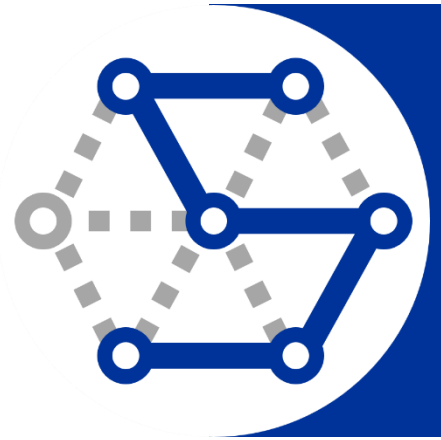

인공지능
Artificial Intelligence

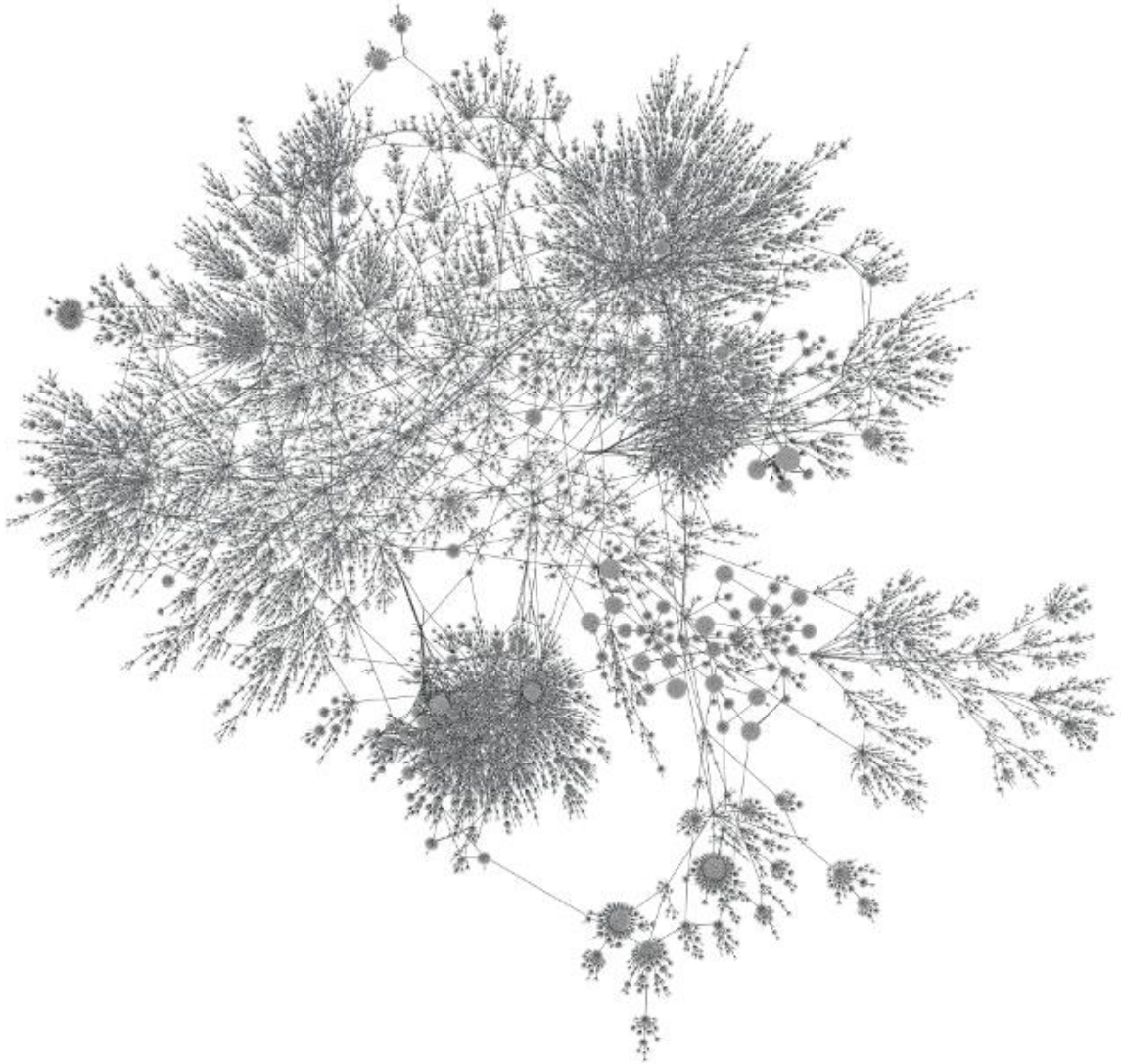

03
탐색



1 상태 공간과 탐색

탐색 (探索, search)

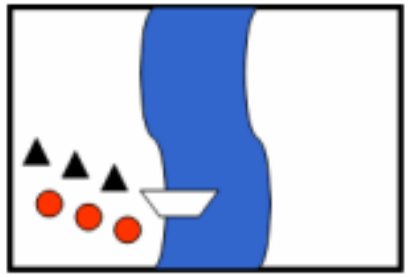
- 문제의 **해(solution)**이 될 수 있는 것들의 집합을 **공간(space)**으로 간주하고, 문제에 대한 **최적의 해**를 찾기 위해 공간을 **체계적으로 찾아 보는 것**
- 인공지능 시스템이 문제해결을 위해서 흔히 사용하는 기법
- 만약 우리가 문제 해결을 위해서 취해야 할 행동들이 무엇인지 알고 있지만 어떤 순서로 행동을 취해야 문제가 해결되는지 알지 못한다면 가능한 모든 순서의 조합을 다 시도해 보아야 함



이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

탐색문제의 예

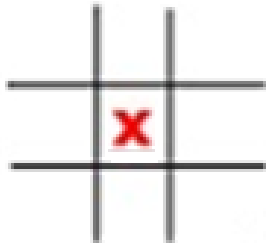
강 건너기 문제



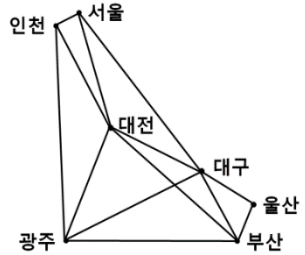
8-퍼즐(puzzle) 문제

8-퀸(queen) 문제

틱-택-토 (tic-tac-toe)



순회 판매자 문제 (traveling salesperson problem, TSP)



루빅스큐브 (Rubik's cube)

이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

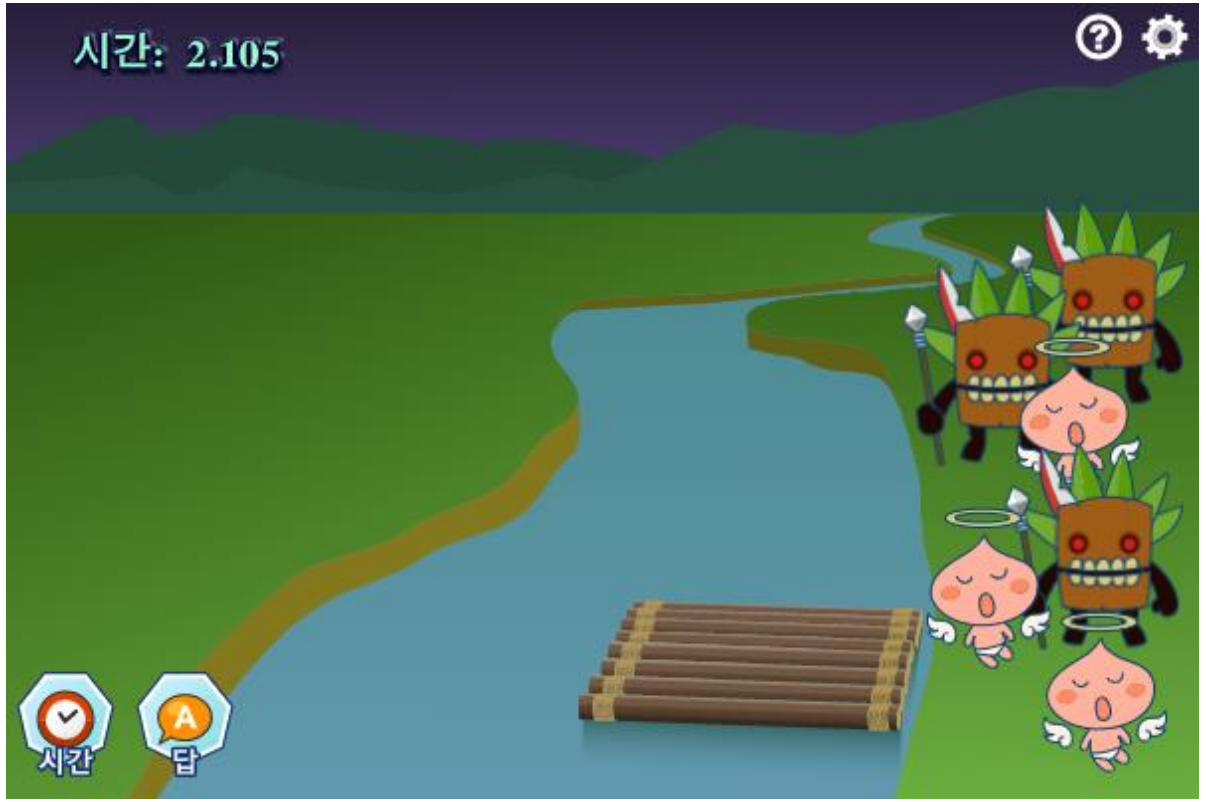
강 건너기 문제 - 늑대, 양, 풀

- 늑대, 양, 풀이 있는데 한 번에 하나씩 밖에 옮길 수 없고, 사람이 없으면 늑대는 양을, 양은 풀을 먹어버린다
- 모두 무사히 건너기 위해서는 어떻게 해야할까?

https://www.novelgames.com/ko/wolf/

강 건너기 문제 - 식인종과 선교사

- 선교사와 식인종이 각각 3명씩 있으며 강을 건너려고 한다
- 선교사가 식인종보다 많거나 같으면 문제가 없지만, 식인종이 선교사보다 많아지면 식인종은 선교사를 먹는다
- 또한 배에는 종류를 막론하고 2명까지 탈 수 있다
- 아무도 죽지 않으면서 모두 강을 건너려면 어떻게 해야 할까?



https://www.novelgames.com/ko/missionaries/

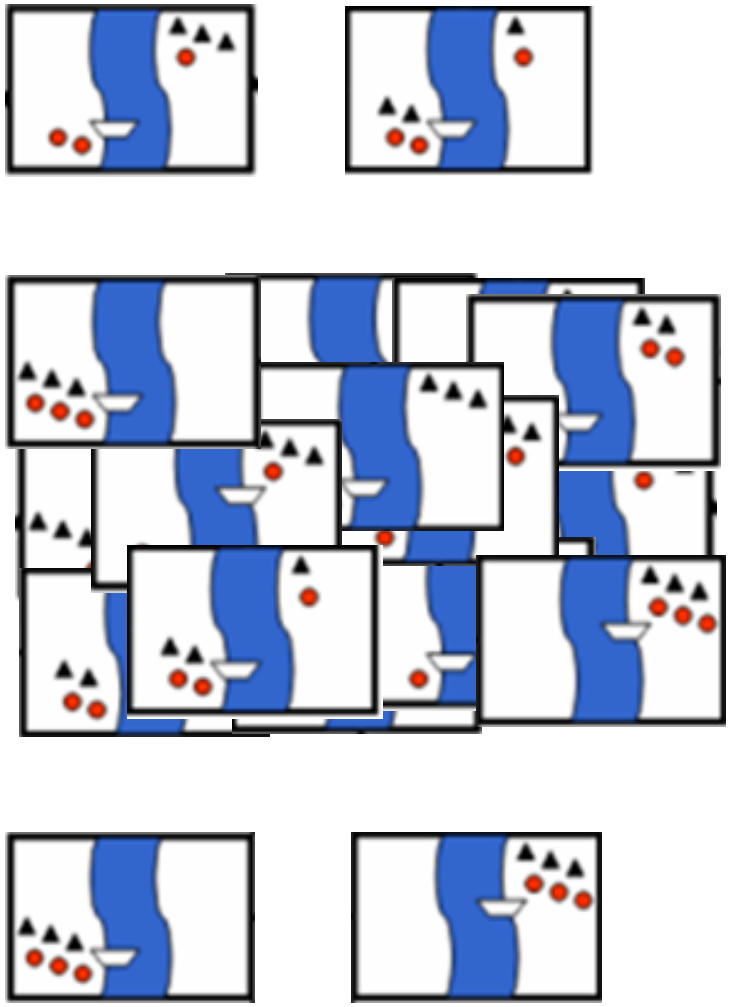
강 건너기 문제 – 콩가루 가족

- 뗏목은 최대 2명까지 탈 수 있다
- 엄마가 없으면 아빠는 여자아이를 때린다
- 아빠가 없으면 엄마가 남자아이를 때린다
- 죄수는 경찰관이 없으면 모든 사람을 때린다
- 아이가 배에 탈 경우 어른과 동행해야 한다
- 아무도 다치지 않고 강을 건너야 한다. 어떻하면 모두 무사히 건널 수 있을까요?

https://vidkidz.tistory.com/713

상태 공간과 탐색

- 상태(state): 특정 시점에 문제의 세계가 처해 있는 모습
- 세계(world): 문제에 포함된 대상들과 이들의 상황을 포괄적으로 지칭
- 상태 공간(state space)
  - 탐색공간(search space): 문제 해결 과정에서 초기 상태로부터 도달할 수 있는 모든 상태들의 집합
  - 초기 상태(initial state): 문제가 주어진 시점의 시작 상태
  - 목표 상태(goal state): 문제에서 원하는 최종 상태

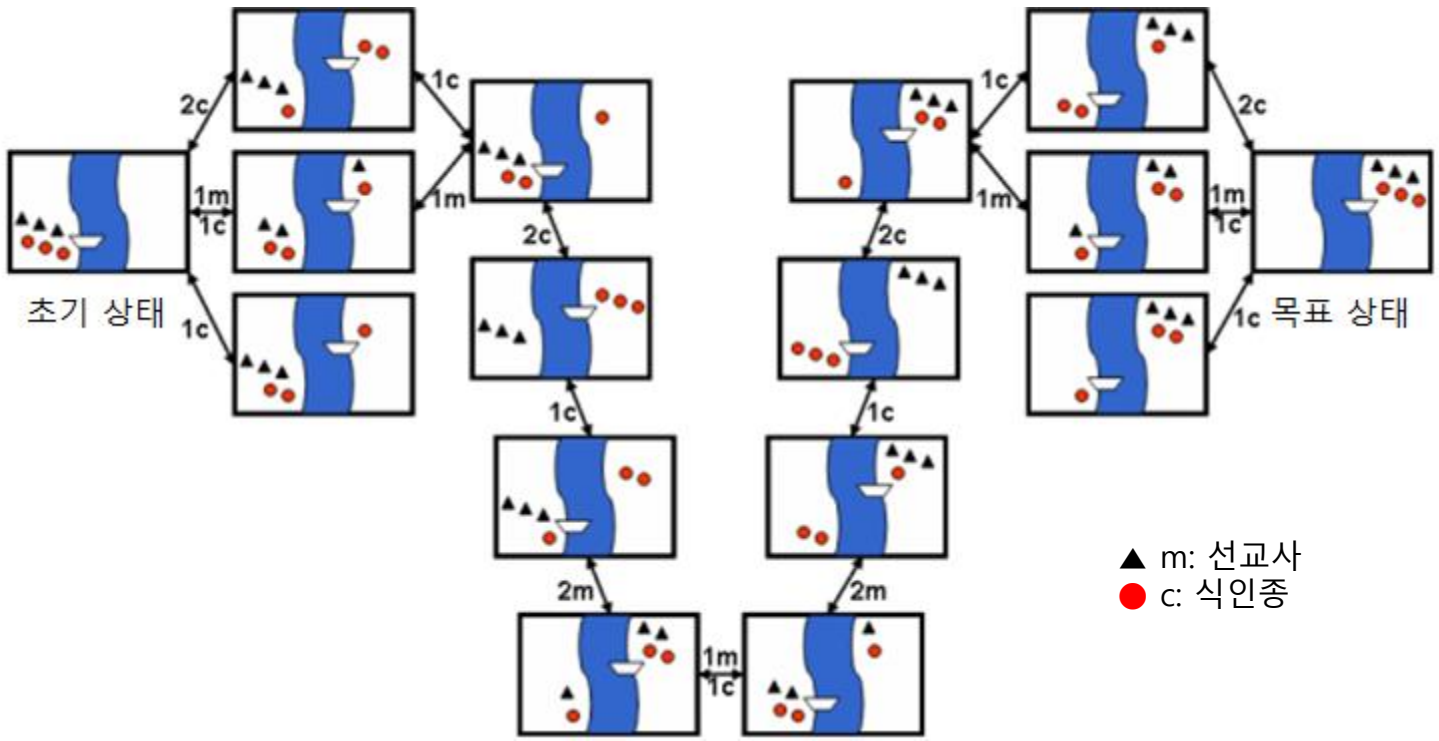


이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

상태 공간 그래프(state space graph)

