $9! / 2 = 181,440$ 개입니다. (9!은 모든 배치의 경우의 수, 하지만 절반은 해결 불가능한 상태이므로 나누기 2.)
- **연산자(Operators):** 빈 공간을 상, 하, 좌, 우로 이동하는 4가지 행동. 각 연산자는 현재 상태에서 새로운 상태를 생성합니다. 예: 빈 공간이 (2,2)에 있으면, 상(1,2)의 타일을 내려서 교환.
- **목적 함수(Goal Function):** 목표 상태와 일치하는지 확인. 비용 함수로는 이동 횟수(경로 비용)나 휴리스틱(예: Manhattan 거리, 각 타일이 목표 위치에서 떨어진 거리의 합)을 사용.
- **제약 조건:** 타일은 격자 밖으로 나갈 수 없음. 일부 초기 상태는 목표에 도달 불가능(패리티 제약: 타일 순열의 짝/홀 성이 맞아야 함).

### 해 구하는 과정 (경우의 수와 제약을 통해)

초심자로서, 모든 가능한 이동 시퀀스를 나열하면 경우의 수가 폭발적으로 증가합니다(트리 깊이  $d$ 에서 최대  $4^d$ ). 그래서 검색 알고리즘을 사용:

1. **Uninformed Search 적용:** BFS(너비 우선 탐색)를 사용해 최단 경로를 찾음. 큐에 초기 상태를 넣고, 각 상태에서 연산자를 적용해 자식 상태 생성. 방문한 상태를 해시로 체크해 중복 피함. 제약: 메모리 사용량 많음 ( $O(b^d)$ ,  $b$ =분기 계수 $\approx 3$ ,  $d$ =깊이).
  2. **Informed Search 적용:** A\* 알고리즘 사용.  $f(n) = g(n) + h(n)$ ,  $g(n)$ 은 시작부터의 이동 수,  $h(n)$ 은 Manhattan 거리. 휴리스틱이 과소추정(admissible)이면 최적 해 보장. 경우의 수 줄이기: 우선순위 큐로 유망한 상태 먼저 탐색.
- **예시:** 초기 상태에서 빈 공간 이동으로 2-3단계 시뮬레이션 해보세요. Python 코드로 시뮬하면 이해 쉬움 (아래 코드 실행 팁).

## 8-Queens 문제

8-Queens는 8x8 체스판에 8개의 퀸을 놓아, 서로 공격하지 않게 하는 문제입니다. 퀸은 행, 열, 대각선으로 공격 가능.

### 수학적 모델링

- **상태 공간:** 각 퀸의 위치를 (행, 열)로 표현. 총 경우의 수는  $8^8 = 16,777,216$ 개지만, 제약으로 줄임 (각 행에 하나씩 놓으면  $8! = 40,320$ ).
- **연산자:** 한 퀸의 위치를 열 내에서 이동하거나, 다음 행에 퀸 놓기.
- **목적 함수:** 모든 퀸이 서로 공격하지 않음 (충돌 수 = 0). 비용 함수: 충돌 쌍 수 (휴리스틱으로 사용).
- **제약 조건:** 한 행/열에 하나의 퀸만. 대각선 충돌 없음 ( $\text{행}i - \text{행}j \neq |\text{열}i - \text{열}j|$ ).

## 해 구하는 과정

경우의 수가 많아 brute-force는 비효율적. 백트래킹으로 제약 위반 시 가지치기.

1. **Uninformed Search 적용:** DFS(깊이 우선 탐색)로 행마다 퀸 놓기. 제약 위반 시 백트랙. 모든 경우 탐색하지만, 제약으로 pruning.
2. **Informed Search 적용:** Hill-Climbing이나 Genetic Algorithm (informed 변형). 휴리스틱: 충돌 수 최소화. 로컬 최적에 빠질 수 있음.

- **초심자 팁:** 행 1부터 퀸 놓고, 충돌 체크. 충돌 시 다음 열 시도. 총 해는 92개.

## 최단 경로 문제 (Shortest Path)

그래프에서 시작 노드에서 목표 노드까지 최소 비용 경로 찾기. 예: 지도상의 도시 간 최단 거리.

## 수학적 모델링

- **상태 공간:** 노드 집합  $V$ , 간선  $E$ 로 그래프  $G=(V,E)$ . 경우의 수: 최악  $2^{|V|}$  (지수적).
- **연산자:** 인접 노드로 이동.
- **목적 함수:** 경로 비용 합 최소화 (비용  $c(u,v)$ ).
- **제약 조건:** 양의 비용 (음수 시 Bellman-Ford), 사이클 없음.

## 해 구하는 과정

1. **Uninformed Search 적용:** BFS (비가중 그래프, 모든 비용 1). Dijkstra (가중, uninformed 변형).
  2. **Informed Search 적용:** A\* with  $h(n)$  = 목표까지 직선 거리 (Euclidean).  $f(n) = g(n) + h(n)$ .
- **예시:** 노드 A-B-C, 비용 1-2. BFS: A->B (1), A->B->C (3).

## 순회 외판원 문제 (TSP: Traveling Salesman Problem)

$n$ 개 도시를 모두 방문하고 시작으로 돌아오는 최소 비용 순회 경로.

## 수학적 모델링

- **상태 공간:** 도시 순열. 경우의 수  $(n-1)!/2$  (대칭성).
- **연산자:** 다음 도시 추가.
- **목적 함수:** 경로 비용 합 최소.
- **제약 조건:** 모든 도시 방문, 사이클.

## 해 구하는 과정

NP-hard, 경우의 수 폭발. Branch-and-Bound로 pruning.

1. **Uninformed Search 적용:** DFS with backtracking.
2. **Informed Search 적용:** A\* with MST 휴리스틱 (남은 도시 최소 신장 트리 비용).

- **초심자:** 작은  $n=4$ 로 손 계산.

## 배낭 문제 (Knapsack Problem)

용량  $W$ 인 배낭에 가치  $v_i$ , 무게  $w_i$  아이템을 최대 가치로 넣기.

### 수학적 모델링

- **상태 공간:** 아이템 부분집합.  $2^n$  경우.
- **연산자:** 아이템 포함/제외.
- **목적 함수:** 가치 합 최대, 무게  $\leq W$ .
- **제약 조건:** 무게 초과 금지.

### 해 구하는 과정

동적 프로그래밍이나 검색.

1. **Uninformed Search 적용:** BFS for 0/1 Knapsack.
2. **Informed Search 적용:** Greedy (가치/무게 비율) approx.

- **예시:** 아이템 3개, DP 테이블로 계산.

## 집합 커버링 문제 (Set Covering Problem)

집합  $U$ 를 커버하는 부분집합  $S$ 의 최소 수 찾기.

### 수학적 모델링

- **상태 공간:** 부분집합 조합.  $2^m$  경우.
- **연산자:** 집합 추가.
- **목적 함수:** 선택 집합 수 최소,  $U$  커버.
- **제약 조건:** 완전 커버.

### 해 구하는 과정

NP-hard. Greedy approx.

1. **Uninformed Search 적용:** BFS.
2. **Informed Search 적용:** A\* with 휴리스틱 (남은 원소 / 최대 집합 크기).

- **초심자:** Greedy: 가장 많은 원소 커버 집합 선택 반복.

이 문제들은 검색 트리로 모델링되며, uninformed는 맹목적, informed는 휴리스틱으로 효율. 코드로 실험 추천 (Python with queue/heapq). 추가 질문 환영!