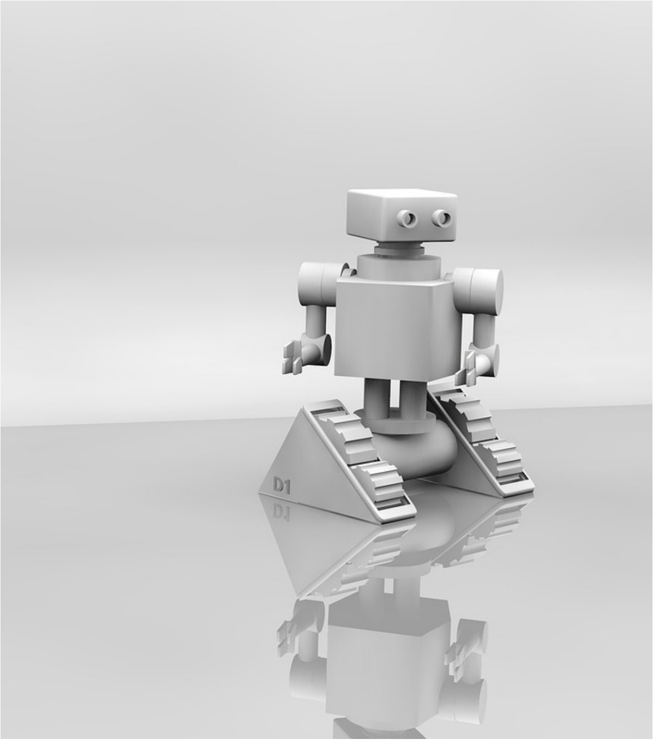


CPSC 481
Artificial Intelligence

Dr. Mira Kim
Mira.kim@fullerton.edu

What we will cover today

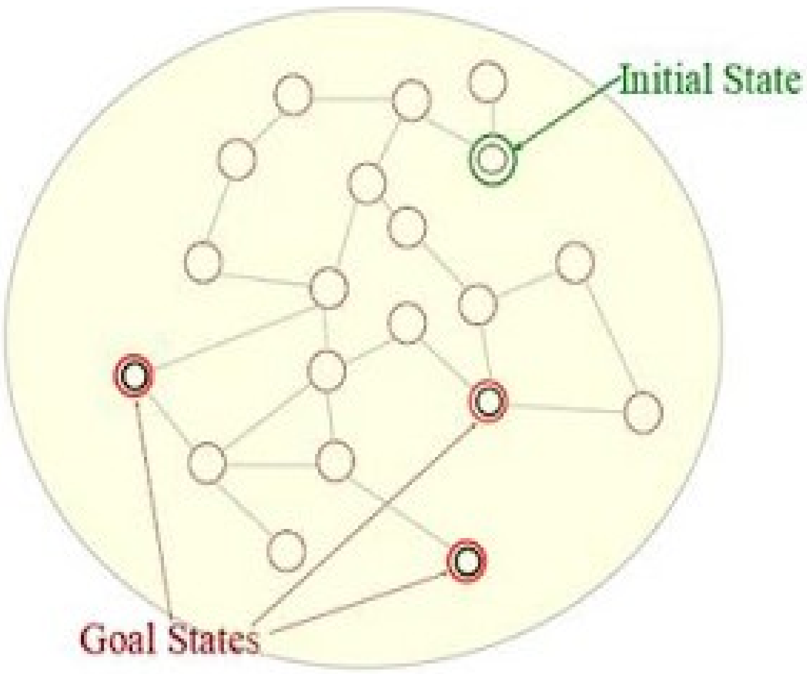
- Problem solving agents
- Search Algorithms

Textbook: Chapter 3.1, 3.2, 3.3

General Problem Solving Strategy

- How does a human solve a problem in general?
  - Do we use thousands of algorithms to solve different problems, or
  - Use only a few general methods to solve all types of problems?
- Is there a general purpose approach to solve all types of problems?
  - Driving a car, Playing chess, Finding the cheapest car, Buying a ticket, …
- State space search as a general problem solving strategy

How do humans solve problems?



