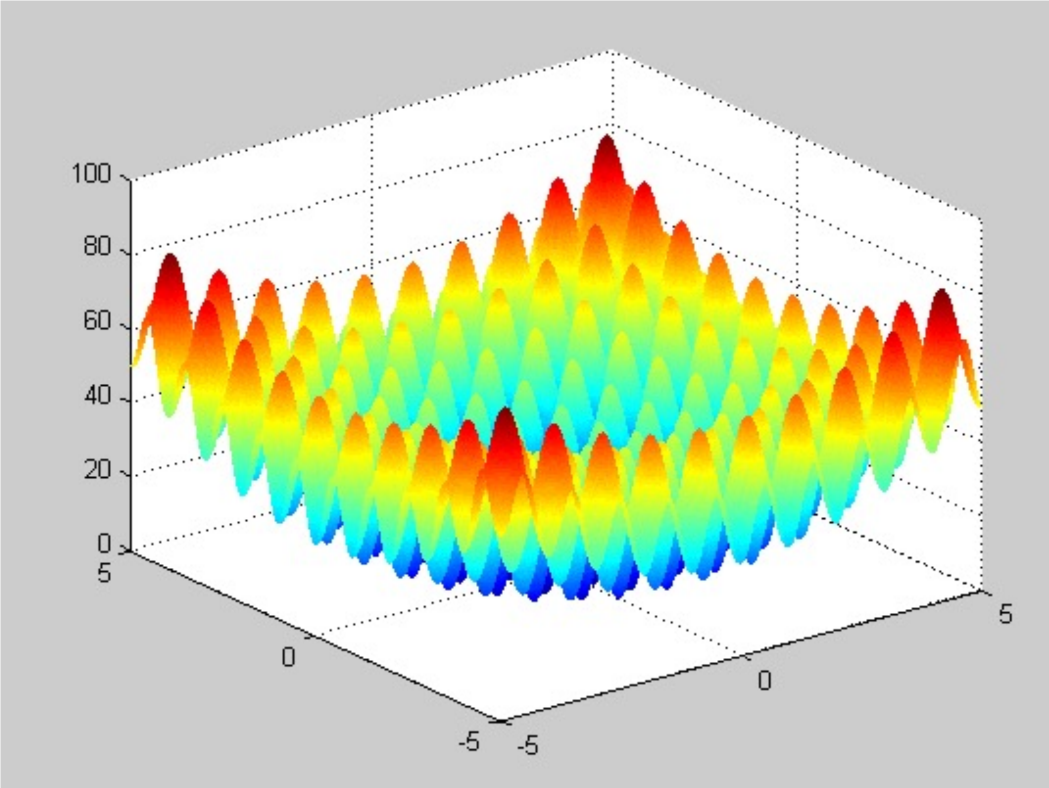
PROBLEM FORMULATION AND SEARCH TREE FUNDAMENTALS OF AI(COMP1037)

Dr Qian Zhang
University of Nottingham,
Ningbo China 2024

MATHEMATIC PROBLEM

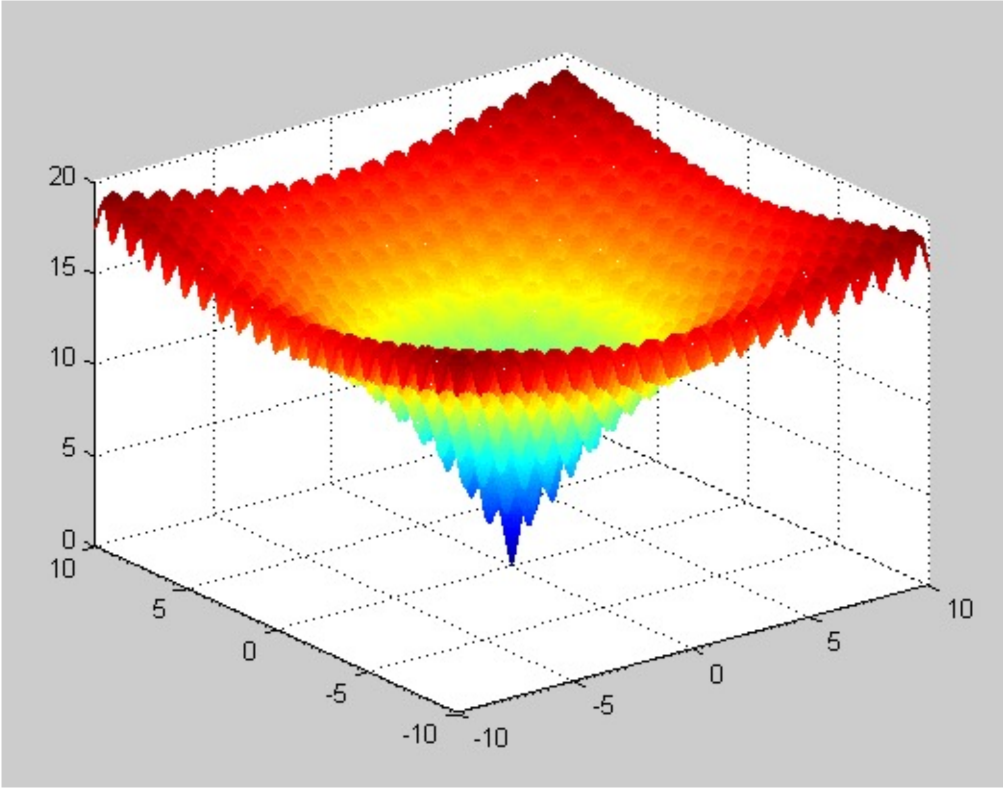
$$Ras(x) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2).$$

$$x_i \in [-5.12, 5.12]$$



$$F(\vec{x}) = -20 \cdot \exp\left(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi \cdot x_i)\right) + 20 + e$$

$$x_i \in [-30, 30]$$

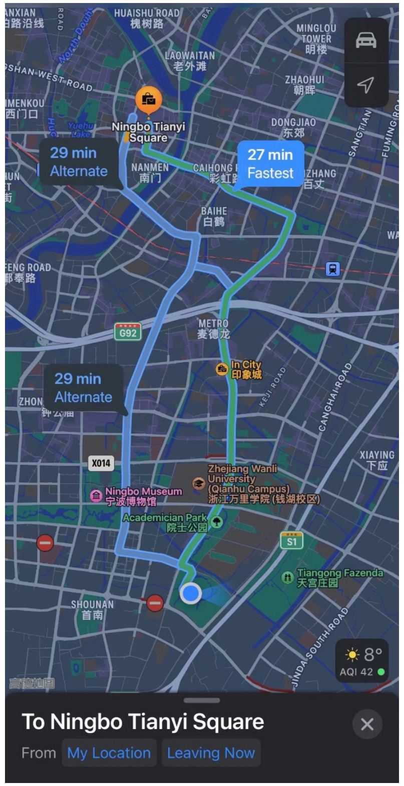


REAL-WORLD PROBLEM

Tree SEARCH

- Do you drive? Have you thought about how the route is planed in your GPS?

- How would you implement a cross-and-nought computer program?



SOLVING PROBLEM BY SEARCHING

- Many problems in real-world exhibit no detectable regular structure to be exploited, they appear "**chaotic**", and do not yield to efficient algorithms
- Often we can't simply write down and solve the equations for a problem

$$Ras(x) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2).$$

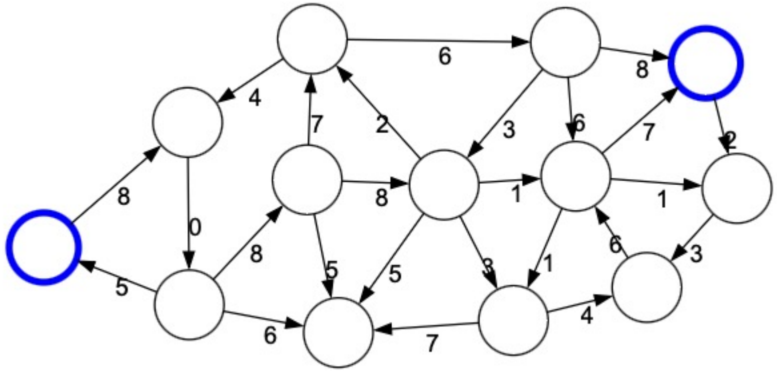
- Exhaustive search of large state spaces appears to be the only viable approach

WHAT TO LEARN? (LEC 1)

Real world route finding problem

Modeling

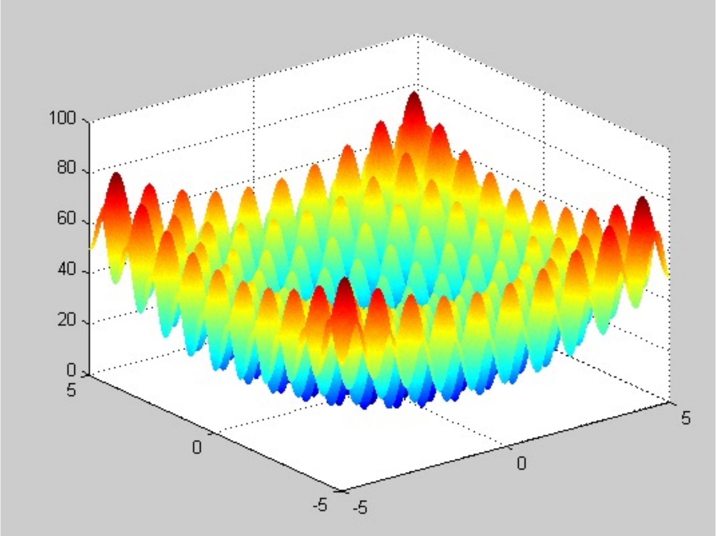
Model



Go through **possible solutions** and find the **optimum one.**

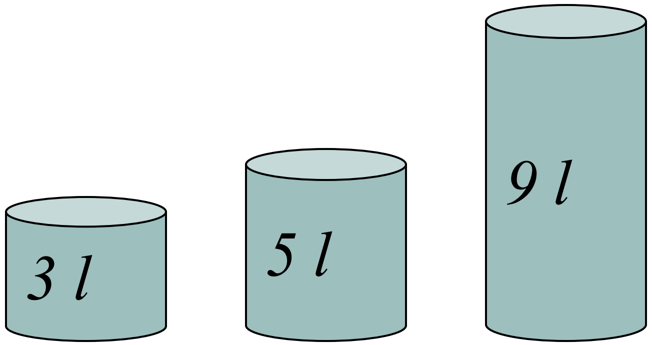
SOLVING PROBLEM BY SEARCHING

- The concept of **search** plays an important role in science and engineering
- Any problem can be seen as a search for "the **right answer**"
- **State space**
  - the set of all possible states that a problem domain can have
- Search **algorithms**
  - Take a problem as input
  - Return a solution to the problem



OUTLINES

- Topic one: Definitions and examples of problems (2nd Session)
  - **Problem formulation**
  - Problem representation
- Topic two: Problem solving by searching
  - Uninformed(blind) search algorithms (3rd Session)
    - Simplest exhaustive search
    - Breadth first search, depth first search, Uniform cost search
  - Informed search algorithms (4th Session)
    - Use of heuristics that apply domain knowledge
    - A^* algorithm, Minimax algorithm

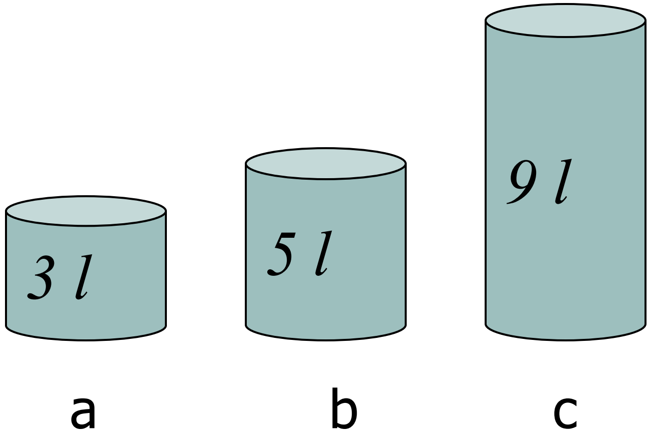


Problem: Using these three buckets, measure 7 liters of water.

