# G51FAI
# Fundamentals of AI

## Instructor: Siang Yew Chong

*Problem Formulation and Representation*

# Outlines

- Problem Formulation
  - goal(s), operator(s)
- State Space & Search Tree
- Representations
  - problem
  - states, nodes
- Performance Evaluation Criteria

# Solving Problems by Searching

- For an intelligent agent to work we need to answer the following questions:
  - What is the goal to be achieved?
  - What are the actions?
  - What is the representation?
    - e.g., what *relevant* information is necessary to encode in order to describe the state of the world, describe the available transitions, and solve the problem?

# Problem Components

- Initial State
  - The starting state of the problem, defined in a suitable manner
- Operator(s)
  - An action or a set of actions that moves the problem from one state to another
  - The set of all possible states reachable from a given state by applying all legal action(s) is known as the neighbourhood and the action(s), the successor function

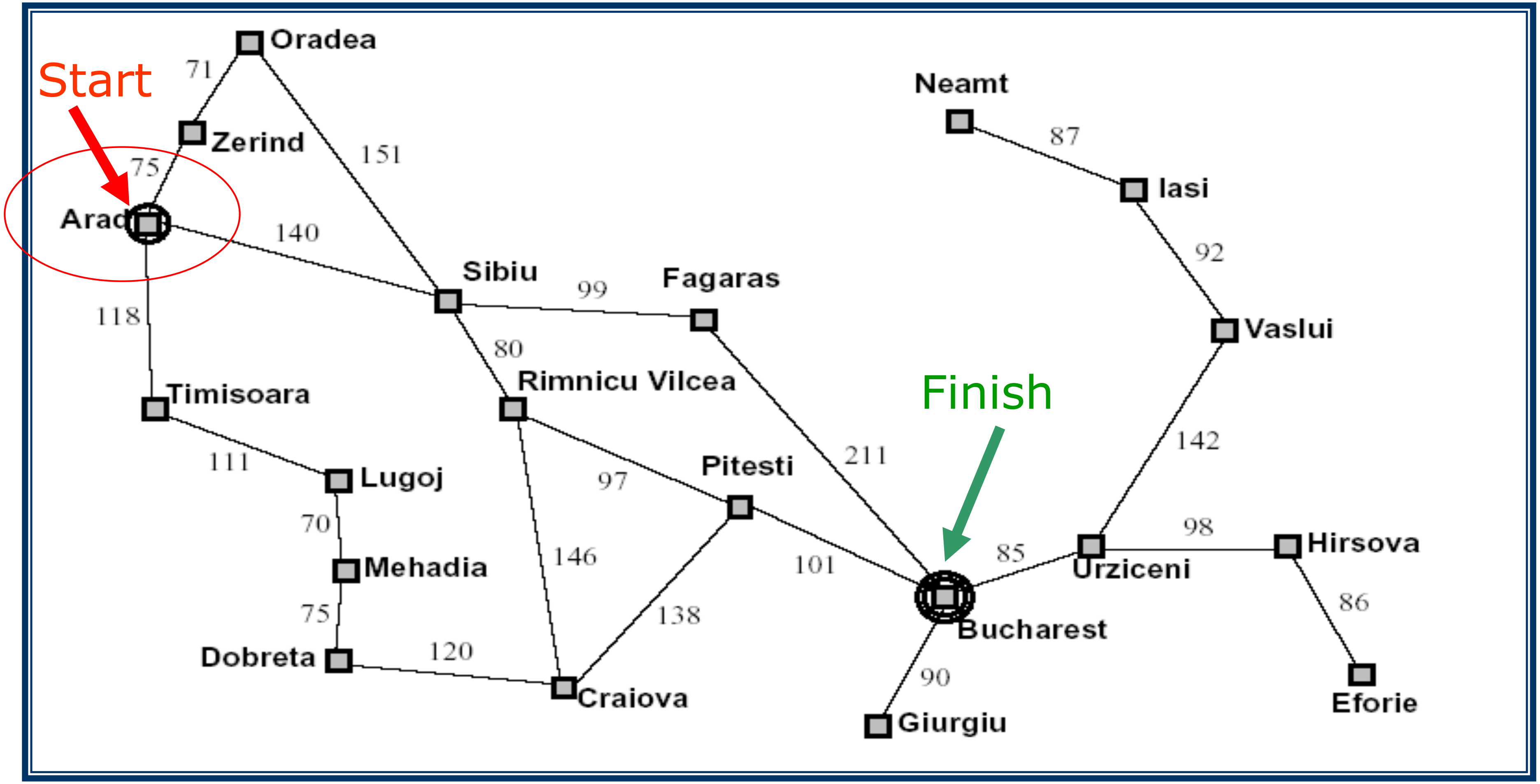
# Problem Components

- Goal Test
  - A test applied to a state which returns *true* if we have reached a state that solves the problem
- Path Cost
  - How much it costs to take a particular sequence of actions

*Note*: The initial state and the successor function define the *state space* which is the set of all states reachable from the initial state

The complexity of a problem depends on the size of the *state space*.