- **Think about these problems:**
  - Playing games like chess or tic-tac-toe
  - Navigate a maze
  - Driving a car
- What do we do to solve a problem?
  - **Understand** the problem
    - solution/goal, constraints, states
  - **Define** a **state** for each step and find a **sequence of states (or steps)**.
    - A state can be a problem-solving step or status (information and available methods),
    - Use available information and methods to move from one state to next state.

State Space Search

- **State**, **State Space**, and **Search**:
  - A state is a representation for a problem solving step that involves **available information** and **methods**.
    - A **state** captures only the features of a problem **essential** to solve it
  - The state space of a problem: set of all possible states.
  - A search is an algorithm for exploring the state space.
- State space search as a **general problem solving strategy** is based on a strategy used by humans to solve difficult problems or almost all problems if resources and time are unlimited!
  - AI was considered as a problem of state representation and search in early AI research.

State Representation

- *Expressiveness* and *efficiency* are the key factors.
  - Need to optimize the trade-off between **expressiveness** and **efficiency**
  - Ultimately we need a *powerful representation scheme* to solve AI problems.

- Different levels of state representation:
  - **Conceptual** (or mental) representation,
    - State
  - ***Symbolic*** *representation*,
    - *Graph*
  - **Computer** representation (data structure)
    - Variable, array, record, object, table, list, tree, queue, …

Definition of a graph

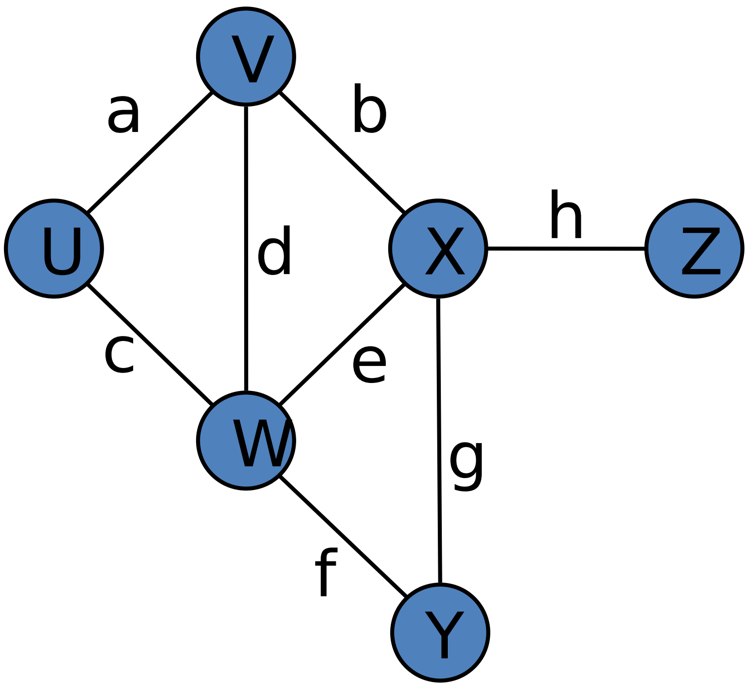
- A graph consists of
- A set of *nodes/vertexes/vertices*
  - can be infinite
- A set of *arcs/edges* that connect pairs of nodes
- An *arc/edge* is an ordered pair, e.g.,

  (a1, b1)

  (b1, a1)

