COMP 3200
Artificial Intelligence

Lecture 3

Problem Solving and Search

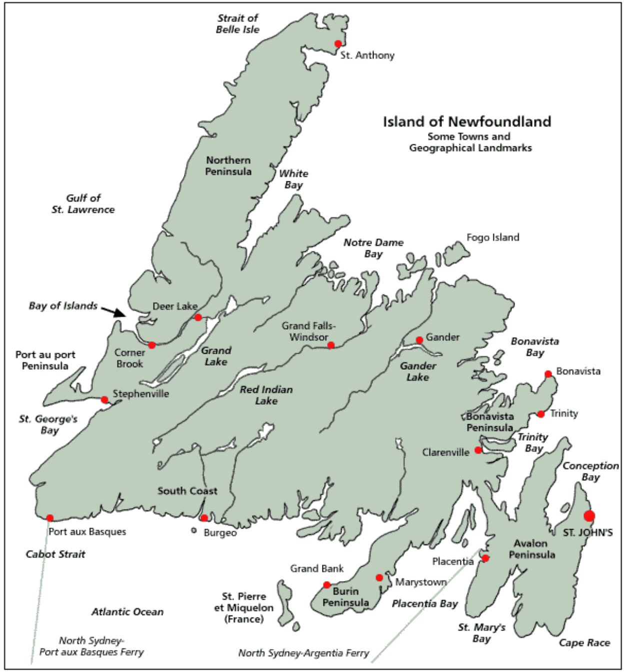
MEMORIAL UNIVERSITY

Problem Solving Agents

- Rational agents maximize performance
- Simplest Case: Satisfy a goal condition
  - Reach the end of a maze
  - Solve a puzzle
- Problem Solving Agent
  - Type of Goal-Based Agent
  - Find a sequence of actions that satisfies goal

Example Scenario

- Tourist comes to NL
  - Lands in St. John's
- Wants to go to one of:
  - St. Anthony
  - Port aux Basques
- Extra constraints?
  - Has X hours total
  - Has $$ total for gas



Goal Formulation

- Goal is a set of states in the environment
  - [St. Anthony, Port aux Basques]
- Goal is satisfied if one of them is reached
  - If a sequence of actions leads us to a goal state, the goal has been satisfied, and the problem is solved
- Goal can also be a function evaluation
  - If (enemy health <= 0)
  - If (enemy in checkmate)
  - If (player has all pieces of triforce AND Ganon dead)

Problem Formulation

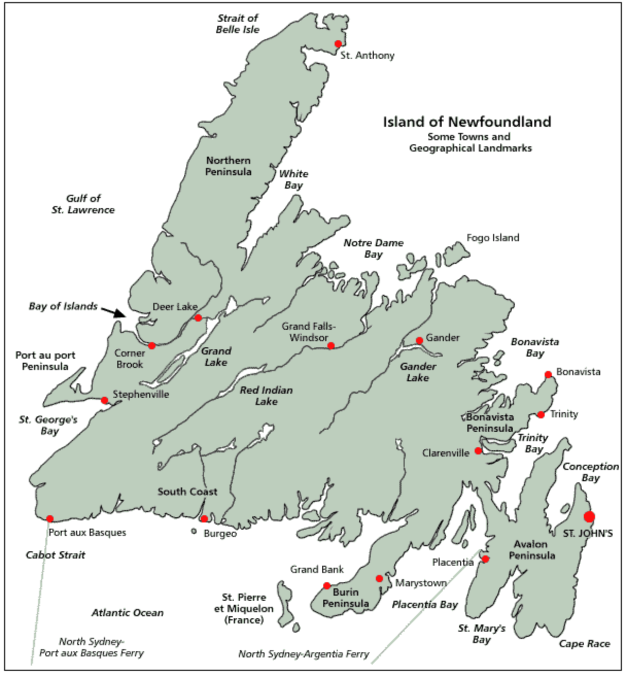
- What states to consider?
  - Every square meter in Newfoundland
  - Towns in Newfoundland
- What actions to consider?
  - Move left by a meter
  - Move from town X to town Y
- What information is stored in a node?
  - Current town location
  - Distance / time driven so far

Well-Defined Problem

1. Initial State that the agent starts in
2. Actions possible from each State
   - Transition / Successor Function
3. The set of goal states
   - Goal test function
4. Action cost function
   - Path cost = sum(cost(actions in path))

Example Problem Definition

1. Initial State
   - St. John’s
2. Actions
   - Towns connected by roads
3. Goal
   - [SA, PAB]
4. Action Cost Function
   - Travel distance or time

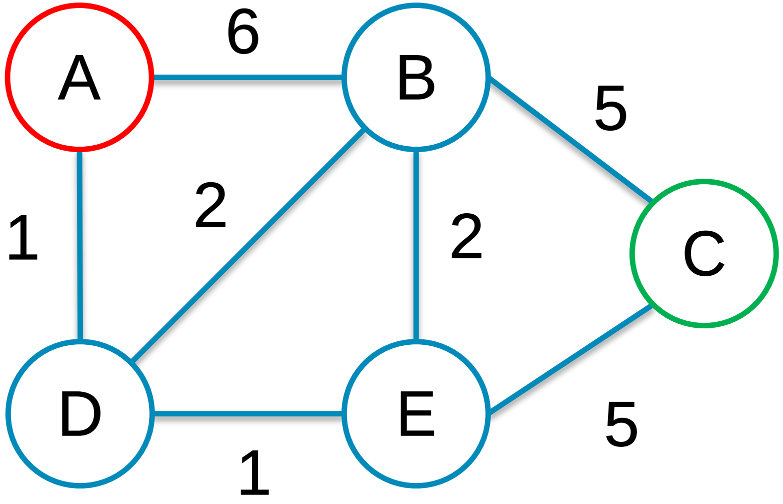


Problem Solution

- Path from start state to a goal state
  - Path = ordered sequence of actions
- Solution cost = path cost
  - Sum cost of actions (usually)
  - Most problems assume costs > 0
- Optimal solution has lowest path cost among all possible solutions

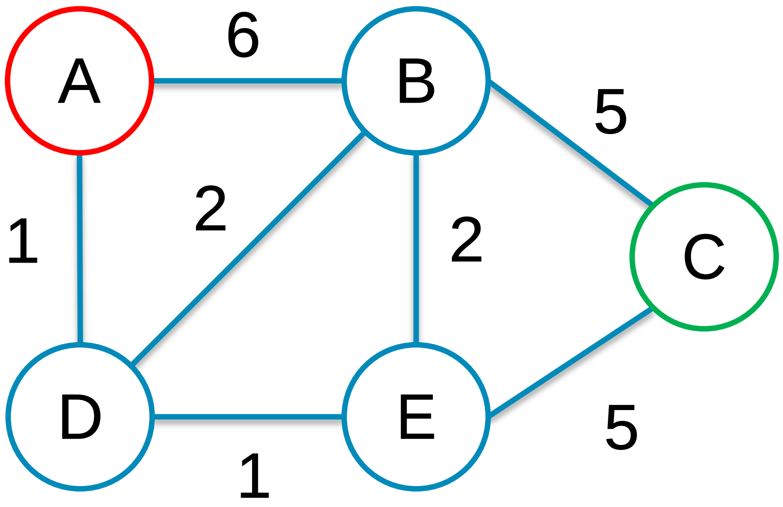
Example Graph Problem

- Initial State
  - State A
- Actions
  - (A,B) legal if edge
- Goal State
  - State C
- Action Cost Function
  - Edge label (A,B)=6



