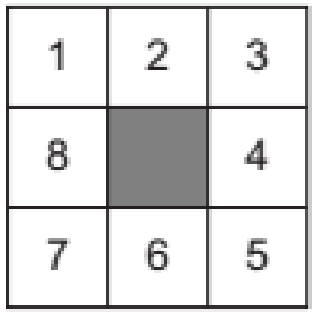
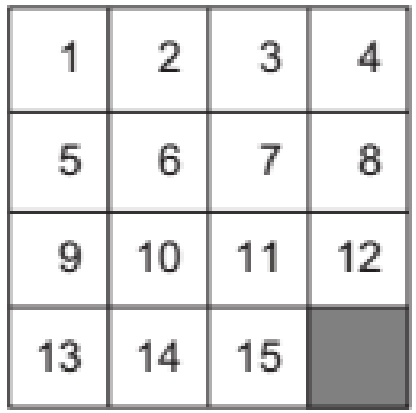
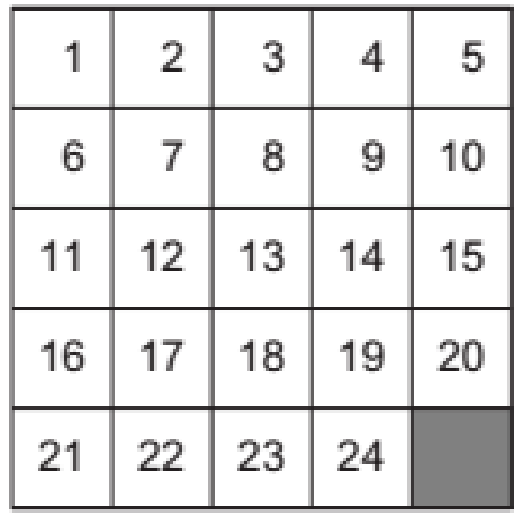
بسم اللّه الرحمن الرحیم

Systematic Search Algorithms (Chapter 3)

Examples of Search Problems: Sliding-Tile (Sliding-Block) Puzzles

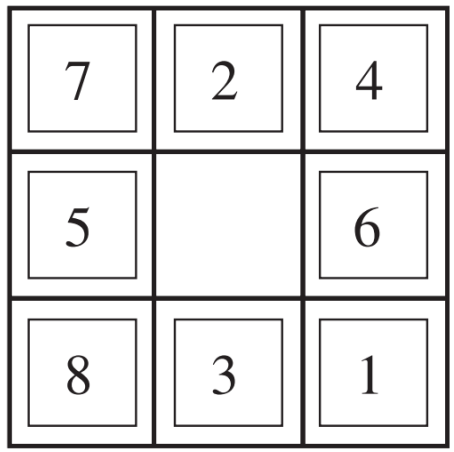




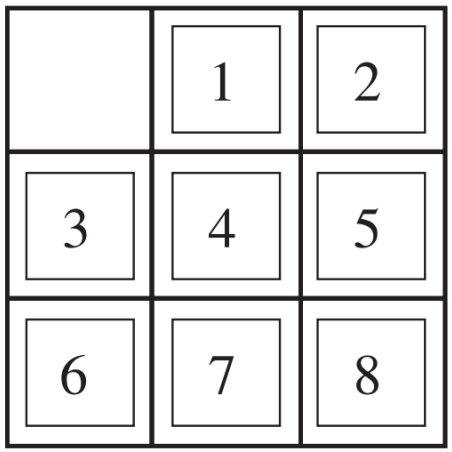


For n-puzzle, finding a solution is easy, but finding shortest solution is NP-hard.