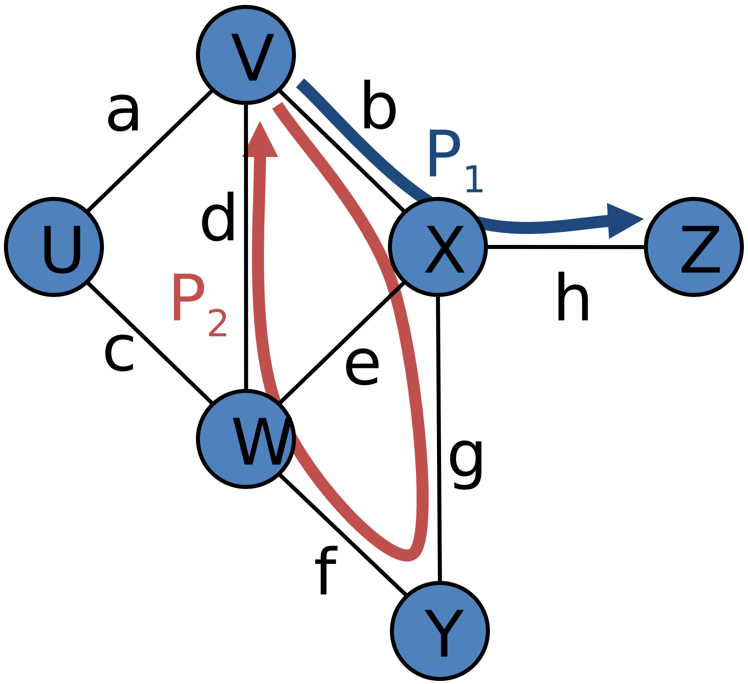
Terminology

- End vertices (or endpoints) of an edge
  - Endpoints of a?
  - U and V
- Edges incident on a vertex
  - Incident on V?
  - a, d, and b
- Adjacent vertices
  - U and V?
    - Yes
  - U and X?
    - No
- Degree of a vertex
  - Degree of X?
    - X has degree 4

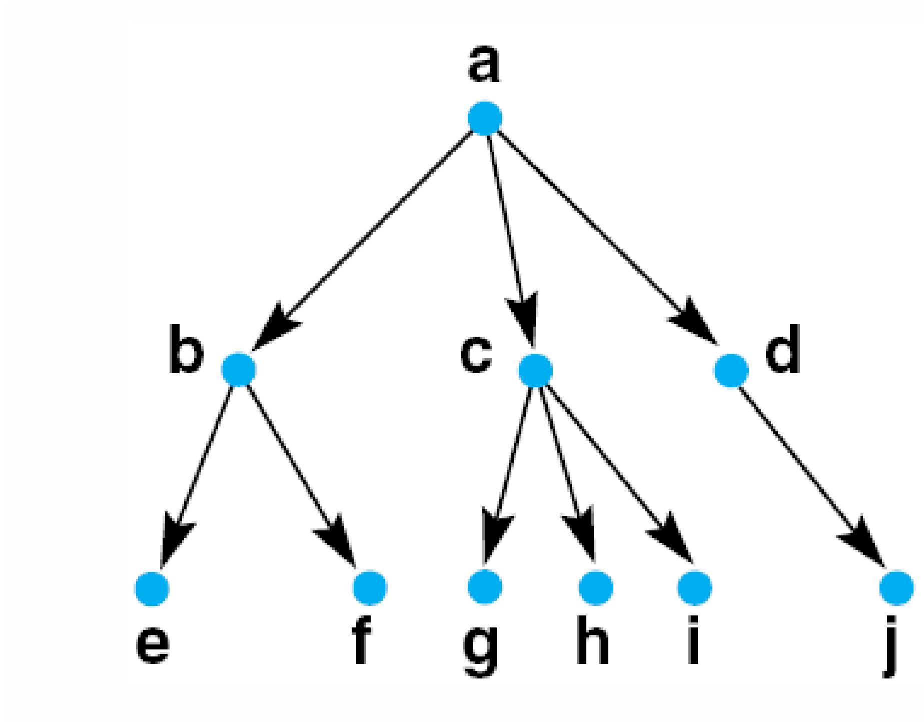


Terminology (cont.)

