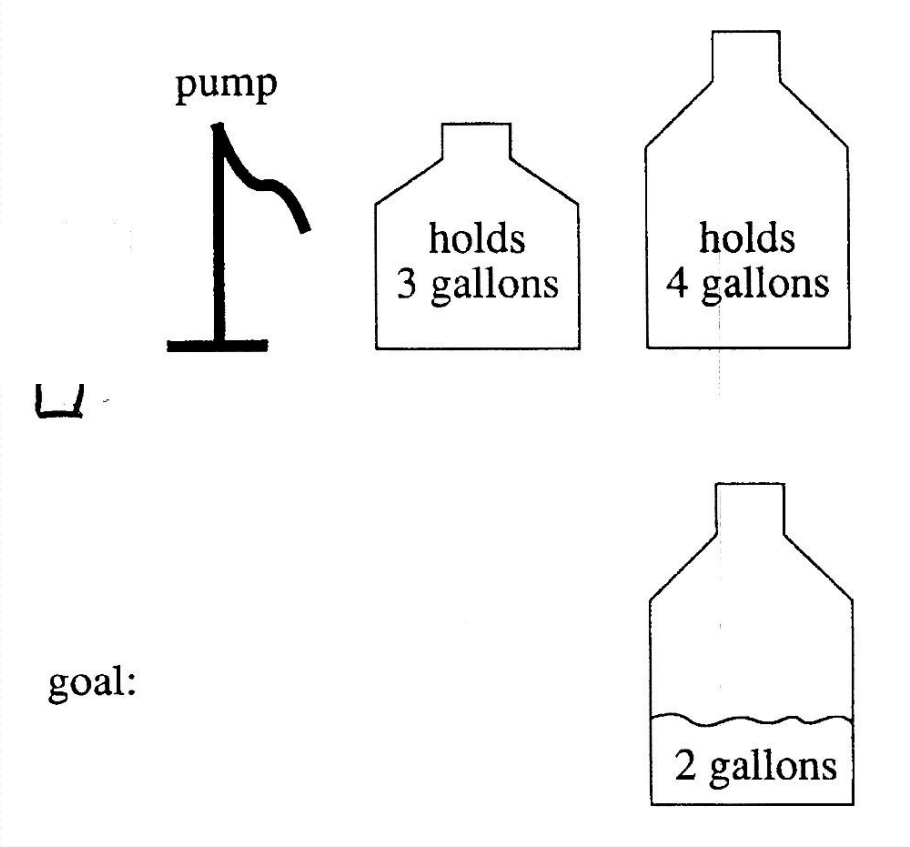
State Space Representation using Variables

Give the initial state, goal test, successor and cost function for Water Jug Problem.

A Water Jug Problem

- You have a 4-gallon and a 3-gallon water jug
- You have a faucet with an unlimited amount of water
- You need to get exactly 2 gallons in 4-gallon jug



Puzzle-solving as Search

- **State representation: (x, y)**
  - **x:** Contents of four gallon
  - **y:** Contents of three gallon
- **Start state: (0, 0)**
- **Goal state (2, n)**
- **Operators**
  - Fill 3-gallon from pump, fill 4-gallon from pump
  - Fill 3-gallon from 4-gallon , fill 4-gallon from 3-gallon
  - Empty 3-gallon into 4-gallon, empty 4-gallon into 3-gallon
  - Dump 3-gallon down drain, dump 4-gallon down drain

The Water Jug Problem

8 $(x,y) \rightarrow (x - (3 - y), 3)$
if $x + y \geq 3$ **and** $x > 0$

Pour water from the 4-gallon jug into the 3-gallon jug **until** the **3-gallon jug is full**

9 $(x,y) \rightarrow (x + y, 0)$
if $x + y \leq 4$ **and** $y > 0$

Pour all the water from the 3-gallon jug into the 4-gallon jug

10 $(x,y) \rightarrow (0, x + y)$
if $x + y \leq 3$ **and** x

Pour all the water from the 4-gallon jug into the 3-gallon jug

One Solution to the Water Jug Problem

1 $(x,y) \rightarrow (4,y)$ if $x < 4$	**Fill** the 4-gallon jug
2 $(x,y) \rightarrow (x,3)$ if $y < 3$	**Fill** the 3-gallon jug
3 $(x,y) \rightarrow (x - d,y)$ if $x > 0$	**Pour** some water out of the 4-gallon jug
4 $(x,y) \rightarrow (x,y - d)$ if $x > 0$	**Pour** some water out of the 3-gallon jug
5 $(x,y) \rightarrow (0,y)$ if $x > 0$	**Empty** the 4-gallon jug on the ground
6 $(x,y) \rightarrow (x,0)$ if $y > 0$	**Empty** the 3-gallon jug on the ground
7 $(x,y) \rightarrow (4,y - (4 - x))$ if $x + y \geq 4$ and $y > 0$	**Pour** water from the 3-gallon jug into the 4-gallon jug until the 4-gallon jug is full
8 $(x,y) \rightarrow (x - (3 - y),3)$ if $x + y \geq 3$ and $x > 0$	**Pour** water from the 4-gallon jug into the 3-gallon jug **until** the **3-gallon jug is full**
9 $(x,y) \rightarrow (x + y, 0)$ if $x + y \leq 4$ and $y > 0$	**Pour** all the water from the 3-gallon jug into the 4-gallon jug
10 $(x,y) \rightarrow (0, x + y)$ if $x + y \leq 3$ and $x > 0$	**Pour** all the water from the 4-gallon jug into the 3-gallon jug

Gallons in the 4-Gallon Jug	Gallons in the 3-Gallon Jug	Rule Applied
0	0	2
0	**3**	9
3	0	2
3	**3**	7
4	2	5
0	2	9
2	0	