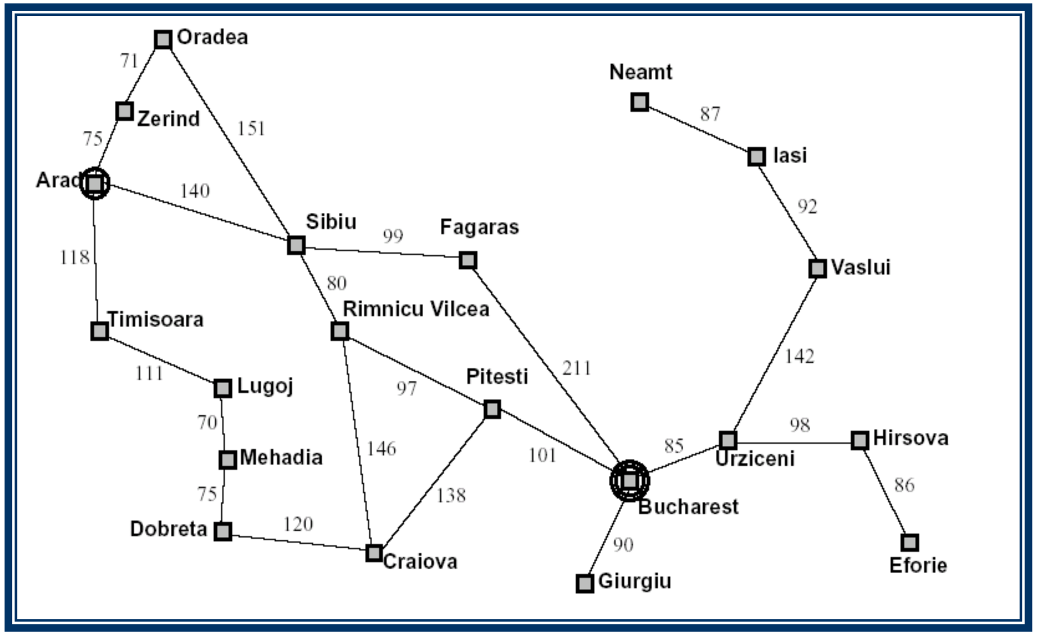
# Problem Formulation - Romania



# Problem Formulation - Romania

- Initial State → Arad
- Operator
  - driving between cities
  - state space consists of all 20 cities in the graph
- Goal Test
  - is the current state (city) Bucharest?
  - a solution is a path from the initial to the goal state
- Path cost is a function of time/distance/risk/petrol/…

*Q: What is the neighbourhood of Arad?*



# Problem Formulation
# 8-Puzzle

| | | |
|---|---|---|
| 5 | 4 | |
| 6 | 1 | 8 |
| 7 | 3 | 2 |

Initial State

| | | |
|---|---|---|
| 1 | 4 | 7 |
| 2 | 5 | 8 |
| 3 | 6 | |

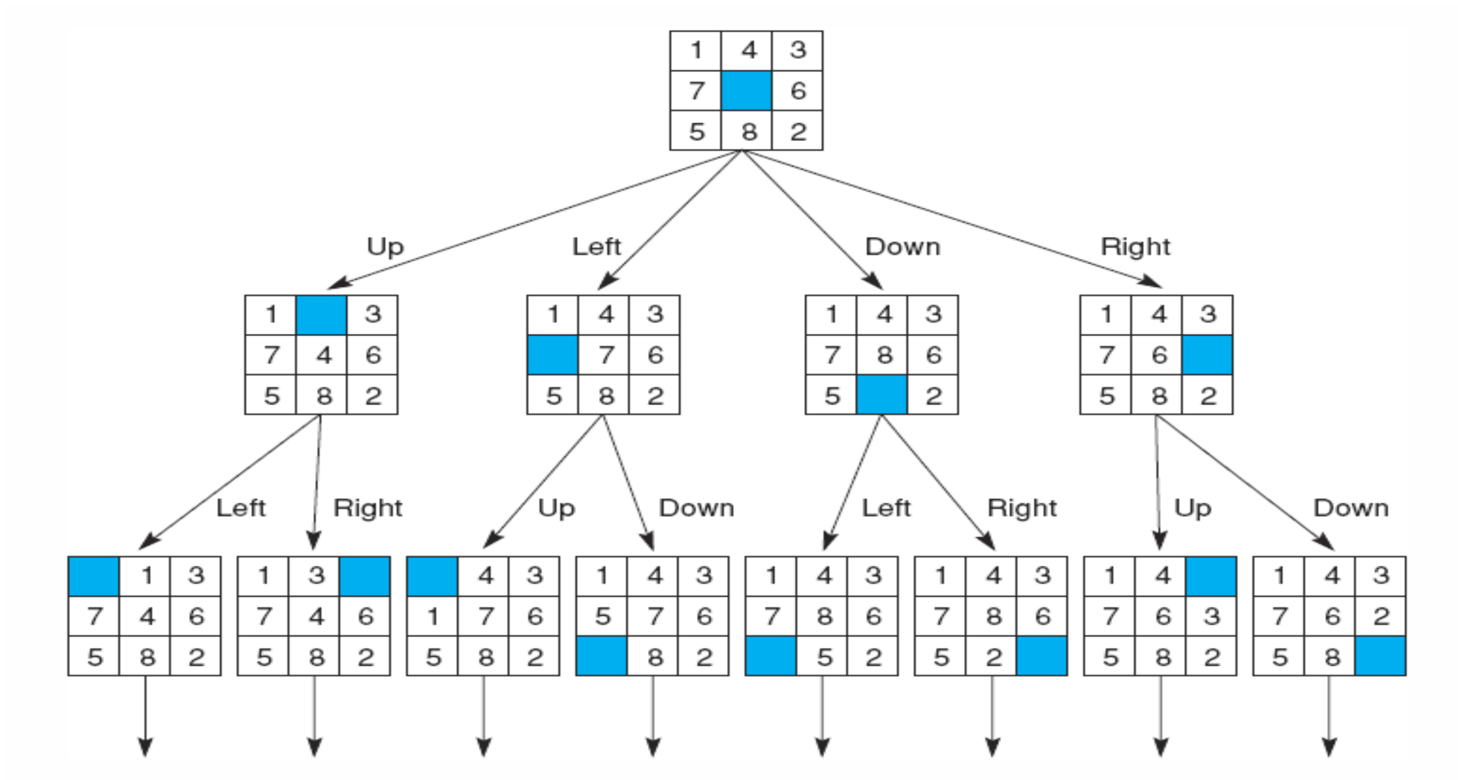
Goal State

# Problem Formulation
# 8-Puzzle

- Initial State
  - specifies the location of each of the eight tiles and the blank in one of the nine squares
- Operators
  - blank tile moves left, right, up or down
- Goal Test
  - the current state matches a certain state (e.g. the goal state shown on the previous slide)
- Path Cost
  - each move of the blank costs 1

*Q: How big is the state space?*

# 8-Puzzle




- The number of actions/operators depends on how they are formulated
  - 4 possible moves could be specified for each of the 8 tiles, resulting in a total of **4*8=32** operators.
  - On the other hand, 4 moves for the “blank” square could be specified instead so only **4** operators are needed.

=> Formulation shift can greatly simplify a problem!

