Module - 3

Problem Solving by searching

Introduction to Problem space

- Problem solving is the major area of concern in Artificial Intelligence.
- It is the process of generating solution from given observed data.
- To solve a particular problem, we need to build a system or a method which can generate required solution.
- Following four things are required for building such system.

1. Define the problem precisely. This definition must precisely specify the initial situation (input).
2. Analyse the problem. To identify those important features which can have an immense impact on the appropriateness
3. Isolate and represent the task knowledge that is necessary to solve the problem.
4. Choose the best problem solving technique and apply it to the particular problem.

- A problem is described formally as:

Define a state space that contains all the possible configurations of relevant objects.

Specify one or more states within that space that describe possible situations form which the problem solving process may start. These states are called initial states.

Specify one or more states that would be acceptable as solutions to the problem. These states are called goal states.

Specify a set of rules that describe the actions available.

- The problem can then be solved by using the rules, in combination with an appropriate control strategy, to move through the problem space until a path from an initial state to a goal state is found.
- This process is known as search.

State Space

- A **state** is a representation of problem elements at a given moment.
- **A State space is the set of all states reachable from the initial state.**
- A state space forms a graph in which the nodes are states and the arcs between nodes are actions.
- In the state space, a path is a sequence of states connected by a sequence of actions.
- The solution of a problem is part of the graph formed by the state space.
- **The state space representation forms the basis of most of the AI methods.**

Its structure corresponds to the structure of problem solving in two important ways:

1. It allows for a formal definition of a problem as per the need to convert some given situation into some desired situation using a set of permissible operations.
2. It permits the problem to be solved with the help of known techniques and control strategies to move through the problem space until goal state is found.

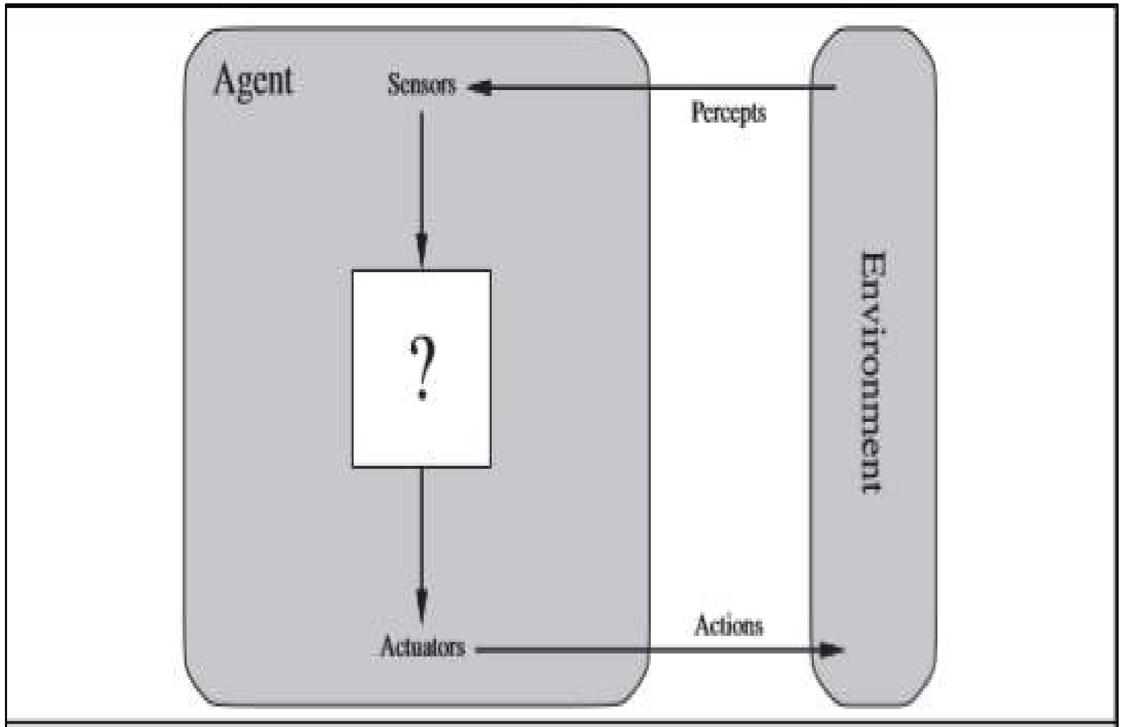
Module– 3 – Contents

Problem Solving by searching, Probabilistic reasoning in AI

- Solving problems by searching
  - Classical Search
  - Adversarial Search
  - Constraint Satisfaction Problems.
- Probabilistic reasoning in AI
- Bayesian networks
- Hidden Markov Model.

Agent & Environment

- ‘Anything’ that can gather information about its environment and take **action** based on that information.
- In AI, an intelligent agent (IA) is an autonomous entity which observes through sensors and acts upon an environment using actuators and directs its activity towards achieving goals.
- **Intelligent** agents may also learn or use knowledge to achieve their goals.
- A **rational** agent is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome.



Agents interact with environments through sensors and actuators.

Agents

- A human agent
  - eyes, ears, and other organs for sensors
  - hands, legs, vocal tract, and so on for actuators.
- A robotic agent
  - might have cameras and infrared range finders for sensors
  - various motors for actuators.
- A software agent
  - receives keystrokes, file contents, and network packets as sensory inputs
  - acts on the environment by displaying on the screen, writing files, and sending network packets.

Agent…

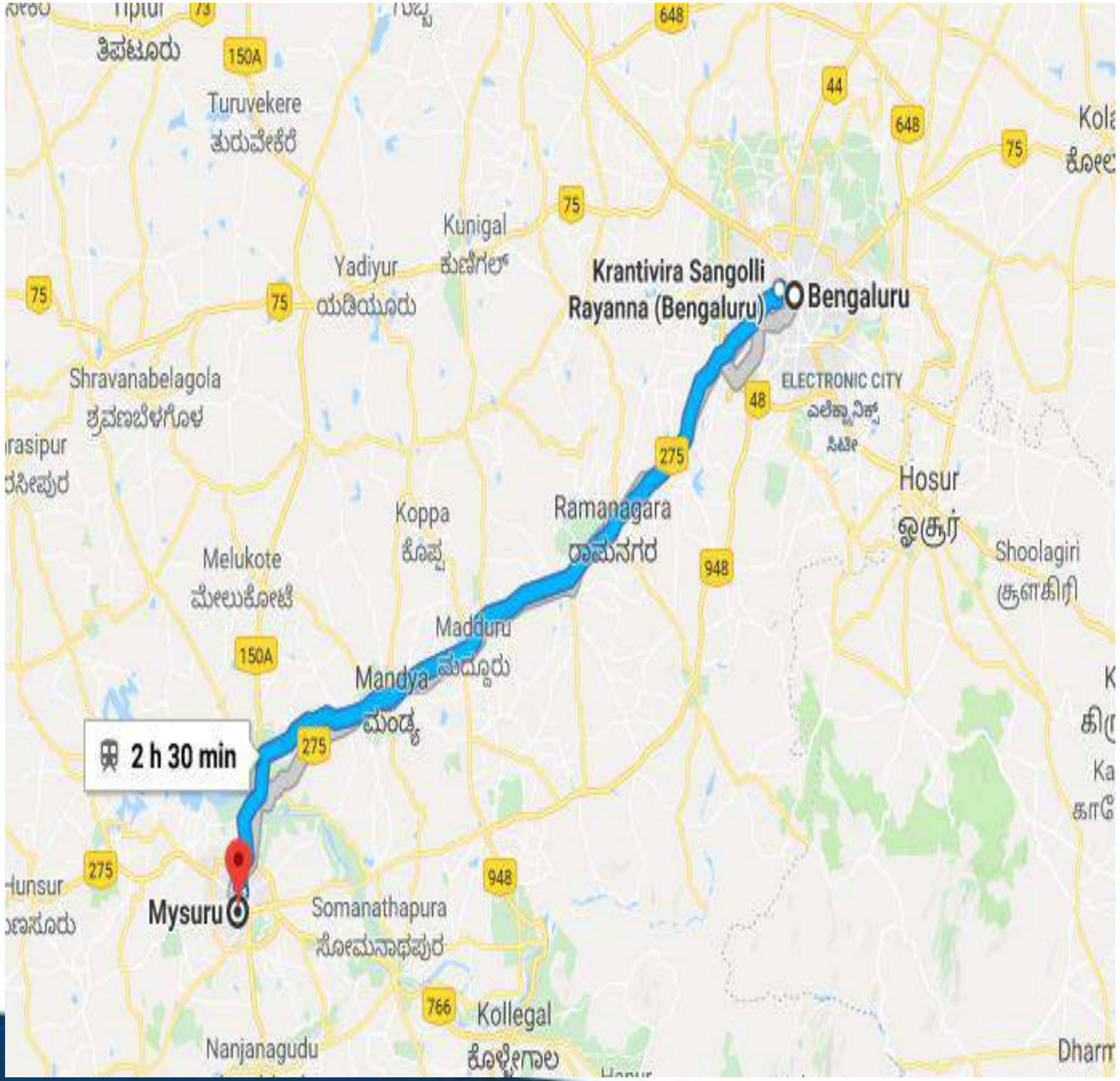
- **Percept** refers to the agent’s perceptual inputs at any given instant.
- An agent’s **percept sequence** is the complete history of everything the agent has ever perceived.
- An agent’s choice of action at any given instant can depend on the entire percept sequence observed to date, but not on anything it hasn’t perceived.
- Mathematically speaking, we say that an agent’s behavior is described by the agent function that maps any given percept sequence to an action
- One could view a hand-held calculator as an agent that chooses the action of displaying “4” when given the percept sequence “2 + 2 =,”

Solving problems by Searching

- Goal-based agent called a ***Problem-solving Agent***
- ***Uninformed Search Algorithms***—algorithms that are given no information about the problem other than its definition.
  - Although some of these algorithms can solve any solvable problem, none of them can do so efficiently
- ***Informed Search Algorithms***, on the other hand, can do quite well given some guidance on where to look for solutions

Problem : Travel from Bangalore to Mysore

- If the agent has no additional information—i.e., if the environment is unknown in the sense—then it is has no choice but to try one of the actions at random.
- But suppose the agent has a map of Karnataka
- The point of a map is to provide the agent with information about the states it might get itself into and the actions it can take.
- The agent can use this information to consider subsequent stages of a hypothetical journey via each of the towns, trying to find a journey that eventually gets to Mysore.
- Once it has found a path on the map from Bangalore to Mysore, it can achieve its goal by carrying out the driving actions that correspond to the legs of the journey.
- ***An agent with several immediate options of unknown value can decide what to do by first examining future actions that eventually lead to states of known value.***
- **OBSERVABLE** :The agent always knows the current state.
- **DISCRETE**: At any given state there are only finitely many actions to choose from
- **KNOWN** : The agent knows which states are reached by each action
- **DETERMINISTIC :** Each action has exactly one outcome
- Under these assumptions, the solution to any problem is a fixed sequence of actions.
- The process of looking for a sequence of actions that reaches the goal is called **search**.



PRESIDENCY UNIVERSITY

Well Defined Problem

- A problem can be defined formally by **five** components:

1. The ***initial state*** that the agent starts in. For example, the initial state for our agent in Bangalore might be described as *In*(Bangalore).
2. A description of the possible ***actions*** available to the agent.
   - Given a particular state s, ACTIONS(s) returns the set of actions that can be executed in s.
   - Each of these actions is applicable in s.
   - For example, from the state ***In***(Bangalore), the applicable
   - Actions are ----- {**Go**(Hosur), **Go**(Kunigal), **Go**(Kolar)}.
3. A description of what each action does; the formal name for this is the ***transition model***, specified by a function RESULT(s, a) that returns the state that results from doing action *a* in state *s*.
   - We also use the term **successor** to refer to any state reachable from a given state by a single action.
   - For example, **RESULT** (In(Bangalore),Go(Hosur)) = In(Hosur) .

Well Defined Problem…..

- Together, the **initial state**, **actions**, and **transition model** implicitly define the **state space** of the problem—the set of all states reachable from the initial state by any sequence of actions.
- The state space forms a directed network or **graph** in which the nodes are states and the links between nodes are actions.
- (The map of Karnataka can be interpreted as a state-space graph if we view each road as standing for two driving actions, one in each direction.)
- A **path** in the state space is a sequence of states connected by a sequence of actions.

4. The **goal test**, which determines whether a given state is a goal state.
    - Sometimes there is an explicit set of possible goal states, and the test simply checks whether the given state is one of them.
    - The agent's goal in Karnataka is the singleton set {In(Mysore)}.
5. A **path cost** function that assigns a numeric cost to each path. The problem-solving agent chooses a cost function that reflects its own performance measure.
    - For the agent trying to get to Mysore, time is of the essence, so the cost of a path might be its length in kilometers.
    - Cost of a path can be described as the *sum* of the costs of the individual actions along the path.
    - The **step cost** of taking action *a* in state *s* to reach state *s'* is denoted by c(s, a, s')
    - The step costs for Karnataka are route distances.

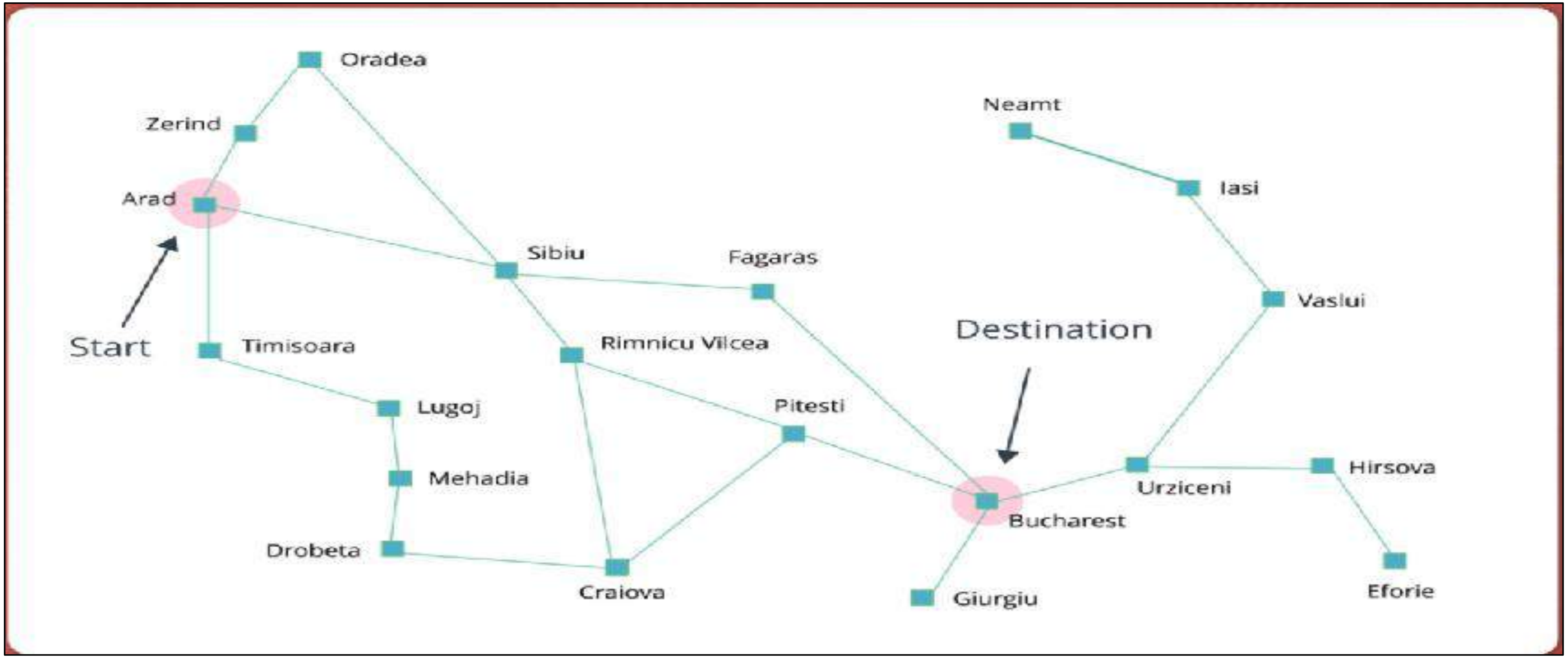
- A **solution** to a problem is an action sequence that leads from the initial state to a goal state.
- Solution quality is measured by the path cost function, and an **optimal solution** has the lowest path cost among all solutions.
- The process of removing detail from a representation is called **abstraction**. (Like Weather, Road Condition etc.)

NAVIGATION PROBLEM

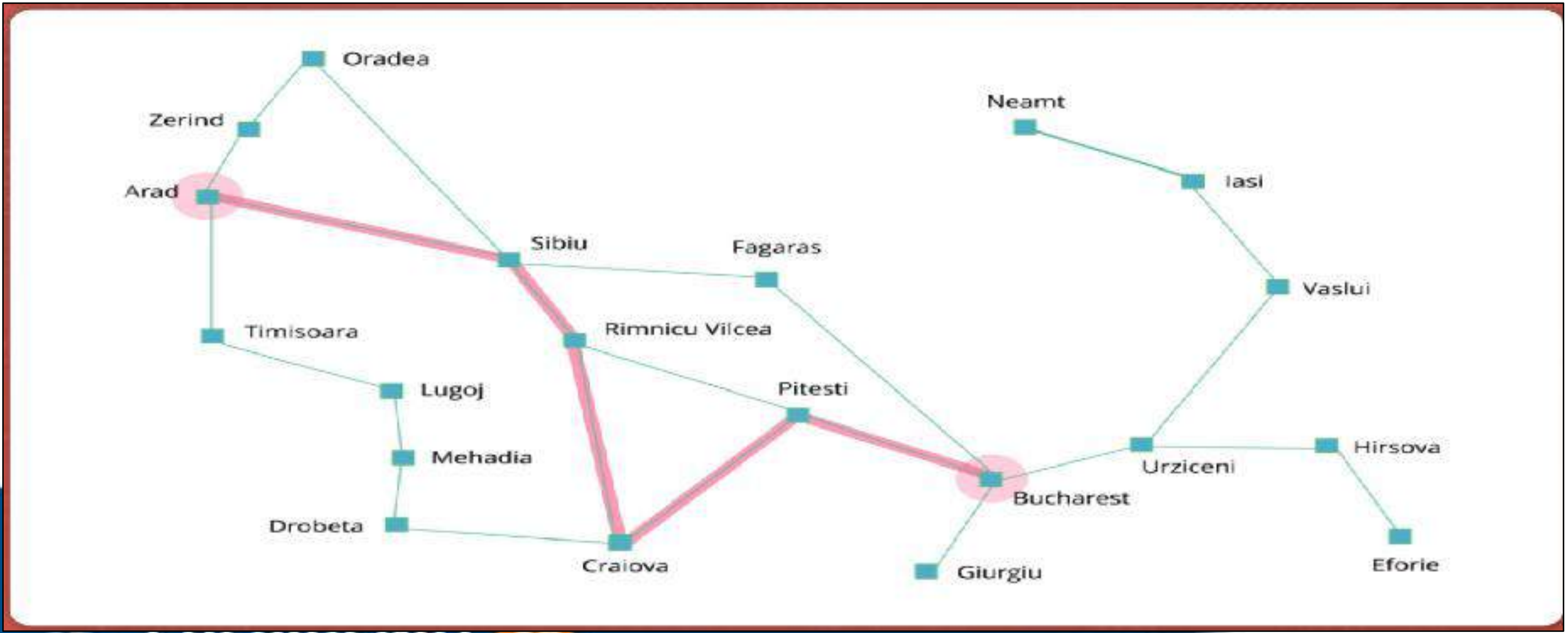
Romania

Problem : Arad To Bucharest

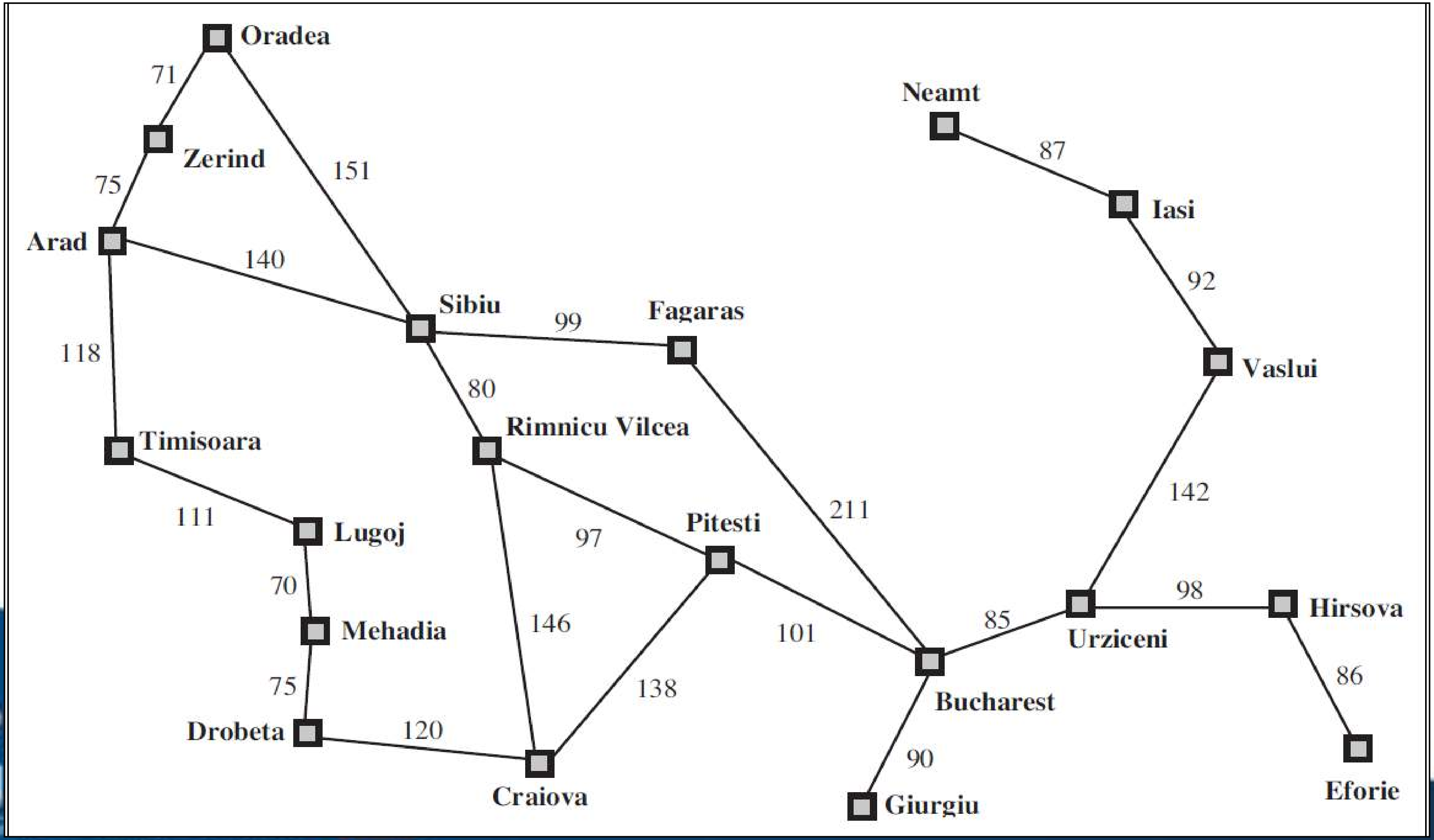
Solution ????

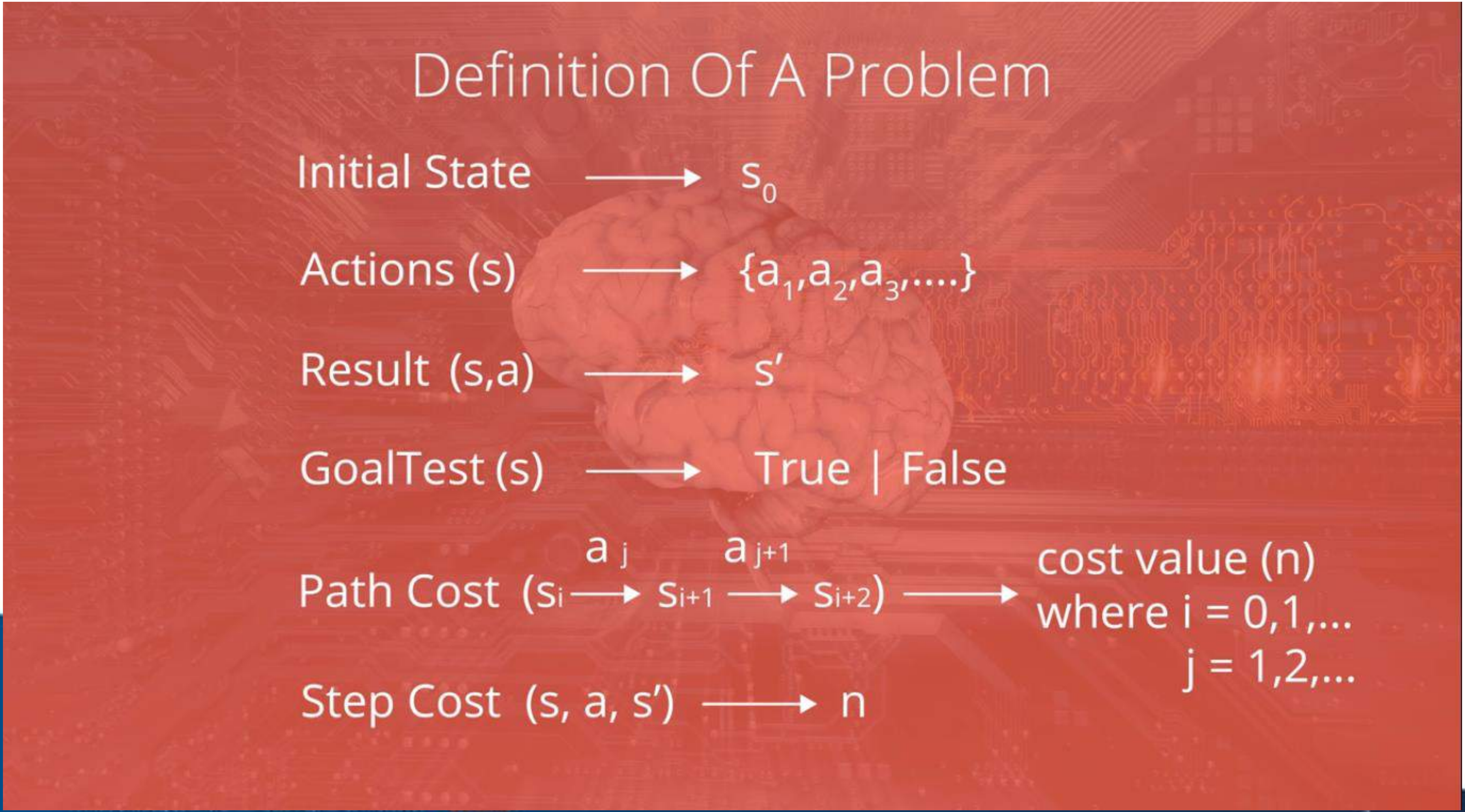


One Solution……



Map of Romania
Problem: Arad to Bucharest





Definition Of A Problem

Initial State ⟶ s_0

Actions (s) ⟶ $\{a_1, a_2, a_3, \ldots.\}$

Result (s,a) ⟶ s’

GoalTest (s) ⟶ True | False

Path Cost $(s_i \xrightarrow{a_j} s_{i+1} \xrightarrow{a_{j+1}} s_{i+2})$ ⟶ cost value (n)
where i = 0,1,...
j = 1,2,...

Step Cost (s, a, s’) ⟶ n

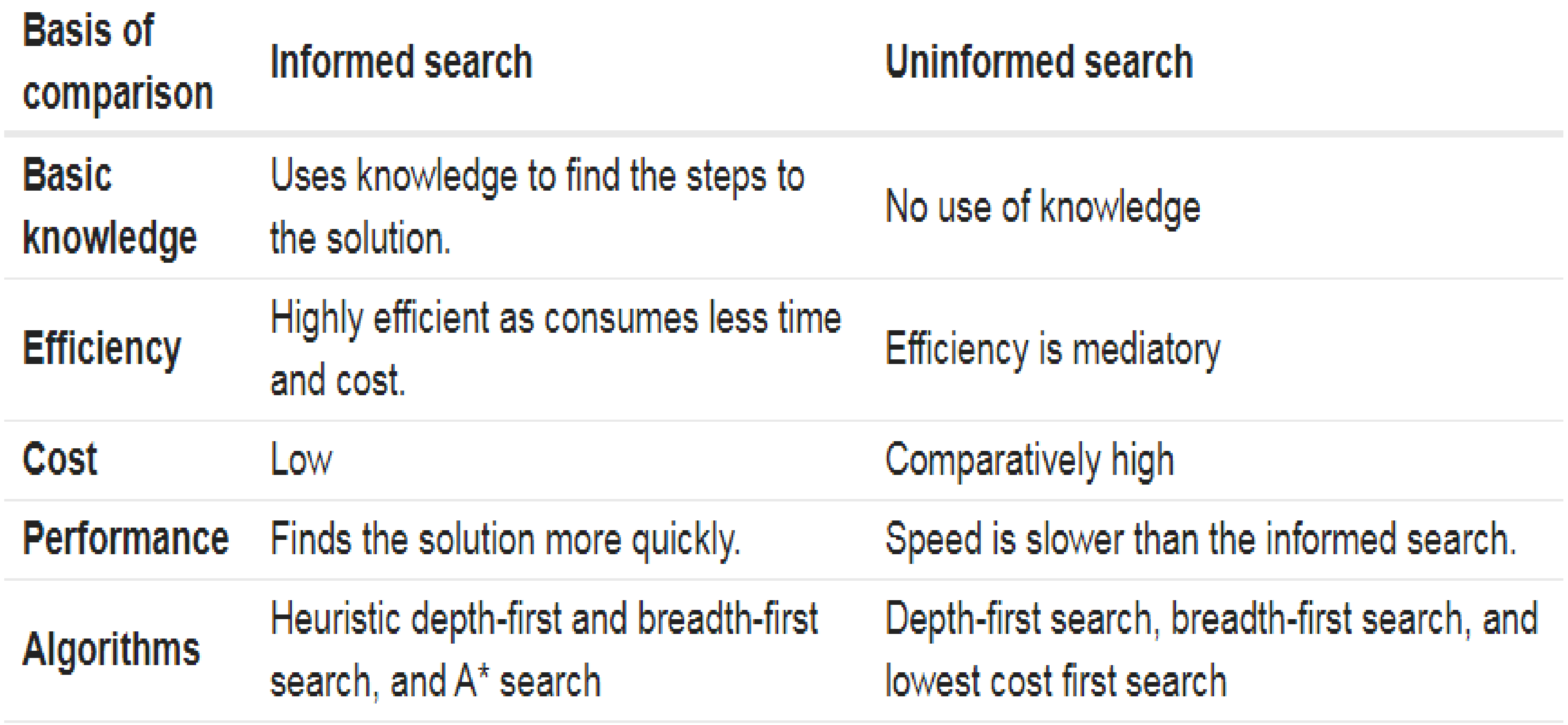
Uninformed Search

- These strategies have no additional information about states beyond that provided in the problem definition, so they can only proceed by generating successors until they find a goal state.

- BREADTH FIRST
- CHEAPEST FIRST
- DEPTH FIRST

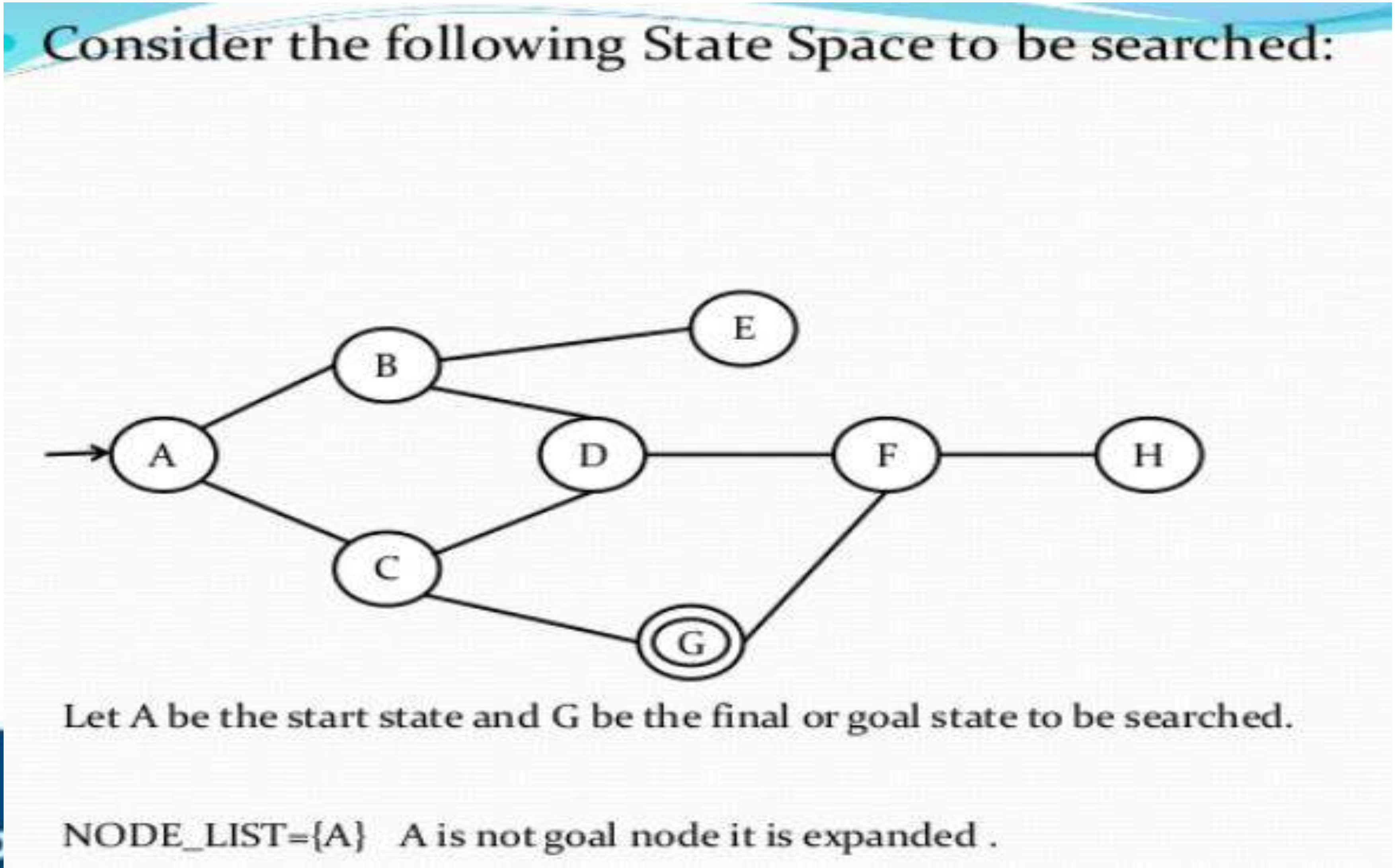
INFORMED SEARCH / (HEURISTIC SEARCH)

- These strategies have additional information about search states, so they can guide the search by ranking successors according to some fitness score until they find a goal state.
- **A* Search**
- **AO* Search**



Basis of comparison	Informed search	Uninformed search
Basic knowledge	Uses knowledge to find the steps to the solution.	No use of knowledge
Efficiency	Highly efficient as consumes less time and cost.	Efficiency is mediatory
Cost	Low	Comparatively high
Performance	Finds the solution more quickly.	Speed is slower than the informed search.
Algorithms	Heuristic depth-first and breadth-first search, and A* search	Depth-first search, breadth-first search, and lowest cost first search

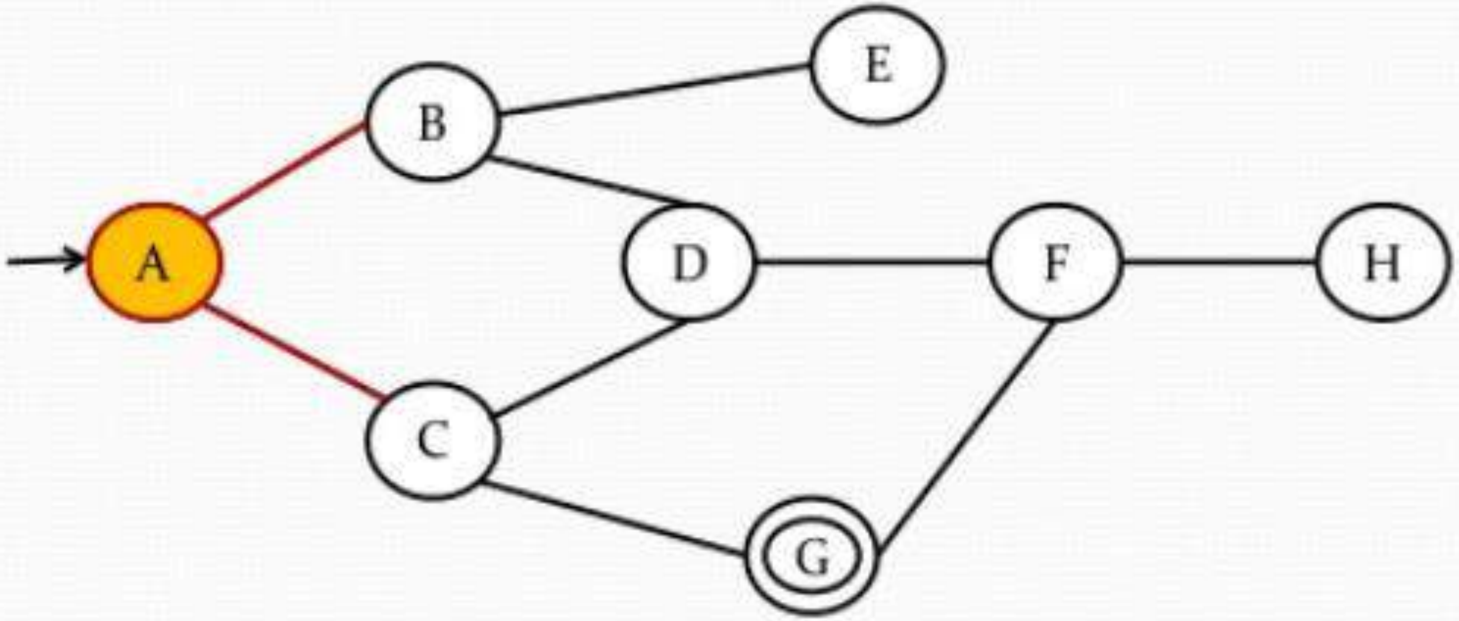
BFS



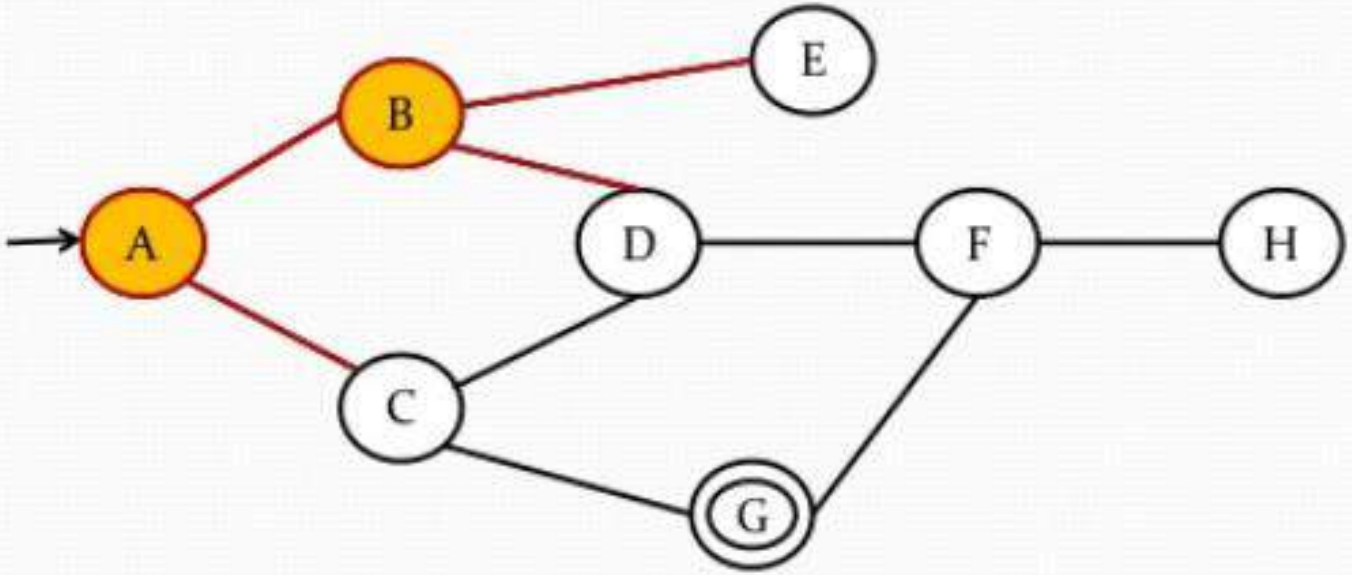
- Consider the following State Space to be searched:

Let A be the start state and G be the final or goal state to be searched.

