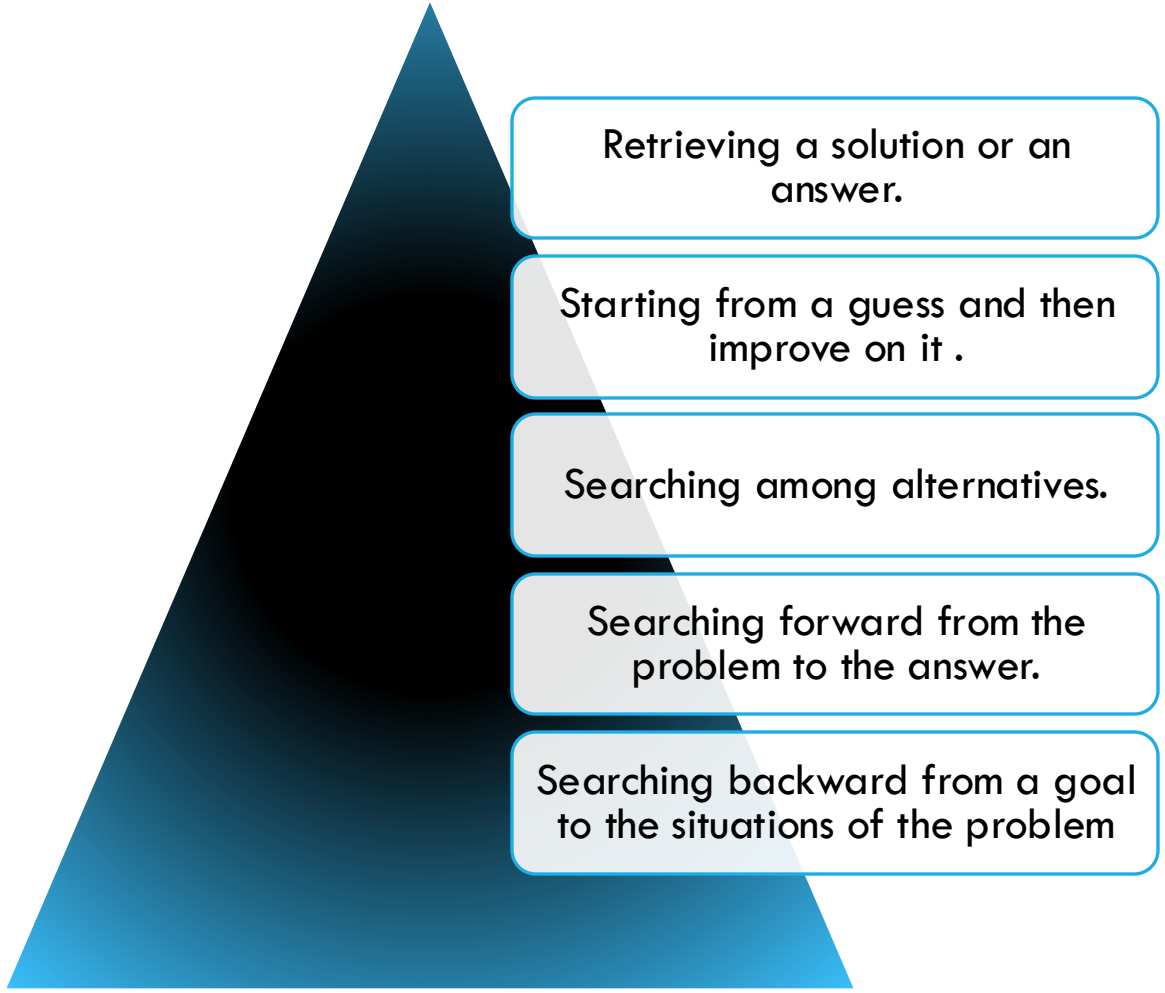
Solution Strategies for Ill-structured Problems

SOLUTION STRATEGIES FOR ILL-STRUCTURED PROBLEMS



- Retrieving a solution or an answer.
- Starting from a guess and then improve on it .
- Searching among alternatives.
- Searching forward from the problem to the answer.
- Searching backward from a goal to the situations of the problem

OPTIMIZATION PROBLEMS

Optimization problem is the **problem** of finding the "best" solution from all feasible solutions subject to a set of constraints.

Optimization problems are ubiquitous.

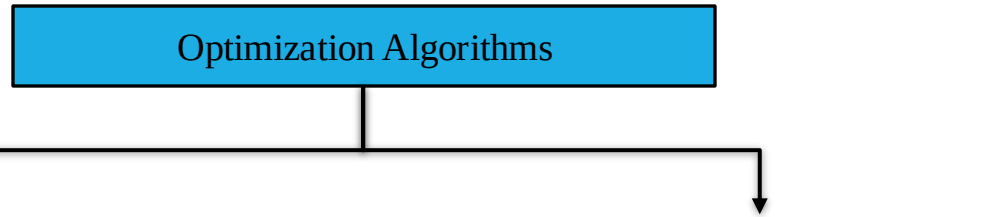
NP-Hard or NP-Complete [very hard to solve]

- NP-problems can be solved by exhaustive search
- The size of the instance grow the running time forbiddingly large even for fairly small sized problems

Key part of our day to day lives.

OPTIMIZATION METHODS

Optimization techniques are ***search methods*** where the goal is to find a solution to an optimization problem [a given quantity is optimized subject to a set of constraints].



Exact Algorithms

- Find the optimal solution
- High computational cost

Approximate Algorithms

- Find near-optimal solution
- Low computational cost

OPTIMIZATION METHODS

Approximate methods to solve combinatorial optimization problems:

- Heuristics is a solution strategy or rules by trial and error to produce acceptable (optimal or sub-optimal) solutions to complex problems in a reasonably practical time

Heuristics aims to efficiently generate good solutions, but does not guarantee optimality

Heuristics characteristics

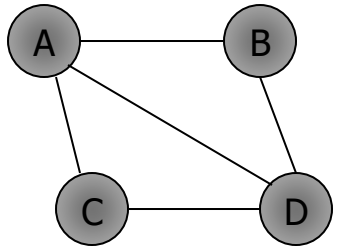
- Short" running times
- Easy to implement
- Flexible
- Simple