Fill in the bucket? Pour out the water?

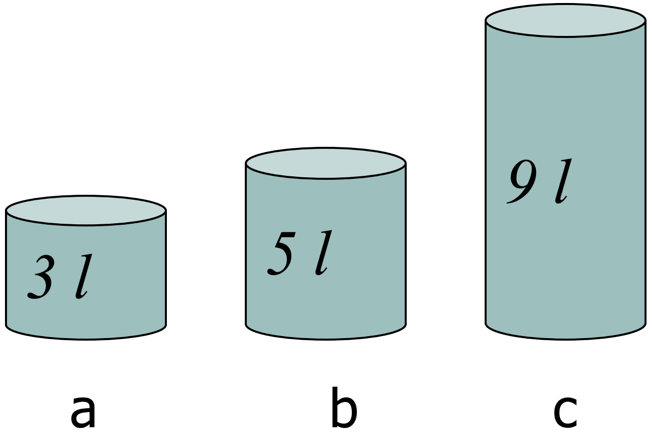
(one possible) Solution:

a	b	c	
0	0	0	start



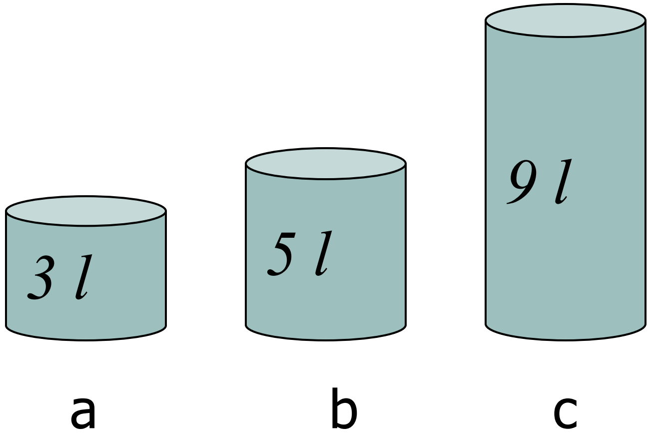
(one possible) Solution:

a	b	c	
0	0	0	start
3	0	0	



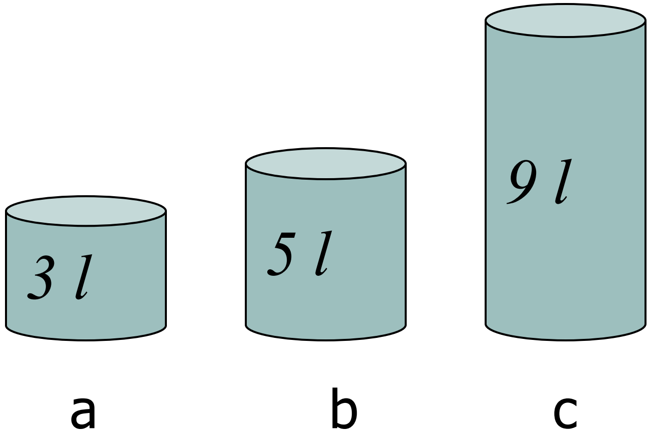
(one possible) Solution:

a	b	c	
0	0	0	start
3	0	0	
0	0	3	



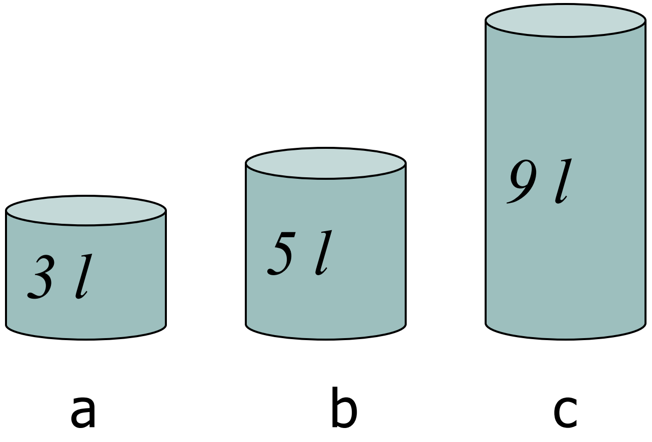
(one possible) Solution:

a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	



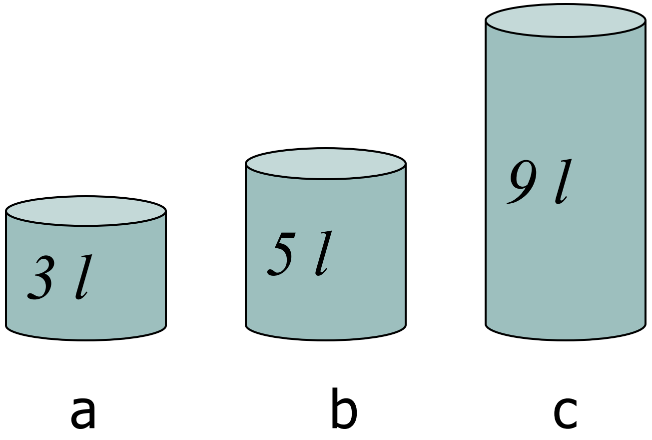
(one possible) Solution:

a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	
0	0	6	



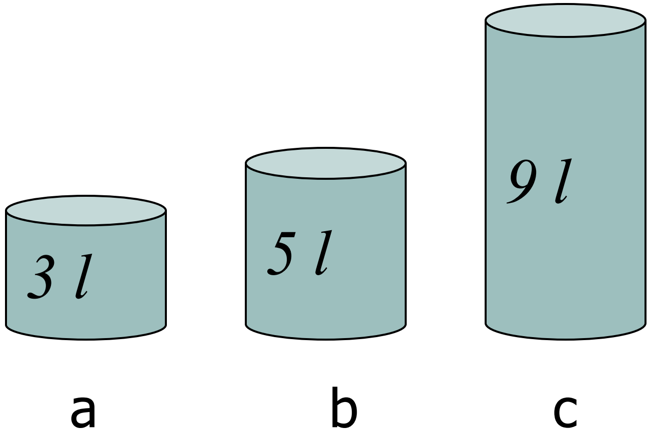
(one possible) Solution:

a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	
0	0	6	
3	0	6	



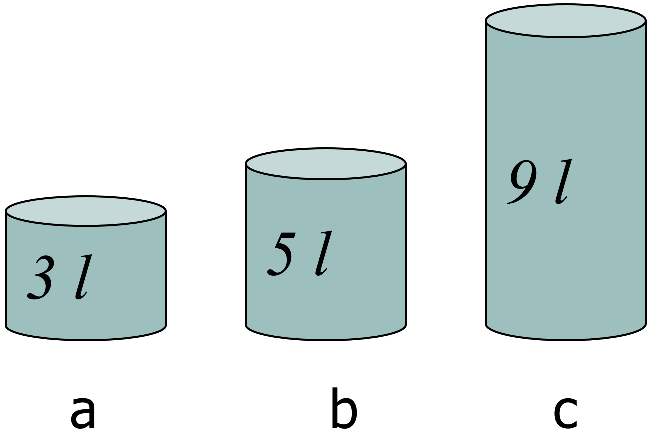
(one possible) Solution:

a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	
0	0	6	
3	0	6	
0	3	6	



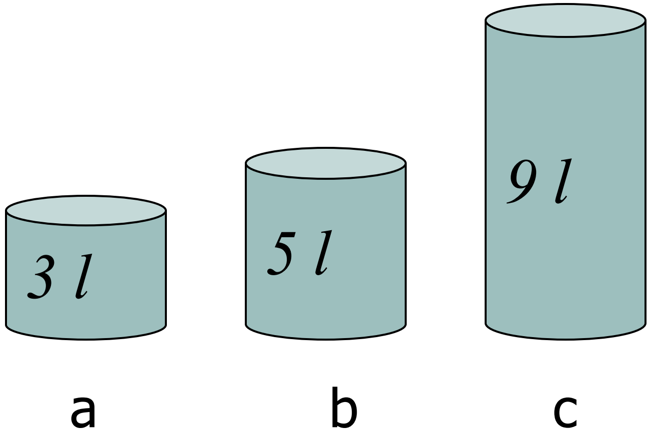
(one possible) Solution:

a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	
0	0	6	
3	0	6	
0	3	6	
3	3	6	



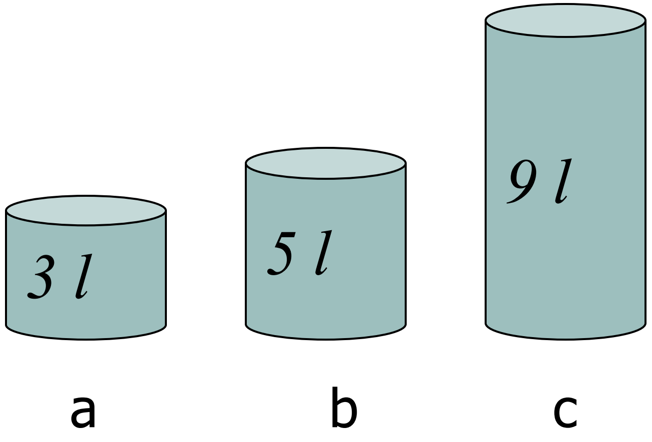
(one possible) Solution:

a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	
0	0	6	
3	0	6	
0	3	6	
3	3	6	
1	5	6	



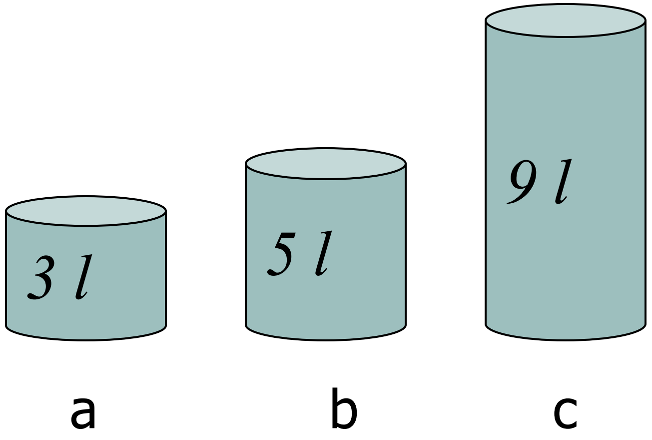
(one possible) Solution:

a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	
0	0	6	
3	0	6	
0	3	6	
3	3	6	
1	5	6	
0	5	**7**	**goal**



