Artificial Intelligence
Chapter 8
Uninformed Search

Biointelligence Lab
School of Computer Sci. & Eng.
Seoul National University

Outline

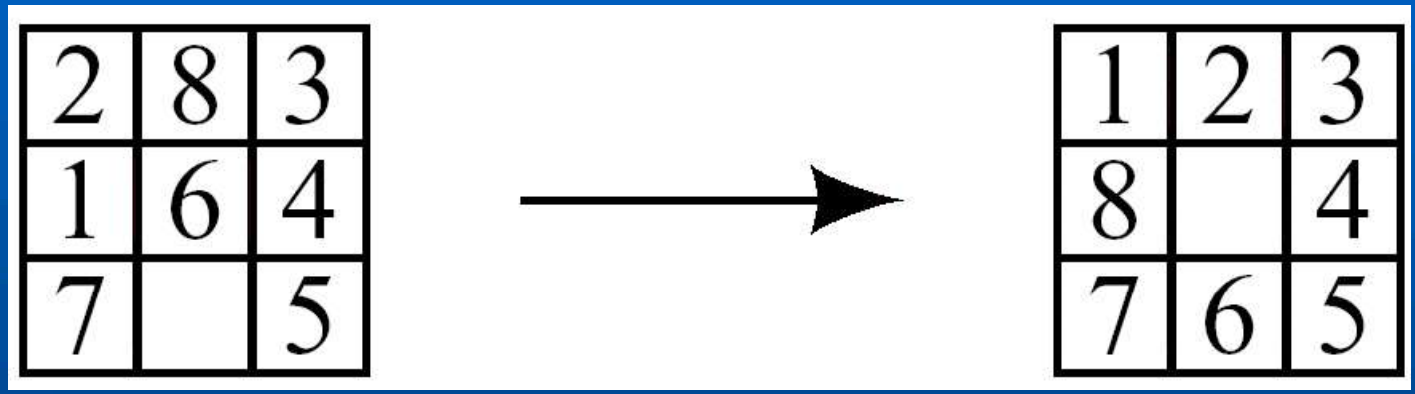
- Search Space Graphs
- Depth-First Search
- Breadth-First Search
- Iterative Deepening

1. Formulating the State Space

- For huge search space we need,
  - Careful formulation
  - Implicit representation of large search graphs
  - Efficient search method

1. Formulating the State Space (Cont'd)

- e.g.) 8-puzzle problem



  - state description
    - 3-by-3 array: each cell contains one of 1-8 or blank symbol
  - two state transition descriptions
    - 8×4 moves: one of 1-8 numbers moves up, down, right, or left
    - 4 moves: one black symbol moves up, down, right, or left

1. Formulating the State Space (Cont'd)

- The number of nodes in the state-space graph:
  - 9! (= 362,880)
- State space for 8-puzzle is
  - Divided into two separate graphs : not reachable from each other

2. Components of Implicit State-Space Graphs

- 3 basic components to an implicit representation of a state-space graph
  1. Description of start node
  2. *Actions*: Functions of state transformation
  3. *Goal condition*: *true-false* valued function
- 2 classes of search process
  1. *Uninformed* search: no problem specific information
  2. *Heuristic* search: existence of problem-specific information

3. Breadth-First Search

- Procedure
  1. Apply all possible operators (*successor function)* to the start node.
  2. Apply all possible operators to all the direct successors of the start node.
  3. Apply all possible operators to their successors till goad node found.
     - ♠ *Expanding* : applying successor function to a node