2. Fill t

9. Pour all the water from the 3-gallon jug into the 4-gallon jug

2 . Fill the 3-gallon jug

7. Pour water from the 3-gallon jug into the 4-gallon jug until the 4-gallon jug is full

5. Empty the 4-gallon jug on the ground

9. Pour all the water from the 3-gallon jug into the 4-gallon jug

Production Rules for the Water Jug Problem

1 $(x,y) \rightarrow (4,y)$ **if** $x < 4$	**Fill** the 4-gallon jug
2 $(x,y) \rightarrow (x,3)$ **if** $y < 3$	**Fill** the 3-gallon jug
3 $(x,y) \rightarrow (x - d,y)$ **if** $x > 0$	**Pour** some water out of the 4-gallon jug
4 $(x,y) \rightarrow (x,y - d)$ **if** $x > 0$	**Pour** some water out of the 3-gallon jug
5 $(x,y) \rightarrow (0,y)$ **if** $x > 0$	**Empty** the 4-gallon jug on the ground
6 $(x,y) \rightarrow (x,0)$ **if** $y > 0$	**Empty** the 3-gallon jug on the ground
7 $(x,y) \rightarrow (4,y - (4 - x))$ **if** $x + y \geq 4$ **and** $y > 0$	**Pour** water from the 3-gallon jug into the 4-gallon jug until the 4-gallon jug is full

Exercise 2

Water Jug Problem II

- Given a full **5-gallon jug** and an **empty 2-gallon jug**, the goal is to **fill the 2-gallon jug with exactly one gallon of water**. You may use the following state space formulation.

- State = (x,y), where x is the number of gallons of water in the 5-gallon jug and y is # of gallons in the 2- gallon jug

- **Initial State** = **(5,0)**

- **Goal State** = **(*,1)**, where * means any amount

Water Jug Problem II

The table below shows the different operators and their effects.

Name	Cond.	Transition	Effect
Empty5	–	(x,y)→(0,y)	Empty 5-gal. jug
Empty2	–	(x,y)→(x,0)	Empty 2-gal. jug
2to5	$x \leq 3$	(x,2)→(x+2,0)	Pour 2-gal. into 5-gal.
5to2	$x \geq 2$	(x,0)→(x-2,2)	Pour 5-gal. into 2-gal.
5to2part	$y < 2$	(1,y)→(0,y+1)	Pour partial 5-gal. into 2-gal.

The basic search algorithm

Initialize: put the start node into OPEN

while OPEN is not empty

take a node N from OPEN
if N is a goal node, report success

put the children of N onto OPEN

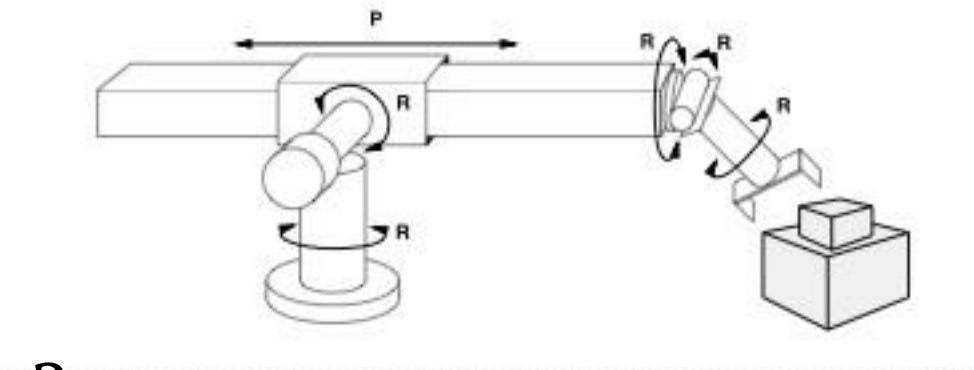
Report failure

- If OPEN is a stack, this is a depth-first search
- If OPEN is a queue, this is a breadth-first search
- If OPEN is a *priority queue*, sorted according to *most promising first*, we have a best-first search

Tree-based Search

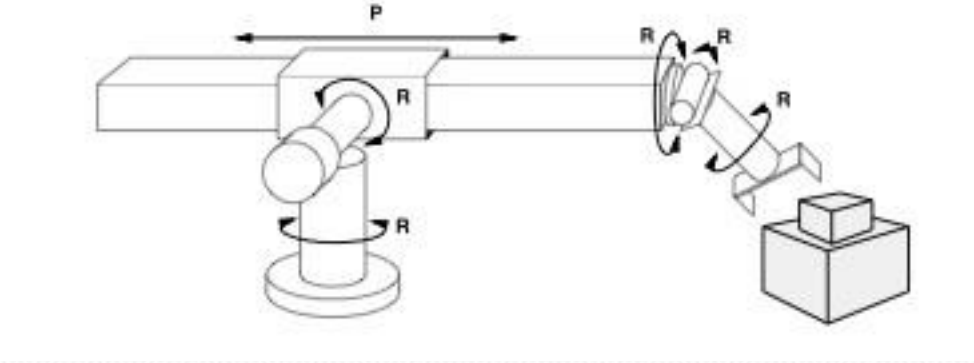
- Basic idea:
  - Exploration of state space by generating successors of already-explored states (a.k.a. expanding states).
  - Every state is evaluated: *is it a goal state*?
- In practice, the solution space can be a graph, not a tree
  - E.g., 8-puzzle
  - More general approach is graph search
  - Tree search can end up repeatedly visiting the same nodes
    - Unless it keeps track of all nodes visited
    - …but this could take vast amounts of memory

Problem 3: Robot Assembly



- States:?
- Initial state:?
- Actions:?
- Goal test:?
- Path Cost:?

