Problem solving by Searching
: Introduction

김성영교수
국립금오공과대학교
컴퓨터공학부

※본 자료는 국립금오공과대학교 컴퓨터공학부 황준하교수님의 자료를 참조 및 수정한 것임

Contents

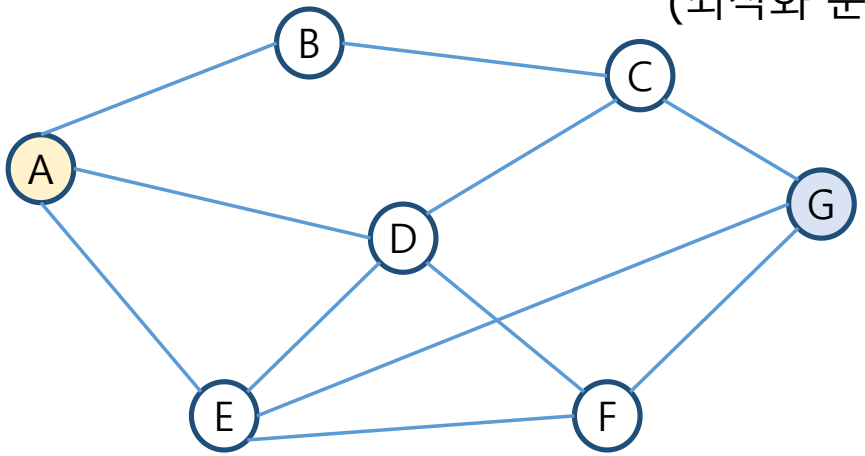
- 탐색을 통한 문제 해결이란 무엇인지 설명할 수 있다.
- 문제를 정의하는 방법을 설명할 수 있다.
- 다양한 탐색 문제를 구분하여 설명할 수 있다.
  - 8-puzzle 문제
  - 8-queens 문제
  - 최단 경로 찾기 문제
  - 순회 외판원 문제
  - 배낭 문제 등
- 탐색 알고리즘의 종류를 구분하여 설명할 수 있다.

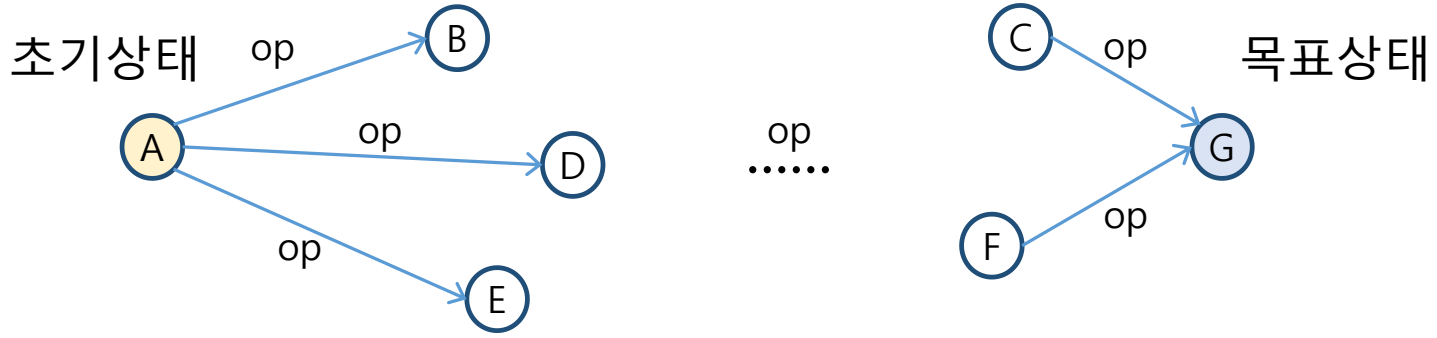
탐색을 통한 문제해결이란?

최단 경로 찾기 문제와 탐색을 통한 문제 해결

- 도시 A로부터 도시 G까지의 **최단 경로**는?
  - 해법: 모든 경로 나열
    - 도시 개수가 많아지면?
  - 다른 해법들은?
- 탐색(Search)
  - 여러 방법(경로)들 중 가장 좋은 방법(경로)을 찾는 과정
  - 도시: 상태
  - 연산자: 상태 이동

최단 경로 문제
(최적화 문제)



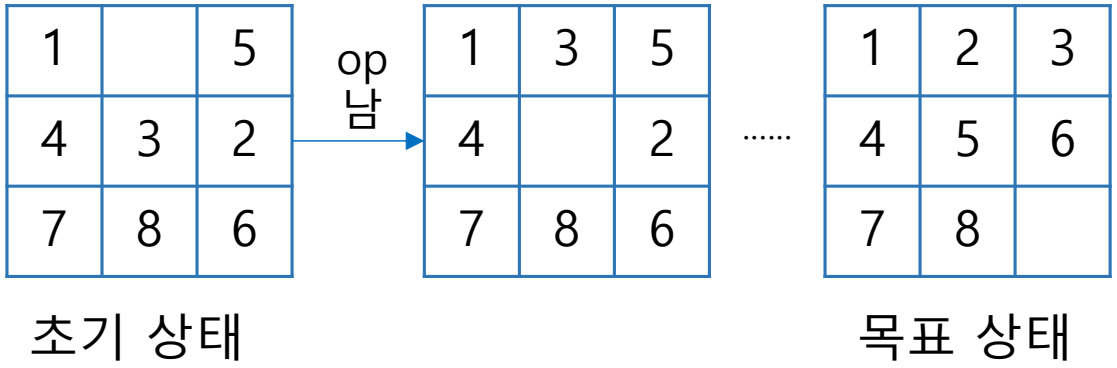


탐색을 통한 문제 해결:
초기 상태로부터 **목표 상태**까지 갈 수 있는 일련의 연산자를 찾는 것

최적해 vs. 준최적해

8-puzzle 문제와 탐색을 통한 문제 해결

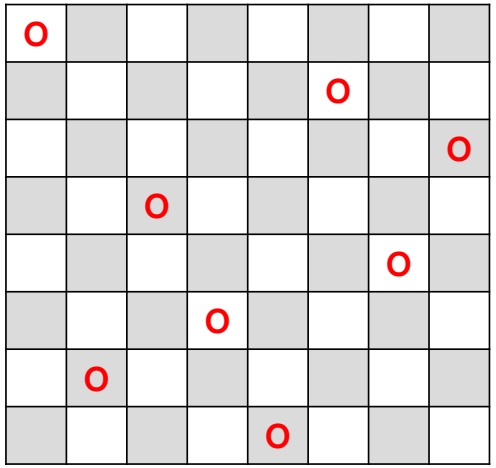
- 8-puzzle 문제 (3x3)
  - 초기 상태로부터 공백 타일을 최소로 이동하여 목표 상태를 만듦