- 상태공간에서 각 행동에 따른 상태의 변화를 나타낸 그래프
  - 노드 : 상태
  - 링크 : 행동
- 해(solution): 초기상태에서 목표 상태로의 경로(path)
- 일반적인 문제에서는 상태공간이 매우 큼
  - 미리 상태 공간 그래프를 만들기 어려움
  - 탐색과정에서 그래프 생성

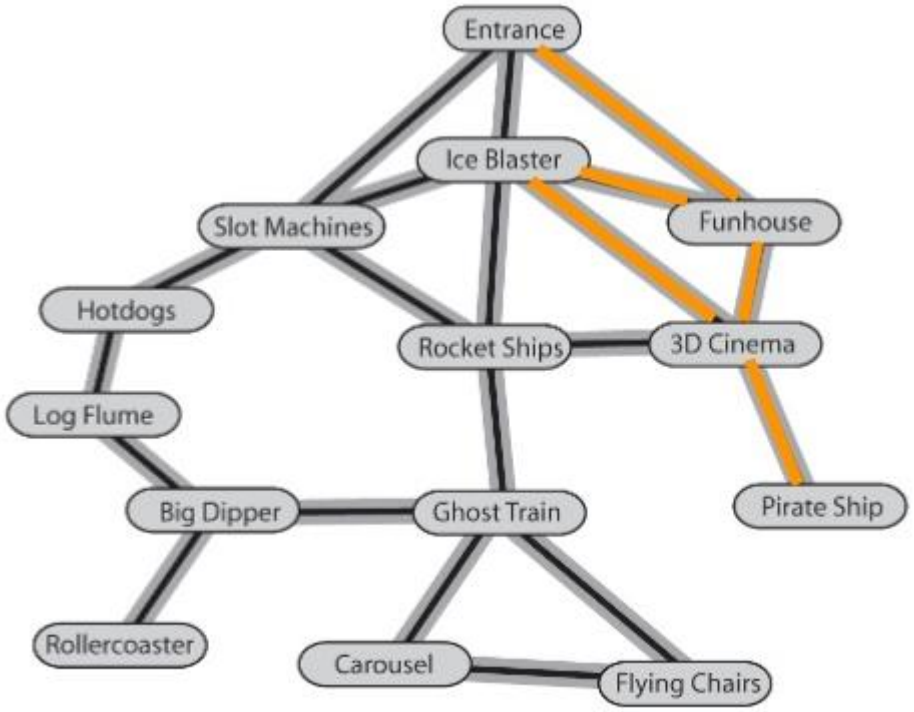
- 선교사-식인종 문제

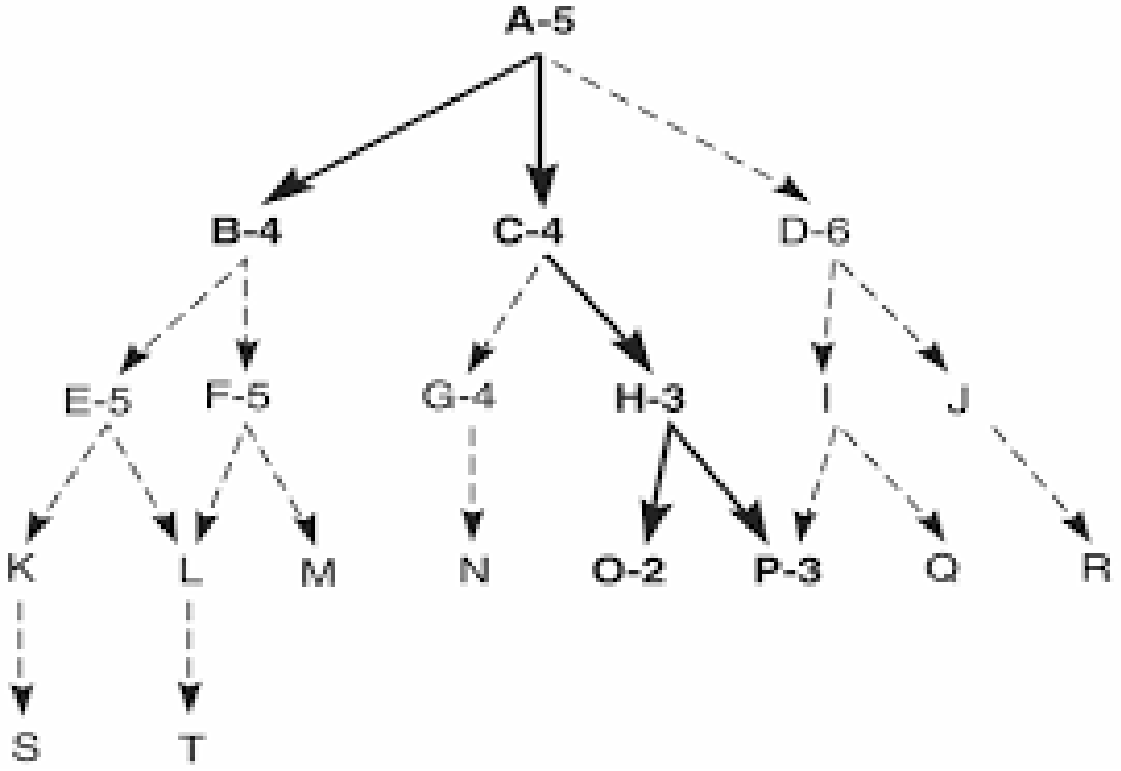


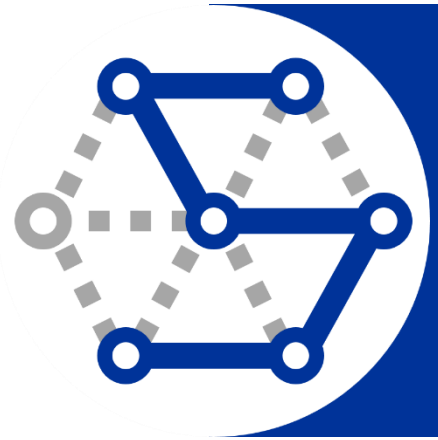
이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

탐색 방법

- 무 정보 탐색: 모든 길을 다 찾아 보는 방법
- 휴리스틱 탐색: 가능성이 높은 곳만을 선별하여 찾아 보는 방법



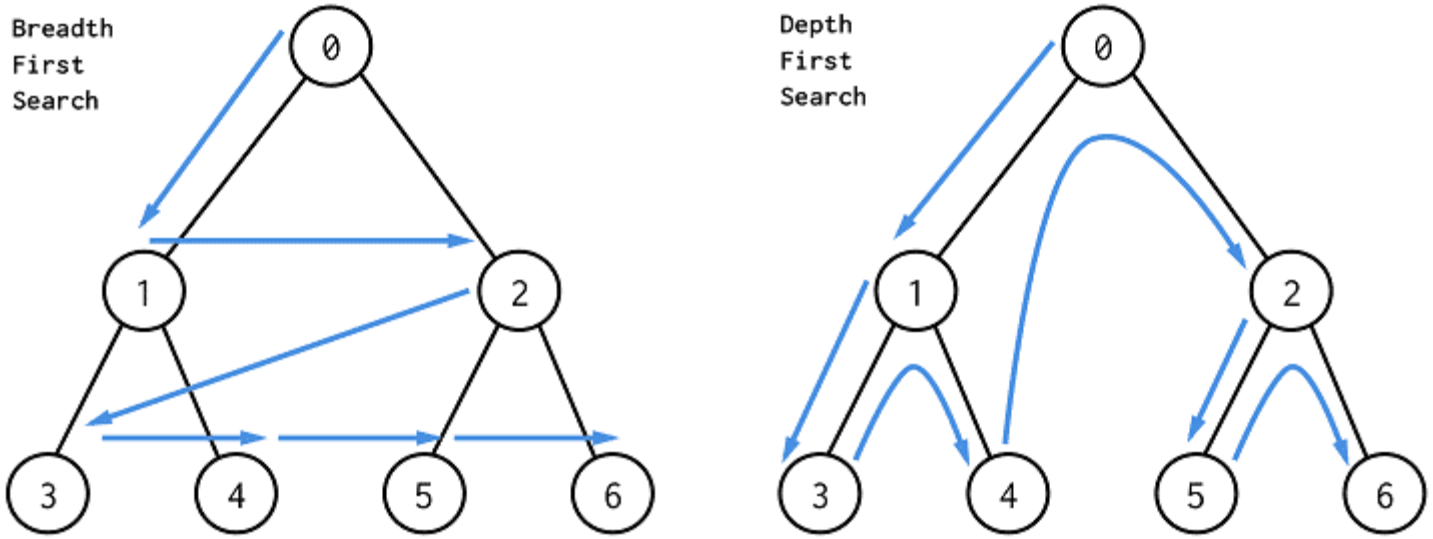




2 무정보 탐색

무정보 탐색

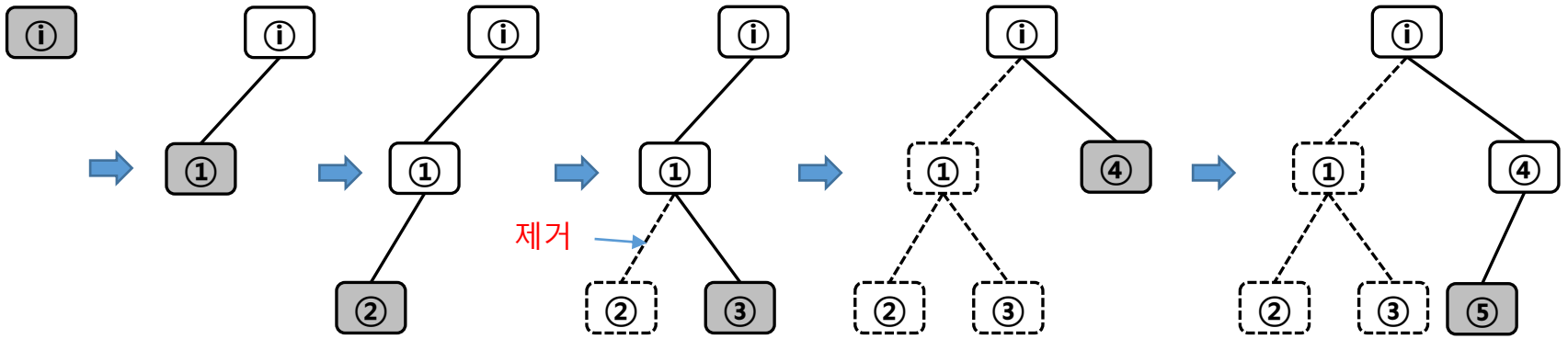
- 탐색공간에 대한 아무런 정보 없이 순서만 정해놓고 탐색을 수행
- 맹목적 탐색(Blind Search) 또는 Brute Force Search
- 종류
  - 깊이우선 탐색(DFS: Depth First Search)
  - 너비우선 탐색(BFS: Breadth First Search)



이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

깊이 우선 탐색(depth-first search, DFS)

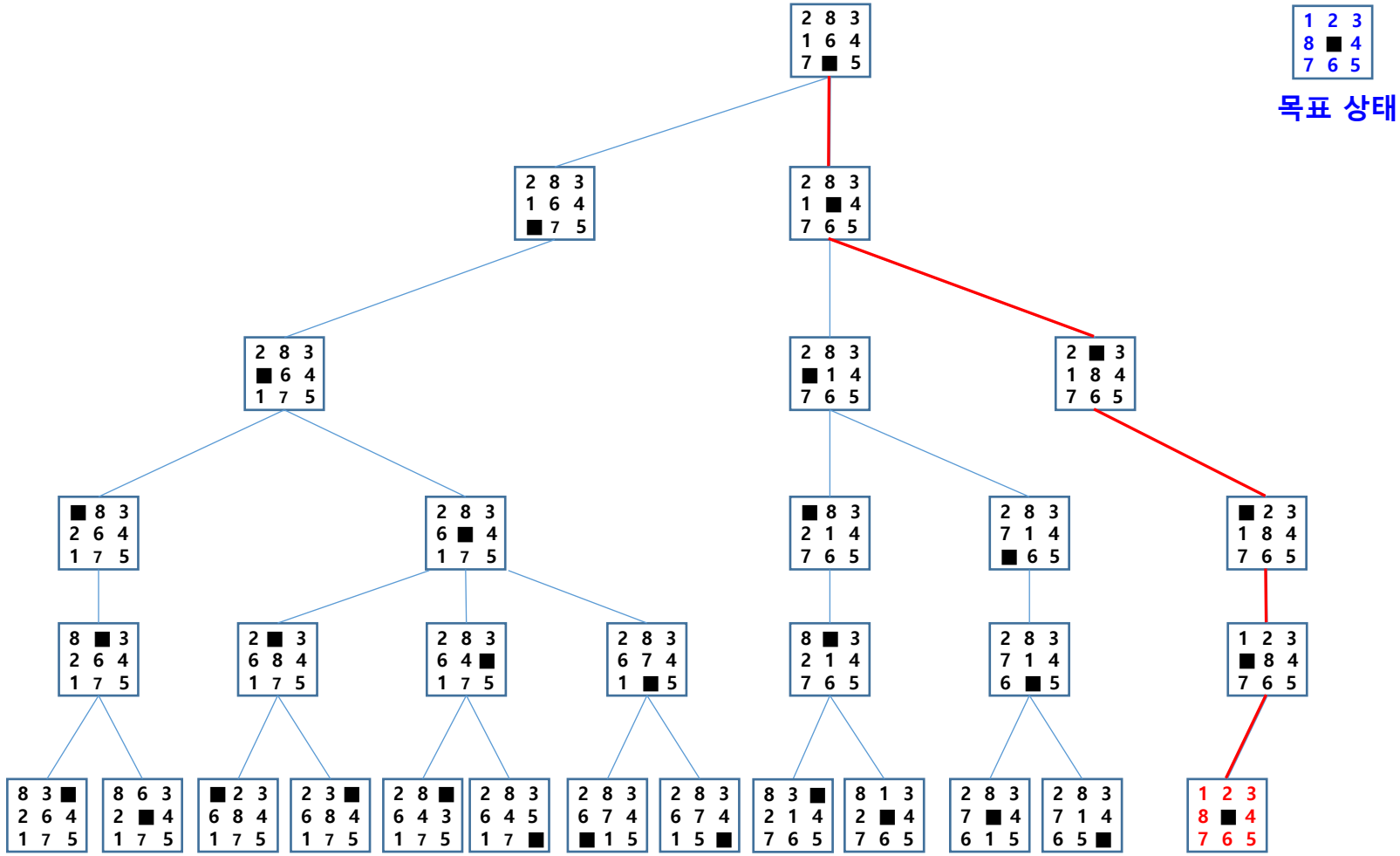
- 초기 노드에서 시작하여 깊이 방향으로 탐색
- 목표 노드에 도달하면 종료
- 더 이상 진행할 수 없으면, 백트랙킹(backtracking, 되짚어가기)
- 방문한 노드는 재방문하지 않음