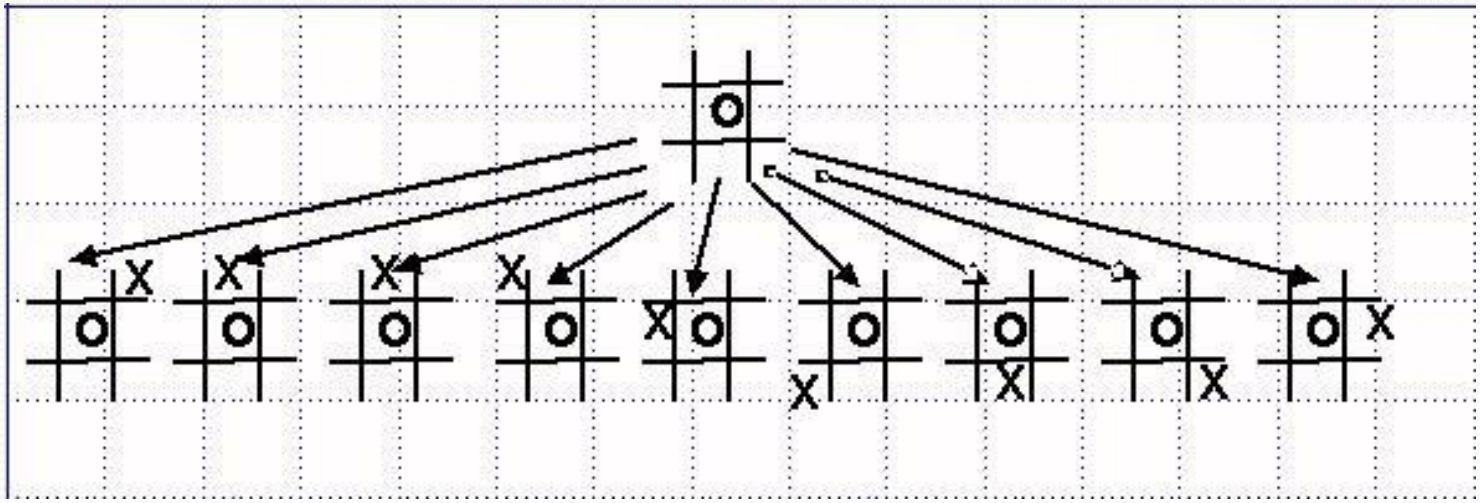
Example: Robot Assembly



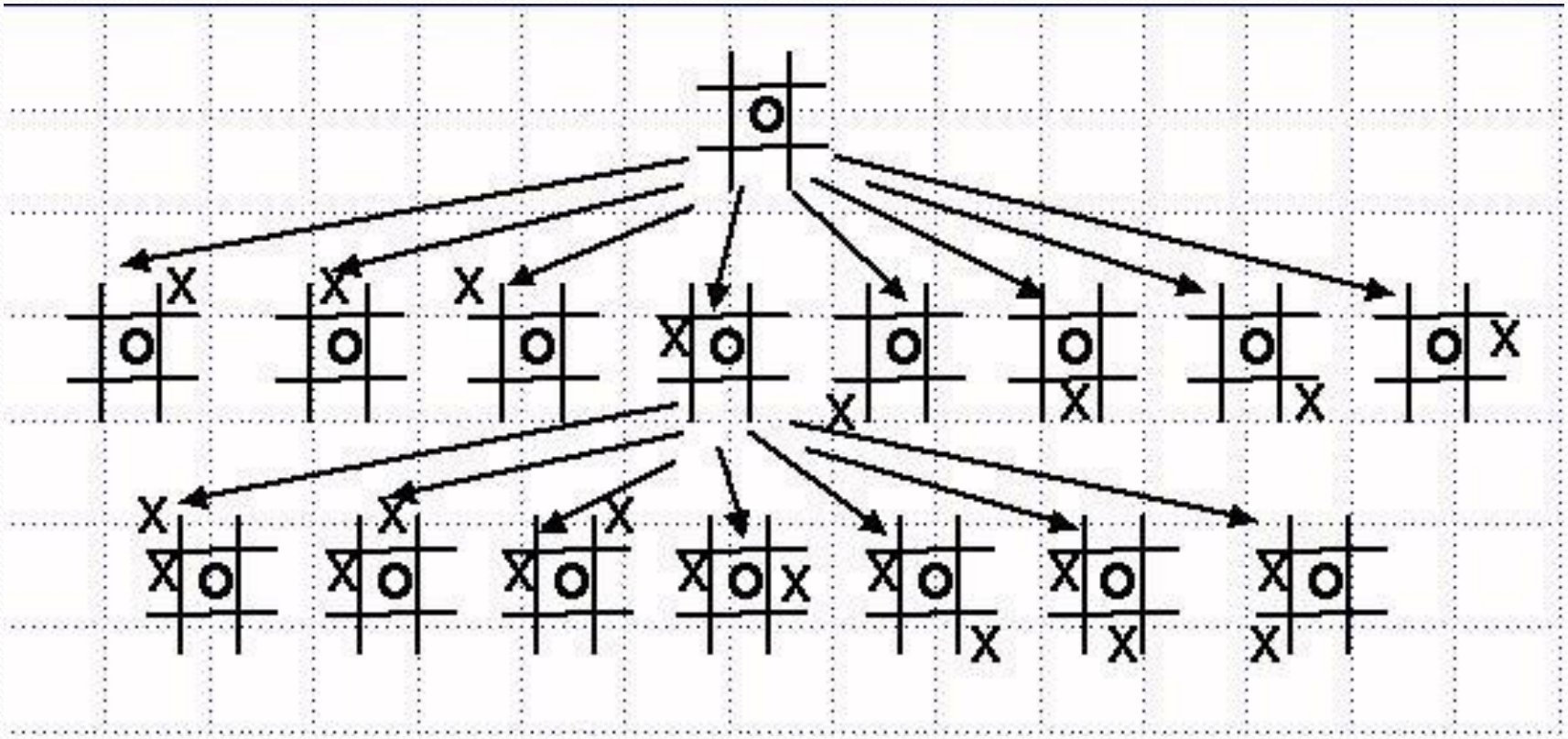
- States: configuration of robot (angles, positions) and object parts
- Initial state: any configuration of robot and object parts
- Actions: continuous motion of robot joints
- Goal test: object assembled?
- Path Cost: time-taken or number of actions

Problem 4

- Tic Tack Too Game – draw state space representation, initial and goal state, operators etc.



