Blind (Uninformed) **Search A**

Russell and Norvig:
Chap. 3, Sect. 3.3

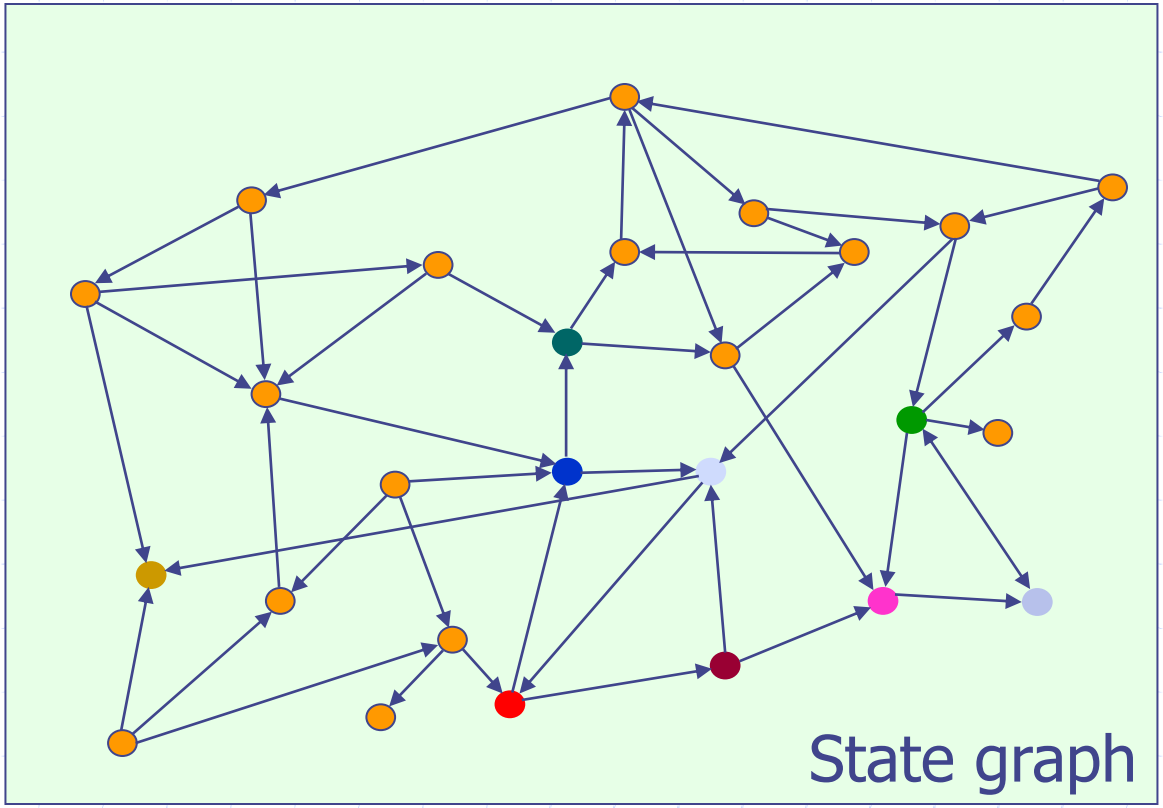
Slides from Jean-Claude Latombe at Stanford University
(used with permission)