이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

8-퍼즐 문제의 깊이 우선 탐색 트리

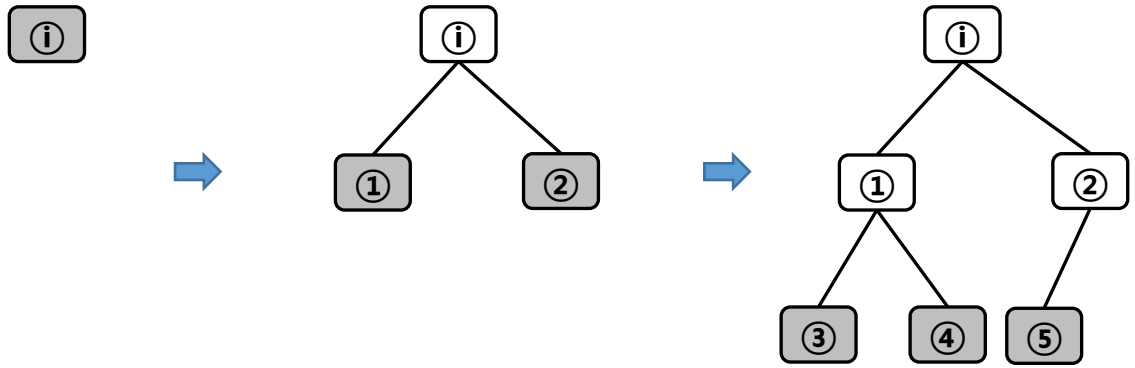
- 루트 노드에서 현재 노드까지의 **경로 하나만 유지**



이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

너비 우선 탐색(breadth-first search, BFS)

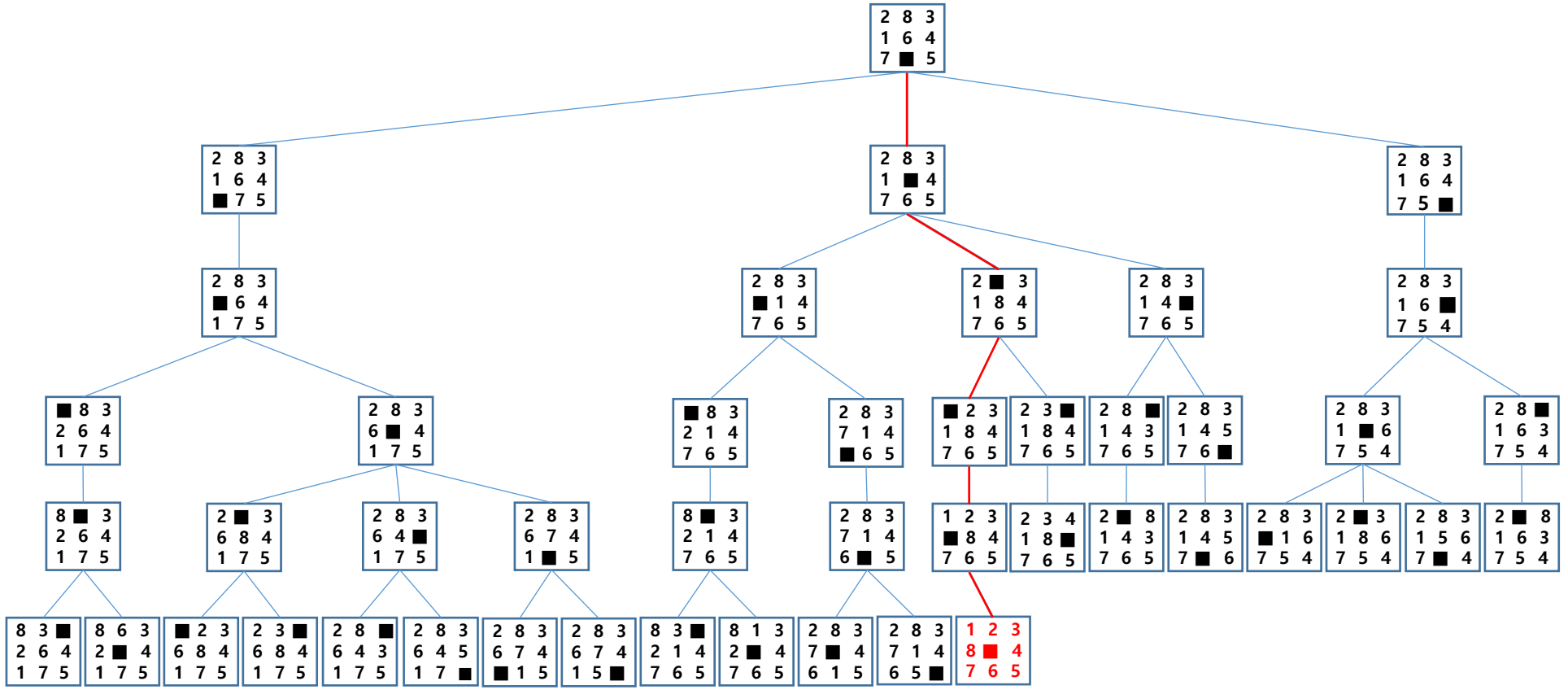
- 초기 노드에서 시작하여 **모든 자식 노드**를 **확장**하여 생성
- 목표 노드가 없으면 **단말노드**에서 다시 **자식 노드 확장**



이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

8-퍼즐 문제의 너비 우선 탐색 트리

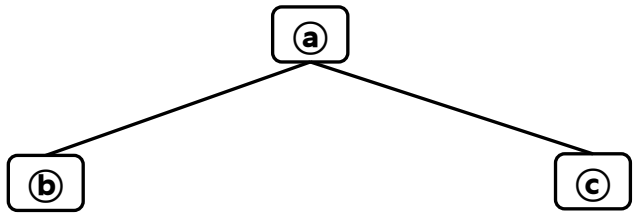
- **전체 트리**를 메모리에서 관리
- 목표 상태에 도달하는 **최단 경로 찾기** 가능



이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

반복적 깊이심화 탐색(iterative-deepening search)

- **깊이 한계**가 있는 **깊이 우선 탐색**을 **반복적**으로 적용

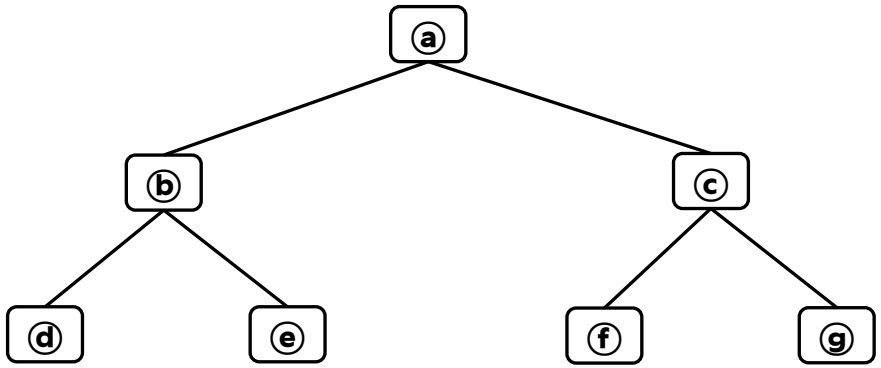


깊이 0: ⓐ
깊이 1: ⓐ,ⓑ,ⓒ

이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

반복적 깊이심화 탐색(iterative-deepening search)

- **깊이 한계**가 있는 **깊이 우선 탐색**을 **반복적**으로 적용

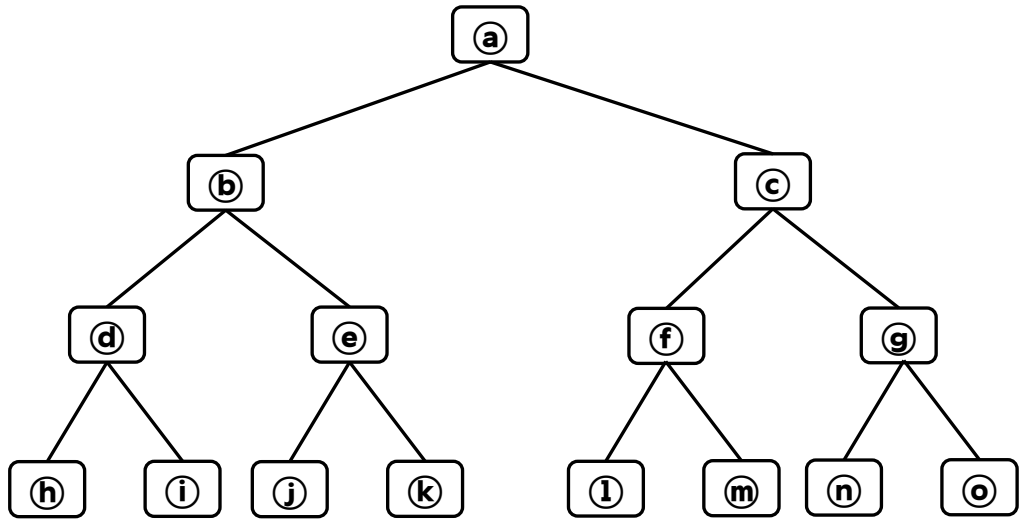


깊이 0: ⓐ
깊이 1: ⓐ,ⓑ,ⓒ
깊이 2: ⓐ,ⓑ,ⓓ,ⓔ,ⓒ,ⓕ,ⓖ

이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

반복적 깊이심화 탐색(iterative-deepening search)

- **깊이 한계**가 있는 **깊이 우선 탐색**을 **반복적**으로 적용

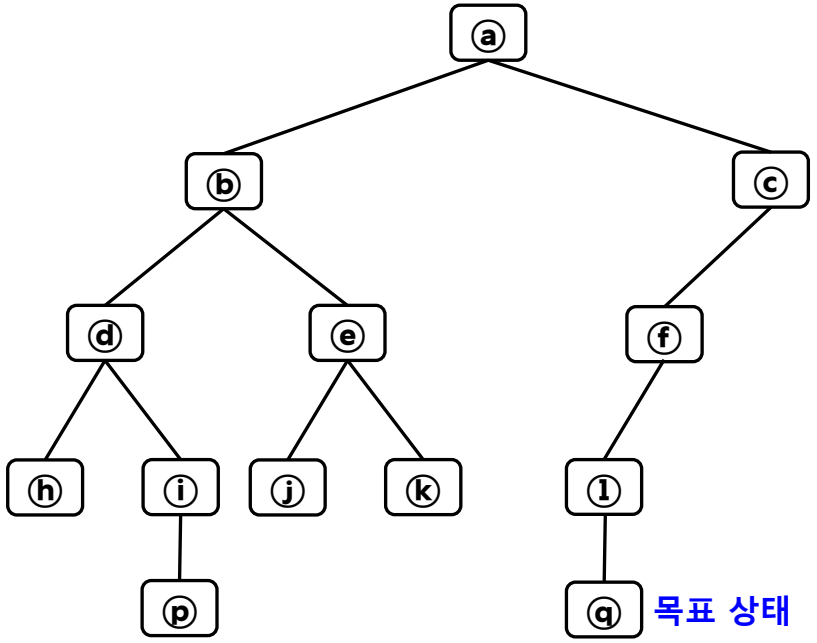


깊이 0: ⓐ
깊이 1: ⓐ,ⓑ,ⓒ
깊이 2: ⓐ,ⓑ,ⓓ,ⓔ,ⓒ,ⓕ,ⓖ
깊이 3: ⓐ,ⓑ,ⓓ,ⓗ,ⓘ,ⓔ,ⓙ,ⓚ,ⓒ,ⓕ,ⓛ,ⓜ,ⓖ,ⓝ,ⓞ

이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

반복적 깊이심화 탐색(iterative-deepening search)

- **깊이 한계**가 있는 **깊이 우선 탐색**을 **반복적**으로 적용



깊이 0: ⓐ
깊이 1: ⓐ,ⓑ,ⓒ
깊이 2: ⓐ,ⓑ,ⓓ,ⓔ,ⓒ,ⓕ,ⓖ
깊이 3: ⓐ,ⓑ,ⓓ,ⓗ,ⓘ,ⓔ,ⓙ,ⓚ,ⓒ,ⓕ,ⓛ,ⓜ,ⓖ,ⓝ,ⓞ
깊이 4: ⓐ,ⓑ,ⓓ,ⓗ,ⓘ,ⓟ,ⓔ,ⓙ,ⓚ,ⓒ,ⓕ,ⓛ,ⓠ

이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

맹목적 탐색 방법의 비교

깊이 우선 탐색

- 메모리 공간 사용 효율적
- 최단 경로 해 탐색 보장 불가

너비 우선 탐색

- 최단 경로 해 탐색 보장
- 메모리 공간 사용 비효율

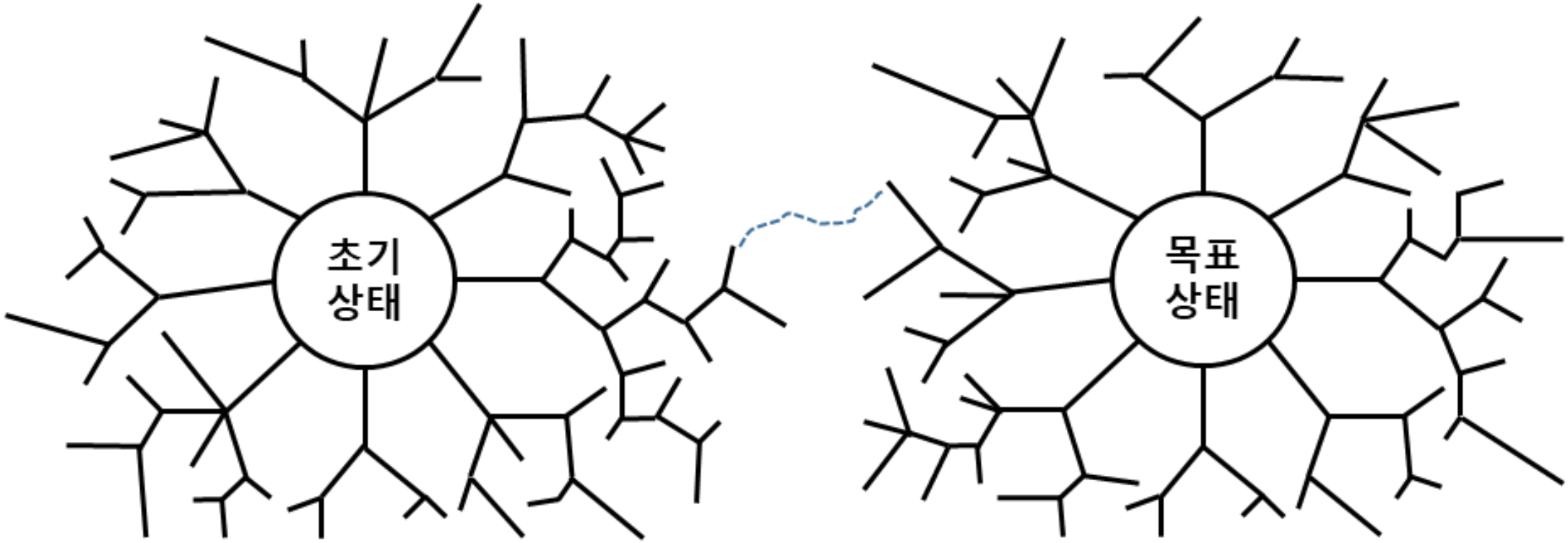
반복적 깊이심화 탐색

- 최단 경로 해 탐색 보장
- 메모리 공간 사용 효율적
- 반복적인 깊이 우선 탐색에 따른 비효율성
  - **실제 비용이 크게 늘지 않음**
  - 각 노드가 **10개의 자식노드**를 가질 때, **너비 우선 탐색 대비 약 11%정도 추가** 노드 생성
- 맹목적 탐색 적용시 우선 고려 대상