- Path
  - sequence of edges
  - begins with a vertex
  - ends with a vertex
- A *path of length n* has n edges
  - [U V X] is a path of length ?
- Cycle
  - path that begins and ends with the same vertex
- Examples
  - P_1=(V,b,h,Z) is a path
  - P_2=(V,b,g,f, d,V) is a cycle



A Tree is a Rooted Graph



A **tree** showing family relationships, *parent* and *children*

***Tree**: has a root that has path from the root to all nodes and every path is unique without cycle.

Problem space

- a *state space:* a set of states representing the possible configurations of the world
- a set of *operators/actions:* change one state into another
- The problem space is a graph where the states are the nodes and the edges represent the operators.

Search problems and solutions

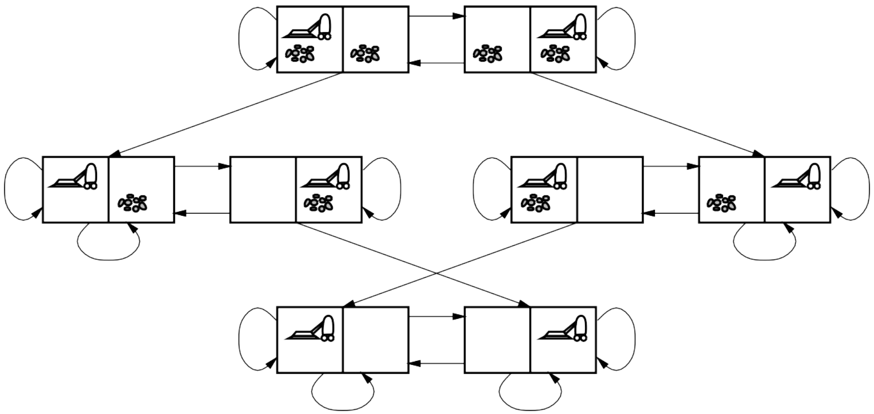
- State space: a set of possible states that the environment can be in
- Initial state: state agent starts in
- Goal state(s): target state(s) agent aims to reach
- Actions: a finite set of actions that can be executed in state *s*

  ACTIONS(s)
- Transition model: describes what each action does

  RESULT(*s*, *a*)
- Action cost function: the numeric cost of applying action a in state s to reach state s'

  ACTION-COST(*s*, *a*, *s'*)

