아래 자료를 기반으로, "탐색을 통한 문제 해결"을 빠르게 복습·발표할 수 있도록 핵심만 정리했습니다. (기존 슬라이드의 기호·모양 요소는 제거하고 평문으로 통일)

1) 핵심 개념 한 줄 정의

- 탐색(Search): 여러 가능한 경로(상태 전이) 중에서 목적함수에 따라 "가장 좋은" 해를 찾는 과정
- 상태(State)·연산자(Operator): 문제를 그래프로 본다면 정점=상태, 간선=연산자(상태 이동)
- 경로비용(Path cost) vs 탐색비용(Search cost): 해 자체의 비용과 그 해를 찾는 데 든 시간·메모리
- 총비용(Total cost) = Path cost + Search cost
- 최적해: 목적함수값이 가능한 해들 중 최소(또는 최대)

2) 문제 정의 체크리스트

1. 초기 상태, 목표 상태(또는 제약조건으로 목표 상태의 성질)
2. 연산자(상태 전이 규칙)
3. 목적함수: 비용(최소화) 또는 이득(최대화)
4. 해의 형태: 초기 상태에서 목표 상태까지의 상태·연산자 시퀀스

3) 대표 문제와 모델링 요약

■ 8-puzzle (n-puzzle)

- 상태: 보드의 타일 배치, 연산자: 공백을 상/하/좌/우로 이동(가능할 때), 비용: 한 번 이동할 때 1
- 목표: 주어진 목표 배치
- 휴리스틱 예: h_1(제자리 아닌 타일 수), h_2(맨해튼 거리의 합; h_2가 h_1을 지배)

■ 8-queens (n-queens) (방법1) 이진 변수 모델: $x_{ij} \in \{0,1\}$. 행·열·두 대각선 제약(각 방향에 1개) (방법2) 순열 모델: x_i는 i열에서의 행 위치(1..n). 서로 다른 행, $|x_i - x_j| \neq |i-j|$

- 목적함수(최적화 정식화 시): 충돌 쌍 최소화(0이면 정답)

■ 최단 경로

- 상태: 도시, 연산자: 간선 이동, 비용: 거리 합
- 목표: 출발 A → 도착 M 의 최소 비용 경로

■ TSP(순회 외판원)

- 모든 도시를 한 번씩 방문하고 출발지로 복귀, 거리 합 최소화
- 이진변수 x_{ij}와 유량/서브투어 제거 제약 등으로 수학적 모델링 가능

■ 배낭 문제(0/1)

- 각 아이템 i에 가치 p_i, 무게 w_i. $\sum w_i x_i \leq W$, $\sum p_i x_i$ 최대화, $x_i \in \{0,1\}$
- 다차원 배낭: 여러 자원 제약(무게가 여러 종류)

■ 집합 커버링 / 최대 커버링

- 주어진 0-1 행렬에서 열을 골라 모든 행을 커버(비용 최소) / 제한 개수로 커버 수 최대

4) n-puzzle 가역성(초기 상태 생성 시 주의)

- 홀수 n(예: 3×3): inversion 수가 짝수여야 목표로 도달 가능
- 짝수 n(예: 4×4): inversion 짝·홀과 "공백의 아래쪽에서부터의 행 번호" 짝·홀이 서로 달라야 함 즉, inversion의 홀짝 ⊕ 공백행(아래에서 센 번호)의 홀짝 = 1 이면 가역 (간단 버전으로 "짝수 n은 inversion이 홀수"라고만 말하면 일반식과 어긋날 수 있어 주의)

5) 탐색 알고리즘 분류와 성질 요약

알고리즘	완전성	최적성	시간복잡도(대략)	공간복잡도	특징/언제 쓰나
DFS	무한깊이면 보장X	보장X	O(b^m)	O(m)	메모리 적게, 해가 깊은 곳에 있을 때 가끔 유리
BFS	보장	단위비용일 때 보장	O(b^{d+1})	O(b^{d+1})	얕은 최단해 확보, 메모리 큼
Uniform-Cost	보장	항상 보장	상태 수·간선 가중치에 의존	큼	가중치 있을 때의 "최단경로"(다익스트라 관점)
Depth-Limited	제한 내	보장X	O(b^ℓ)	O(b·ℓ)	DFS에 깊이 제한
Iterative Deepening	보장	단위비용일 때 보장	O(b^d)	O(b·d)	BFS 최단해 + DFS 저메모리의 절충
Greedy Best-First	보장X	보장X	휴리스틱 품질에 의존	작지 않음	빠른 근사, 길 잃기 쉬움
A*	보장	h가 admissible(낙관적)일 때 보장, consistent면 더 효율	상태 공간에 의존	큼	실용적 표준. 휴리스틱이 핵심(지배성이 좋을수록 노드 전개 감소)

휴리스틱 성질

- Admissible: 실제 비용을 절대 초과하지 않음 → A*의 최적성 보장
- Consistent(삼각부등식): h(n) ≤ c(n,a,n′) + h(n′) → 닫힌 집합에 들어간 노드는 재확인 불필요
- 지배성: $h_2(n) \geq h_1(n)$ ∀n 이고 둘 다 admissible이면 h_2가 더 효율적

6) 모델링 관점에서의 포인트

- 같은 문제라도 모델링에 따라 탐색공간 크기가 급격히 달라짐 예) 8-queens 방법1: 64×63×…×57 ≈ 3×10^{14} 방법2: 8^8 = 16,777,216 순열 모델(방법2)이 탐색 성능상 훨씬 유리
- 탐색 전 "문제 정의"와 "모델링"을 잘하면 탐색 자체가 훨씬 쉬워짐

7) 성능 평가 관점

- 효과성: 결국 해를 찾을 수 있는가(완전성)
- 해의 질: 찾은 해가 얼마나 좋은가(최적성/경로비용)
- 효율성: 시간·메모리(탐색비용)
- 목적에 따라 균형 선택: 100일 걸리는 최적해 vs 1시간 내 준최적해

8) 발표용 짧은 흐름 예시(3–4분 요약)

1. 탐색의 정의와 상태/연산자/비용 함수 개념
2. 8-puzzle, 8-queens, 최단 경로로 문제 정의 예시
3. n-puzzle 가역성 규칙과 휴리스틱(A*에서 왜 중요한지)
4. 알고리즘 스펙트럼(BFS/DFS/UCS/IDS/Greedy/A*)과 선택 기준
5. 모델링의 힘(8-queens 두 모델의 경우의 수 비교)
6. 총비용 관점에서 "좋은 알고리즘"은 상황에 따라 다름

9) 빠른 확인 문제(구두 체크)

- Q1. 8-puzzle에서 h_1과 h_2 중 어느 휴리스틱이 더 강한가? 왜인가? A. 맨해튼 거리 h_2가 h_1을 지배한다. 타일 이동 최소 횟수의 하한을 더 촘촘히 반영하기 때문.
- Q2. A*가 최적 보장하려면 휴리스틱에 어떤 성질이 필요한가? A. 적어도 admissible. 일관성까지 만족하면 더 효율적이다.
- Q3. 4×4 퍼즐의 가역성 판단에 무엇이 필요한가? A. inversion의 홀짝과 공백의 아래에서부터의 행 번호 홀짝이 달라야 한다.
- Q4. 8-queens에서 순열 모델이 유리한 이유는? A. 제약을 모델에 직접 반영해 탐색공간을 크게 줄였기 때문이다.

필요하면 위 내용을 "슬라이드 10~12장 분량" 또는 "한 페이지 요약본(PDF)"으로 깔끔하게 다시 정리해 드릴게요.