The entire set of possible states makes up the *State Space* for the game



Practice Questions

Give PEAS description for an Autonomous Mars Rover. Characterize its environment.
Give the initial state, goal test, successor function, and cost function for the following problem
"You have to colour a planar map using only 4 colours, in such a way that no two adjacent regions have the same colour".

PEAS Descriptor

Mathematician's theorem-proving assistant

P: good math knowledge, can prove theorems accurately and in minimal steps/time

E: Internet, library

A: display

S: keyboard

PEAS Descriptor

Autonomous Mars rover

P: Terrain explored and reported, samples gathered and analyzed

E: Launch vehicle, lander, Mars

A: Wheels/legs, sample collection device, analysis devices, radio transmitter

S: Camera, touch sensors, accelerometers, orientation sensors, wheel/joint encoders, radio receiver

PEAS Descriptor

Internet book-shopping agent

P: Obtain requested/interesting books, minimize expenditure

E: Internet

A: Follow link, enter/submit data in fields, display to user

S: Web pages, user requests

PEAS Descriptor

Robot soccer player

P: Winning game, goals for/against

E: Field, ball, own team, other team, own body

A: Devices (e.g., legs) for locomotion and kicking

S: Camera, touch sensors, accelerometers, orientation sensors, wheel/joint encoders

Define heuristic function. Give an example heuristics function for 8-puzzle problem. Find the heuristics value for a particular state of the Blocks World Problem.

What are PEAS descriptors? Give PEAS descriptors for a robot meant for cleaning the house.

Define heuristic function. Give an example heuristics function for 8-puzzle problem. Find the heuristics value for a particular state of the Blocks World Problem.

Consider the given instance of 8-puzzle. [10]

1	2	3
4	5	6
7	8	

Goal State

1	2	3
4	6	
7	5	8

Initial state

Compare and contrast uninformed search strategies with respect to solving 8-puzzle problem.

Graph Coloring

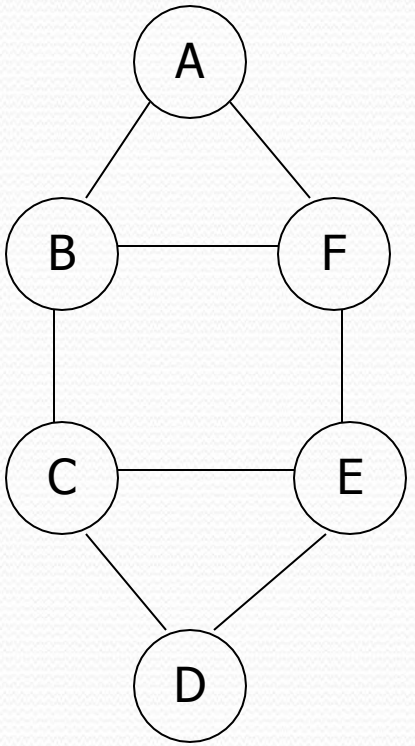
- Graph coloring is the problem of coloring each vertex in a graph such that no two adjacent vertices are the same color
- Some direct examples:
  - Map coloring

Graph Coloring

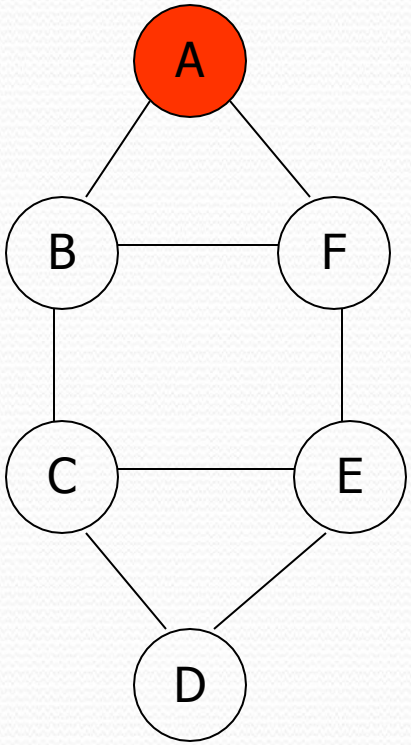
- We want to prune the state-space tree as soon as we find something that won't work
  - This implies that we need a sequence of vertices to color
  - As we color the next vertex we need to make sure it doesn't conflict with any of its previously colored neighbors
- We may need to backtrack

Graph Coloring

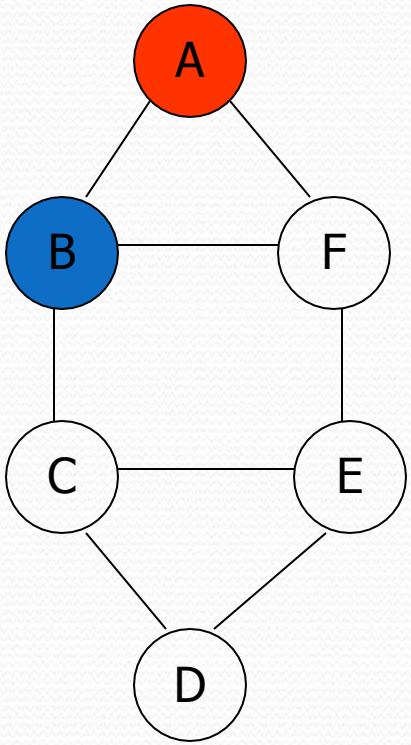
- As an example:
  - The vertices are enumerated in order A-F
  - The colors are given in order: R, G, B