# State Space Representation

- We can use graphs to model the deeper structure of a problem → state space graphs
- A graph consists of a set V of nodes, and a set E of edges
  - Nodes
    - have a unique label for identification
    - represent possible stages of a problem
  - Edges
    - connection between two nodes
    - represent inferences, moves in a game, or other steps in a problem solving process

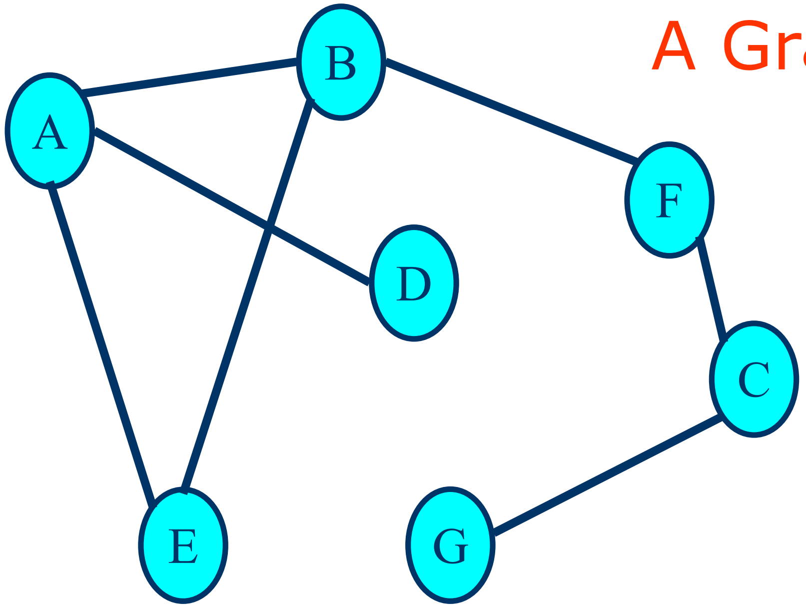
# Searching the State Space

- Representing the entire space of problems as *state space* provides a powerful tool for measuring the structure and complexity of problems and analysis of the efficiency, correctness and generality of solution strategies
- ***Problem solving*** is a ***process*** of ***searching*** the ***state space*** for a ***path*** to a ***solution***
- The ***choice*** of which ***state to expand*** is determined by the ***search strategy***
- The corresponding ***sequences*** of ***state expansion*** is ***represented*** by a data structure known as a ***search tree***

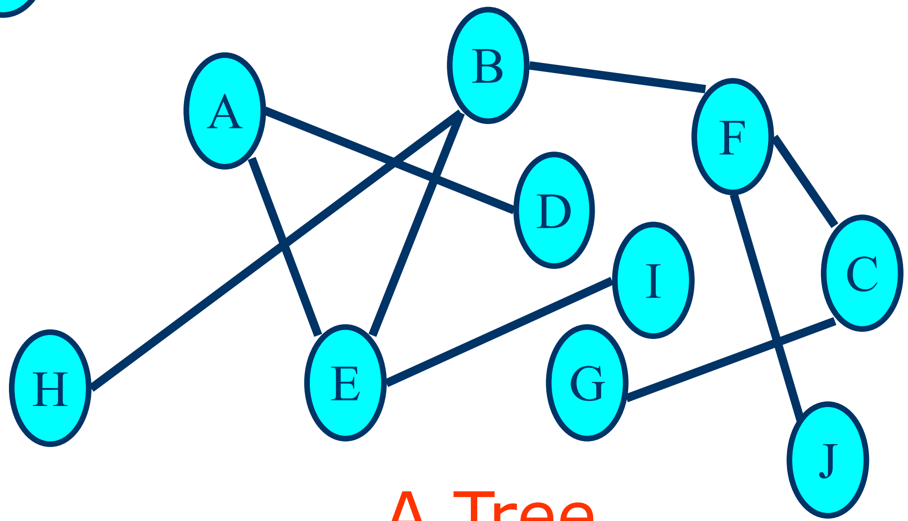
# Search Trees

- A ***tree*** is a ***graph*** that:
  1. is connected but becomes ***disconnected on removing any edge***
  2. is ***connected and acyclic***
  3. has ***precisely one path between any two nodes***
- Property 3, unique paths, makes them much easier to search and so we will start with search on (rooted) trees