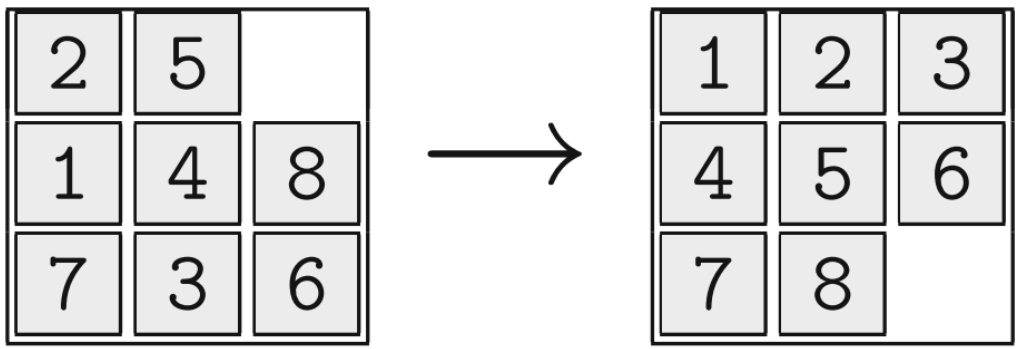
8-Puzzle



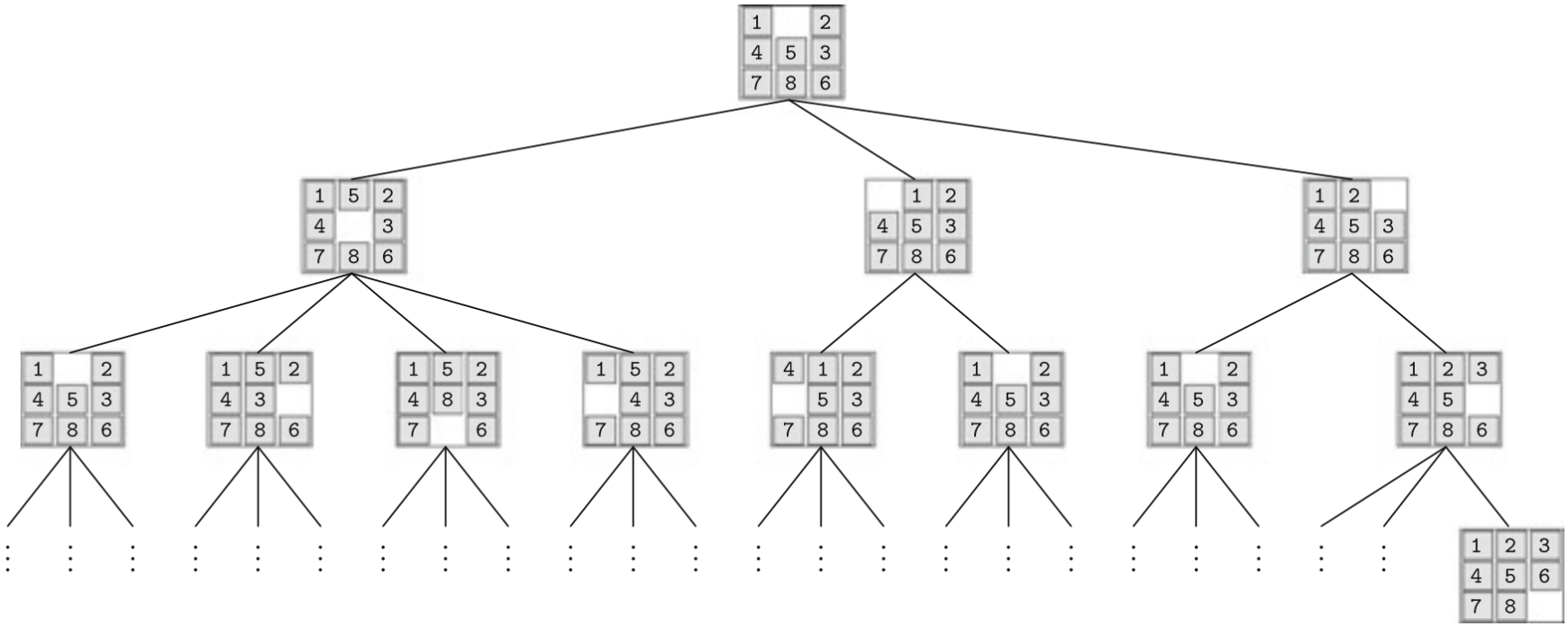