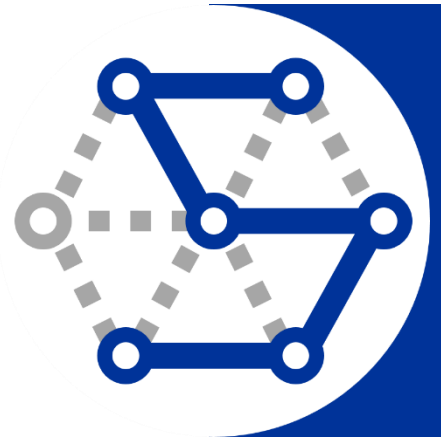
이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

양방향 탐색(bidirectional search)

- **초기 노드**와 **목적 노드**에서 **동시**에 **너비 우선 탐색**을 진행
- 중간에 만나도록 하여 초기 노드에서 목표 노드로의 최단 경로를 찾는 방법



이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판



3 정보 이용 탐색

정보이용 탐색(informed search)

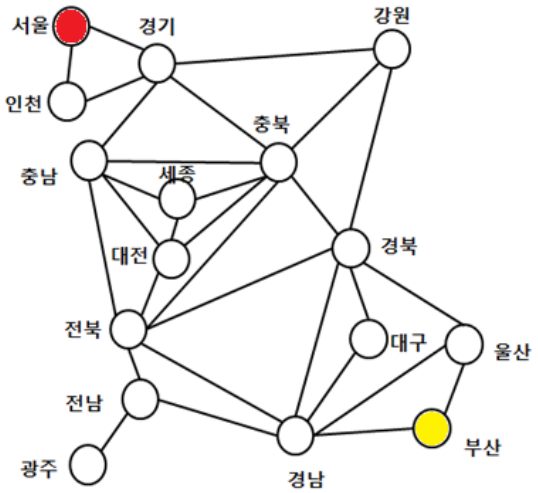
- 휴리스틱 탐색(heuristic search)
- 언덕 오르기 방법, 최상 우선 탐색, 빔 탐색, A* 알고리즘 등
- 휴리스틱(heuristic)
  - 그리스어 Εὑρίσκω (Eurisko, 찾다, 발견하다)
  - 시간이나 정보가 불충분하여 합리적인 판단을 할 수 없거나, 굳이 체계적이고 합리적인 판단을 할 필요가 없는 상황에서 신속하게 어림짐작하는 것
  - 탐색공간에 관한 정보를 탐색에 활용하여 탐색공간을 줄이거나, 정확한(최선의) 답은 아닐지라도 답으로 사용 가능한 근사치를 빨리 찾을 수 있도록 하는 유용한 정보
- 휴리스틱 예제
  - 최단 경로 문제에서 목적지까지 남은 거리 → 현재 위치에서 목적지(목표 상태)까지 지도상 직선 거리
- 휴리스틱 탐색이 유용한 두 가지 경우
  - 모호성 때문에 문제의 해가 정확히 존재하지 않을 때
  - 주어진 시간 내에 모든 탐색공간을 다 방문 할 수 없을 때

이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

휴리스틱 비용 추정의 예제

최단경로 문제

- 현재 위치에서 목적지까지 직선 거리



8-퍼즐 문제

- 제자리에 있지 않는 타일의 개수

2	8	3
1	6	4
7		5

현재 상태

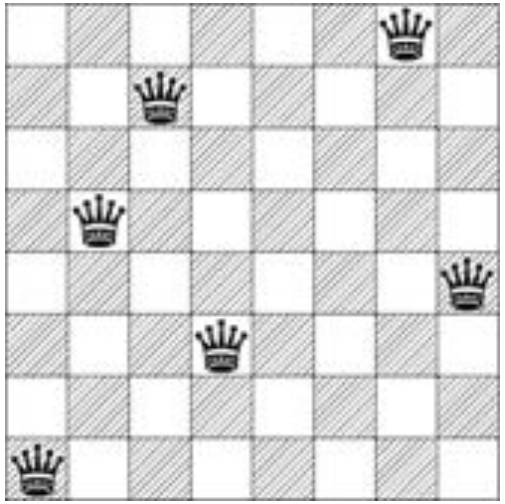
1	2	3
8		4
7	6	5

목표 상태

추정비용 : 4

8-퀸 문제

- 충돌하는 회수

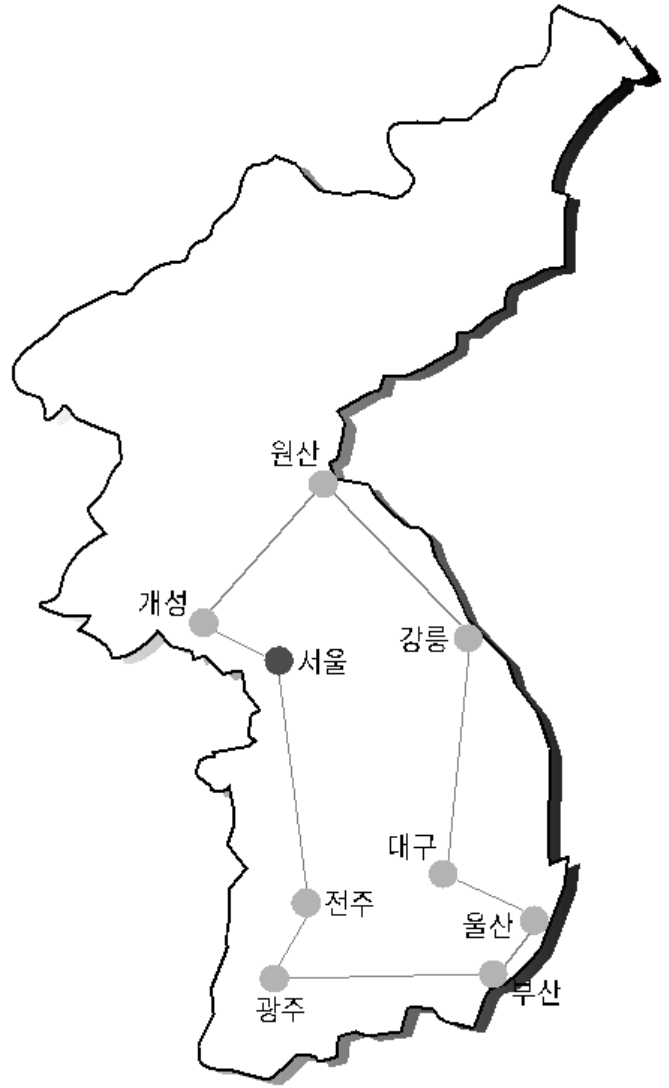


이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

방문 상인 문제(Traveling Salesman Problem)

- 모든 도시를 한 번만 방문하는 경로(해밀턴 경로)
- 도시 수 가 N 개 이면 (N – 1)! / 2 가지의 서로 다른 경로가 가능
- 도시의 수가 20개만 되더라도 가능한 조합의 수는 60,822,550,204,416,000
- 도시의 수가 많아지면 모든 경로를 정해진 시간에 다 계산해 본다는 것은 불가능

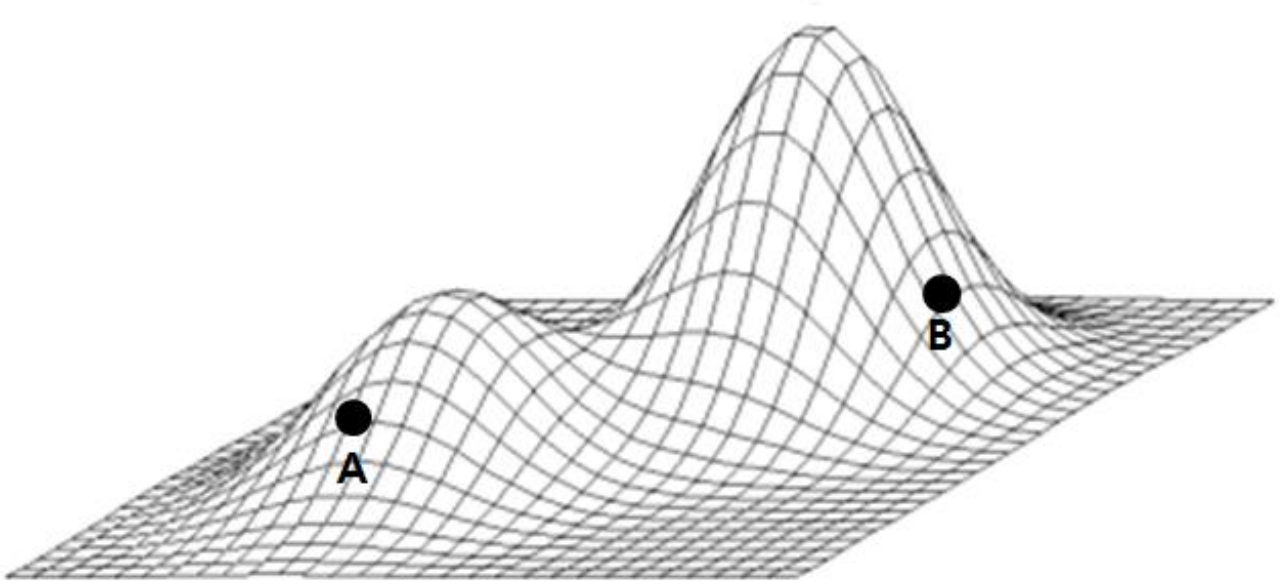
- 휴리스틱
  - 현재의 도시에서 가장 가까운 도시로 이동
  - 바다쪽으로 돌아 가면서 이동



양기철, 김명철, 인공지능 이론 및 실제, 홍릉과학출판사

언덕 오르기 방법(hill climbing method)

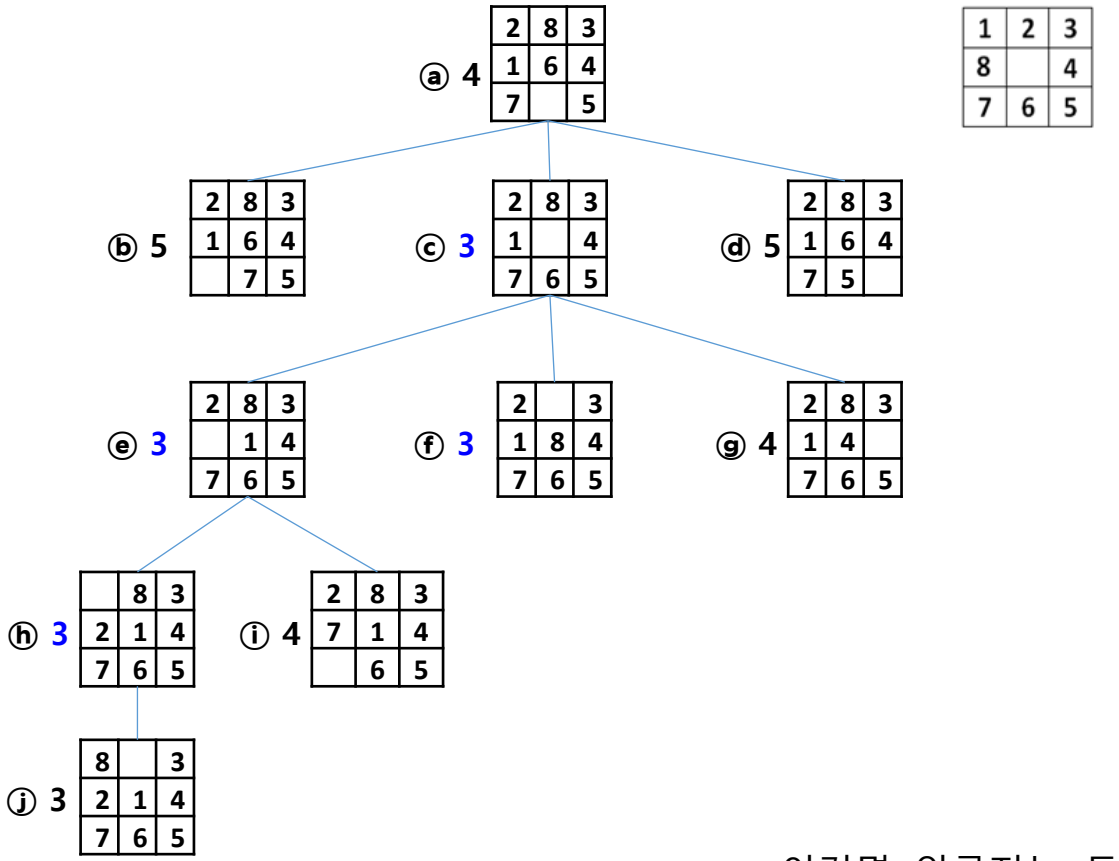
- **지역 탐색(local search), 휴리스틱 탐색(heuristic search)**
- 현재 노드에서 휴리스틱에 의한 평가값이 **가장 좋은 이웃 노드** 하나를 **확장**해 가는 탐색 방법
- **국소 최적해(local optimal solution)**에 빠질 가능성



이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

최상 우선 탐색(best-first search)

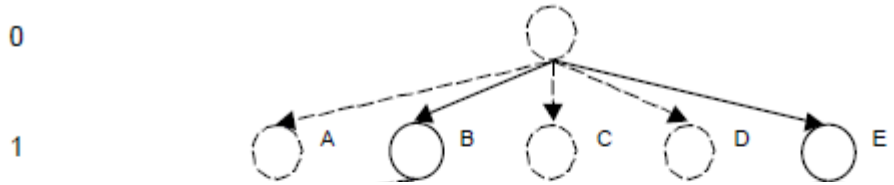
- 확장 중인 노드들 중에서 목표 노드까지 **남은 거리가 가장 짧은 노드**를 확장하여 탐색
- 남은 거리를 정확히 알 수 없으므로 **휴리스틱** 사용: 제자리가 아닌 타일의 개수



이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

빔 탐색(beam search)