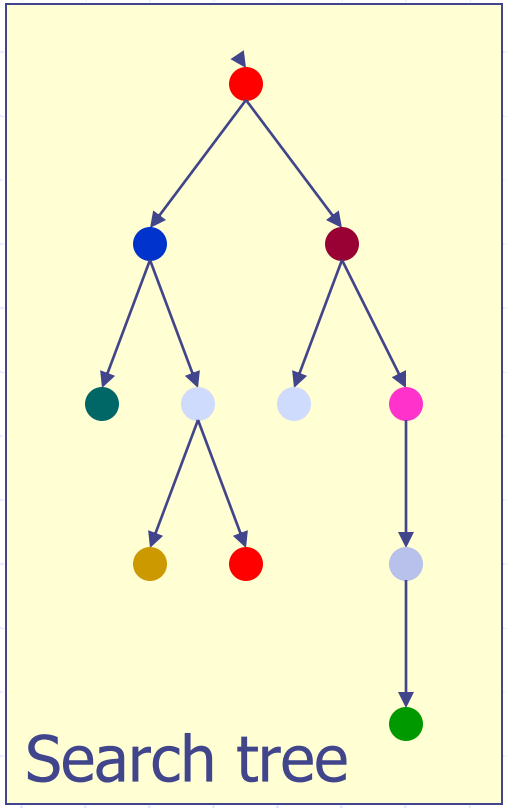
Simple Agent Algorithm

Problem-Solving-Agent

1. formulate: (abstraction!)
    1. S_0 = initial-state ◂ sense/read state
    2. GOAL? = goal test ◂ select/read goal test
    3. actions ◂ select/read action models
    4. transition model ◂ select/read model
    5. problem ◂ (S_0, GOAL?, actions, transition model)
2. solution ◂ search(problem)
3. perform(solution)