Vacuum example

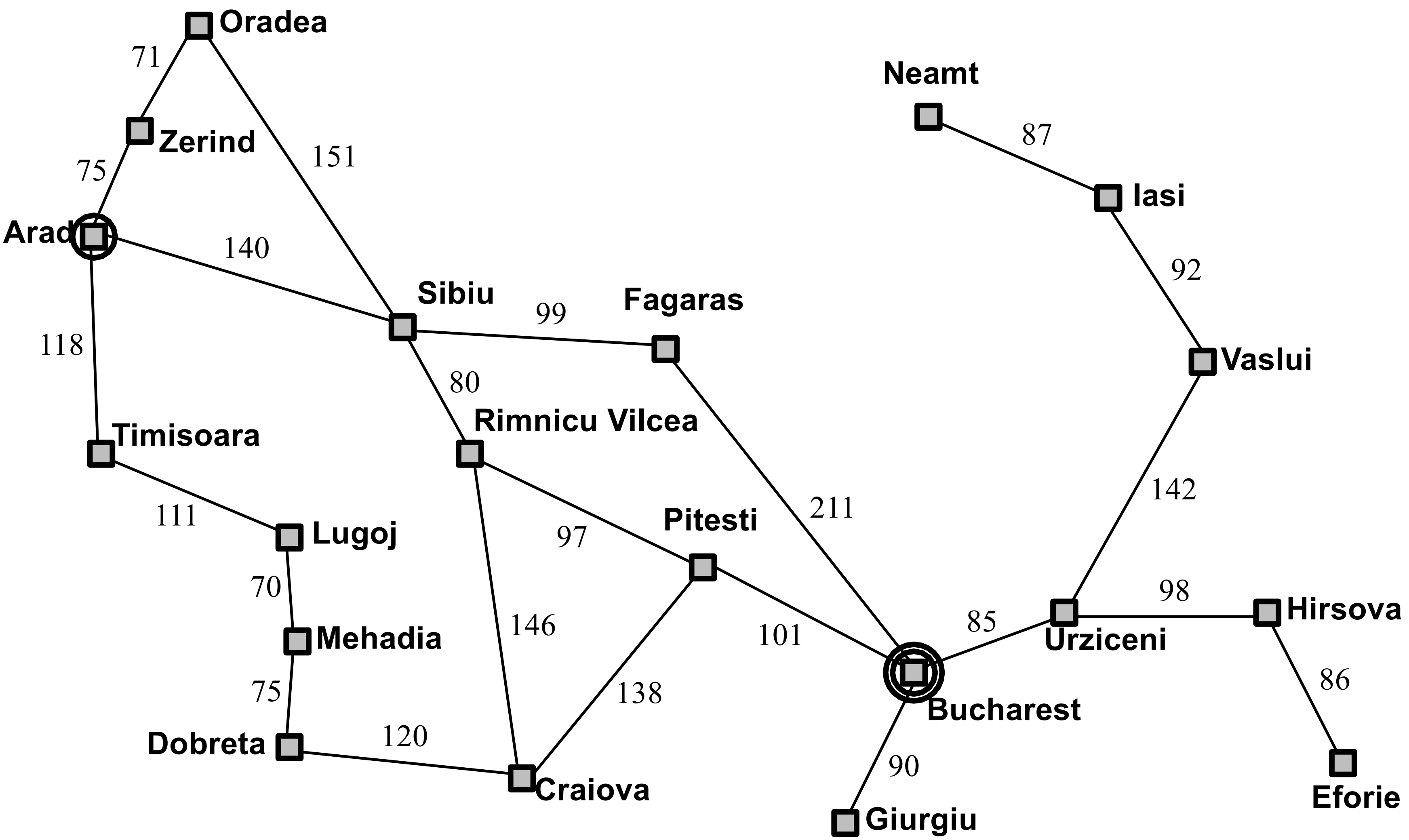


- State space: Objects are the agent and any dirt
- Initial state: Any state can be designated as the initial state
- Goal state(s): The states in which every cell is clean
- Actions: Suck, move *Left* and move *Right*
- Transition model: *Suck* removes any dirt from the cell; *TurnRight* and *Turnleft* change the direction it is facing by 90 degrees
- Action cost function: Each action costs 1

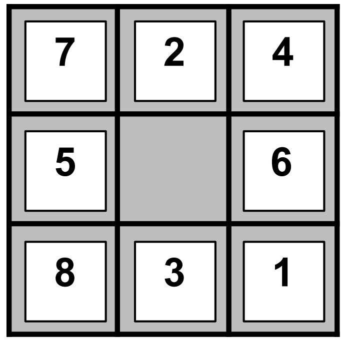
Example) Romania

- On a vacation in Romania; currently in Arad.
- Flight leaves tomorrow from Bucharest
- Formulate goal:
  - be in Bucharest
- Formulate problem:
  - states: various cities
  - actions: drive between cities
- Find solution:
  - sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest

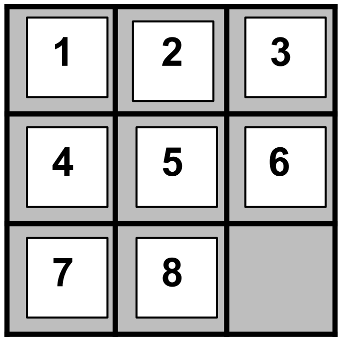
Example) Romania



Example) Sliding-tile puzzle



Start State



Goal State

8-puzzle that consists a 3 x 3 grid with eight numbered tiles and one blank space

- States: A state description specifies the location of each of the tiles
- Initial state: any state can be designed as initial
- Actions: Left, Right, Up or Down
- Transition Model: maps a state and action to a resulting state
- Goal state: a state with the numbers in order but can be anything
- Action cost: each action of moving a tile costs 1