# Graphs/Trees

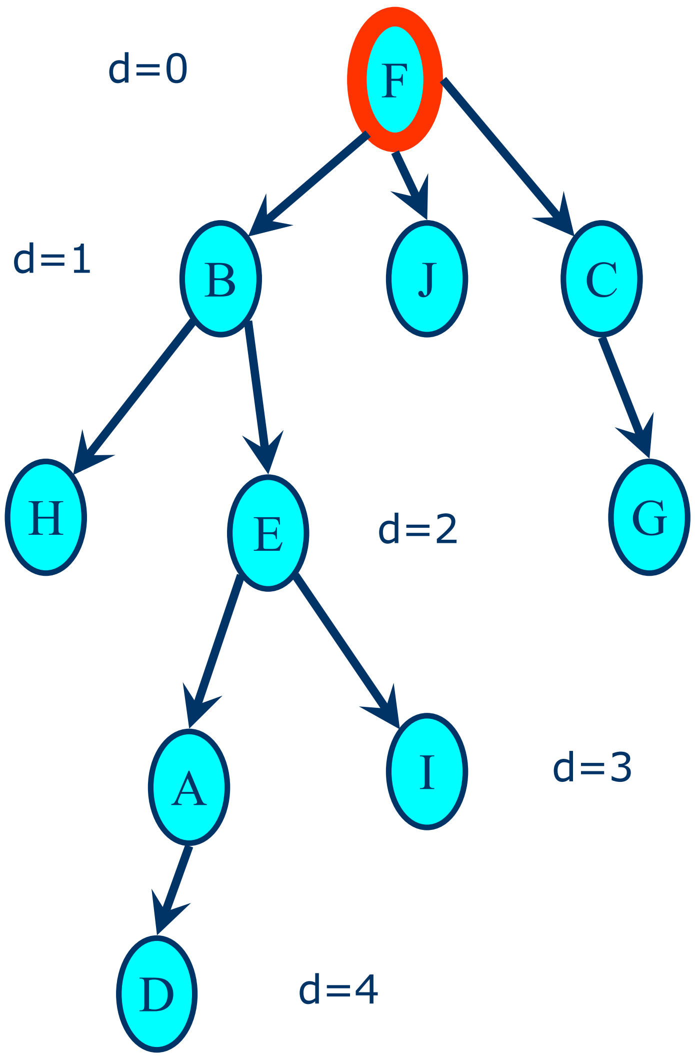



A Graph




A Tree

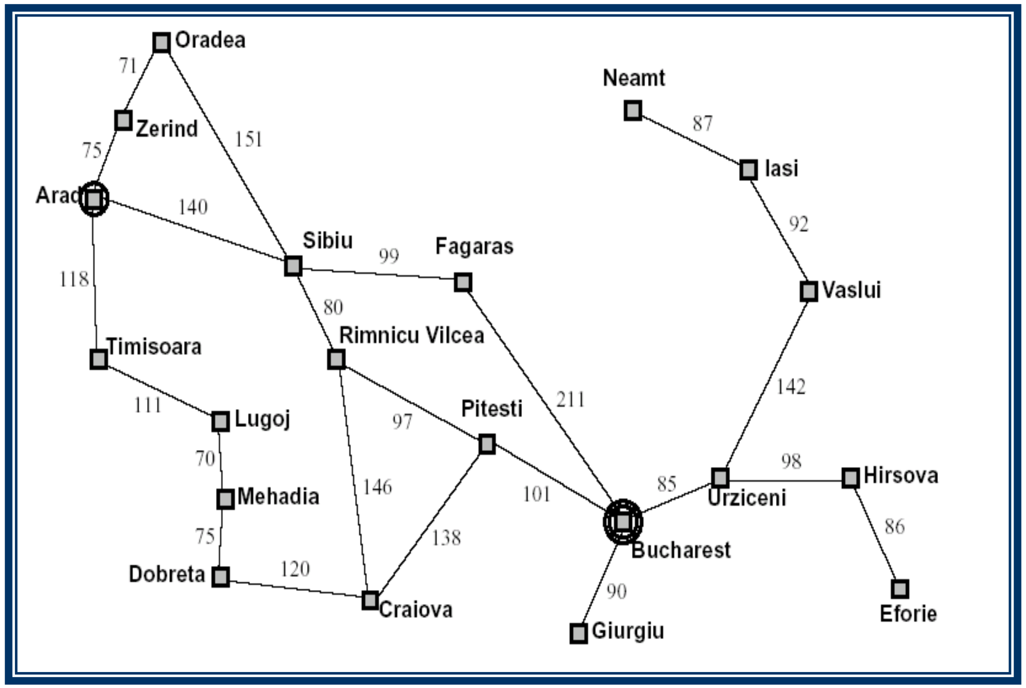
# Trees - Terminologies




- Nodes
  - ✓ Root Node
  - ✓ Children Node
  - ✓ Parent Node
  - ✓ Leaves
- Branching Factor
  - ✓ Average number of children for the nodes of a tree
- The depth, d, of a node is just the number of edges away from the root node
- The depth of a tree is the depth of the deepest node
  - ✓ in this case, depth=4