Start State



Goal State

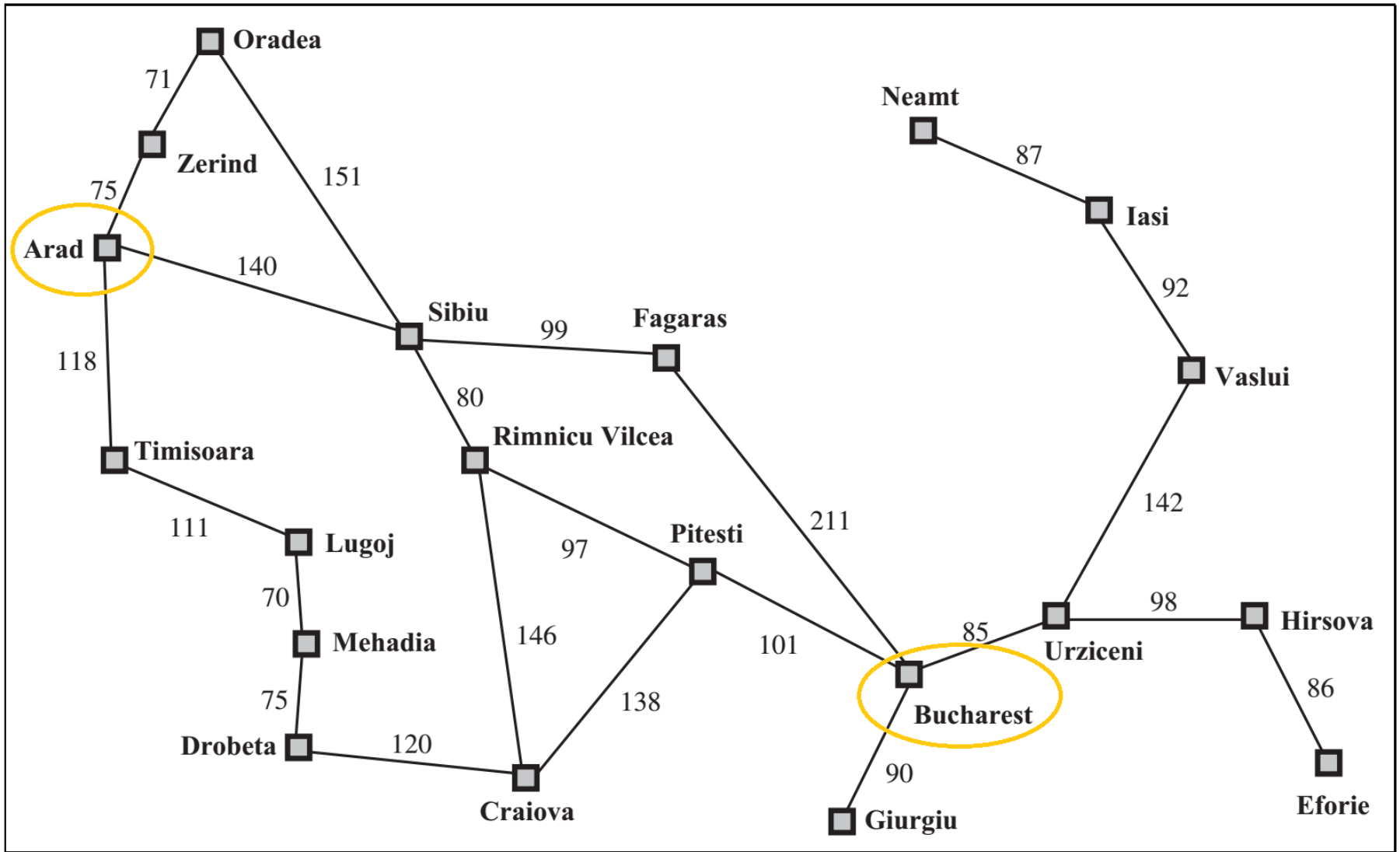