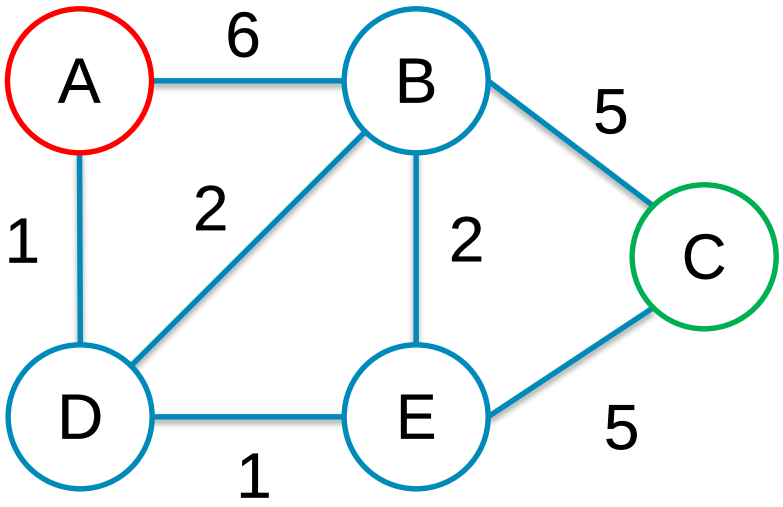
Example Graph Problem

- Fully Observable
- Static
- Discrete
- Deterministic
- Single Agent
- Sequential



Example Graph Problem

- Objective
  - Path to goal possible?
  - Shortest path to goal?
- Algorithm
  - Which to use?
- Let's try search



What is Search?

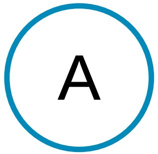
- Agent has several available actions
- Agent can explore various sequences of those actions
- Agent chooses the best sequence found

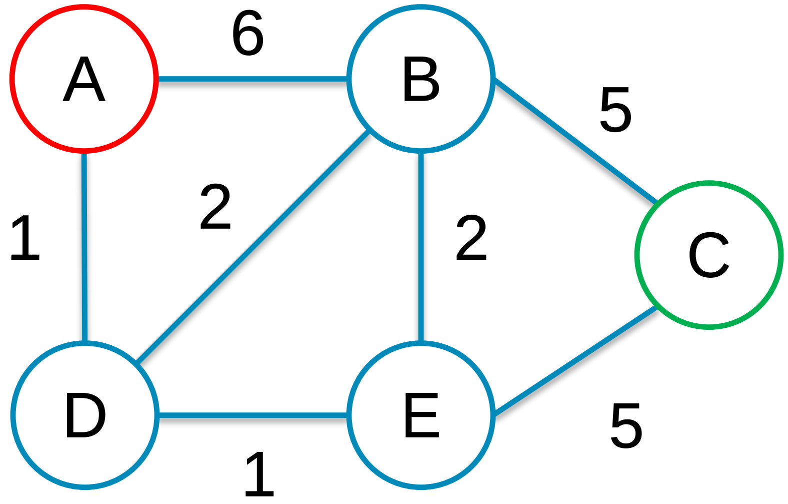
- This process in general is called search

Search

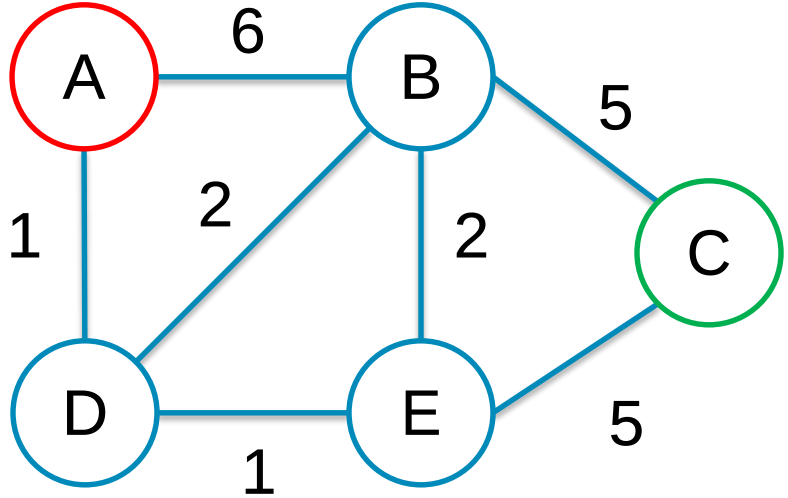
- Search will explore the problem state space
- Searching this space generates a search tree
- The search tree has nodes and edges