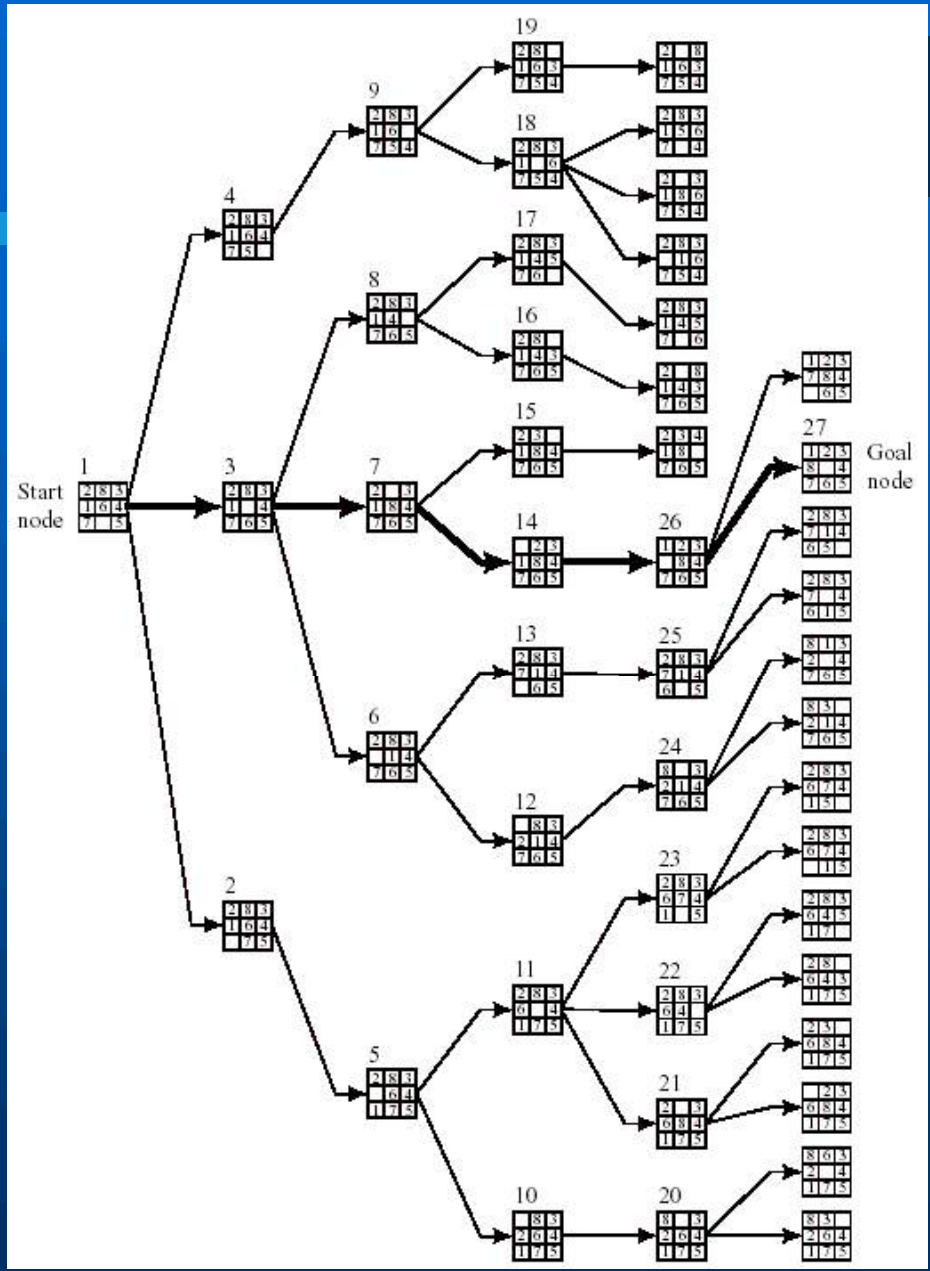


Figure 8.2 Breadth-First Search of the Eight-Puzzle

3. Breadth-First Search (Cont'd)

- Advantage
  - Finds the path of minimal length to the goal.
- Disadvantage
  - Requires the generation and storage of a tree whose size is exponential the the depth of the shallowest goal node
- *Uniform-cost* search [Dijkstra 1959]
  - Expansion by *equal cost* rather than equal depth

4. Depth-First or Backtracking Search

- Procedure
  - Generates the successor of a node just one at a time.
  - Trace is left at each node to indicate that additional operators can be applied there if needed.
  - At each node a decision must be made about which operator to apply first, which next, and so on.
  - Repeats this process until the *depth bound.*
  - *chronological Backtrack* when search depth is depth bound.

4. Depth-First or Backtracking Search (Cont'd)

- 8-puzzle example
  - Depth bound: 5
  - Operator order: left → up → right → down