In-class exercise

- Define the following for the 2 scenarios' 'airline travel problem' and 'robot navigation system'
  - States
  - Initial state
  - Actions
  - Transition model
  - Goal state
  - Action cost

Example) Airline travel problem

- States: each state includes a location and the current time as well as ”historical” info
- Initial state:
- Actions:
- Goal state:
- Transition model:
- Action cost:

Example) Airline travel problem

- States: each state includes a location and the current time as well as ”historical” info
- Initial state: departing airport
- Actions: take any flight from the current location, in any seat class, leaving after the current time, leaving enough time for within-airport transfers, if needed
- Transition model: the state resulting from taking a flight will have the flight’s destination as the new location and the flight’s arrival time as the new time
- Goal state: destination airport
- Action cost: a combination of monetary cost, waiting time, flight time, customs procedures, seat class, type of plane, etc.

Example) Robot navigation system

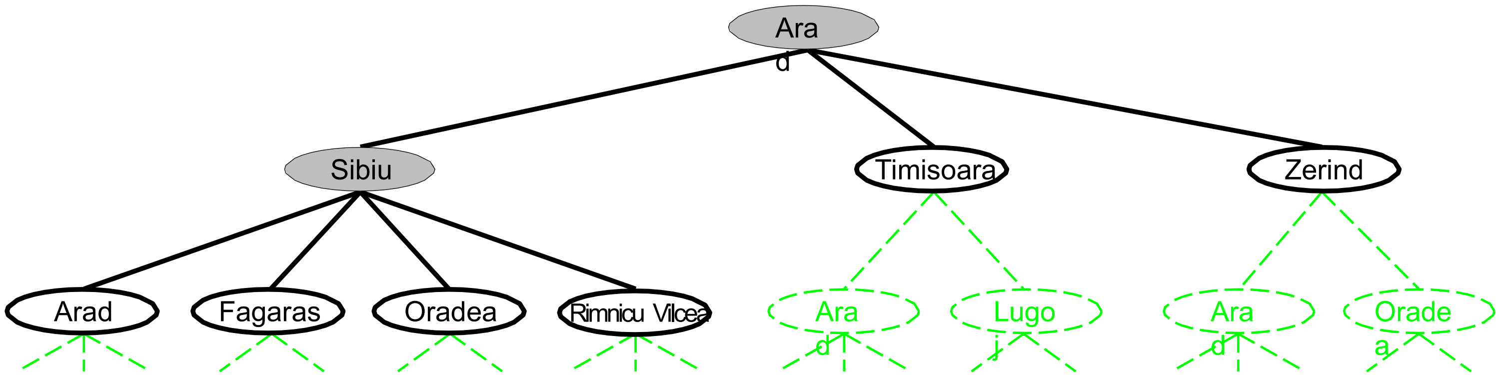
- States: Each state typically includes the robot's location, orientation, and other relevant attributes.
- Initial State:
- Actions:
- Transition Model:
- Goal State:
- Action Cost:

Example) Robot navigation system

- States: Each state typically includes the robot's location, orientation, and other relevant attributes.
- Initial State: The initial state is the starting configuration or position of the robot when the navigation task begins.
- Actions: Moving forward, turning left or right, stopping, or any other relevant movement
- Transition Model: Specifies how the robot's position and orientation change based on the chosen action.
- Goal State: The goal state is the desired configuration or position that the robot needs to end up.
- Action Cost: Quantifies the resources, such as time, energy, or distance, required to perform a particular action.

Search algorithms

- Takes a search problem as input and returns a solution



- Root node of the search tree is at the initial state
- Follow up on one option and put others aside
- Newly generated nodes added to the frontier which acts as a queue to determine the order in which nodes are expanded
- If we expand Sibiu, what nodes are in the frontier?