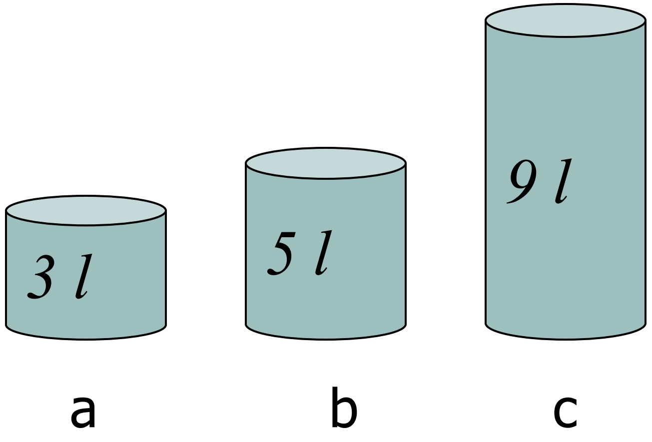
Another Solution:

a	b	c	
0	0	0	start
0	5	0	



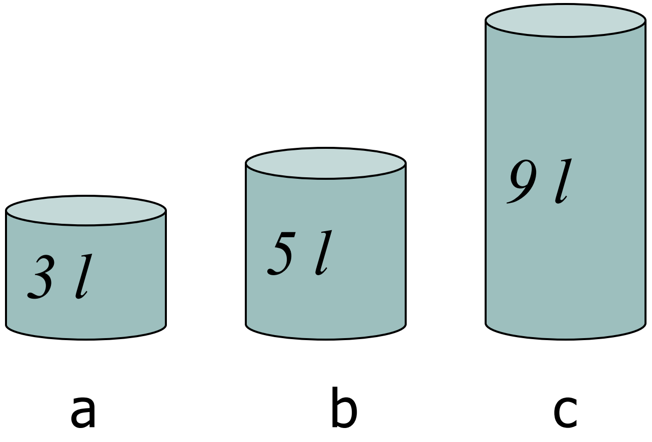
Another Solution:

a	b	c	
0	0	0	start
0	5	0	
3	2	0	



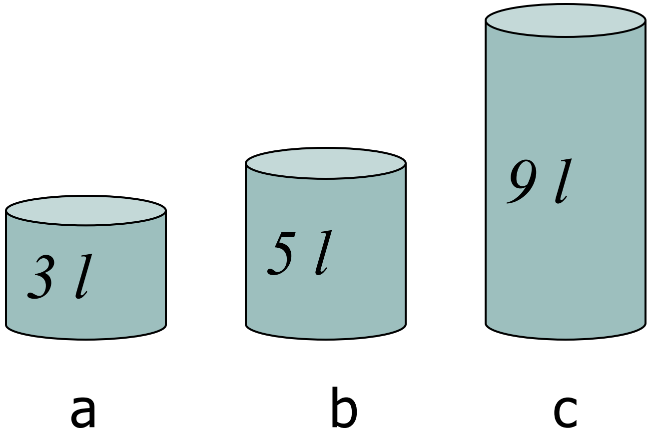
Another Solution:

a	b	c	
0	0	0	start
0	5	0	
3	2	0	
3	0	2	



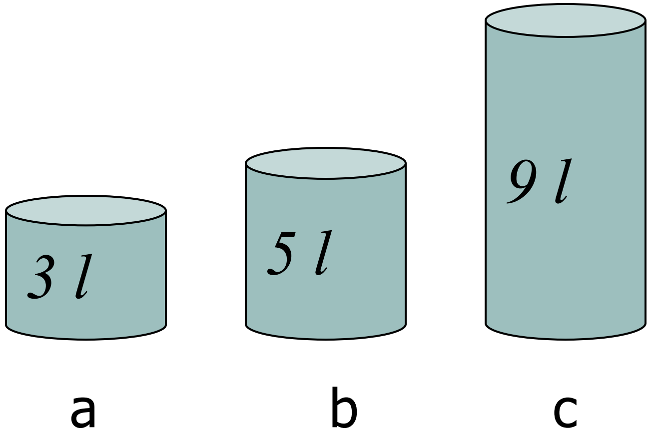
Another Solution:

a	b	c	
0	0	0	start
0	5	0	
3	2	0	
3	0	2	
3	5	2	



Another Solution:

a	b	c	
0	0	0	start
0	5	0	
3	2	0	
3	0	2	
3	5	2	
3	**0**	**7**	**goal**



- **Solution 1:**

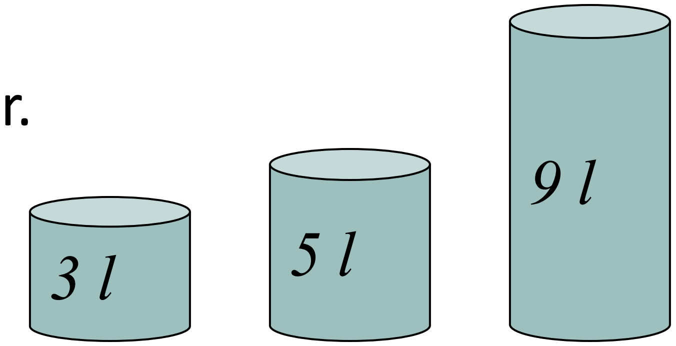
a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	
0	0	6	
3	0	6	
0	3	6	
3	3	6	
1	5	6	
0	5	**7**	**goal**

- **Solution 2:**

a	b	c	
0	0	0	start
0	5	0	
3	2	0	
3	0	2	
3	5	2	
3	**0**	**7**	**goal**

PROBLEM FORMULATION

Problem: Using these three buckets, measure 7 liters of water.



What is the environment?

What is the actions?

What does it mean by 'success'?

What is solution?

What to search?

Problem formulation is the process of deciding what actions and states to consider, given a goal.

PROBLEM COMPONENTS

- **Initial State**
  - The starting state of the problem, defined in a suitable manner
- **Actions (Operators)**
  - 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 S** by applying all legal actions is known as the *neighbourhood* (successors) and the the action(S) can be recognized as the **successor function.**

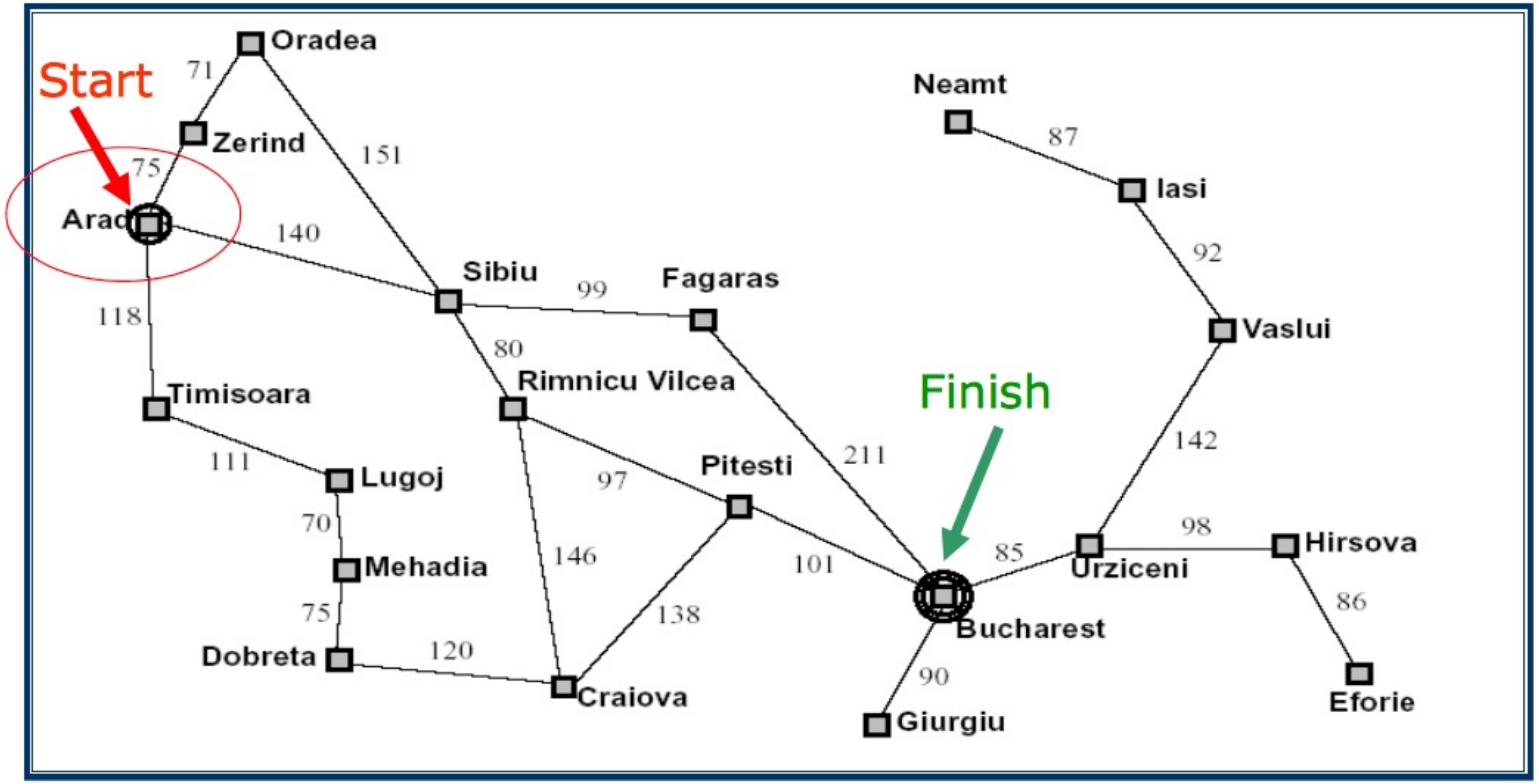
PROBLEM COMPONENTS

- **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
  - A *solution* to a problem is an *action sequence* that leads from the initial state to the goal state.

Note: The initial state and the successor function define the ***state space*** which is the set of all states reachable from the initial state (it forms a directed network or graph)

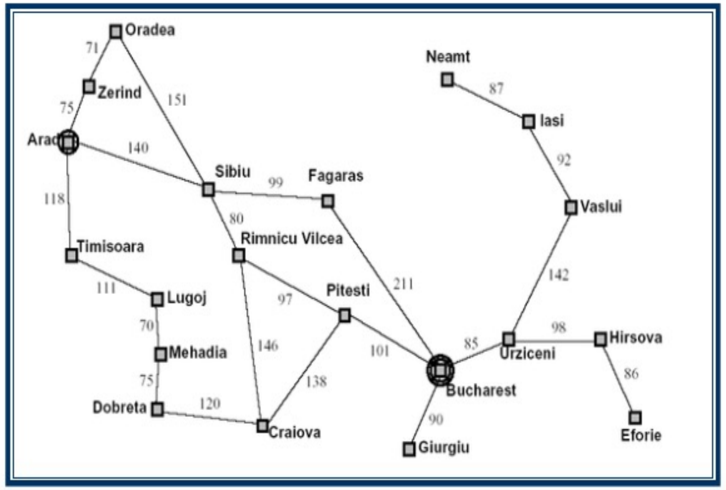
The complexity of a problem depends on the size of the state space.

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

http://mypuzzle.org/sliding

PROBLEM FORMULATION: 8-PUZZLE

- Initial State
  - specifies the location of each of the eight number 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
- Path Cost
  - each move of the blank costs 1

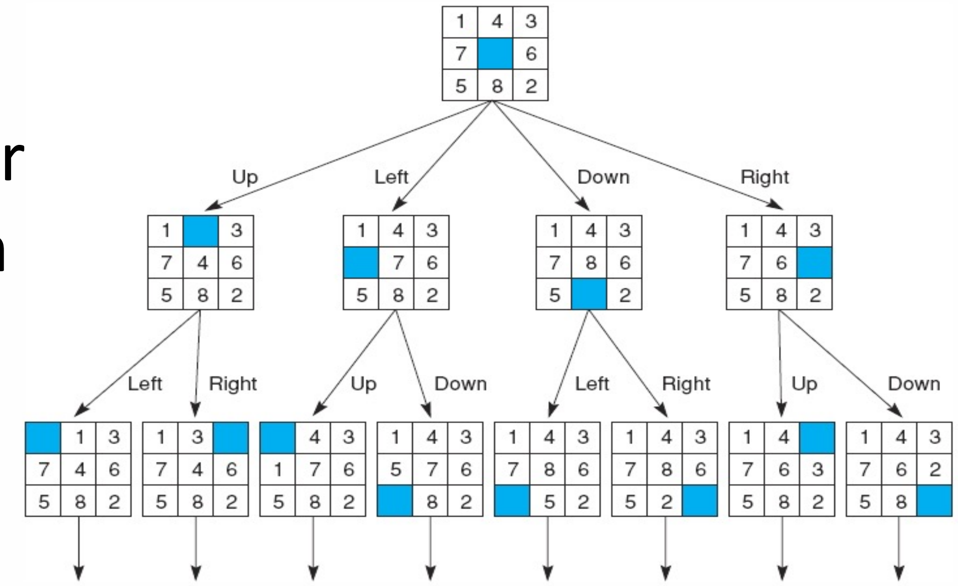
5	4	
6	1	8
7	3	2

1	4	7
2	5	8
3	6	

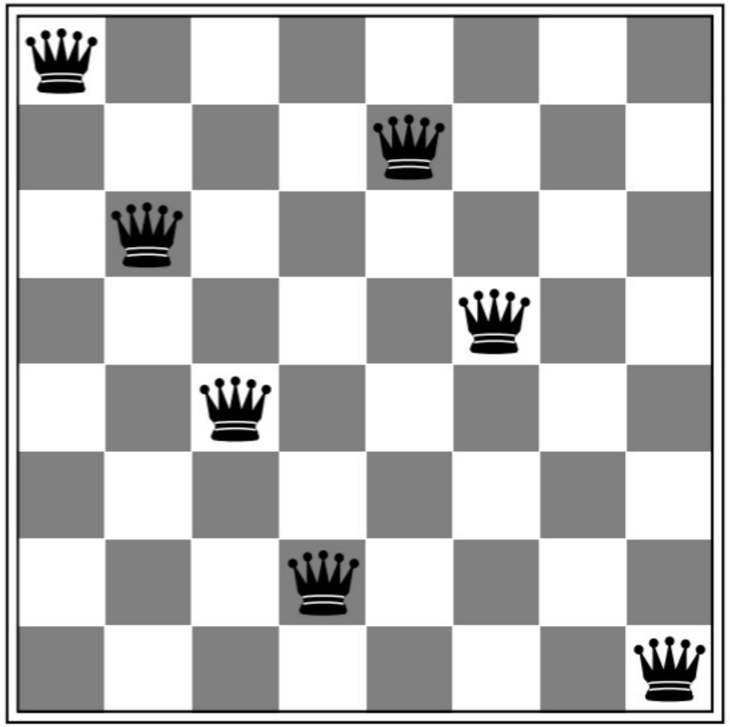
Q: How big is the state space?