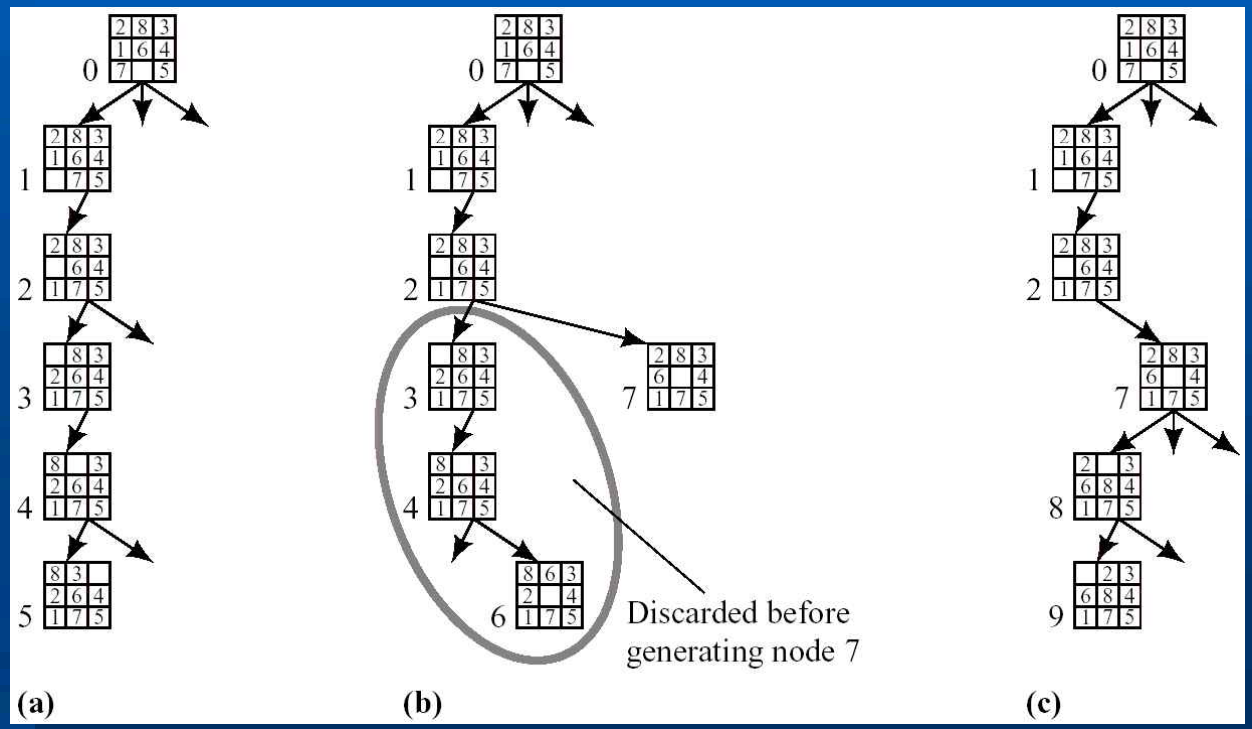
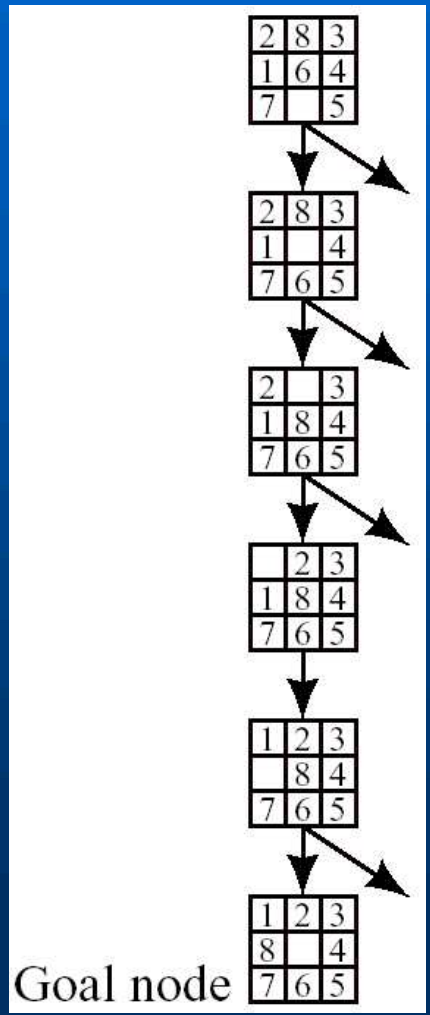


Figure 8.3 Generation of the First Few Nodes in a Depth-First Search

4. Depth-First or Backtracking Search (Cont'd)

- The graph when the goal is reached in depth-first search



4. Depth-First or Backtracking Search (Cont'd)

- Advantage
  - Low memory size: linear in the depth bound
    - saves only that part of the search tree consisting of the path currently being explored plus traces
- Disadvantage
  - No guarantee for the minimal state length to goal state
  - The possibility of having to explore a large part of the search space

5. Iterative Deepening

- Advantage
  - Linear memory requirements of depth-first search
  - Guarantee for goal node of minimal depth
- Procedure
  - Successive depth-first searches are conducted – each with depth bounds increasing by 1