Operations on a frontier

- Data structure needed to store frontier, often times queue or priority queue to apply following operations
- IS-EMPTY(*frontier*)
  - Returns true only if there are no nodes in the frontier
- POP(*frontier*)
  - removes the top node from the frontier and returns it
- TOP(*frontier*)
  - returns (but does not remove) the top node of the frontier
- ADD(*node*, *frontier*)
  - inserts node into its proper place in the queue

Queues

- Three types of queues
  - Priority queue: node popped based on their priority
    - Example) Tasks with highest priorities popped before lower priorities
  - FIFO queue: First-In, First-Out. Node that is inserted first will be popped first. Preserves the order of insertion
    - Example) People waiting in line at Starbucks
  - LIFO queue: Last-In, First-Out. Node that is inserted last will be popped first. Reverses the order of insertion.
    - Example) Stack of plates at revolving sushi bar

Search Strategies

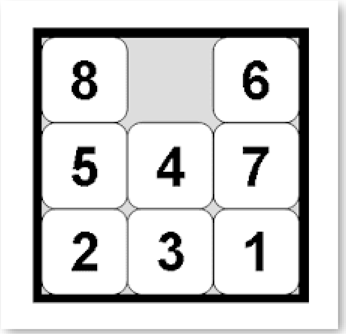
- A strategy is defined by picking the order of node expansion
- Strategies are evaluated along the following dimensions:
  - Completeness: does it always find a solution if one exists?
  - Optimality: does it always find a least-cost solution?
  - time complexity: number of nodes generated/expanded
  - space complexity: maximum number of nodes in memory
- Time and space complexity are measured in terms of
  - *b*: maximum branching factor of the search tree
  - *d*: depth of the least-cost solution
  - *m*: maximum depth of the state space (may be ∞)

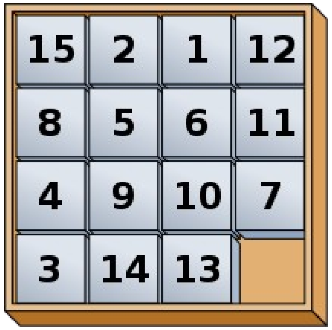
Branching factor

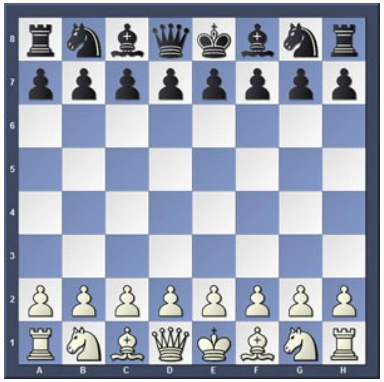
- Number of next states from a given state
- Number of children of a node
- Can vary at different positions/levels of tree or graph
- How many moves can you do in a given position?

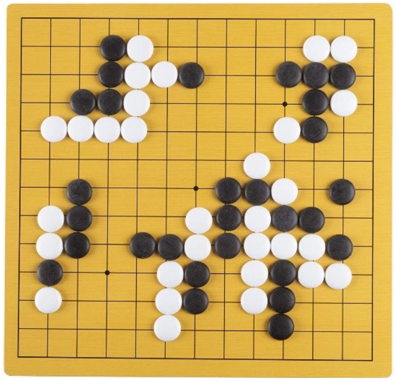
Branching factor

- Examples:
  - 8-sliding tile puzzle?
    - 2-4, average=2.67
  - 15-sliding tile puzzle?
    - 2-4
  - Chess?
    - 35 on average
  - Go?
    - 250 on average