- 휴리스틱에 의한 평가값이 우수한 **일정 개수의 확장 가능한 노드**만을 메모리에 **관리**하면서 **최상 우선 탐색**을 적용



이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

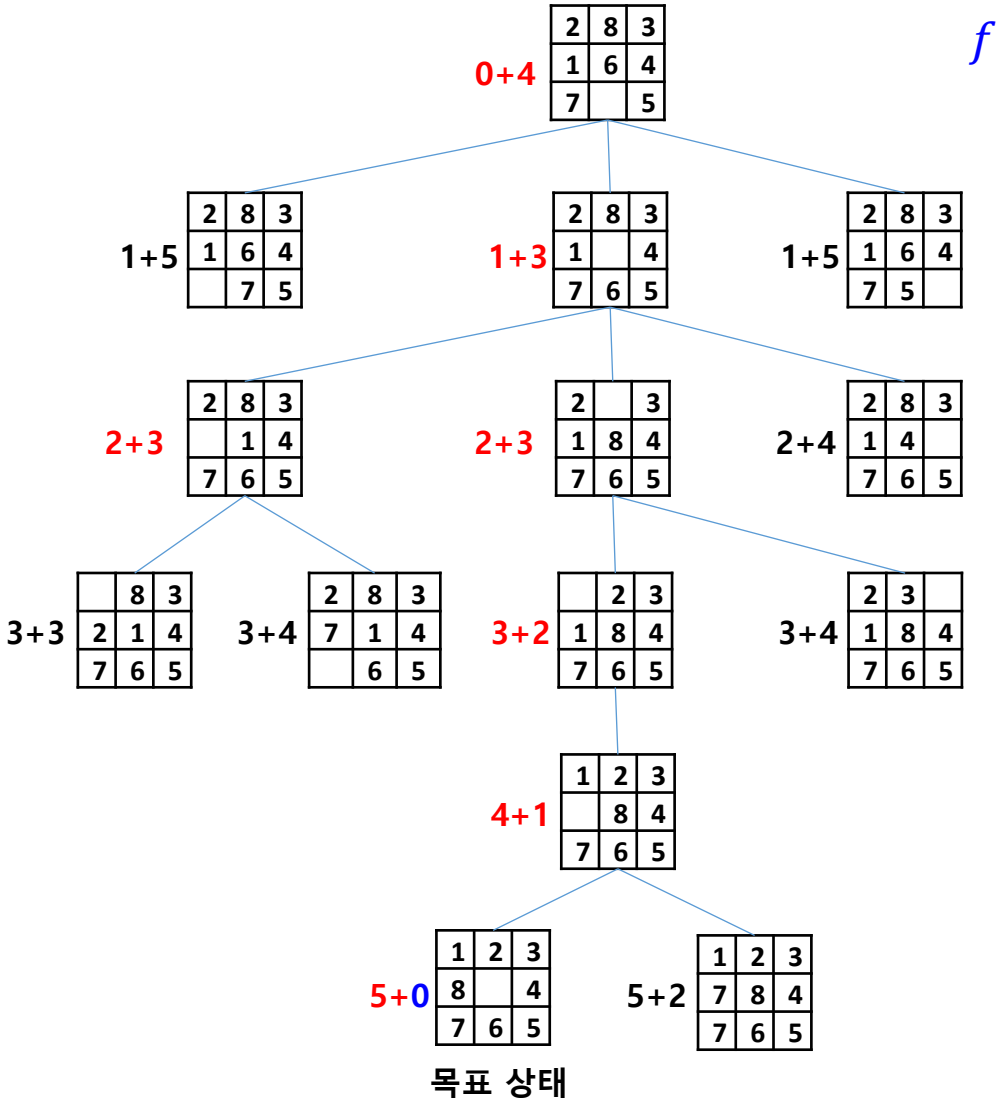
A* 알고리즘

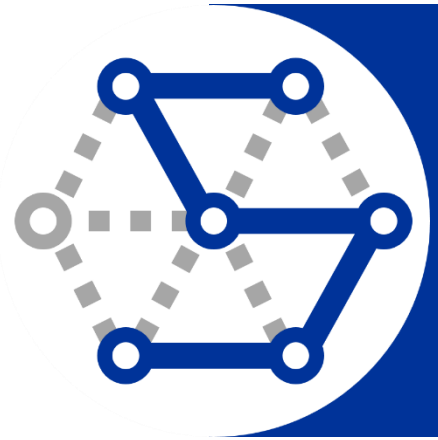
- 지금까지는 현재의 노드에서 목적노드까지 어떻게 하면 빨리 찾아갈 수 있는가에 관심
- 어떤 문제는 목적노드만을 찾는 것이 중요한 것이 아니고 시작노드에서 목적노드까지 최단경로를 찾아야 하는 경우도 있음
- 추정한 전체 비용 $\hat{f}(n)$을 최소로 하는 노드를 확장해 가는 방법
- $f(n)$: 노드 n 을 경유하는 전체 비용
  - 현재 노드 n까지 이미 **투입된 비용** $g(n)$과 목표 노드까지의 **남은 비용** $h(n)$의 합
  - $f(n) = g(n) + h(n)$
- $h(n)$: 남은 비용의 정확한 예측 불가
  - $\hat{h}(n)$: $h(n)$에 대응하는 휴리스틱 함수(heuristic function)
- $\hat{f}(n)$: 노드 n 을 경유하는 추정 전체 비용
  - $\hat{f}(n) = g(n) + \hat{h}(n)$

이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

8-퍼즐 문제의 A* 알고리즘 적용

$$f(n) = g(n) + h(n)$$





4 게임 탐색

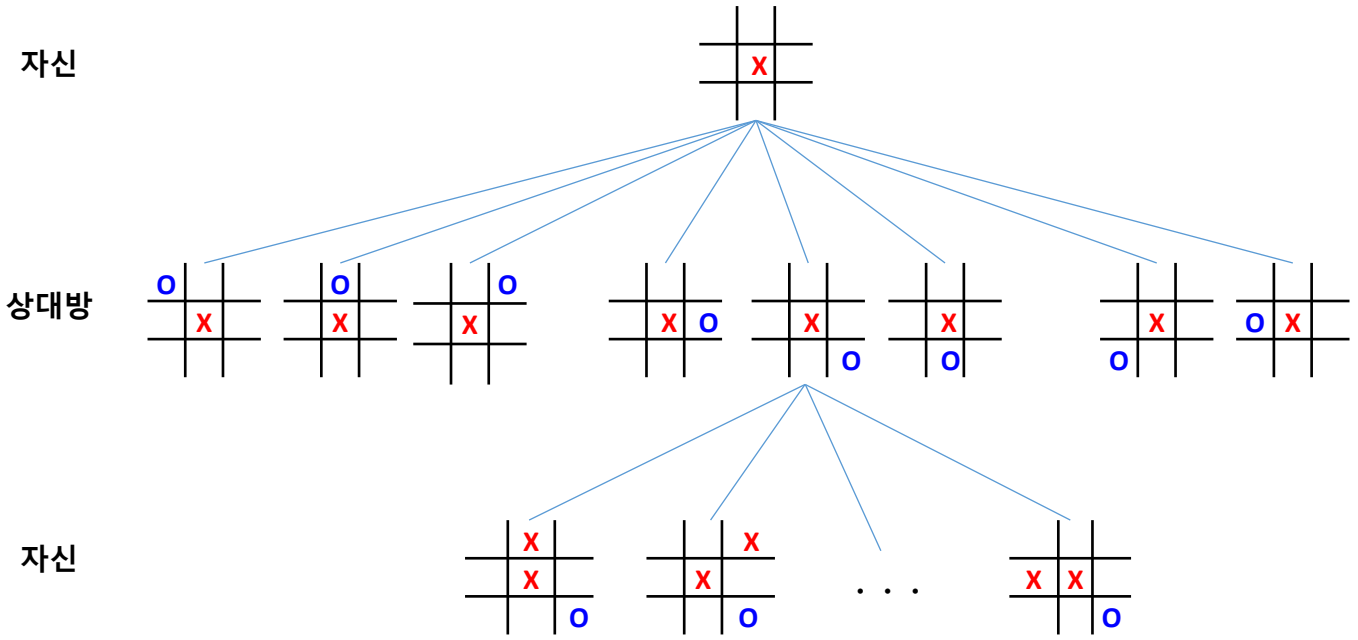
게임과 탐색

- 게임은 8-퍼즐게임처럼 한 사람이 하는 게임도 있지만, 상대가 있는 게임도 많음
- 상대가 있는 게임은 상대의 의도 까지를 고려해야 하기 때문에 일반적으로 혼자 하는 게임보다 더욱 복잡
- 대부분의 게임은 제한된 시간 내에 모든 상태를 전부 탐색 할 수 없으므로 탐색공간을 줄여야 함
- 휴리스틱을 잘 활용하면 탐색공간을 크게 줄일 수 있음

이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

게임 트리(game tree)

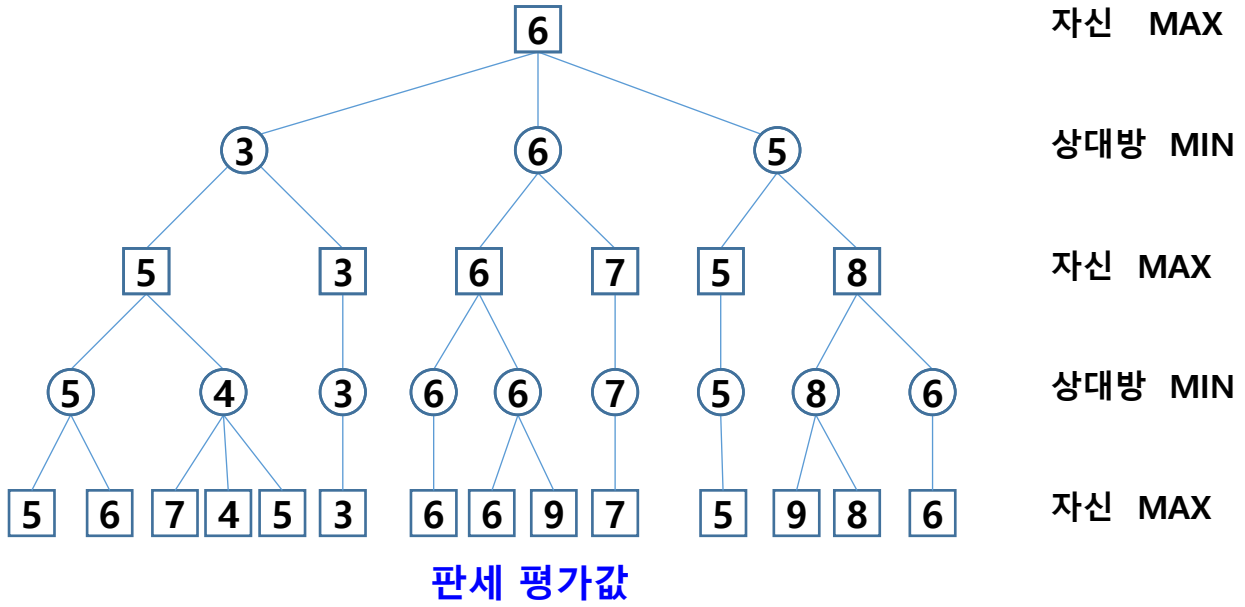
- 상대가 있는 게임에서 자신과 상대방의 가능한 게임 상태를 나타낸 트리
- 틱-택-톡(tic-tac-toc), 바둑, 장기, 체스 등
- 게임의 결과는 마지막에 결정
- 많은 수(lookahead)를 볼 수록 유리



이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

mini-max 알고리즘(mini-max algorithm)

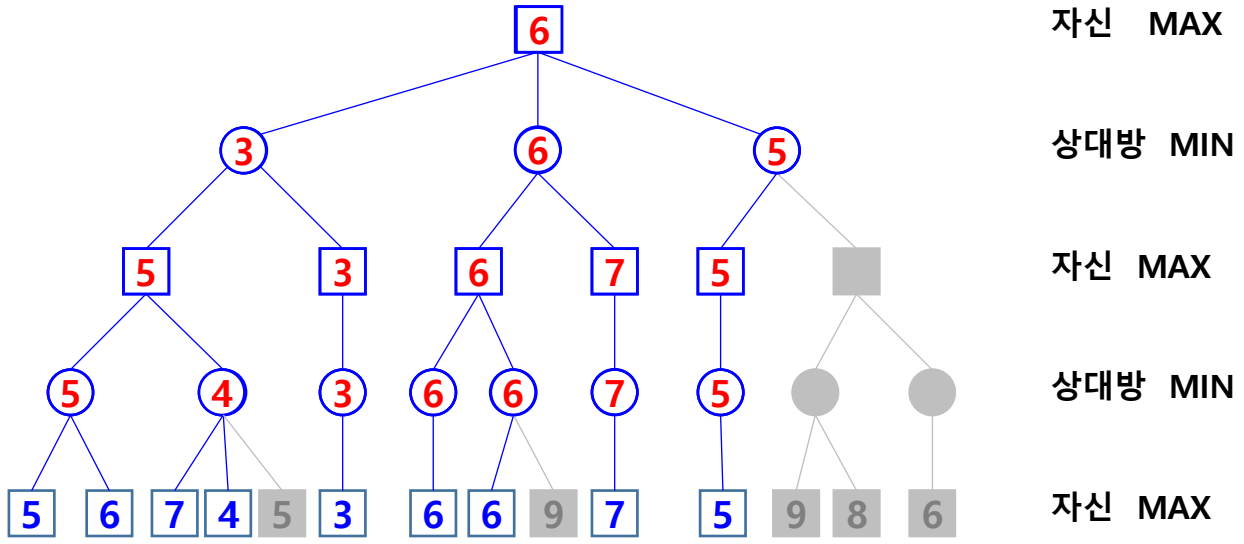
- MAX 노드: 자신에 해당하는 노드로 자기에게 유리한 최대값 선택
- MIN 노드: 상대방에 해당하는 노드로 최소값 선택
- 단말 노드부터 위로 올라가면서 **최소(minimum)-최대(maximum) 연산**을 **반복**하여 자신이 선택할 수 있는 방법 중 가장 좋은 것은 값을 결정



이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

α-β 가지치기(prunning)

- 상태공간 중에 탐색에 고려하지 않아도 최종결과에는 영향을 미치지 않는 노드들을 절단하여 탐색공간을 줄이는 탐색기법
- 깊이 우선 탐색으로 제한 깊이까지 탐색을 하면서, MAX 노드와 MIN 노드의 값 결정
- **α-자르기:** MIN 노드의 현재값이 부모노드의 현재 값보다 작거나 같으면, 나머지 자식 노드 탐색 중지
- **β-자르기:** MAX 노드의 현재값이 부모노드의 현재 값보다 같거나 크면, 나머지 자식 노드 탐색 중지

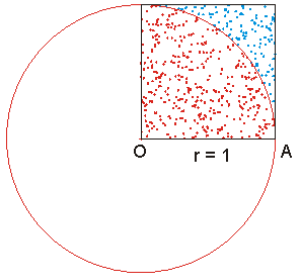


간단한 형태의 α-β 가지치기 예

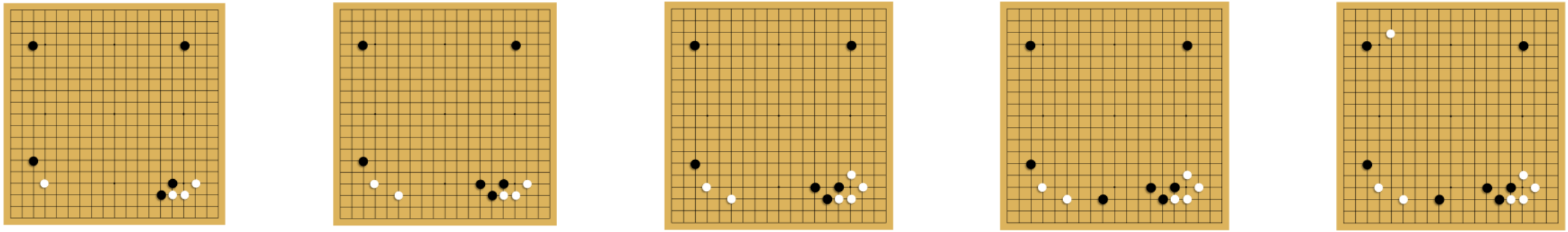
이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

몬테카를로 시뮬레이션 (Monte Carlo Simulation)

- 특정 **확률 분포**로부터 **무작위 표본**(random sample)을 생성
- 이 표본에 따라 **행동**을 하는 과정을 반복하여 결과를 확인
- 이러한 결과확인 과정을 반복하여 최종 결정을 하는 것



$$\frac{\text{원 안의 샘플 개수}}{\text{전체 샘플의 개수}} \rightarrow \frac{\pi}{4}$$

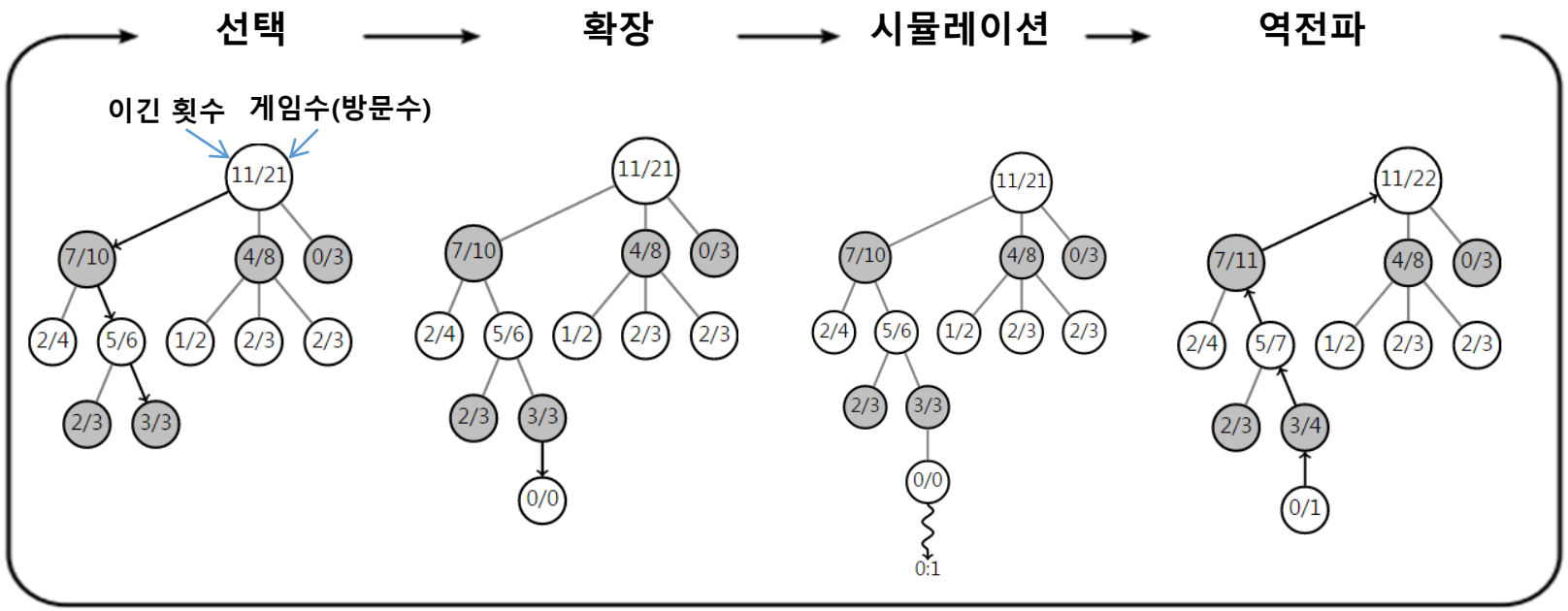


이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

몬테카를로 트리 탐색(Monte Carlo Tree Search, MCTS)