OPTIMIZATION METHODS

Approximate algorithms can be divided to

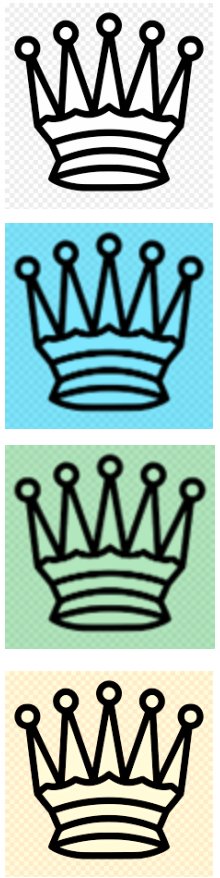
a. Constructive methods

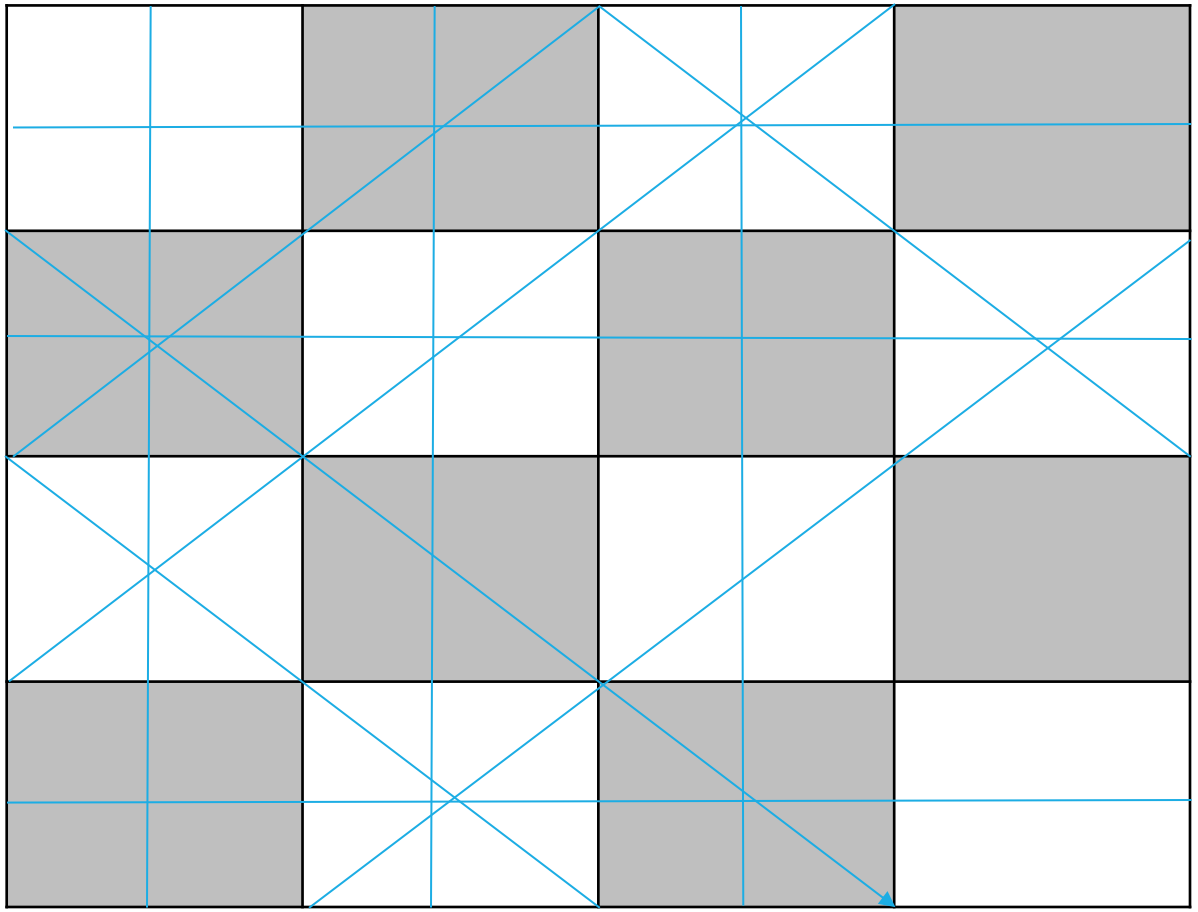
- start from scratch and try to build the complete solution by adding one component at a time