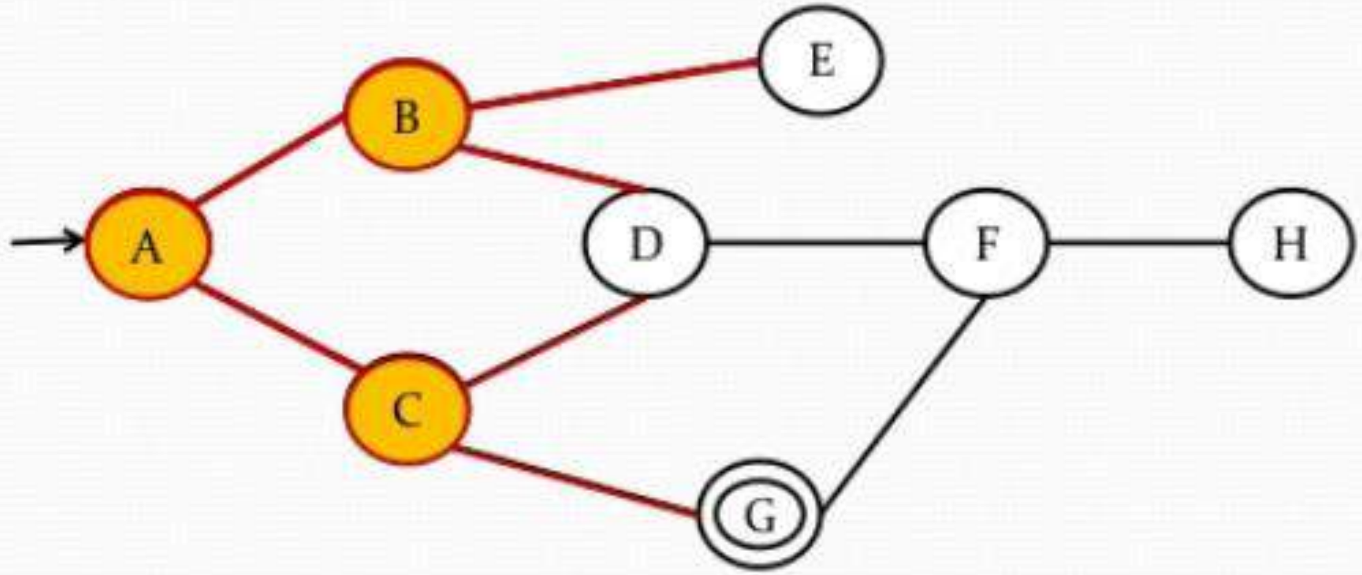
NODE_LIST={A}   A is not goal node it is expanded .



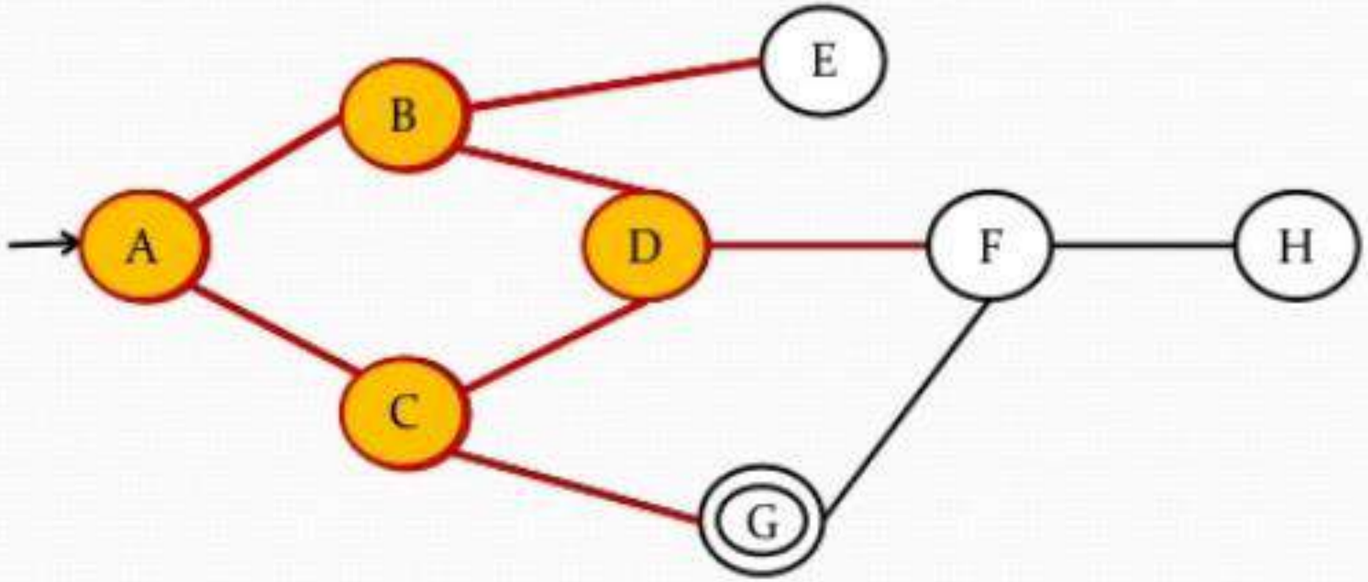
NODE_LIST={B,C}



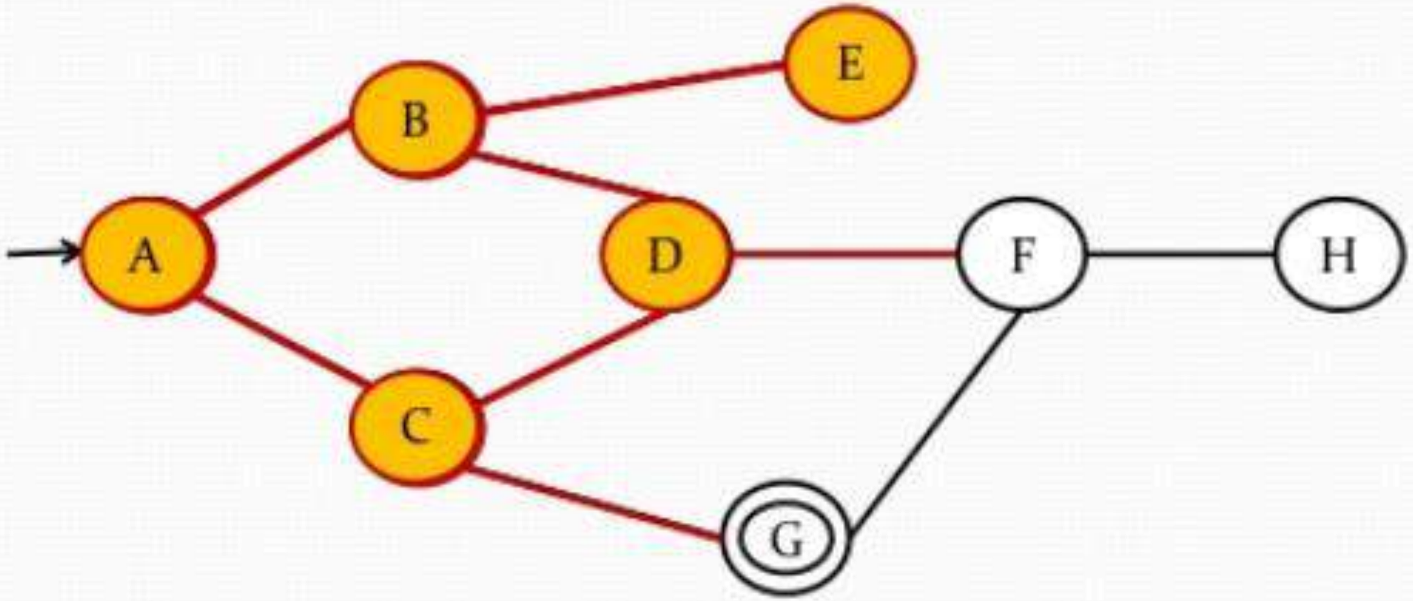
NODE_LIST={C,**D**,**E**}



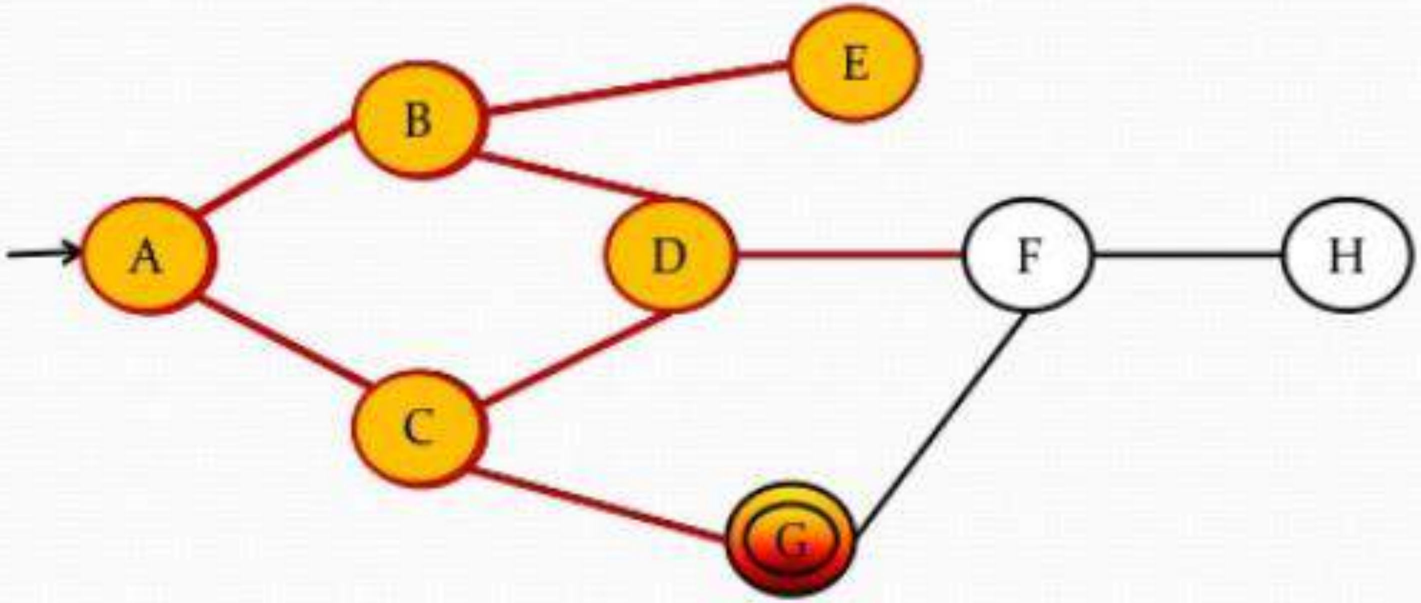
NODE_LIST={D,E,**G**}



NODE_LIST={E,G,**F**}

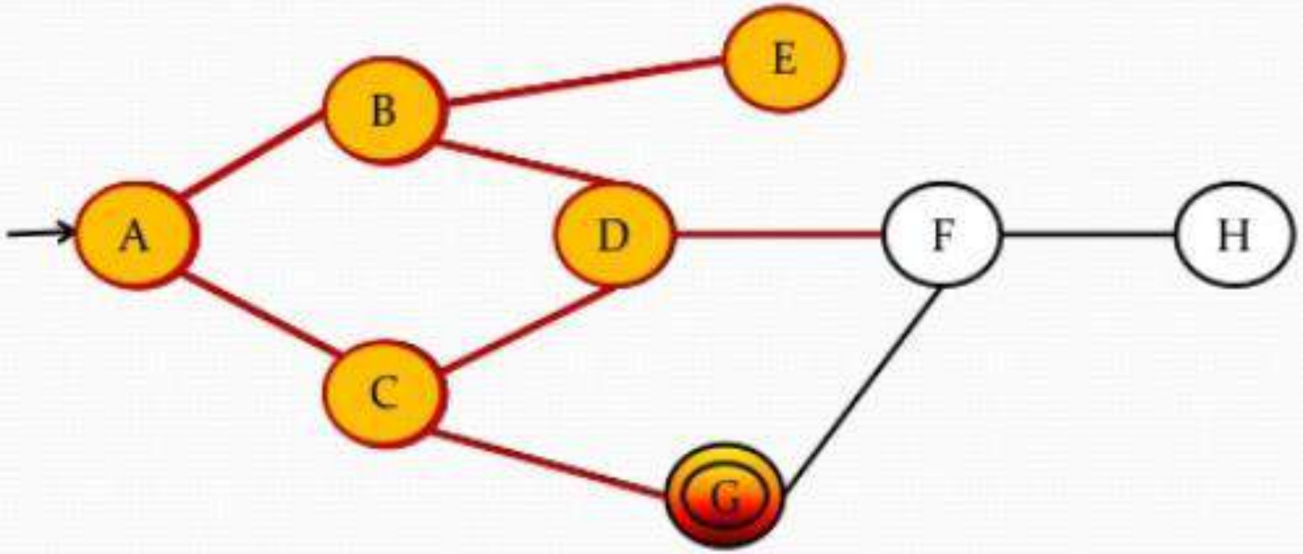


NODE_LIST={G,F}



NODE_LIST={G,F}

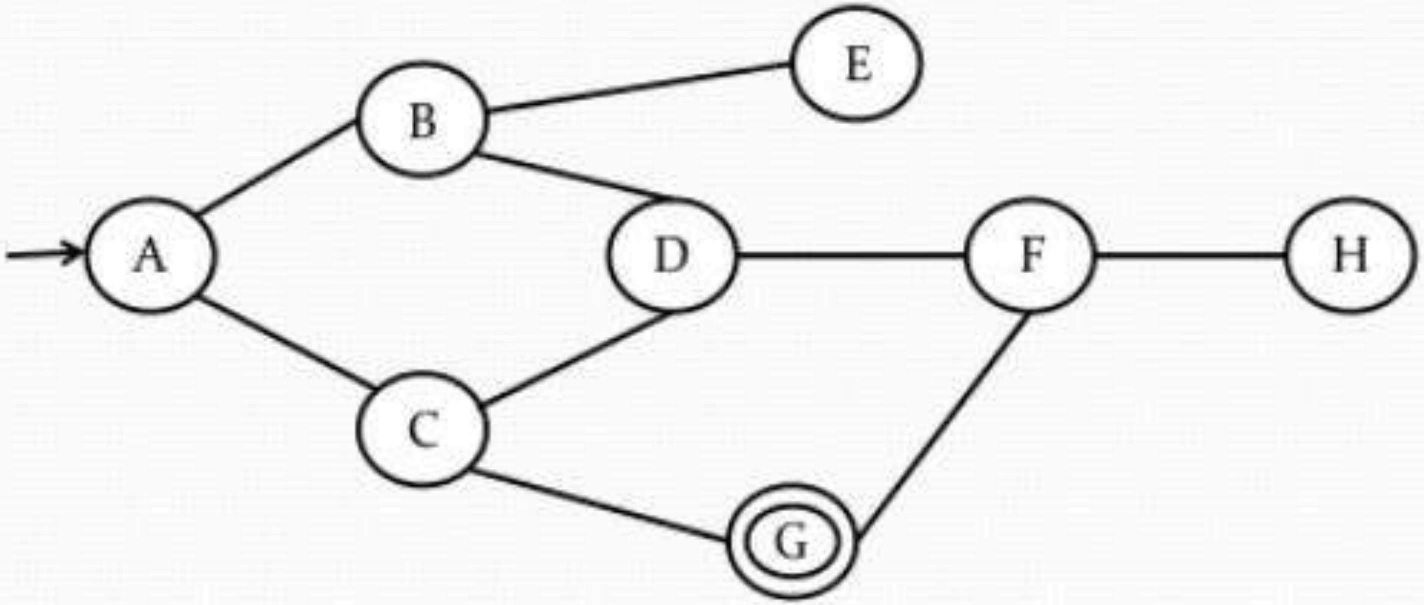
GOAL NODE FOUND!!



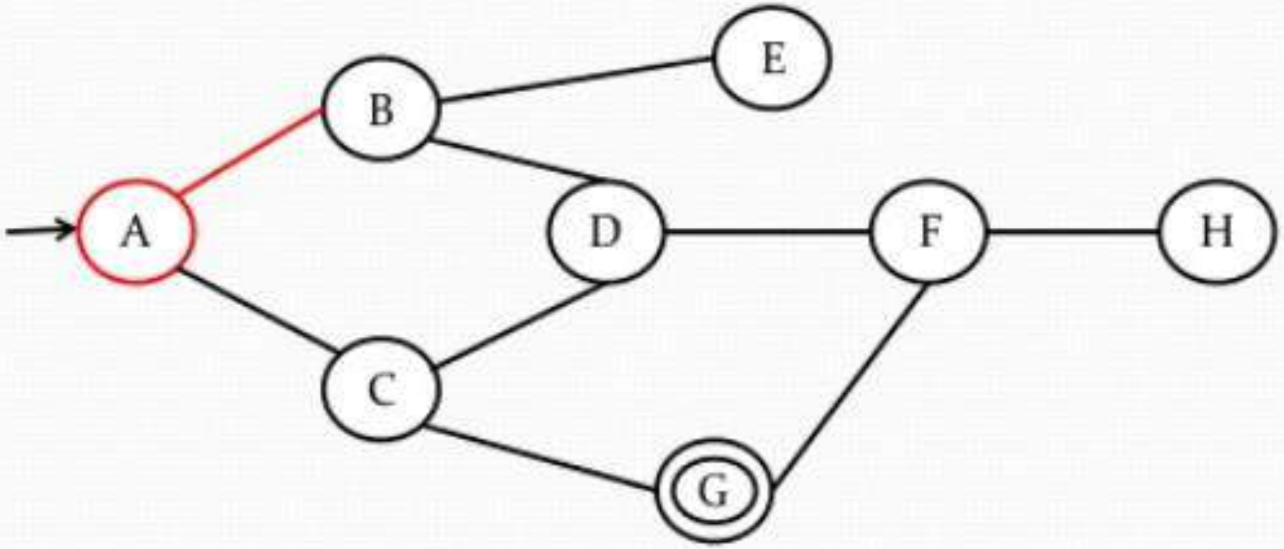
NODE_LIST={G,F}

TRAVERSAL ORDER: A-B-C-D-E-G

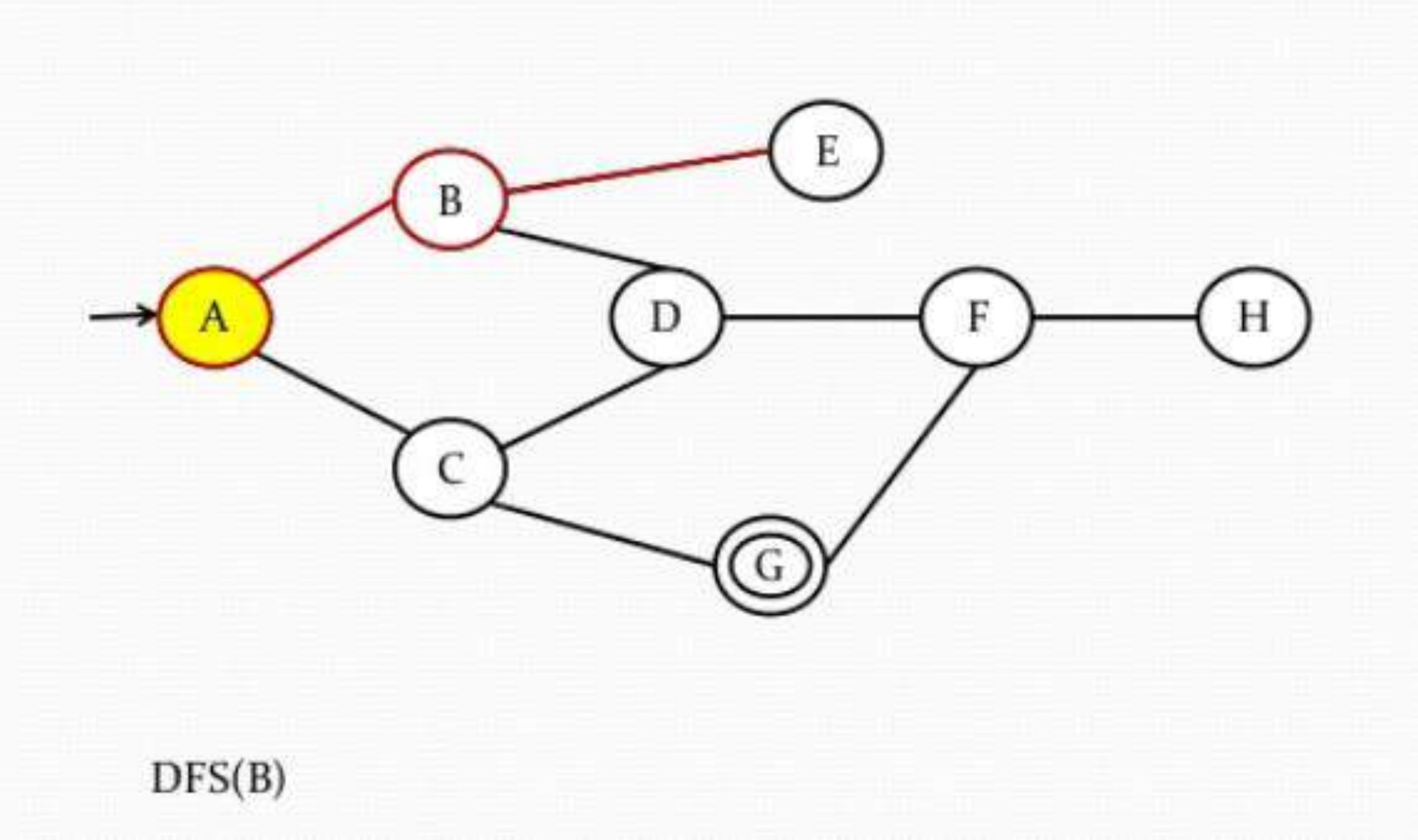
DFS



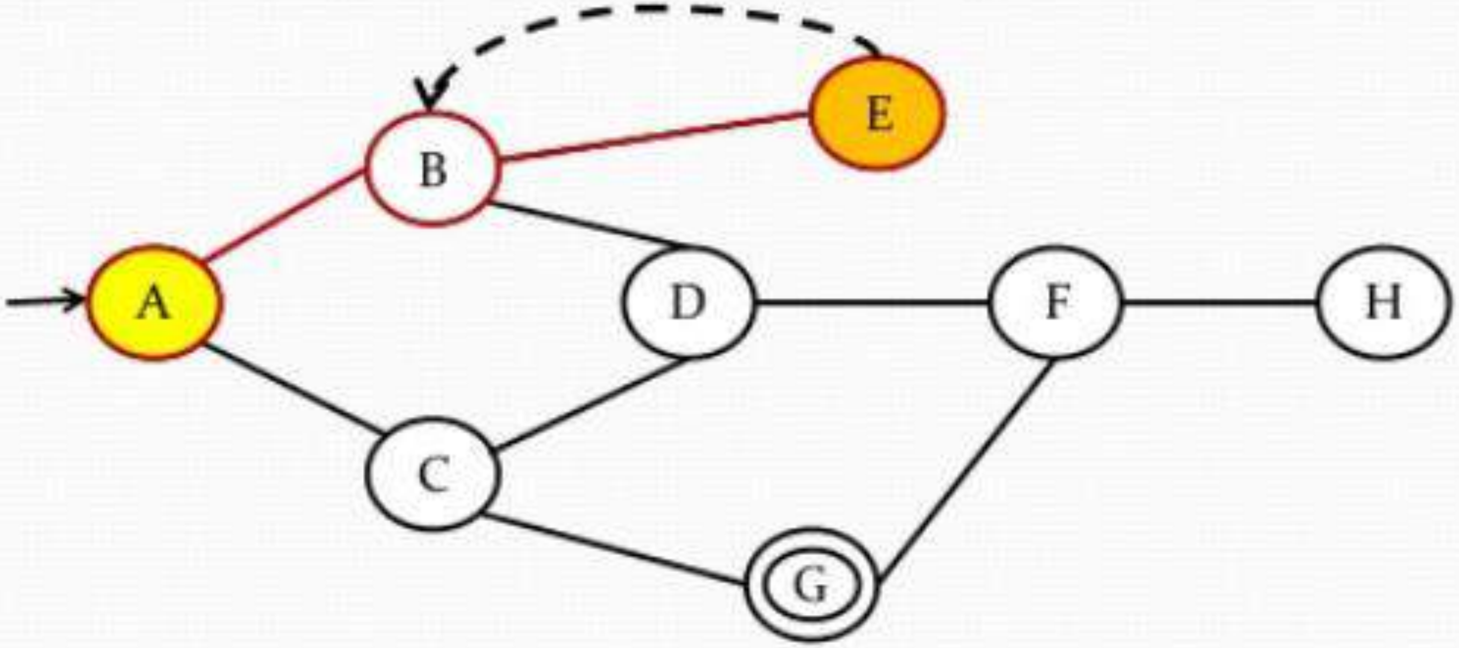
DFS(A)



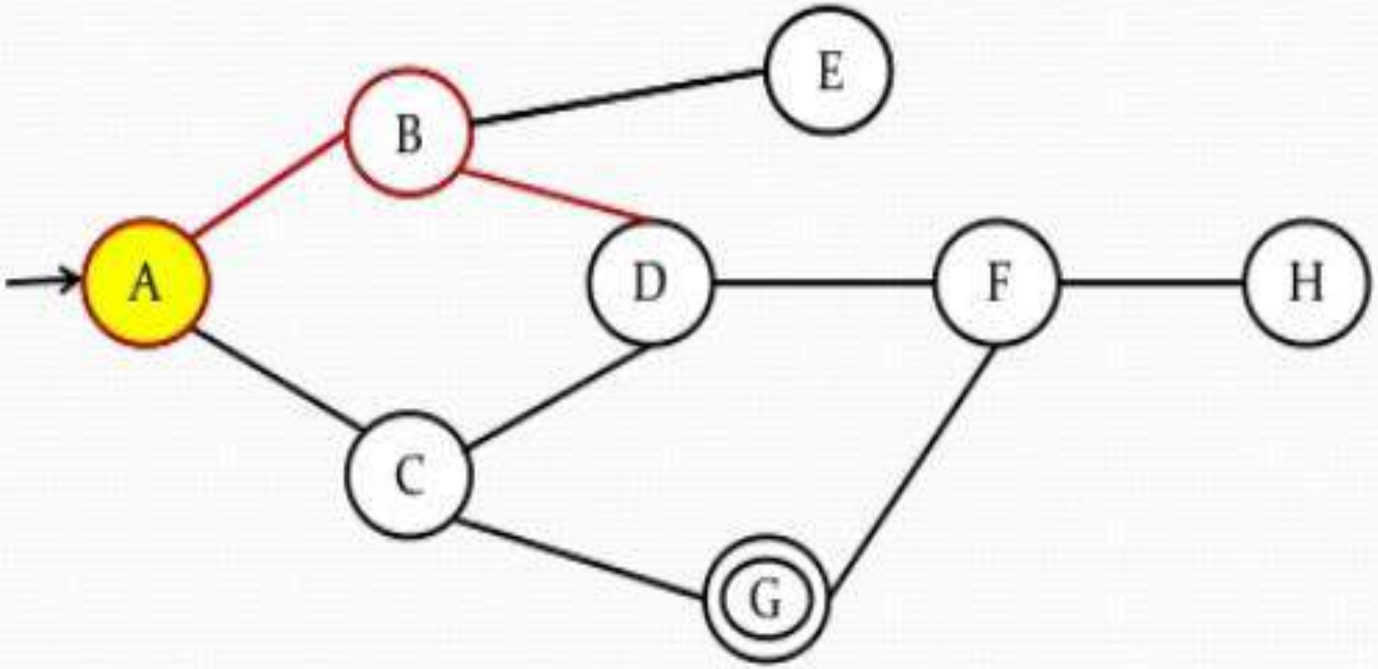
DFS(A)



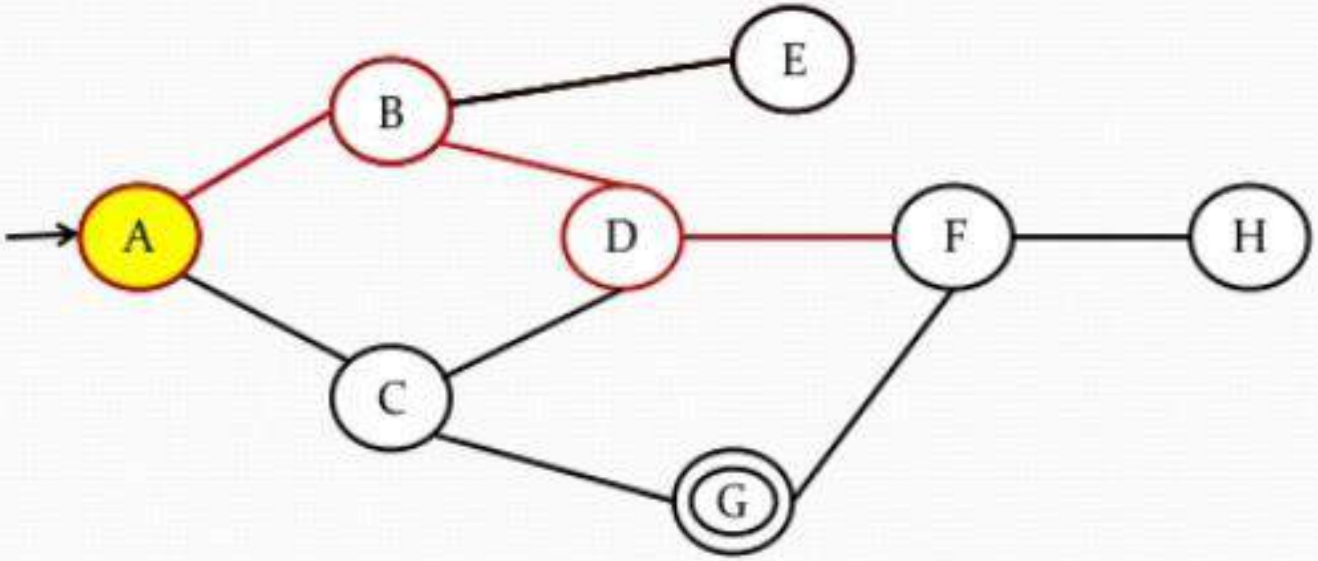
DFS(B)