- 탐색 공간(search space)을 무작위 표본추출(random sampling)을 하면서, 탐색트리를 확장하여 가장 좋아 보이는 것을 선택하는 휴리스틱 탐색 방법
- 4개 단계를 반복하여 시간이 허용하는 동안 트리 확장 및 시뮬레이션

  선택(selection) → 확장(expansion) → 시뮬레이션(simulation) → 역전파(back propagation)



이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

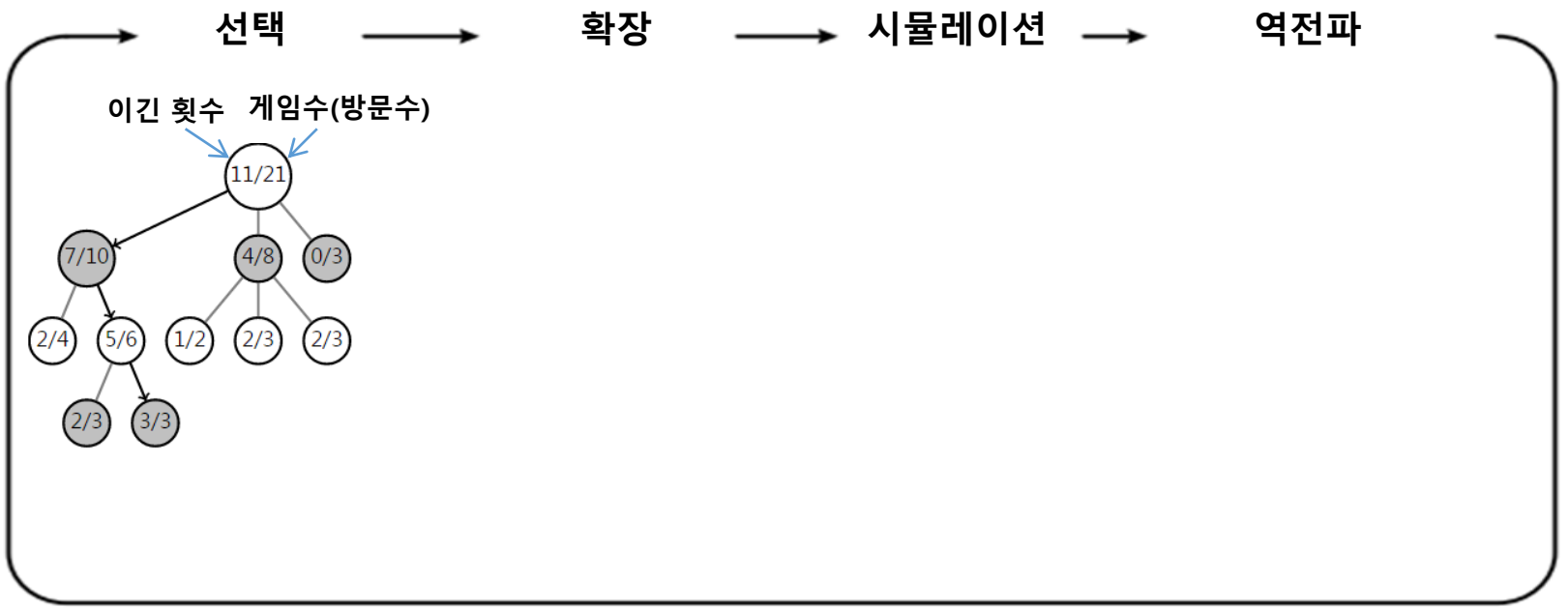
몬테카를로 트리 탐색(Monte Carlo Tree Search, MCTS)

- 선택(selection): 트리 정책(tree policy) 적용
  - 루트노드에서 시작
  - 정책에 따라 자식 노드를 선택하여 단말노드까지 내려 감
  - 승률과 노드 방문횟수 고려하여 선택
  - UCB(Upper Confidence Bound) 정책 : UCB가 큰 것 선택

v : 부모노드
v': 자식노드
$N(v')$: 방문 횟수
$Q(v')$: 점수 (이긴 횟수)

$$\frac{Q(v')}{N(v')} + c\sqrt{\frac{2\ln N(v)}{N(v')}}$$

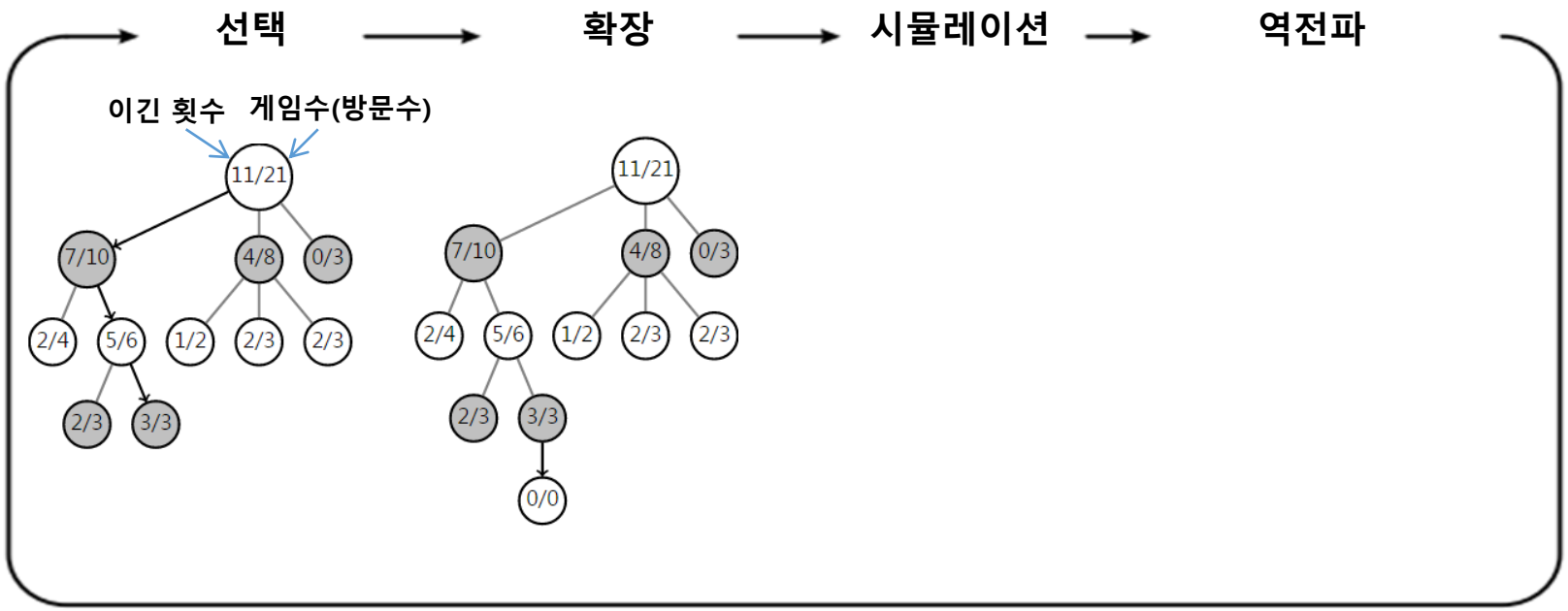
활용(exploitation) 탐험(exploration)



이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

몬테카를로 트리 탐색(Monte Carlo Tree Search, MCTS)

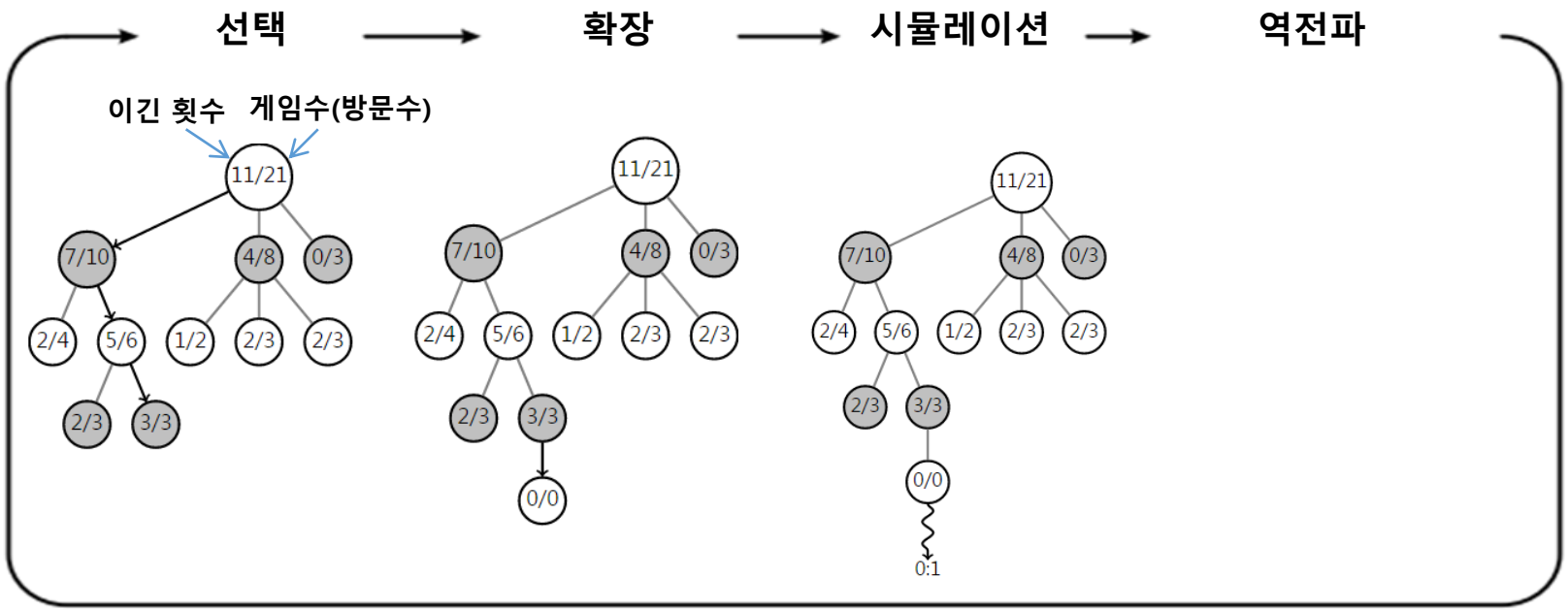
- 확장(expansion)
  - 단말노드에서 트리 정책에 따라 노드 추가
  - 예. 일정 횟수이상 시도된 수(move)가 있으면 해당 수에 대한 노드 추가



이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

몬테카를로 트리 탐색(Monte Carlo Tree Search, MCTS)

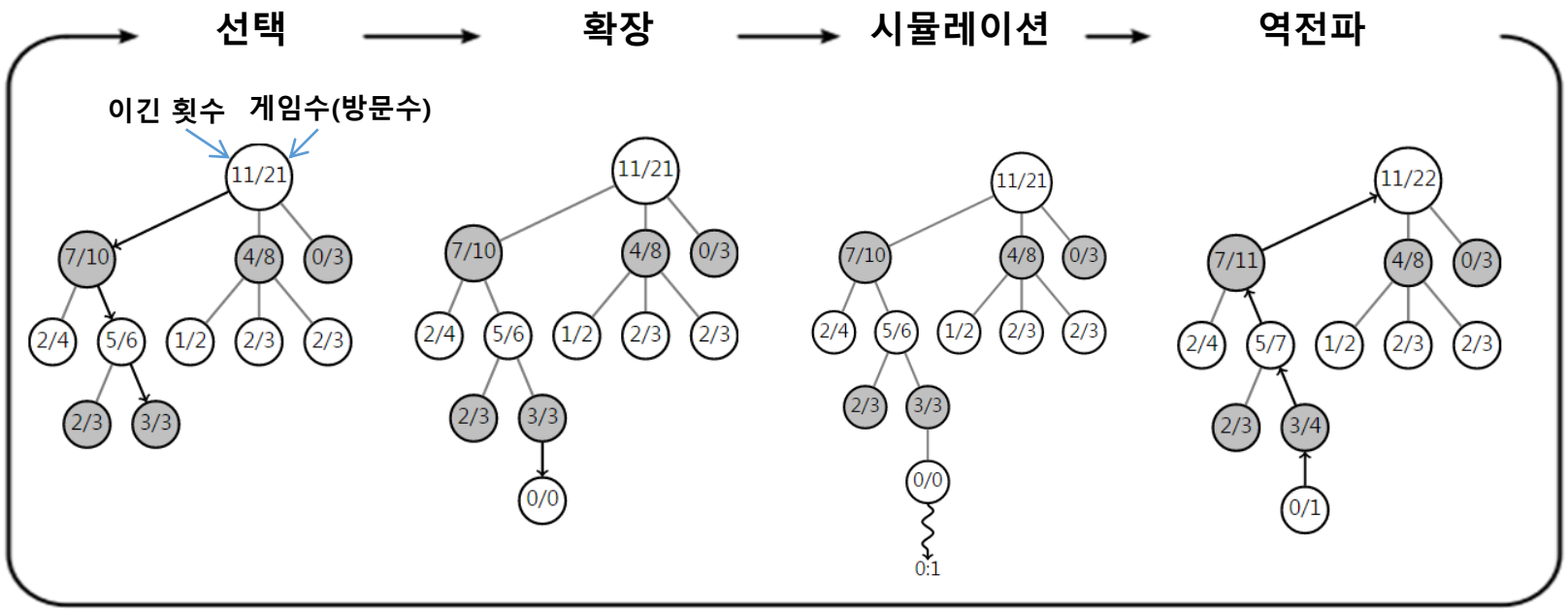
- 시뮬레이션(simulation)
  - 기본 정책(default policy)에 의한 몬테카를로 시뮬레이션 적용
  - 무작위 선택(random moves) 또는 약간 똑똑한 방법으로 게임 끝날 때까지 진행



이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

몬테카를로 트리 탐색(Monte Carlo Tree Search, MCTS)