Search space for constructive methods

4-QUEEN PROBLEM

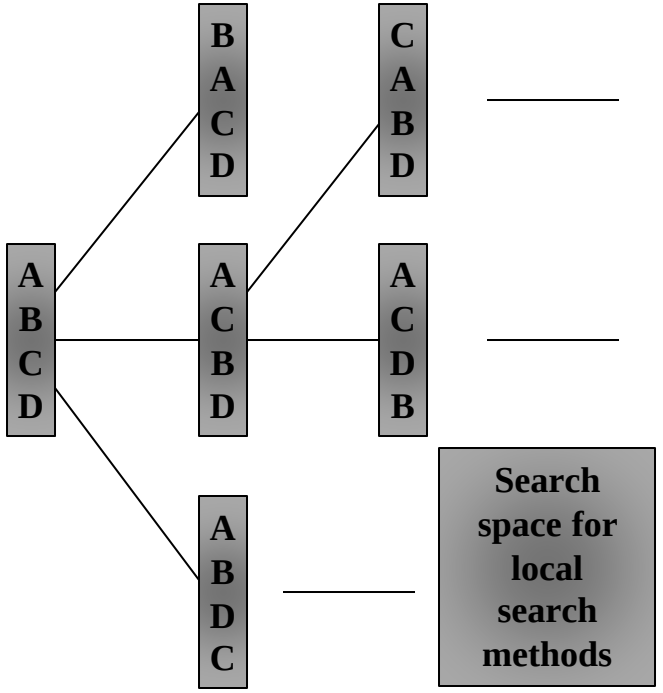




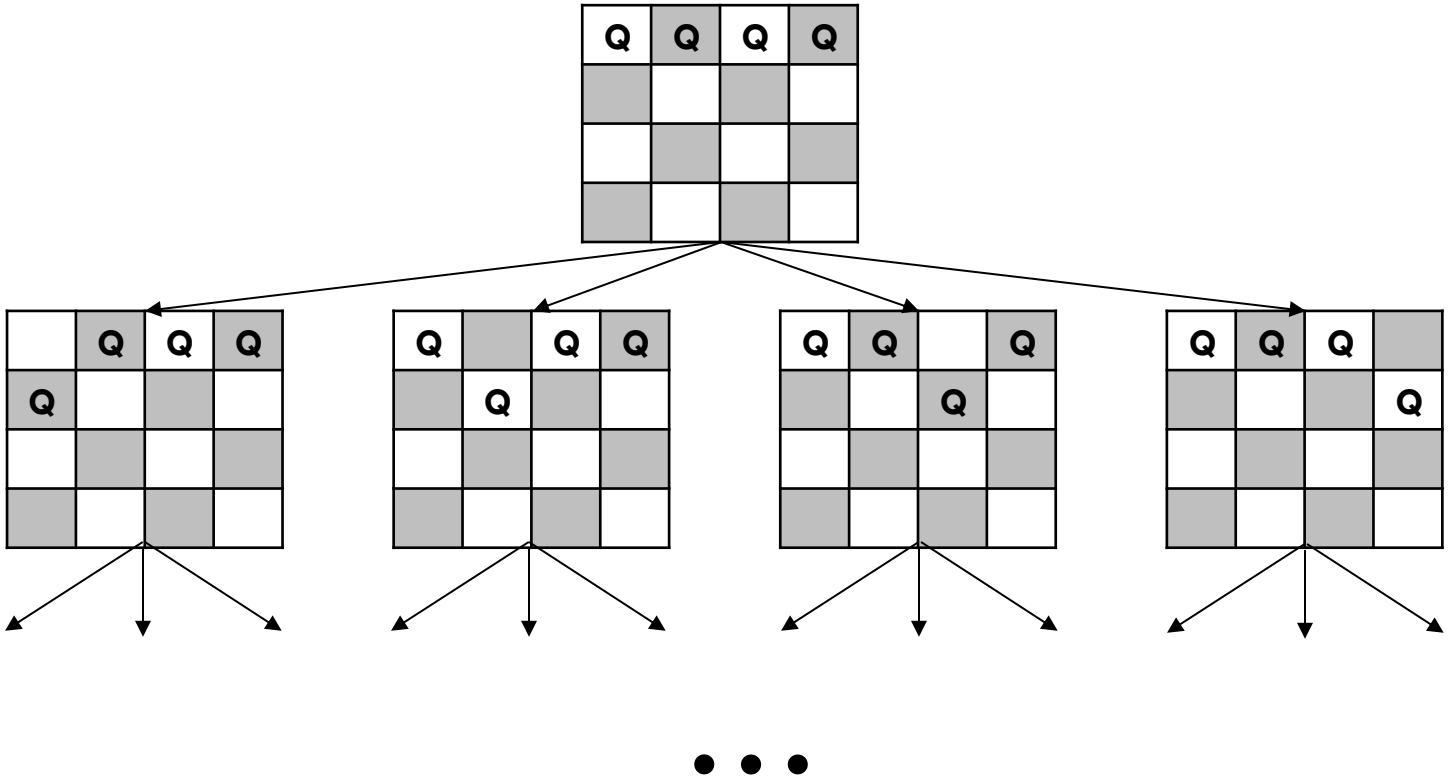
OPTIMIZATION METHODS

b. Local search methods

- starts from an initial solution and iteratively tries to replace the current solution with better one



OPTIMIZATION METHODS



Local Search Methods

Goal and Problem Formulation

PROBLEM SOLVING BY SEARCHING

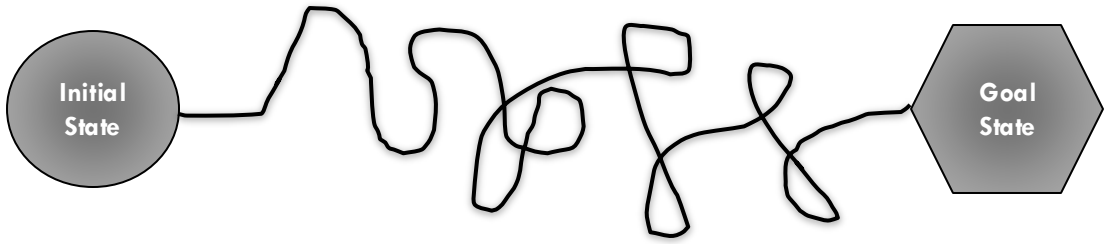
Search is a sequence of actions that should be taken to reach a goal.

Many optimization problems can described as search problem, often be solved using search algorithms.

- Central in many AI systems:
  - Theorem proving, VLSI layout, game playing, navigation, scheduling, etc.

The search process involves moving from ***state-to-state*** in the problem space to find a ***goal*** (or to terminate without finding a goal).

BUILDING GOAL-BASED AGENTS

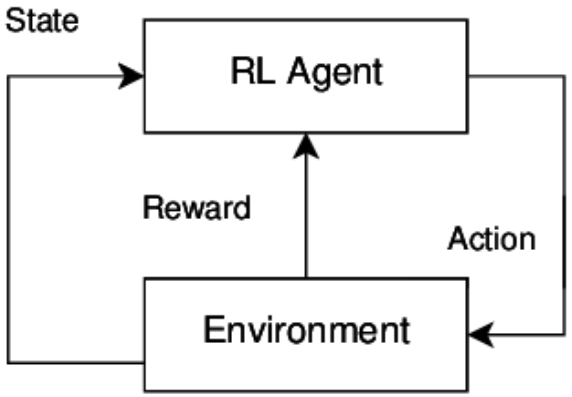


Goal-based Search vs Utility-based search

- Goal-based agents use information about ***environment & current*** state to reach an end goal.
- Utility-based agents also take into consideration the ***performance*** of accomplishing the goal.

To build a goal-based agent we need to answer the following questions:

- What is a state?
- What is the goal to be achieved?
- What are the actions?



PROBLEM SOLVING BY SEARCHING

What *relevant* information is necessary to encode in order to describe the state of the world, describe the available transitions, and solve the problem?

Two requirements to use search:

- **Goal Formulation.**
- **Problem formulation.**

GOAL AND PROBLEM FORMULATION

In goal formulation, we decide which aspects we are ***interested*** in and which aspects can be ***ignored***.

In problem formulation, we decide how to manipulate the ***important*** aspects, and ignore the others.

So, without doing goal formulation, if we do the problem formulation, we would not know what to include in our problem and what to leave, and what should be achieved.

So problem formulation must follow goal formulation. That means problem formulation must be done only after the goal formation is done.

GOAL FORMULATION

Need goals to **limit search** and **allow termination.**

- Define a goal
  - What is the agent ***searching for***?
    - Certainly psychologists and motivational speakers always stress the importance of people establishing clear goals.
    - Requires defining a **"goal test"** so that we know what it means to have achieved/satisfied our goal.
    - This is a hard question that is rarely tackled in AI, usually assuming that the system designer or user will specify the goal to be achieved.
- Define the solution
  - The goal itself?
  - The path (i.e. sequence of actions) to get to the goal?

GOAL FORMULATION

If you don't know where you are going,
You might never get there!

Yogi Berra

PROBLEM FORMULATION

Compact representation of problem space.

- Represent the search space by states

Define **actions** valid for a given state.

- Define the actions the agent can perform and their cost
- Define a transition model
- Neighborhood of a state S is the states S can move to after any action is taken.

Define **cost** of actions.

PROBLEM FORMULATION

Search requires a well-defined problem space including:

- **State space**
  - A state can represent either a complete configuration of a problem (e.g., 8-puzzle) or a partial configuration of a problem (e.g., routing).
- **Initial state**
  - A search starts from here.
- **Goal state and goal test**
  - A search terminates here.
- **Sets of actions**
  - This allows movement between states (successor function).
- **Concept of cost** (action and path cost)
  - This allows costing a solution.