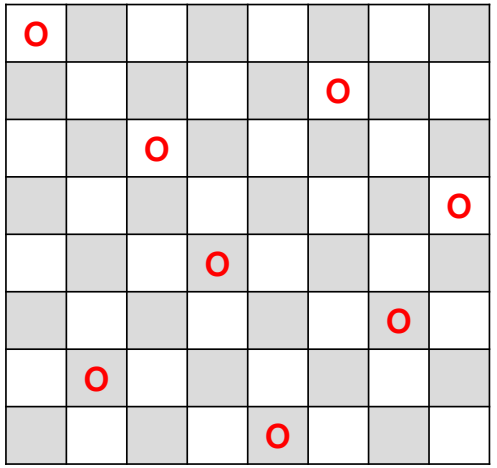
- 탐색을 통한 문제 해결
  - 목표 상태가 명시적으로 주어짐
  - 최단 경로와 동일하게 초기 상태로부터 목표 상태로 갈 수 있는 일련의 연산자를 찾는 것
- 연산자
  - 3을 북쪽으로 이동?
  - 공백을 남쪽으로 이동!
    - 항상 동, 서, 남, 북 4개
- 15-puzzle(4x4), 24-puzzle(5x5), ..., n-puzzle 문제 확장 가능

8-queens 문제와 탐색을 통한 문제 해결

- 8-queens 문제
  - 8x8 체스판 위에 8개의 queen들을 배치하되 가로, 세로, 대각선 방향으로 직선 상에 겹치지 않도록 queen들을 배치
- 탐색을 통한 문제 해결
  - 목표 상태가 명시적으로 주어지지 않음
  - 따라서, **목표 상태 자체를 찾는 것**!
- 문제 확장
  - 4-queens부터 해 존재
  - 10-queens, 100-queens, …, n-queens 문제로 확장 가능



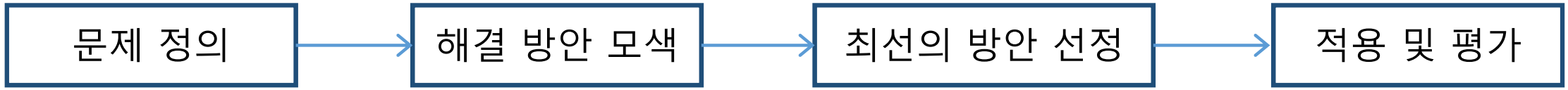
(a) 해인 경우



(b) 해가 아닌 경우

문제 정의하기

문제 해결 과정



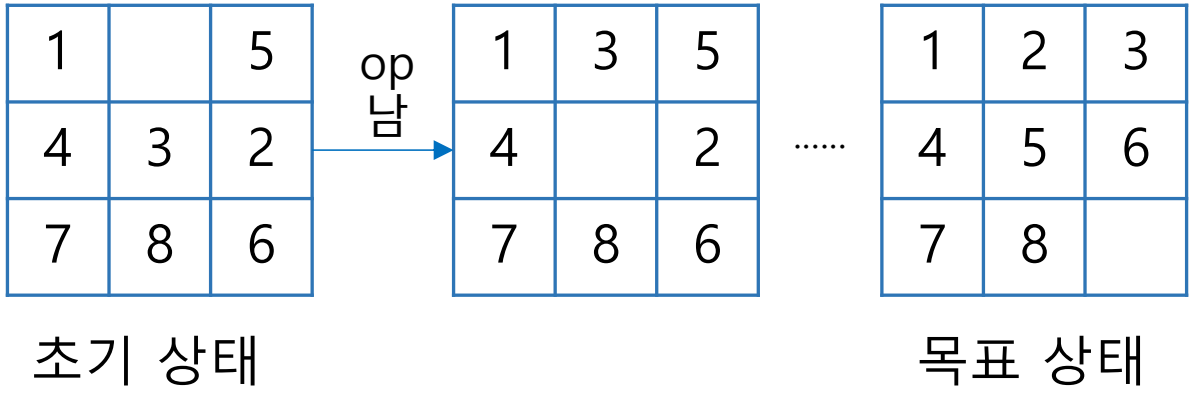
- 문제 정의
  - 문제를 형식화된 형태로 서술하는 것
- 해결 방안 모색
  - 다양한 아이디어 도출 (브레인 스토밍, 마인드맵, 여섯 생각 모자 등을 활용)
- 최선의 방안 선정
  - 최적해의 도출 여부, 소요 시간, 필요한 메모리 크기 등을 고려하여 최선의 방안 선택
- 적용 및 평가
  - 3단계 방안을 적용하고 결과 평가

문제 정의

- 문제 정의
  - 문제를 형식화된 형태로 서술하는 것
  - 문제 정의를 위해 기술되어야 할 사항
    - 초기 상태, 목표 상태 및 연산자
      - 목표 상태가 명시적으로 주어지지 않는 경우 제약 조건을 포함할 수 있음
    - 비용 함수(최소화 문제) 또는 이득 함수(최대화 문제) ⇒ 목적 함수(objective function)
      - 초기 상태로부터 연산자를 적용하여 변화된 일련의 상태까지의 비용의 합
        (예, 최단 경로 찾기 문제 → 거리의 합)
- 해(solution): 초기 상태에서 목표 상태까지의 경로 (최단 경로 문제)
  - 최적해(optimal solution): 비용 함수값이 가장 작은 해

8-puzzle 문제 정의하기

- 초기 상태와 목표 상태 : 문제에서 주어짐
- 연산자: 공백 기준 동쪽, 서쪽, 남쪽, 북쪽으로 이동
- 비용 함수: 공백을 한 번 움직일 때 1 증가
- 최적화 문제: 최소화 문제
  - 비용 함수를 최소화하는 경로 찾기