PROBLEM FORMULATION: 8-PUZZLE

- The number of actions/operators depends on how they are formulated
  - 4 possible moves could be specified for each of the 8 number tiles, resulting in a total of **4*8=32** operators.
  - On the other hand, **4** moves for the "blank" tile could be specified instead, so only 4 operators are needed.
  - => Formulation shift can greatly simplify a problem!



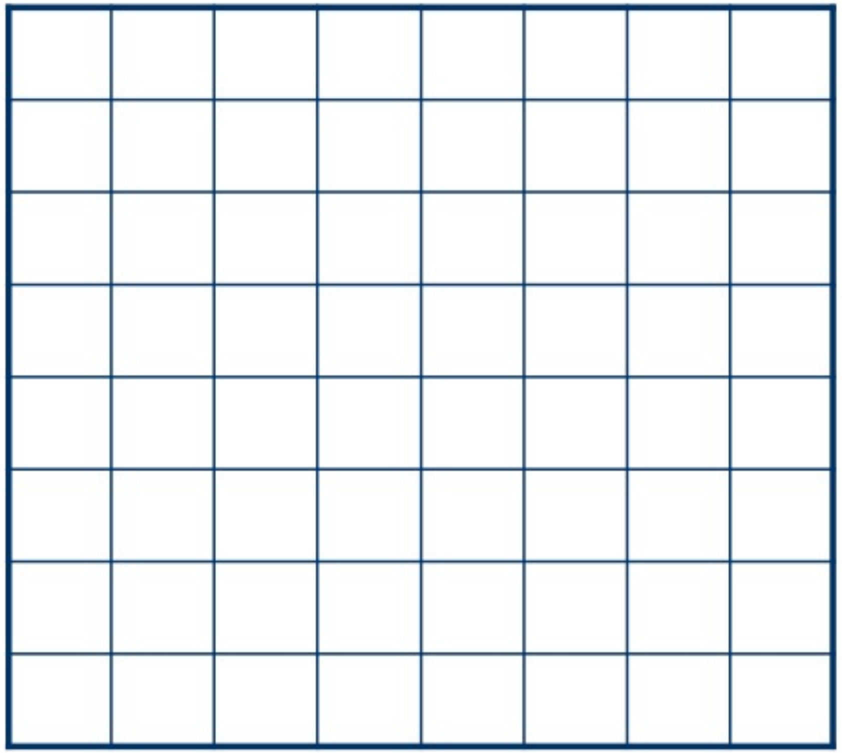
PROBLEM FORMULATION: 8-QUEEN

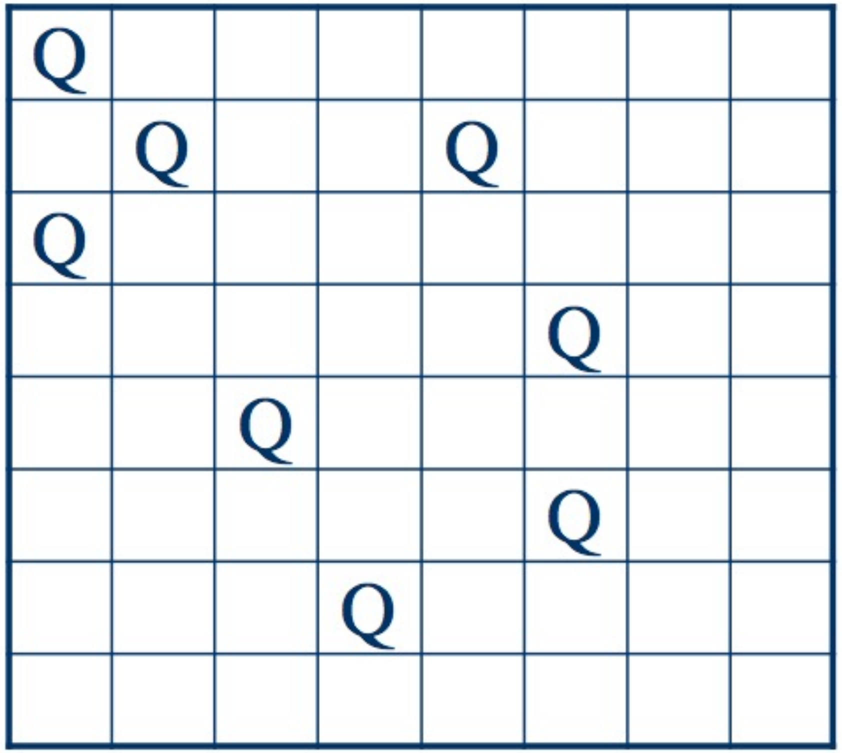


- Initial state
  - An empty 8X8 board
- Operator
  - Add a queen into a cell on the board
- States
  - Any arrangement of 0 to 8 queens on the board
- Goal state
  - A valid arrangement of 8 queens on board (none attacked)

PROBLEM FORMULATION:8-QUEEN

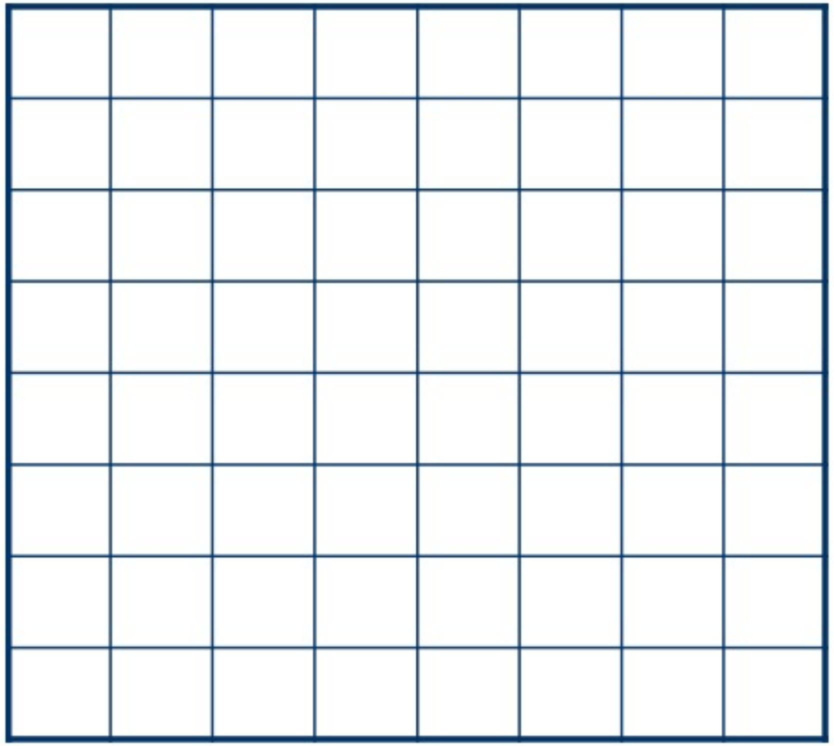
- 1st Formulation: add a queen to any empty square

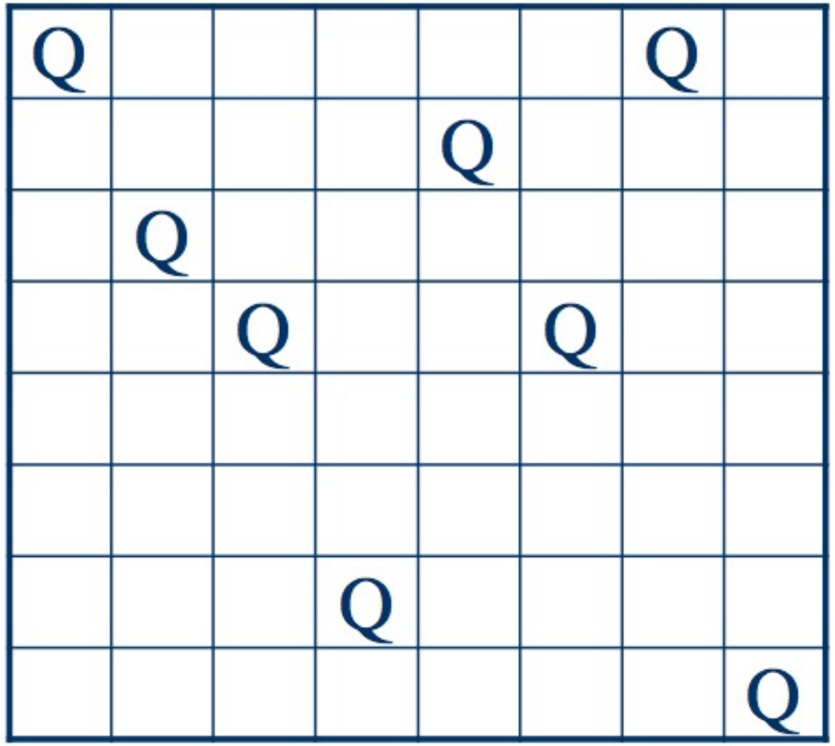




PROBLEM FORMULATION:8-QUEEN

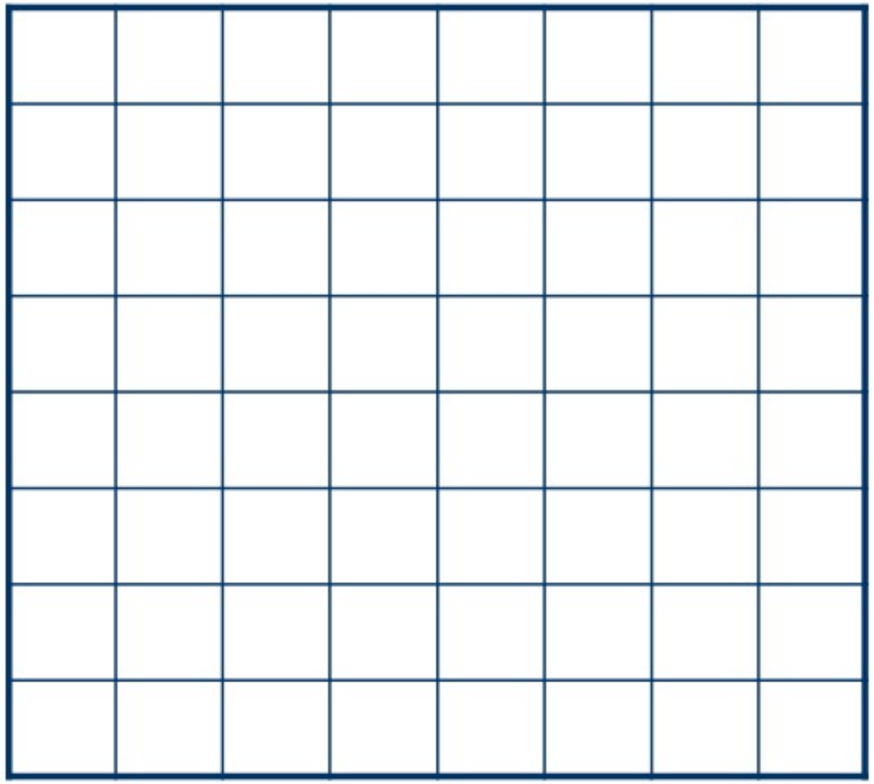
- 2nd Formulation: add a queen to any square in the leftmost empty column