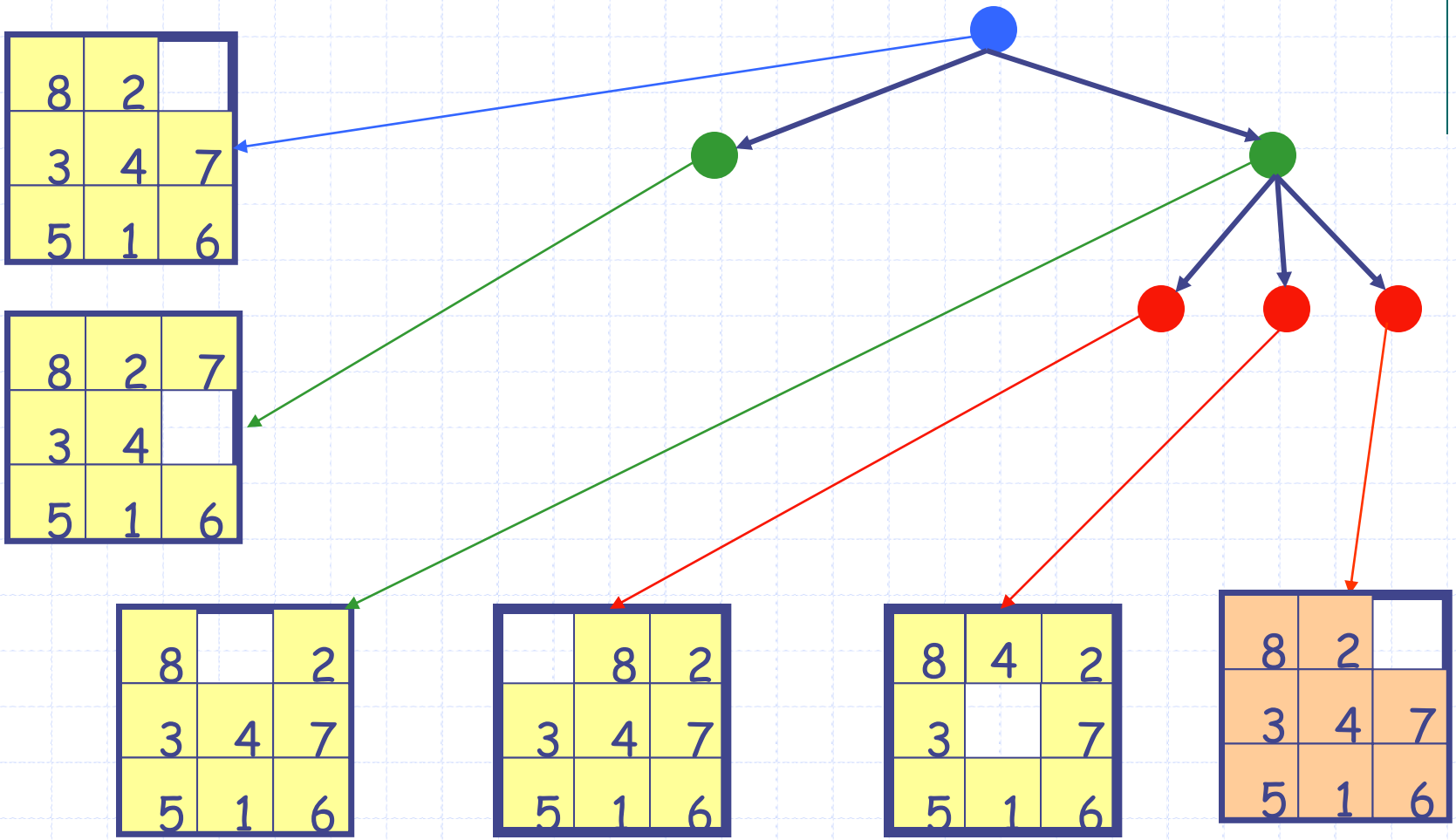
Search Tree



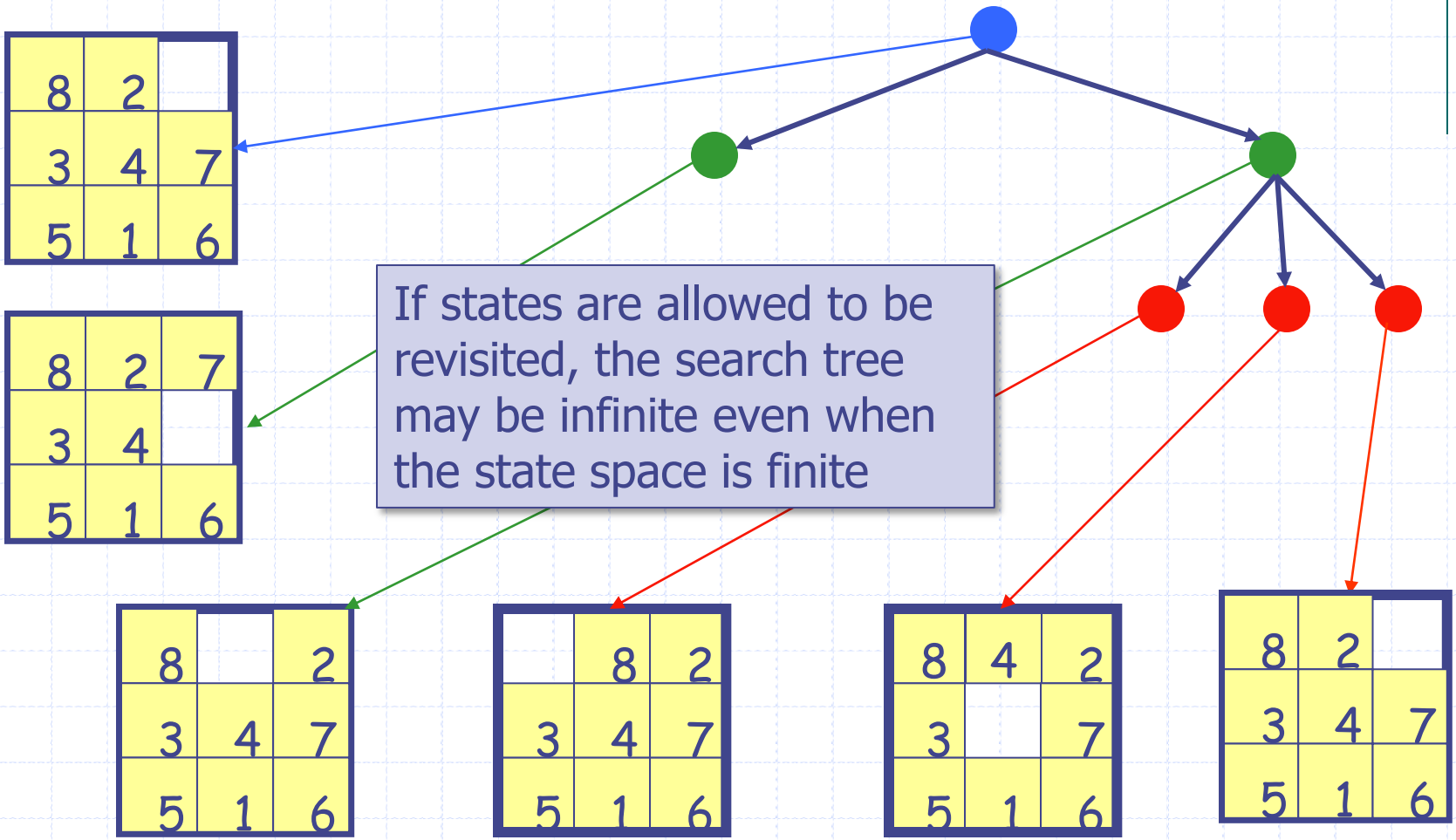


Note that some states may be visited multiple times

Search Nodes and States