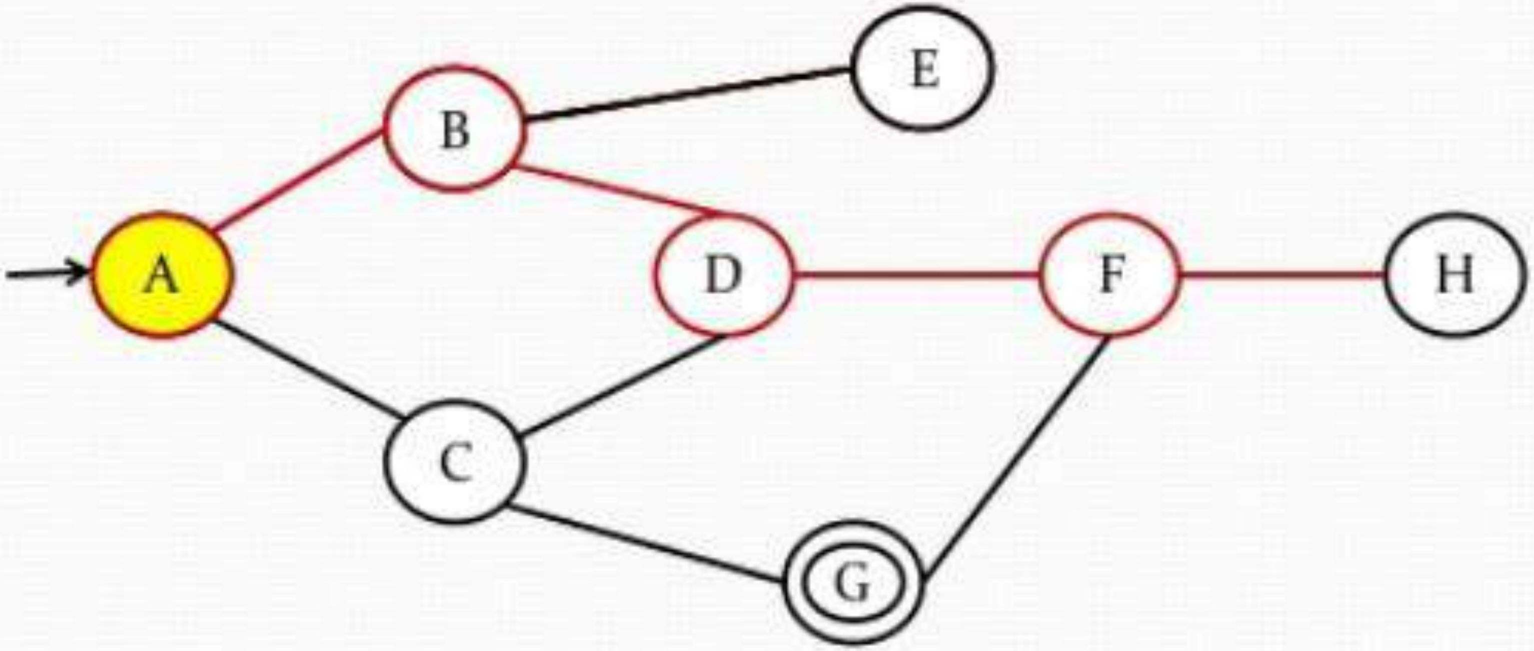
DFS(E)



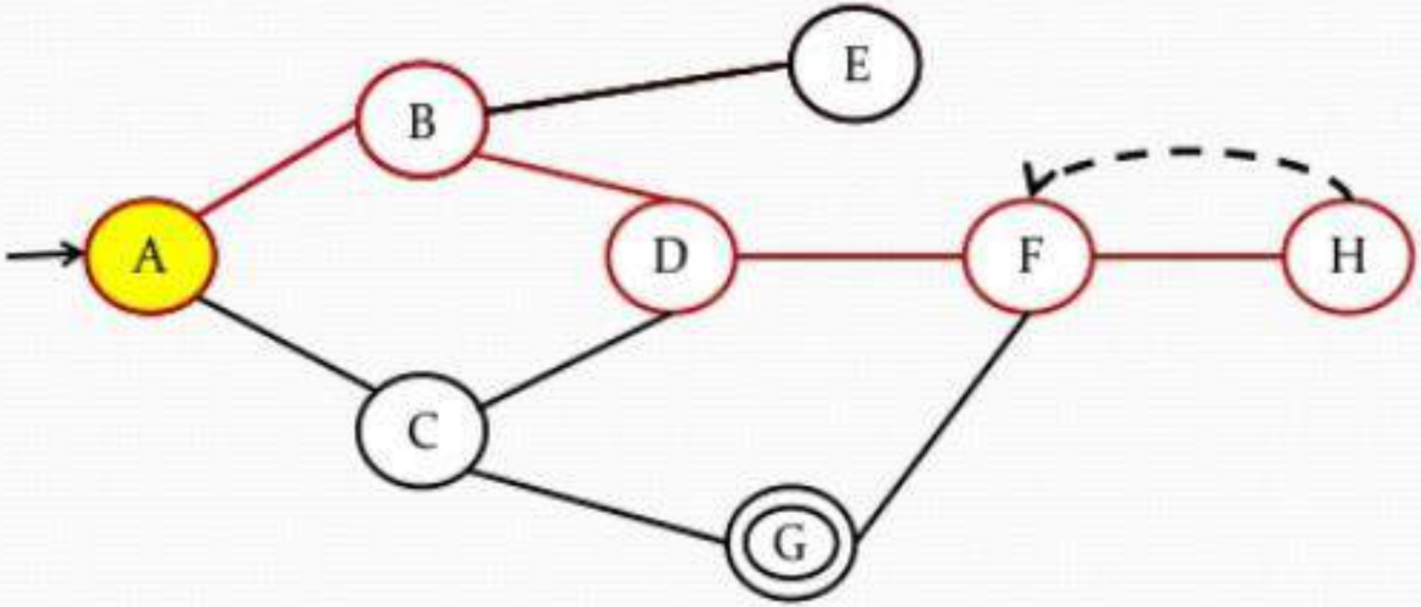
DFS(B)



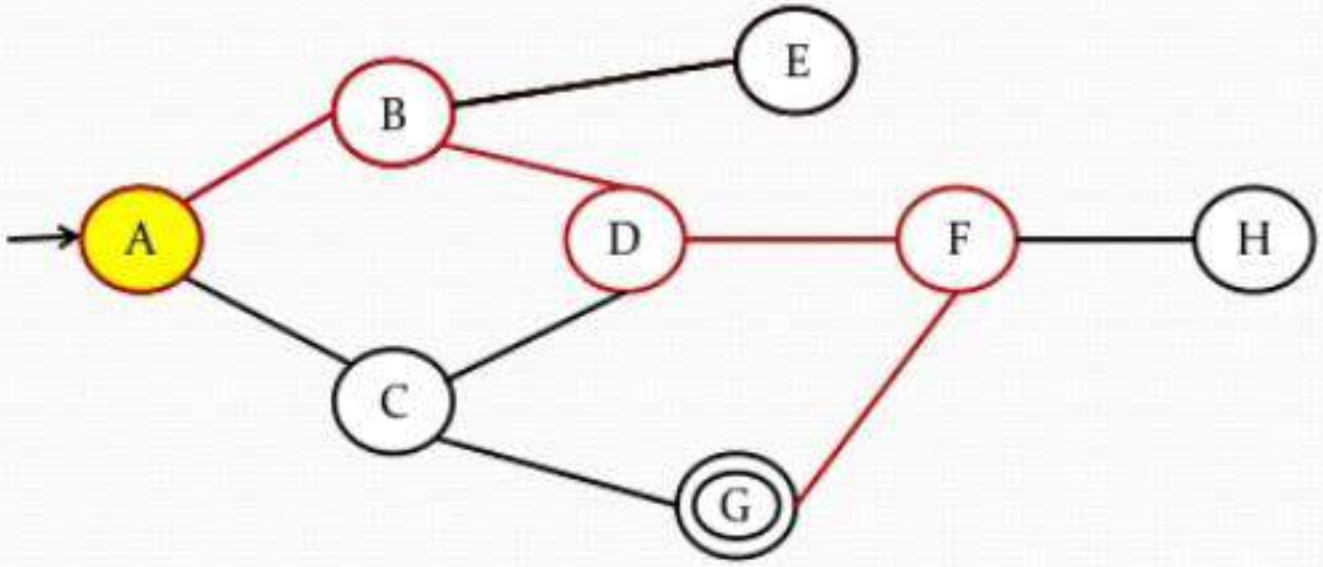
DFS(D)



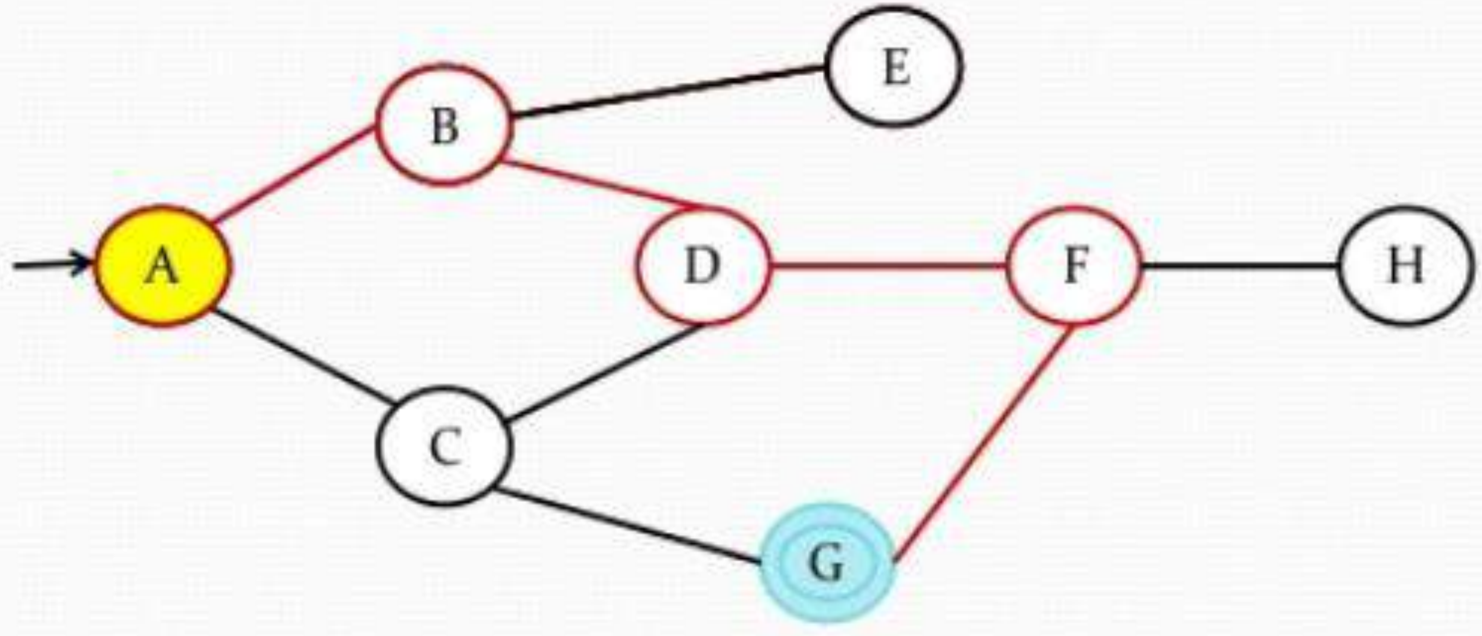
DFS(F)



DFS(H)

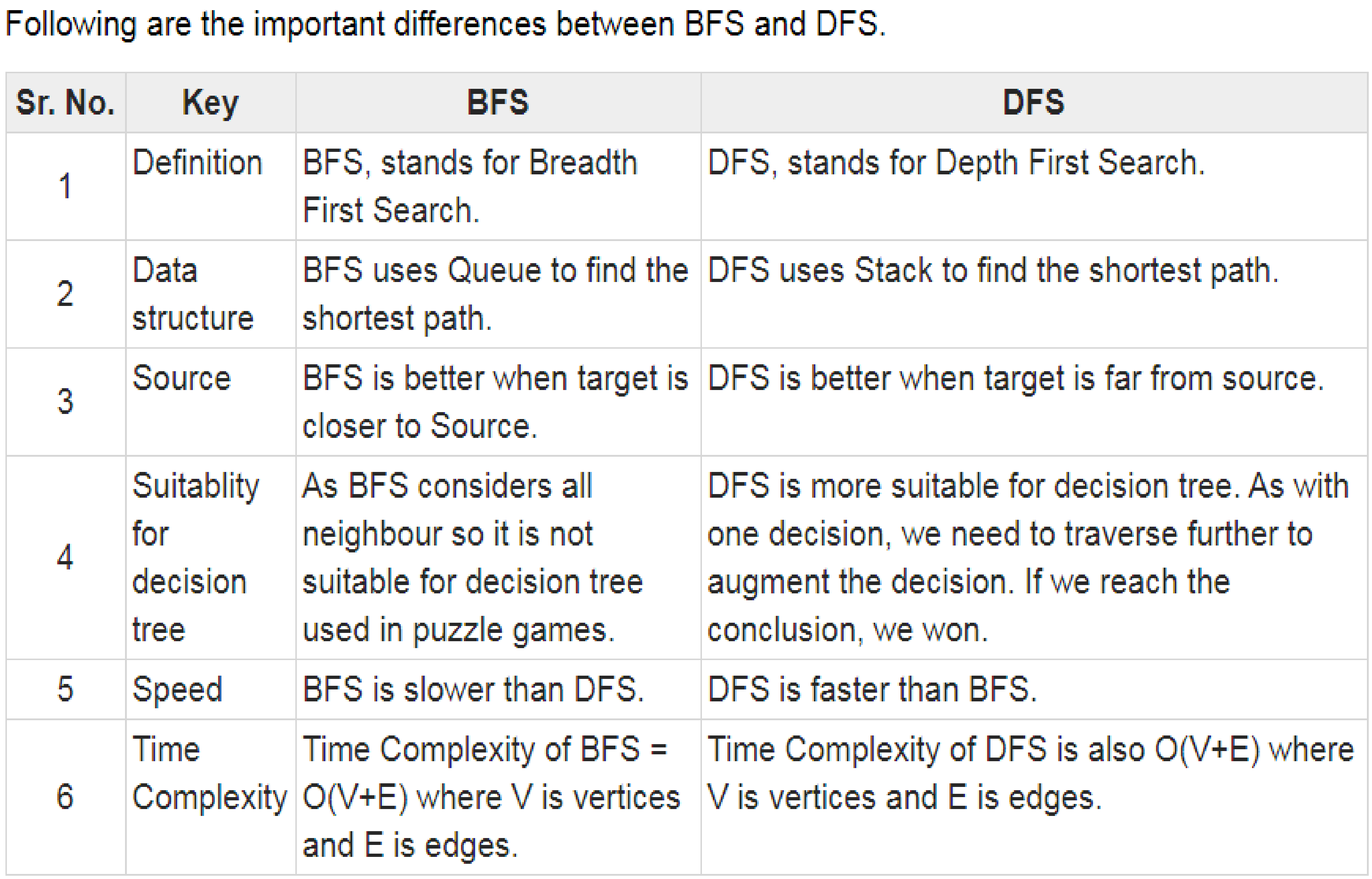


DFS(F)



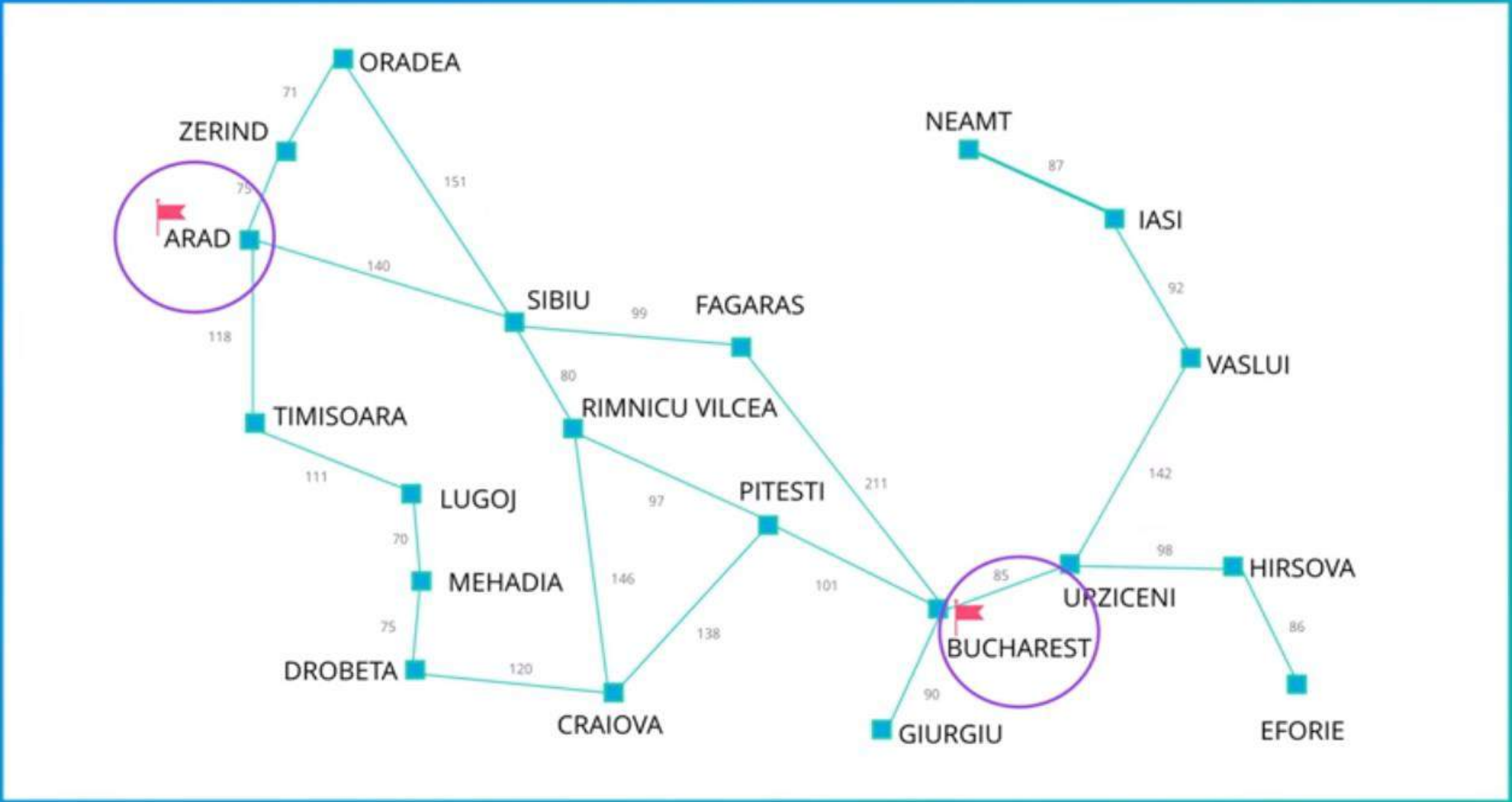
DFS(G)

GOAL NODE FOUND!!



Following are the important differences between BFS and DFS.

Sr. No.	Key	BFS	DFS
1	Definition	BFS, stands for Breadth First Search.	DFS, stands for Depth First Search.
2	Data structure	BFS uses Queue to find the shortest path.	DFS uses Stack to find the shortest path.
3	Source	BFS is better when target is closer to Source.	DFS is better when target is far from source.
4	Suitablity for decision tree	As BFS considers all neighbour so it is not suitable for decision tree used in puzzle games.	DFS is more suitable for decision tree. As with one decision, we need to traverse further to augment the decision. If we reach the conclusion, we won.
5	Speed	BFS is slower than DFS.	DFS is faster than BFS.
6	Time Complexity	Time Complexity of BFS = O(V+E) where V is vertices and E is edges.	Time Complexity of DFS is also O(V+E) where V is vertices and E is edges.

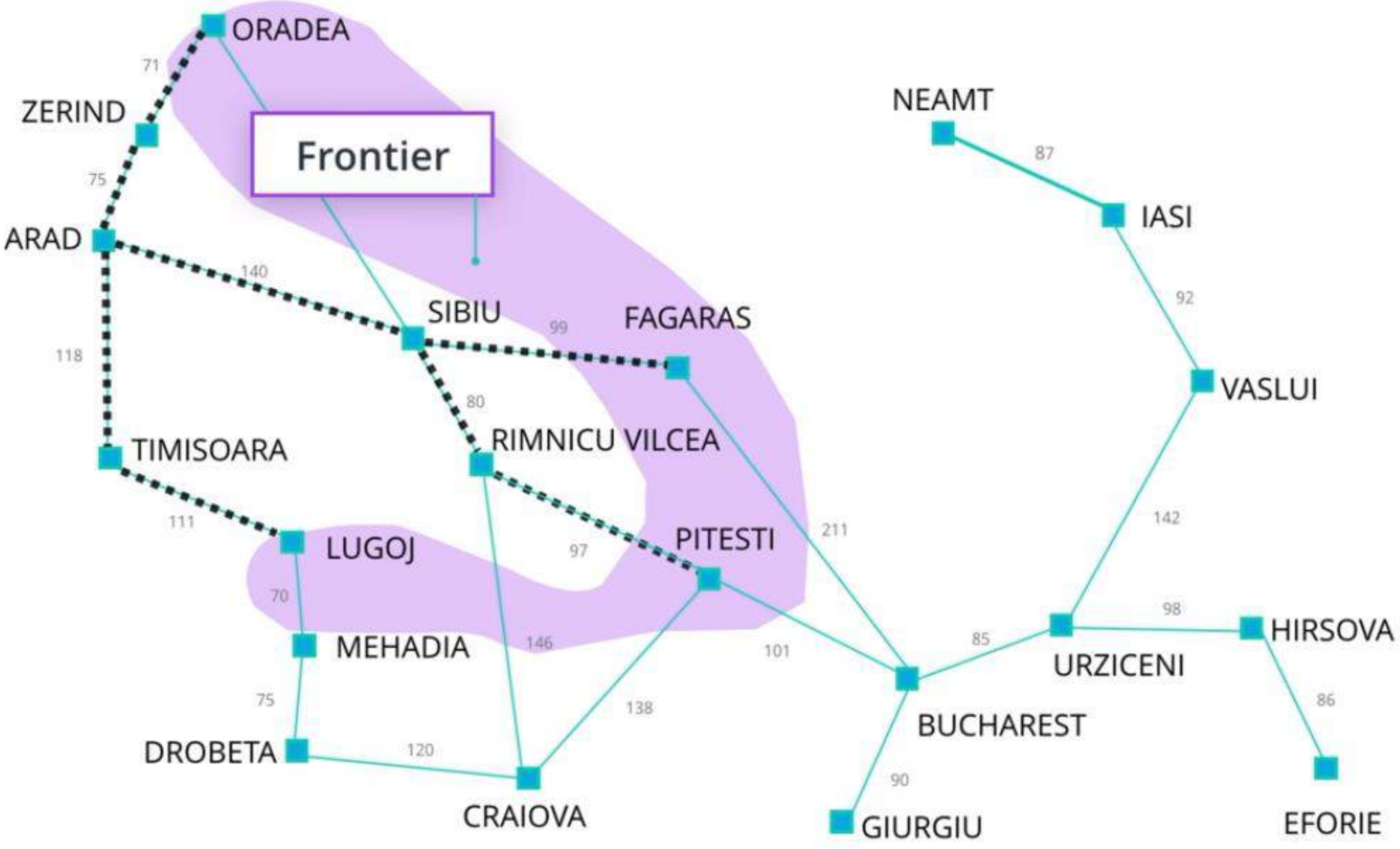

ORADEA
71
ZERIND
75
ARAD
151
140
SIBIU
99
FAGARAS
NEAMT
87
IASI
92
VASLUI
118
80
RIMNICU VILCEA
TIMISOARA
111
LUGOJ
97
PITESTI
211
142
70
MEHADIA
146
101
98
HIRSOVA
85
URZICENI
75
DROBETA
120
138
BUCHAREST
86
90
CRAIOVA
GIURGIU
EFORIE

Searching For Solutions

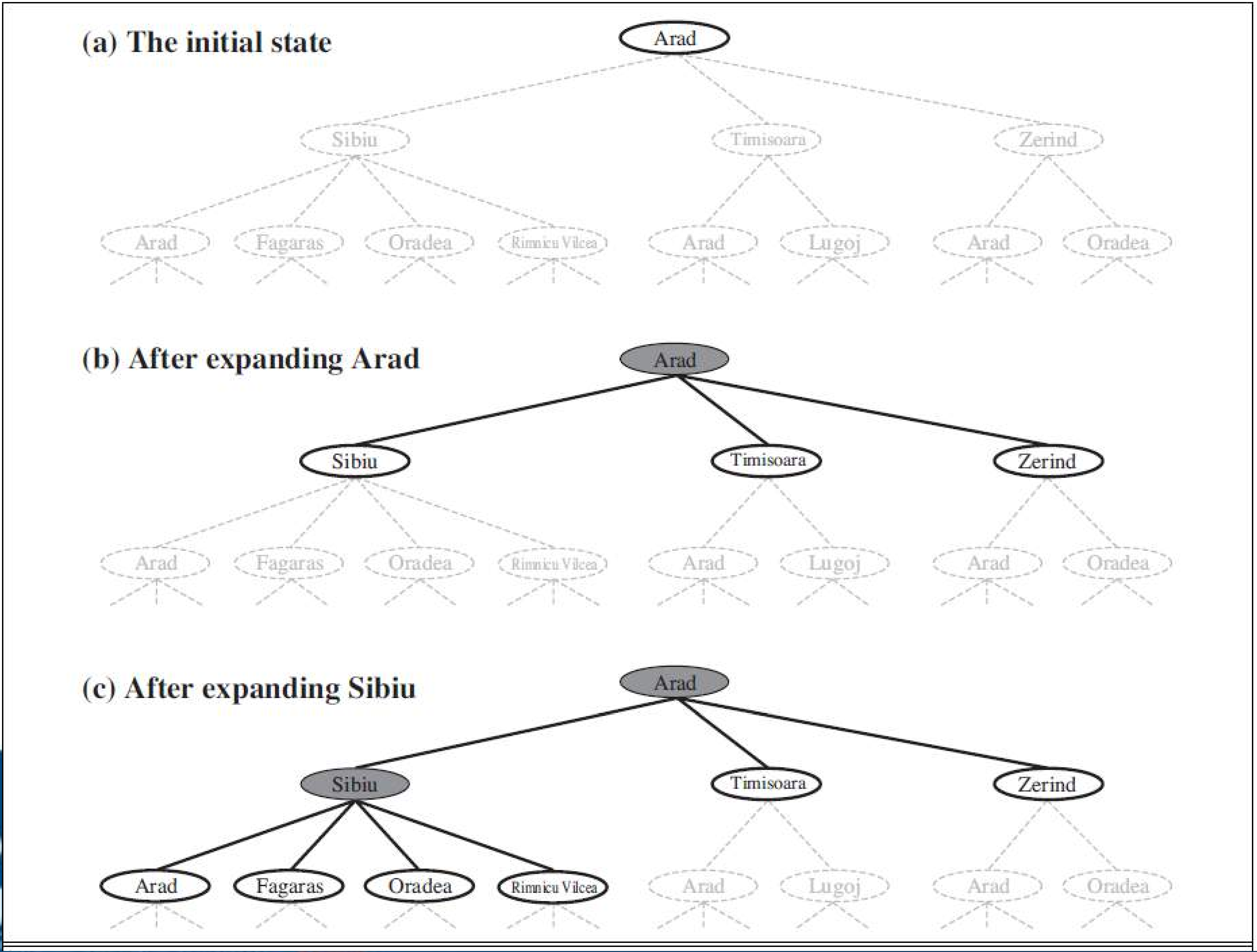
- The possible action sequences starting at the initial state form a **search tree** with the initial state at the root; the branches are actions and the **nodes** correspond to states in the state space of the problem.
- The root node of the tree corresponds to the initial state, *In(Arad)*.
- The first step is to test whether this is a goal state. As it is not
- We need to consider taking various actions. We do this by **expanding** the current state; that is, applying each legal action to the current state, thereby **generating** a new set of states.
- In this case, we add three branches from the **parent node** *In(Arad)* leading to three new **child nodes**: *In(Sibiu), In(Timisoara),* and *In(Zerind)*.
- Now we must choose which of these three possibilities to consider further.
- **Leaf node** is a node with no children in the tree.
- The set of all leaf nodes available for expansion at any given point is called the **frontier**.
- **Loopy Path**
- **Redundant Path**

Terms…

- State Space : Entire Area
- Actions : Navigate the State Space
    - Actions are specific to each state
- Frontier
- Explored & Unexplored


ORADEA
ZERIND
ARAD
TIMISOARA
LUGOJ
MEHADIA
DROBETA
CRAIOVA
SIBIU
FAGARAS
RIMNICU VILCEA
PITESTI
BUCHAREST
GIURGIU
URZICENI
HIRSOVA
EFORIE
VASLUI
IASI
NEAMT
71
75
140
118
111
70
75
120
99
80
97
146
138
101
211
85
90
98
86
142
92
87

Frontier



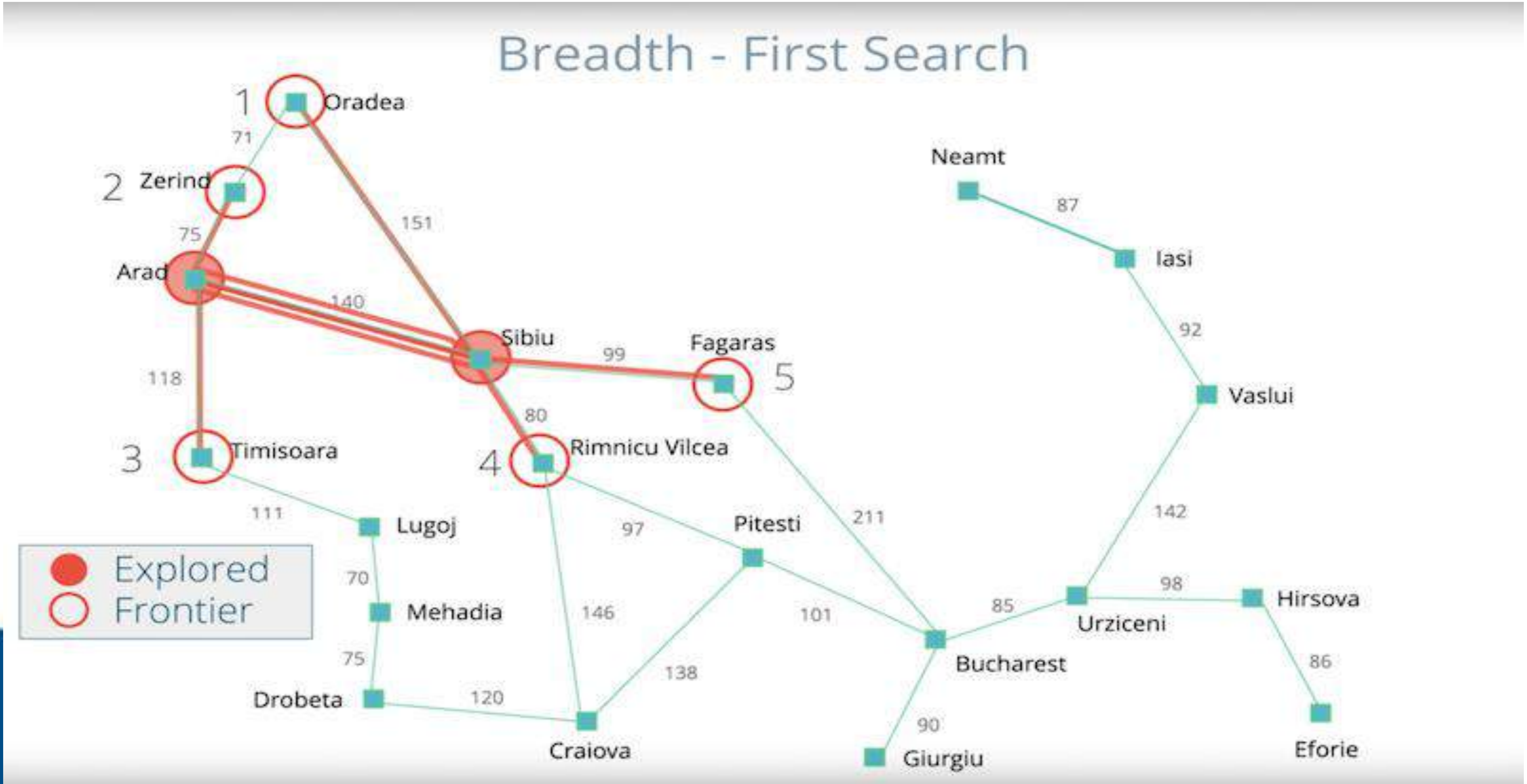
(a) The initial state

(b) After expanding Arad

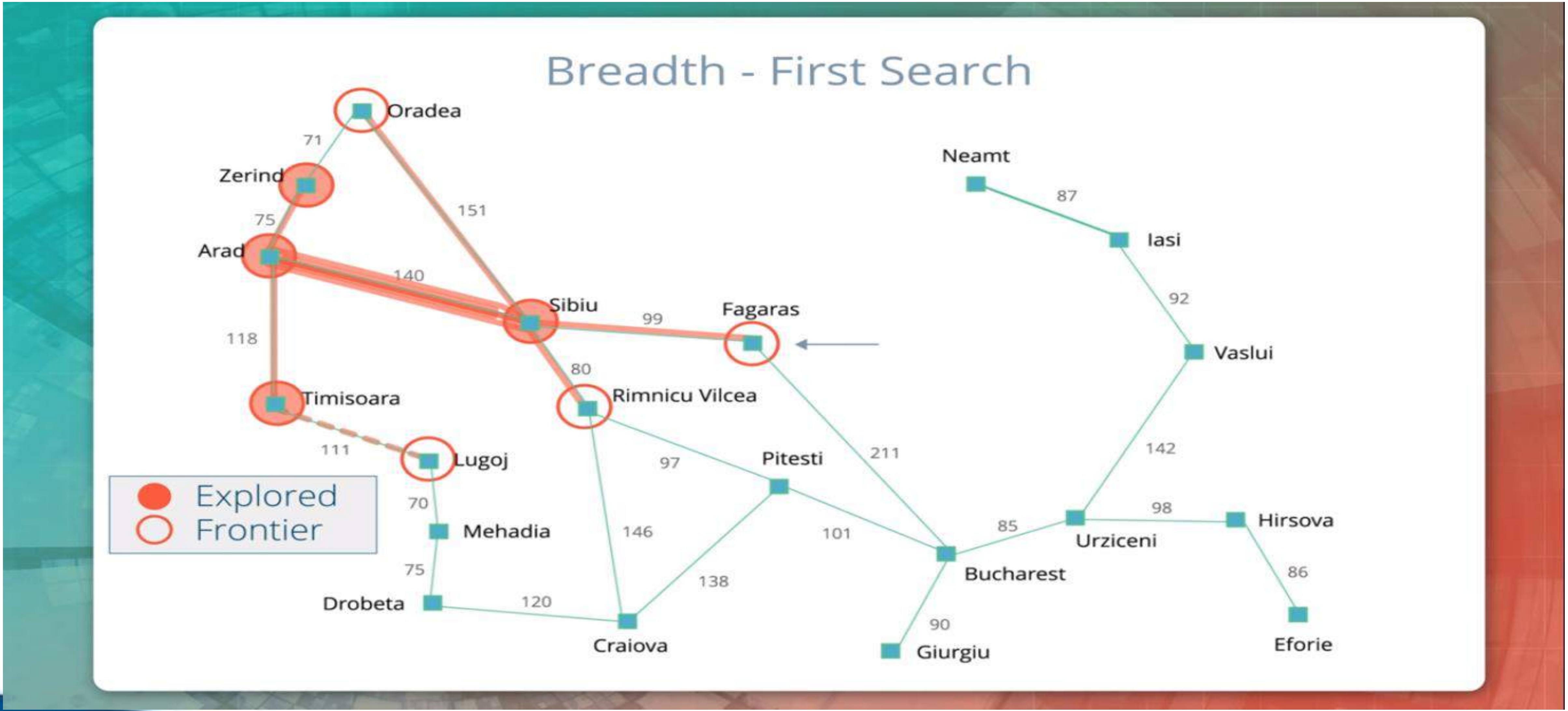
(c) After expanding Sibiu

- The way to avoid exploring redundant paths is to remember where one has been.
- To do this, we augment the TREE-SEARCH algorithm with a data structure called the **explored set** which remembers every expanded node.
- Newly generated nodes that match previously generated nodes—ones in the explored set or the frontier—can be discarded instead of being added to the frontier.
- The new algorithm, called GRAPH-SEARCH
- In GS the frontier **separates** the state-space graph into the explored region and the unexplored region, so that every path from the initial state to an unexplored state has to pass through a state in the frontier

Breadth FIRST SEARCH / Shortest FIRST SEARCH

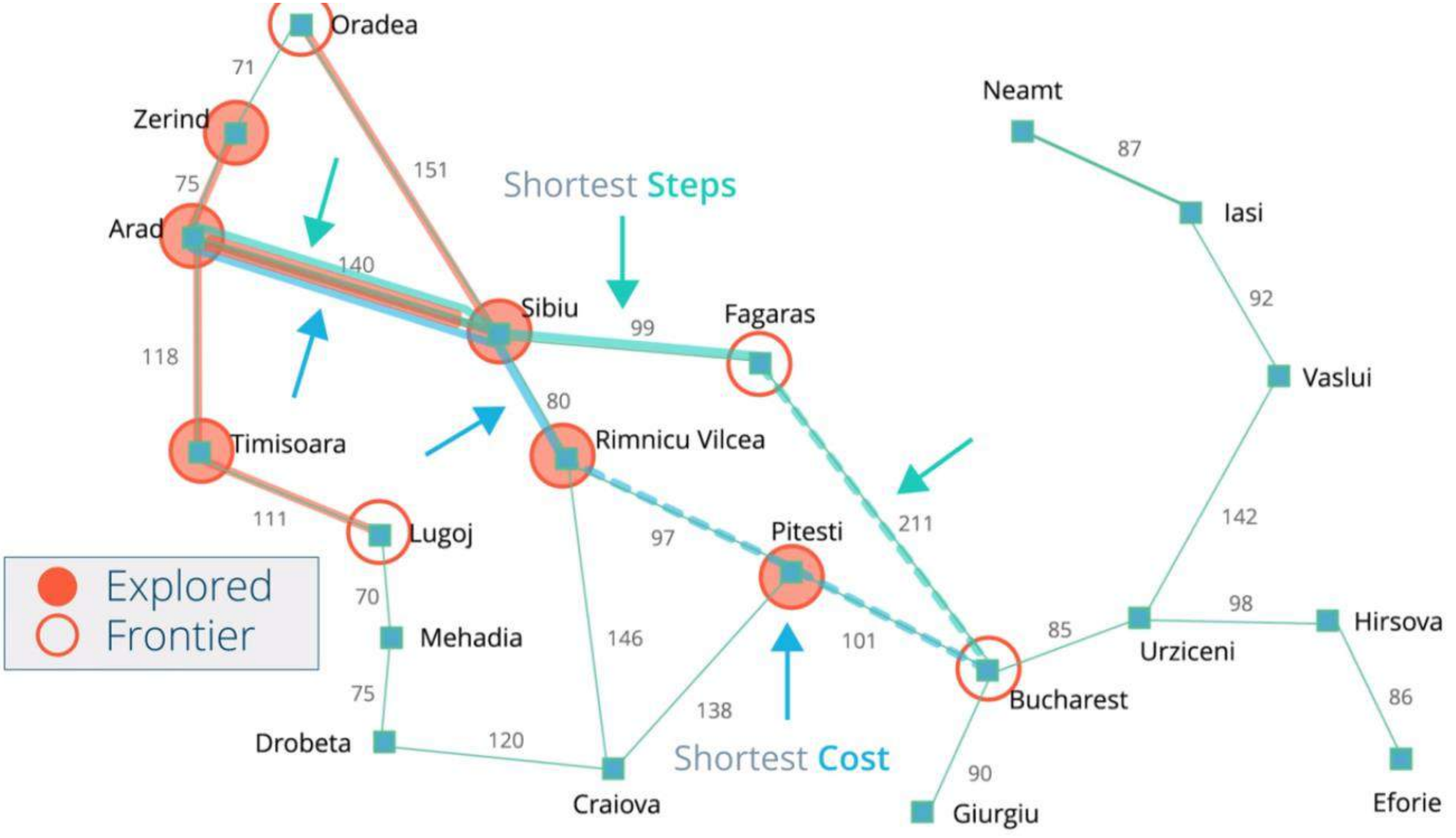


NEXT WHICH To Explore ????