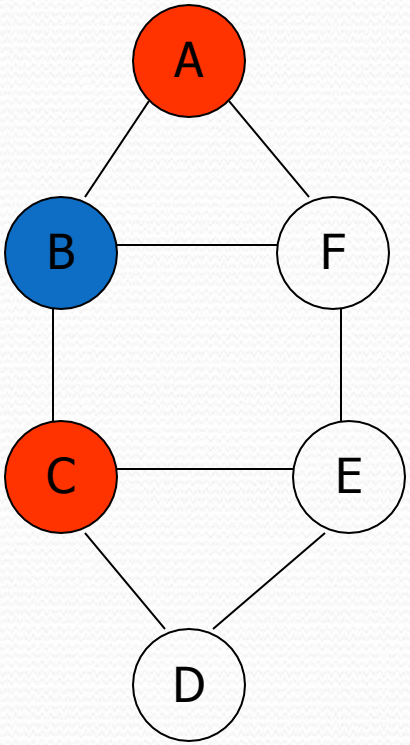
Graph Coloring



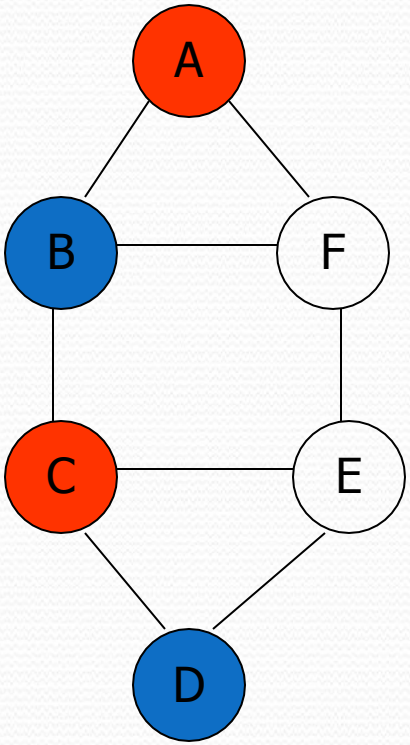
Graph Coloring



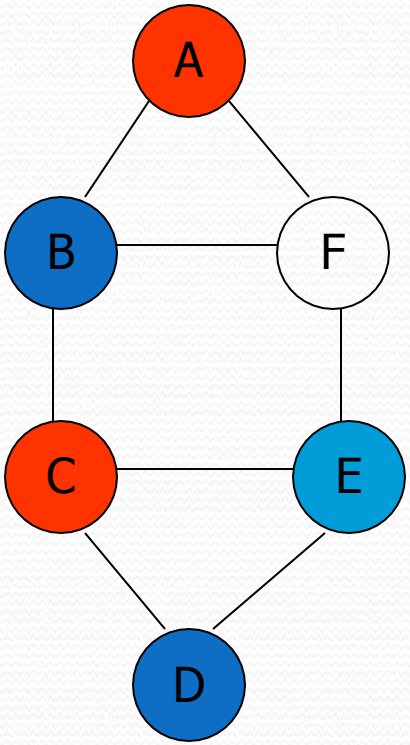
Graph Coloring



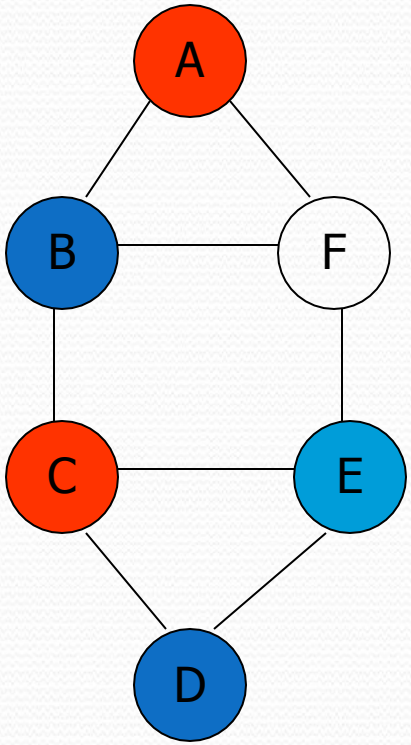
Graph Coloring



Graph Coloring

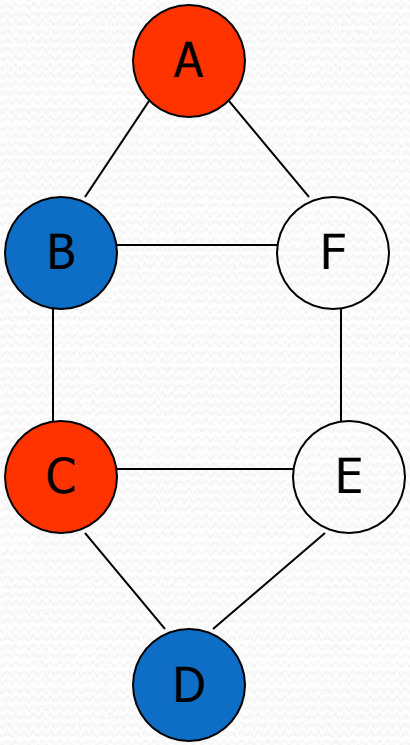


Graph Coloring

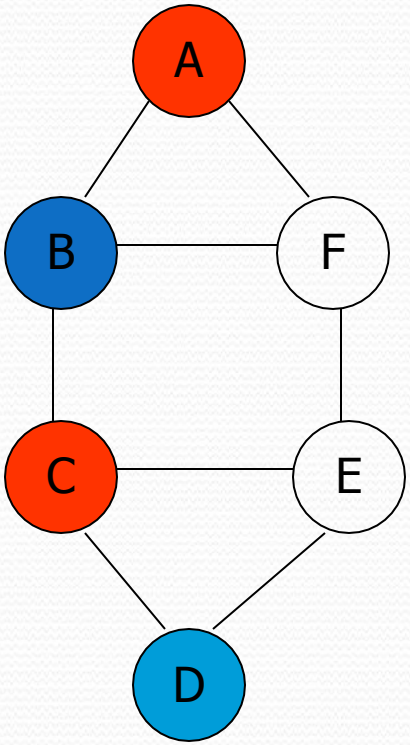


Stuck!