The above defines a **well-defined state-space formulation** of a problem.

CLOSED WORLD ASSUMPTION

We will generally use the **Closed World Assumption**.

All necessary information about a problem domain is available in each percept so that ***each state is a complete description of the world***.

There is no incomplete information at any point in time.

Environment

- Fully observable
- Deterministic
- Sequential
- Static
- Discrete

EXAMPLE – 8 PUZZLE

Action

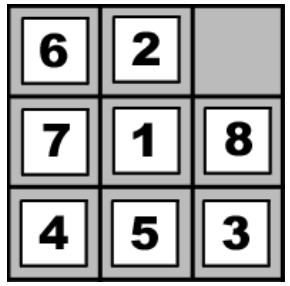
- Move blank left, right, up or down, provided it does not get out of the game

Goal test

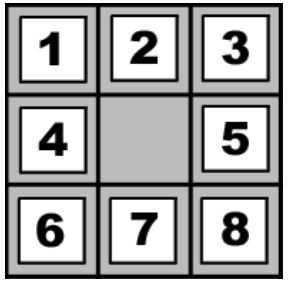
- Are the tiles in the "goal state" order?

Cost

- Each move costs 1
- Path cost is the sum of moves

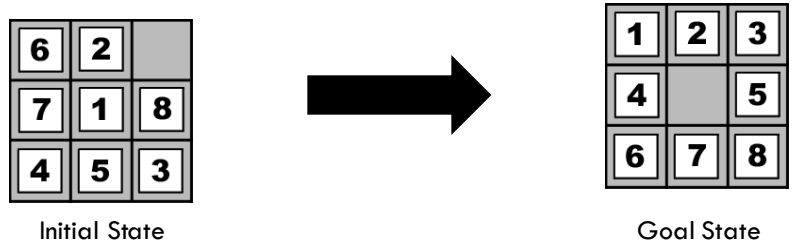


Initial State



Goal State

EXAMPLE – 8 PUZZLE: PROBLEM FORMULATION



Initial State

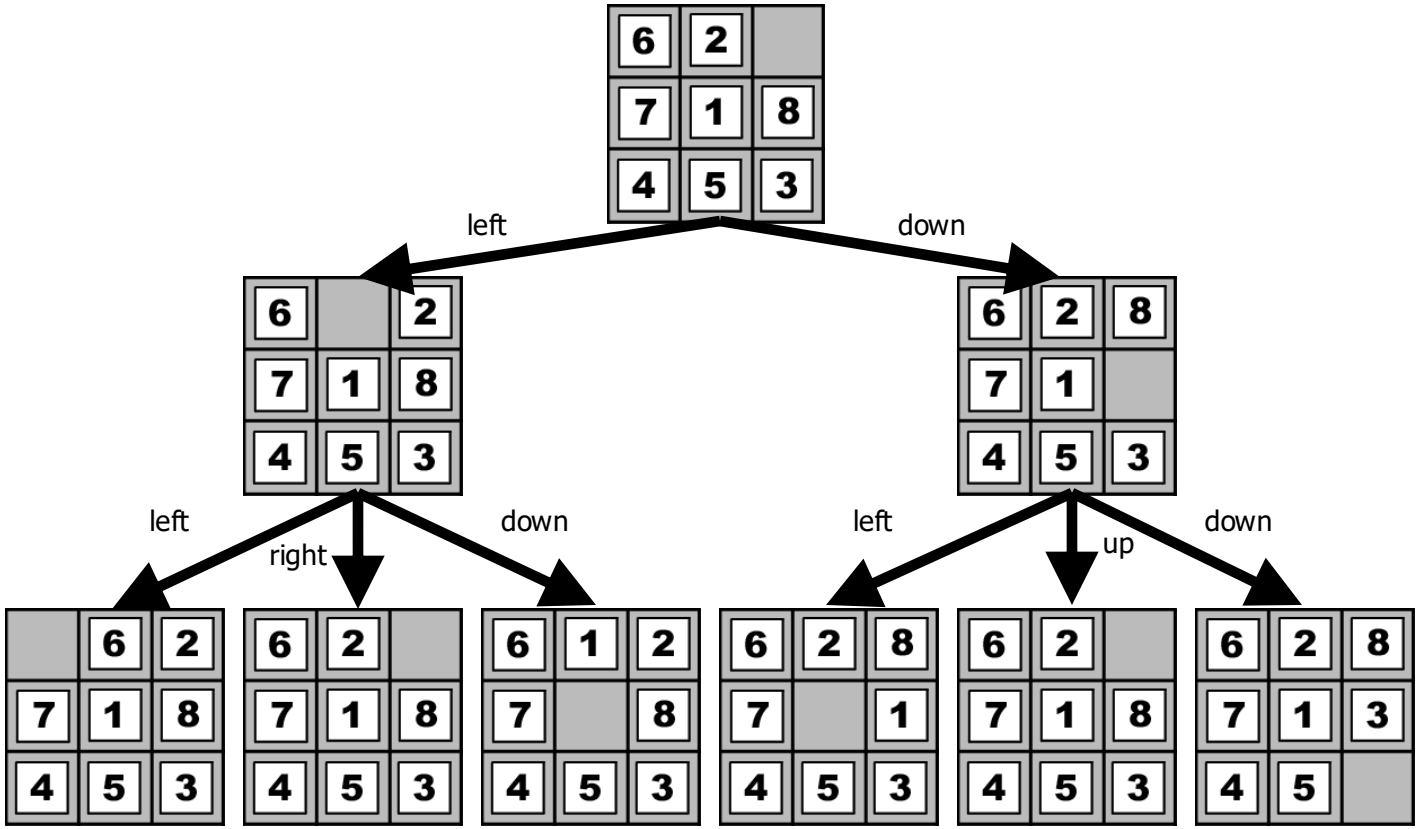
Goal State

Initial arrangement of tiles → Desired arrangement.

- **States:** encode location of each of 8 tiles and the blank.
- **Initial State:** any arrangement of tiles.
- **Goal State:** predefined arrangement of tiles.
- **Actions:** slide blank UP, DOWN, LEFT or RIGHT (without moving off the grid).
- **Goal Test:** position of tiles and blank match goal state.
- **Cost:** sliding the blank is 1 move (cost of 1). Path cost will equal the number of moves from initial state to goal state.

Solution is a path of moves to get to the goal state. Fewest moves is best.