Which Next ???

Do We Terminate ???



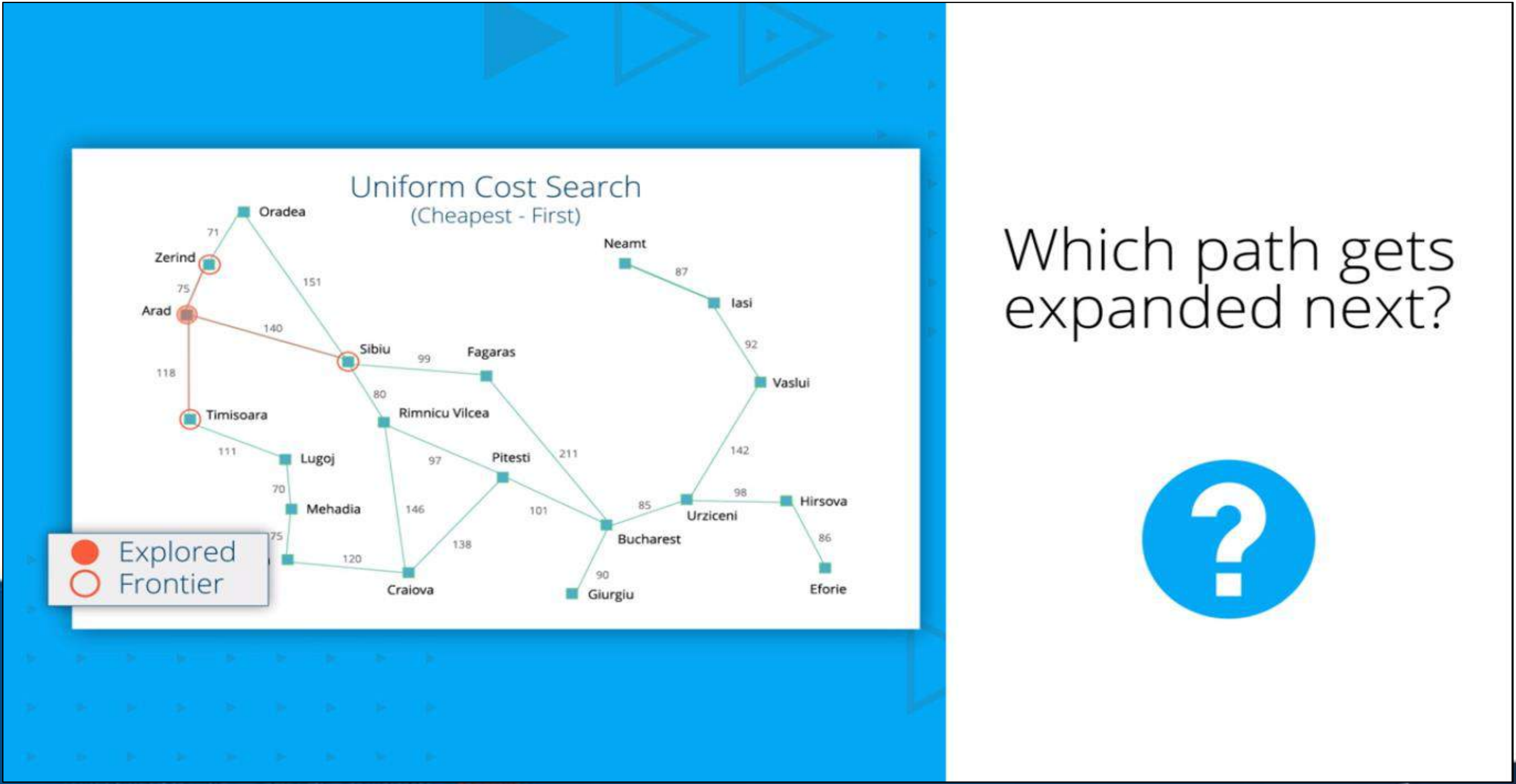
Shortest **Steps**

Shortest **Cost**

● Explored

○ Frontier

UNIFORM- COST SEARCH/Cheapest First



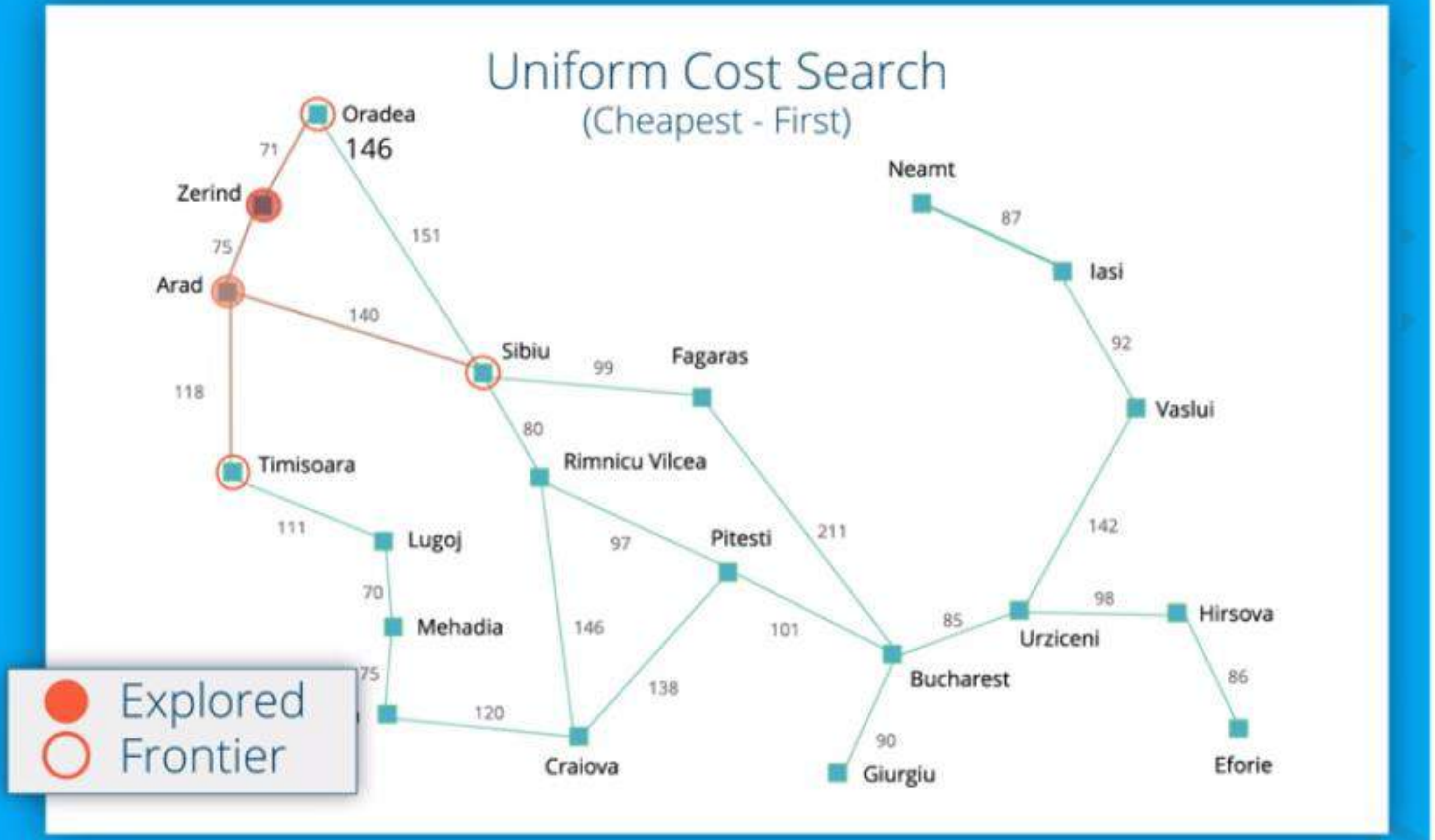
Uniform Cost Search

(Cheapest - First)

● Explored

○ Frontier

Which path gets expanded next?



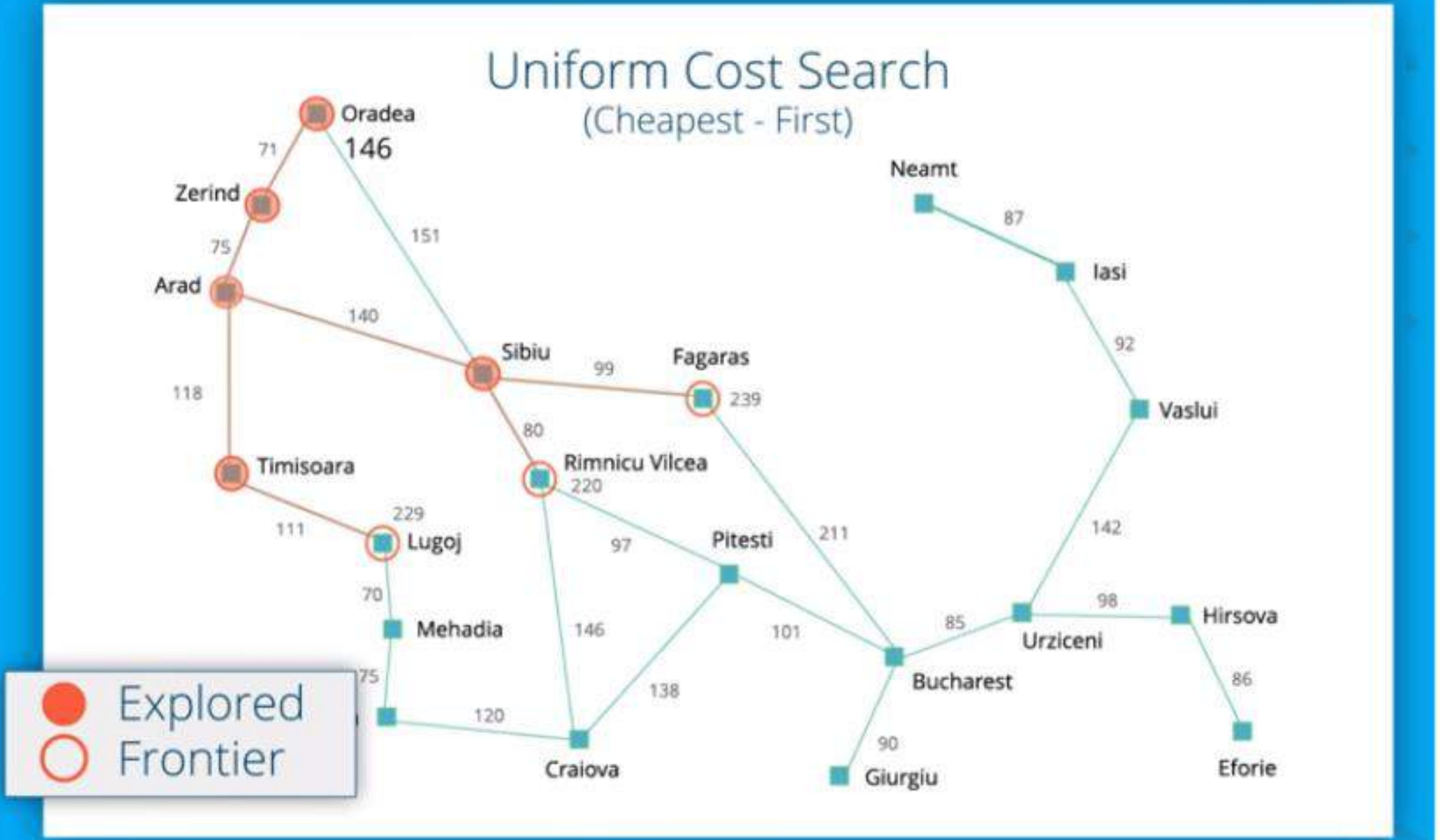
Uniform Cost Search

(Cheapest - First)

● Explored

○ Frontier

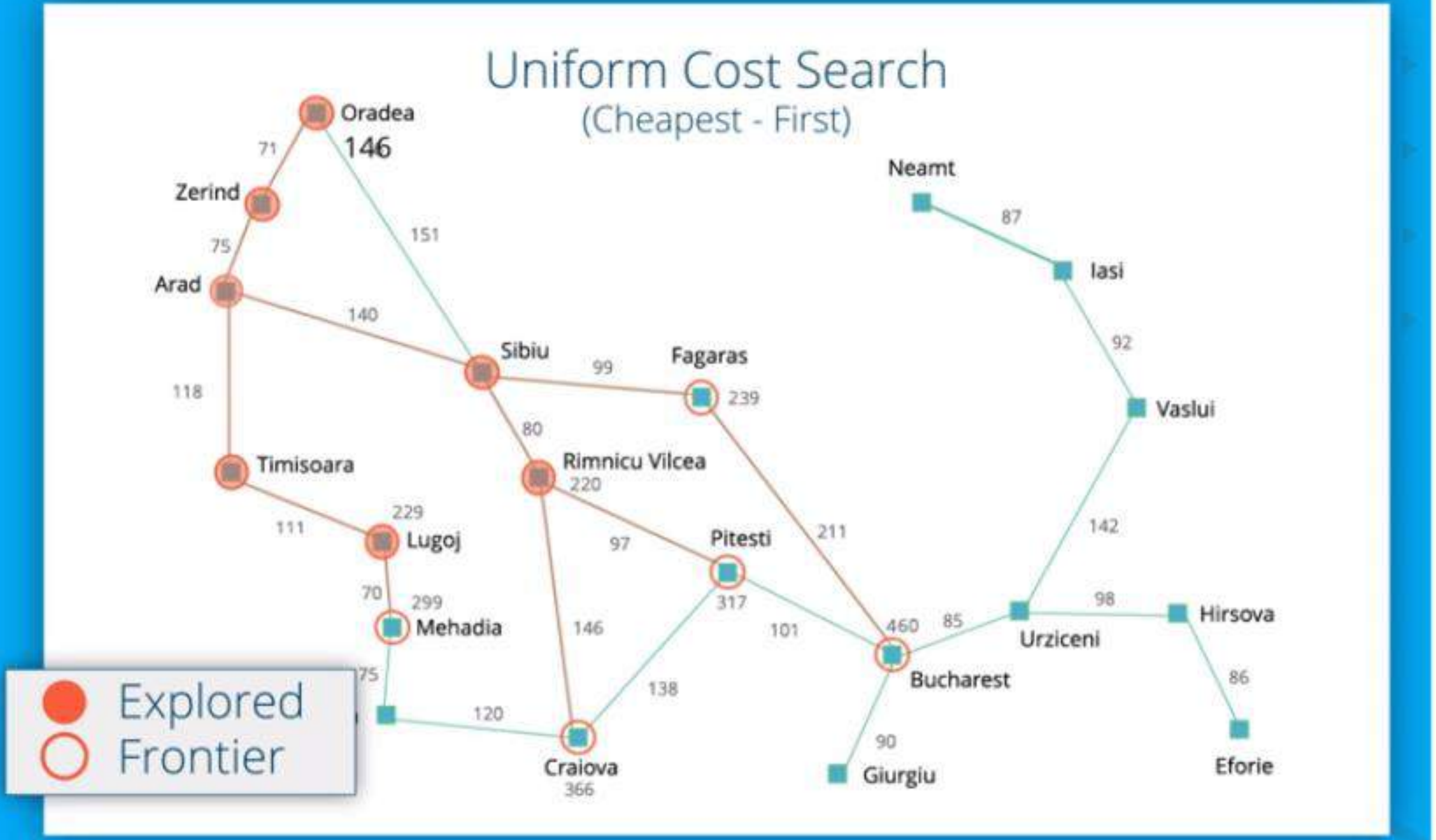
Which path gets expanded next?



Uniform Cost Search

(Cheapest - First)

Which path gets expanded next?



Uniform Cost Search

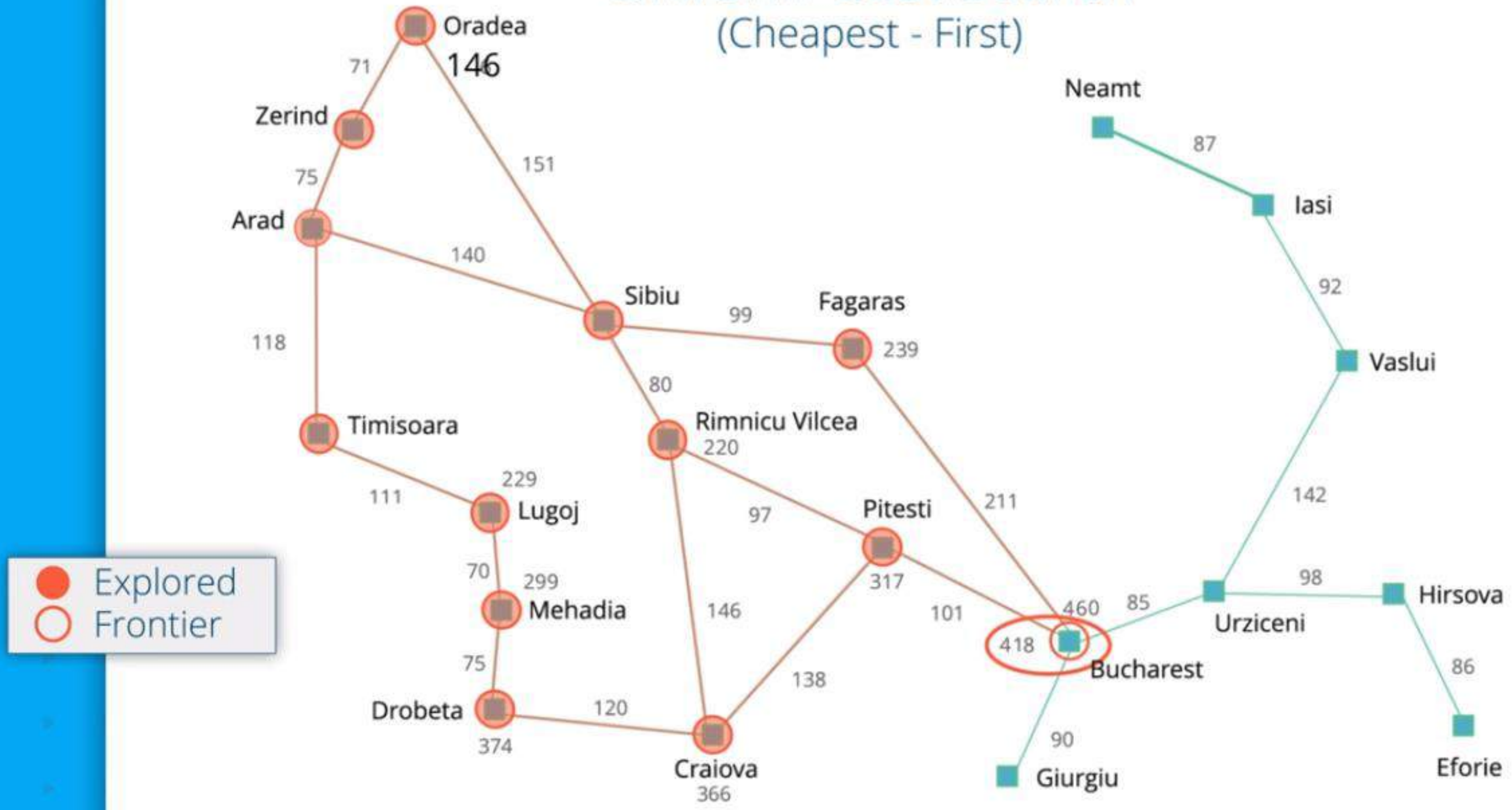
(Cheapest - First)

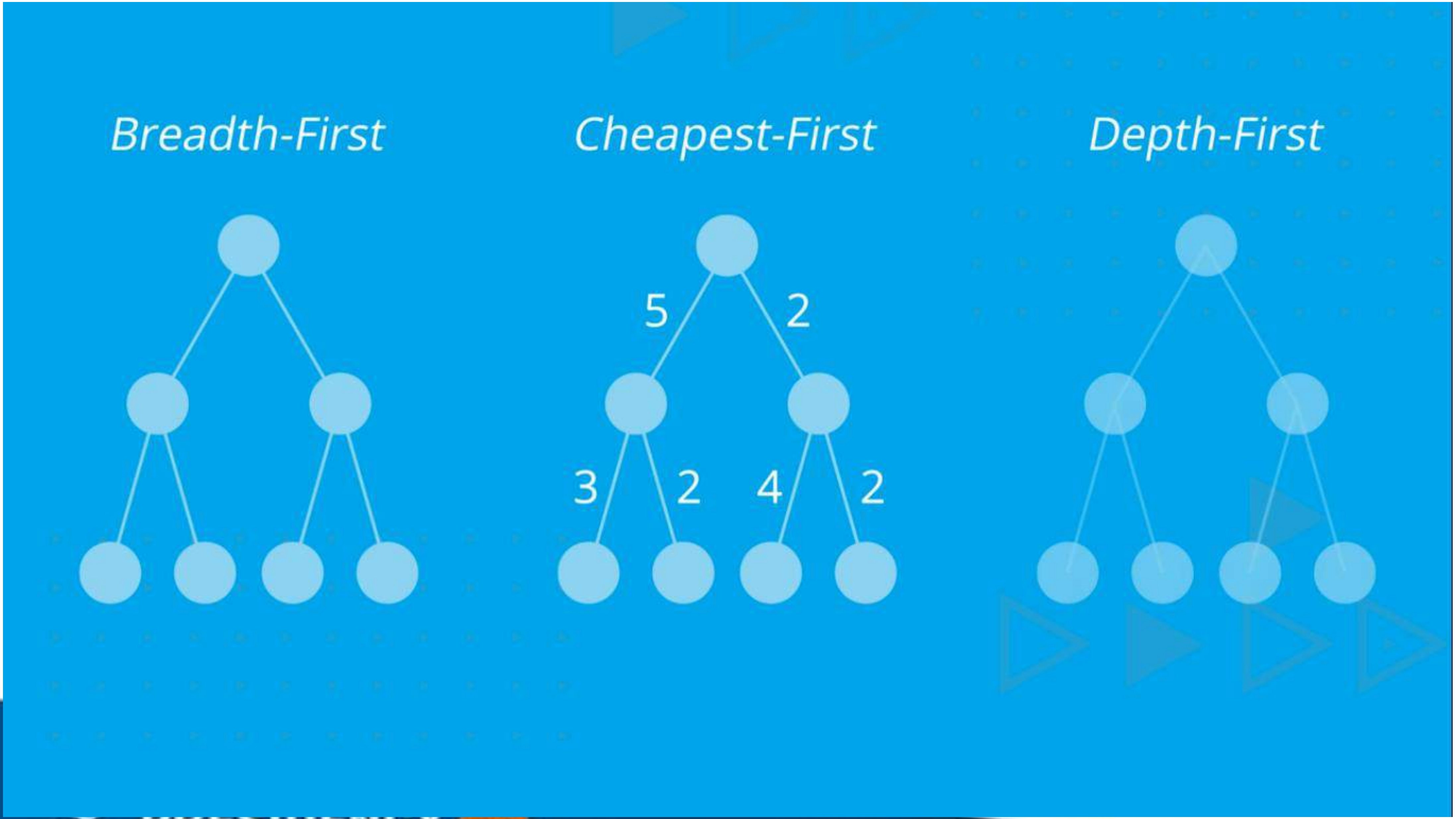
Can we terminate?

UNIFORM COST SEARCH CONTINUES TILL WE POP OFF THE PATH FROM THE FRONTIER

Uniform Cost Search

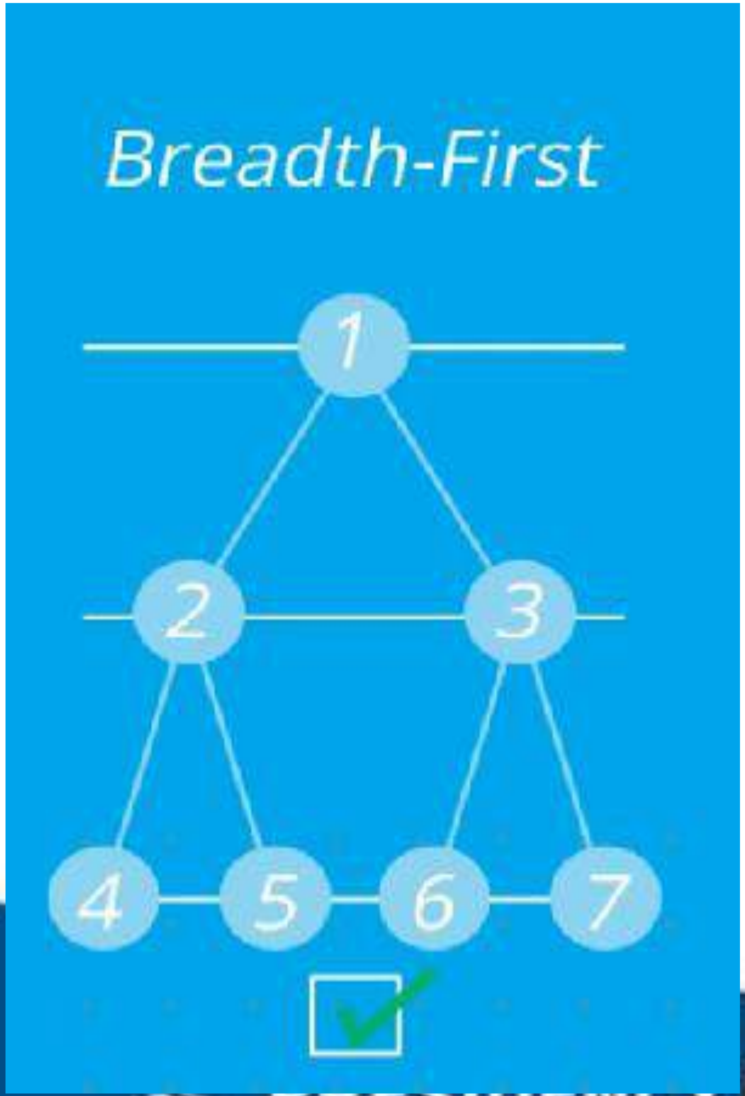
(Cheapest - First)