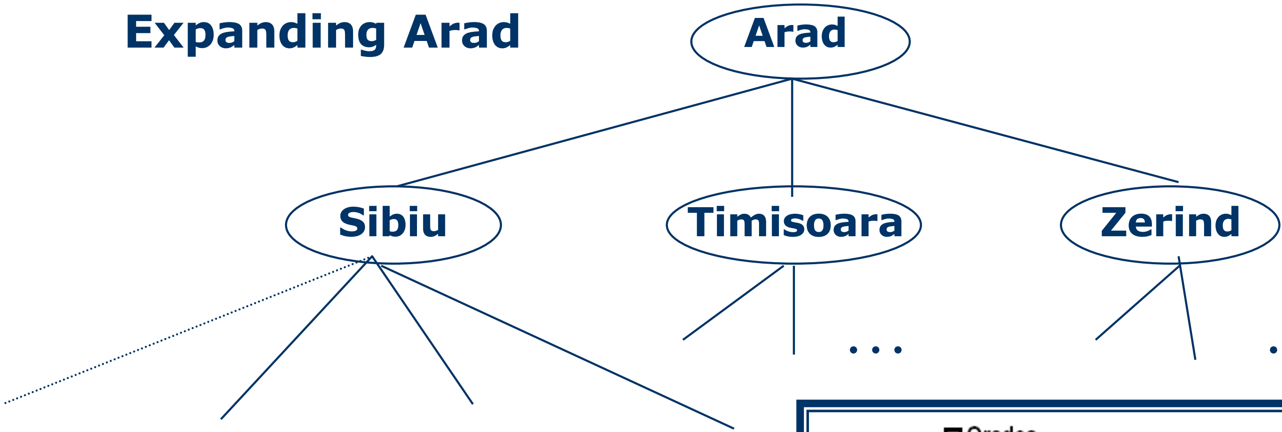
# Search Tree – Romania

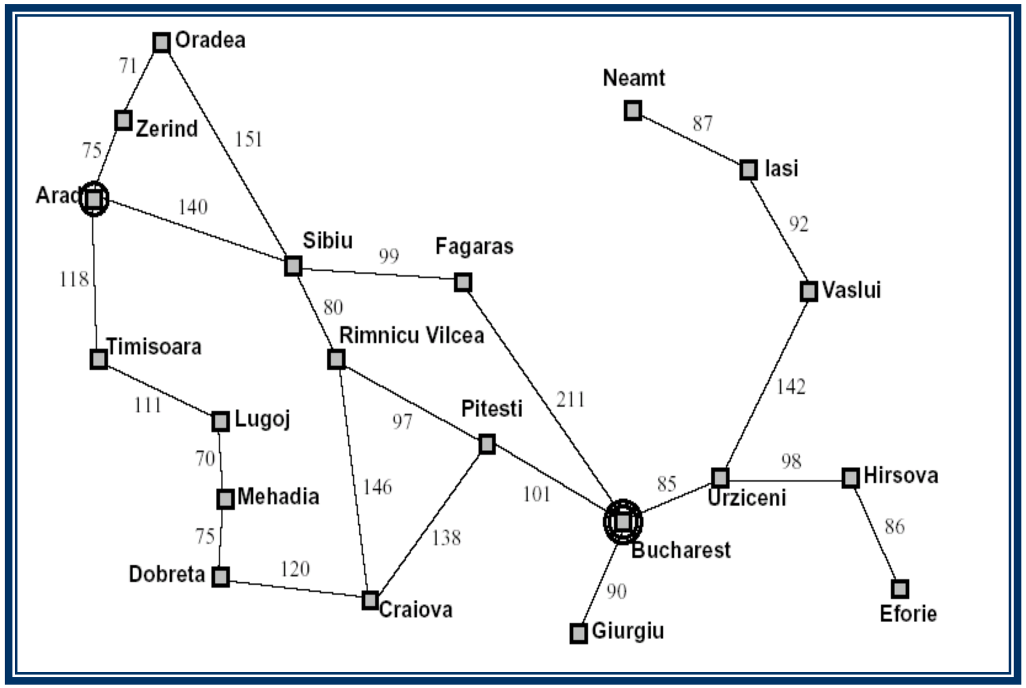
**Initial State**

# Search Tree – Romania

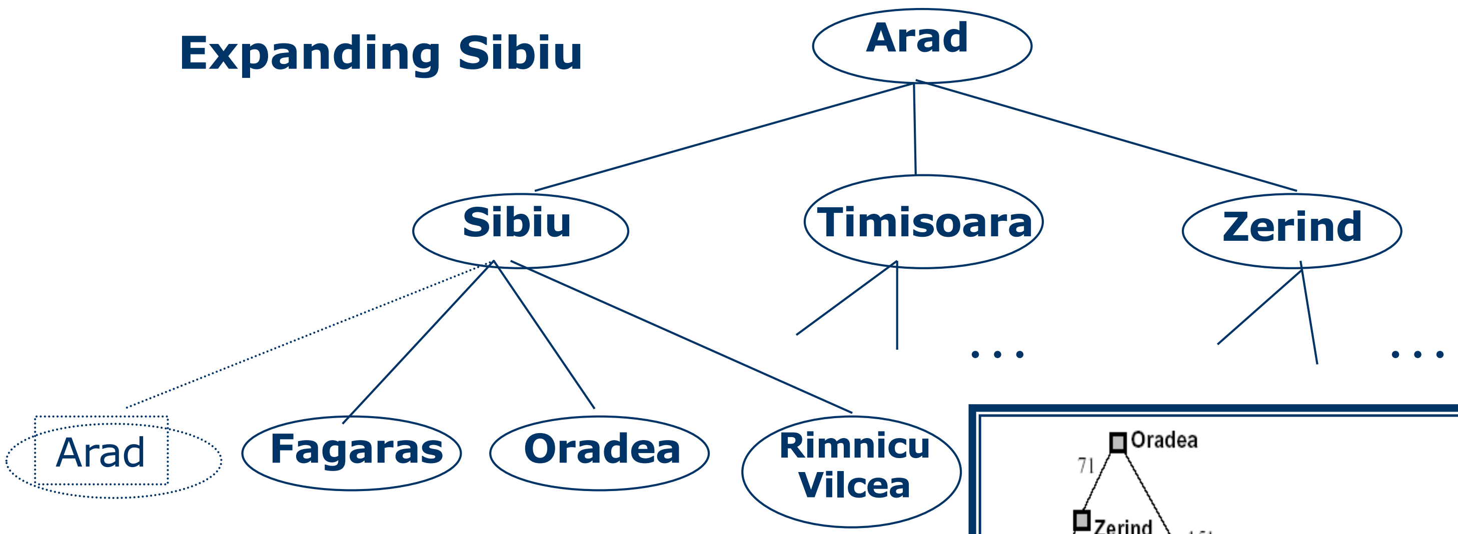
**Expanding Arad**

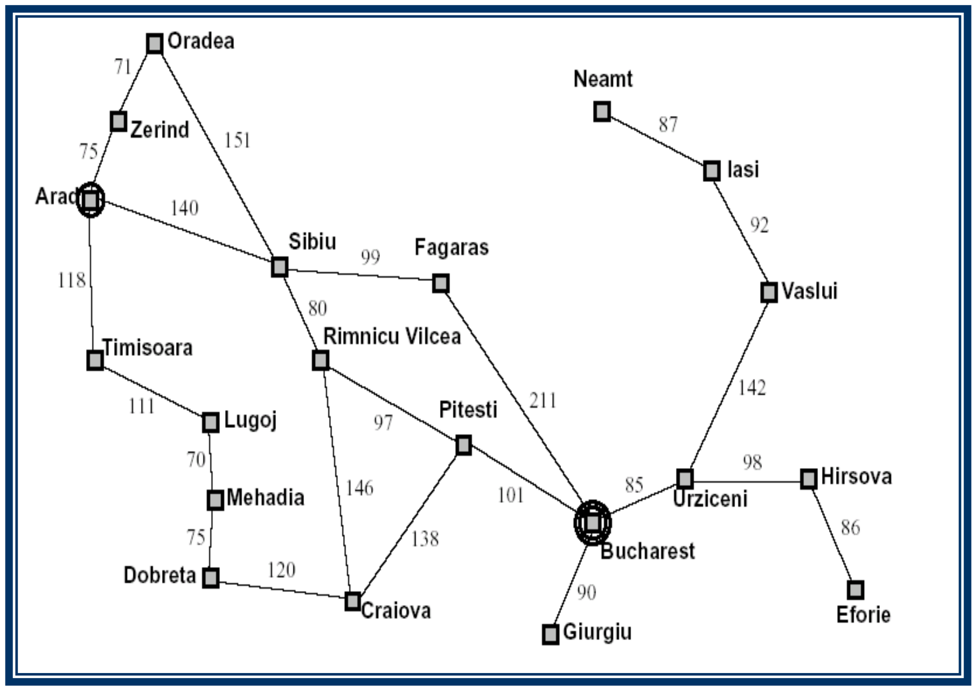


# Search Tree – Romania

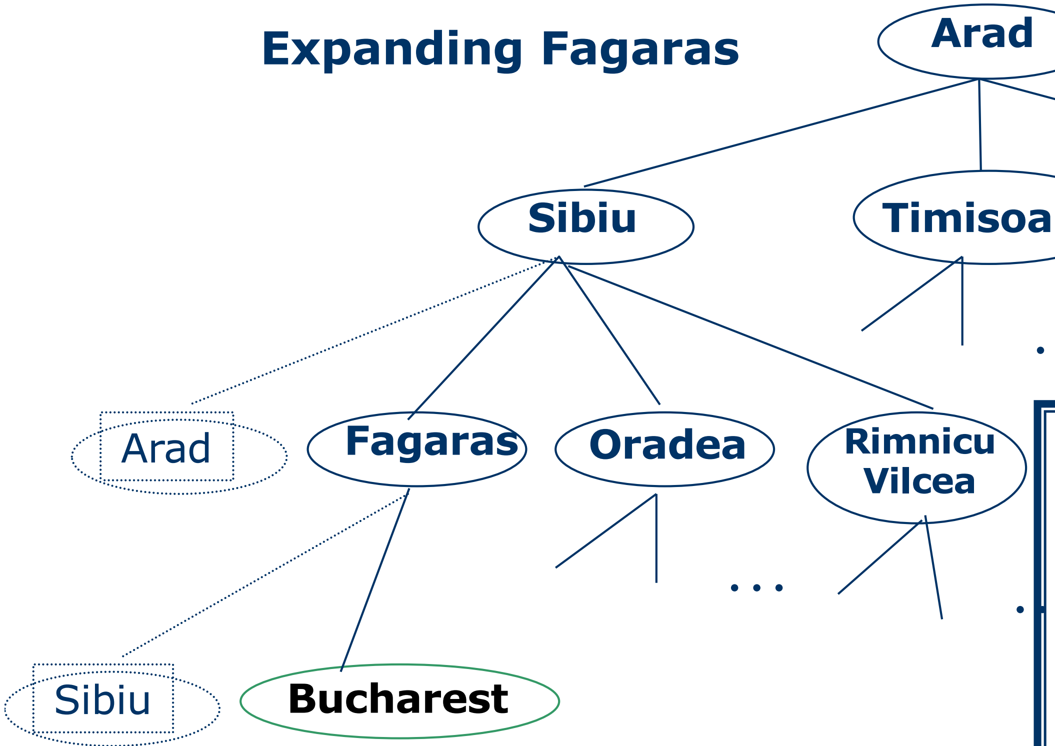
**Expanding Sibiu**

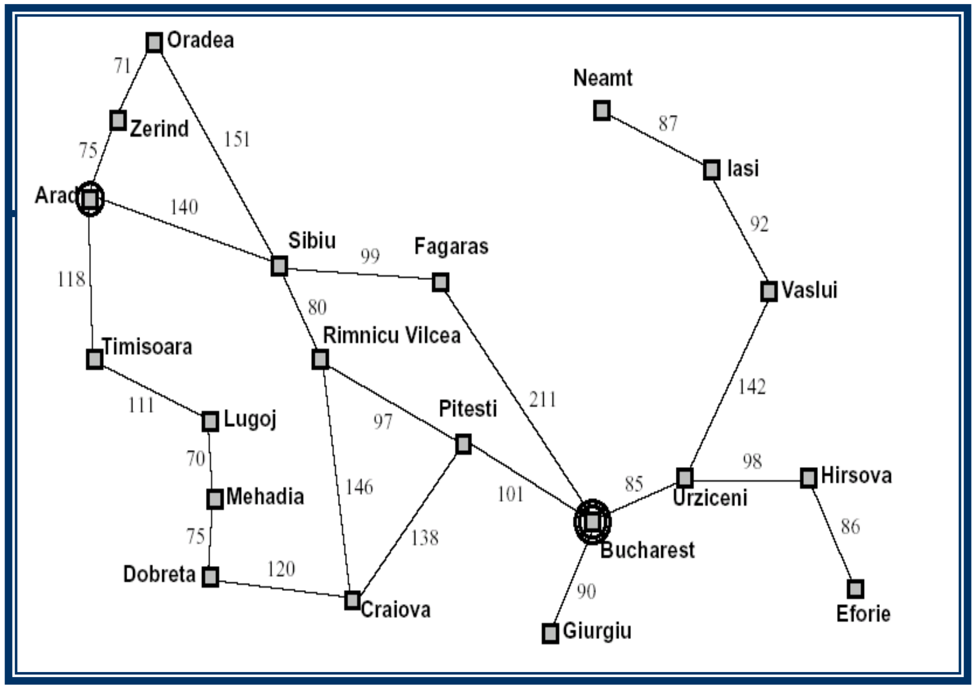


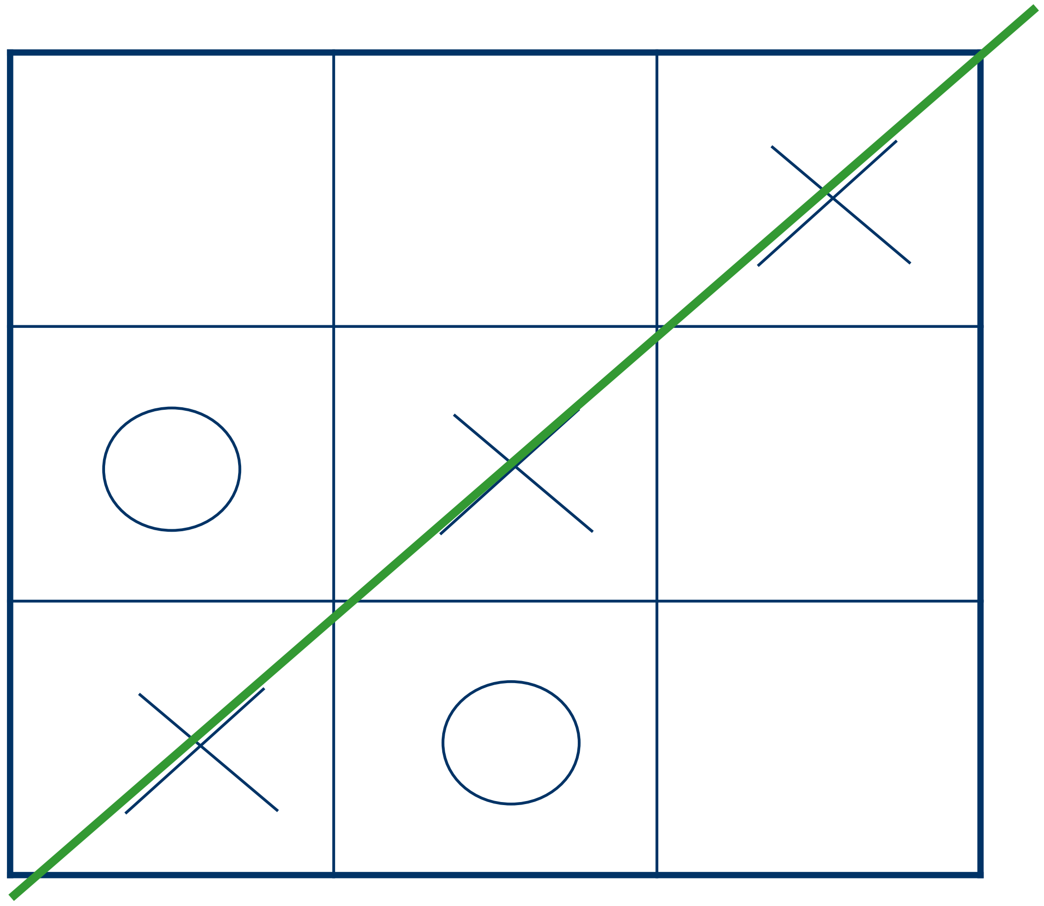
# Search Tree – Romania

**Expanding Fagaras**

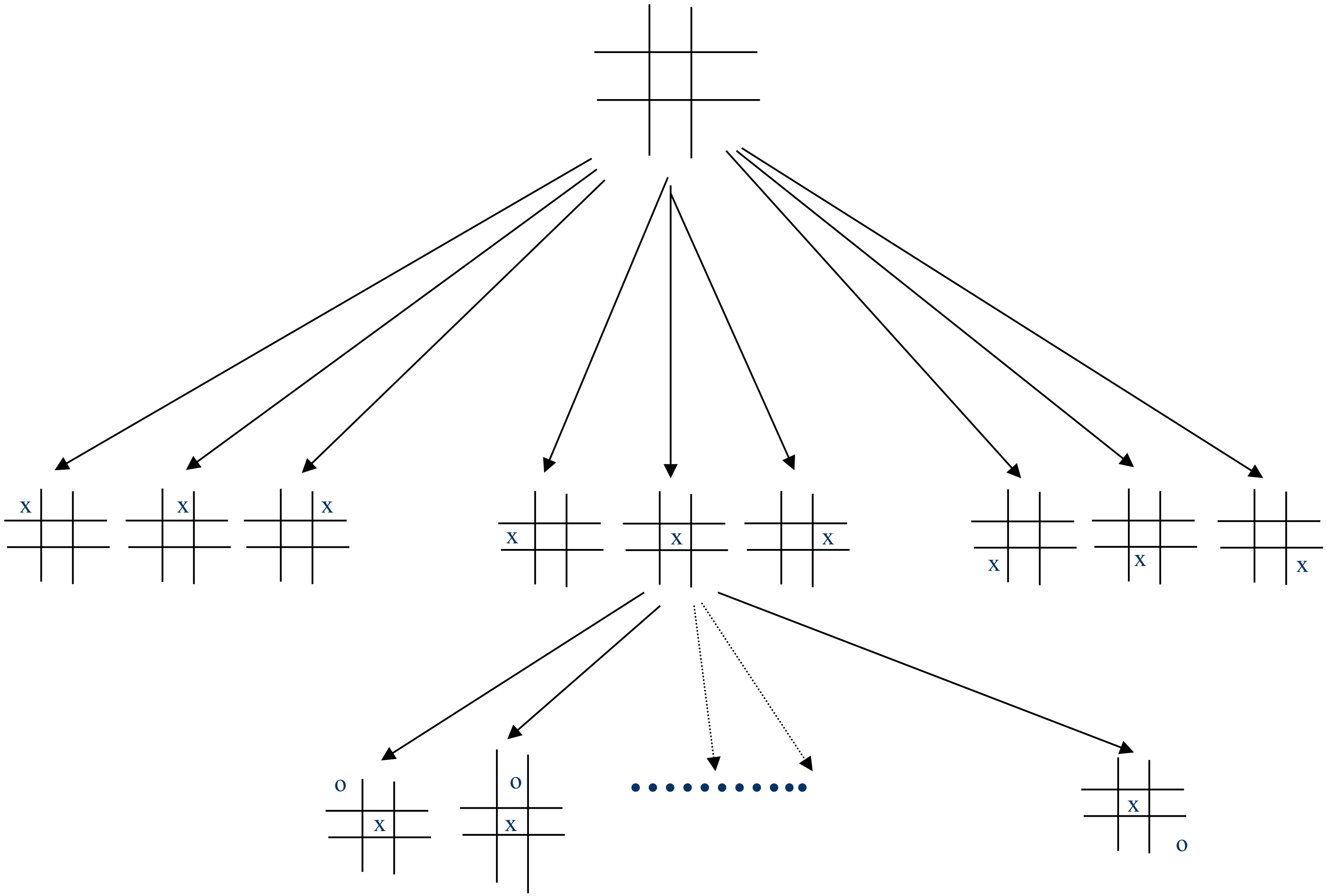


# Search Tree – Tic-Tac-Toe

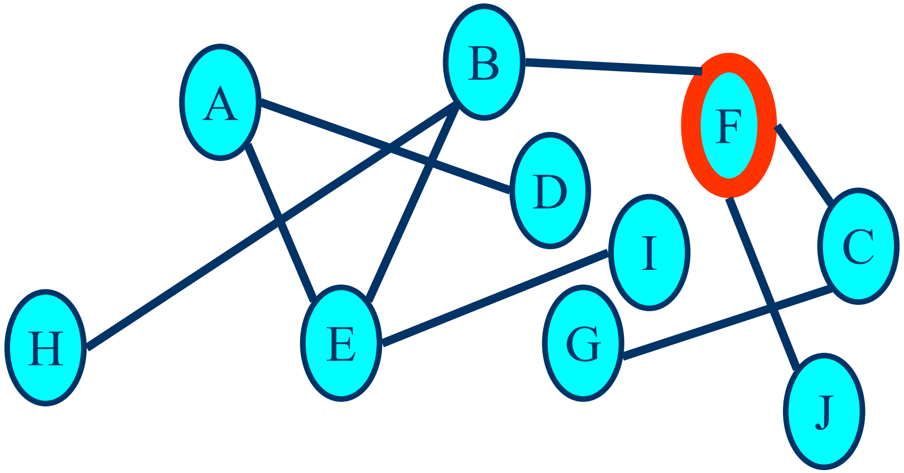


# Search Tree – Tic-Tac-Toe



# Finding Goals in Trees

- Does the following tree contain a node "I"?



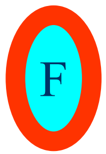

- Yes. How did you know that?
  – "read it"?
- "Trivial!": so why the big deal about search?

# Why is Goal Search Not Trivial?

- Because the graph is ***not given in a nice picture*** “on a piece of paper”
- At the start of the search, the ***search algorithm does not know***
  - the ***size*** of the tree
  - the ***shape*** of the ***tree***