Search Tree

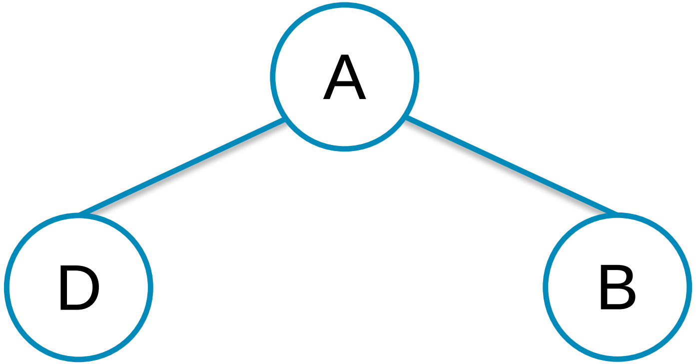


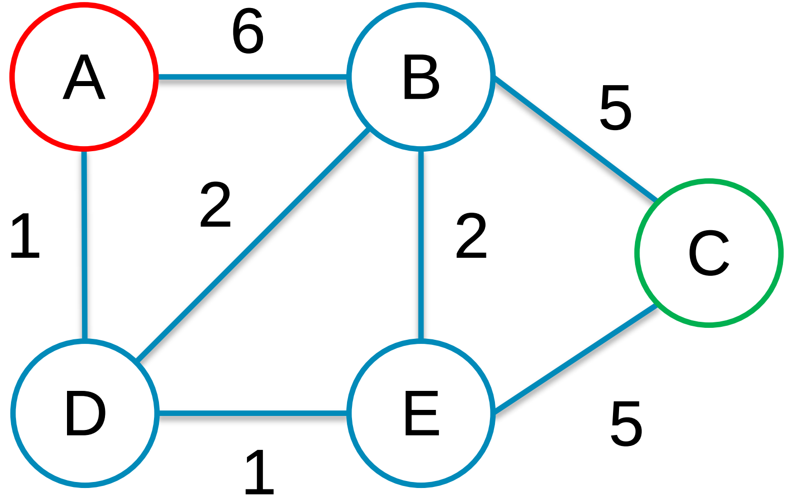


Search Tree

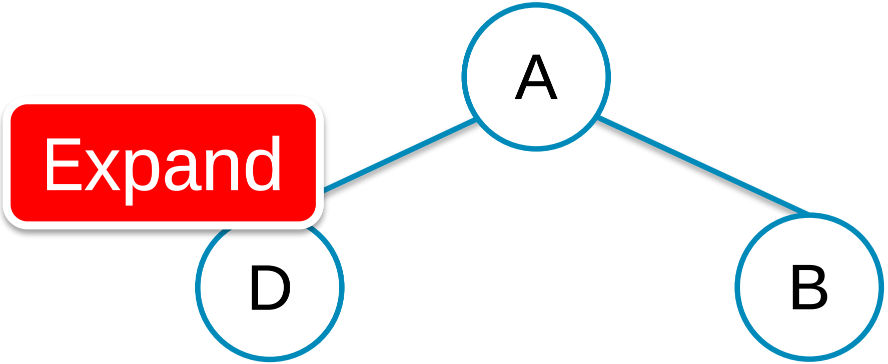


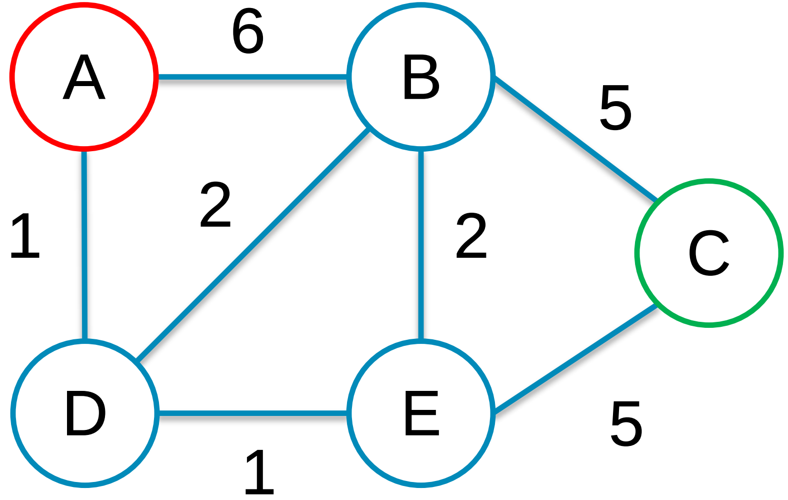
Search Tree