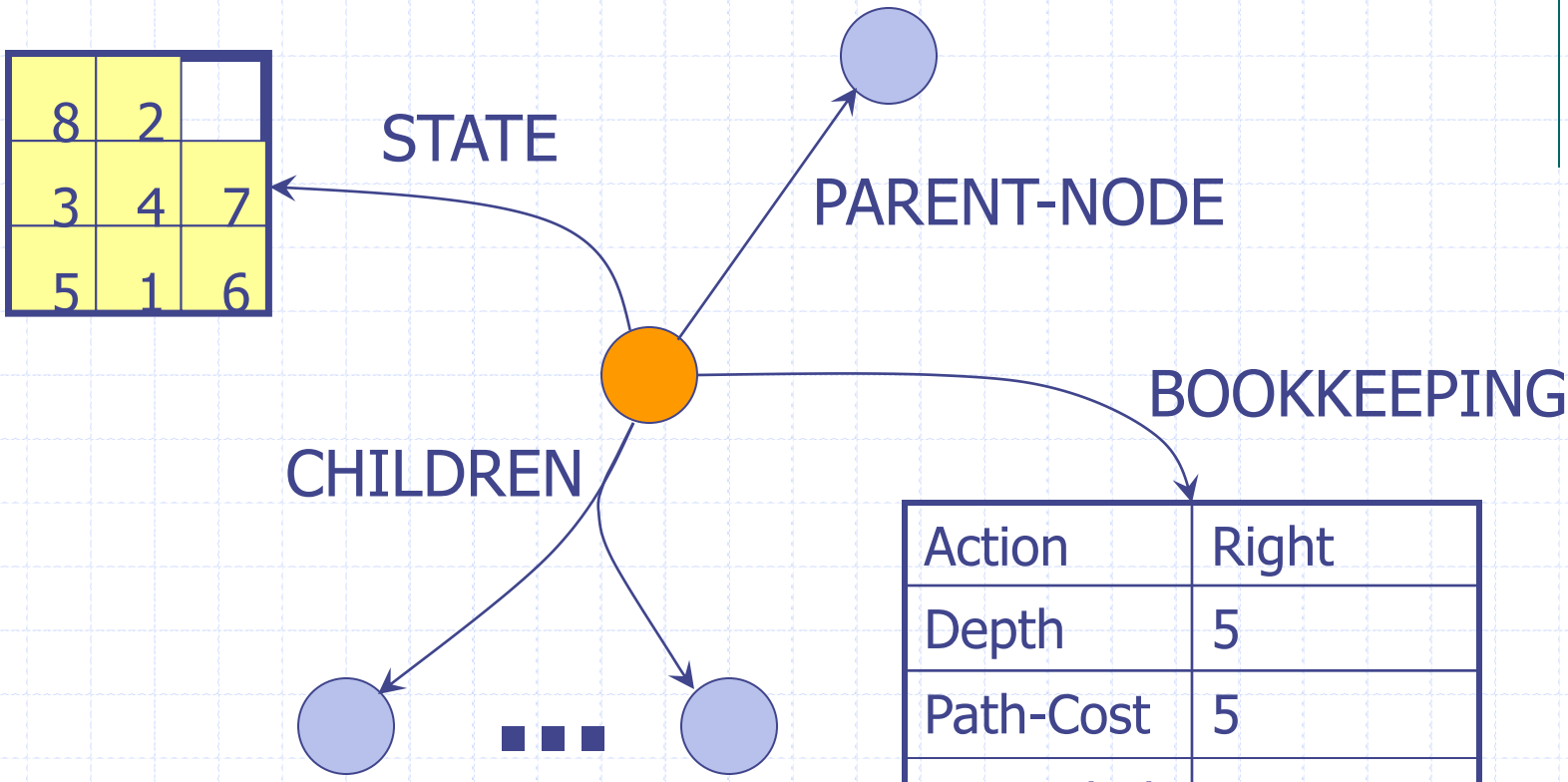


Search Nodes and States



If states are allowed to be revisited, the search tree may be infinite even when the state space is finite