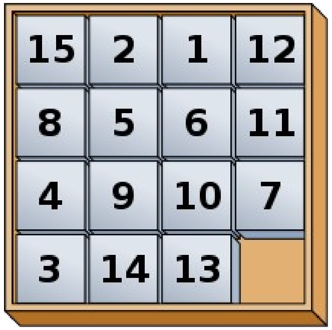




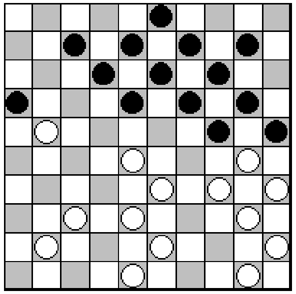


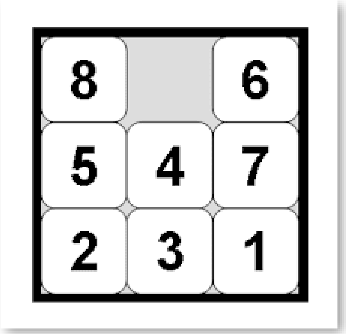


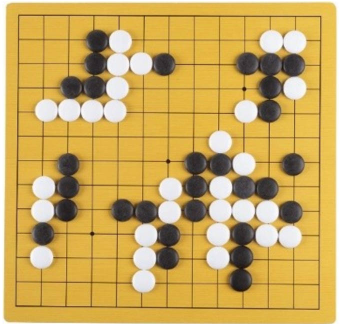
State space complexity

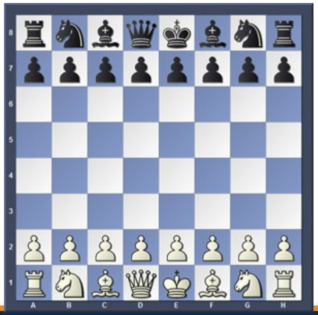


- State space: set of all possible states of a problem
- State space complexity: number of states in the state space
- Examples:
  - 8-sliding tile puzzle?
  - 15-sliding tile puzzle?
  - Checkers?
  - Chess?
  - Go?





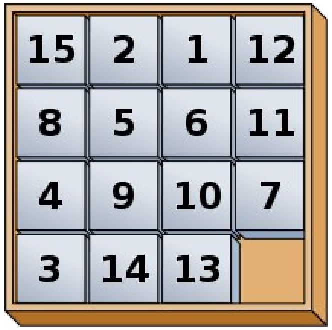


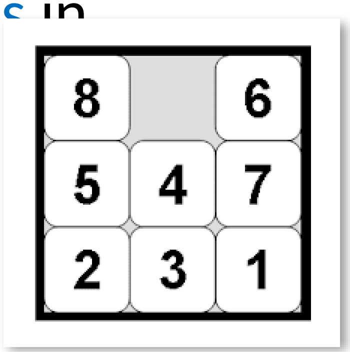


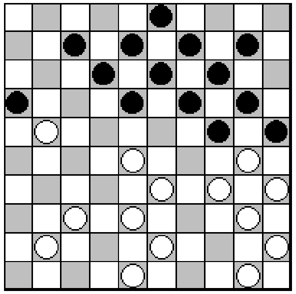
https://en.wikipedia.org/wiki/Game_complexity

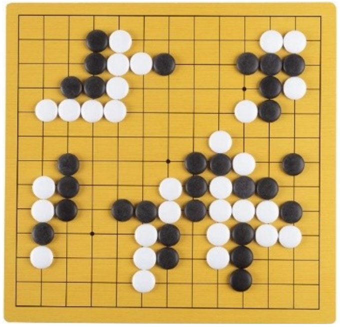
State space complexity

- State space complexity: number of states in the state space
- Examples:
  - 8-sliding tile puzzle?
  - 15-sliding tile puzzle?
  - Checkers?
  - Chess?
  - Go?
- if branching factor is 10, then
  - 10 nodes one level down from the current position
  - 10^2 (or 100) nodes two levels down
  - 10^3 (or 1,000) nodes three levels down, …
  - State space explosion









Searching a graph: the challenge

- Number of nodes is practically infinite
  - Cannot pre-compute all nodes and store it in a computer/data structure
  - “Search”, not a “traversal”
- Nodes can reappear in the search
  - Possible to get into loops

References

- Russel and Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Prentice Hall, 2010.