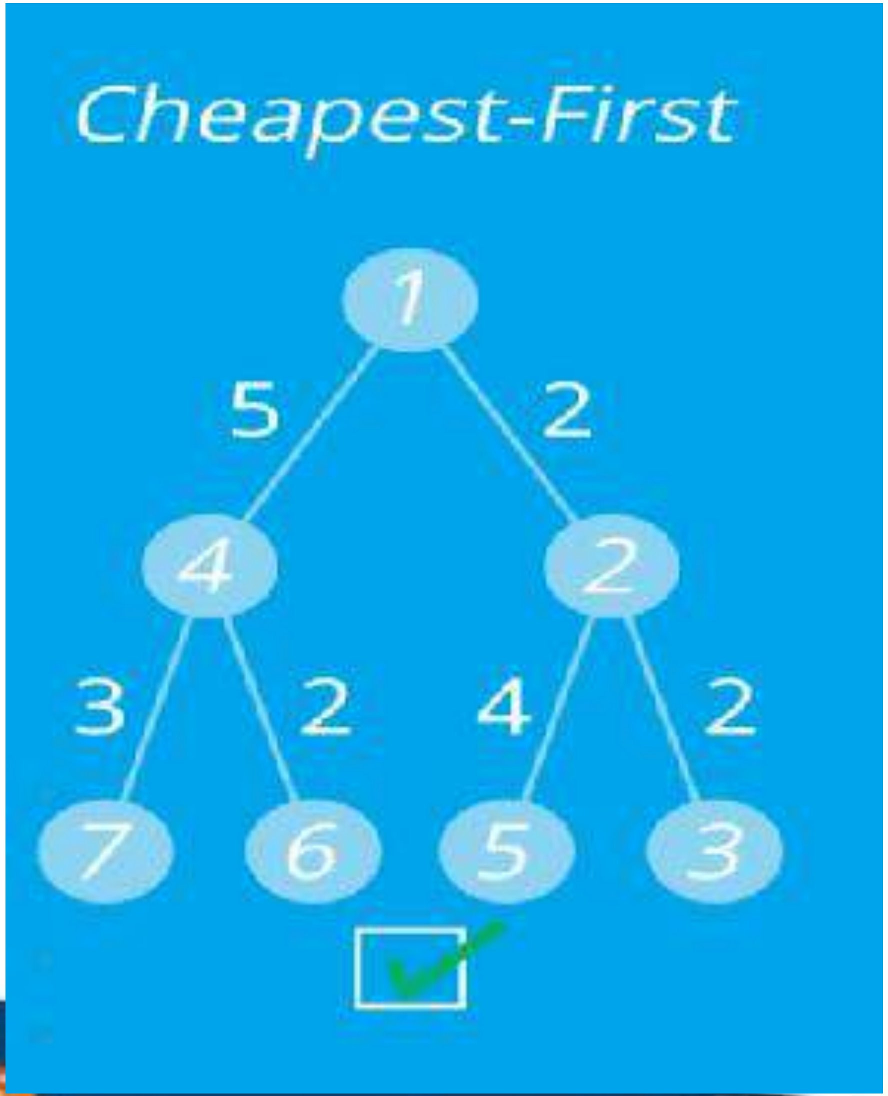


NUMBER IT IN ORDER OF PROCESSING

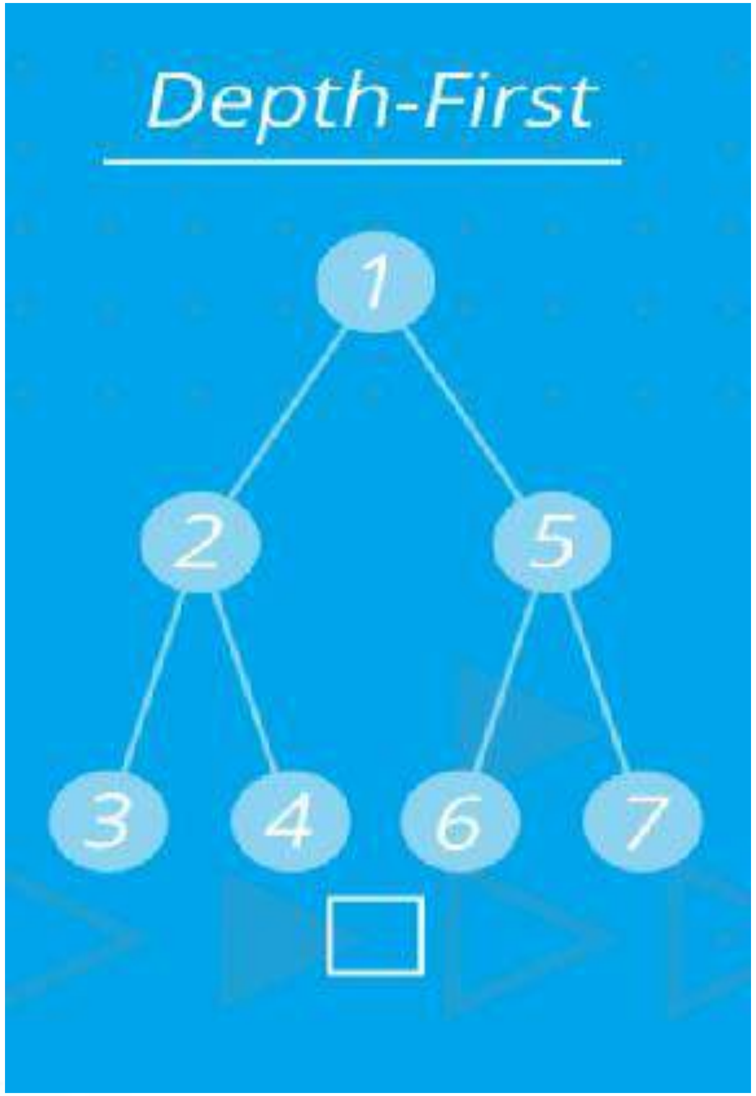
BREAK TIE FROM LEFT TO RIGHT



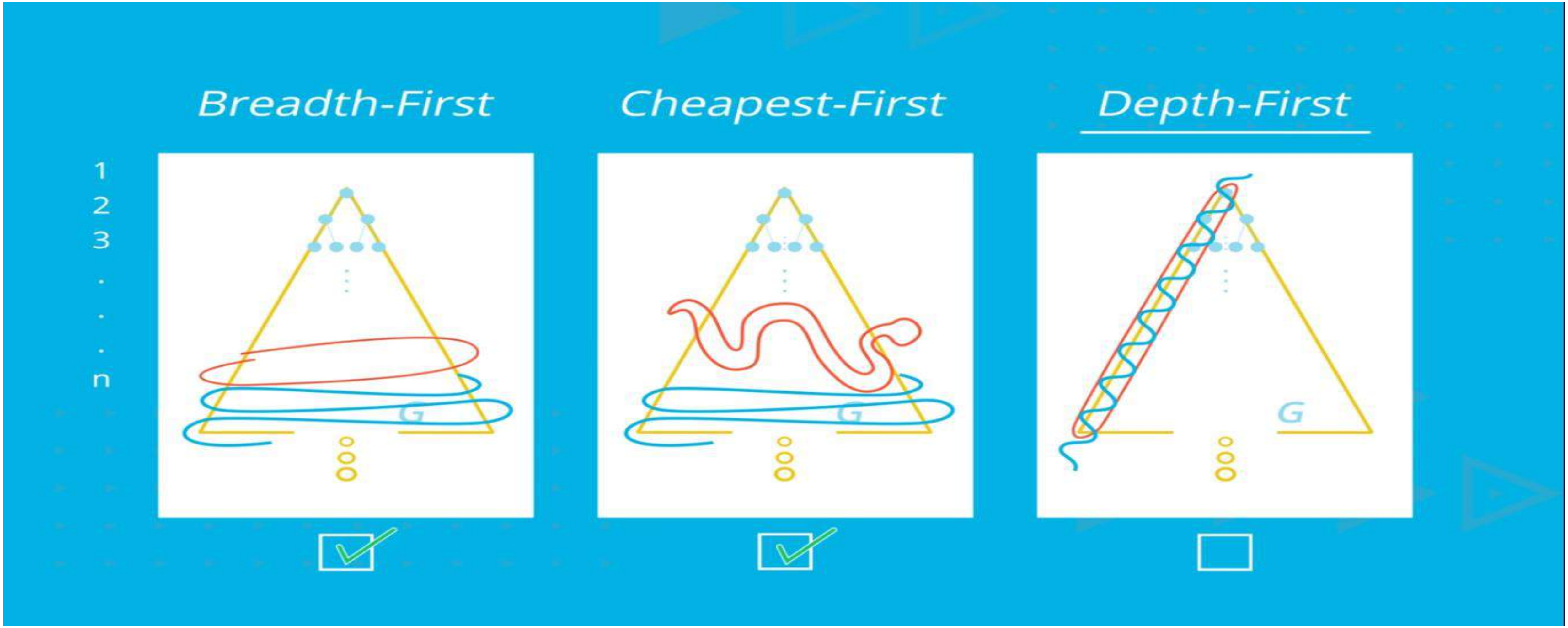
Breadth-First

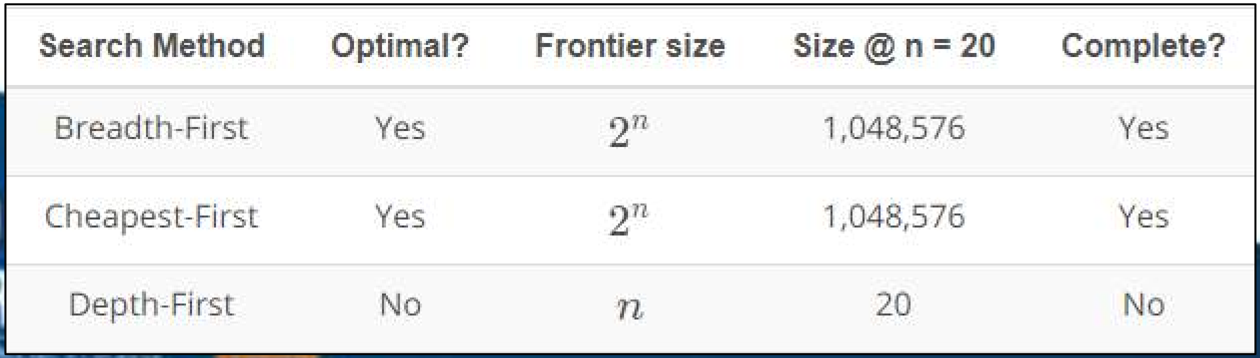


Cheapest-First



Depth-First

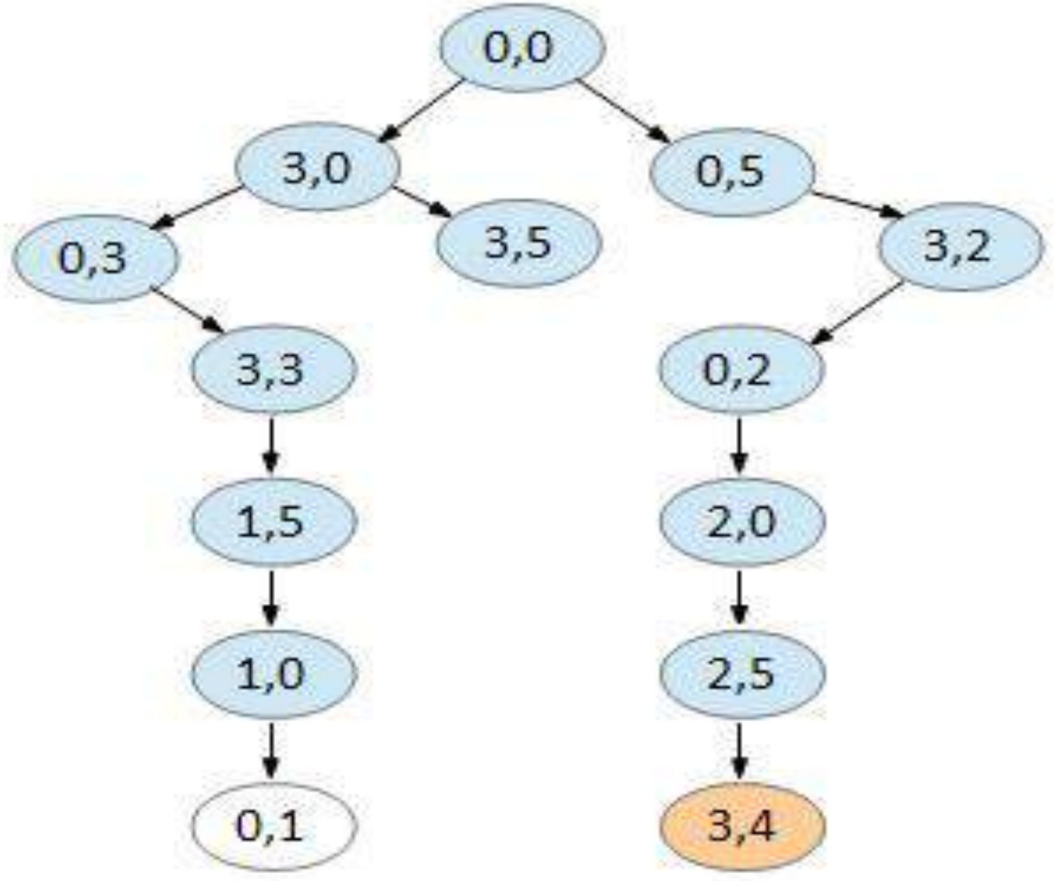




Search Method	Optimal?	Frontier size	Size @ n = 20	Complete?
Breadth-First	Yes	2^n	1,048,576	Yes
Cheapest-First	Yes	2^n	1,048,576	Yes
Depth-First	No	n	20	No

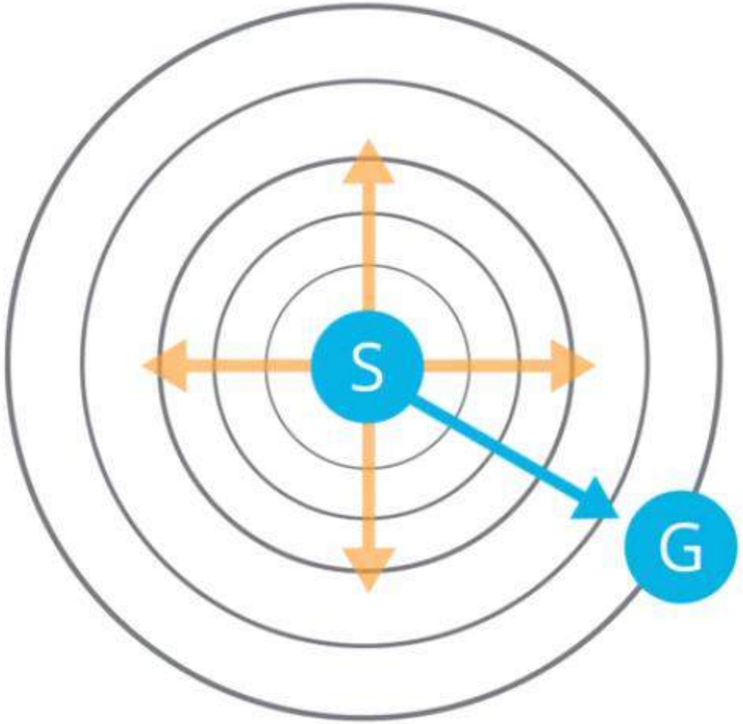
“You are given a 5 litre jug and a 3 litre jug. Both the jugs are initially empty. The jugs don’t have markings to allow measuring smaller quantities. You have to use the jugs to collect 4 litres of water.”

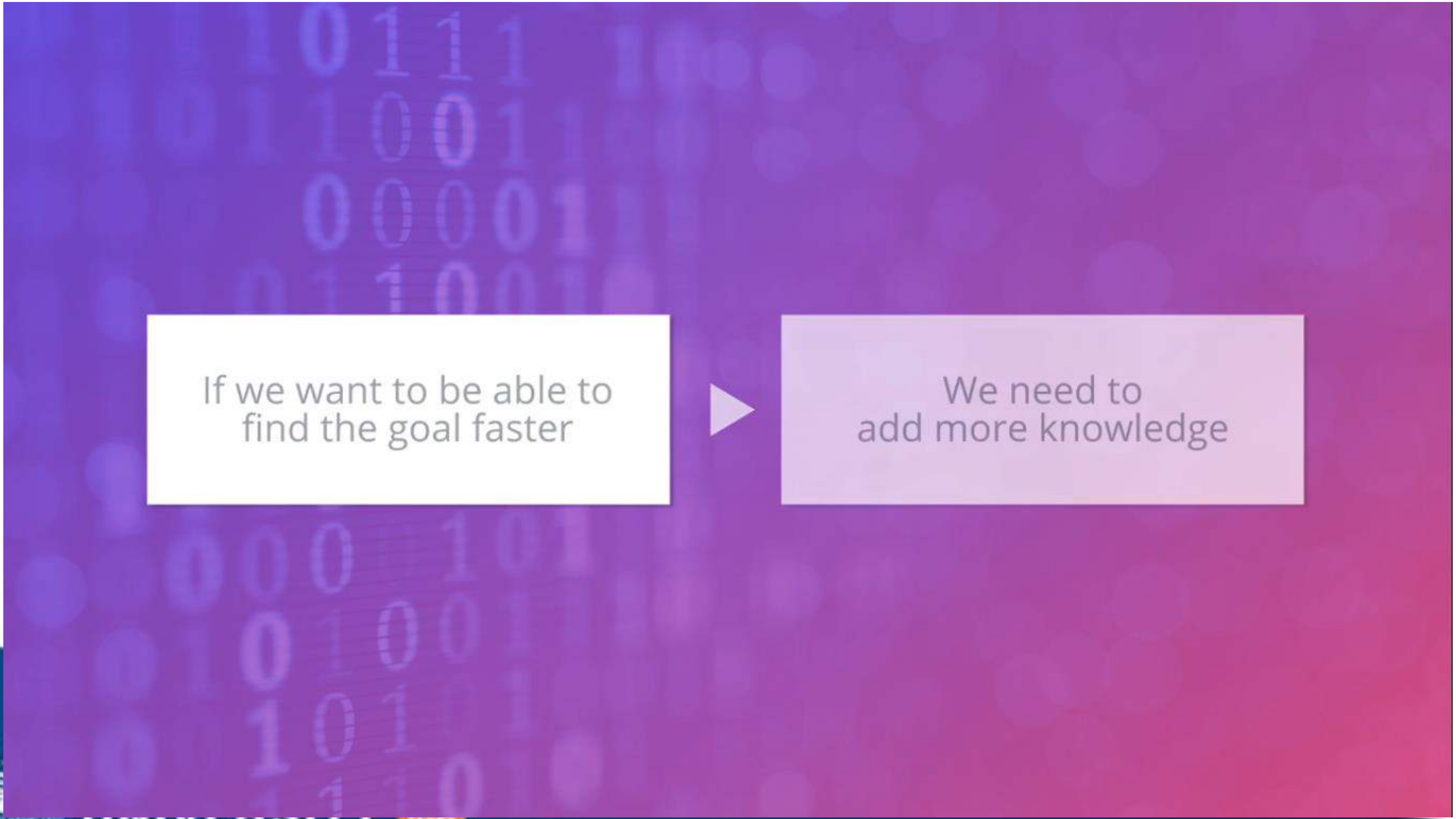
Solve the water jug problem given below using BFS. Specify the state space, the rules applied and the final solution.



INFORMED SEARCH / (HEURISTIC SEARCH)

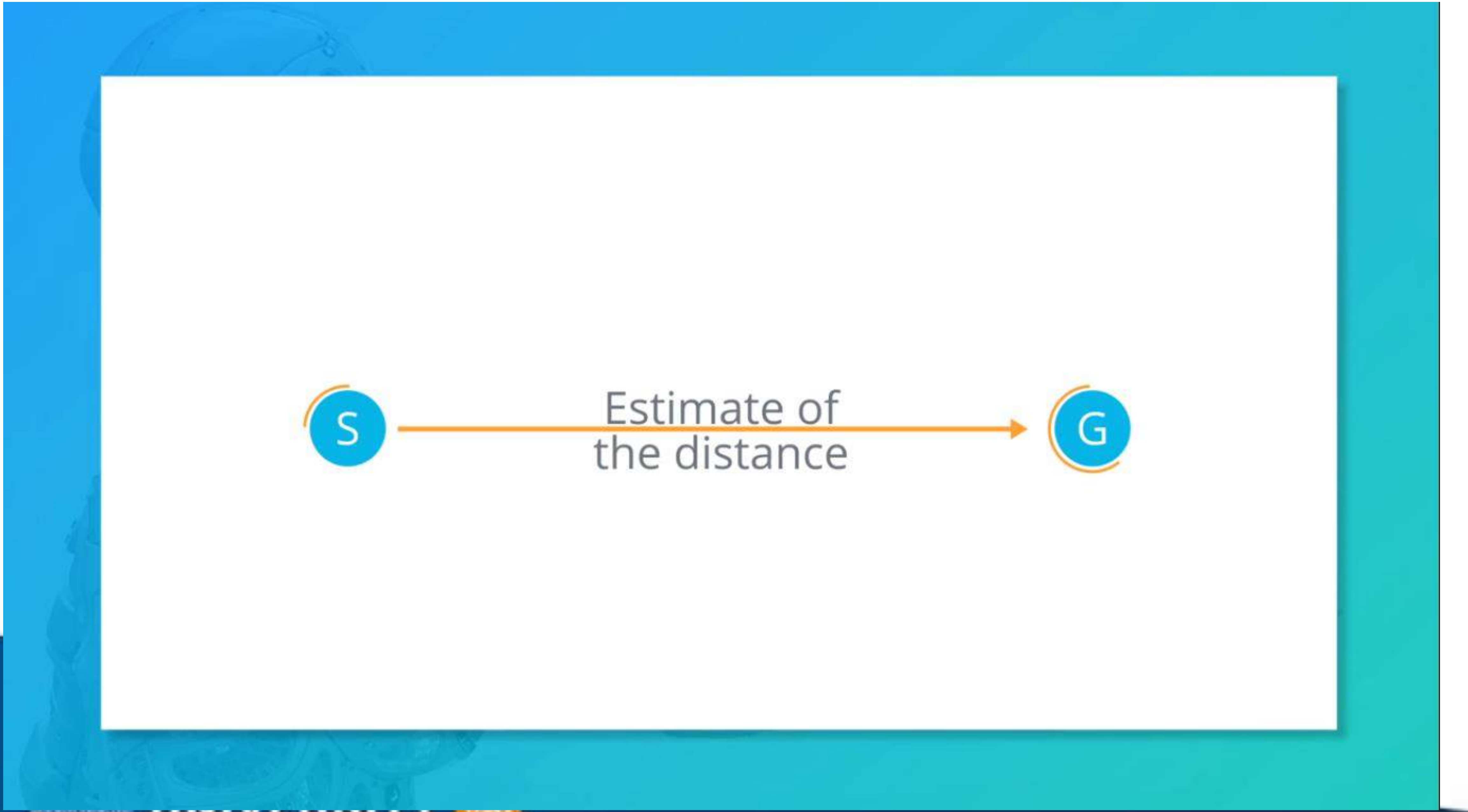
- These strategies have additional information about search states, so they can guide the search by ranking successors according to some fitness score until they find a goal state.
- **Heuristics function:** Heuristic is a function which is used in Informed Search, and it finds the most promising path.
- **Greedy best-first search**
- **A* Search**


S
G

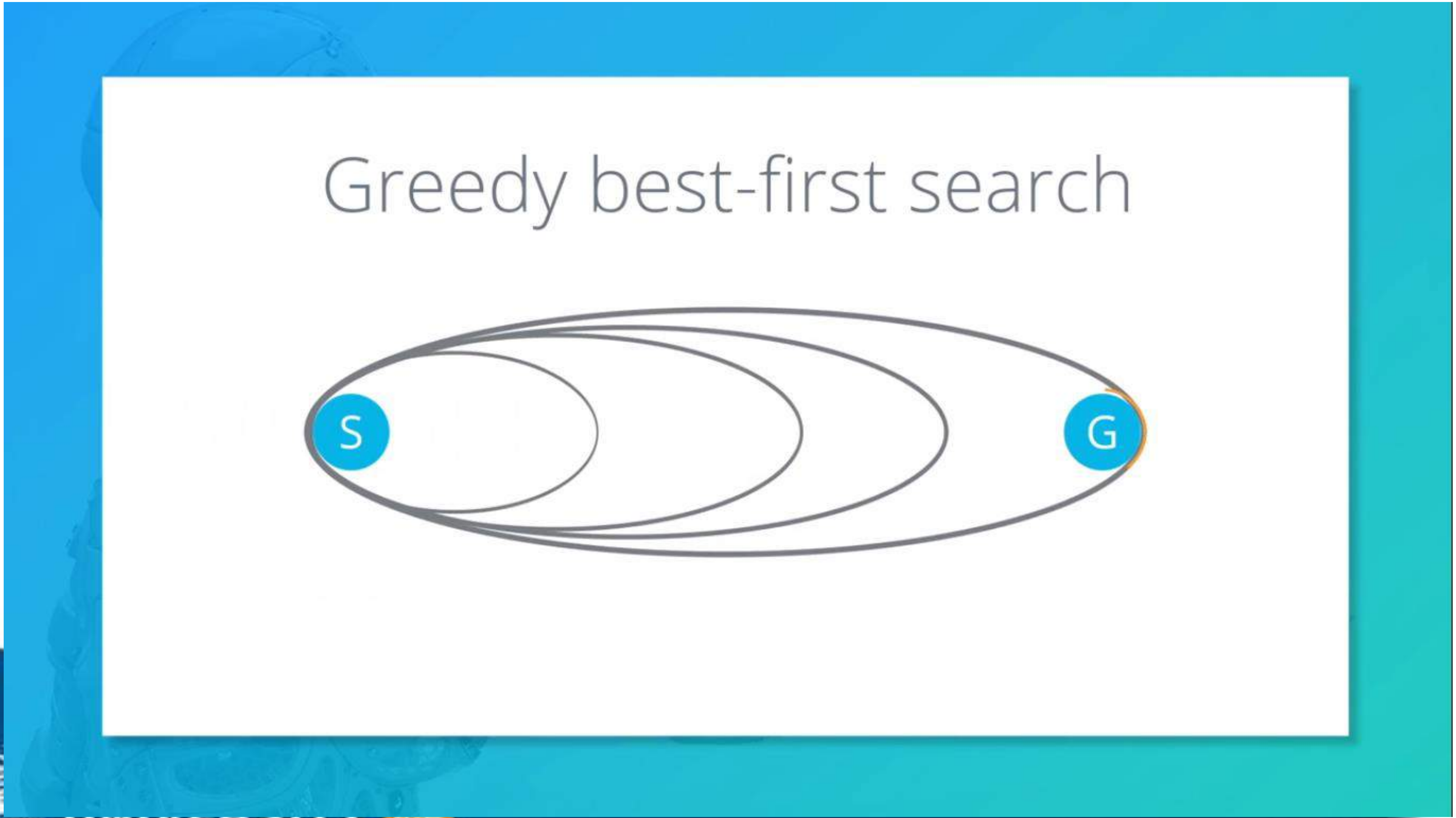


If we want to be able to find the goal faster

We need to add more knowledge



Estimate of the distance



Greedy best-first search

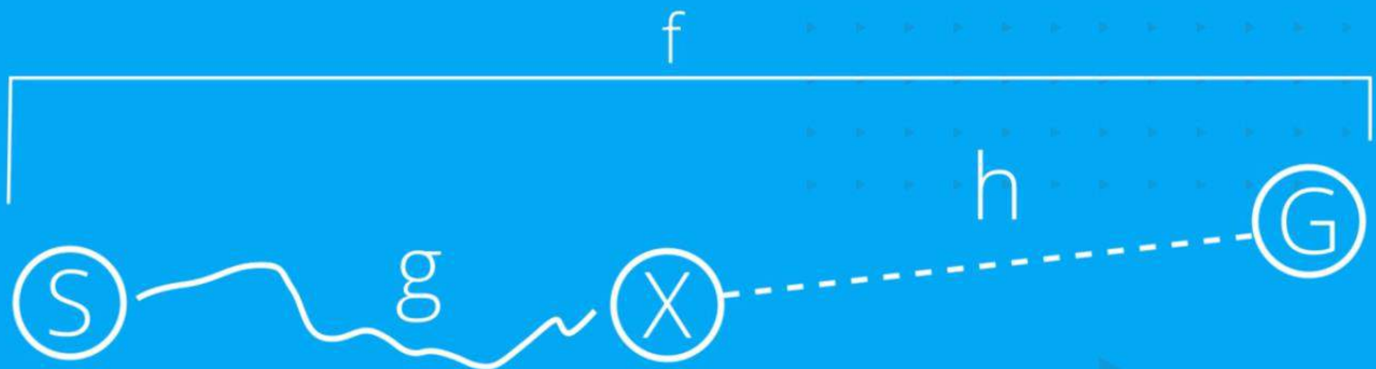
A* Search

Minimum value

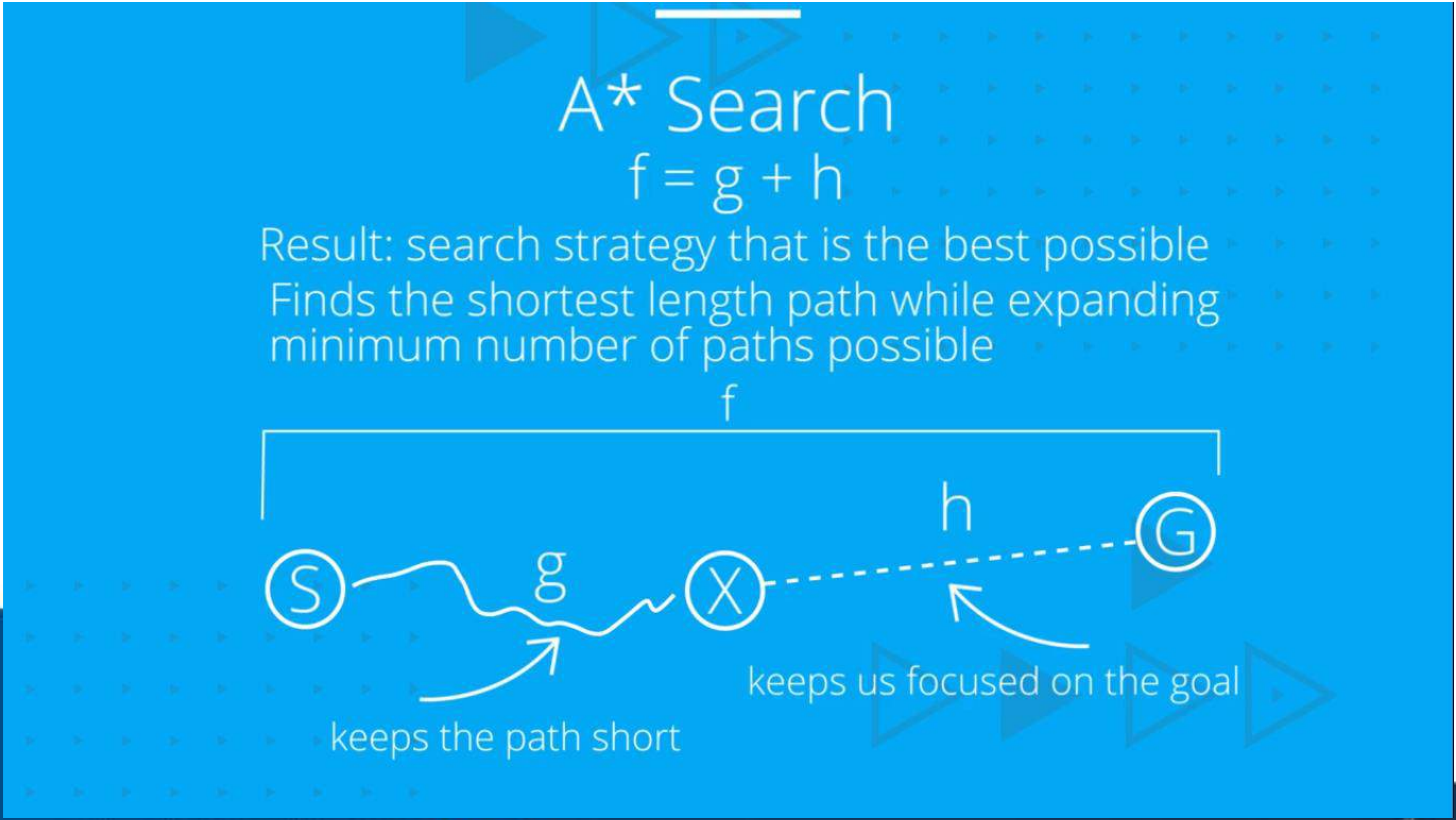
f = g + h

g (path) = path cost

h (path) = h(s) =
estimated distance to goal



BEST ESTIMATED TOTAL PATH COST FIRST



A* Search

$$f = g + h$$

Result: search strategy that is the best possible

Finds the shortest length path while expanding minimum number of paths possible

keeps the path short

keeps us focused on the goal

A* Algorithm

- A* Algorithm is one of the best and popular techniques used for path finding and graph traversals.
- A lot of games and web-based maps use this algorithm for finding the shortest path efficiently.
- It is essentially a best first search algorithm.

Working-

- It maintains a tree of paths originating at the start node.
- It extends those paths one edge at a time.
- It continues until its termination criterion is satisfied.

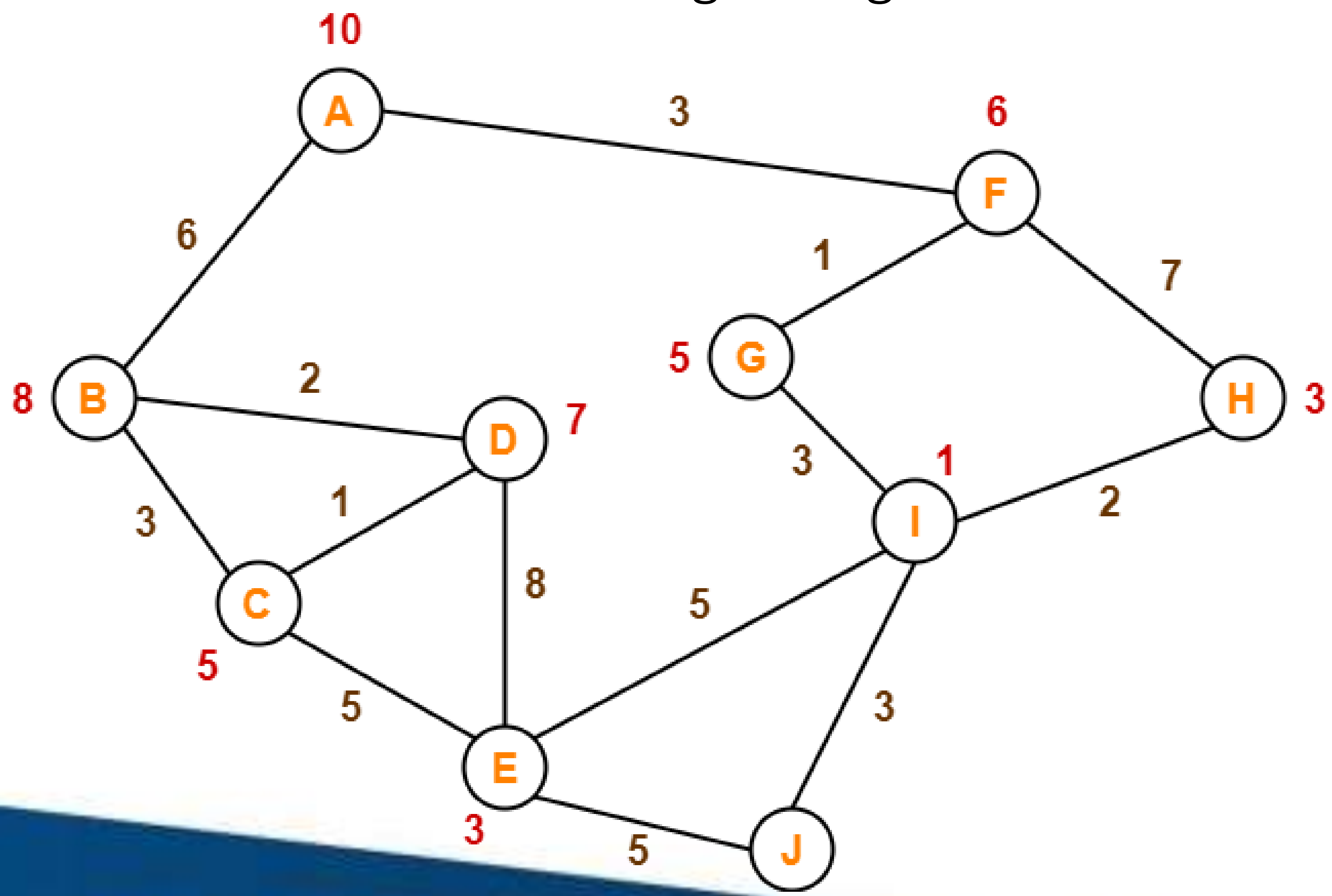
- A* Algorithm extends the path that minimizes the following function-

$$f(n) = g(n) + h(n)$$

Here,

- ‘n’ is the node on the path
- g(n) is the cost of the path from start node to node ‘n’
- h(n) is a heuristic function that estimates cost of the cheapest path from node ‘n’ to the goal node

Find the most cost-effective path to reach from start state A to final state J using A* Algorithm



Step-01:

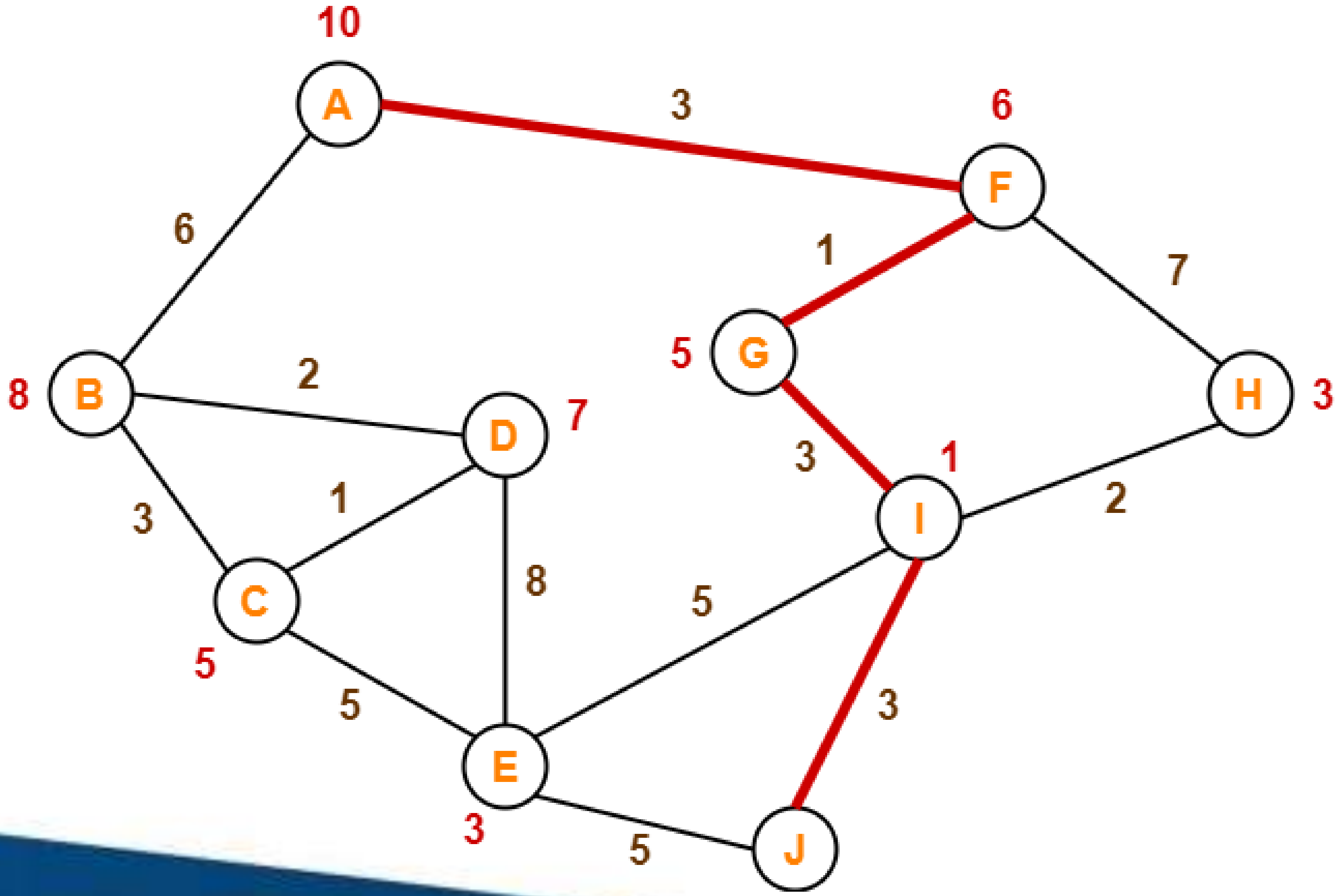
- We start with node A.
- Node B and Node F can be reached from node A.
- A* Algorithm calculates f(B) and f(F).
- f(B) = 6 + 8 = 14
- f(F) = 3 + 6 = 9
- Since f(F) < f(B), so it decides to go to node F.
- **Path- A → F**
- **Step-02:**
- Node G and Node H can be reached from node F.
- A* Algorithm calculates f(G) and f(H).
- f(G) = (3+1) + 5 = 9
- f(H) = (3+7) + 3 = 13
- Since f(G) < f(H), so it decides to go to node G.
- **Path- A → F → G**

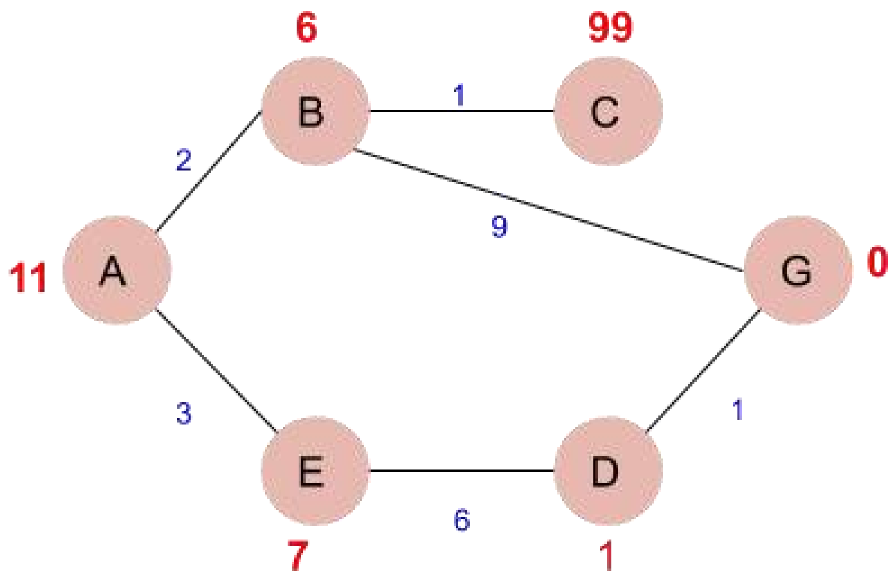
Step-03:

- Node I can be reached from node G.
- A* Algorithm calculates f(I).
- f(I) = (3+1+3) + 1 = 8
- It decides to go to node I.
- **Path- A → F → G → I**