Data Structure of a Node

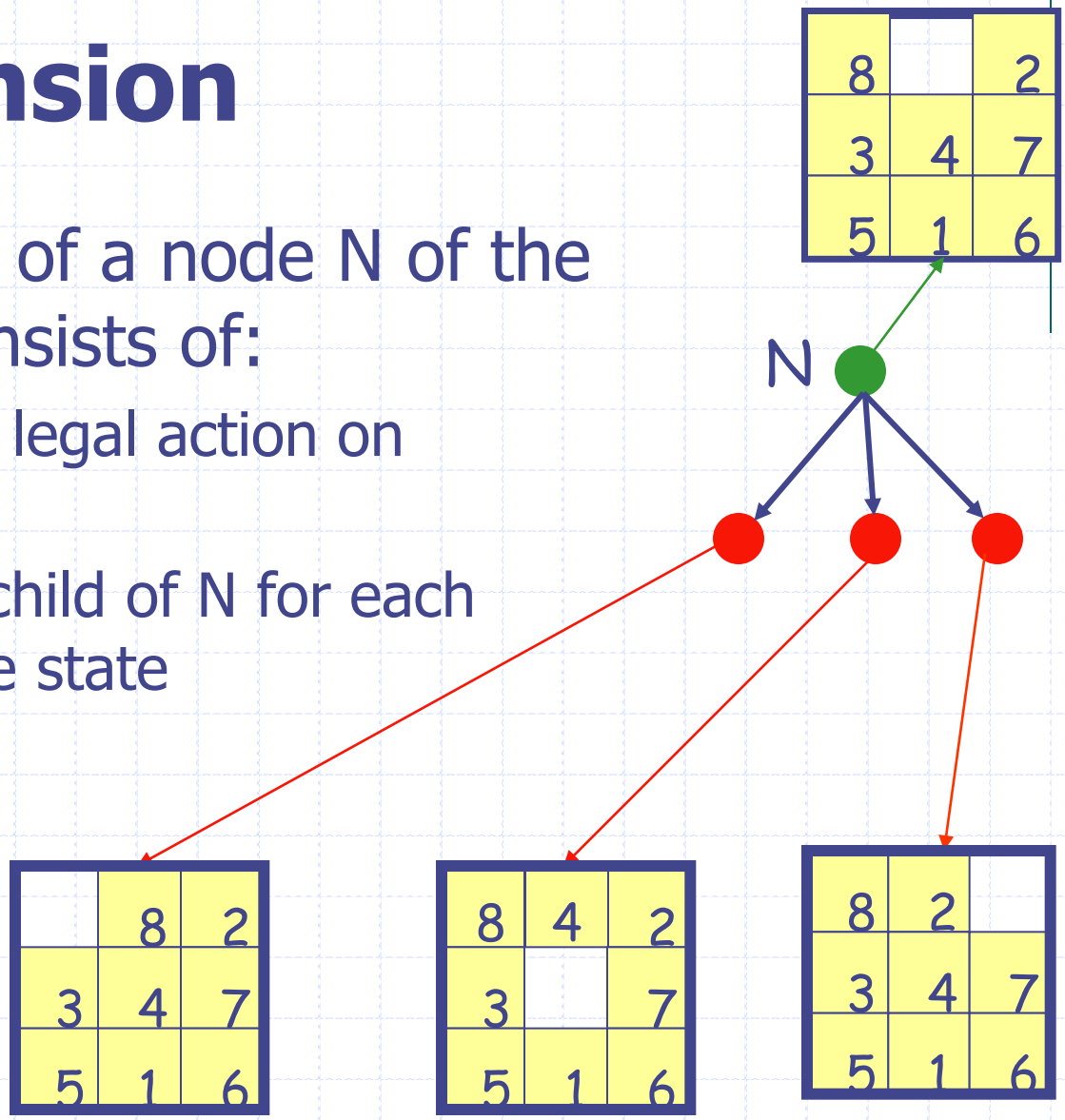


Depth of a node N = length of path from root to N
(depth of the root = 0)