  - the ***depth*** of the goal states
- How big can a search tree be?
  - say there is a constant branching factor b
  - and one goal exists at depth d
  - search tree which includes a goal can have *$b^d$ different branches in the tree (worst case)*
- Examples:
  - $b = 2, d = 10$: $b^d = 2^{10} = 1024$
  - $b = 10, d = 10$: $b^d = 10^{10} = 10{,}000{,}000{,}000$

# Finding Goals in Trees: Reality

- Does the tree under the following root contain a node "G"?




- All you get to see at first is the root node
  - and a guarantee that it is a tree
- The rest is up to you to discover during the process of search → discover/create "on the fly"

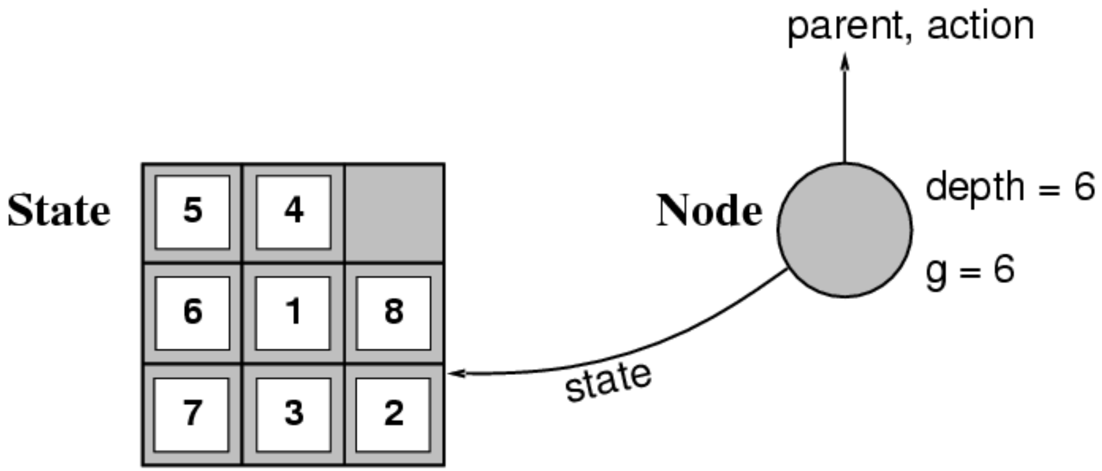
# Properties of Search

- We will say a ***search method*** is ***complete*** if it has both ***propertie***s:
  - if a goal exists then the search will always find it
  - if no goal exists then the search will eventually finish and be able to say for sure that no goal exists
- We only look at complete search methods
  - incomplete search methods often work better in practice, but are the topics related to evolutionary algorithms