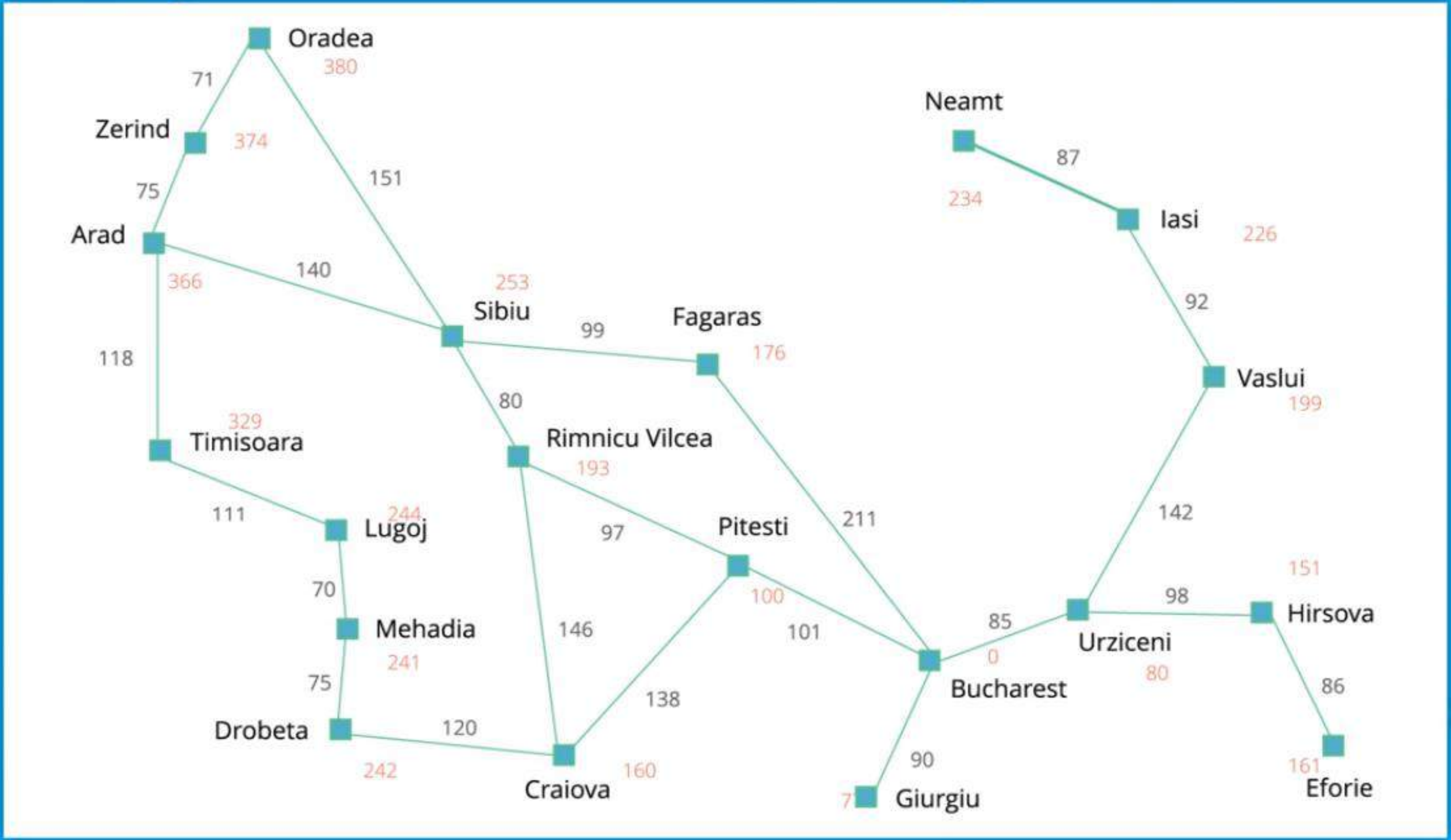
Step-04:

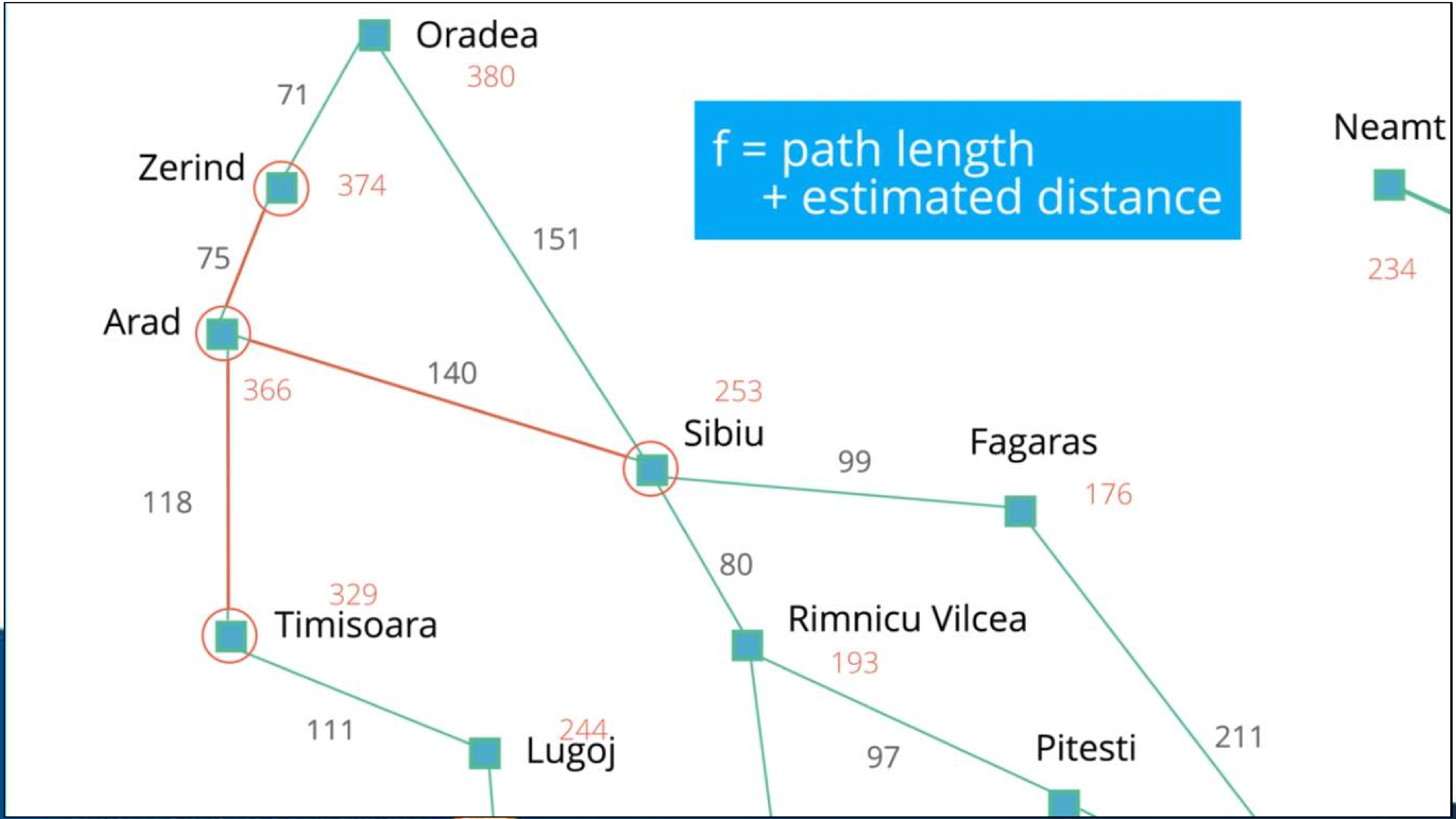
- Node E, Node H and Node J can be reached from node I.
- A* Algorithm calculates f(E), f(H) and f(J).
- f(E) = (3+1+3+5) + 3 = 15
- f(H) = (3+1+3+2) + 3 = 12
- f(J) = (3+1+3+3) + 0 = 10
- Since f(J) is least, so it decides to go to node J.
- **Path- A → F → G → I → J**
- This is the required shortest path from node A to node J.


10
A
3
6
F
6
1
7
8
B
2
D
7
5
G
H
3
3
1
3
1
I
2
8
5
C
5
5
3
E
3
5
J


6
99
B
1
C
2
9
11
A
G
0
3
1
E
6
D
7
1

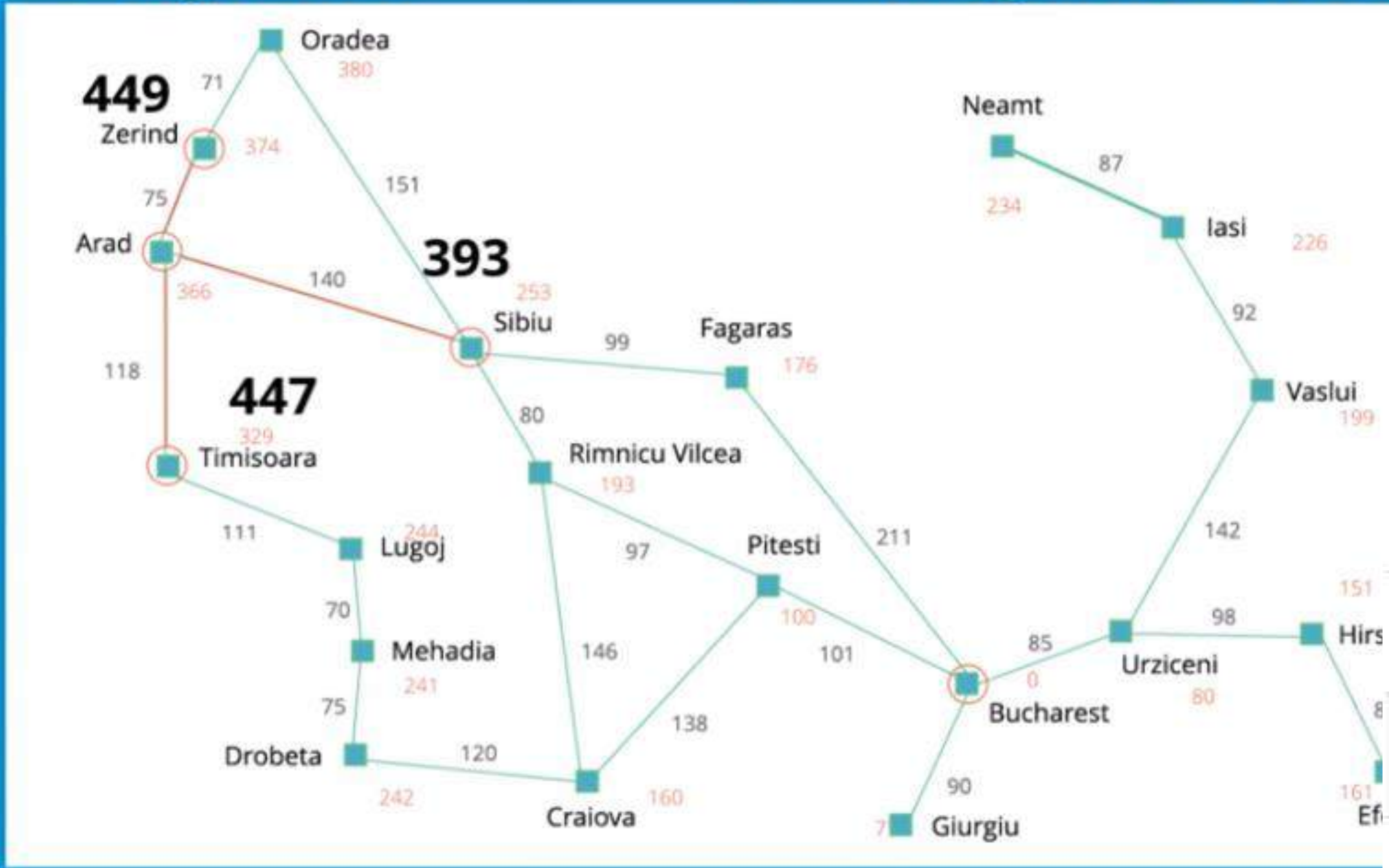
A* algorithm h = straight line distance



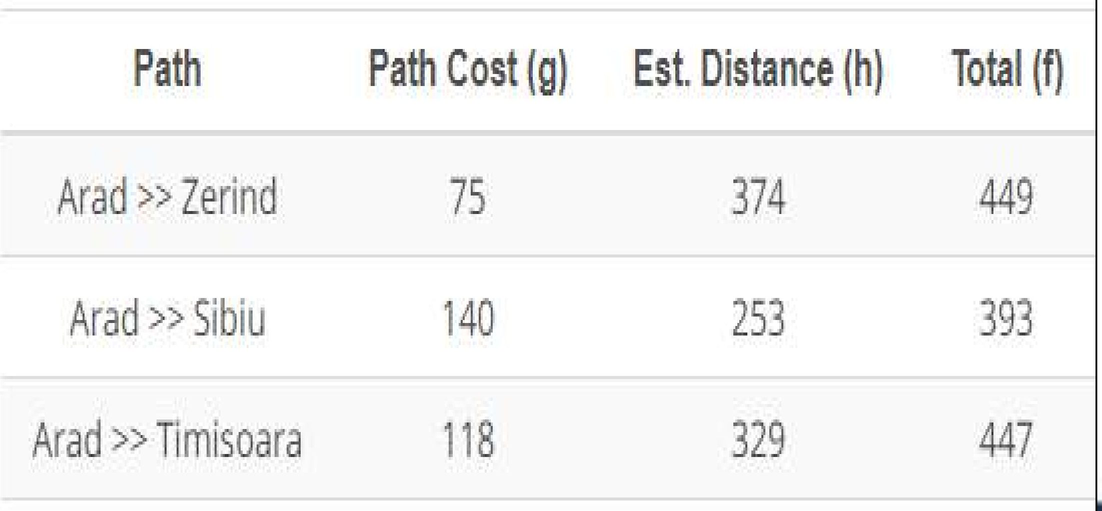


f = path length
+ estimated distance

A* algorithm h = straight line distance



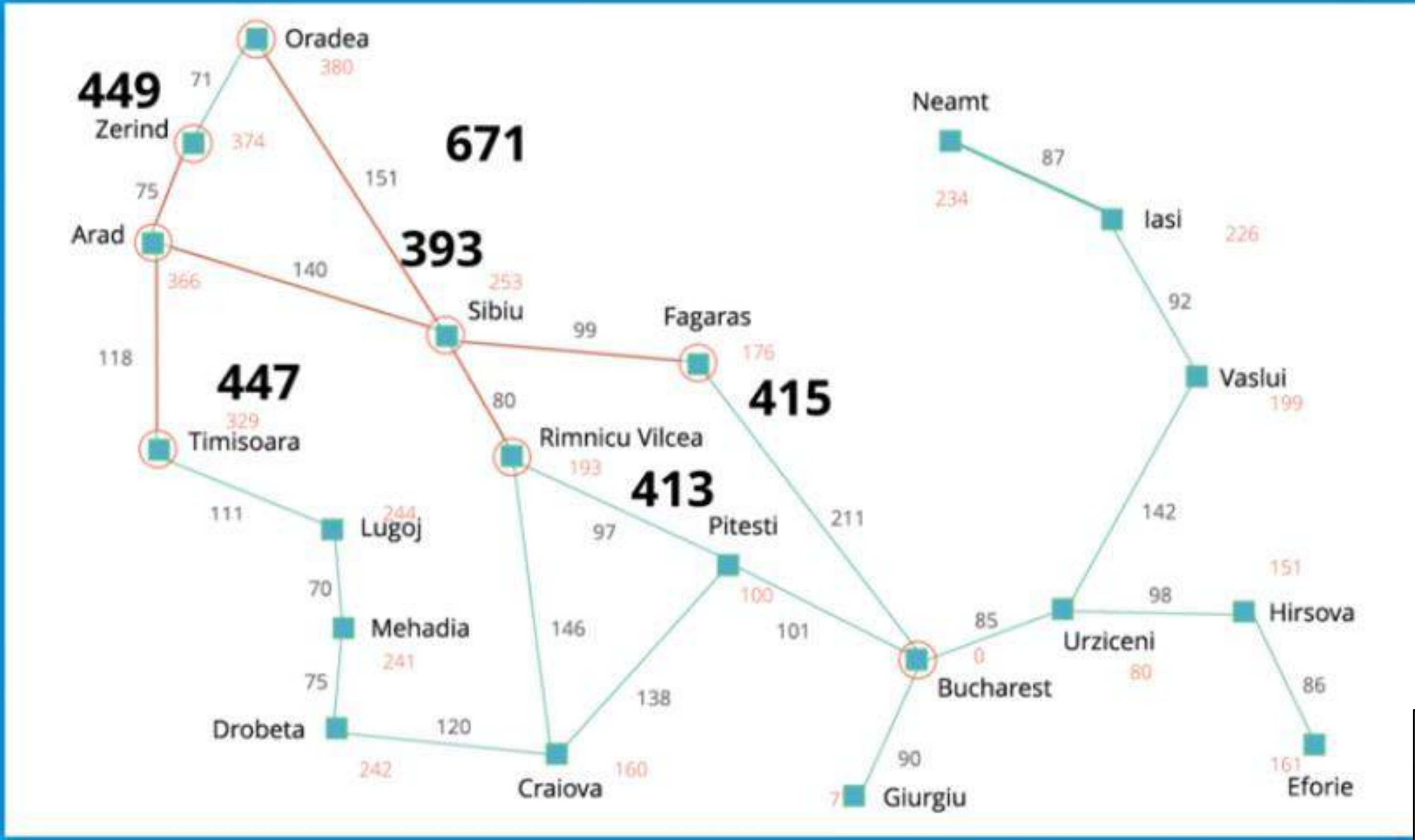
Which path gets expanded next?



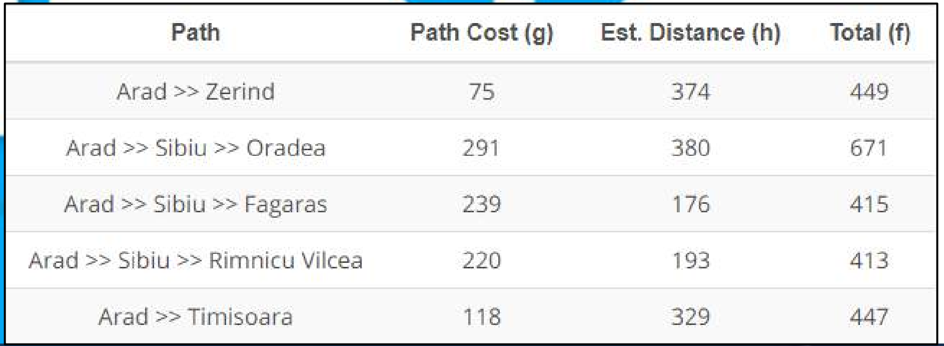
Path	Path Cost (g)	Est. Distance (h)	Total (f)
Arad >> Zerind	75	374	449
Arad >> Sibiu	140	253	393
Arad >> Timisoara	118	329	447

A* algorithm

h = straight line distance



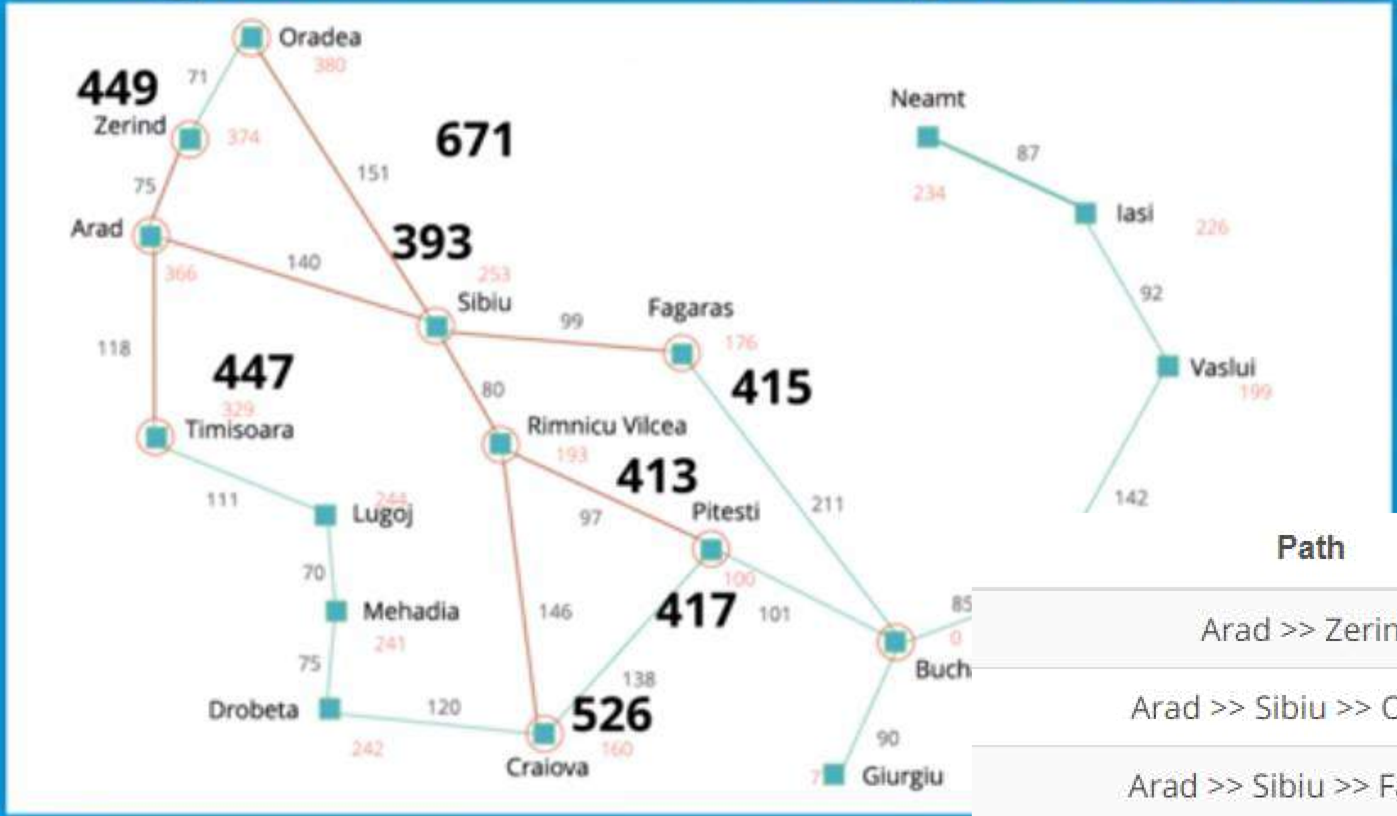
Which path gets expanded next?



Path	Path Cost (g)	Est. Distance (h)	Total (f)
Arad >> Zerind	75	374	449
Arad >> Sibiu >> Oradea	291	380	671
Arad >> Sibiu >> Fagaras	239	176	415
Arad >> Sibiu >> Rimnicu Vilcea	220	193	413
Arad >> Timisoara	118	329	447

A* algorithm

h = straight line distance

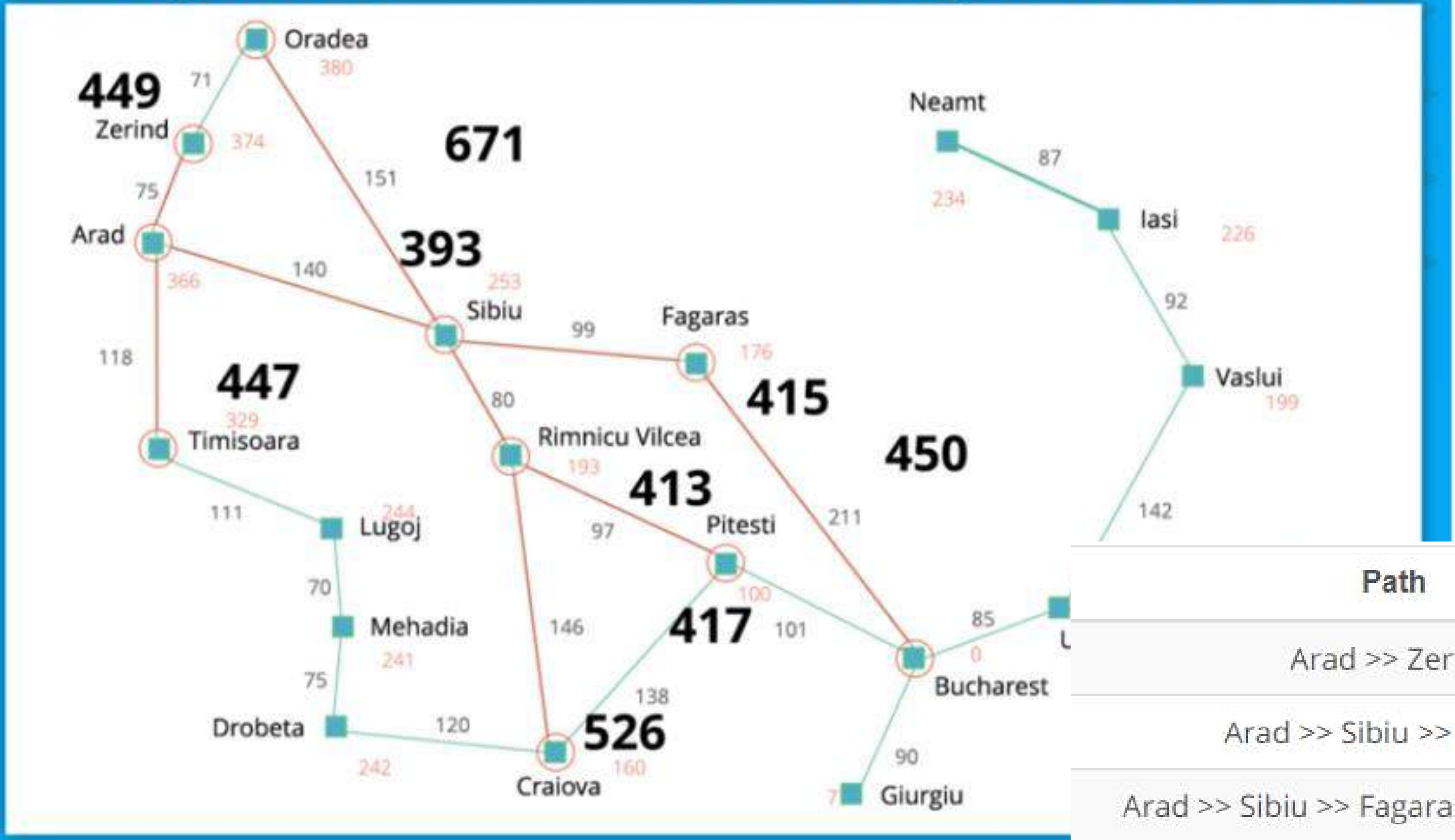


Which path gets expanded next?

Path	Path Cost (g)	Est. Distance (h)	Total (f)
Arad >> Zerind	75	374	449
Arad >> Sibiu >> Oradea	291	380	671
Arad >> Sibiu >> Fagaras	239	176	415
Arad >> Sibiu >> Rimnicu Vilcea >> Pitesti	317	100	417
Arad >> Sibiu >> Rimnicu Vilcea >> Craiova	366	160	526
Arad >> Timisoara	118	329	447

A* algorithm

h = straight line distance

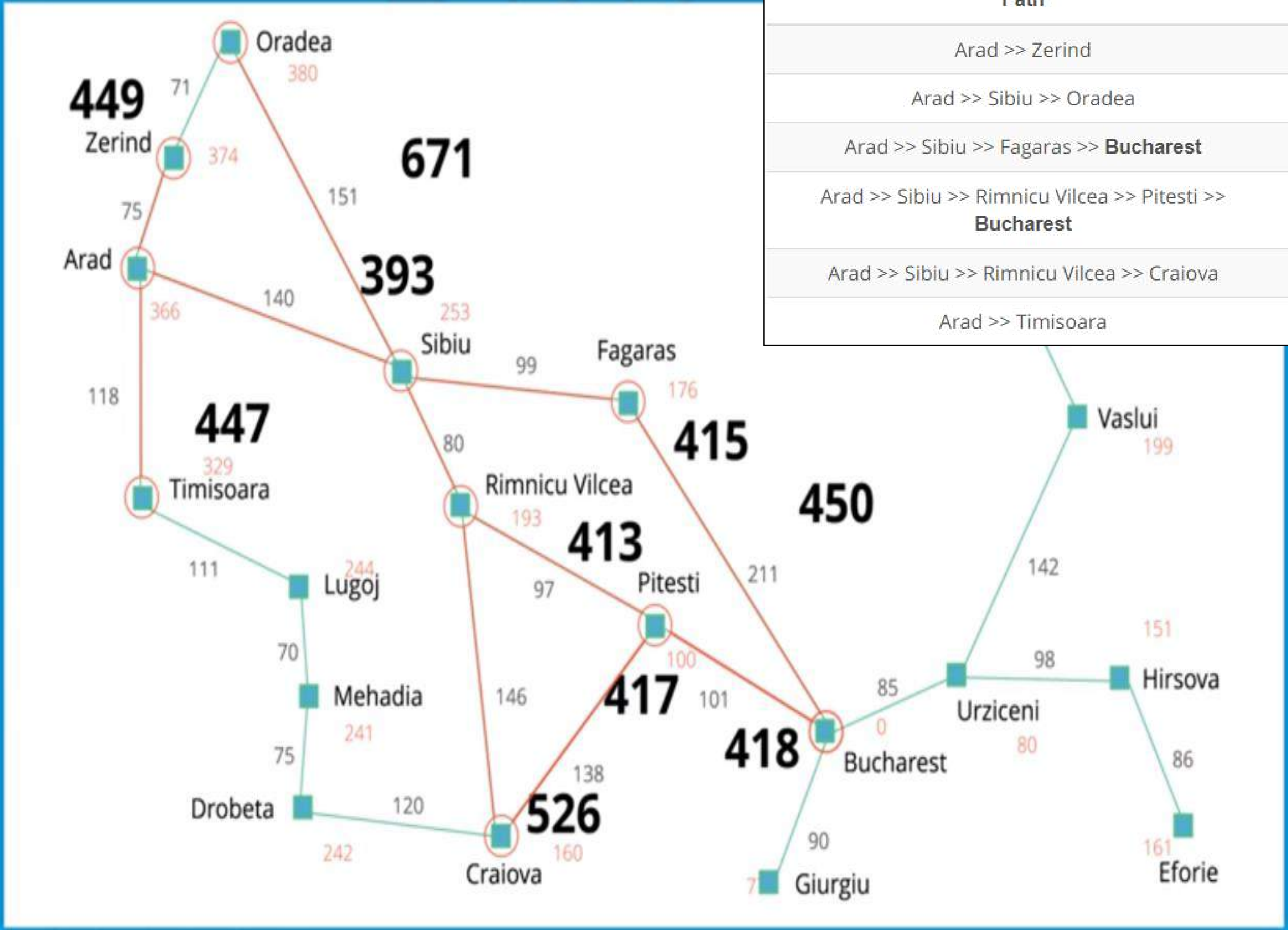


Goal test works by taking paths off the frontier, not putting paths on the frontier.

Which path gets expanded next?

Path	Path Cost (g)	Est. Distance (h)	Total (f)
Arad >> Zerind	75	374	449
Arad >> Sibiu >> Oradea	291	380	671
Arad >> Sibiu >> Fagaras >> **Bucharest**	450	0	450
Arad >> Sibiu >> Rimnicu Vilcea >> Pitesti	317	100	417
Arad >> Sibiu >> Rimnicu Vilcea >> Craiova	366	160	526
Arad >> Timisoara	118	329	447

Path	Path Cost (g)	Est. Distance (h)	Total (f)
Arad >> Zerind	75	374	449
Arad >> Sibiu >> Oradea	291	380	671
Arad >> Sibiu >> Fagaras >> **Bucharest**	450	0	450
Arad >> Sibiu >> Rimnicu Vilcea >> Pitesti >> **Bucharest**	418	0	**418**
Arad >> Sibiu >> Rimnicu Vilcea >> Craiova	366	160	526
Arad >> Timisoara	118	329	447

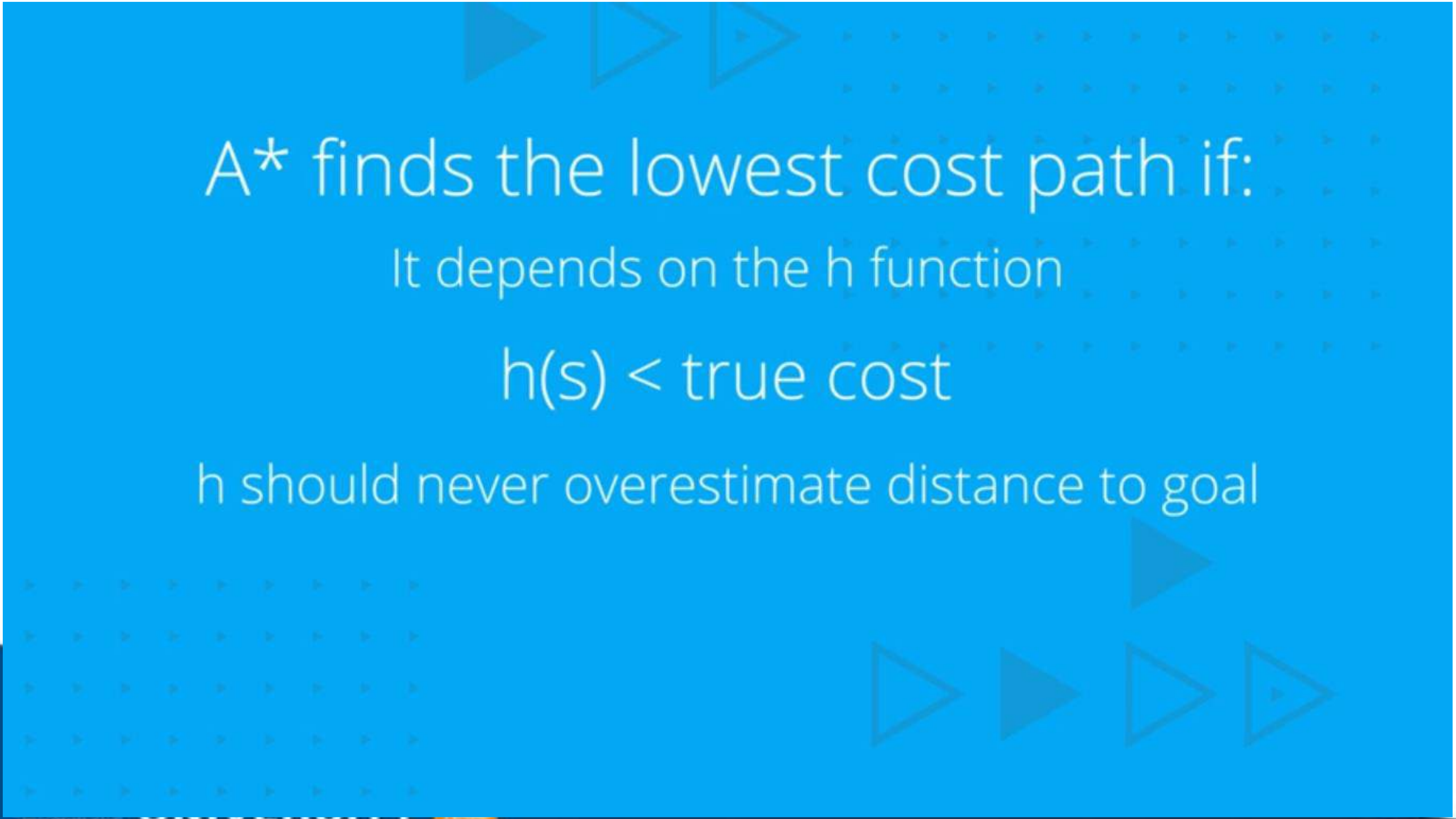


Will A* always find the lowest cost path?