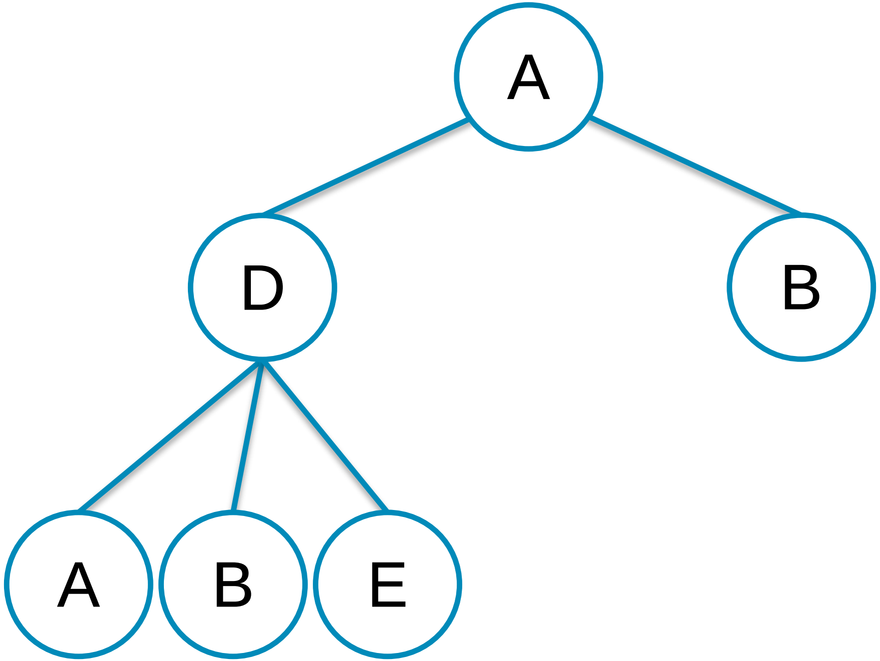


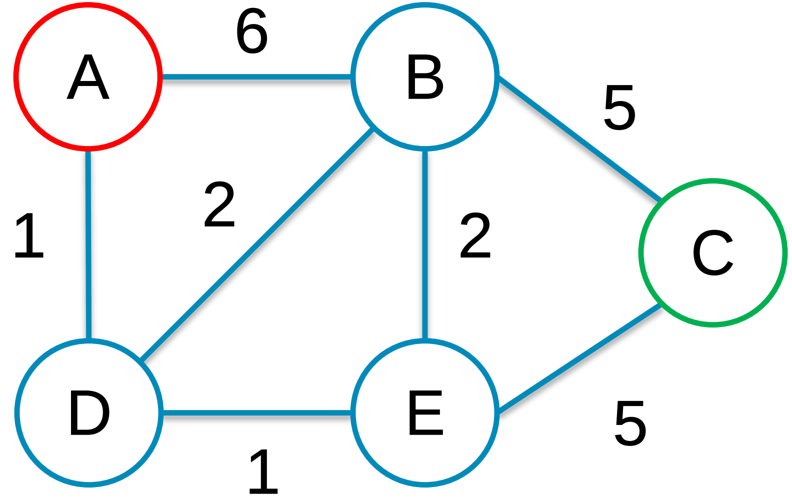
Search Tree



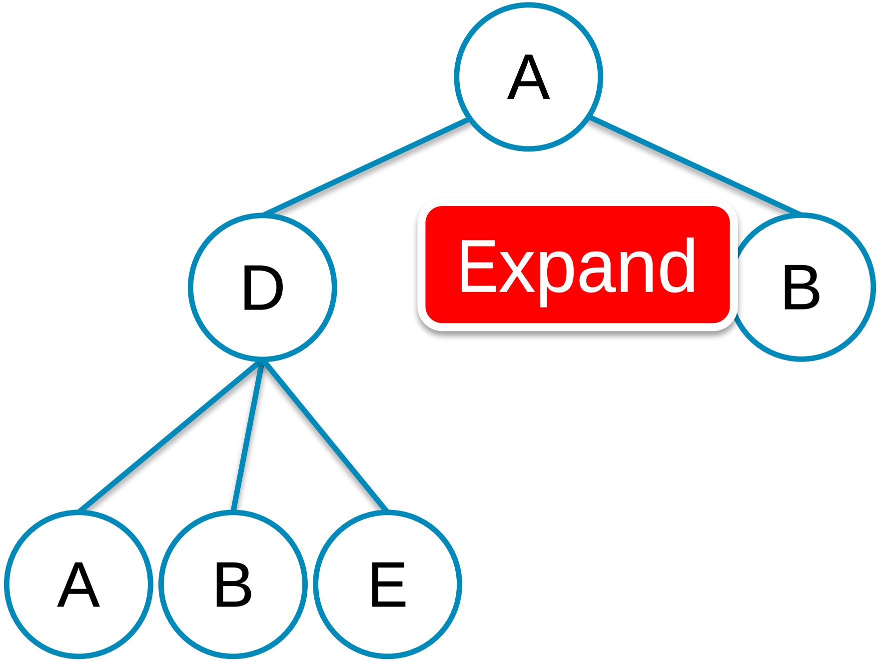


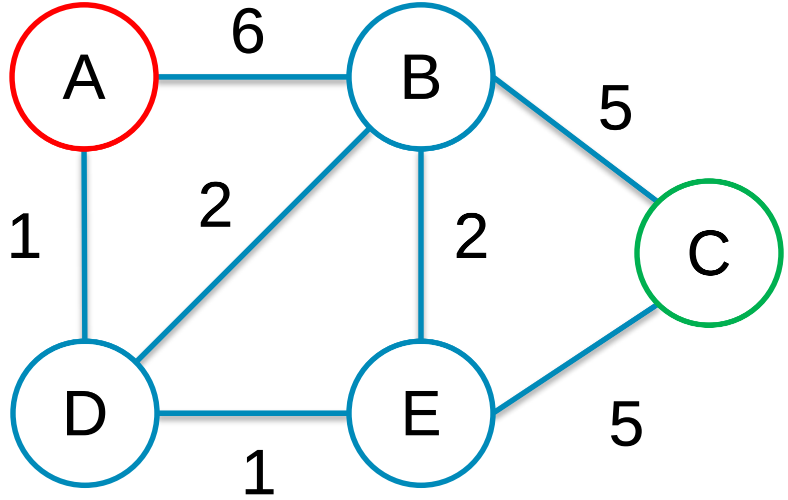
Search Tree



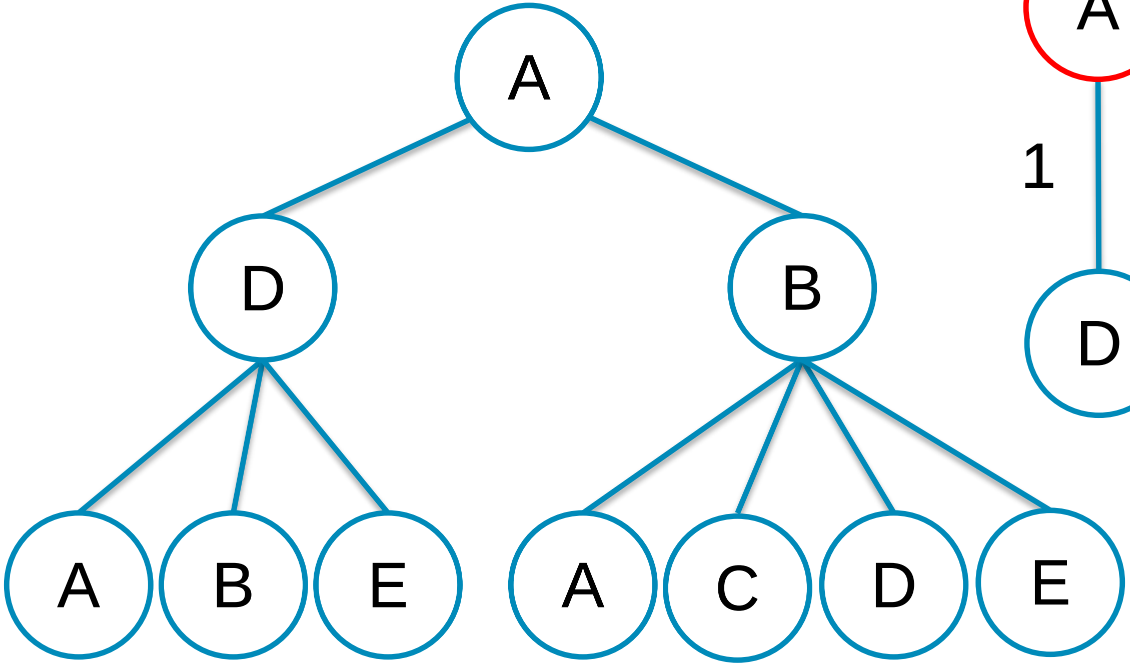