EXAMPLE – 8 PUZZLE: PROBLEM FORMULATION



EXAMPLE - ROUTE FINDING

Travelling salesman problem:

Find the shortest round trip to visit each city exactly once

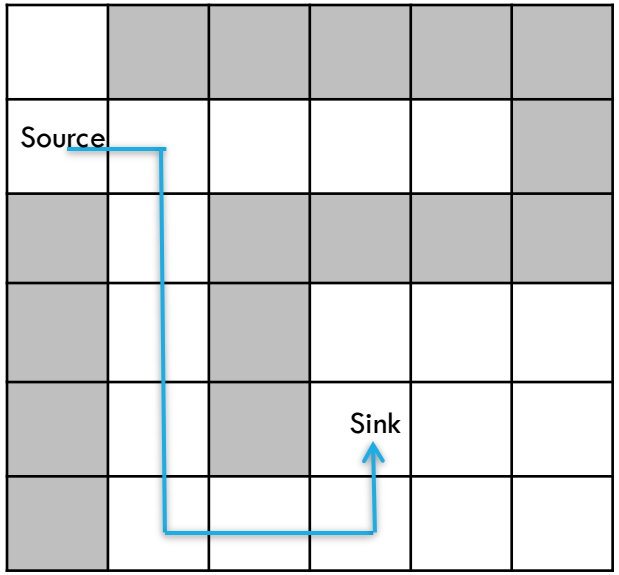
- Set of actions
  - Move to an unvisited city
- Goal test
  - Is the agent in the initial city after having visited every city?
- Concept of cost
  - Action cost: distance between cities
  - Path cost: total distance travelled

Example – Route Finding: Problem Formulation

Problem is to connect a source to a sink while avoiding obstacles on 2-D grid.

- **States:** ordered pair (x,y) of the trace head.
- **Initial State:** trace at location of source.
- **Goal State:** trace at location of sink.
- **Actions:** move trace head UP, DOWN, LEFT or RIGHT (without moving off the grid and avoiding obstacles).
- **Goal Test:** trace at location of sink.
- **Cost:** moving trace head costs 1. Path cost is length of trace.

Solution might be (i) trace with shortest path or (ii) a path if one even exists.

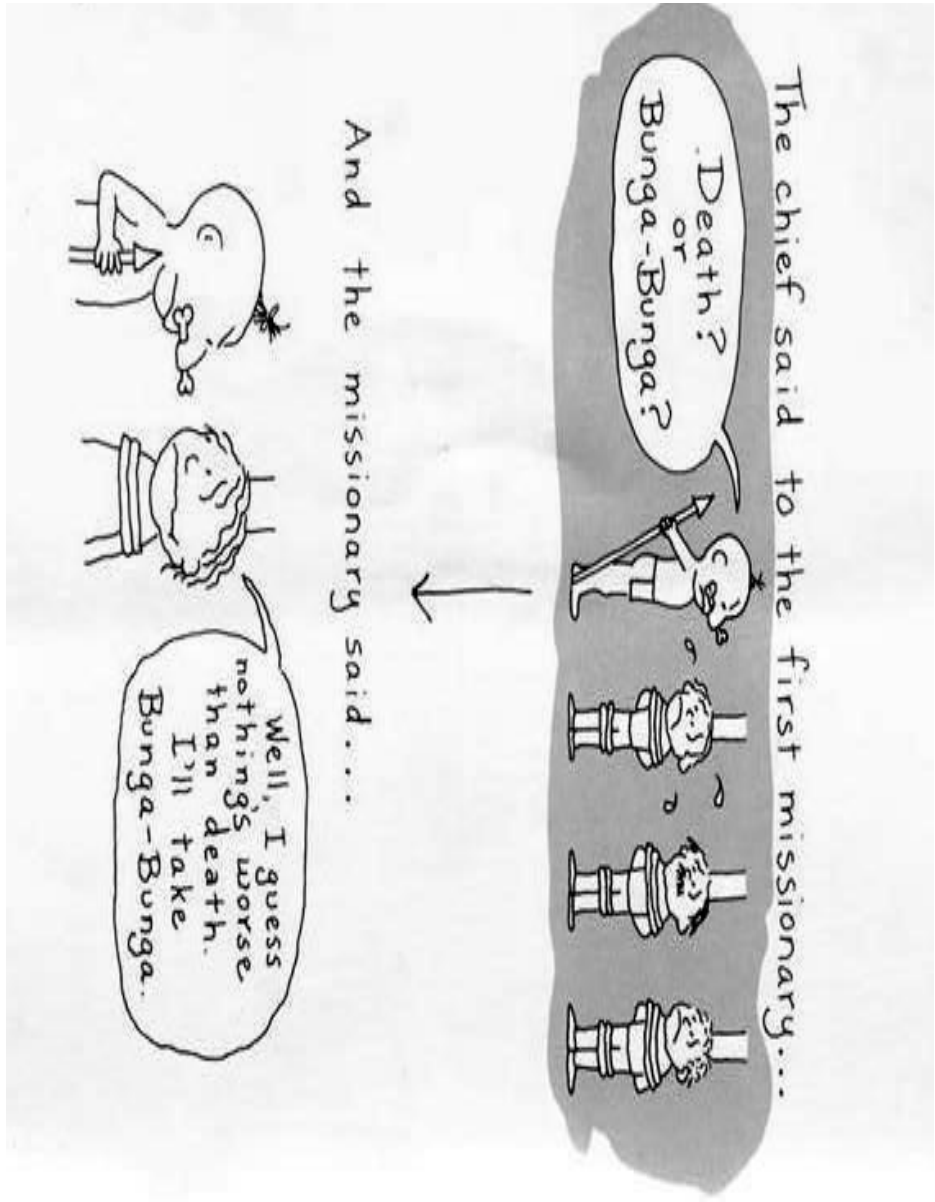


MISSIONARIES AND CANNIBALS

In the missionaries and cannibals problem, three missionaries and three cannibals must cross a river using a boat which can carry at most two people, under the constraint that, for both banks, if there are missionaries present on the bank, they cannot be outnumbered by cannibals (if they were, the cannibals would eat the missionaries). The boat cannot cross the river by itself with no people on board. And, in some variations, one of the cannibals has only one arm and cannot row.

Goal/ Goal Test:
State:
Initial State: **Actions:**

Cost:



MISSIONARIES AND CANNIBALS FORMULATION

STAT SPACE

State space: triple (x,y,z) with $0 \leq x,y,z \leq 3$, where x,y, and z represent the number of missionaries, cannibals and boats currently on the original bank.

Missionaries and Cannibals Solution

Initial State: (3,3,1)

Successor function: From each state, either bring one missionary, one cannibal, two missionaries, two cannibals, or one of each type to the other bank.

Note: Not all states are attainable (e.g., (0,0,1)), and some are illegal.

Goal State: (0,0,0)

Path Costs: 1 unit per crossing

Terminologies and Fundamentals

SEARCH TREES

Search trees are superimposed over top of the graph representation of a problem.