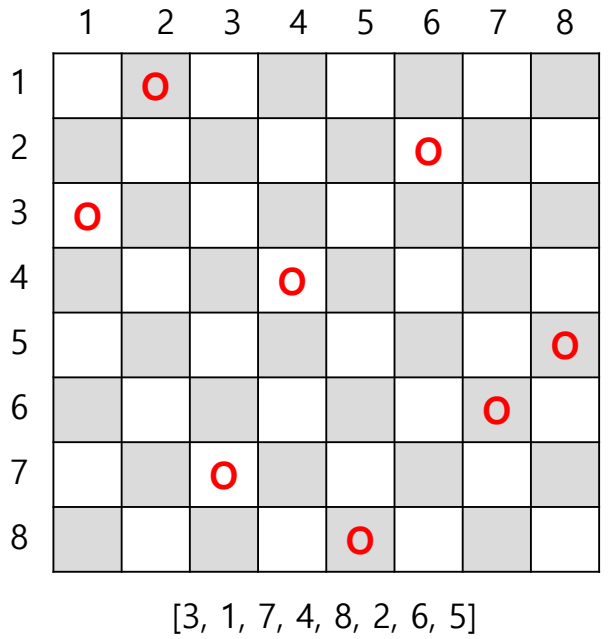


8-queens 문제 정의하기 (1)

- 문제 정의 방법 (1) : 이진 변수 기반 모델
  - 초기 상태
    - 비어있는 8x8 체스판
  - 목표 상태
    - 모든 queen 쌍에 대해 가로, 세로, 대각선 동일 직선 상에 배치되어 있지 않은 상태
  - 연산자
    - 하나의 queen을 현재 비어 있는 위치에 배치
  - 비용 함수?
    - 정의하기 어려움

8-queens 문제 정의하기 (2)

- 문제 정의 방법 (2) : 순열 기반 모델
  - 초기 상태
    - 각 열의 임의 위치에 queen 배치 (모두 다른 행): 초기해
  - 목표 상태
    - 정의 방법 (1)과 동일
  - 연산자
    - queen 위치를 다른 행으로 이동
    - 두 개 queen의 행 위치를 상호 교환
  - 비용 함수
    - 직선 위치에 있는 queen 쌍의 수
  - 최적화 문제: 최소화 문제
    - 비용 함수값이 0이 되면 문제 해결 완료



[3, 1, 7, 4, 8, 2, 6, 5]

초기상태(초기해)의 예

초기해의 비용은?

탐색 알고리즘의 성능 평가

- 탐색 문제 → 탐색 알고리즘 적용 : 성능의 우수성 평가
  - 효과성(effectiveness)
    - 충분한 시간이 주어진다면 결국 하나의 해를 찾을 수 있다고 보장할 수 있는가?
  - 해의 품질(solution quality, path cost)
    - 찾은 해가 얼마나 좋은가?
    - 궁극적으로는 최적해를 찾을 수 있는가? (찾을 수 있지만 너무 많은 시간이 걸린다면?)
  - 효율성(efficiency, search cost)
    - 하나의 해를 찾기까지 필요한 시간과 메모리 등의 소모 자원
- Total cost = path cost + search cost
  - 어떤 search 알고리즘이 좋은가? ⇒ 목표에 따라 결정
  - (최적해 보장 but 100일 소요) Vs. (쓸만한 해 and 1시간 소요)

다양한 탐색 문제

- 8-puzzle 문제(8-puzzle problem), N-puzzle 문제
- 8-queens 문제(8-queens problem), N-queens 문제
- 최단 경로 찾기 문제(shortest path finding problem)
- 순회 외판원 문제(traveling salesman problem)
- 배낭 문제(knapsack problem)
- 다차원 배낭 문제(multidimensional knapsack problem)
- 집합 커버링 문제(set covering problem)
- 최대 커버링 문제(maximal covering problem)

8-puzzle 문제 (1)

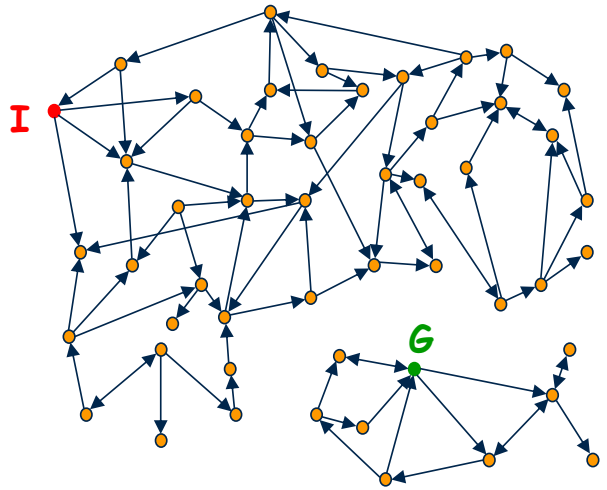
- Sam Loyd의 퍼즐 이야기
  - 19세기 말 세계적인 퍼즐 작가, "America's greatest puzzle-expert"
    - 제안: 다음 초기 상태를 목표 상태로 만드는 문제를 푸는 사람에게 1,000달러 상금 지급
    - 불가능
      - 16!개의 모든 상태는 절반씩 나뉘어 있어 두 그룹 사이는 이동 불가능

1	2	3	4
5	6	7	8
9	10	11	12
13	15	14	

초기 상태

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

목표 상태



8-puzzle 문제 (2)

- 초기 상태를 만들 때는 목표 상태로 이동 가능한지 확인 필요
- 초기 상태 만드는 법
  - 방법: 목표 상태로부터 무작위 이동
  - 방법2: 목표 상태와 비교하여 inversion(역전) 쌍의 개수가 다음과 같으면 가능

n x n 크기의 N-puzzle 문제의 경우
- n의 크기가 홀수이면 inversion의 개수가 짝수일 때
- n의 크기가 짝수이면 inversion의 개수가 홀수일 때

1		5
4	3	2
7	8	6

Inversion:
(5, 4), (5, 3), (5, 2), (4, 3), (4, 2), (3, 2), (7, 6), (8, 6)

8-queens 문제 : 수학적 모델링의 중요성 (1)