8-Puzzle



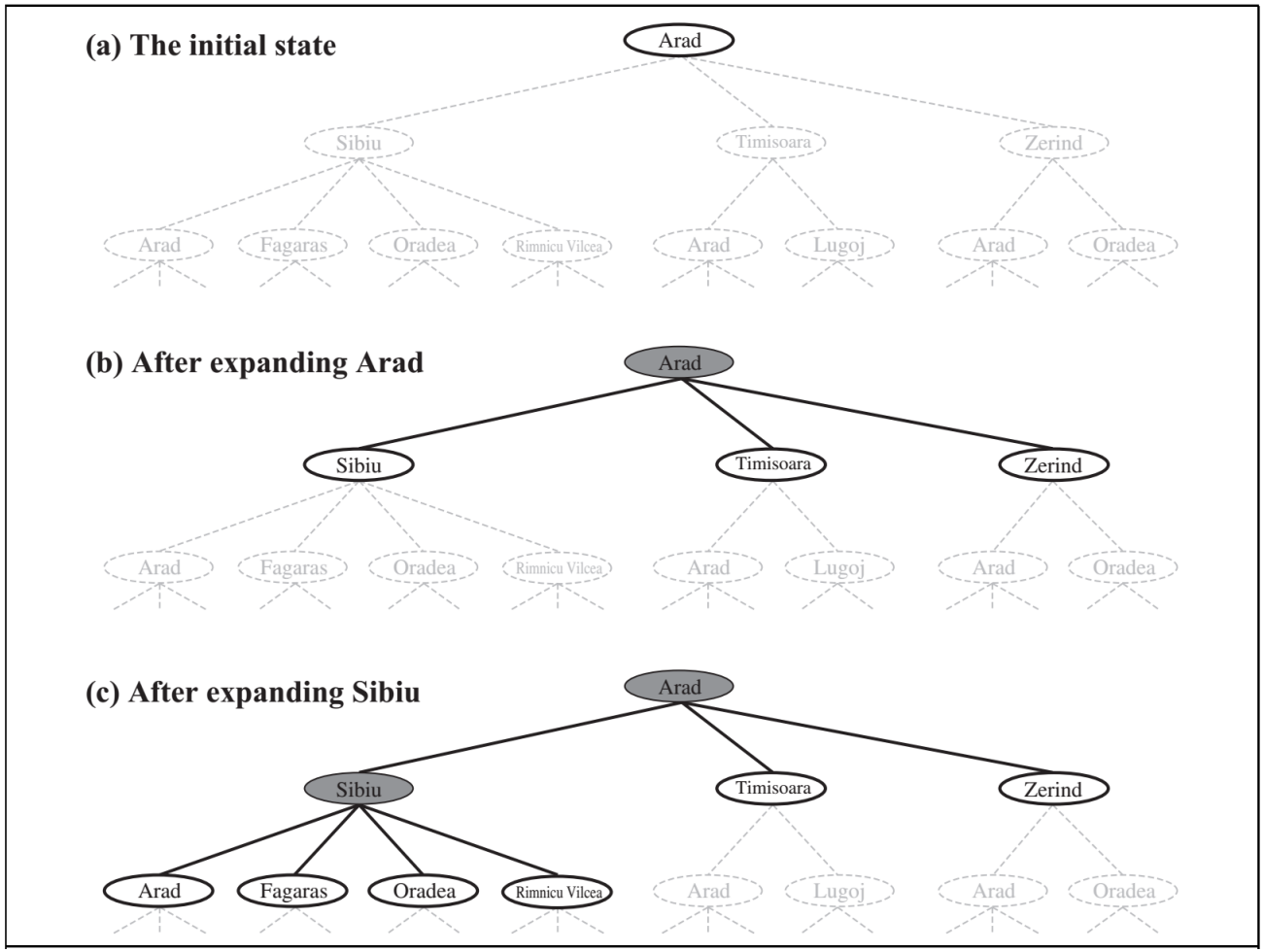
Be careful of repeated states!