5. Iterative Deepening (Cont'd)

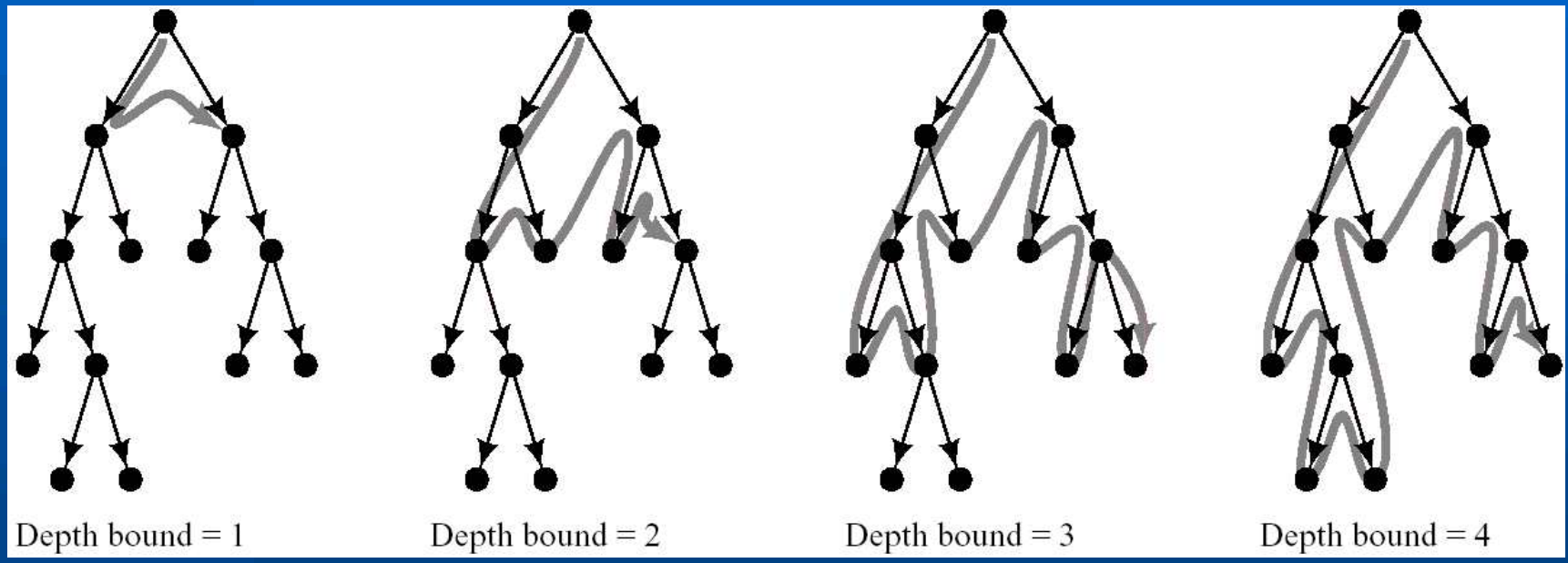


Figure 8.5 Stages in Iterative-Deepening Search

5. Iterative Deepening (Cont'd)

- The number of nodes
  - In case of breadth-first search

$$N_{\text{bf}} = 1 + b + b^2 + \cdots + b^d = \frac{b^{d+1} - 1}{b - 1} \quad (b : \text{branching factor}, d : \text{depth})$$

  - In case of iterative deepening search

$$N_{\text{df}_j} = \frac{b^{j+1} - 1}{b - 1} : \text{ number of nodes expanded down to level } j$$

$$N_{\text{id}} = \sum_{j=0}^{d} \frac{b^{j+1} - 1}{b - 1}$$

$$= \frac{1}{b-1}\left[b\left(\sum_{j=0}^{d} b^j \right) - \sum_{j=0}^{d} 1 \right] = \frac{1}{b-1}\left[b\left(\frac{b^{d+1} - 1}{b - 1} \right) - (d+1) \right]$$

$$= \frac{b^{d+2} - 2b - bd + d + 1}{(b-1)^2}$$

5. Iterative Deepening (Cont'd)

- For large d the ratio N_{id}/N_{df} is $b/(b-1)$
- For a branching factor of 10 and deep goals, 11% more nodes expansion in iterative-deepening search than breadth-first search
- Related technique *iterative broadening* is useful when there are many goal nodes

6. Additional Readings and Discussion

- Various improvements in chronological backtracking
  - *Dependency-directed backtracking* [Stallman & Sussman 1977]
  - *Backjumping* [Gaschnig 1979]
  - *Dynamic backtracking* [Ginsberg 1993]