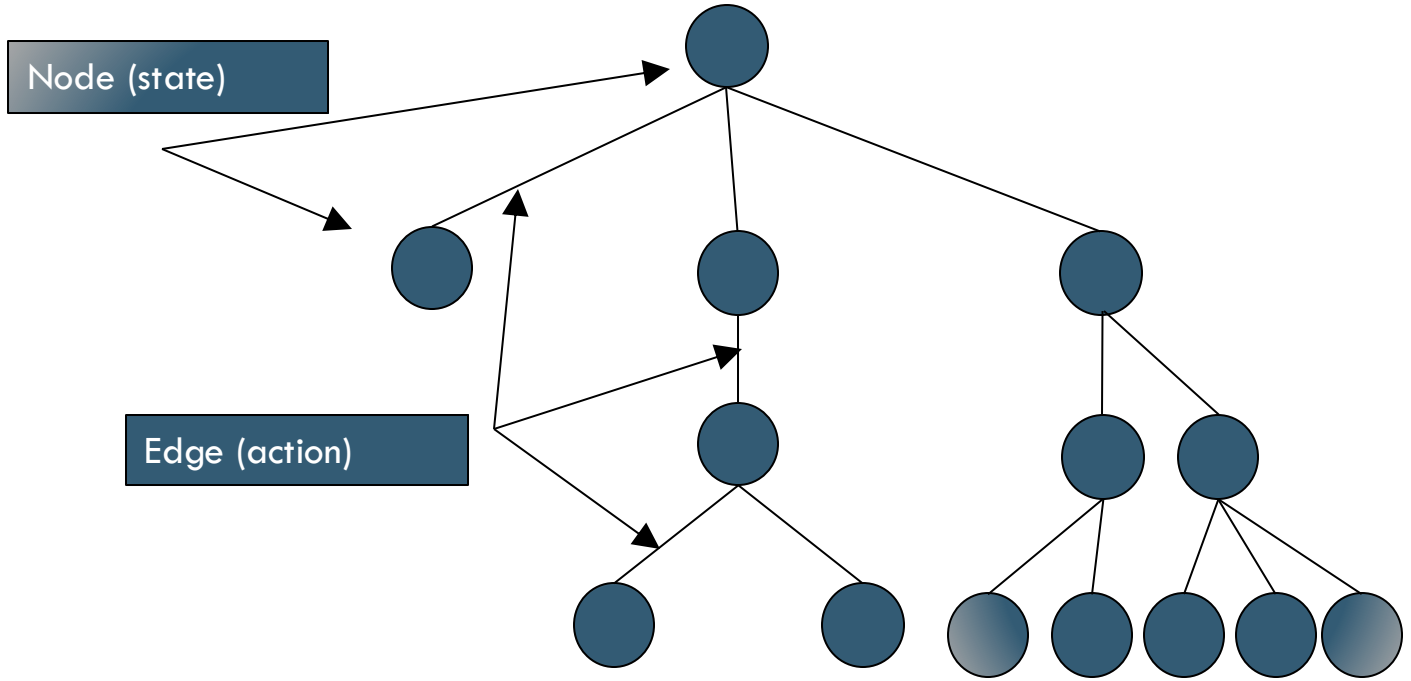
While the graph might be finite, the search tree can be either finite or infinite.

- Infinite if we allow repeated states due to reversible actions and/or cycles of actions.

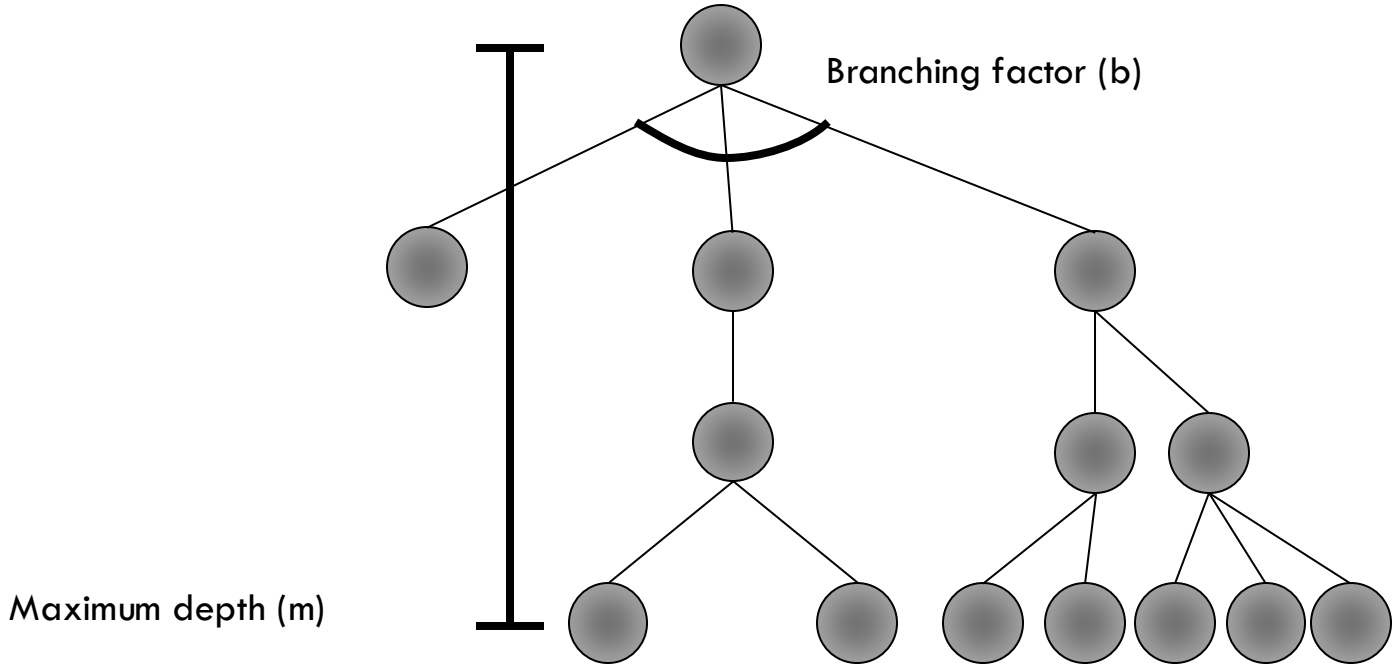
Some useful terminology:

- The maximum number of children possible for a node, b, in a search tree is called the **branching factor**.
- A finite tree has a **maximum depth**, m.
- Any node in the search tree occurs at a **level**, l, in the tree (or depth: d).