Node expansion

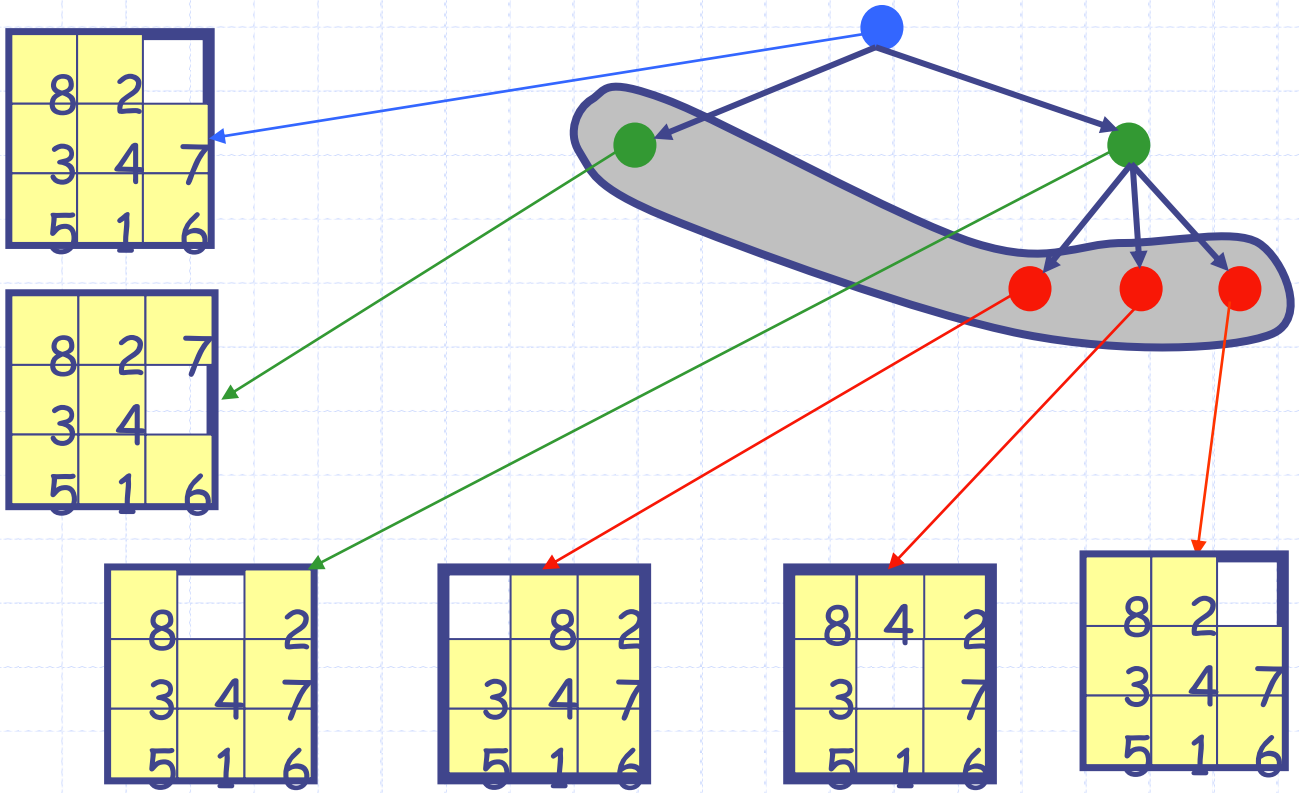
- The expansion of a node N of the search tree consists of:
  - Applying each legal action on STATE(N)
  - Generating a child of N for each new accessible state