- 문제 정의 단계는 해당 문제를 분석하고 모델링하는 것을 포함
- 모델링
  - 복잡한 현실 세계를 단순화 또는 추상화하여 표현
  - 모델 하우스: 실제 집을 단순화하여 표현함으로써 실제 집에 대한 이해 제공
  - 모델링을 통해 개발자들은 실제 개발해야 될 목표물에 대해 서로 의사소통을 할 수 있고 문제점을 미리 파악하고 해결 가능
- 수학적 모델링
  - 수학적 개념과 표현을 사용하여 주어진 문제를 기술
  - 모델링 자체로 문제에 대한 많은 부분 해결
    - 예) 연립 방정식으로 표현되면, 방적식만 풀면 됨

8-queens 문제 : 수학적 모델링의 중요성 (2)

- n-queens 문제에 대한 수학적 모델링: 문제 정의 방법1 (이진 변수 기반 모델)
  - 각 queen은 체스판의 어느 위치든 배치 가능
    - x_{ij} : (i, j)에 queen이 놓이면 1, 아니면 0인 결정 변수

가로 $\sum_{j=1}^{n} x_{ij} = 1 \; for \; i = 1, \dots, n$

세로 $\sum_{i=1}^{n} x_{ij} = 1 \; for \; j = 1, \dots, n$

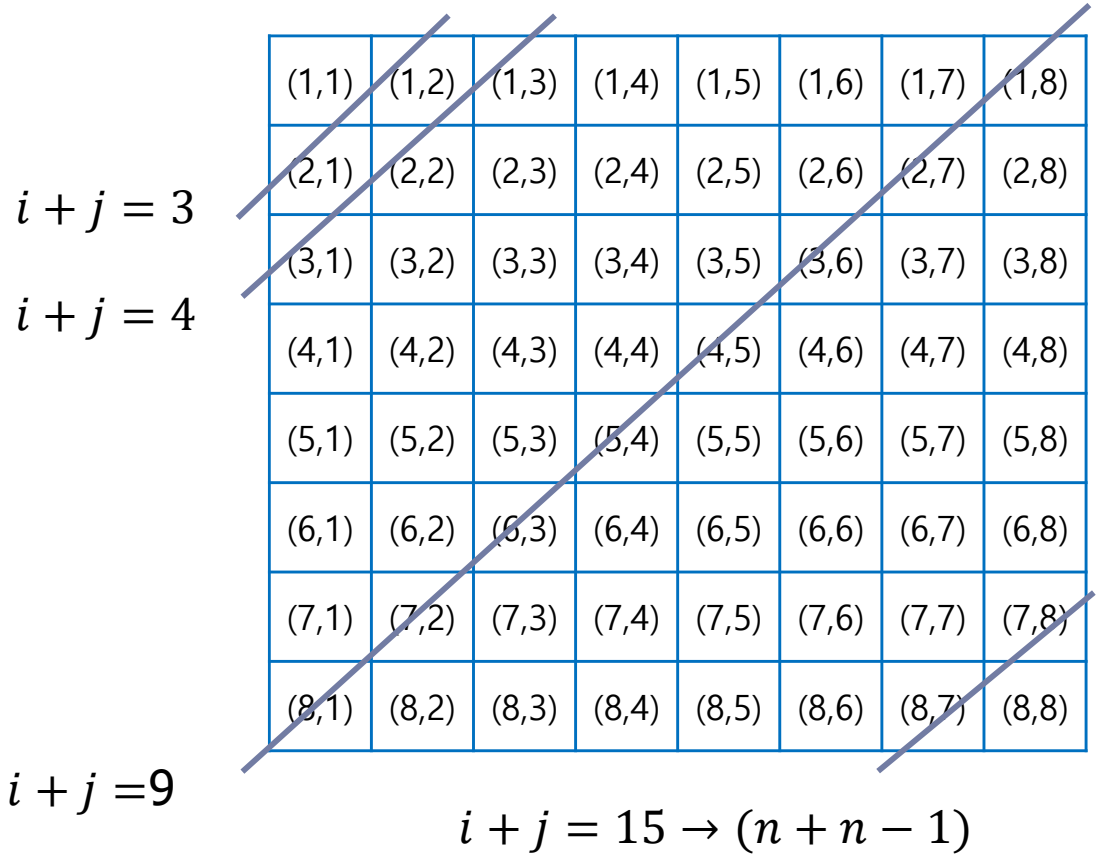
/ 대각선 $\sum_{i=0}^{n} \sum_{j=1}^{n} x_{ij} \leq 1 \; for \; i + j = 3, \dots, (n + n - 1)$

□ 대각선 $\sum_{i=0}^{n} \sum_{j=1}^{n} x_{ij} \leq 1 \; for \; i - j = -(n - 2), \dots, (n - 2)$