☐ Yes, always

☐ No, depends on the problem

☐ No, depends on h



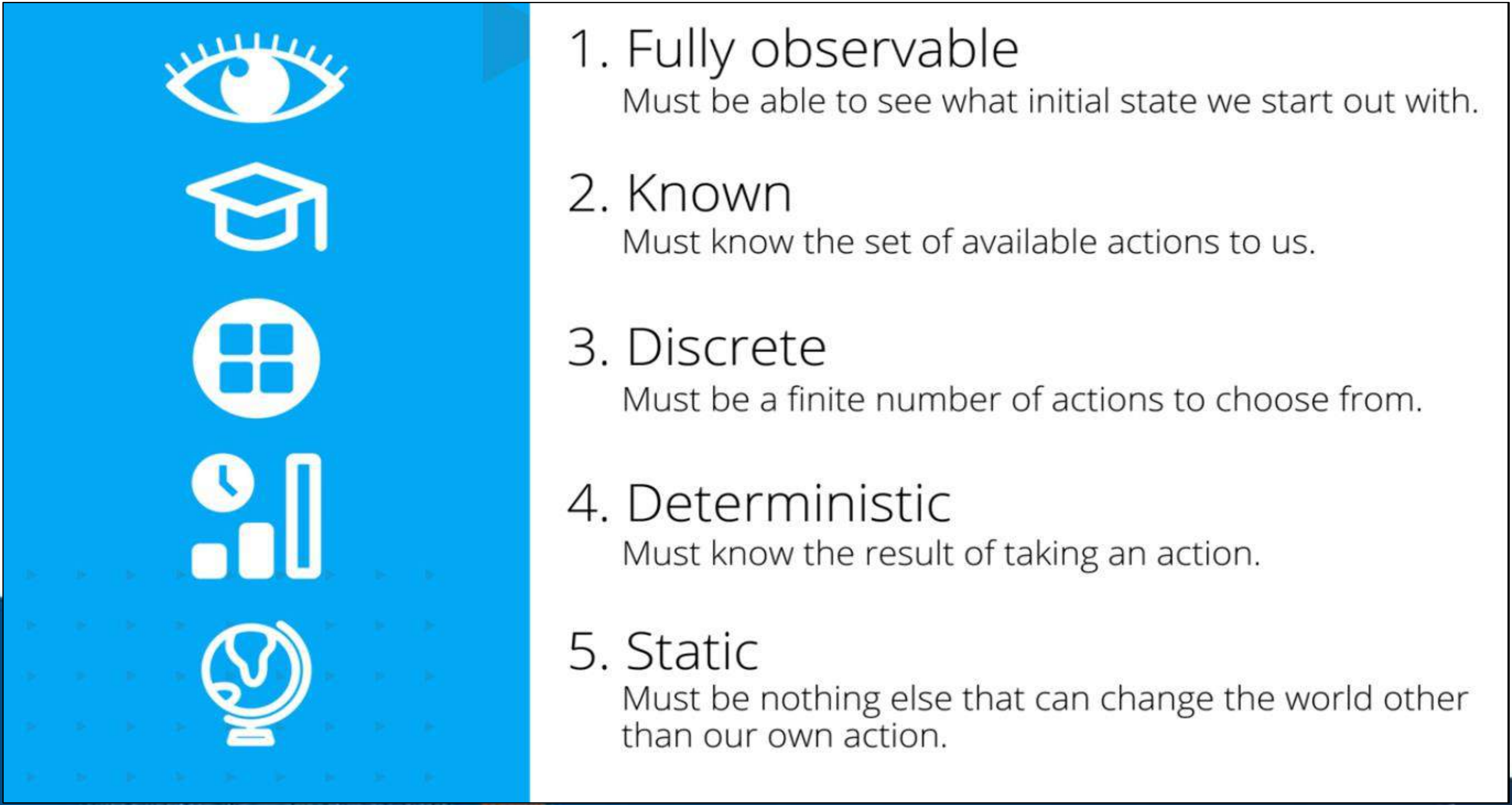
A* finds the lowest cost path if:

It depends on the h function

$h(s) <$ true cost

h should never overestimate distance to goal

CONDITIONS FOR PROBLEM SOLVING BY SEARCHING



1. Fully observable
   Must be able to see what initial state we start out with.

2. Known
   Must know the set of available actions to us.

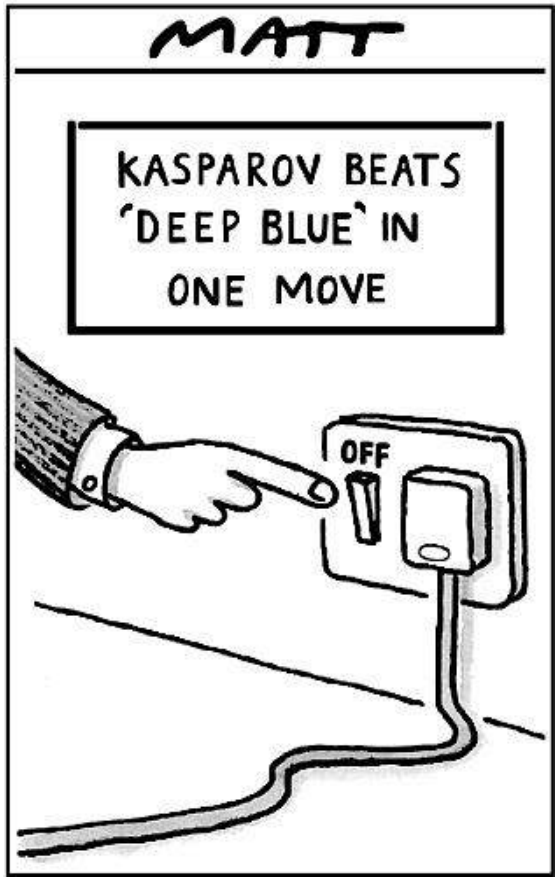
3. Discrete
   Must be a finite number of actions to choose from.

4. Deterministic
   Must know the result of taking an action.

5. Static
   Must be nothing else that can change the world other than our own action.

Adversarial search

World Champion chess player Garry Kasparov is defeated by IBM's Deep Blue chess-playing computer in a six-game match in May, 1997



© Telegraph Group Unlimited 1997

Games vs. search problems

- "Unpredictable" opponent → specifying a move for every possible opponent reply
- Time limits → unlikely to find goal, must approximate
- Turn based games
- Competitive : Commonly zero sum(One player wins and other loses)
- Perfect Information: Player knows results of all previous moves
- There is one best move for each player

Why study games?

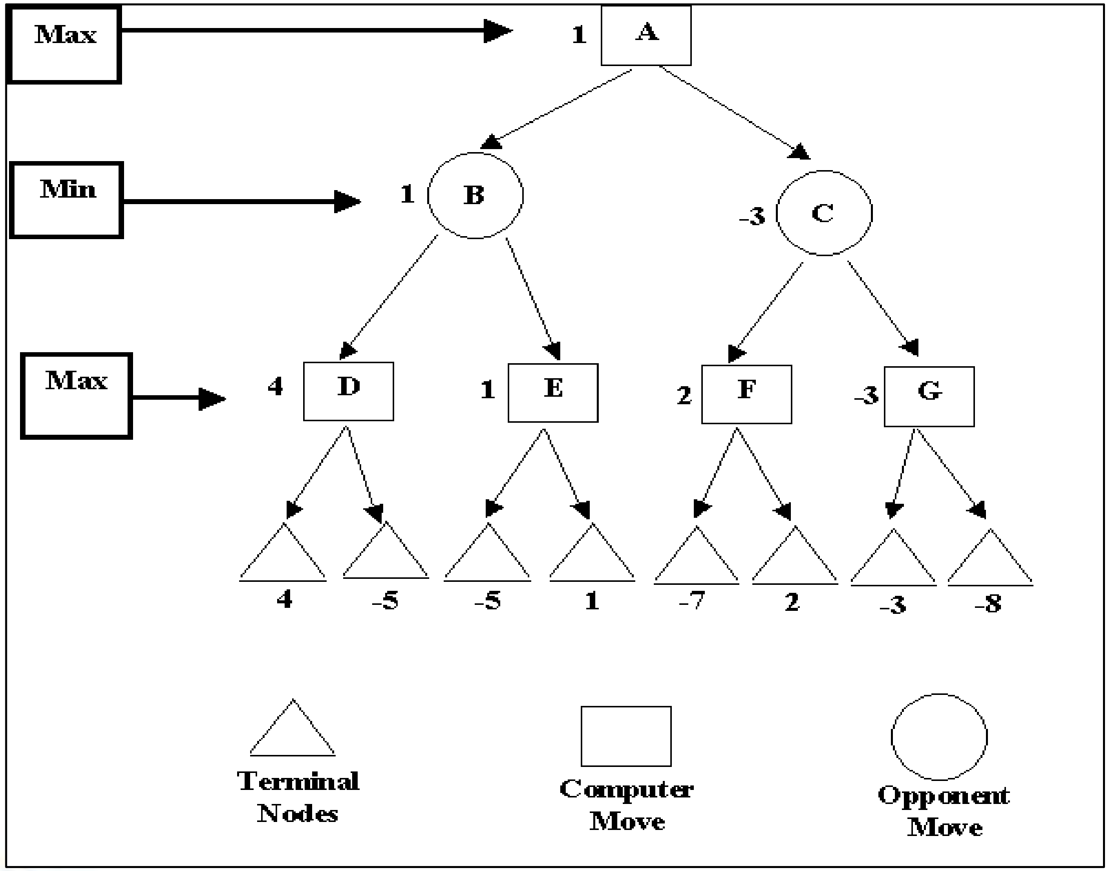
- Games are a traditional hallmark of intelligence
- Games are easy to formalize
- Games can be a good model of real-world competitive or cooperative activities
  - Military confrontations, negotiation, etc.

Types of game environments

	Deterministic	**Stochastic**
Perfect information (fully observable)	Chess, checkers, go	Backgammon, monopoly
Imperfect information (partially observable)	Battleships	Scrabble, poker, bridge

Game Tree

- Nodes are states
- Edges are decisions
- Levels are called “plys”

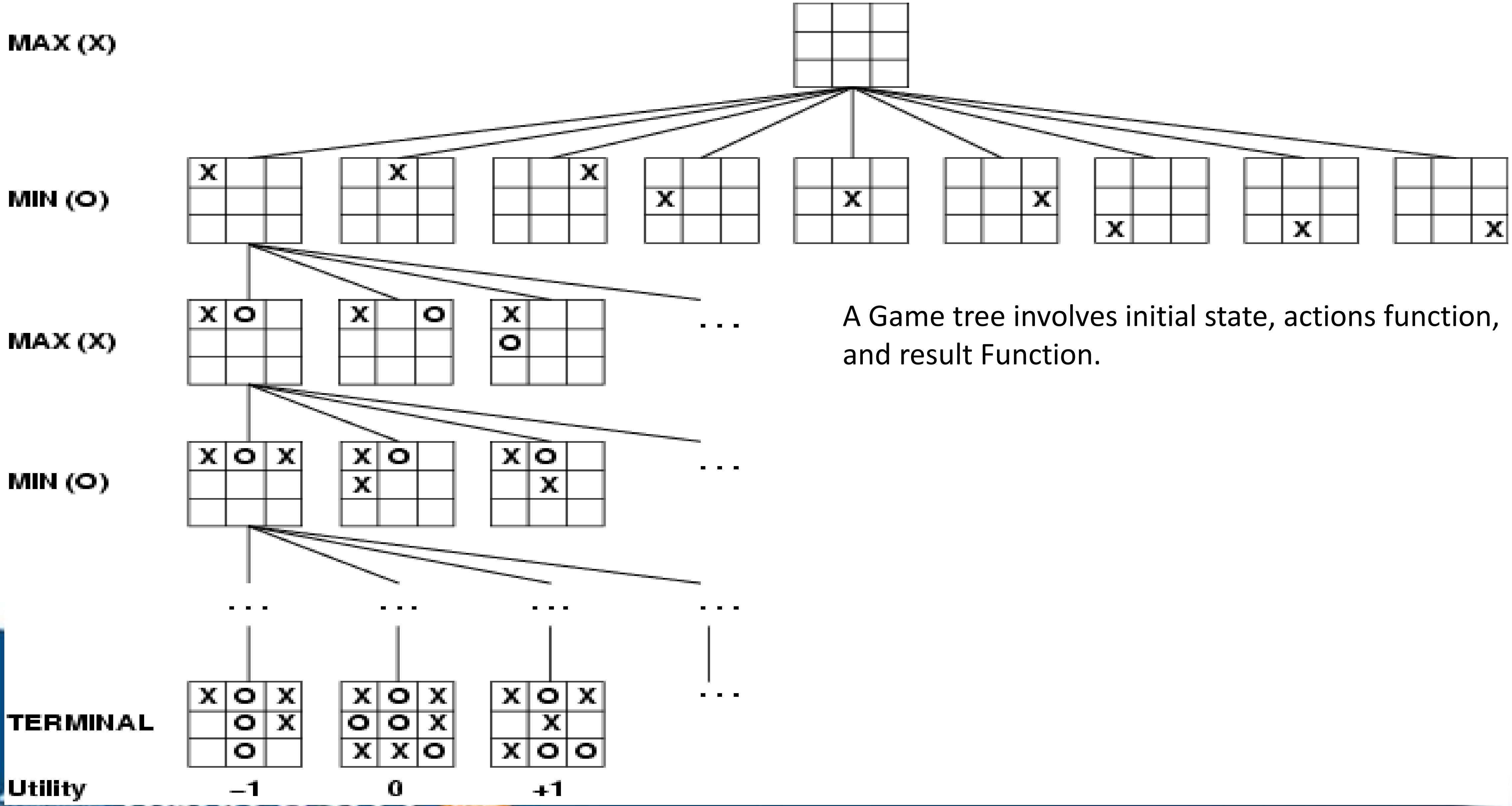


A game can be defined as a type of search in AI which can be formalized of the following elements:

- **Initial state:** It specifies how the game is set up at the start.
- **Player(s):** It specifies which player has moved in the state space.
- **Action(s):** It returns the set of legal moves in state space.
- **Result(s, a):** It is the transition model, which specifies the result of moves in the state space.
- **Terminal-Test(s):** Terminal test is true if the game is over, else it is false at any case. The state where the game ends is called terminal states.
- **Utility(s, p):** A utility function gives the final numeric value for a game that ends in terminal states s for player p.

It is also called payoff function. For Chess, the outcomes are a win, loss, or draw and its payoff values are +1, 0, ½. And for tic-tac-toe, utility values are +1, -1, and 0.

Game tree (2-player, deterministic, turns)



A Game tree involves initial state, actions function, and result Function.

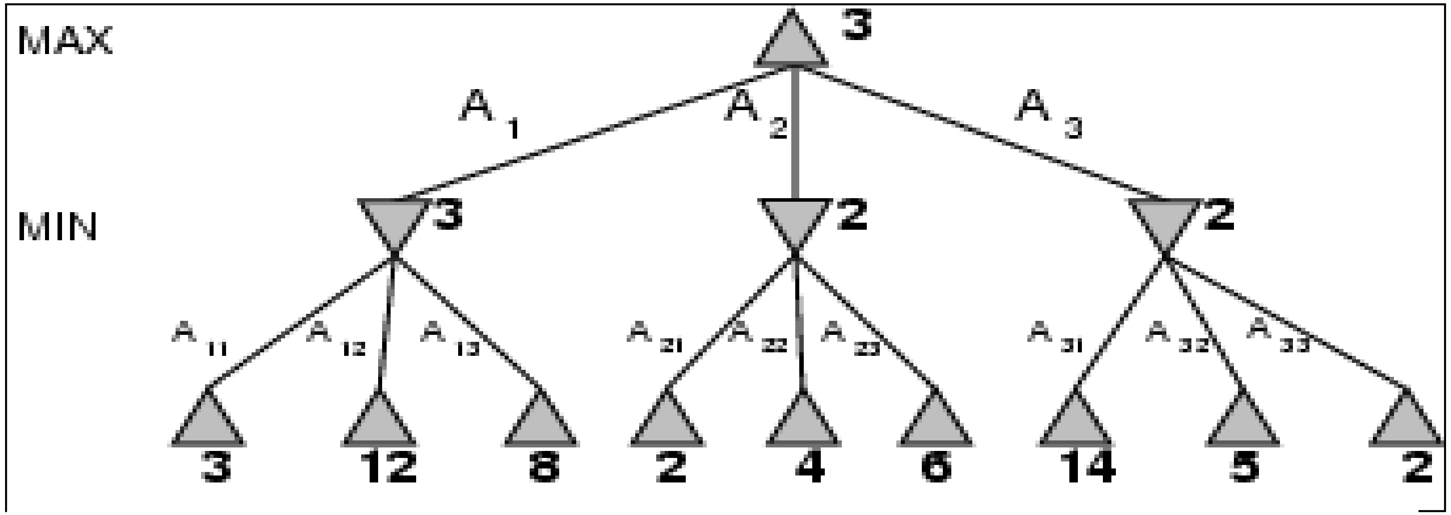
MINIMAX

Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.

- Actions(s) and Result(s,a) defines a game tree
- Two players Max and Min
- Choose move which results in best state
  - Select highest expected score for you
- Assume opponent is playing optimally too
  - Will choose lowest expected score for you
- Minimax uses depth-first search
- ***d*** is the maximum depth ***b*** is the branching-factor
- Time complexity- ?
- Space complexity- ?

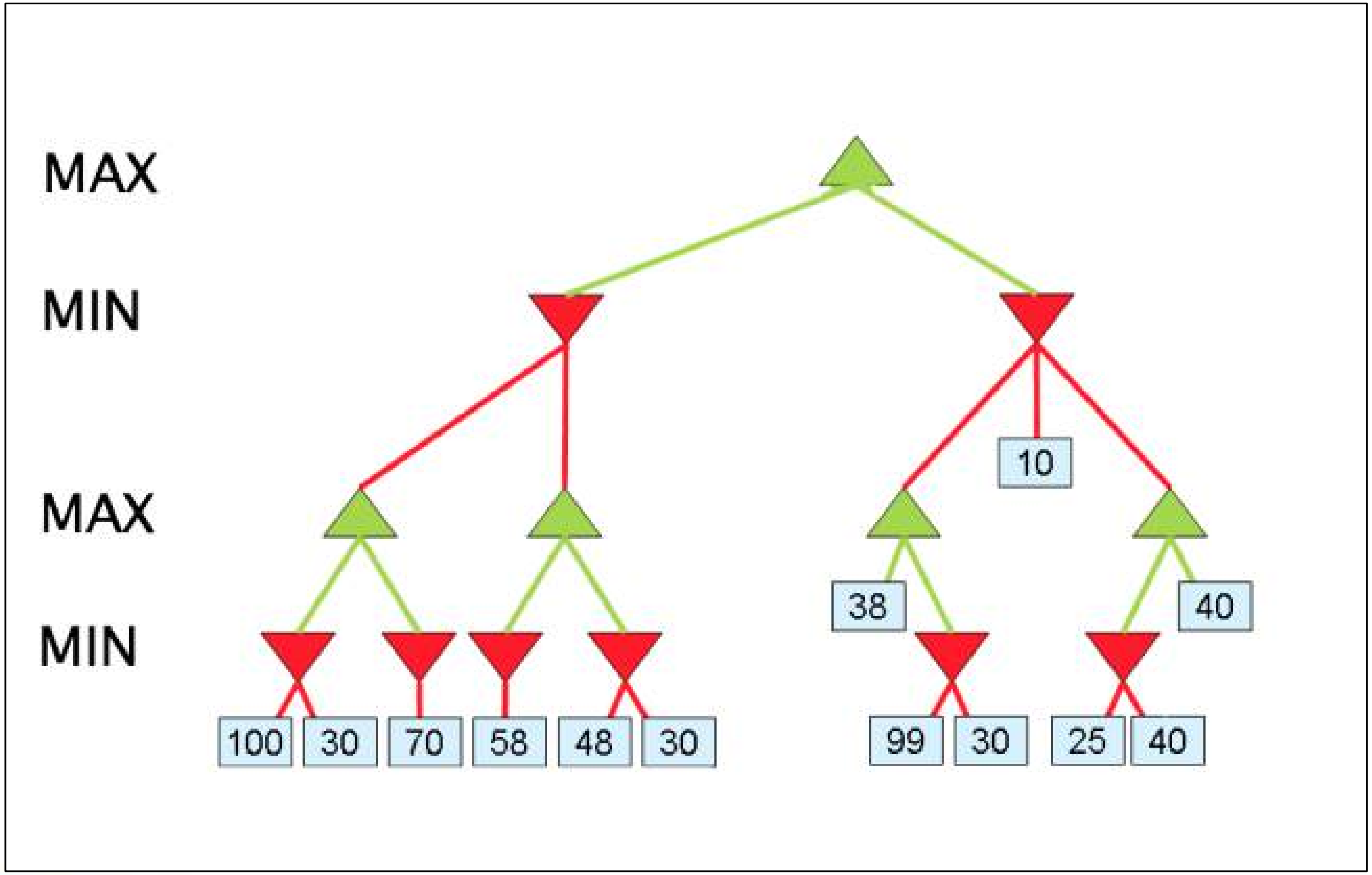
Minimax

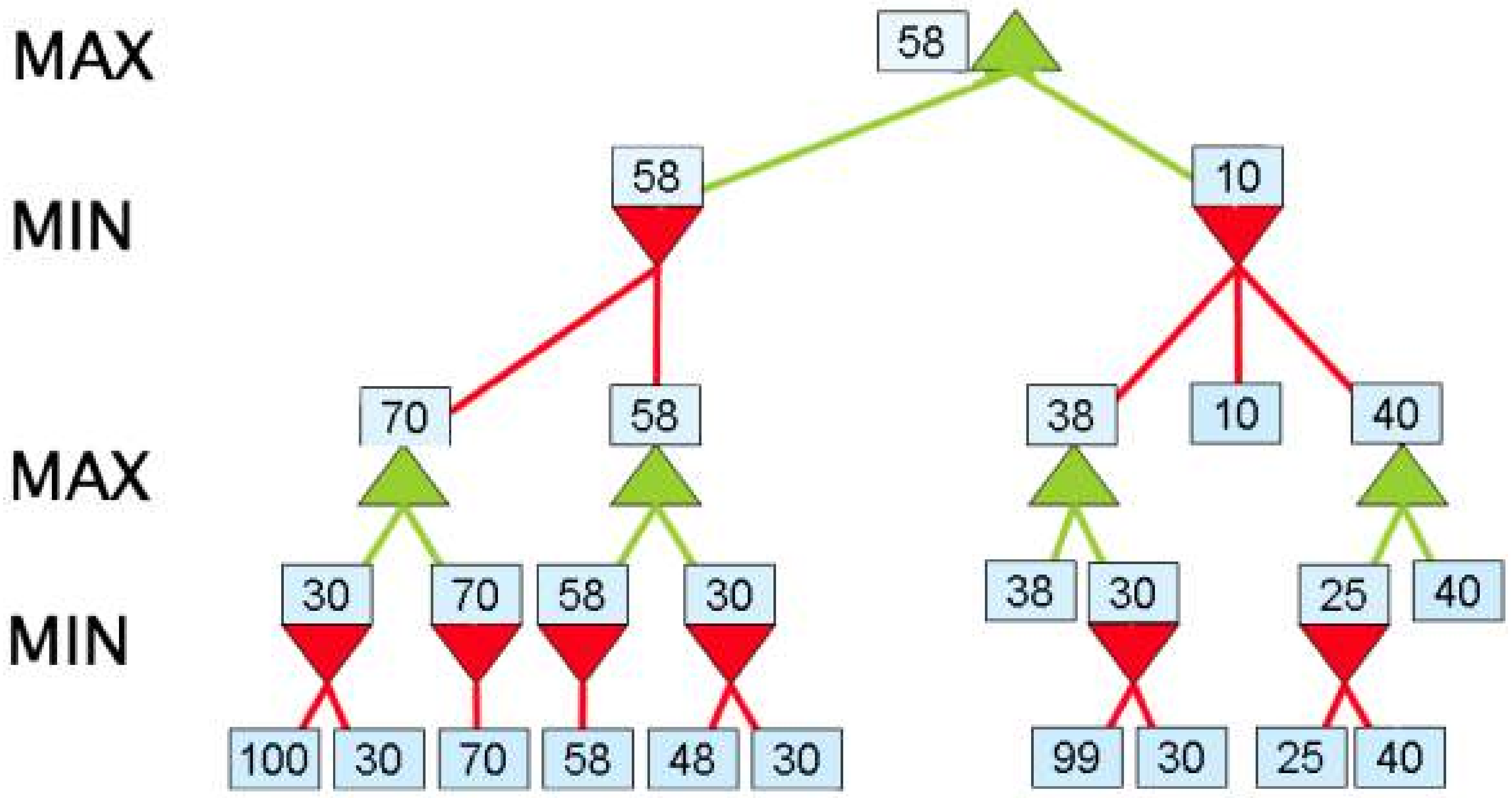
- Choose a move to position with highest minimax value
- Pick the best next move against your best move
- E.g., 2-ply game:



$$minimax(s) = \begin{cases} utility(s) & \text{if } terminal(s) \\ \max_{a \in action(s)} minimax(result(s,a)) & \text{if } player(s) = MAX \\ \min_{a \in action(s)} minimax(result(s,a)) & \text{if } player(s) = MIN \end{cases}$$

Solve...




MAX
MIN
MAX
MIN
58
58
10
70
58
38
10
40
30
70
58
30
38
30
25
40
100
30
70
58
48
30
99
30
25
40

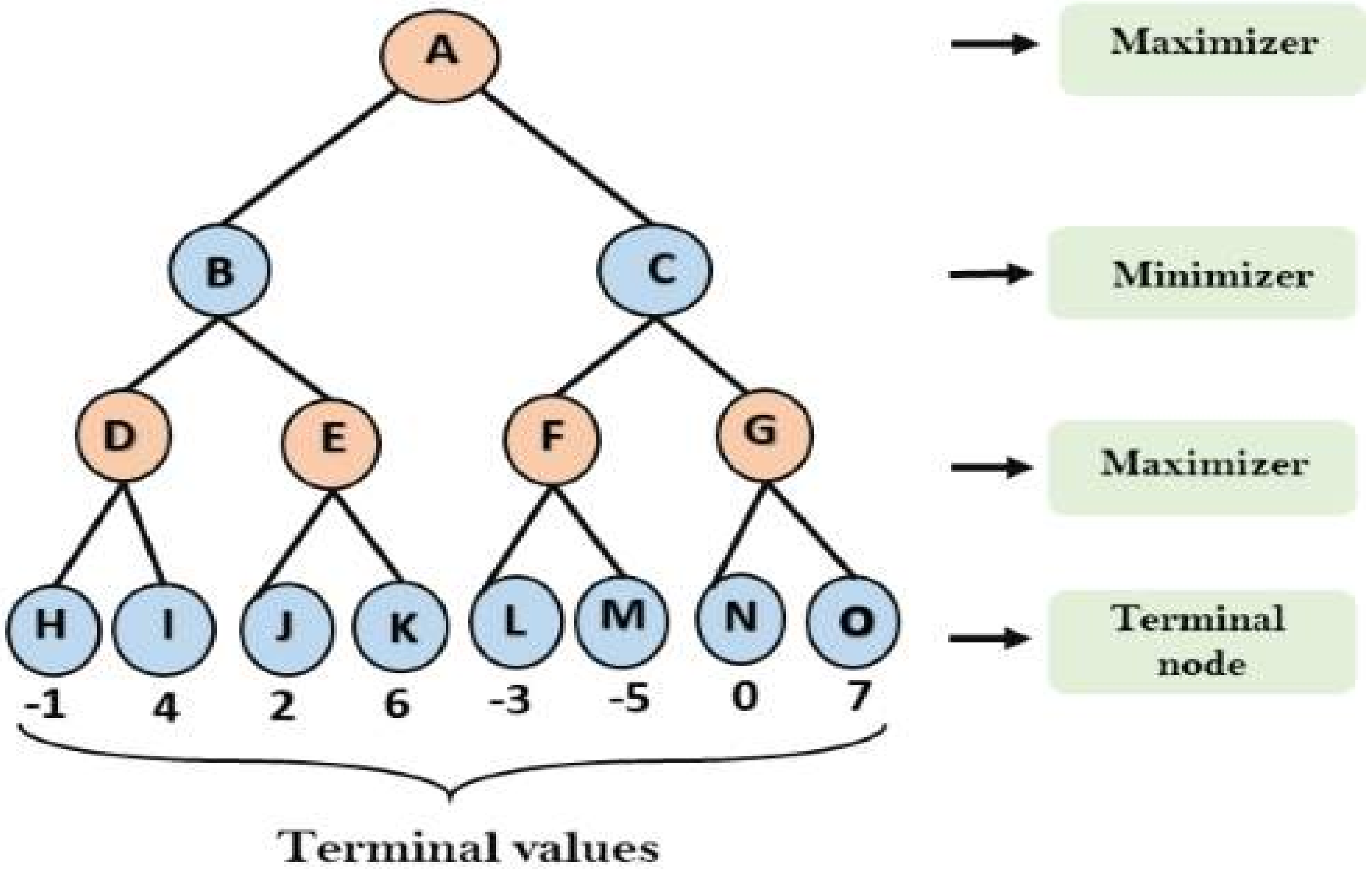
Limitation of the minimax Algorithm:

The main drawback of the minimax algorithm is that it gets really slow for complex games such as Chess, go, etc.

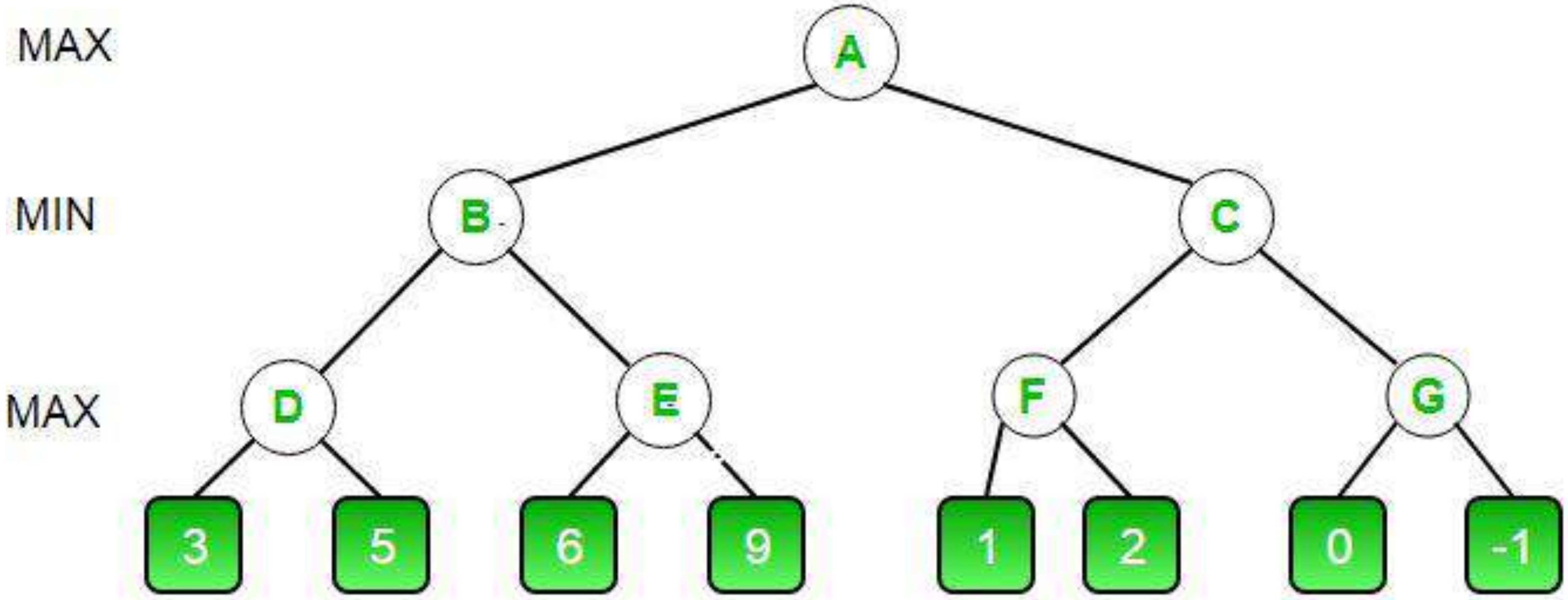
This type of games has a huge branching factor, and the player has lots of choices to decide.

This limitation of the minimax algorithm can be improved from **alpha-beta pruning**

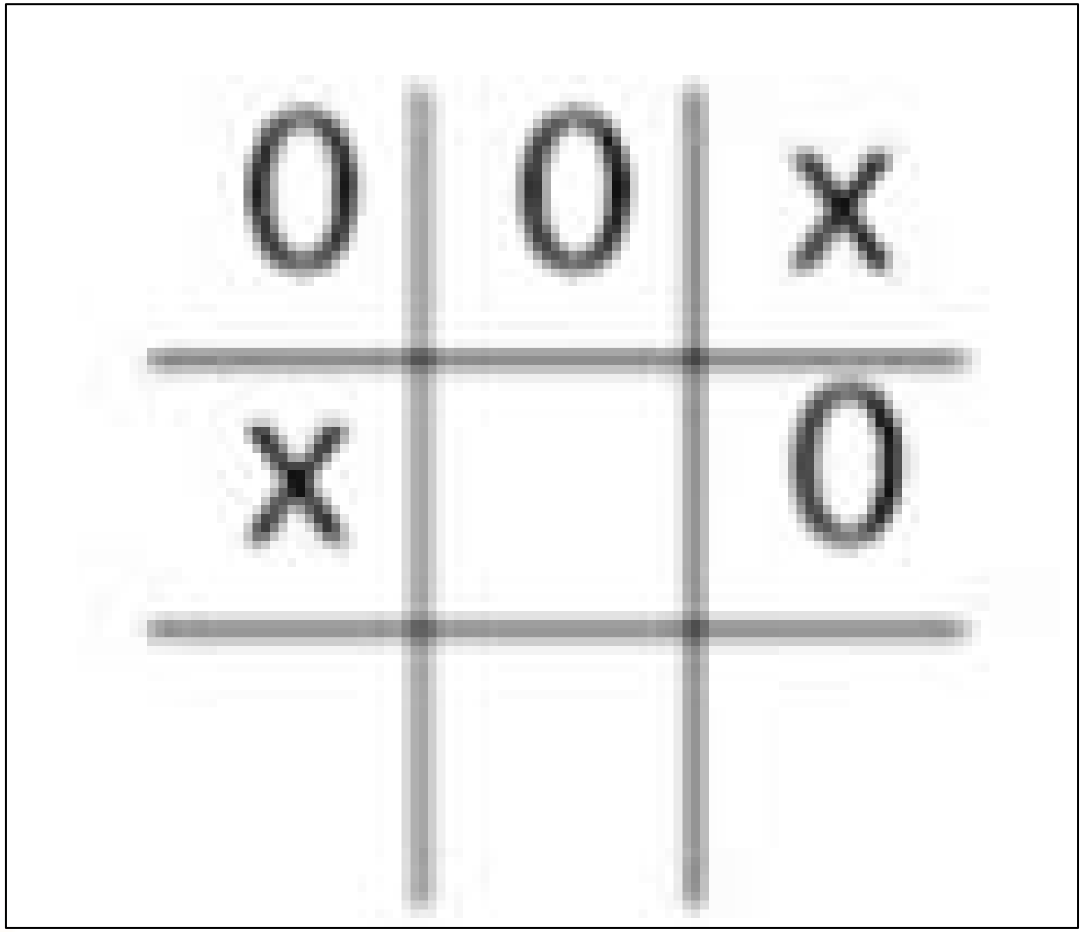
Solve…


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
-1
4
2
6
-3
-5
0
7
Terminal values
Maximizer
Minimizer
Maximizer
Terminal node

Minimax Algorithm

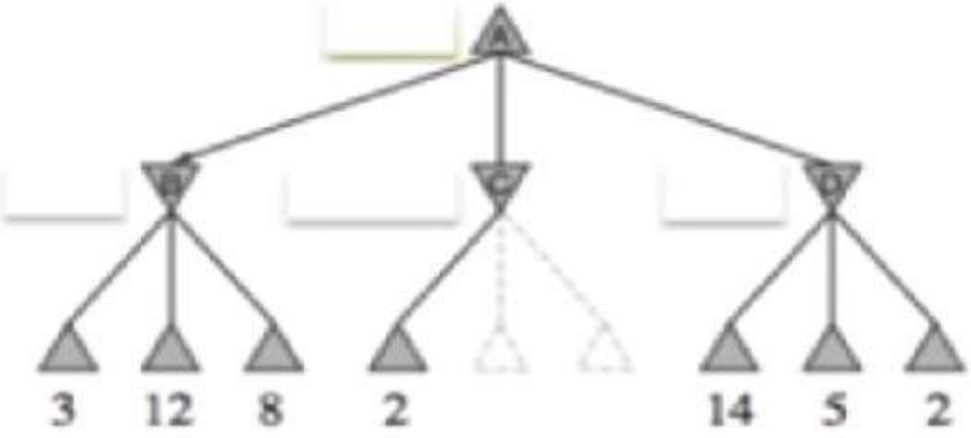


SOLVE .....