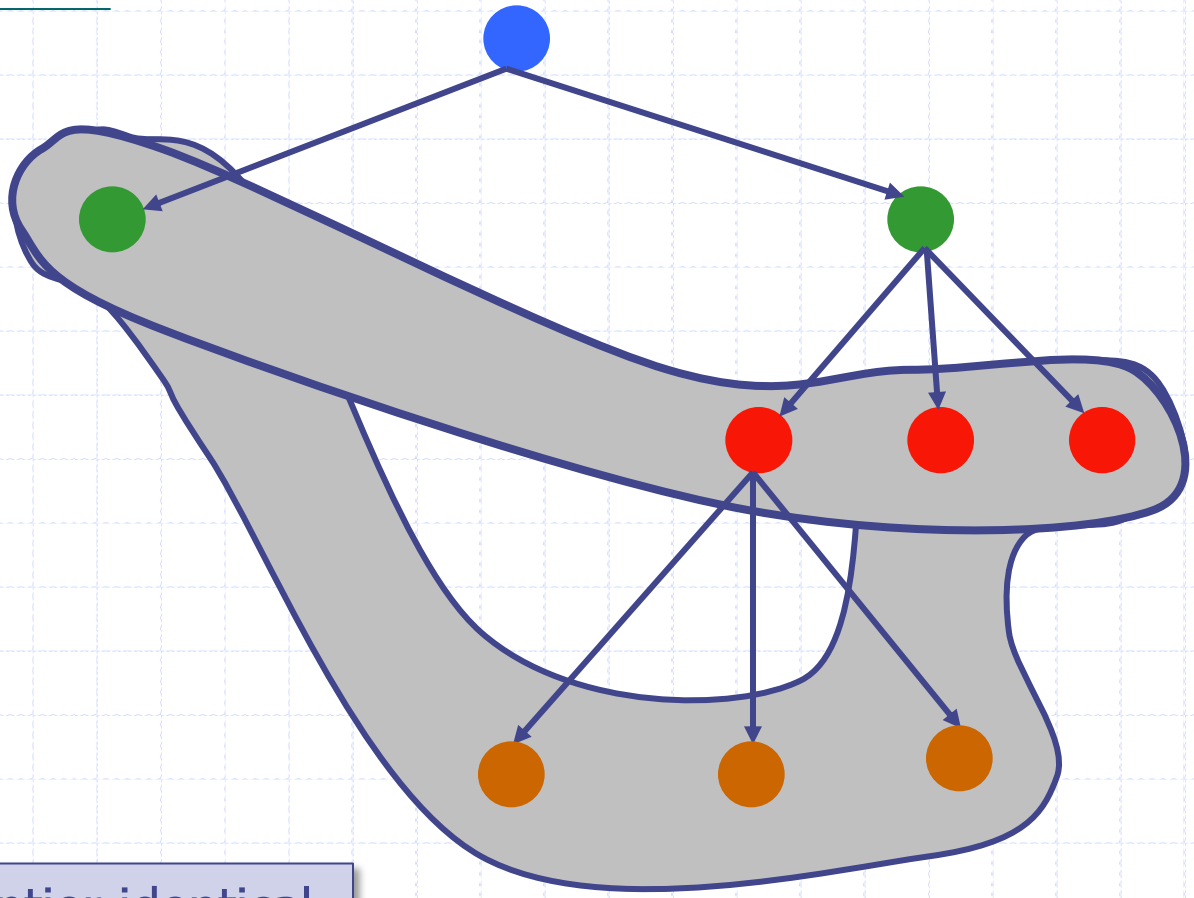
node generation
≠
node expansion!



Frontier of Search Tree

- The **frontier** is the set of all search nodes that haven't been expanded yet





Is the frontier identical to the set of leaves?

Search Strategy

- The frontier is the set of all search nodes that haven't been expanded yet
- The frontier is implemented as a priority queue FRONTIER
  - INSERT(node, FRONTIER)
  - POP(FRONTIER)
- The ordering of the nodes in FRONTIER defines the search strategy

Search Algorithm #1

1. If GOAL?(S_0) then return S_0
2. INSERT(N_0,FRONTIER)
3. Repeat:
   a. If EMPTY?(FRONTIER) then return failure
   b. N = POP(FRONTIER)
   c. s = STATE(N)
   d. For every state s′ in SUCCESSORS(s)
      i. Create a new node N′ as a child of N
      ii. If GOAL?(s′) then return path or goal state
      iii. INSERT(N′,FRONTIER)

SUCCESSORS(s) returns set of states reachable by single legal actions from s

Expansion of N

Performance Measures

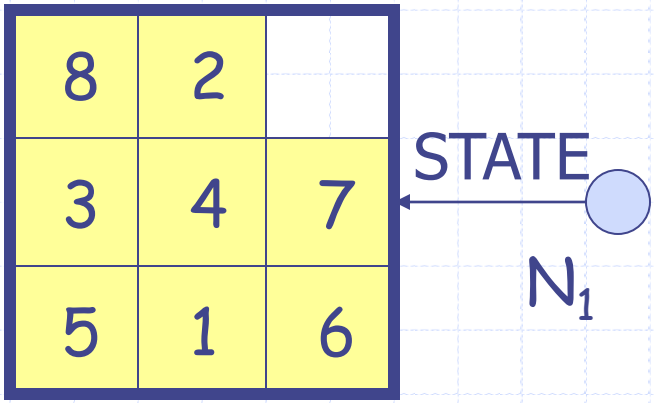
- Completeness
  A search algorithm is complete if it finds a solution whenever one exists
  [What about the case when no solution exists?]
- Optimality
  A search algorithm is optimal if it returns a minimum-cost path whenever a solution exists
- Complexity
  It measures the time and amount of memory required by the algorithm