Examples of Search Problems: Route-Finding Problem



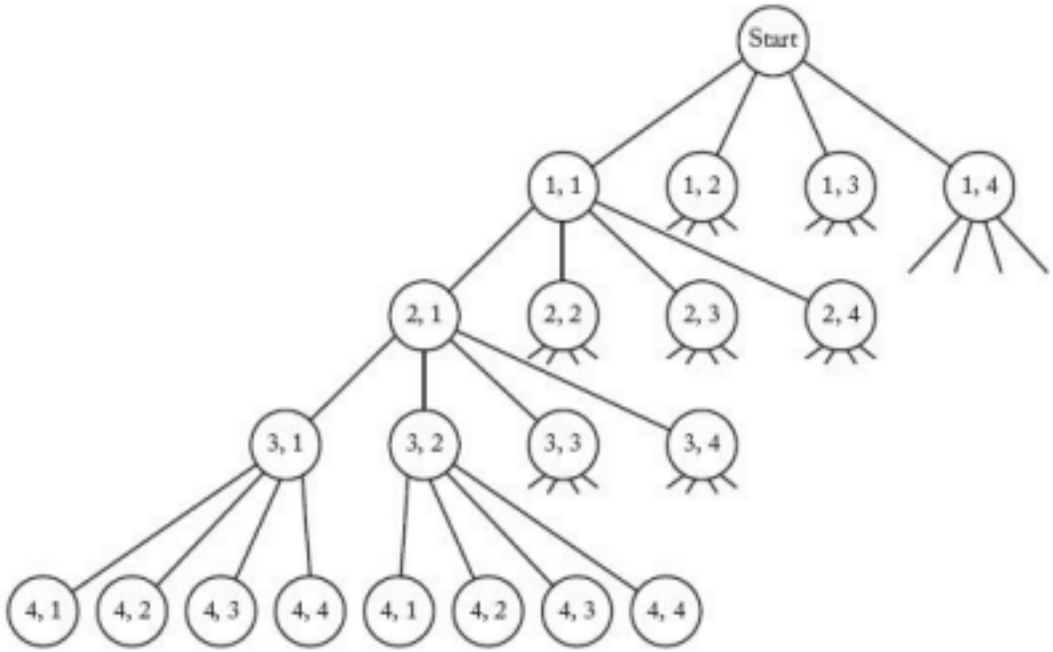
A simplified road map of part of Romania.