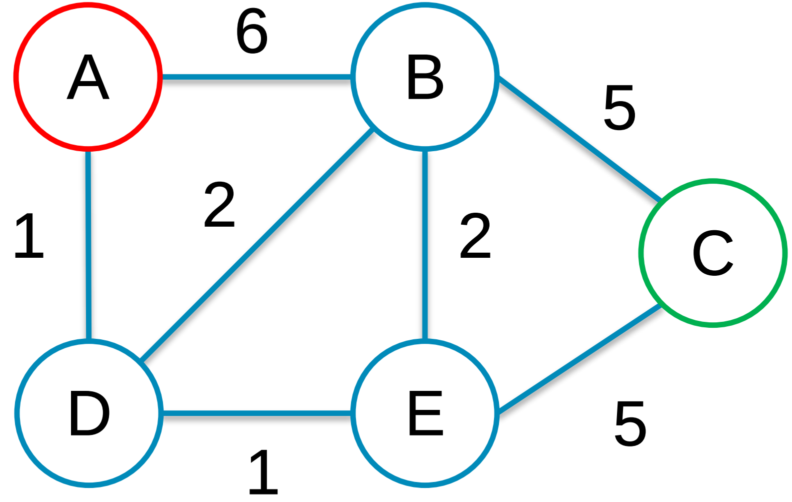
Search Tree



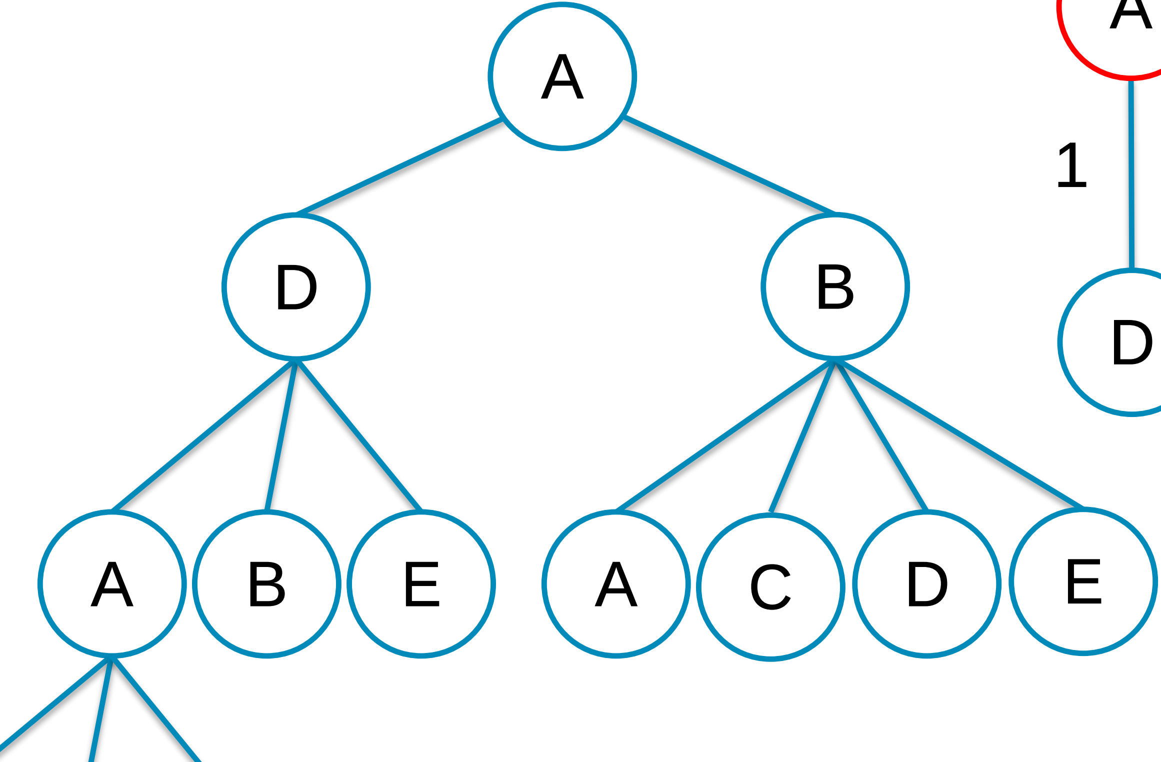


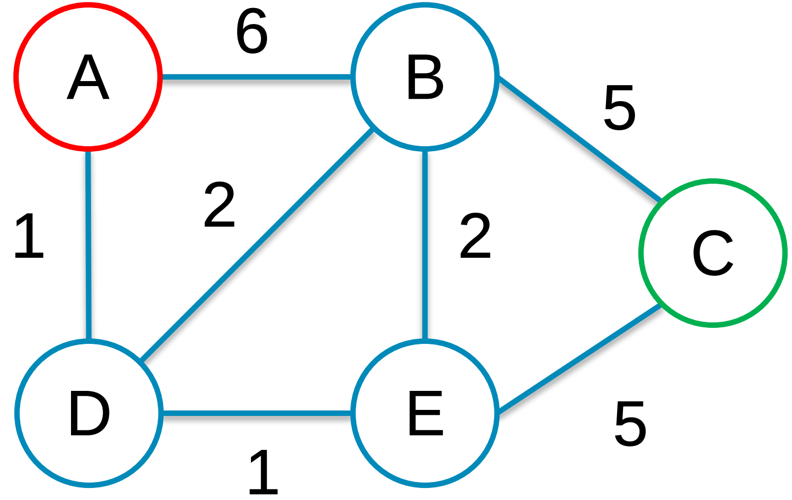
Search Tree



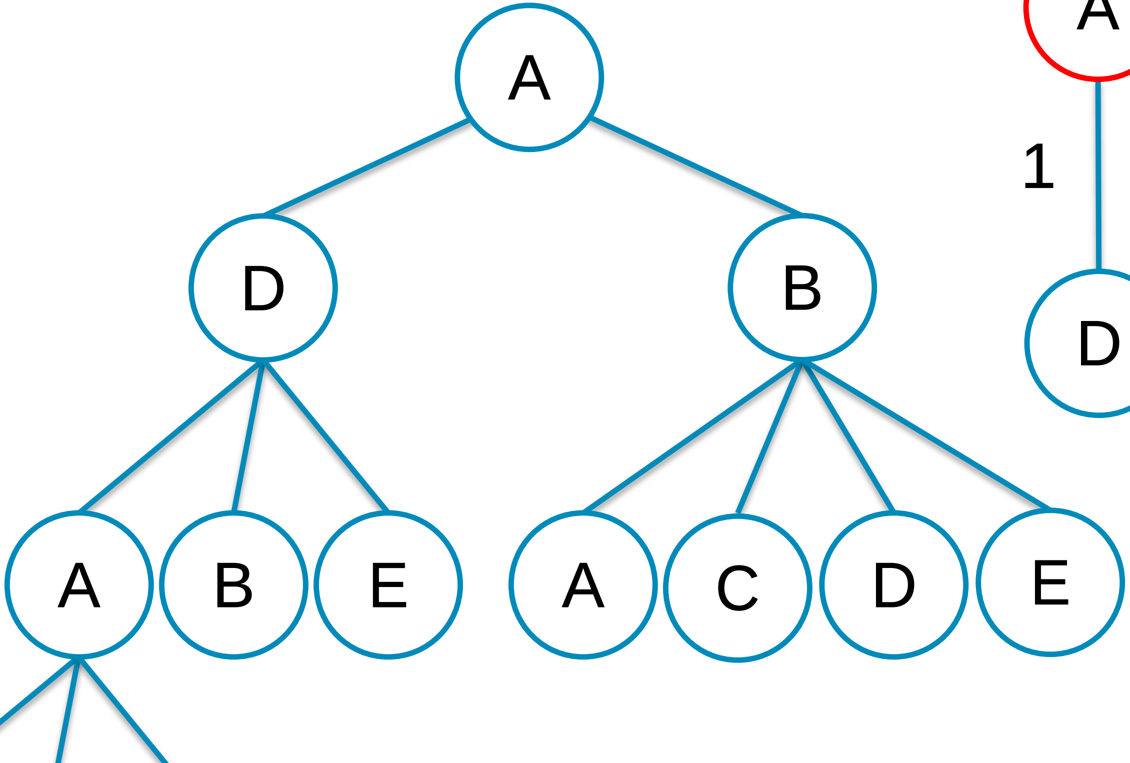


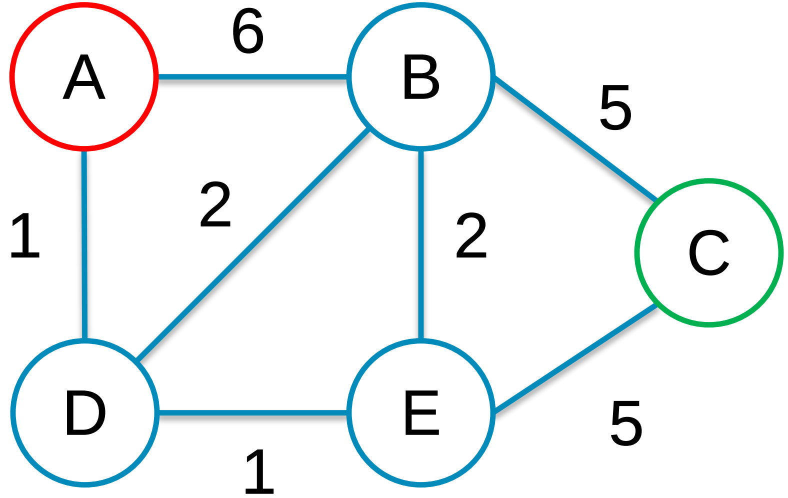