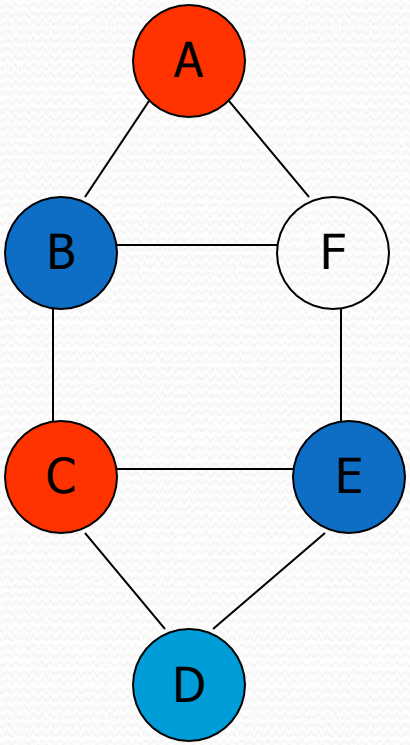
Graph Coloring



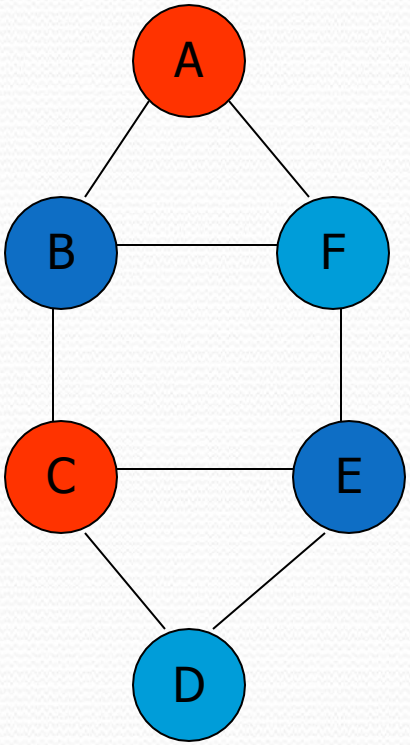
Graph Coloring



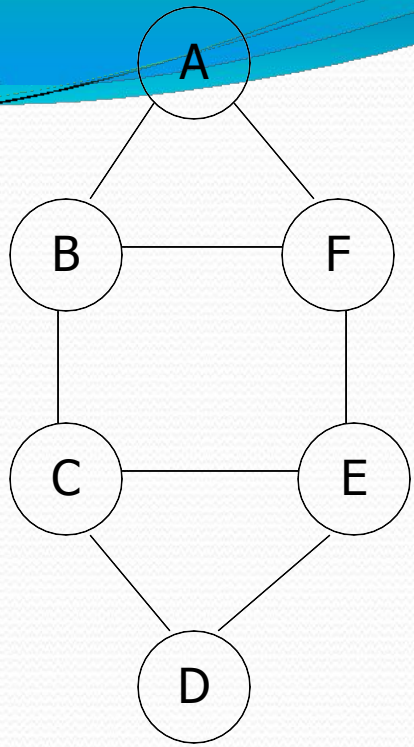
Graph Coloring



Graph Coloring



Graph Coloring



A

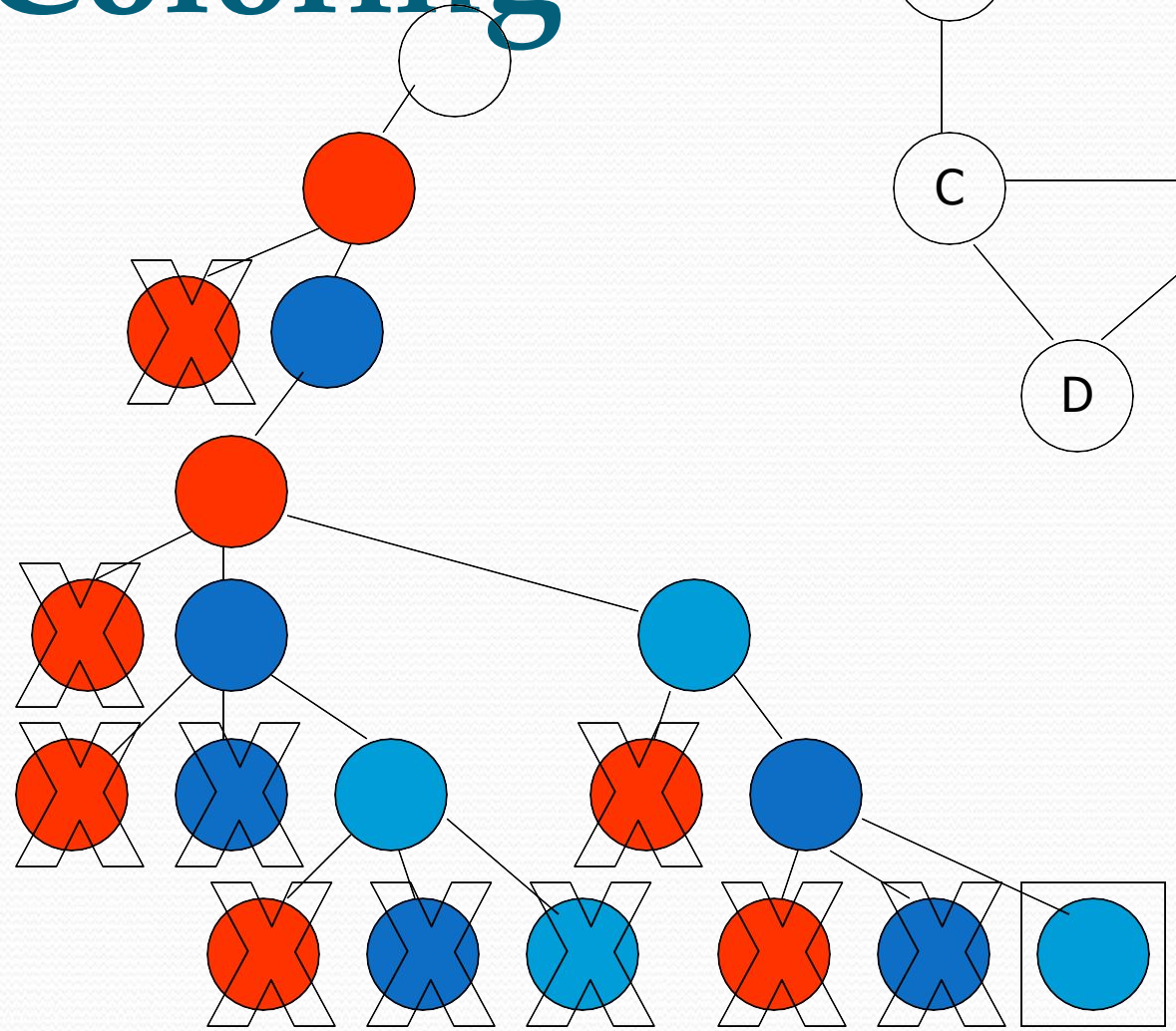
B

C

D

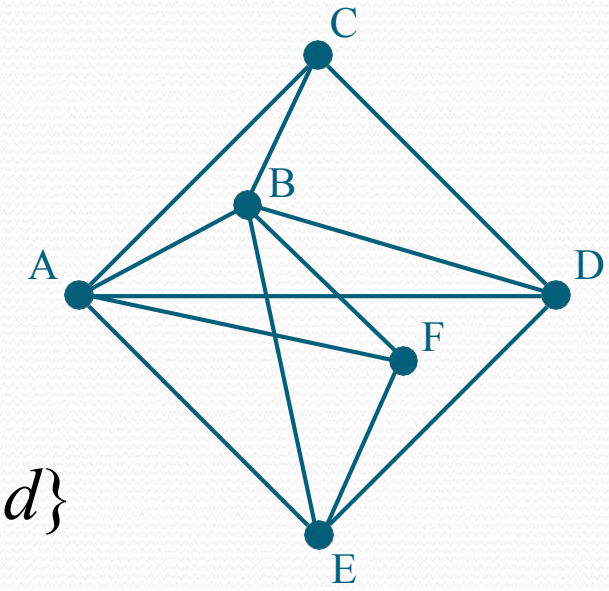
E

F



The Traveling Salesperson Problem (a touring problem)

- Find the shortest tour that visits all cities without visiting any city twice and return to starting point.
- States: sequence of cities visited



- S_o = A
- S_G = a complete tour

$\{a, c, d\}$ $\{(a, c, d, x) \mid X \notin a, c, d\}$

Exercises