Examples of Search Problems: Route-Finding Problem



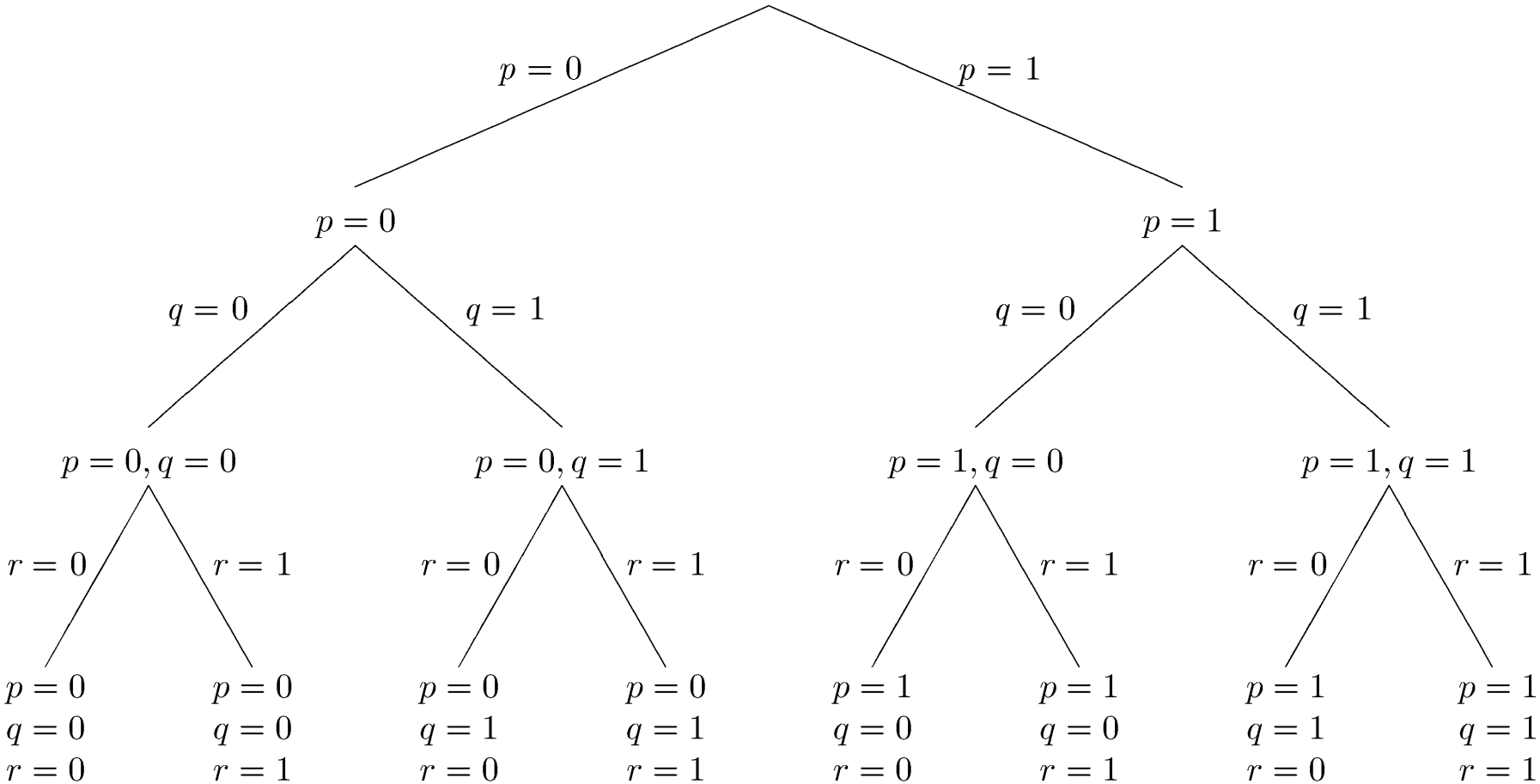
Be careful of repeated states!

Examples of Search Problems: The n-Queens Problem



A portion of the state space tree for the instance of the n-Queens problem in which $n = 4$. The ordered pair $\langle i, j \rangle$, at each node means that the queen in the ith row is placed in the jth column. Each path from the root to a leaf is a candidate solution.

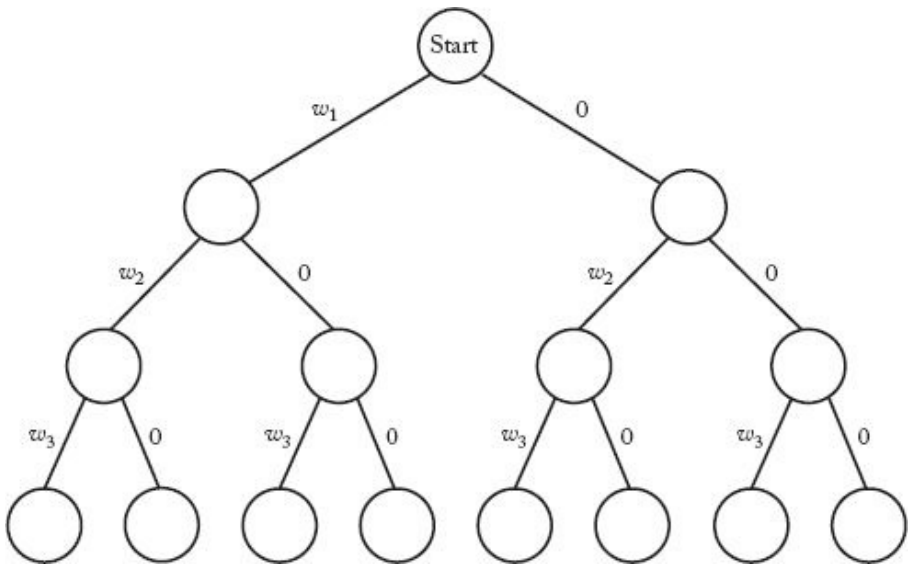
Examples of Search Problems: The Satisfiability Problem