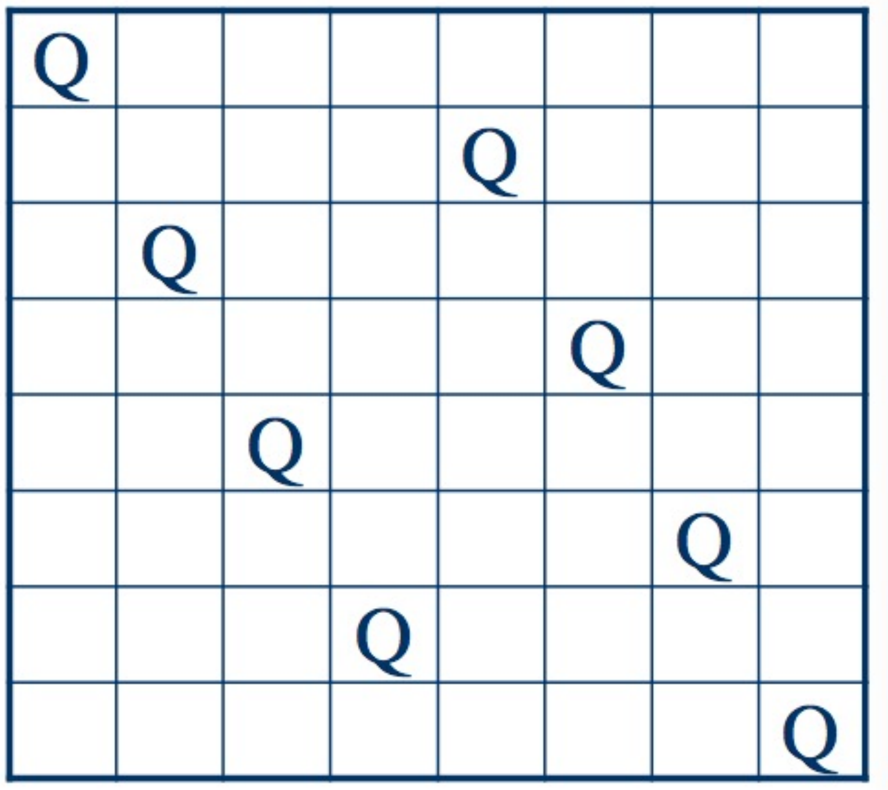




PROBLEM FORMULATION:8-QUEEN

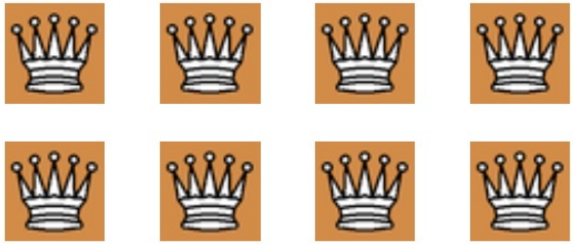
- 3rd Formulation: add a queen to any square in the leftmost empty column such that it is not attacked by any other queen

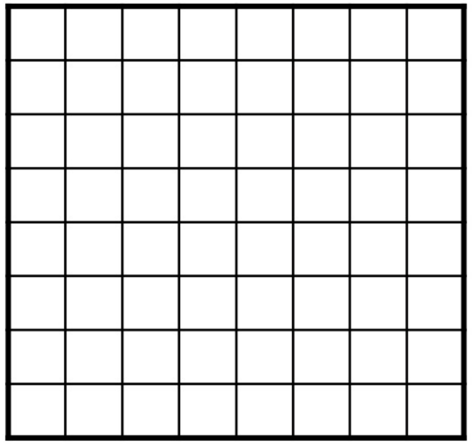




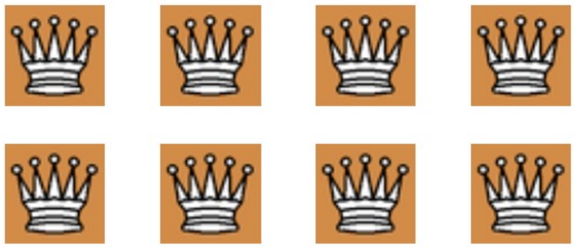
PROBLEM FORMULATION: 8-QUEEN (1^{ST} FORMULATION)

- 1st level: 1 root node (empty board)
- 2^{nd} level: **64** nodes
- 3^{rd} level: **63** nodes for each of the node on the 2^{nd} level
- …

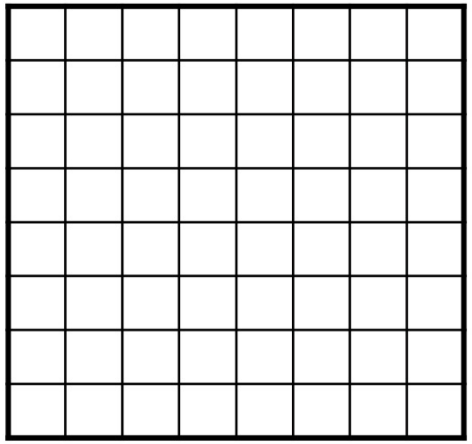




PROBLEM FORMULATION: 8-QUEEN (2^{ND} FORMULATION)



- 1st level: 1 root node (empty board)
- 2^{nd} level: **8** nodes
- 3^{rd} level: **8** nodes for each of the node on the 2^{nd} level
- …
- Smaller tree

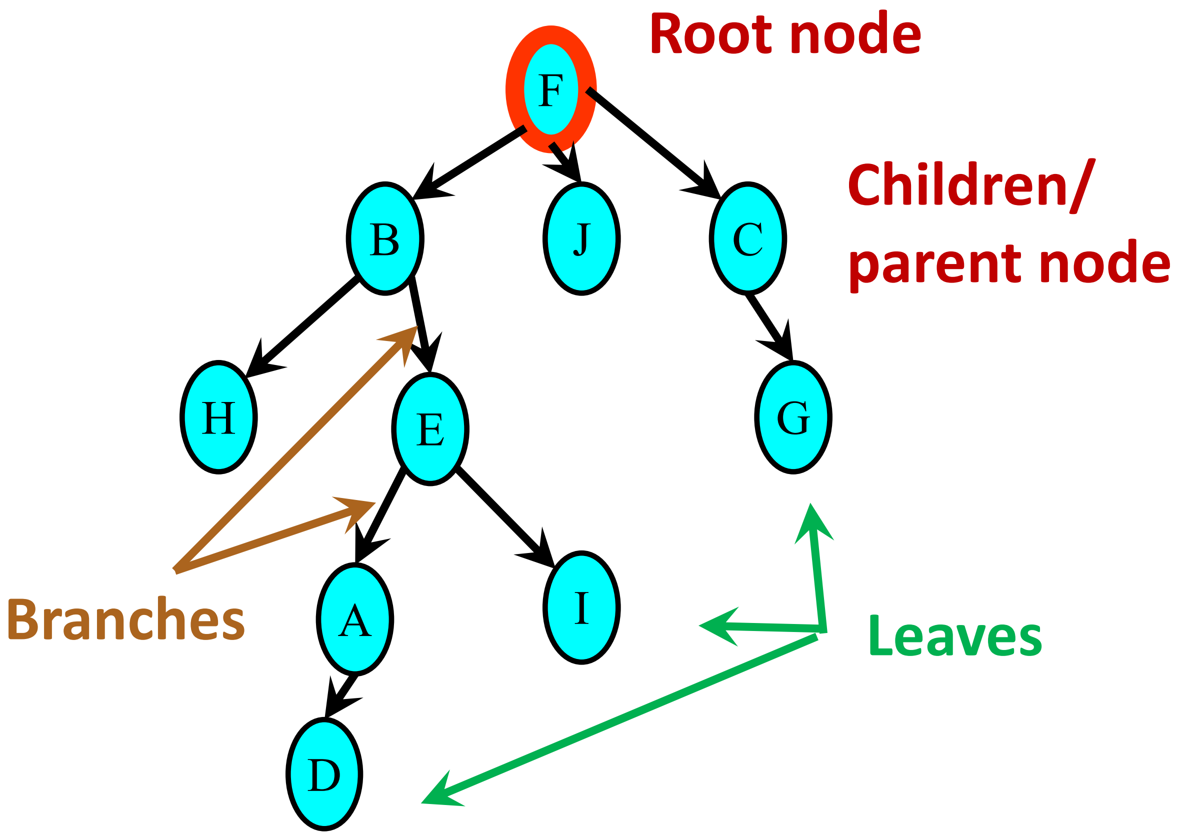


OUTLINES

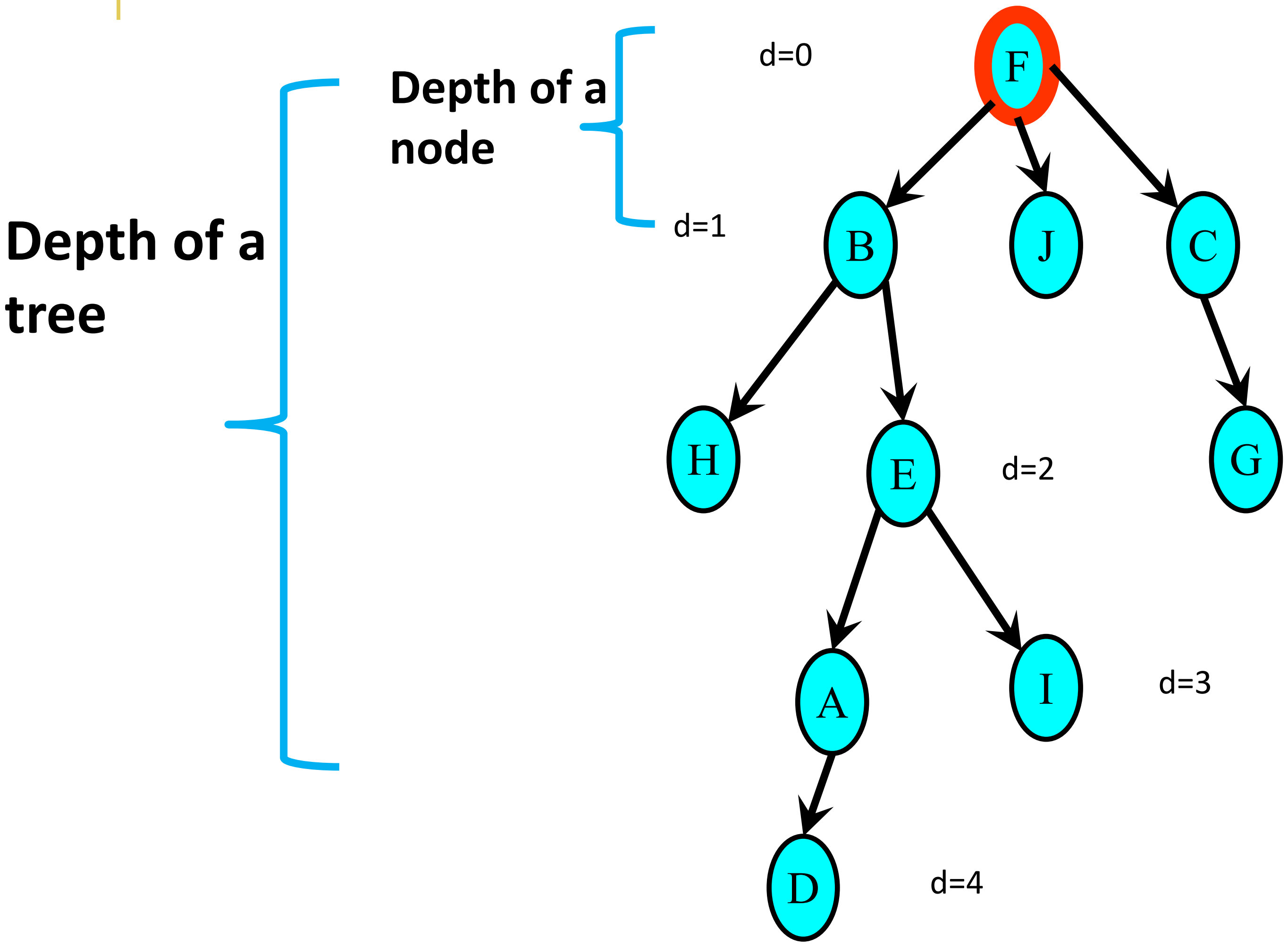
- Topic one: Definitions and examples of problems (2nd Session)
  - Problem formulation
  - **Problem representation**
- Topic two: Problem solving by searching
  - Uninformed(blind) search algorithms (3rd Session)
    - Simplest exhaustive search
    - Breadth first search, depth first search, Uniform cost search
  - Informed search algorithms (4th Session)
    - Use of heuristics that apply domain knowledge
    - A^* algorithm, Minimax algorithm