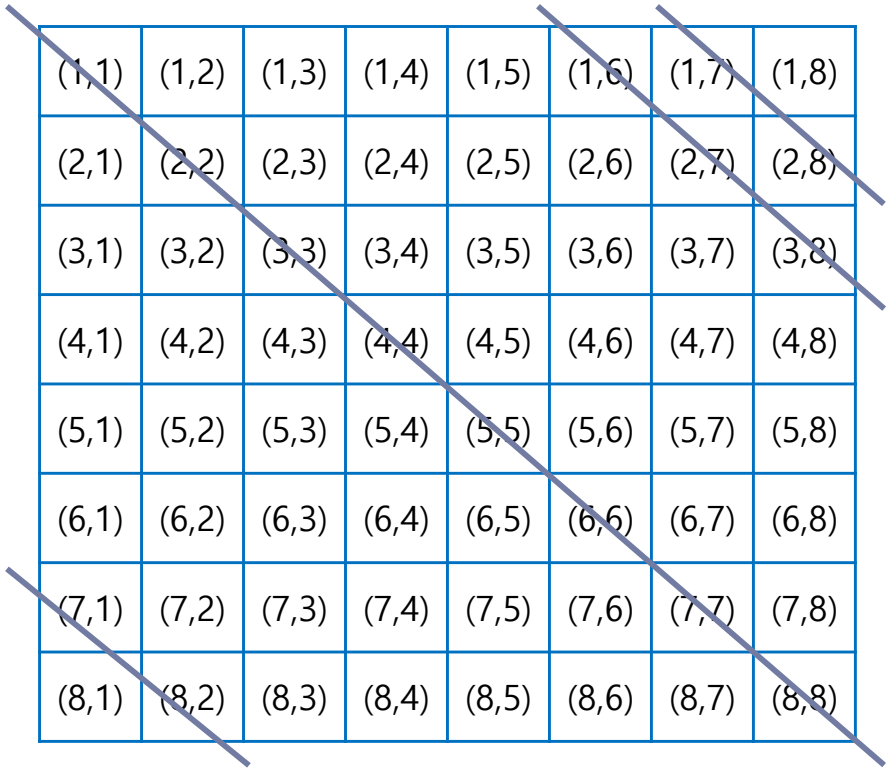
8 x 8 체스판

i \ j							
(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	(1,8)
(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)	(2,8)
(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)	(3,8)
(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)	(4,8)
(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)	(5,8)
(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)	(6,8)
(7,1)	(7,2)	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)	(7,8)
(8,1)	(8,2)	(8,3)	(8,4)	(8,5)	(8,6)	(8,7)	(8,8)

8-queens 문제에 대한 부대각선 (/ 대각선)



8-queens 문제에 대한 주대각선 (□대각선)



$i - j = -6 \rightarrow -(n-2)$

$i - j = -5 \rightarrow -(n-3)$

$i - j = 0 \rightarrow (n-n)$

$i - j = 6 \rightarrow (n-2)$

8-queens 문제 : 수학적 모델링의 중요성 (3)

- n-queens 문제에 대한 수학적 모델링: 문제 정의 방법2 (순열 기반 모델)
  - 각 queen은 특정 열에만 배치
  - 결정 변수 x_i : i 번째 열에서 queen이 놓인 위치(행) 의미
    - 8-queens의 경우 해의 예

7	4	1	5	2	8	6	3

(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	(1,8)
(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)	(2,8)
(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)	(3,8)
(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)	(4,8)
(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)	(5,8)
(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)	(6,8)
(7,1)	(7,2)	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)	(7,8)
(8,1)	(8,2)	(8,3)	(8,4)	(8,5)	(8,6)	(8,7)	(8,8)

각 행당 하나	$x_i \neq x_j \; for \; i,j = \{1,\ldots,n\}, i \neq j$
□와 / 대각선 당 하나 이하	$x_i - x_j \neq i - j \text{ and } x_i - x_j \neq j - i$

i, j는 행 번호

8-queens 문제에 대한 대각선

예를 들어, (2, 3)과 (5, 6)에 queen이 위치하면 제약 조건을 만족하지 않음

$$i = 2, j = 5$$

$$x_i = 3, x_j = 6$$

$$|x_i - x_j| = |i - j| = 3$$

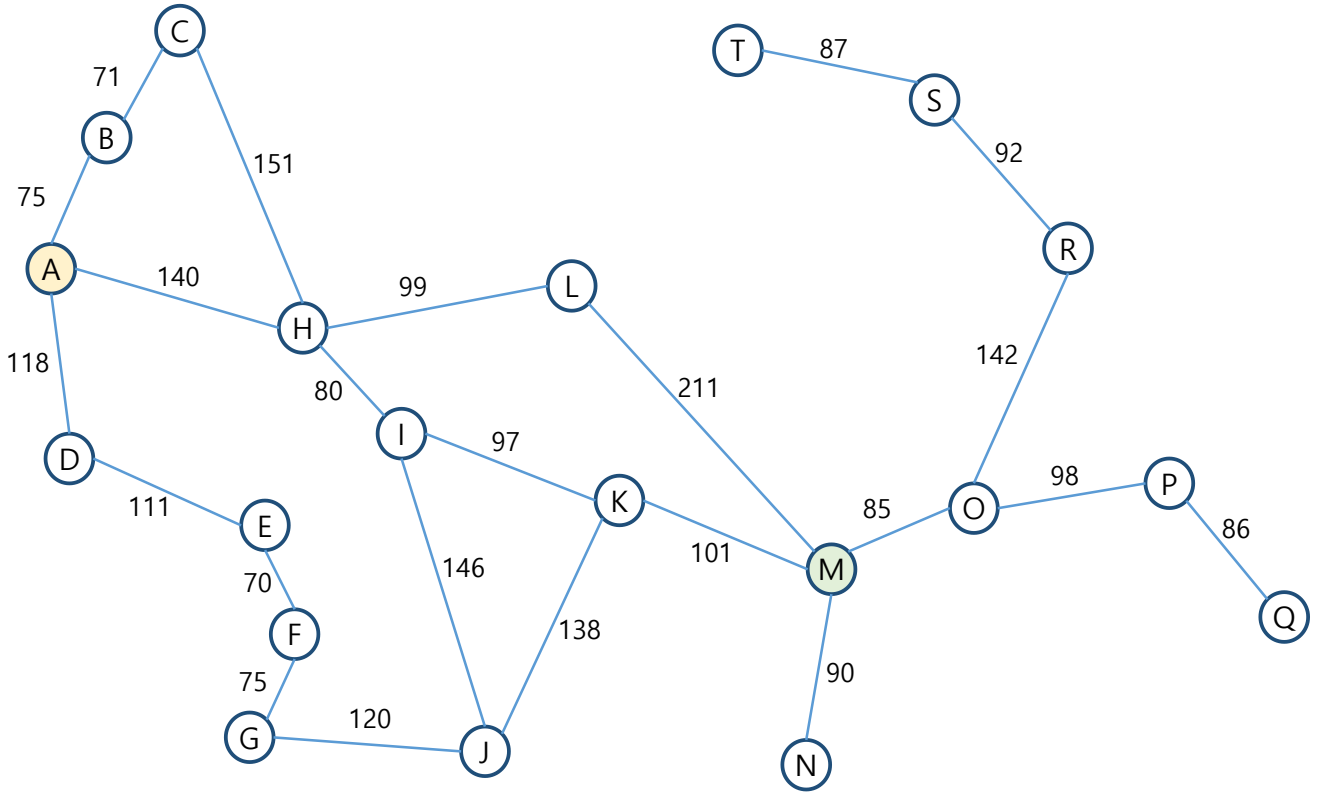
(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	(1,8)
(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)	(2,8)
(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)	(3,8)
(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)	(4,8)
(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)	(5,8)
(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)	(6,8)
(7,1)	(7,2)	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)	(7,8)
(8,1)	(8,2)	(8,3)	(8,4)	(8,5)	(8,6)	(8,7)	(8,8)