# Problem Representation

- The elements of problem formulation can be gathered together into a single data structure as input to a problem solving agent
  - Datatype PROBLEM
    - ✓ Components
      - INITIAL-STATE,
      - OPERATORS,
      - GOAL-TEST,
      - PATH-COST-FUNCTION

# Implementation: Representation




- A node is a bookkeeping data structure from which a search tree is constructed. It contains information such as: state, parent node, action, path cost *g(x)*, depth
- A state is the configuration in the state space to which the node corresponds
- The **`Successor-Fn(x)`** (successor function) returns all states that can be reached from state x

# Implementation
# Node Representation

Datatype NODE

- Components:
  - STATE,
  - PARENT-NODE,
  - OPERATOR,
  - DEPTH,
  - PATH-COST

- State
  - This represents the state in the state space to which a node corresponds
- Parent-Node
  - This points to the node that generated this node. In a data structure representing a tree it is usual to call this the parent node
- Operator
  - The operator that was applied to generate this node
- Depth
  - The number of nodes from the root
- Path-Cost
  - The path cost from the initial state to this node

# How Good is a Solution?

- Does our search method actually find a solution?
- Is it a good solution?
  - Path Cost
  - Search Cost (Time and Memory)
- Does it find the optimal solution?
  - But what is optimal?

# Evaluating a Search

- Completeness
  - Is the strategy guaranteed to find a solution?
- Time Complexity
  - How long does it take to find a solution?
- Space Complexity
  - How much memory does it take to perform the search?
- Optimality
  - Does the strategy find the optimal solution where there are several solutions?

# Summary

- Problem formulation
  - goal(s), operator(s)
- Differences between
  - state space vs. search tree
- Representation
  - problem
  - states vs. nodes
- Evaluation criteria for search
  - completeness, time/space complexity, optimality

# Acknowledgements

Most of the lecture slides are
adapted from the same module
taught in Nottingham UK
by
Professor Graham Kendall,
Dr. Rong Qu
and
Dr. Andrew Parker