TREES - TERMINOLOGY

- Nodes
  - Root node
  - Children/parent of nodes
  - Leaves
- Branches
- **Average branching factor**
  - average number of branches of the nodes in the tree

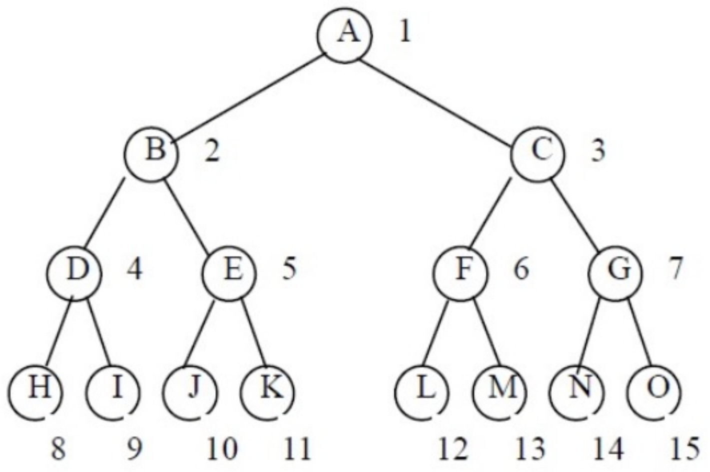


TREES – TERMINOLOGY (2)



- The depth, d, of a node is just the number of edges (branches) away from the root node
- The depth of a tree is the depth of the deepest node
  - in this case, depth=4

TREE SIZE



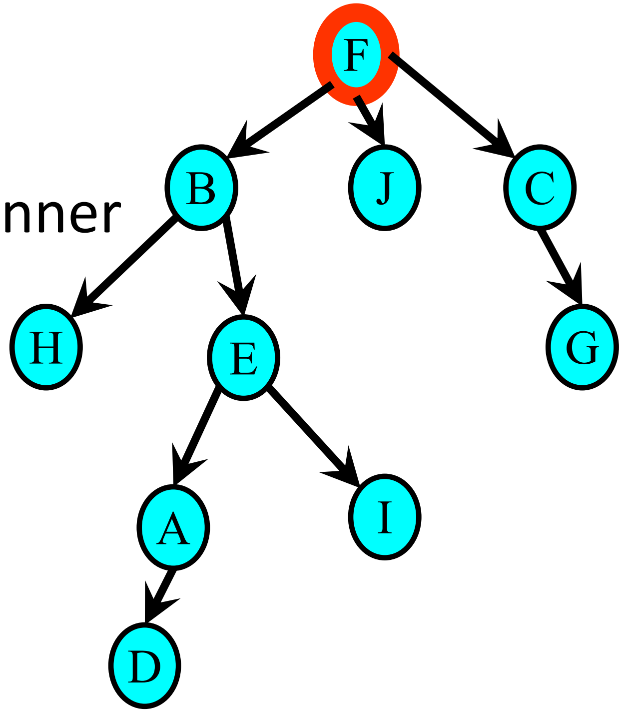
- Branching factor b=2 (binary tree)

Depth d	# Nodes at d = 2^d	# Nodes in a tree = $2^{d+1}-1$
0	1	1
1	2	3
2	4	7
3	8	15
4	16	31
5	32	63

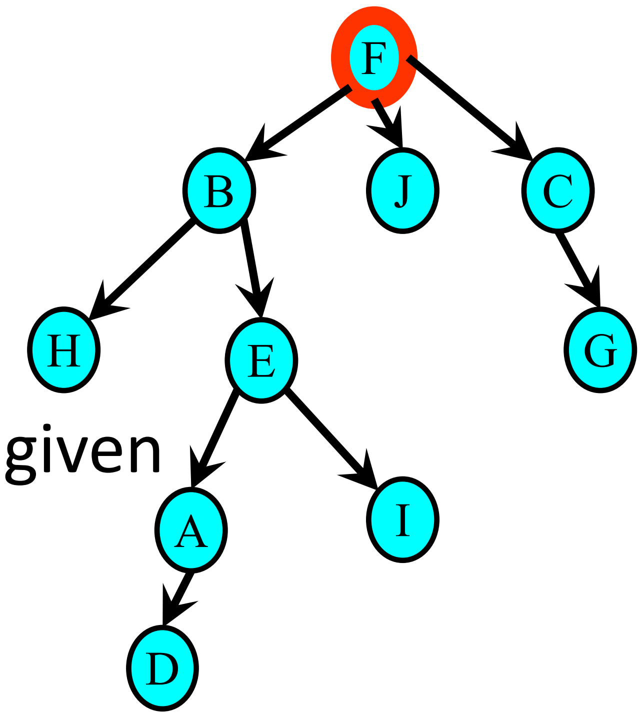
- Why we discuss the size of the tree?

PROBLEM REPRESENTATION

- States – nodes
  - Possible states of problem, defined in some suitable manner
- Initial State - root node
  - The starting state of the problem
- State Space – all nodes in the tree
  - The set of all states reachable from the initial state

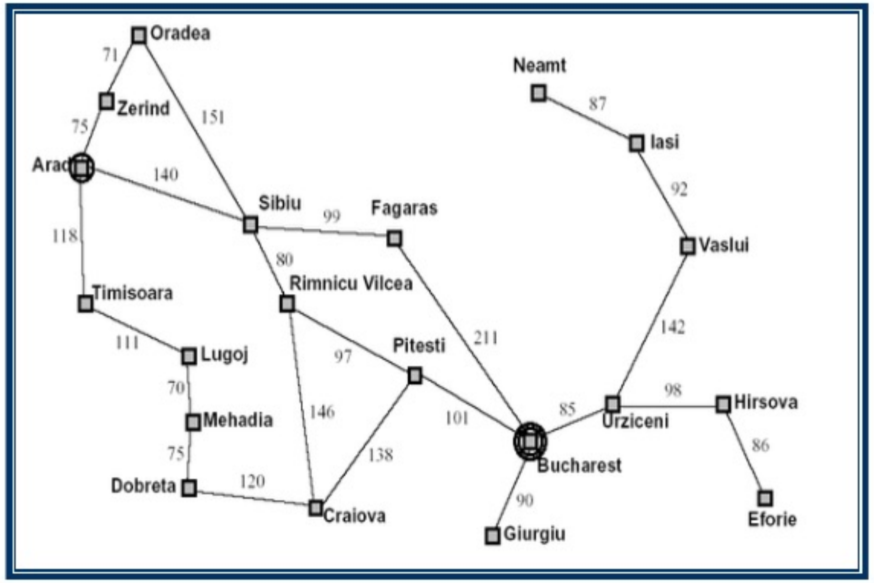


PROBLEM REPRESENTATION

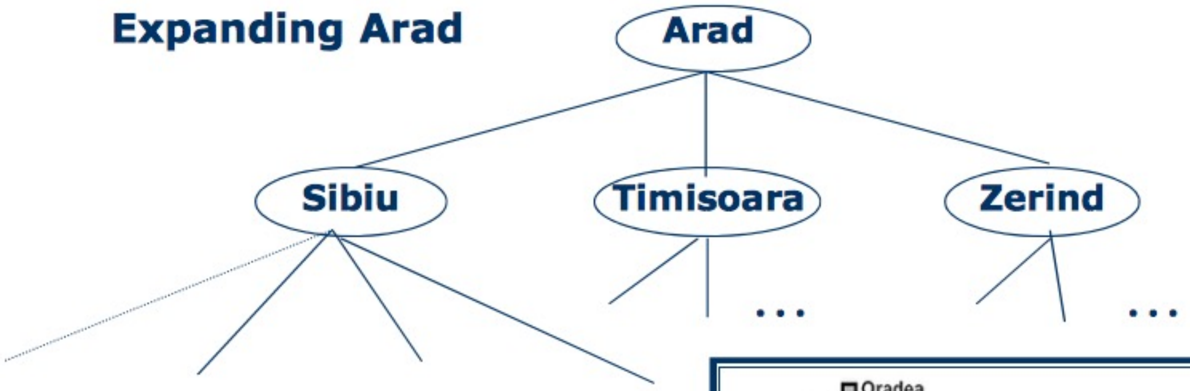


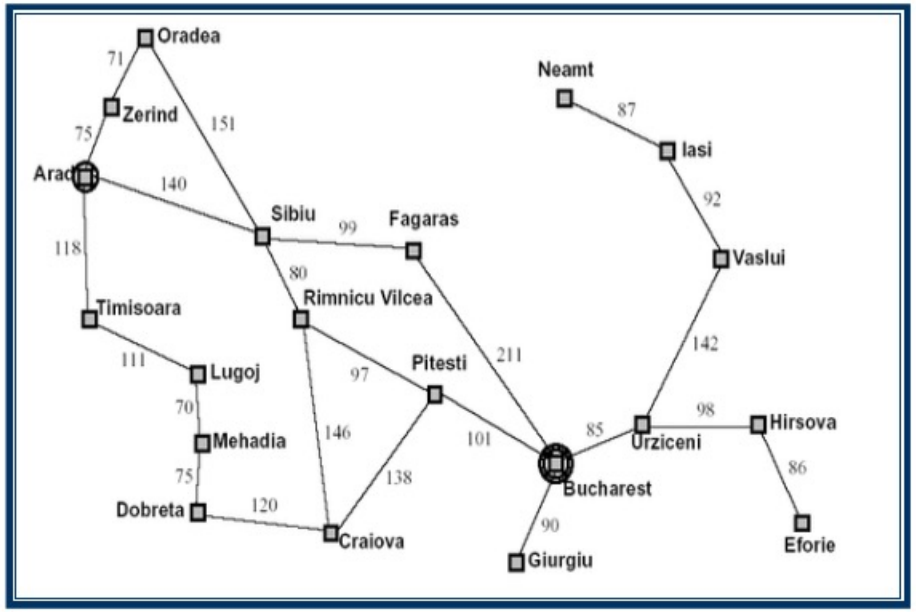
- Successors - Neighborhood
  - The set of all possible states reachable from a given state
  - Branching factor: number of neighborhoods
- Operator(s) – branches
  - A set of actions that moves the problem from one state to another