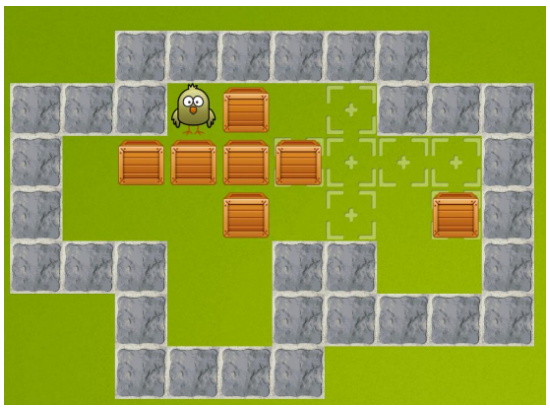
The Sum-of-Subsets Problem & The 0-1 Knapsack Problem

In the Sum-of-Subsets problem, there are n positive integers (weights) w_i and a positive integer W. The goal is to find all subsets of the integers that sum to W.

A state space tree for instances of the Sum-of-Subsets problem in which $n = 3$:



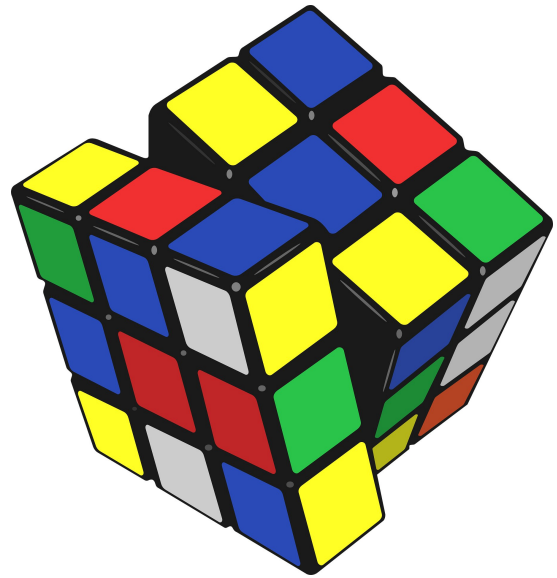
Examples of Search Problems: Miscellaneous



Sokoban

			6					9
	6				1	5		7
	9		5	7				
	5					1		
1		7				9		3
		2					7	
				3	2		1	
3		6	9				8	
4					7			

Sudoku



Rubik's Cube

- ☞ Graph Coloring (The m-Coloring Problem)
- ☞ The Hamiltonian Circuits Problem
- ☞ Traveling Salesperson Problem (TSP)

An informal description of the general tree-search and graph-search algorithms

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

---

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  initialize the explored set to be empty
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set

Measuring problem-solving performance

We can evaluate an algorithm's performance in four ways:

- ☞ **Completeness:** Is the algorithm guaranteed to find a solution when there is one?
- ☞ **Optimality:** Does the strategy find the optimal solution? (A solution that is best according to some criterion is called an optimal solution.)
- ☞ **Time complexity:** How long does it take to find a solution?
- ☞ **Space complexity:** How much memory is needed to perform the search?

Uninformed and Informed Search Strategies

A problem determines the graph, the start node and the goal but not which path to select from the frontier. This is the job of a search strategy.

Search algorithms all share this basic structure; they vary primarily according to how they choose which state to expand next—the so-called search strategy.

Different strategies are obtained by modifying how the selection of paths in the frontier is implemented.

Uninformed search strategies do not take into account the location of the goal. Intuitively, these algorithms ignore where they are going until they find a goal and report success.

Uninformed and Informed Search Strategies

Uninformed search (also called blind search): The term means that the strategies have no additional information about states beyond that provided in the problem definition. All they can do is generate successors and distinguish a goal state from a non-goal state. All search strategies are distinguished by the order in which nodes are expanded.

Strategies that know whether one non-goal state is "more promising" than another are called informed search or heuristic search strategies.