- 역전파(backpropagation)
  - 단말 노드에서 루트 노드까지 올라가면서 승점 반영

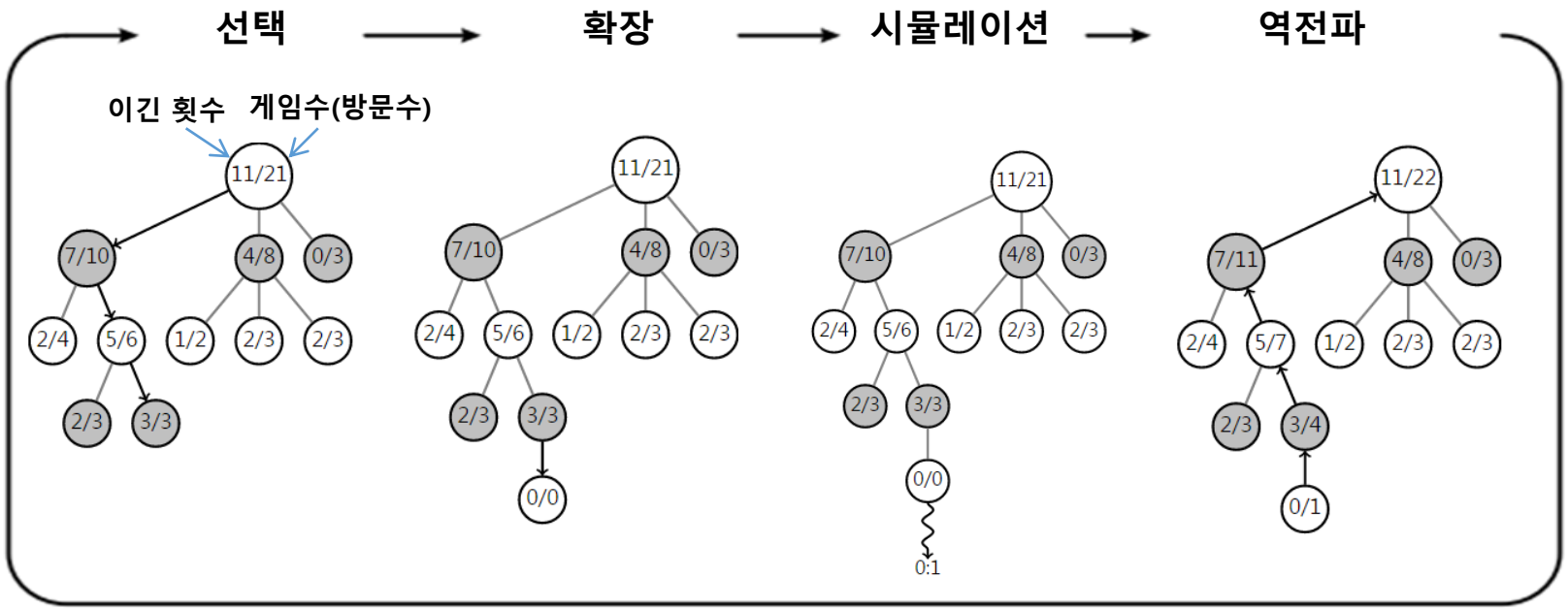


이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

몬테카를로 트리 탐색(Monte Carlo Tree Search, MCTS)

- 동작 선택 방법
  - 가장 승률이 높은, 루트의 자식 노드 선택
  - 가장 빈번하게 방문한, 루트의 자식 노드 선택
  - 승률과 빈도가 가장 큰, 루트의 자식 노드 선택, 없으면 조건을 만족하는 것이 나올 때까지 탐색 반복
  - 자식 노드의 confidence bound값의 최소값이 가장 큰, 루트의 자식 노드 선택

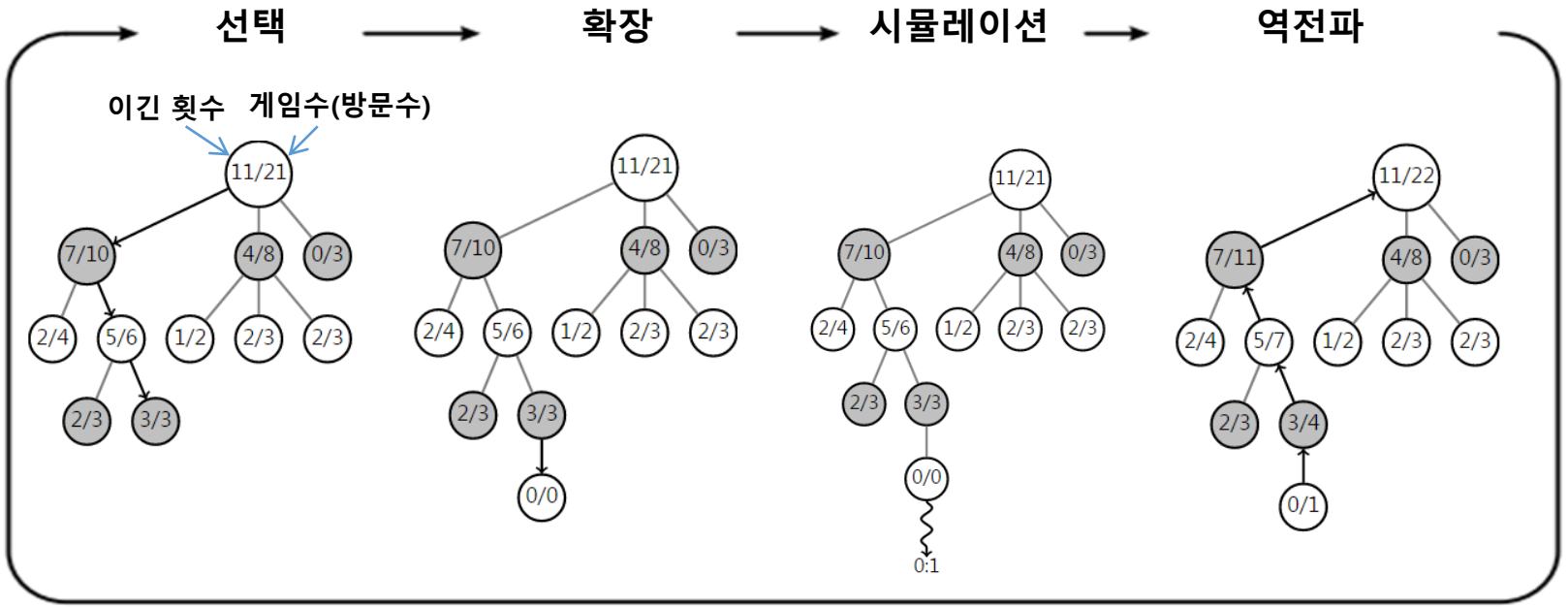
$$\frac{Q(v')}{N(v')} + c\sqrt{\frac{2\ln N(v)}{N(v')}}$$



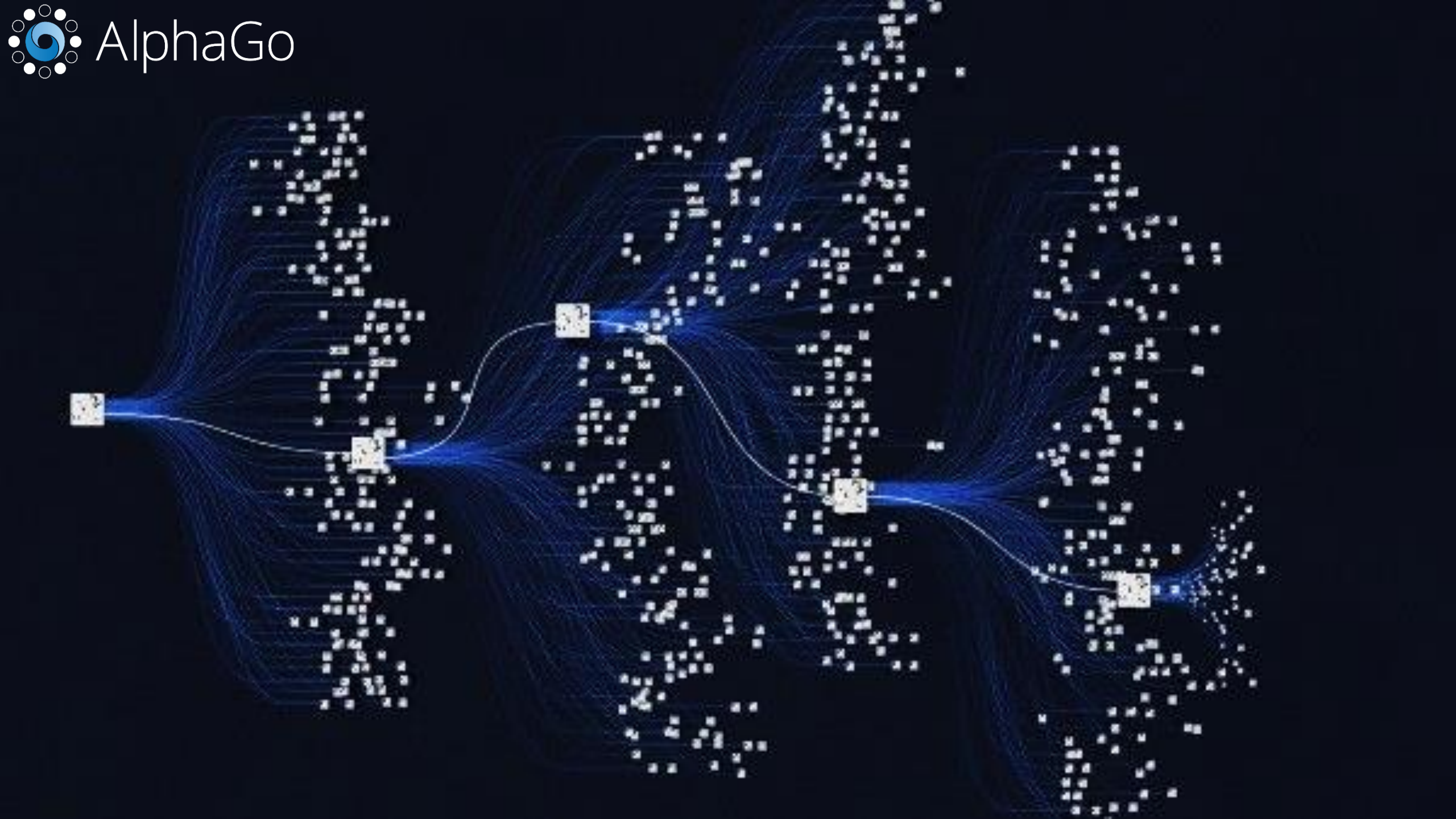
이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

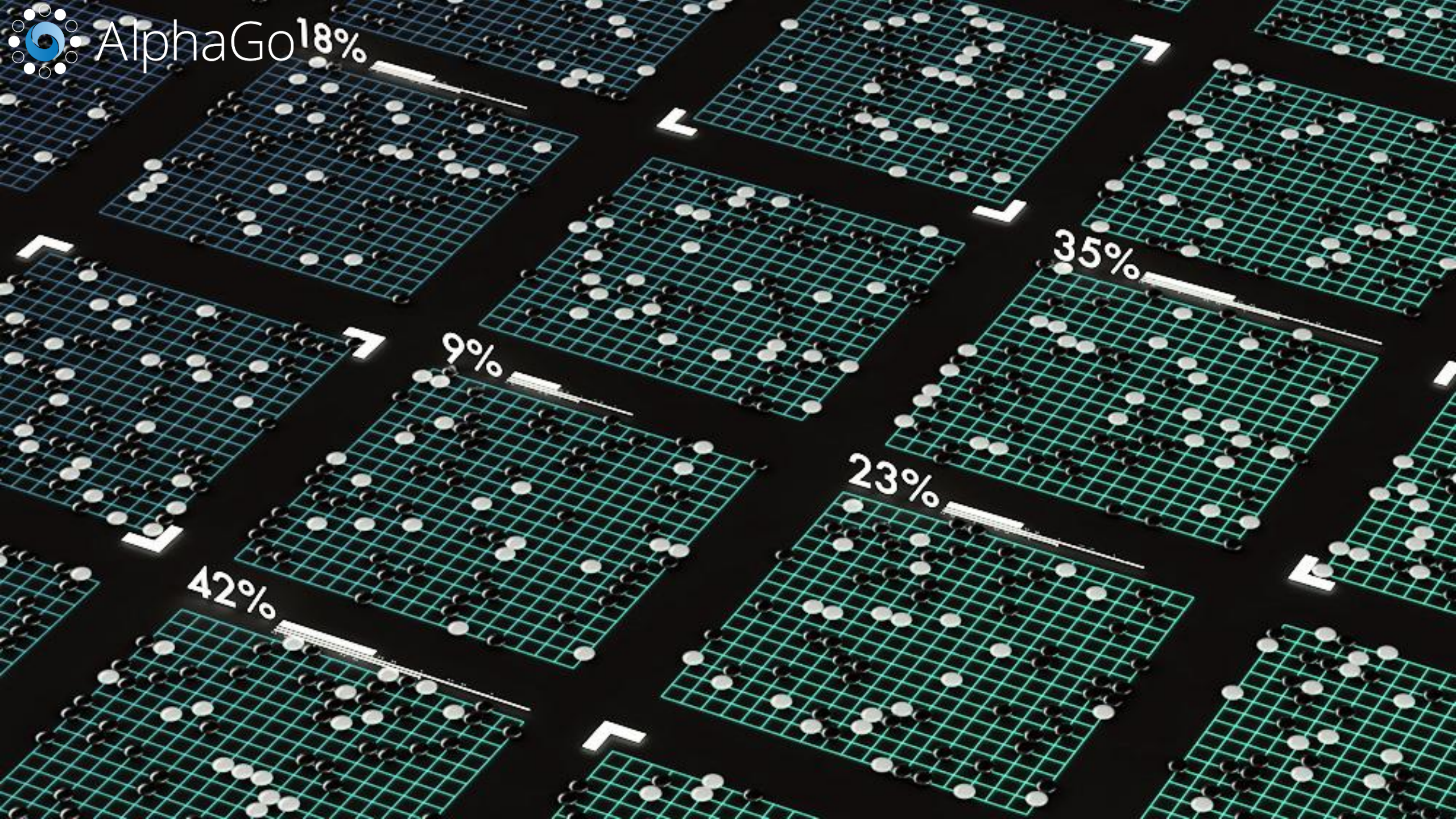
몬테카를로 트리 탐색(Monte Carlo Tree Search, MCTS)

- 형세판단을 위해 **휴리스틱** 사용 **대신**, 가능한 많은 수의 **몬테카를로 시뮬레이션** 수행
- 일정 조건을 만족하는 부분은 트리로 구성하고, 나머지 부분은 **몬테카를로 시뮬레이션**
- 가능성이 높은 수(move)들에 대해서 노드를 생성하여 트리의 탐색 폭을 줄이고, 트리 깊이를 늘리지 않기 위해 몬테카를로 시뮬레이션을 적용
- **탐색 공간 축소**



이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판


AlphaGo


AlphaGo
18%
9%
42%
35%
23%

AlphaGo Overview

Input Board Position
as 19 x 19 Image
48 Feature Planes

Policy Network

Expert Games
130 000 Games
30 M Positions

Supervised Learning
SL Policy
Position --> Next Move
Accuracy: 56%

Fast Policy Network

Expert Games
140 000 Patterns
130 000 Games
30 M Positions

Supervised Learning
Fast Policy
Pattern --> Next Move
Accuracy: 24%

Reinforcement Learning Policy Network

Self-Play Games
1.3 M Games by
various versions
of RL Policy

Reinforcement Learning
RL Policy
Position --> Next Move
Wins 80% vs. SL Policy

Value Network

Self-Play Games
30 M Positions
by fixed version
of RL Policy

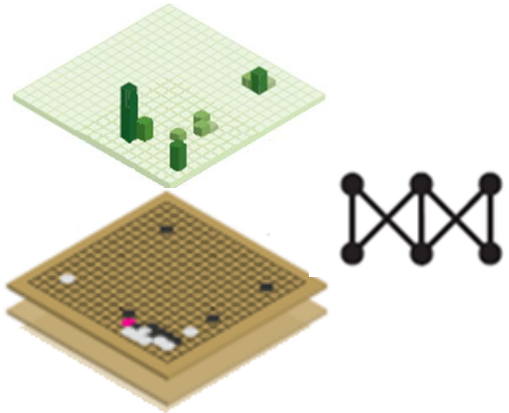
Reinforcement Learning
Position --> Win Probability
15 000 times faster than
MCTS Rollouts evaluations