- Three missionaries and three cannibals come to a river. There is a boat on their side of the river that can be used by either one or two persons. How should they use this boat to cross the river in such a way that cannibals never outnumber the missionaries on either side of the river.

(a) Specify the form of state description, the initial state and the goal state for this problem. Describe the state space using variables (as if you are using an array in a program). Determine how many states are in state space.

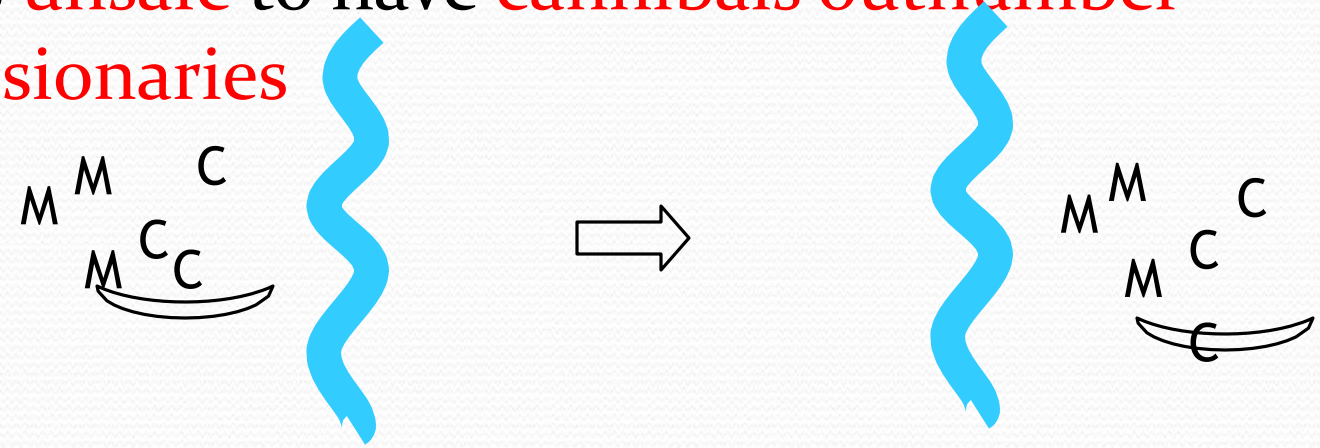
(b) Describe the set of operators using if-then rules.