SEARCH TREE – ROMANIA

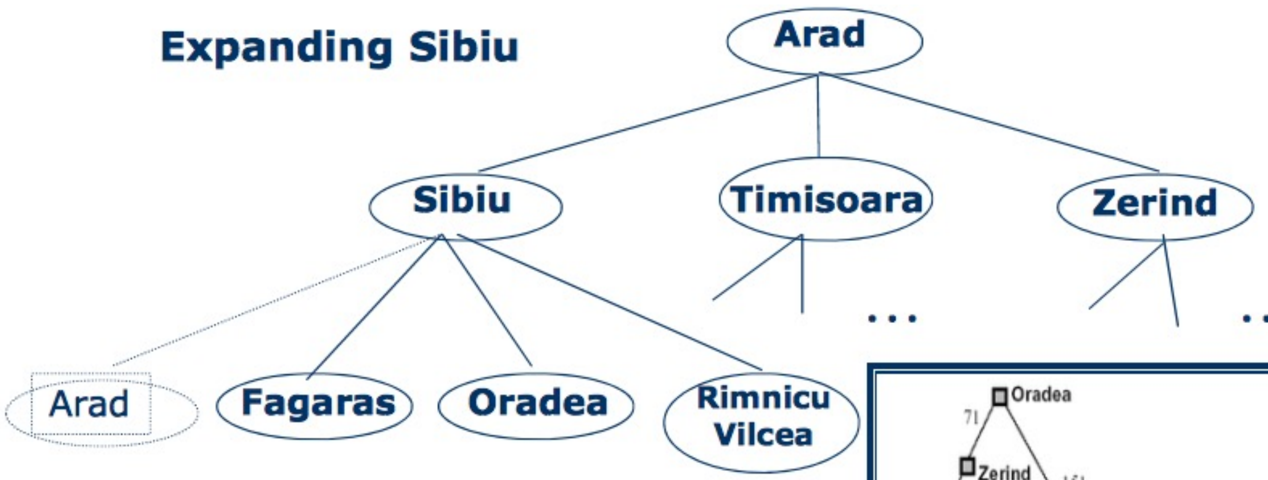


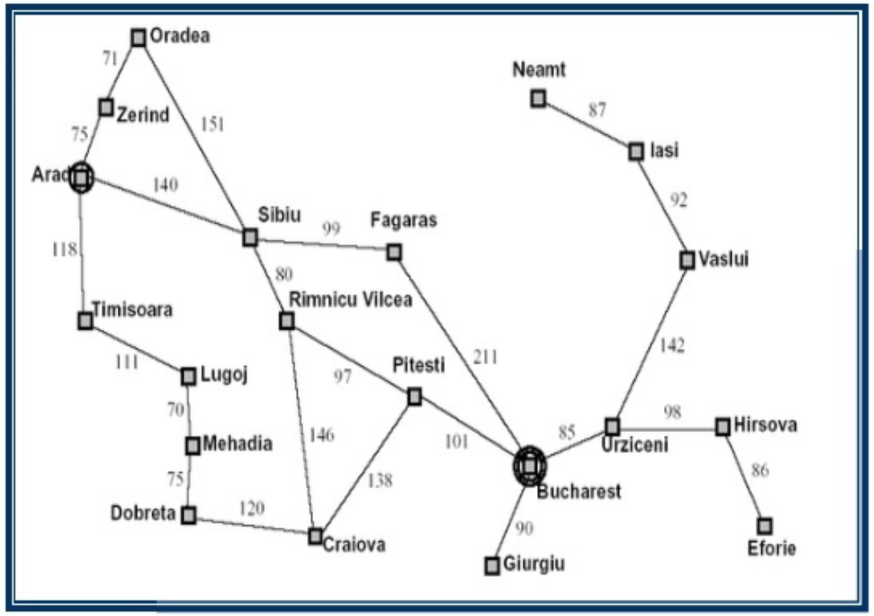
SEARCH TREE – ROMANIA



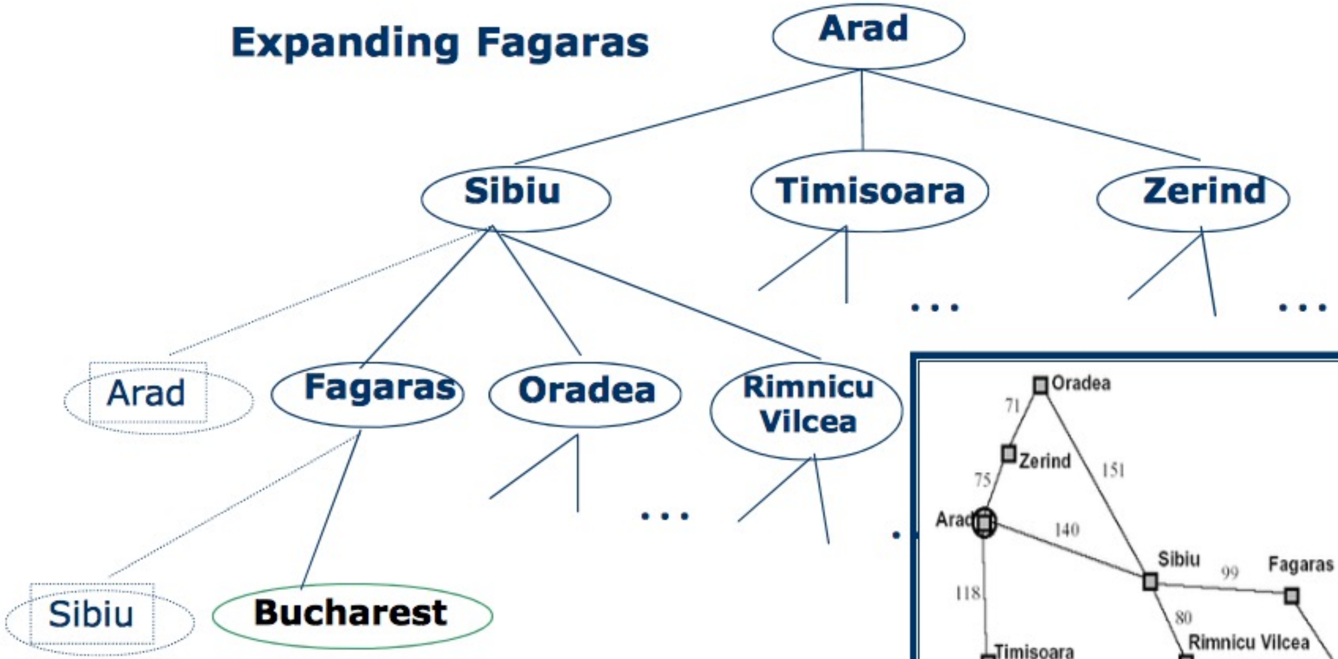


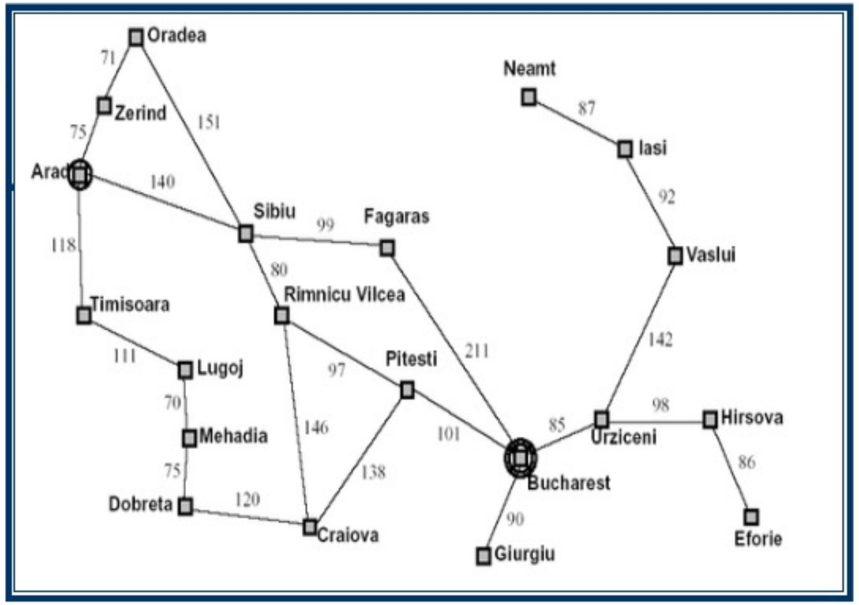
SEARCH TREE – ROMANIA





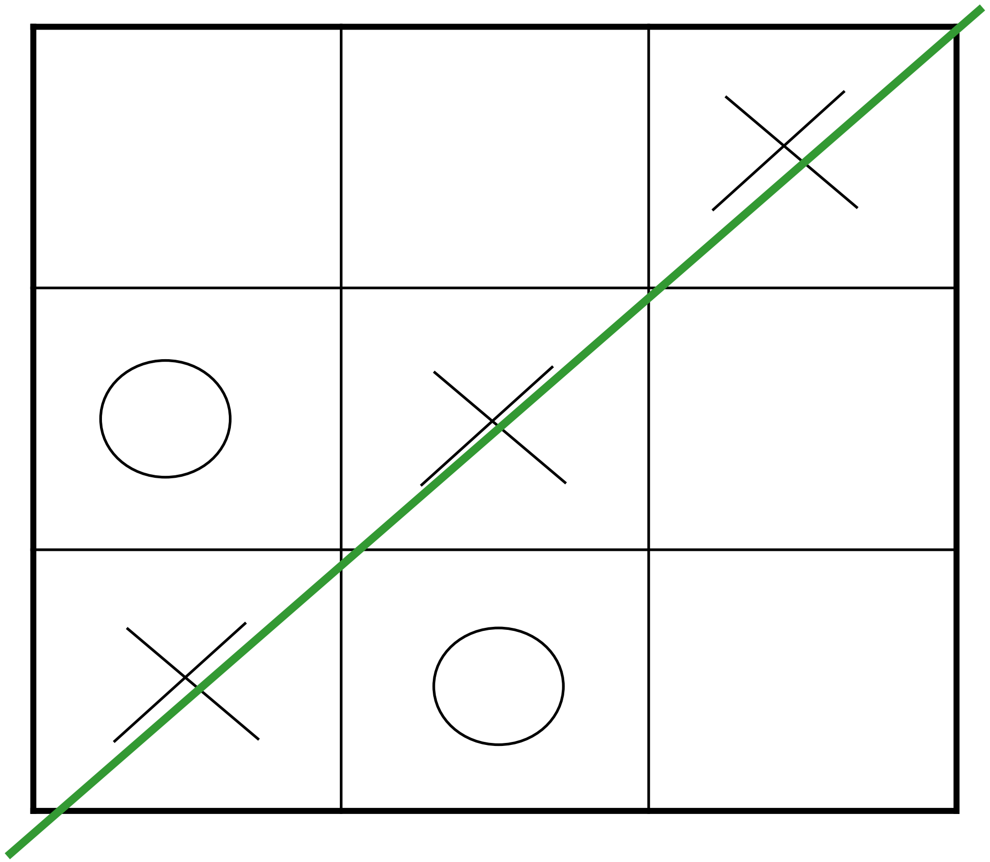
SEARCH TREE – ROMANIA





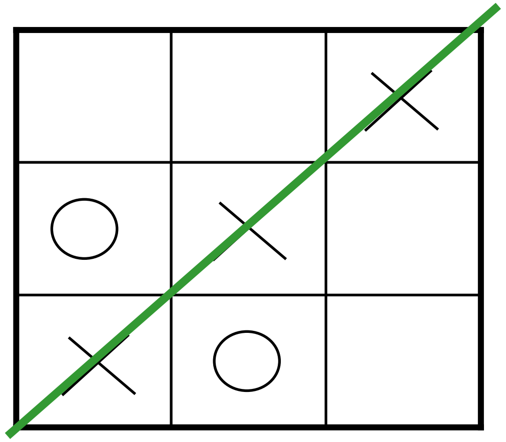
SEARCH TREE – TIC-TAC-TOE

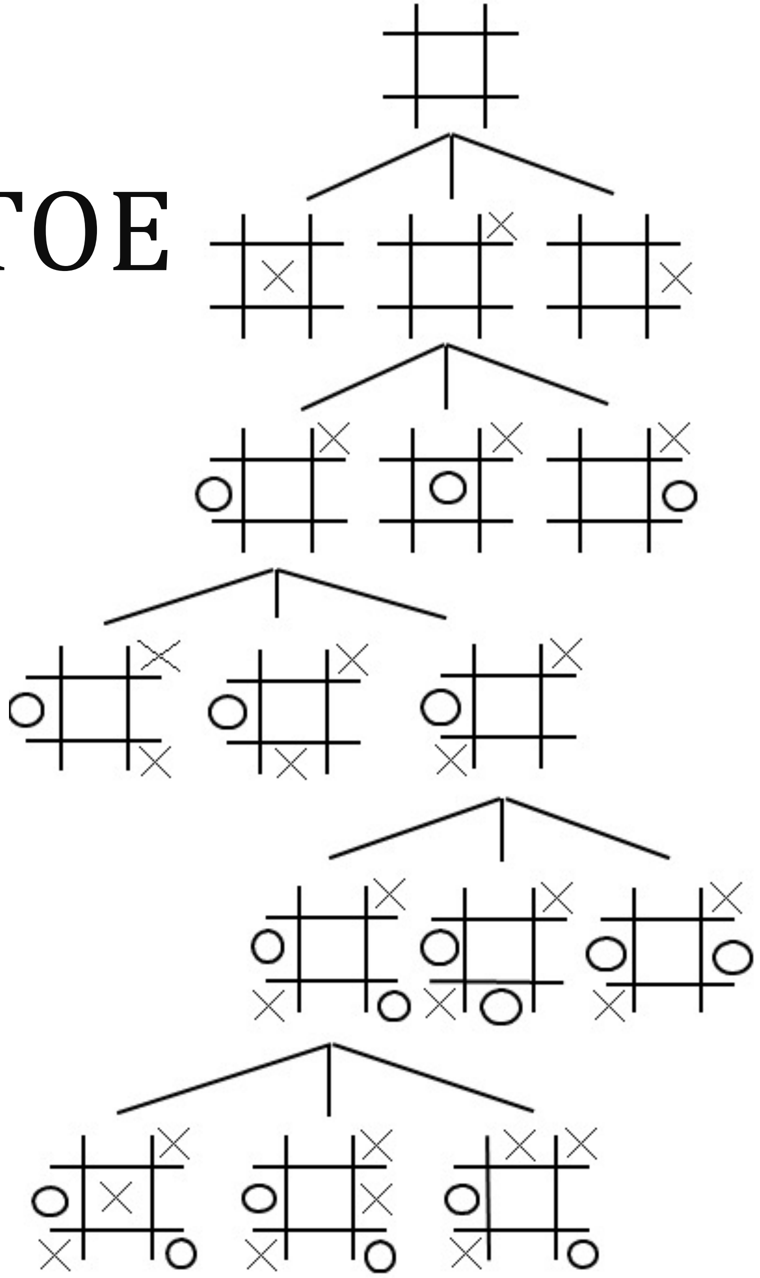
- Noughts and Crosses (Tic-Tac-Toe)



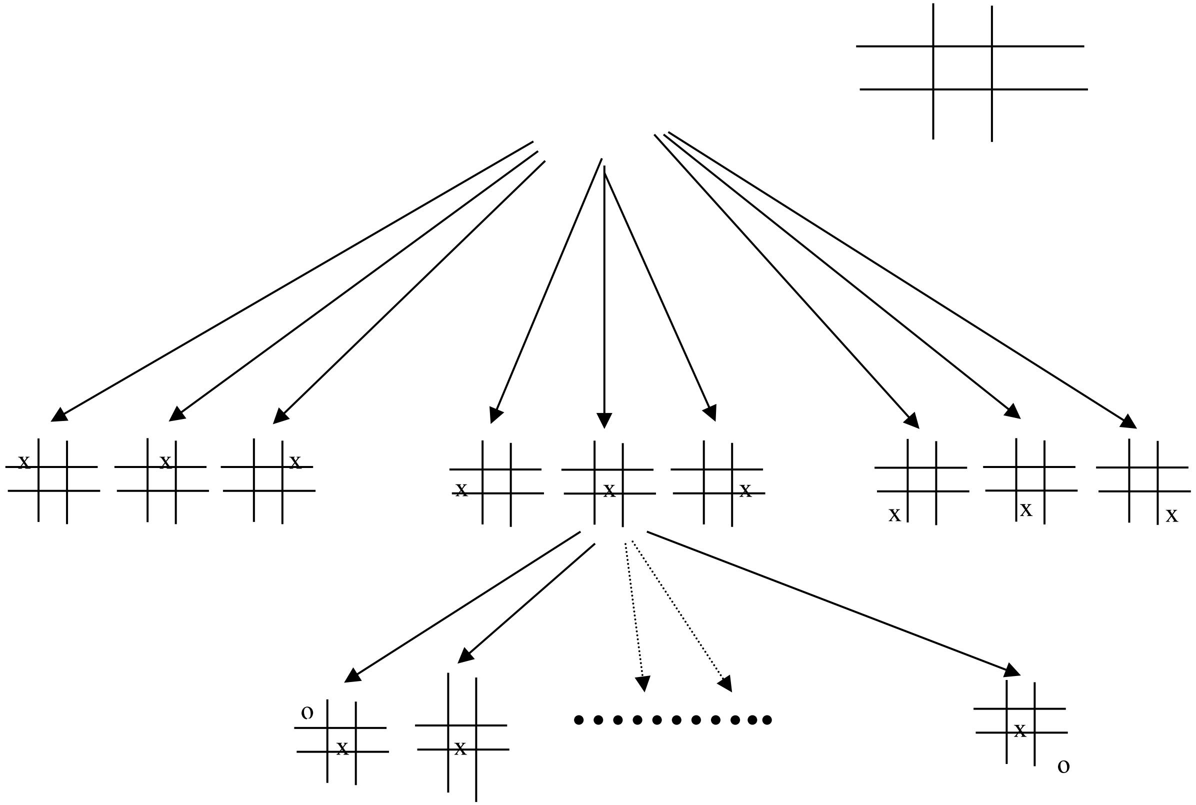
SEARCH TREE – TIC-TAC-TOE

- Nodes: states of problem
- Root node: initial state of the problem
- Branches: moves by operator
- Branching factor: number of neighbours



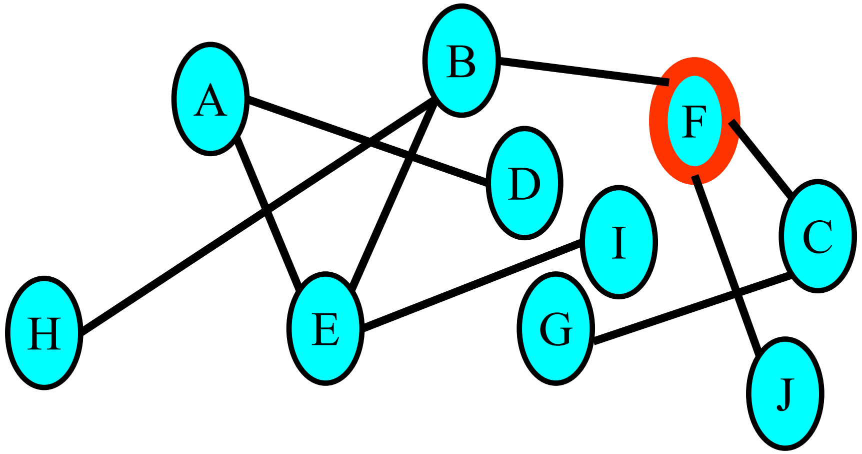


FULL SEARCH TREE- TIC-TAC-TOE



FINDING GOALS IN TREES

- Does the following tree contain a node "I"?
- Easily read from the graph



- so why the big deal about search?

FINDING GOALS IN TREES: REALITY

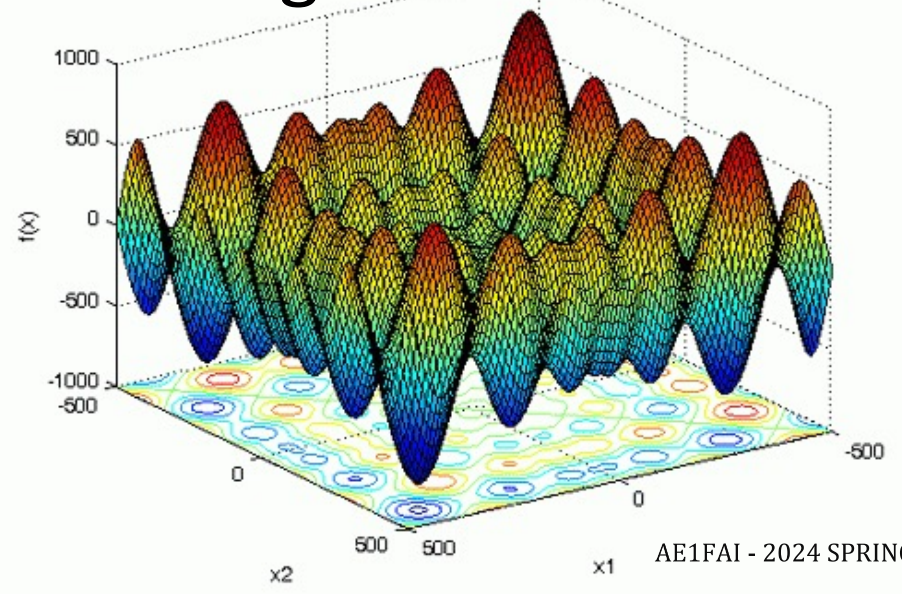
- Does the tree under the following root contain a node "G"?

- 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"

WHY IS GOAL SEARCH NOT TRIVIAL?

- Because the tree is not given as a pretty picture "on a piece of paper"

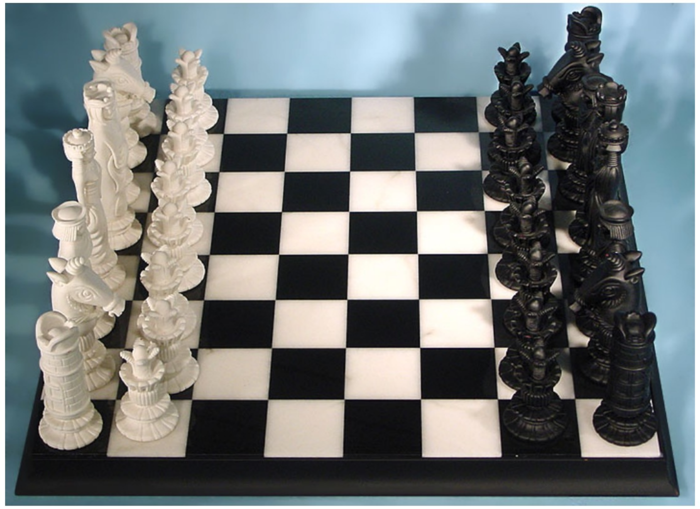
- At the start of the search, the search algorithm does not know
  - the size of the tree
  - the shape of the tree
  - the depth of the goal states



WHY IS GOAL SEARCH NOT TRIVIAL?

- 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:
  - $b=2, d=10$: $b^d = 2^{10} = 1024$
  - $b = 10$, $d = 10$: $b^d = 10^{10} = 10{,}000{,}000{,}000$

SEARCH TREE ISSUES



- Search trees grow very quickly
- The size of the search tree is governed by the **branching factor**
- Even the simple game with branching factor of **3** has a complete search tree of large number of potential nodes
- The search tree for chess has a branching factor of about 35

SEARCH TREE ISSUES

- **Claude Shannon** delivered a paper in 1949 at a New York conference on how a computer could play chess.
- Chess has 10^{120} unique games (based an average of 40 moves - the average length of a master game).
- Working at 200 million positions per second, **Deep Blue** would require 10^{100} years to evaluate all possible games.
- 10^{100} is larger than the number of atoms in the universe.

SUMMARY

- Problem formulation
- Representation
  - Problem
  - State space
  - Search tree
- Properties of search

FURTHER READING

- AIMA Chapter 3.1-3.2
- Self study slides: fundamental issues in AI
- Next week: AIMA Chapter 3.3-3.4