Search Tree





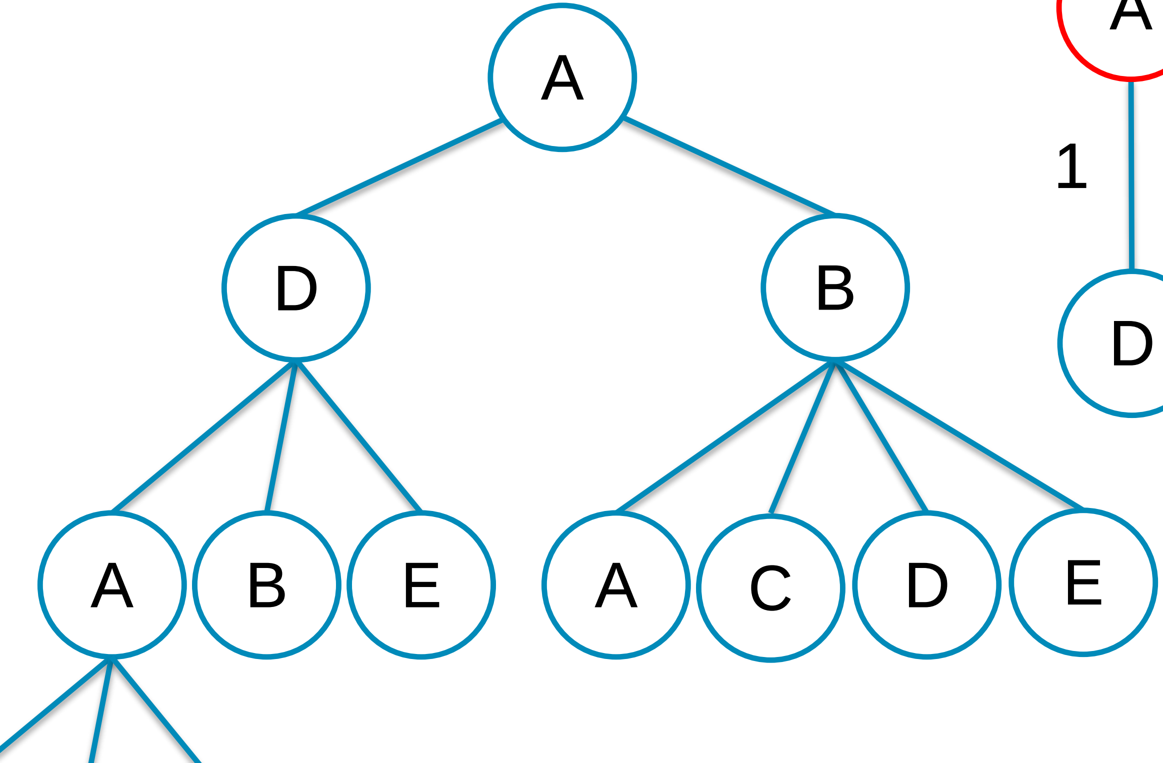
Search Tree

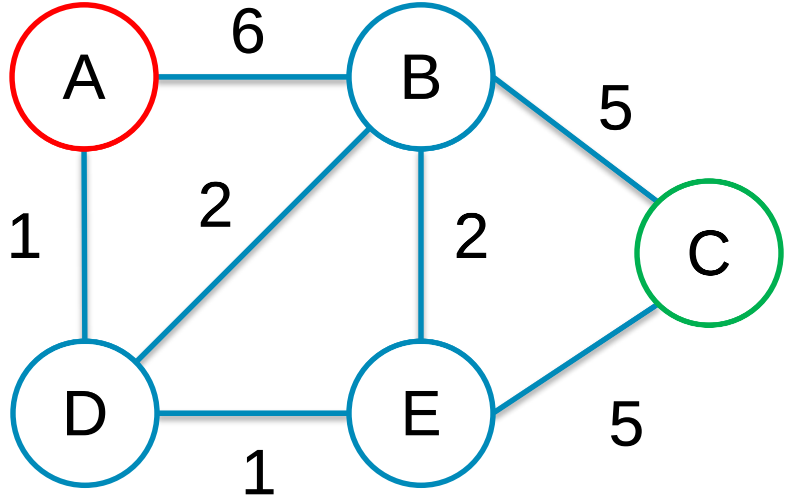




Continue Until Goal Found

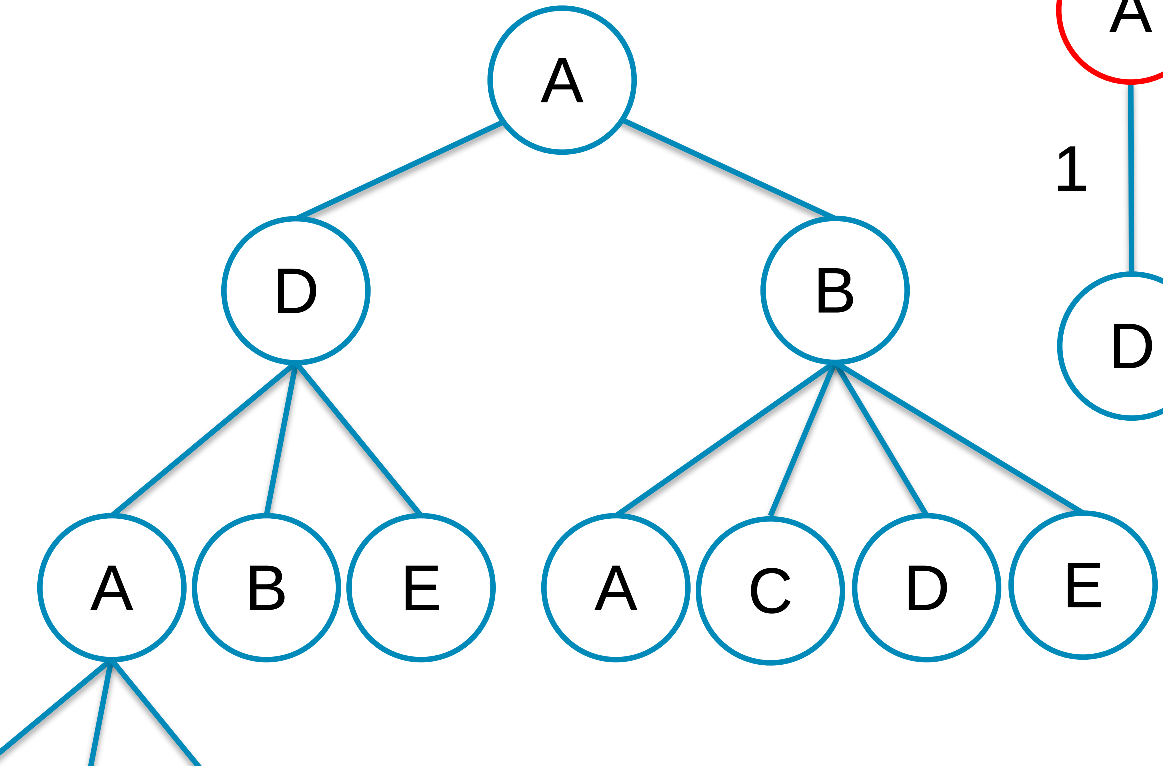
Search Tree