(c) Draw the entire state space graph (include only legal states, that is, states in which cannibals do not outnumber missionaries on either side of the river)

Exercises

Example 3: Missionaries and cannibals

- An old puzzle is the "Missionaries and cannibals" problem
- The missionaries and cannibals wish to cross a river
- They have a boat that can hold two people
- It is unsafe to have cannibals outnumber missionaries



Initial state

Goal state

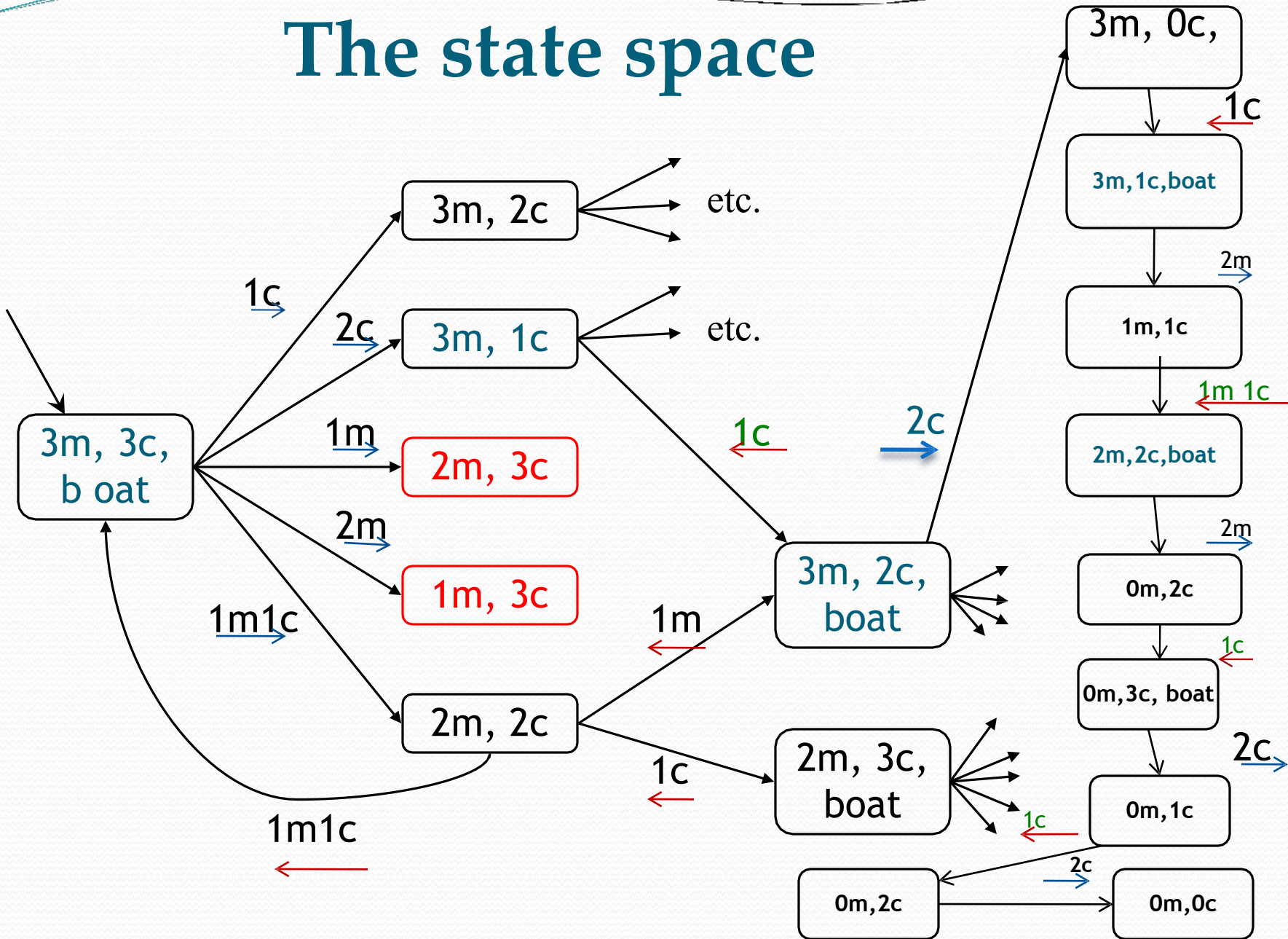
States

- A ***state*** **c**an be represented by the number of missionaries and cannibals on each side of the river
  - **Initial state:** **3m,3c,boat**
  - **Goal state: 0m,0c / 3m,3c,boat**
  - We assume that crossing the river is a simple procedure that always works (so we don't have to represent the boat being in the middle of the river)
- However, this is redundant; we only need to represent how many missionaries/cannibals are on *one* side of the river
  - **Initial state: 3m,3c,boat**
  - **Goal state: 0m,0c**

Operations

- An *operation* takes us from one state to another
- Here are five possible operations:
  - Boat takes 1 missionary across river (1m)
  - Boat takes 1 cannibal across river (1c)
  - Boat takes 2 missionaries across river (2m)
  - Boat takes 2 cannibals across river (2c)
  - Boat takes 1 missionary and 1 cannibal across river (1m1c)
- We don't have to specify "west to east" or "east to west" because only one of these will be possible at any given time

The state space



Example : Missionaries and cannibals

- 3 missionaries, 3 cannibals, 1 boat
- The boat can hold at most two people
- Cannibals may never outnumber missionaries (on either side)
- Initial state is (3, 3, 1), representing the number of missionaries, cannibals, boats on the *initial* side
- The goal state is (0, 0, 0)
- Operators are addition or subtraction of the vectors
  (1 0 1), (2 0 1), (0 1 1), (0 2 1), (1 1 1)
- Operators apply if result is between (0 0 0) and

Search Strategies

- A **search 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?
  - **Time complexity:** number of nodes generated
  - **space complexity:** maximum number of nodes in memory
  - **optimality:** does it always find a least-cost solution?

- 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 ∞)

State-space searching

- Most problems in AI can be cast as searches on a state space
- The space can be tree-shaped or graph-shaped
  - If a graph, need some way to keep track of where you have been, so as to avoid loops
- The state space is often very, very large
- We can minimize the size of the search space by careful choice of operators
- Exhaustive searches don't work—we need *heuristics*