8-queens 문제 : 수학적 모델링의 중요성 (4)

- 두 가지 방법의 성능 비교
  - 방법1 : 경우의 수 = 64 x 63 x ... x 57 = 3 x 10^{14} (3백조)
  - 방법2 : 경우의 수 = 8 x 8 x ... x 8 = 8^8 3 (1677만)
  - 방법2로 모델링하는 것이 탐색 성능 우수

최단 경로 찾기 문제

- 본 수업에서 소개할 다양한 탐색 알고리즘의 대상 문제
  - 초기 상태(출발 도시): A, 목표 상태(최종 도시): M
  - 비용 함수: 거리의 합
  - 최단 경로는?



조합 최적화 문제 (1)

- 조합 최적화 문제
  - 하나의 탐색 문제가 아니라 특정 탐색 문제들의 그룹
- 조합 최적화 문제의 모델링
  - 결정 변수 집합 $X = \{x_1, \ldots, x_n\}$
  - 각 변수의 도메인 $D = \{D_1, \ldots, D_n\}$
    - 결정 변수가 가질 수 있는 값들의 집합
  - 변수들 사이의 제약 조건들
    - 제약 조건들을 만족하는 결정 변수의 값들이 하나의 해(solution)가 됨
  - 최대화 또는 최소화를 위한 목적 함수

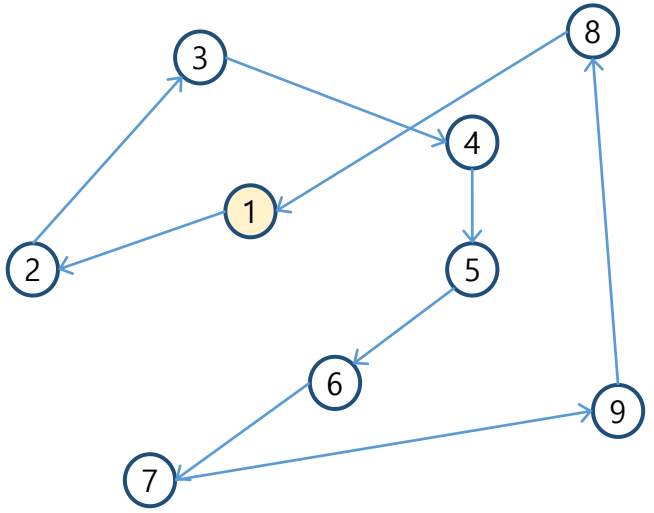
조합 최적화 문제 (2)

- 8-queens 문제 (모델링 방법 2: 순열 기반 모델)
  - 결정 변수 집합 $X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$
    - 각 열 별 하나의 queen
  - 각 변수의 도메인 $D_i = \{1, 2, 3, 4, 5, 6, 7, 8\}$
    - 각 변수(열)가 놓일 수 있는 행 위치로 모든 결정 변수의 도메인은 동일함
  - 변수들 사이의 제약 조건들
    - 행 당 하나의 queen 배치 및 대각선 당 하나의 queen 배치
  - 최대화 또는 최소화를 위한 목적 함수
    - 직선 위치에 있는 queen 쌍의 개수 최소화
- 조합 최적화 문제 : 순회 외판원 문제, 배낭 문제, 집합 커버링 문제 등

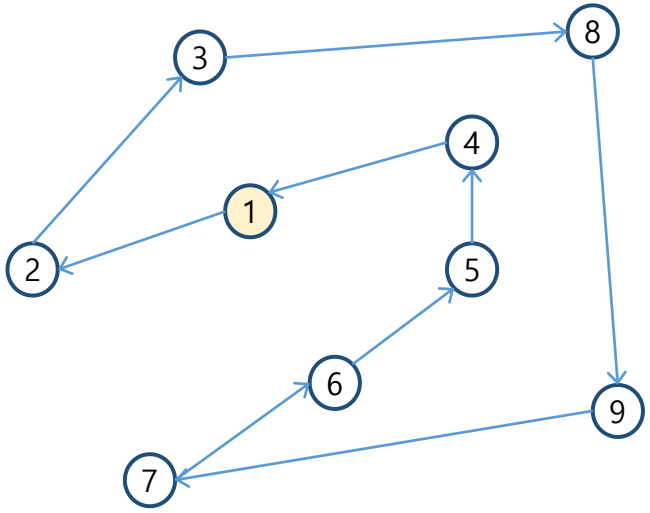
순회 외판원 문제 (1)

- n개의 모든 도시를 단 한 번씩 방문하여 다시 출발 도시로 되돌아오되 방문 거리를 최소화
  - 예 : 9개의 도시로 이루어진 순회 외판원 문제
  - 경우의 수 : 9! (출발 도시를 고정하면 8!) ⇒ 40,320개 경로

도시의 개수가 21개
⇒ 20! (2,432,902,008,176,640,000)
⇒ 초당 100만개 경로 평가 (77,146년)



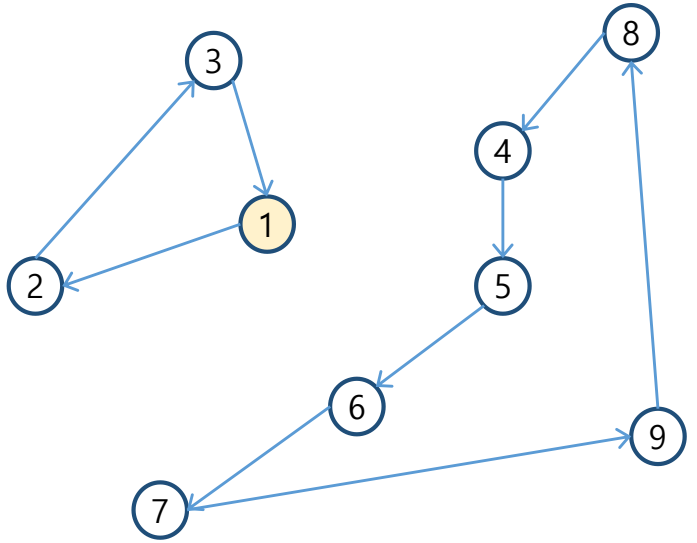
(a) 하나의 해



(b) 또 다른 해

순회 외판원 문제 (2)