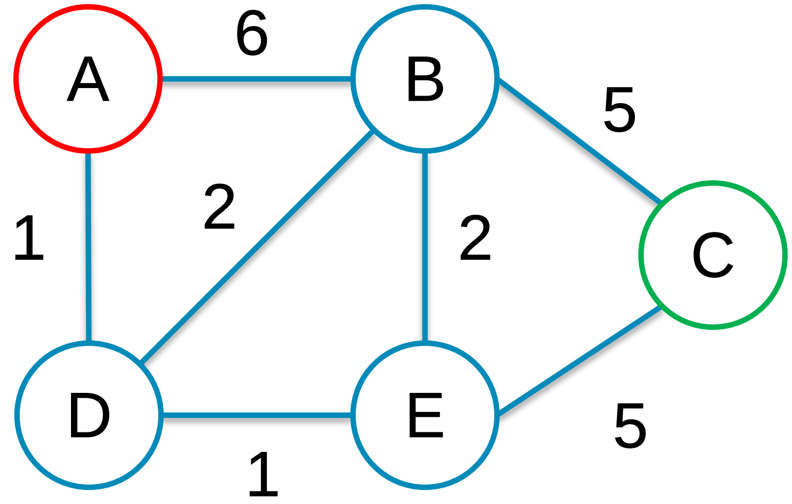




State Space: Finite

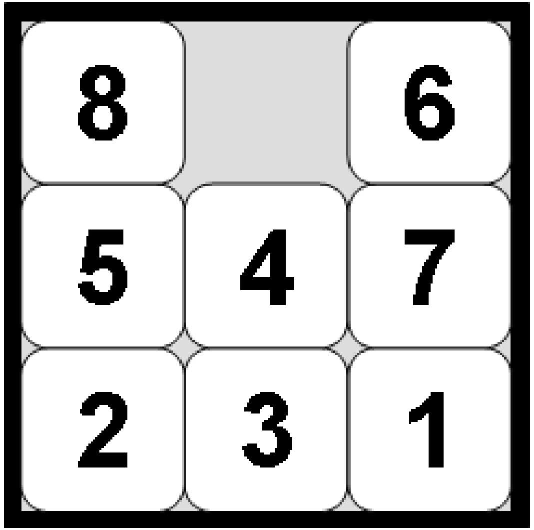
Search Tree

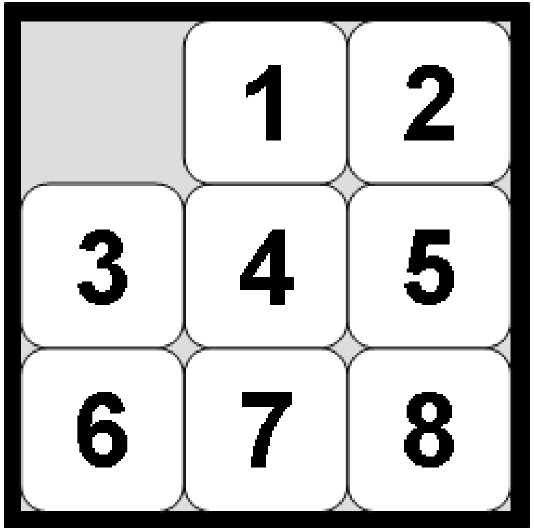




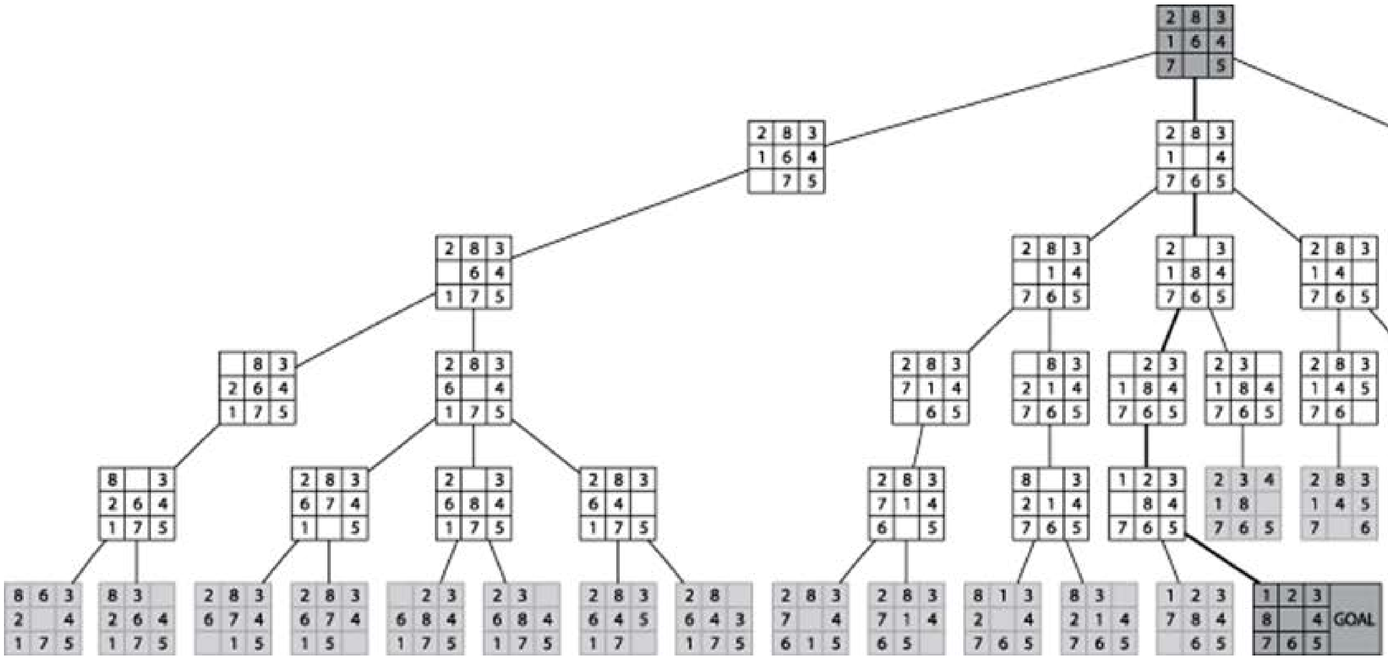
State Space: Finite
Search Tree: Infinite