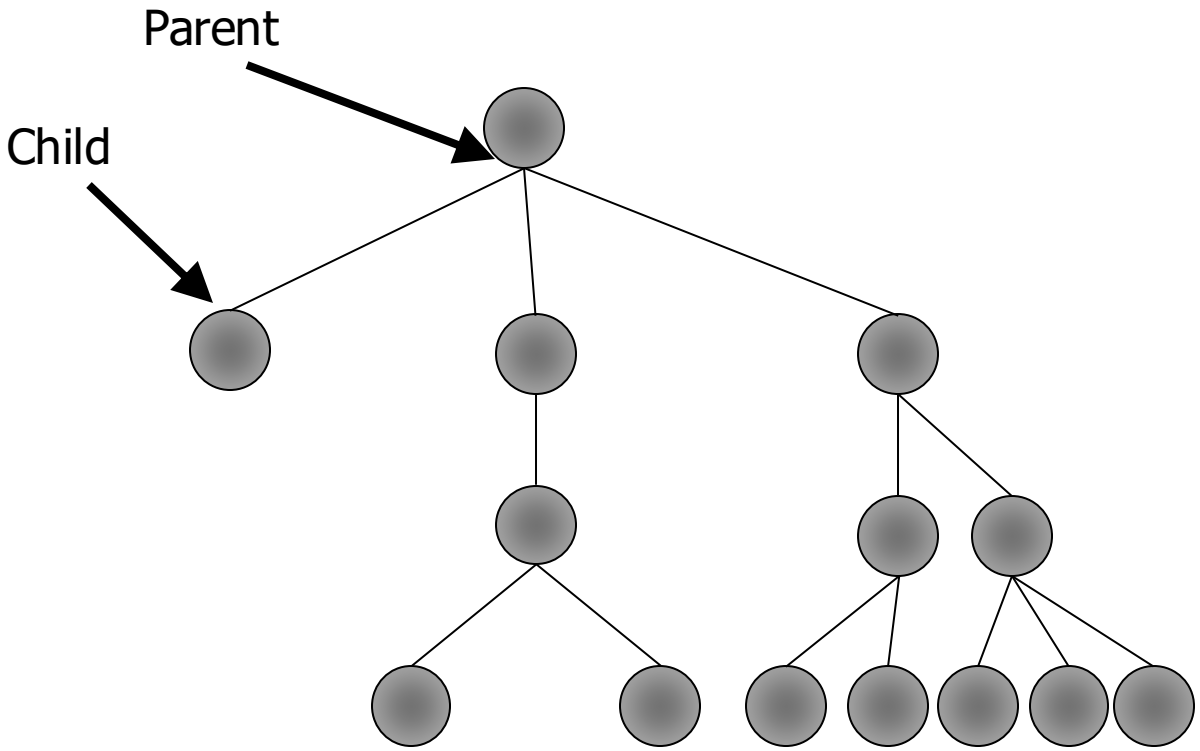
SEARCH TREE - TERMINOLOGY



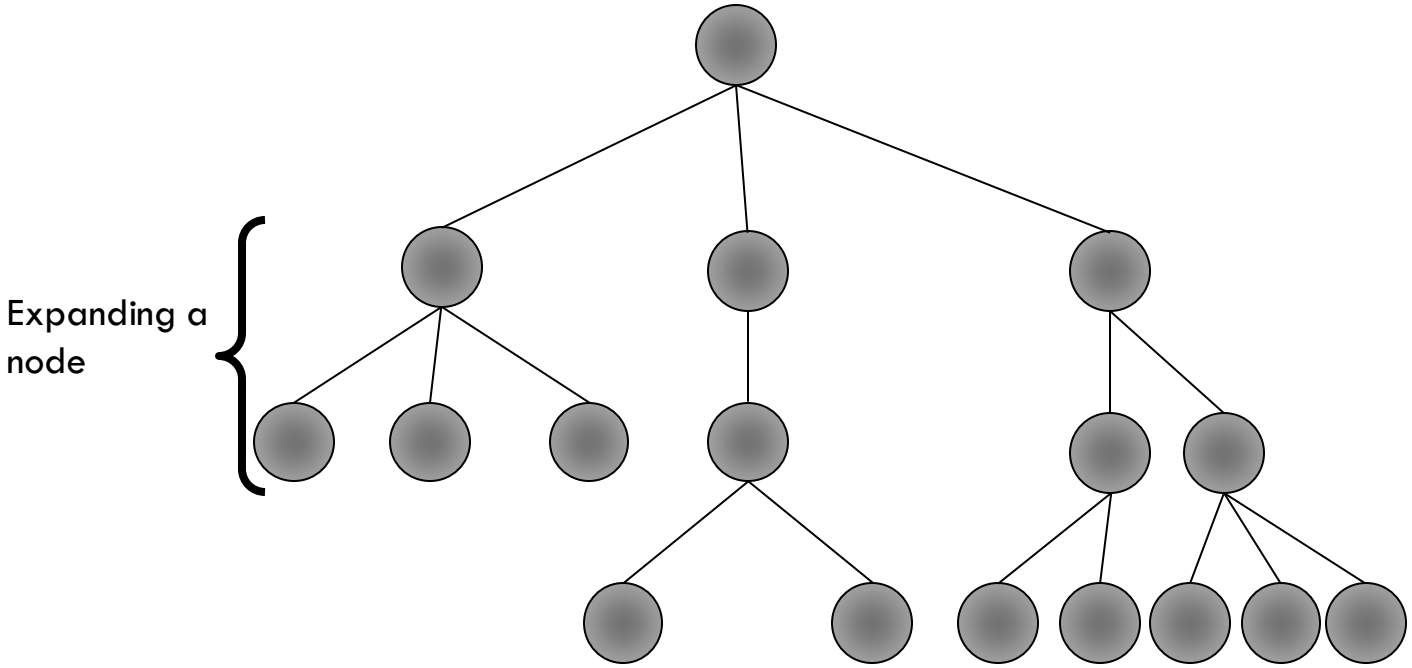
SEARCH TREE - TERMINOLOGY



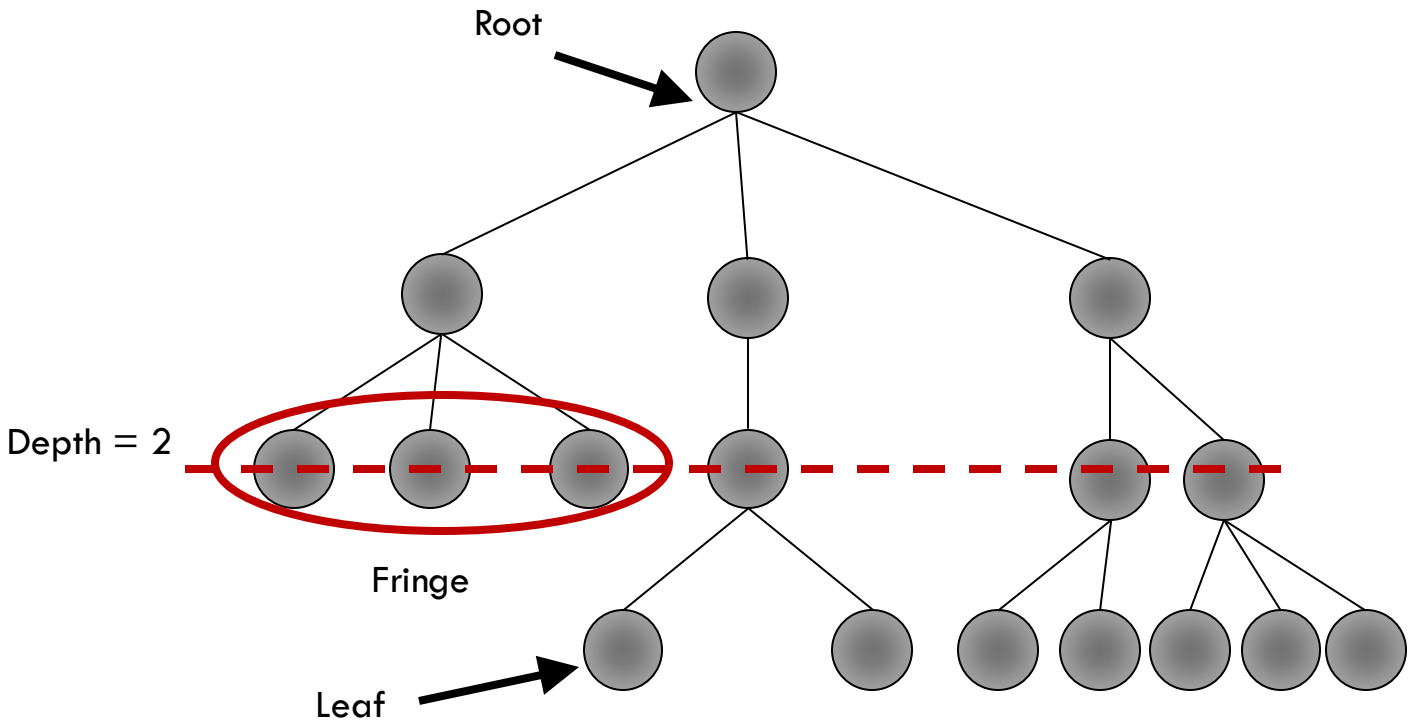
SEARCH TREE - TERMINOLOGY



SEARCH TREE - TERMINOLOGY

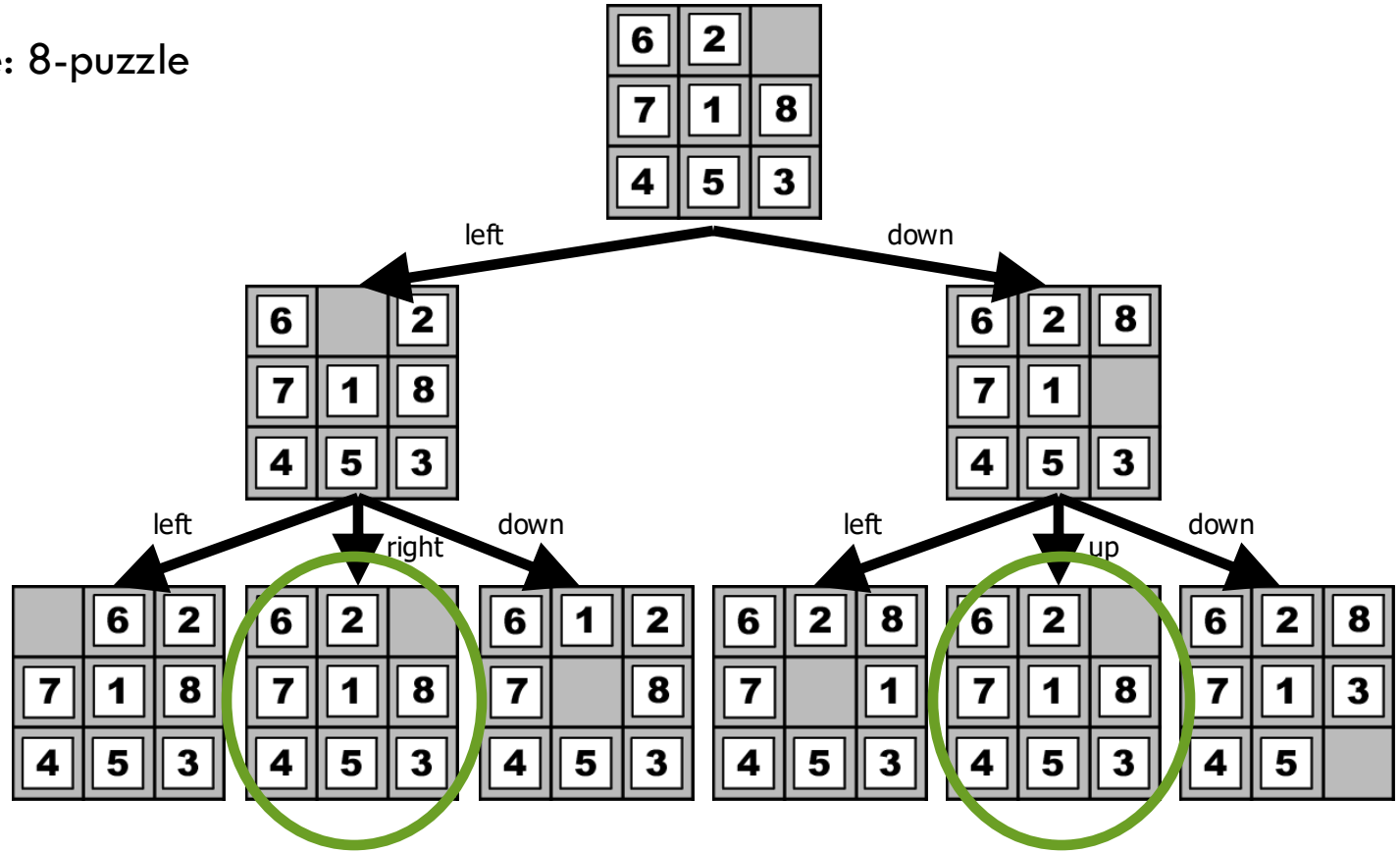


SEARCH TREE - TERMINOLOGY



REPEATED STATES

Example: 8-puzzle



PROPERTIES OF SEARCH ALGORITHMS

Completeness

- Is the algorithm guaranteed to find *a* goal node, if one exists?

Optimality

- Is the algorithm guaranteed to find *the best* goal node, i.e. the one with the cheapest path cost?

Time complexity

- How many nodes are generated?

Space complexity

- What's the maximum number of nodes stored in memory?

TEMPLATE OF GENERIC SEARCH

Given concept of graph and search tree, generic search is a repetition of **choose, test, and expand**.

A particular search strategy influences how we choose the next node to consider in the search! (this is where types of searches differ).

Generally, a **queue** structure is used to store nodes on the fringe to be expanded.

Different search strategies use different queue structures.