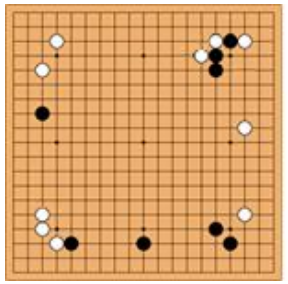
AlphaGo

Monte Carlo Tree Search

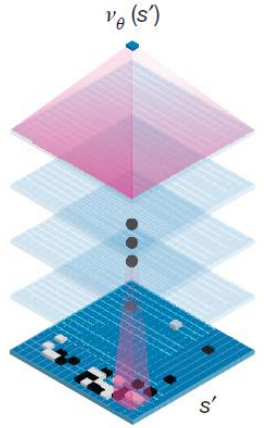
알파고의 몬테카를로 트리 검색

- **바둑판 형세 판단**을 위한 한가지 방법으로 **몬테카를로 트리 검색** 사용
- 무작위로 바둑을 두는 것이 아니라, 프로 바둑기사들을 기보를 학습한 **확장 정책망** (rollout policy network)이라는 간단한 계산모델을 사용





0.6



정책망 : 가능한 착수(着手)들에 대한 **선호 확률분포**

가치망 : 바둑판의 **형세 값**을 계산하는 계산모델

- 확률에 따라 착수를 하여 **몬테카를로 시뮬레이션**을 **반복**하여 **형세판단값 계산**
- 별도로 학습된 **딥러닝 신경망**인 **가치망(value network)**을 사용하여 형세판단값을 계산

이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

알파고의 몬테카를로 트리 검색

- 많은 수의 **몬테카를로 시뮬레이션**과 **딥러닝 모델**의 신속한 계산을 위해 다수의 **CPU와 GPU**를 이용한 **분산처리**

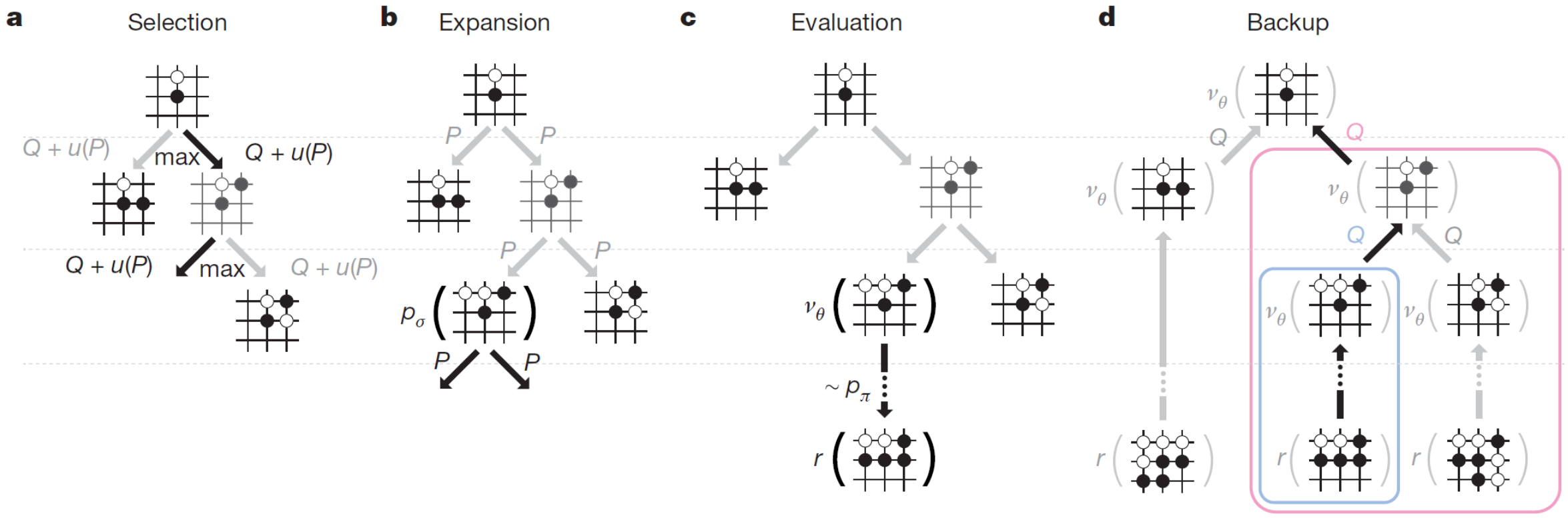


Image : Nature

이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

알파고 발전

AlphaGo Pan

- 2015년 10월 유럽 바둑 챔피언 판후이(Pan Hui)와 대국
- CPU 1,202개와 GPU 176개 사용

AlphaGo Lee

- 2016년 3월 이세돌 기사와 대국
- CPU 1,920개와 TPU 48개 사용 (분산처리 시스템)

AlphaGo Master

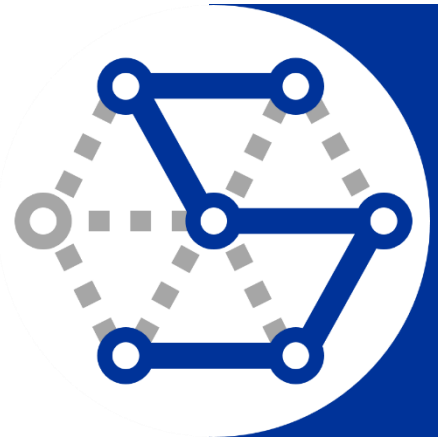
- 2017년 5월 중국 기사 커제와 대국
- 구글의 2세대 TPU 4개 사용 (컴퓨터 한대)

AlphaGo Zero

- 바둑 기보나 바둑 지식을 전혀 사용하지 않고 바둑의 규칙과 자체 경기만을 통해서 학습
- TPU 4개 사용 (컴퓨터 한대)
- AlphaGo Lee와의 대전에서 100:0, AlphaGo Master와의 대전에서 89:11

AlphaZero

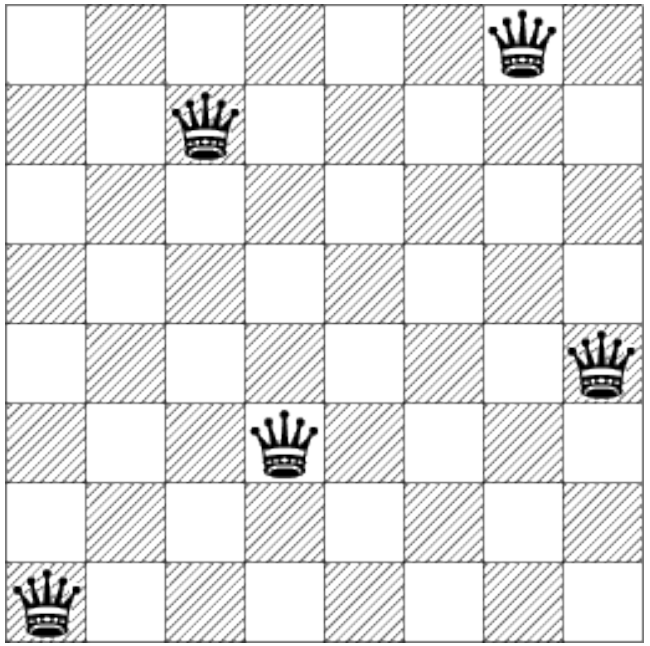
- 2017년 12월, 바둑 뿐만 아니라 체스, 장기 등의 보드 게임은 자체 경기만으로 매우 빠른 시간에 학습
- 당시 최고 체스 프로그램 Stockfish보다 월등한 성능을 보여주고, 일본 장기 최고 프로그램인 Elmo를 절대적으로 압도



5 제약조건 만족 문제

제약조건 만족 문제(constraint satisfaction problem)

- 주어진 제약조건을 만족하는 조합 해(combinatorial solution)를 찾는 문제
- 탐색 기반의 해결방법
  - 백트랙킹 탐색
  - 제약조건 전파



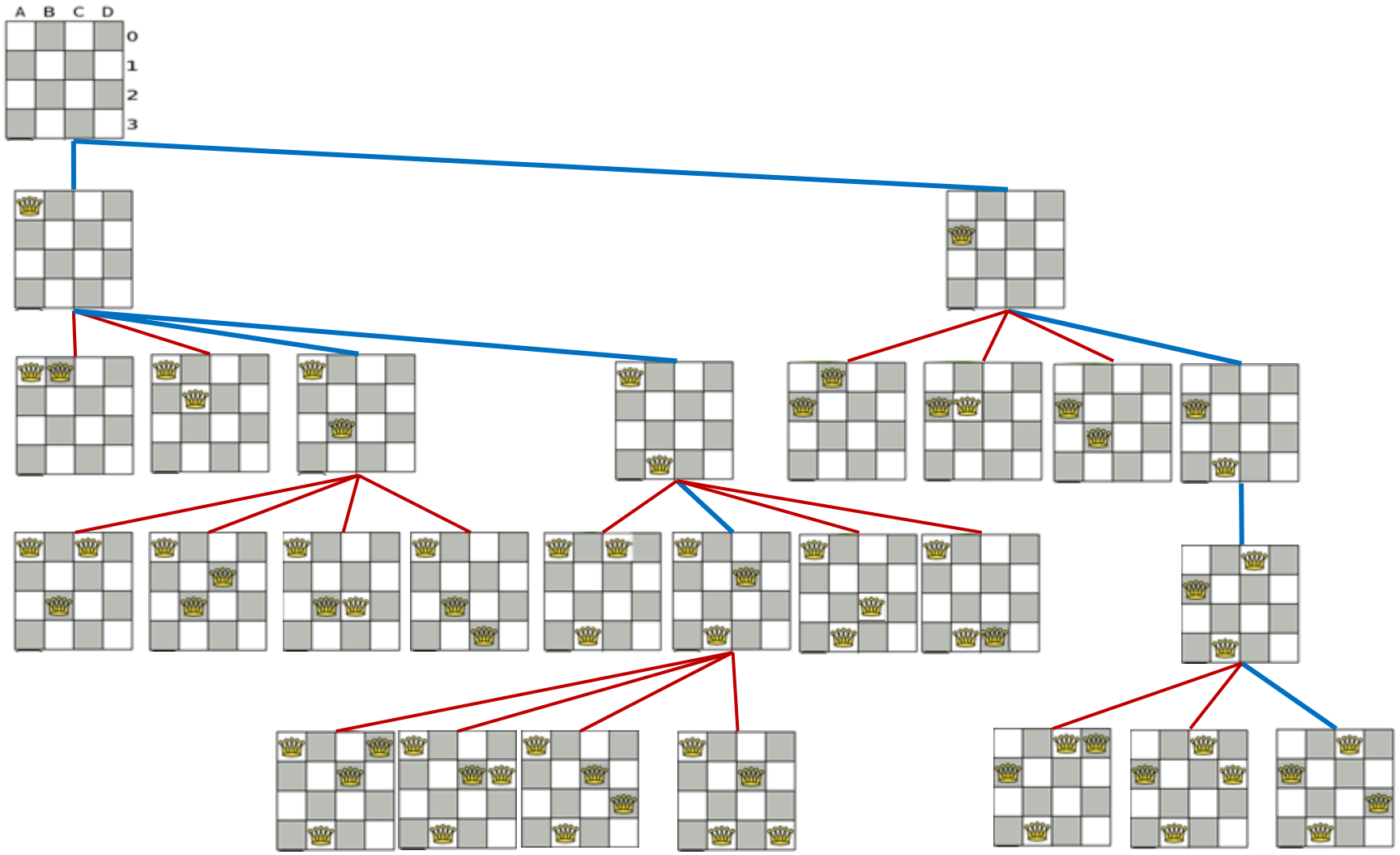
이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

백트랙킹 탐색(backtracking search)

- 깊이 우선 탐색을 하는 것처럼 변수에 허용되는 값을 하나씩 대입
- 모든 가능한 값을 대입해서 만족하는 것이 없으면 이전 단계로 돌아가서 이전 단계의 변수에 다른 값을 대입

이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

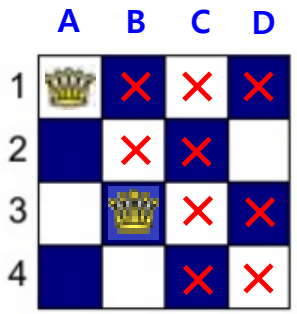
백트랙킹 탐색을 이용한 4-퀸(queen) 문제

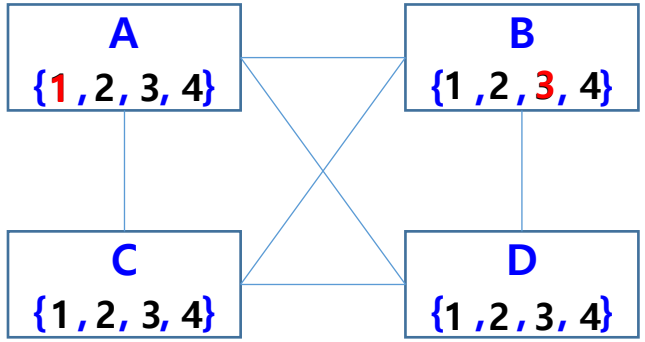


이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

제약조건 전파(constraint propagation)

- **인접 변수 간**의 **제약 조건**에 따라 각 변수에 **허용될 수 없는 값**들을 **제거**하는 방식

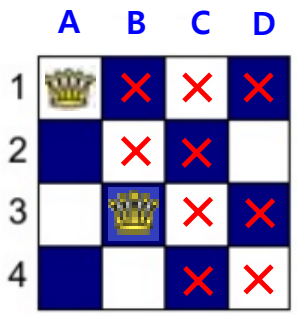


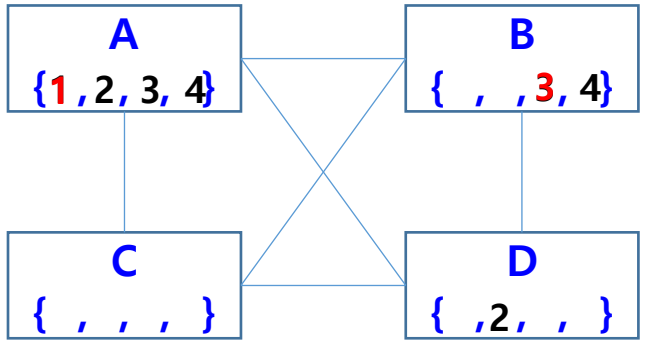


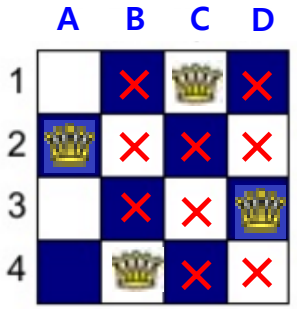
이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

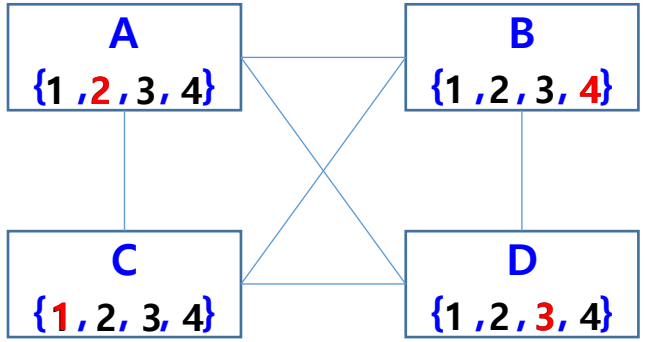
제약조건 전파(constraint propagation)

- **인접 변수 간**의 **제약 조건**에 따라 각 변수에 **허용될 수 없는 값**들을 **제거**하는 방식









이건명, 인공지능: 튜링 테스트에서 딥러닝까지, 생능출판

요약

상태공간과 탐색	무정보 탐색	정보이용 탐색	게임에서의 탐색	제약조건 만족 문제
• 상태 공간 • 상태 공간 그래프	• 깊이 우선 탐색 • 너비 우선 탐색 • 반복적 깊이 심화 탐색 • 양방향 탐색	• 휴리스틱 • 언덕 오르기 방법 • 최상우선 탐색 • 빔탐색 • A* 알고리즘	• 게임트리 • mini-max 알고리즘 • α-β 가지치기 • 몬테카를로 트리 탐색	• 백트랙킹 탐색 • 제약조건 전파 방법


suanlab
이수안 컴퓨터 연구소
suan computer laboratory