Example: 15 Puzzle

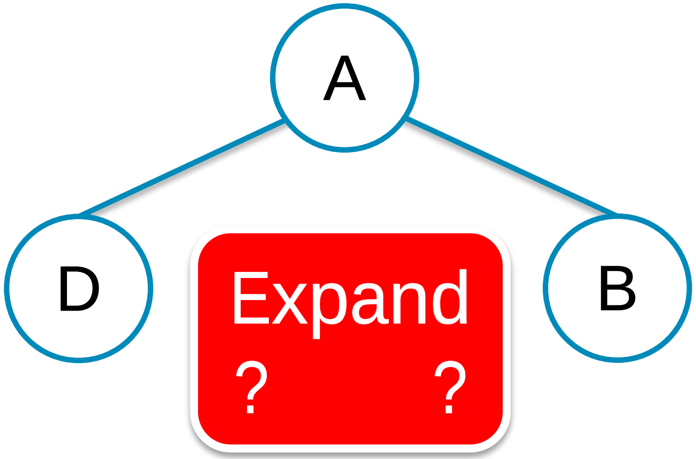


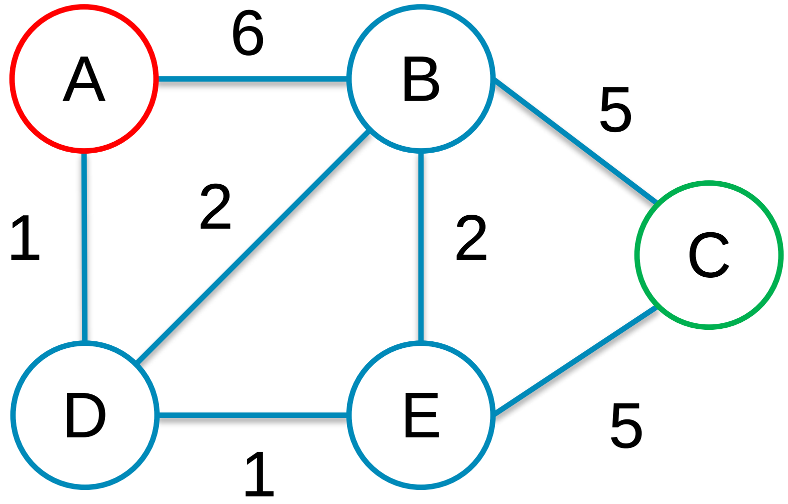


Example: 15 Puzzle



Search Strategy





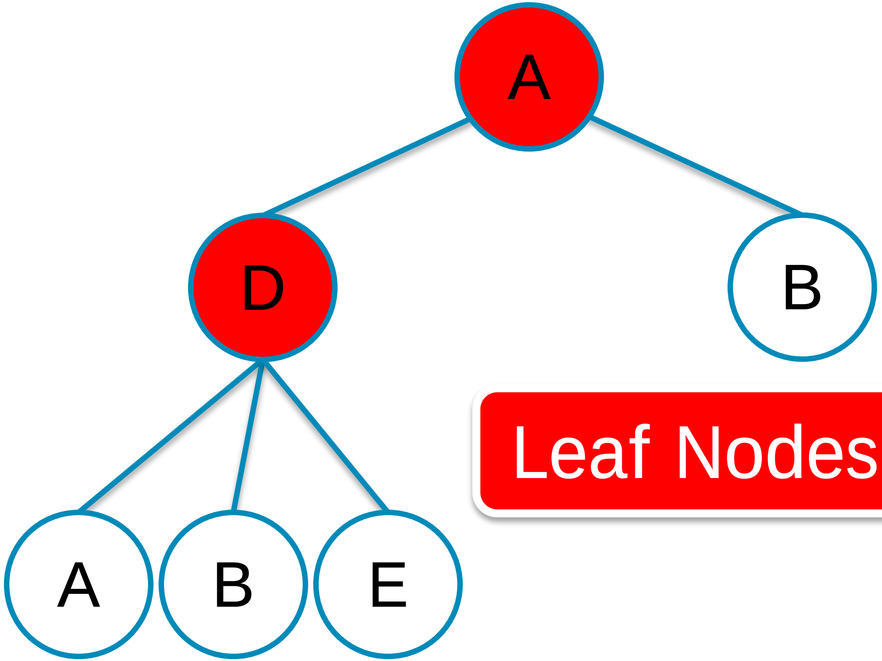
Search Tree Node Data

- State: The state in the graph
- Parent Node: Tree node that generated this node
- Action: Action that produced this node
- Path Cost: Cost of path so far from start
  - Traditionally denoted by g(n)
- Depth: Number of actions to this node
- Other: Can contain precomputed data (optimization)
  - Legal actions, heuristics, etc

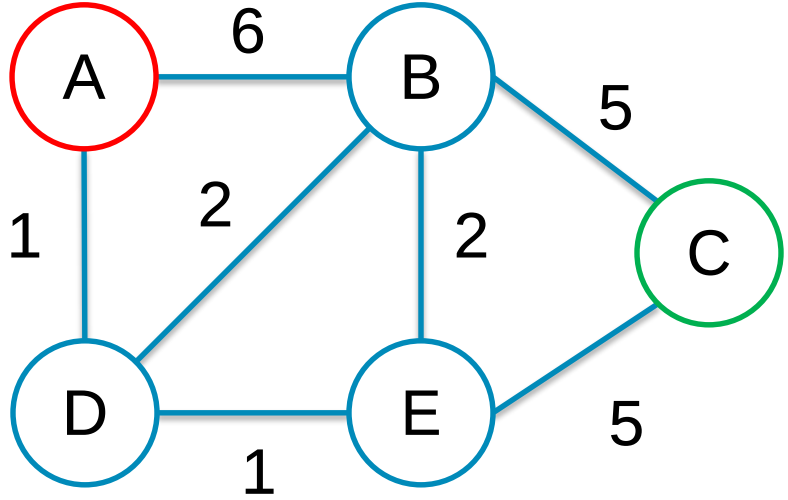
Node vs. State

- Important distinction
- State
  - Configuration of the environment
- Node
  - Bookkeeping data structure
  - Exists only within the search tree
  - Nodes are ‘on paths’ in the search tree

The Fringe



Leaf Nodes



Nodes that have been generated but not yet expanded

General Uninformed Tree Search

```
Function Tree-Search(problem, strategy)
    fringe = {Node(problem.initial_state)}
    while (true)
        if (fringe.empty) return fail
        node = strategy.select_node(fringe)
        if (node.state is goal) return solution
        else fringe.add(Expand(node, problem))
```

Node Expansion

```
Function Expand(node, problem)
    successors = { }
    for a : problem.actions(node)
        s = Node(node.state.do_action(a))
        s.parent = node
        s.g = node.g + s.action.cost
        s.action = a
        s.depth = node.depth + 1
        succesors.add(s)
    return successors
```

Problem Solving Performance

- Completeness
  - Is it guaranteed to find a solution if it exists?
- Optimality
  - Does it find the optimal solution?
- Time Complexity
  - How long does it take to find a solution?
- Space Complexity
  - How much memory is needed to run the search?

Time + Space Complexity

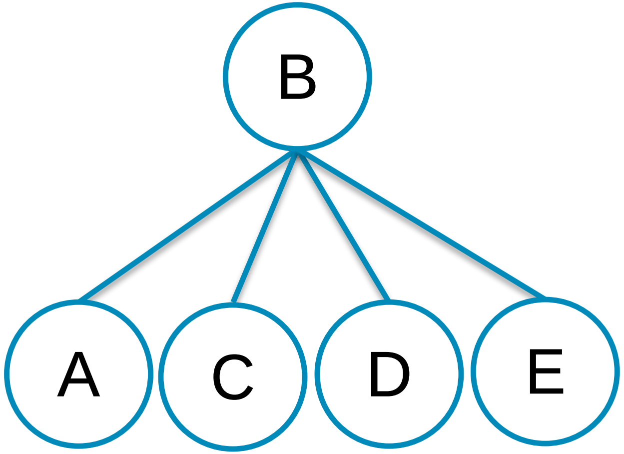
- Measured on size of input problem / graph
- Time measures number of nodes generated
- Space measured by how much memory is needed to store the maximum number of nodes (worst case) needed at any given time
- Search / AI traditionally measures:
  - Branching factor b (max successors at any node)
  - Depth d (of the shallowest goal node)
  - Tree size ~ b^d

Heuristic Search

- Improve upon uninformed tree search
- Incorporate knowledge (inform it)
- Large Search Trees
  - How to make the tree smaller?
  - How to choose a search strategy?
  - How to choose an action to perform?
- Data structures + other optimizations

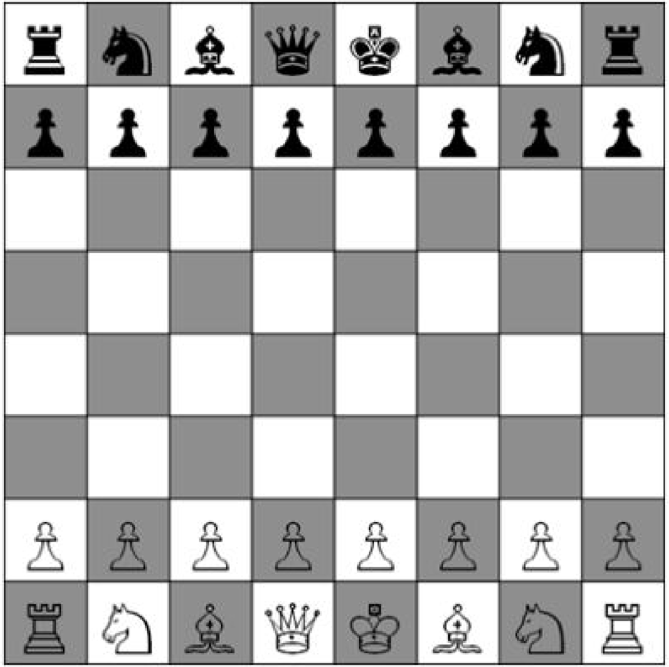
Heuristic Search

- Apply ‘heuristic’ guesses of value of nodes
- Node selection strategy
  - Move ordering
  - Iterative deepening
- “Pruning” nodes
- Bounding solutions



Heuristic Search Example

- Chess has ~40 moves at a state
- Most of them are bad
- Kasparov claimed to have only considered 2 moves per state
- Naïve search considers all 40
- Depth of search:
  - 40: 40, 1600, 64000, 2560000
  - 2: 2, 4, 8, 16, 32, ... depth 21
  - Pruning = Exponential gains



Recap

- Problems are defined by
  - Initial + Goal States, Actions, Cost Function
- Optimal solution
  - Has lowest cost function among all solutions
- Search
  - Finds solution by exploring the search tree
  - Search tree consists of Nodes, Edges
  - Nodes in search tree != states in problem
- Problem solving performance measured by
  - Completeness, Optimality, Space + Time Complexity