- Subtour(서브투어)
  - 전체 도시를 한 번씩 모두 방문하는 단일 경로가 아니라 도시 집합이 둘 이상으로 나누어져 각각 별도의 작은 경로를 만드는 현상
  - Subtour가 있다면 답이 여러 개의 경로로 쪼개져서 모든 도시를 한 번만 방문하지만 전체를 한 번에 연결하지 못하는 문제 발생



순회 외판원 문제 (3)

- 수학적 모델링

x_{ij}: 도시 i로부터 도시 j로 방문한 경우 1

c_{ij}: 도시 i로부터 도시 j로 가는 이동 비용

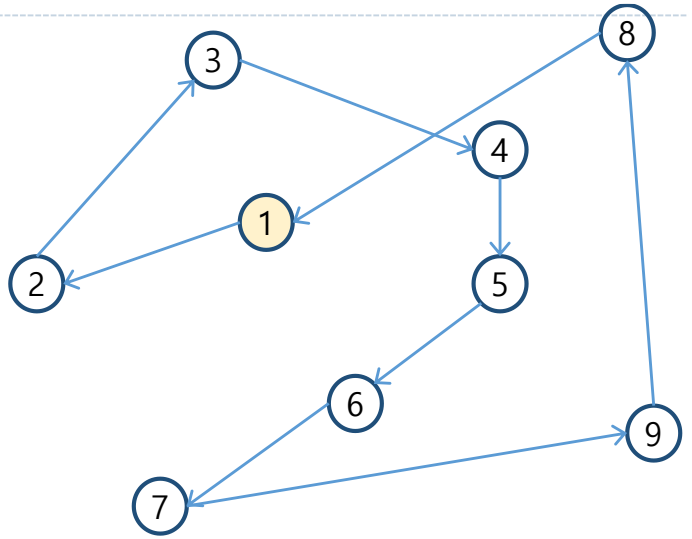
$$min \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} c_{ij} x_{ij}$$

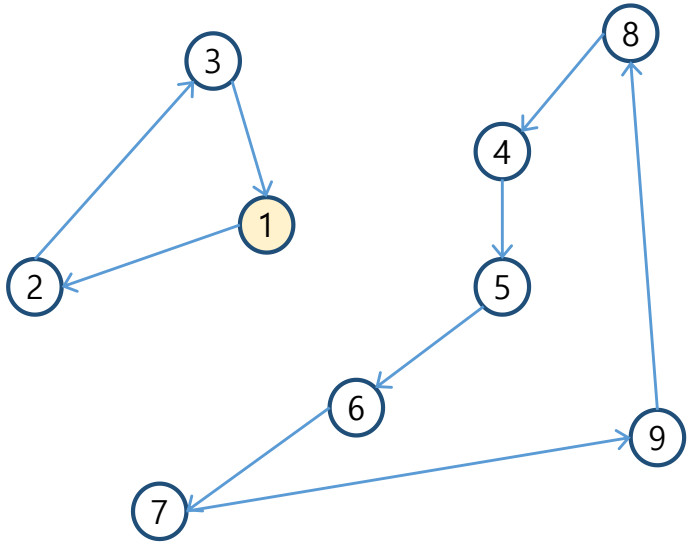
$$\sum_{j=1, j \neq i}^{n} x_{ij} = 1 \ (i = 1, \cdots, n)$$

$$\sum_{i=1, i \neq j}^{n} x_{ij} = 1 \ (j = 1, \cdots, n)$$

$$\sum_{i \in V} \sum_{j \in W} x_{ij} \geq 1, all\ disjoint\ V, W \subset \{1, \ldots, n\}$$

$$x_{ij} \in \{0, 1\}$$





배낭 문제

- 하나의 배낭과 여러 개의 아이템들이 주어지며, 각 아이템은 무게(w)와 값(p)을 가지고 있다. 배낭에 담을 수 있는 최대 무게(W)를 넘지 않는 한도 내에서 값의 합을 최대화할 수 있도록 아이템들을 배낭에 담아보라.
  - 9개 아이템으로 이루어진 배낭 문제에 대한 해의 예
  - 경우의 수 : 2^9

1	0	0	1	1	0	1	0	0

- 수학적 모델링

$$Maximize \quad z = \sum_{j=1}^{n} p_j x_j$$

$$Subject\ to \sum_{j=1}^{n} w_j x_j \leq W$$

$$x_j \in \{0,1\}, j = 1, \ldots, n$$

다차원 배낭 문제

- 하나의 배낭과 n개의 아이템으로 구성되며 각 아이템에는 값(p)과 서로 다른 m개의 무게(w_i)가 부여되어 있다. 여기서 값들의 합을 최대화하도록 아이템들을 선택하여 배낭에 담되, 선택된 아이템들의 각 무게 별로 해당 무게의 합이 주어진 최대값(W_i)을 초과하지 않아야 한다.

아이템 번호	값(p)	무게1(w_1)	무게2(w_2)	무게3(w_3)
1	640	766	928	644
2	336	384	599	385
3	476	634	530	411
4	458	54	181	384
5	904	146	267	760
최대 무게	-	992	1252	1292

$$Maximize \quad z = \sum_{j=1}^{n} p_j x_j$$

$$Subject\ to \sum_{j=1}^{n} w_{ij} x_j \leq W_i, i = 1, \ldots, m$$

$$x_j \in \{0,1\}, j = 1, \ldots, n$$

집합 커버링 문제

- m행 n열의 0-1 행렬이 주어질 때 열들의 부분집합을 선택하여 모든 행들을 커버하되 선택된 열들의 비용(c_j)을 최소화
  - 예제

	열1	열2	열3	열4	열5	열6	열7	열8
행1	1	1		1				
행2			1			1	1	
행3	1		1					1
행4				1	1		1	
행5		1	1	1				1
열의 비용	1	2	4	3	1	1	2	1