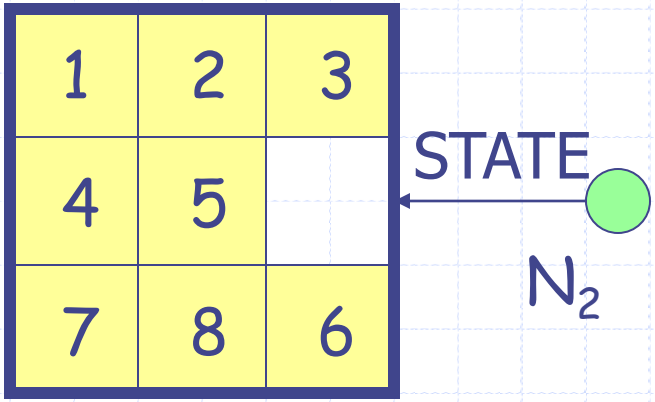
Blind vs. Heuristic Strategies

- Blind (or un-informed) strategies do not exploit state descriptions to order FRONTIER. They only exploit the positions of the nodes in the search tree

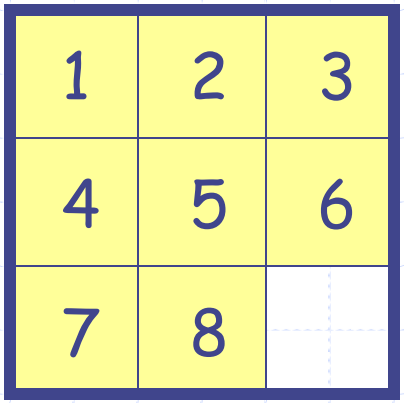
- Heuristic (or informed) strategies exploit state descriptions to order FRONTIER (the most "promising" nodes are placed at the beginning of FRONTIER)

Example



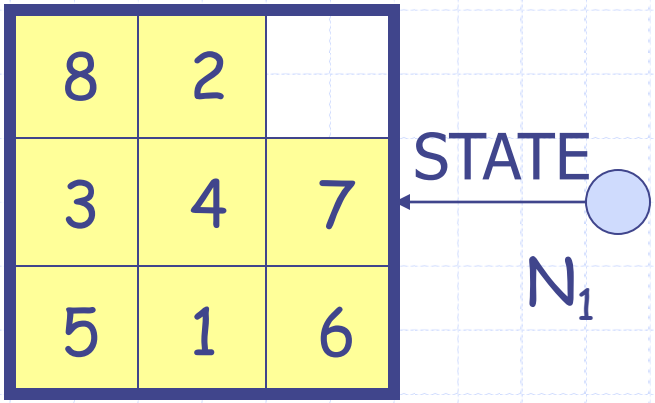


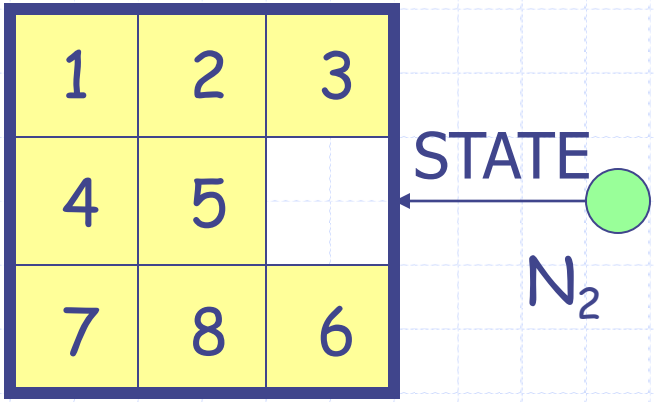
For a blind strategy, N_1 and N_2 are just two nodes (at some position in the search tree)



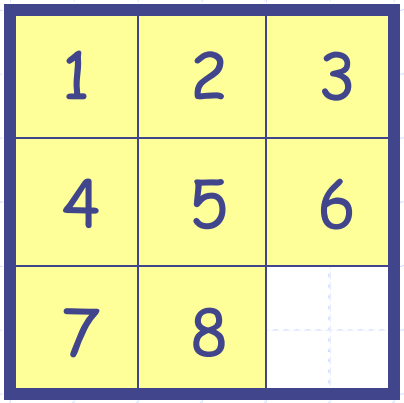
Goal state

Example





For a heuristic strategy counting the number of misplaced tiles, N_2 is more promising than N_1



Goal state

Remark

- Some search problems, such as the (n^2-1)-puzzle, are NP-hard
- One can't expect to solve all instances of such problems in less than exponential time (in n)
- One may still strive to solve each instance as efficiently as possible

This is the purpose of the search strategy