- 모델링

$$Minimize \quad z = \sum_{j=1}^{n} c_j x_j$$

$$Subject\ to \sum_{j=1}^{n} a_{ij} x_j \geq 1, i = 1, \dots, m$$

$$x_j \in \{0,1\}, j = 1, \dots, n$$

최대 커버링 문제

- m행 n열의 0-1 행렬이 주어질 때 n개의 열들 중 d개의 열을 선택하되 커버되는 행의 개수를 최대화
  - 예제

	열1	열2	열3	열4
행1	1	1		
행2		1	1	1
행3				
행4	1		1	
행5				1

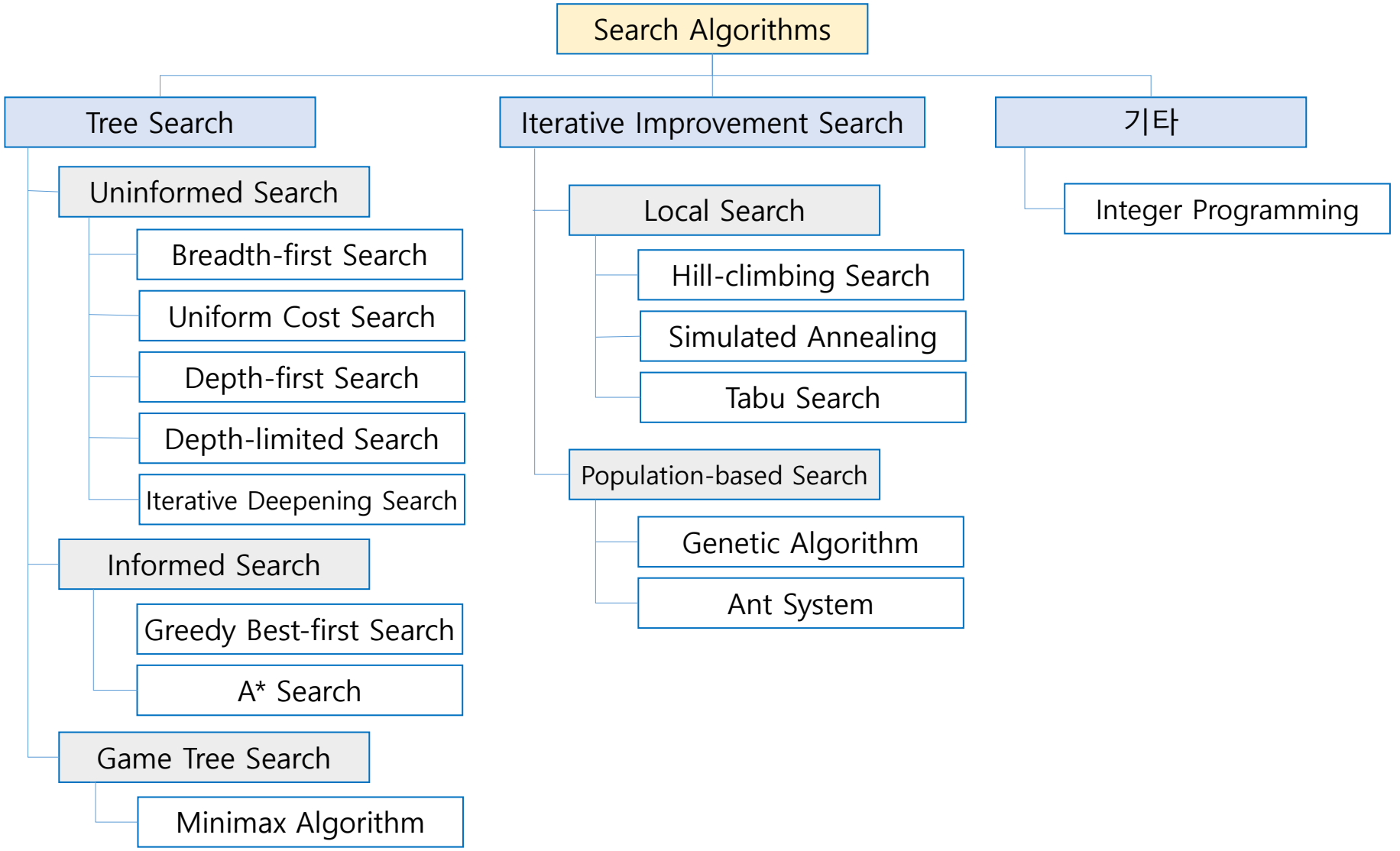
- 모델링

$$Maximize \sum_{i=1}^{m} y_i \qquad Subject\ to \sum_{j=1}^{n} x_j = d \qquad \sum_{j=1}^{n} a_{ij}x_j - y_i \geq 0, i = 1, \dots m$$

$$x_j \in \{0,1\}, j = 1, \dots n \qquad y_i \in \{0,1\}, i = 1, \dots m$$

탐색 알고리즘 종류

탐색 알고리즘의 종류


Search Algorithms
Tree Search
Uninformed Search
Breadth-first Search
Uniform Cost Search
Depth-first Search
Depth-limited Search
Iterative Deepening Search
Informed Search
Greedy Best-first Search
A* Search
Game Tree Search
Minimax Algorithm
Iterative Improvement Search
Local Search
Hill-climbing Search
Simulated Annealing
Tabu Search
Population-based Search
Genetic Algorithm
Ant System
기타
Integer Programming

요약

- 탐색을 통한 문제 해결
  - "초기 상태로부터 목표 상태까지 갈 수 있는 일련의 연산자를 찾는 것" 또는 "목표 상태 자체를 찾는 것"으로 정의
- 문제 해결을 위한 문제 정의
  - 초기 상태, 목표 상태, 연산자, 목적 함수를 명확히 기술해야 함
- 최적해
  - 해는 초기 상태부터 목표 상태까지의 경로를 나타내며, 이들 중 목적 함수값이 가장 좋은 해를 최적해
- 탐색 알고리즘 성능 평가
  - 효과성, 해의 질(path cost), 효율성(search cost)을 종합적으로 고려
- 대표적인 탐색 문제
  - 8-puzzle 문제, 8-queens 문제, 최단 경로 찾기 문제, 순회 외판원 문제, 배낭 문제, 집합 커버링 문제 등