O	O	x
x		O

Alpha Beta Pruning

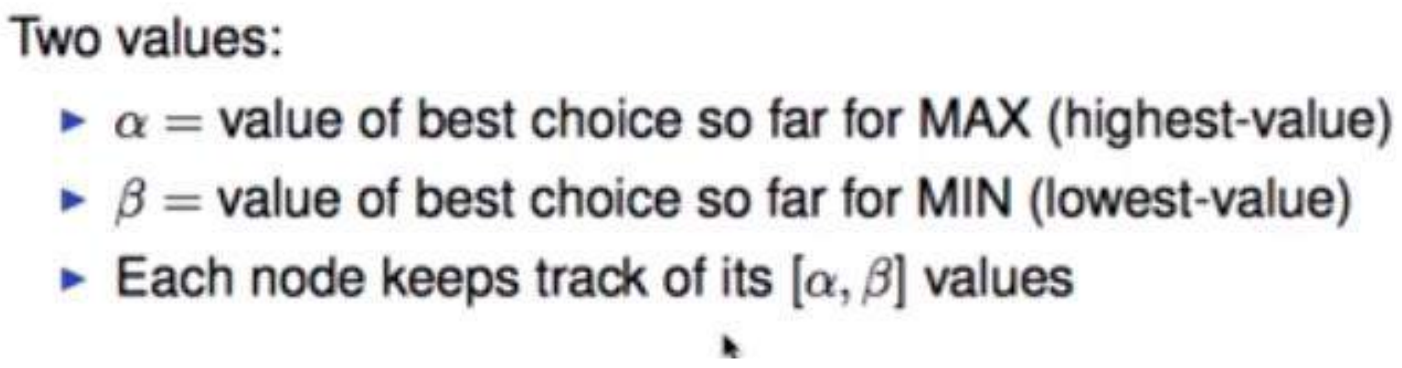


Do we need to expand all nodes?

$$
\begin{aligned}
minimax(root) &= max(min(3,12,8), min(2,x,y), min(14,5,2)) \\
&= max(3, min(2,x,y), 2) \\
&= max(3,z,2) \\
&= 3
\end{aligned}
$$

Do we need z?

Alpha Beta Pruning



Two values:

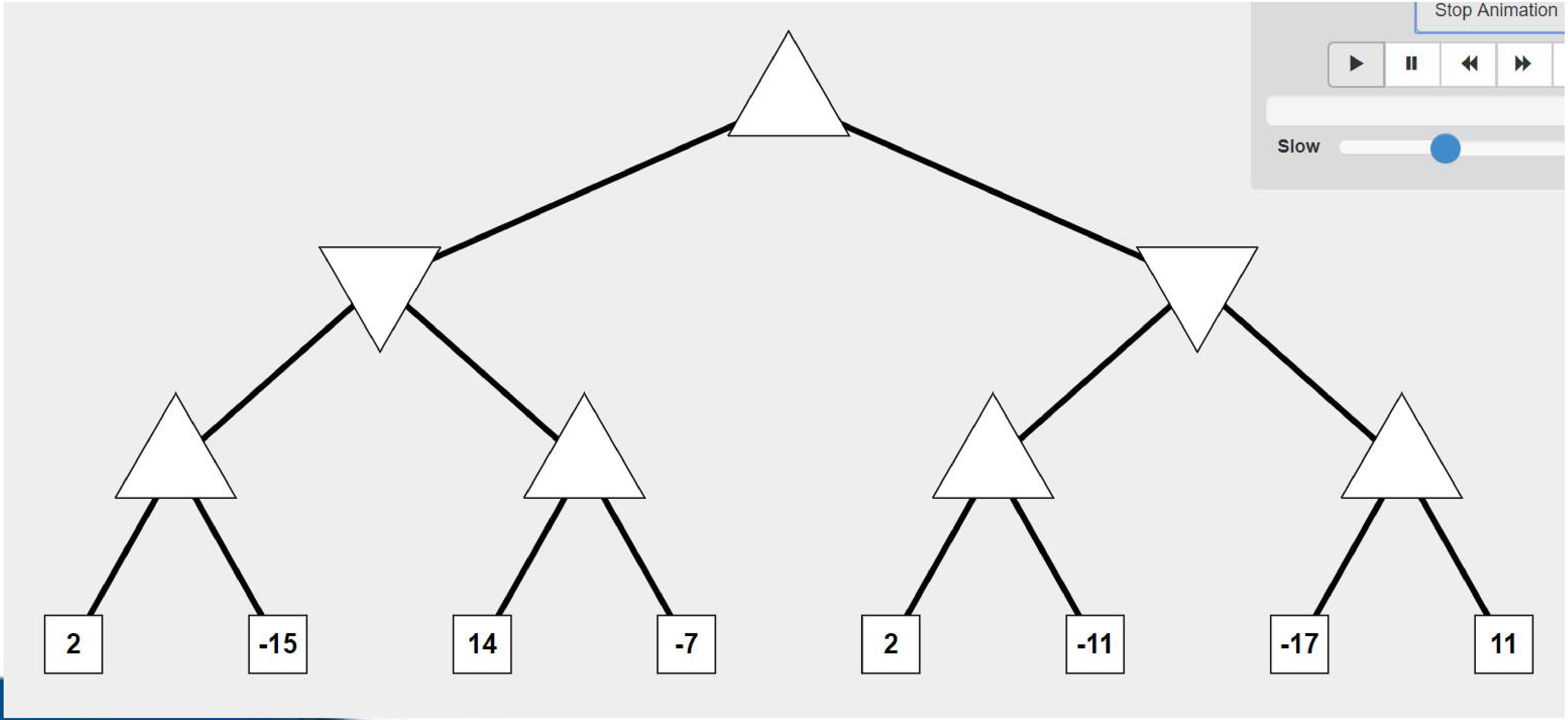
- $\alpha =$ value of best choice so far for MAX (highest-value)
- $\beta =$ value of best choice so far for MIN (lowest-value)
- Each node keeps track of its $[\alpha, \beta]$ values

Alpha Beta Pruning

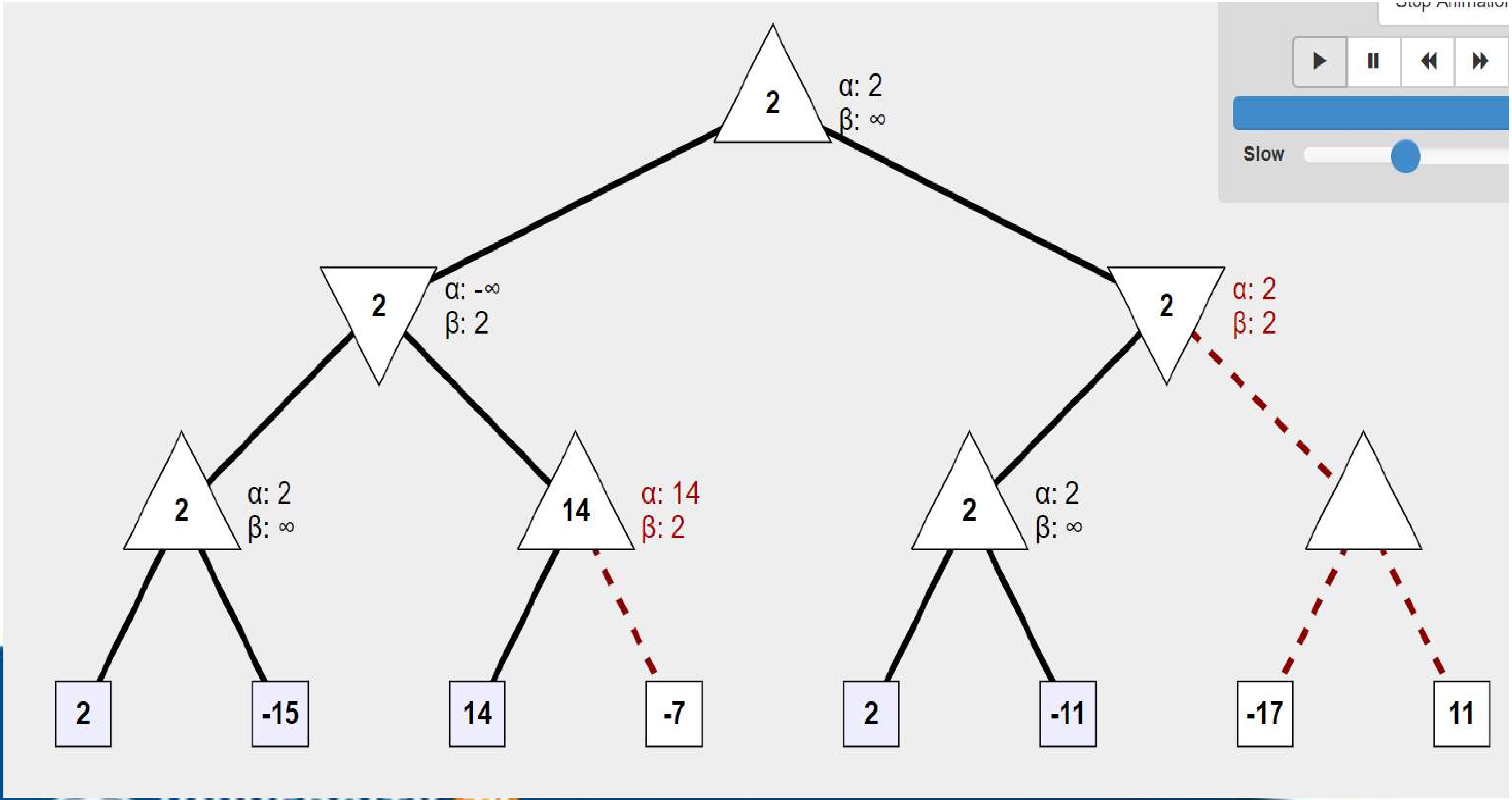
- Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.
- The two-parameter can be defined as:
  - **Alpha:** The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is **-∞**.
  - **Beta:** The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is **+∞**.
- The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.

Key points about alpha-beta pruning:

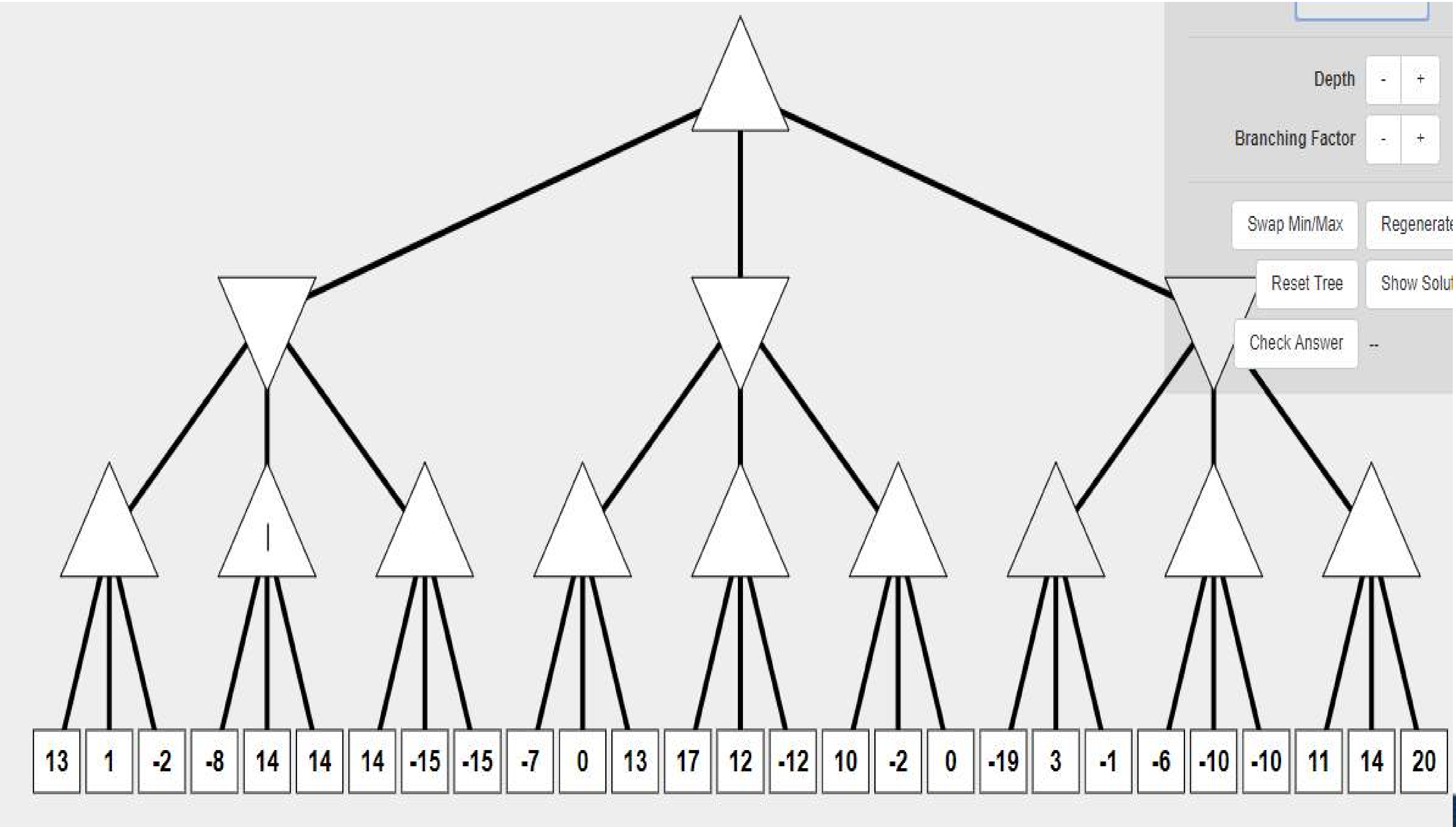
- The Max player will only update the value of alpha.
- The Min player will only update the value of beta.
- While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.
- We will only pass the alpha, beta values to the child nodes.

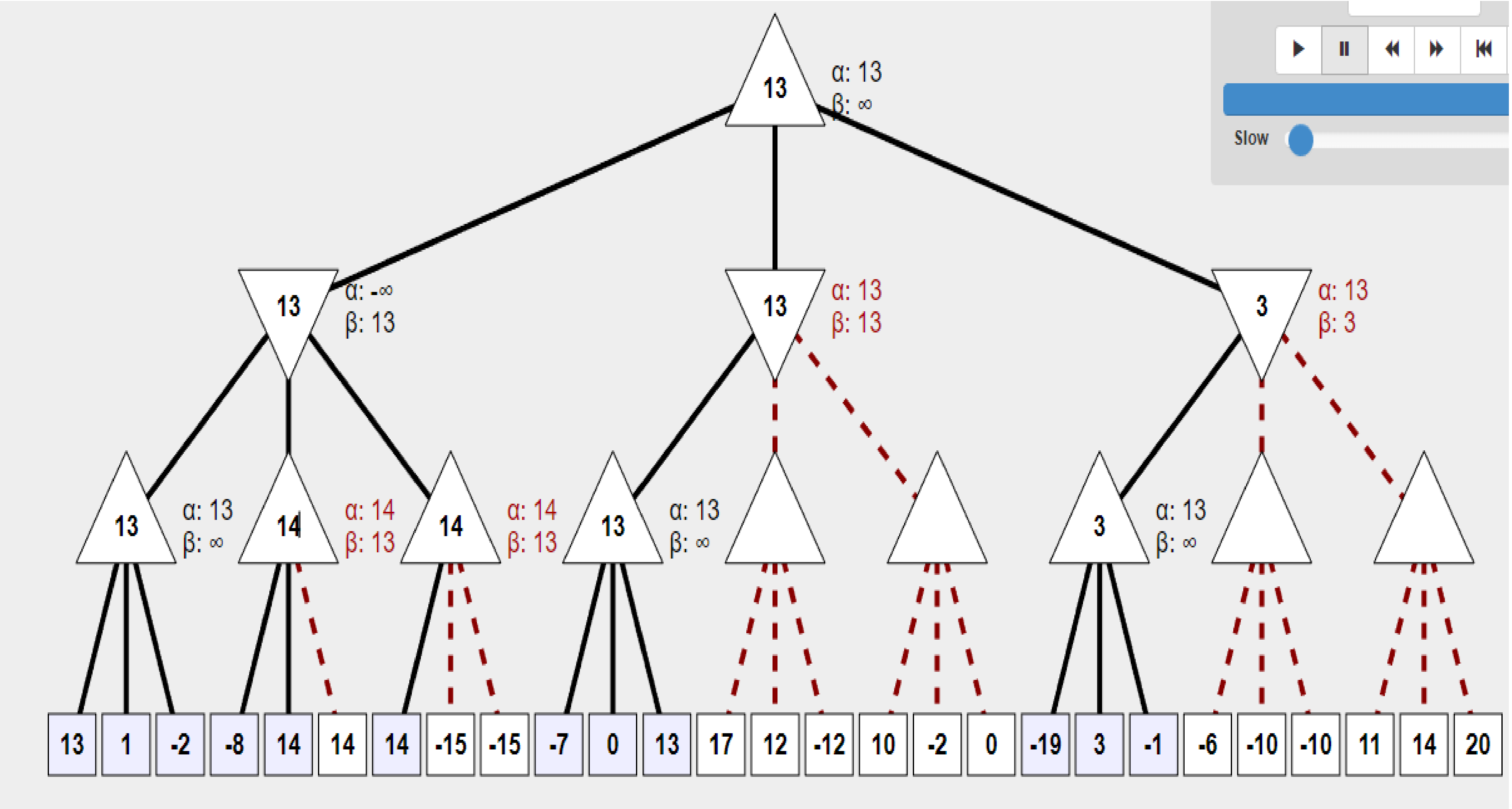


http://inst.eecs.berkeley.edu/~cs61b/fa14/ta-materials/apps/ab_tree_practice/

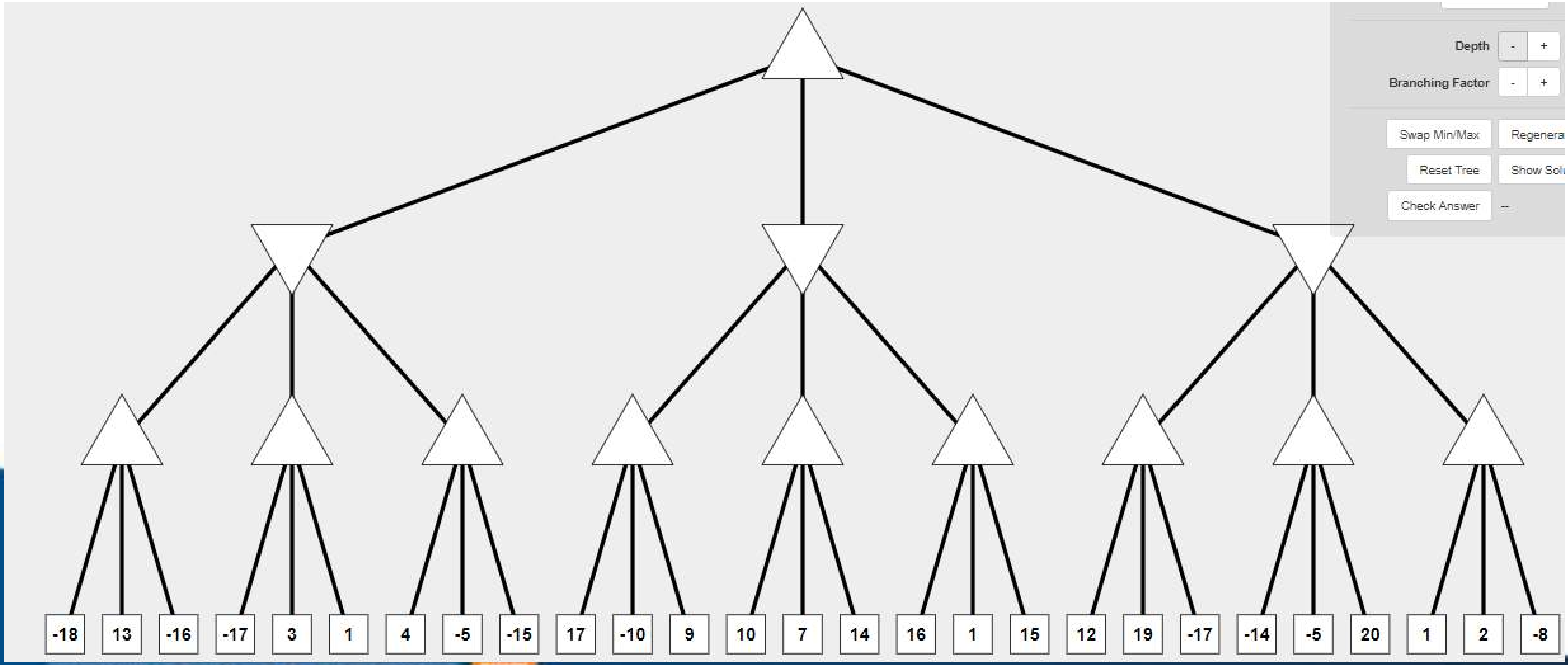

Stop Animation
Slow
2
α: 2
β: ∞
2
α: -∞
β: 2
2
α: 2
β: 2
2
α: 2
β: ∞
14
α: 14
β: 2
2
α: 2
β: ∞
2
-15
14
-7
2
-11
-17
11

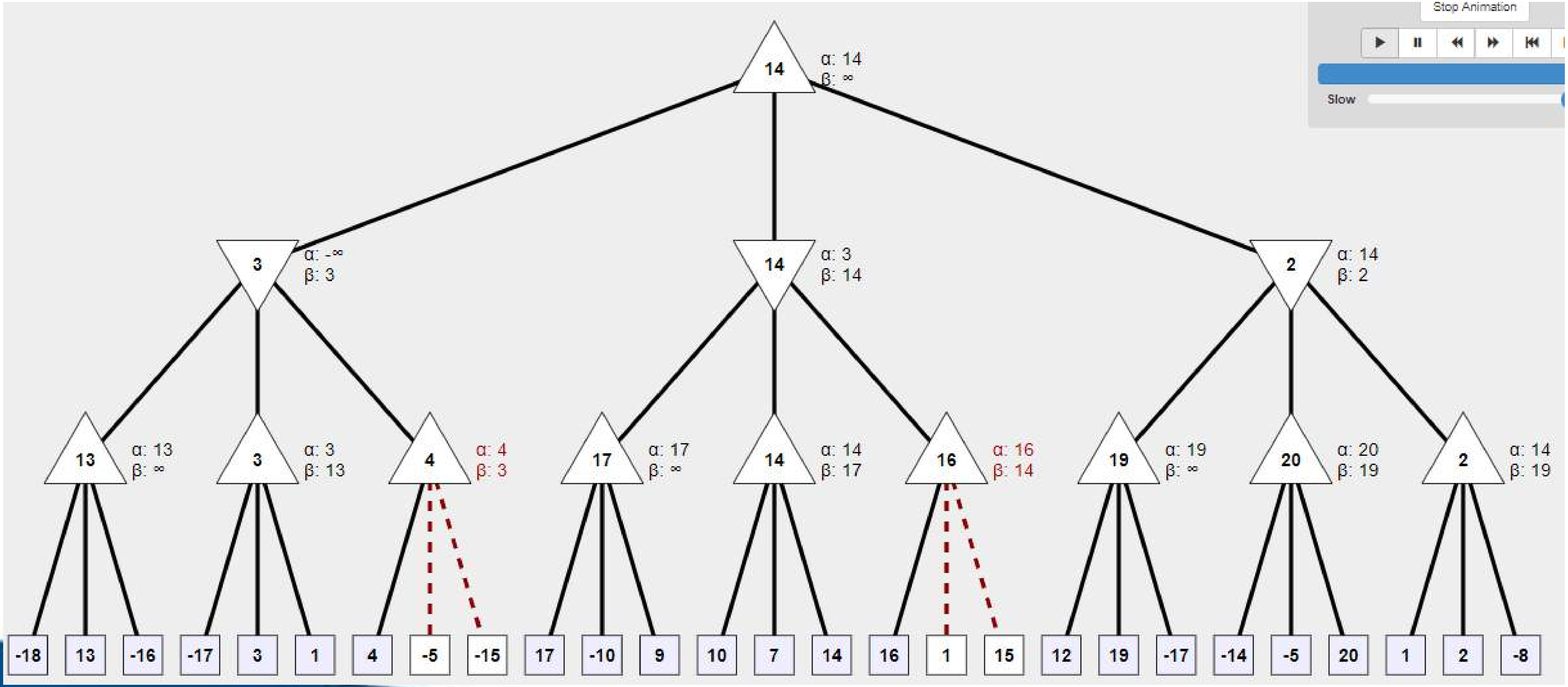
SOLVE…


Depth
-
+
Branching Factor
-
+
Swap Min/Max
Regenerate
Reset Tree
Show Solut
Check Answer
--
13
1
-2
-8
14
14
14
-15
-15
-7
0
13
17
12
-12
10
-2
0
-19
3
-1
-6
-10
-10
11
14
20


13
α: 13
β: ∞
13
α: -∞
β: 13
13
α: 13
β: 13
3
α: 13
β: 3
13
α: 13
β: ∞
14
α: 14
β: 13
14
α: 14
β: 13
13
α: 13
β: ∞
3
α: 13
β: ∞
13
1
-2
-8
14
14
14
-15
-15
-7
0
13
17
12
-12
10
-2
0
-19
3
-1
-6
-10
-10
11
14
20
Slow

Solve using Alpha Beta Pruning




Stop Animation
Slow
14
α: 14
β: ∞
3
α: -∞
β: 3
14
α: 3
β: 14
2
α: 14
β: 2
13
α: 13
β: ∞
3
α: 3
β: 13
4
α: 4
β: 3
17
α: 17
β: ∞
14
α: 14
β: 17
16
α: 16
β: 14
19
α: 19
β: ∞
20
α: 20
β: 19
2
α: 14
β: 19
-18
13
-16
-17
3
1
4
-5
-15
17
-10
9
10
7
14
16
1
15
12
19
-17
-14
-5
20
1
2
-8

Constraint Satisfaction Problem

- *Constraint satisfaction is a technique where a problem is solved when its values satisfy certain constraints or rules of the problem.*
- Such type of technique leads to a deeper understanding of the problem structure as well as its complexity.

- Constraint satisfaction depends on three components, namely:

- **X:** It is a set of variables.
- **D:** It is a set of domains where the variables reside. There is a specific domain for each variable.
- **C:** It is a set of constraints which are followed by the set of variables.

Constraint Satisfaction Problem

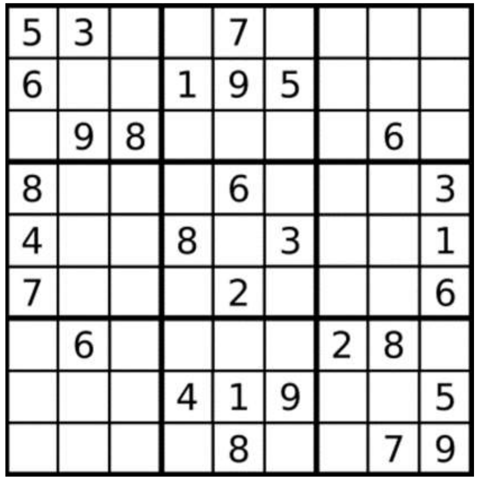
constraint satisfaction is the process of finding a solution to a set of constraints that impose conditions that the variables must satisfy.

A solution is therefore a set of values for the variables that satisfies all constraints—that is, a point in the feasible region.

Constraint Satisfaction Problem

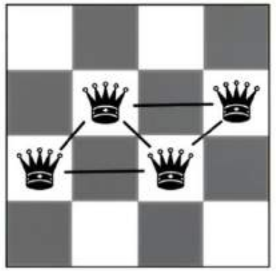
- **Sudoku**
  - **Sudoku CSP Definition**
    - **Variables:** the 81 boxes that must be assigned a value
    - **Domains:** every variable has the same domain, the single digits 1-9
    - **Constraints:** each row, column, and 3x3 group must contain one of each digits 1-9
- **4-Queens**
  - **4-Queens CSP Definition**
    - **Variables:** the row assignment of each of the 4 queens (the variables represent the queen assigned to each of the four columns)
    - **Domains:** every variable has the same domain, the single digits 1-4
    - **Constraints:** No pair of queens can be on the same row or diagonal
- **Map Coloring**
  - **Map Coloring CSP Definition**
    - **Variables:** One for each region of the map (in this case WA, NT, SA, Q, NSW, V, and T)
    - **Domains:** All variables have the same domain, the list of colors that may be assigned to each region
    - **Constraints:** No pair of adjacent regions can have the same color

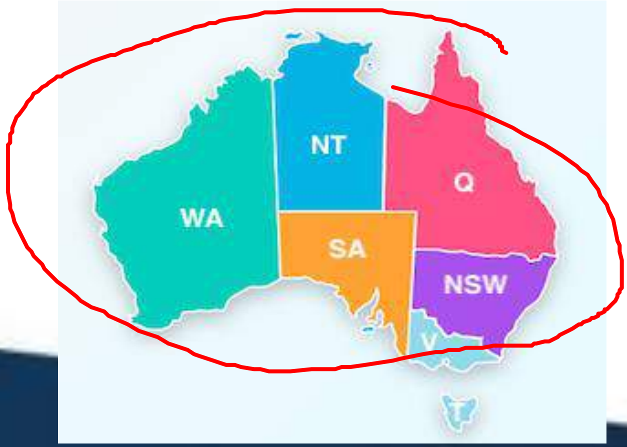
R G B



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

4 Queens





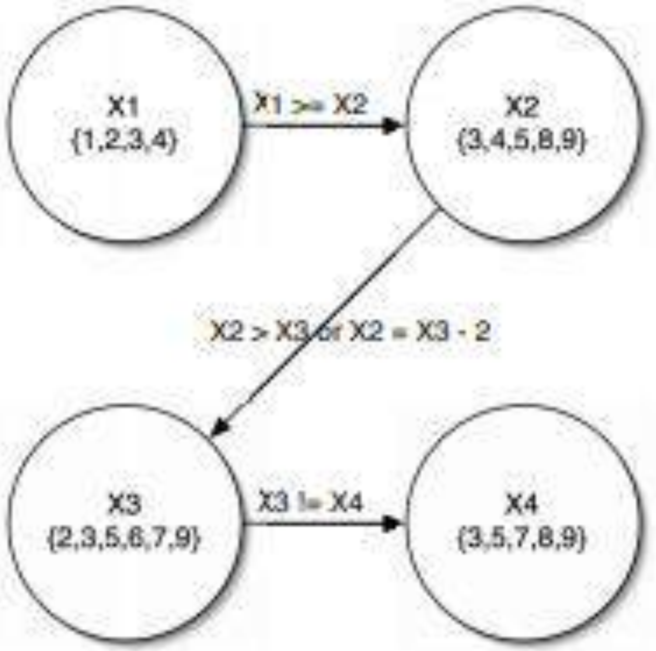
Consider the following binary-constraint network: There are 4 variables: X_1, X_2, X_3, X_4, with the domains:

$$D_1 = \{1, 2, 3, 4\}, D_2 = \{3, 4, 5, 8, 9\}, D_3 = \{2, 3, 5, 6, 7, 9\}, D_4 = \{3, 5, 7, 8, 9\}.$$

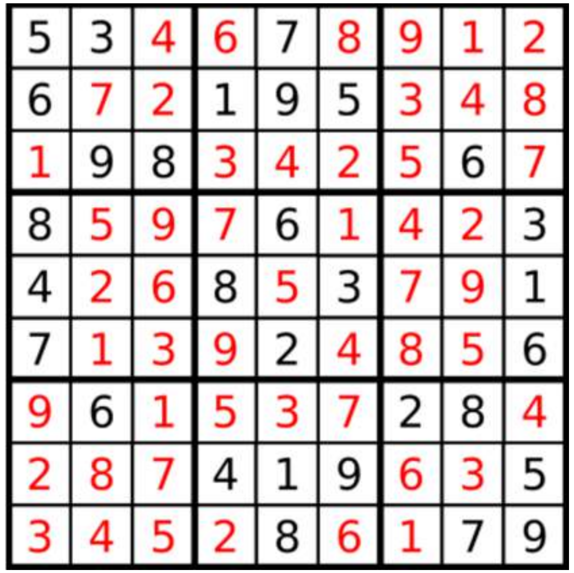
The constraints are: $X_1 \geq X_2$, $X_2 > X_3$ or $X_3 - X_2 = 2$, $X_3 \neq X_4$.

Write the constraints in a relational form and draw the constraint graph.

Relation	Domain
$X_1 \geq X_2$	(3,3), (4,3), (4,4)
$X_2 > X_3$ or $X_3 - X_2 = 2$	(3,2), (4,2), (4,3), (5,2), (5,3), (8,2), (8,3), (8,5), (8,6) (8,7), (9,2), (9,3), (9,5), (9,6), (9,7), (3,5), (4,6), (5,7)
$X_3 \neq X_4$	(2,3), (2,5), (2,7), (2,8), (2,9), (3,5), (3,7), (3,8), (3,9) (5,3), (5,7), (5,8), (5,9), (6,3), (6,5), (6,7), (6,8) (6,9) (7,3), (7,5), (7,8), (7,9), (9,3), (9,5), (9,7), (9,8)

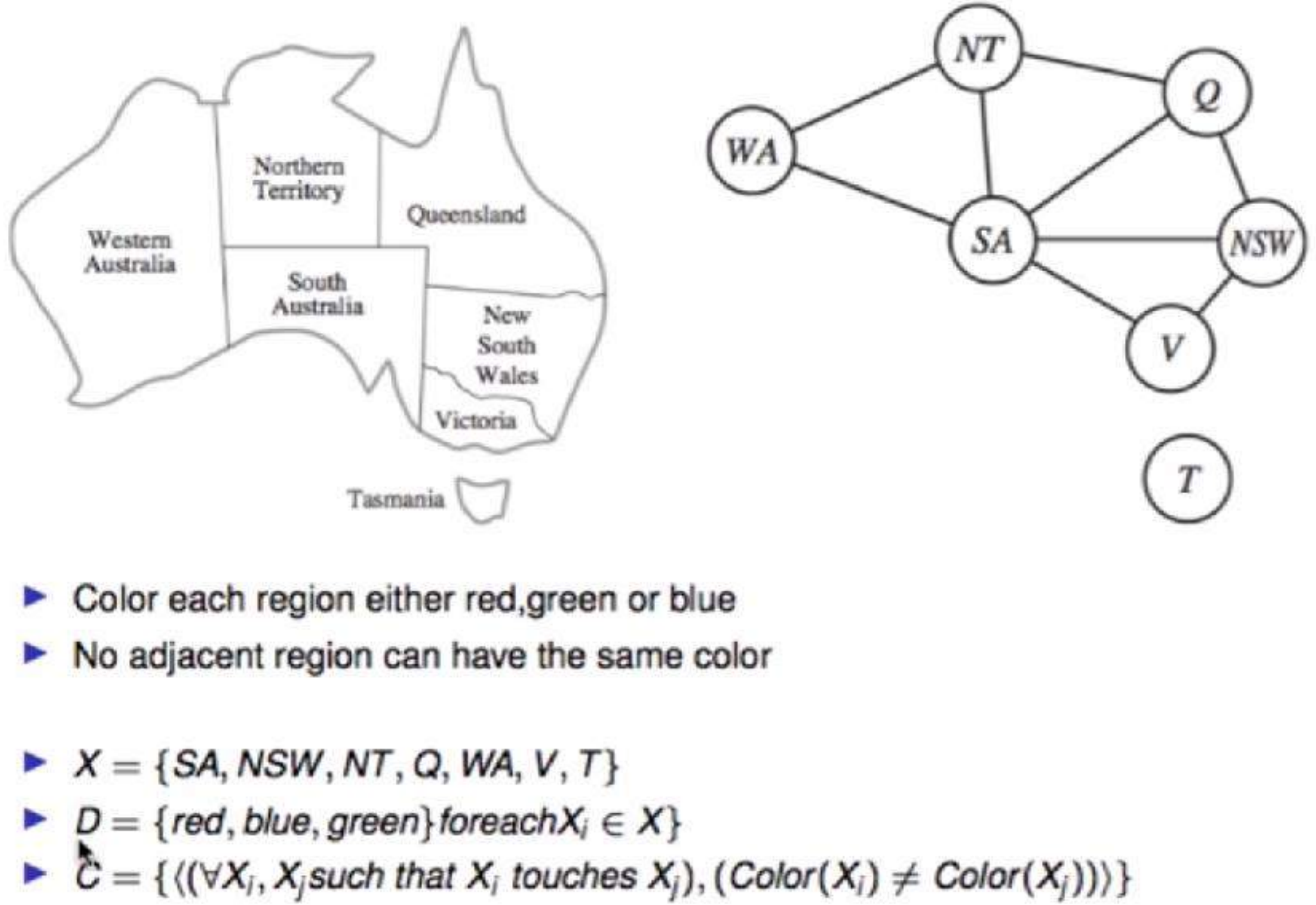


Is the network arc-consistent ? If not, compute the arc-consistent network.



5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

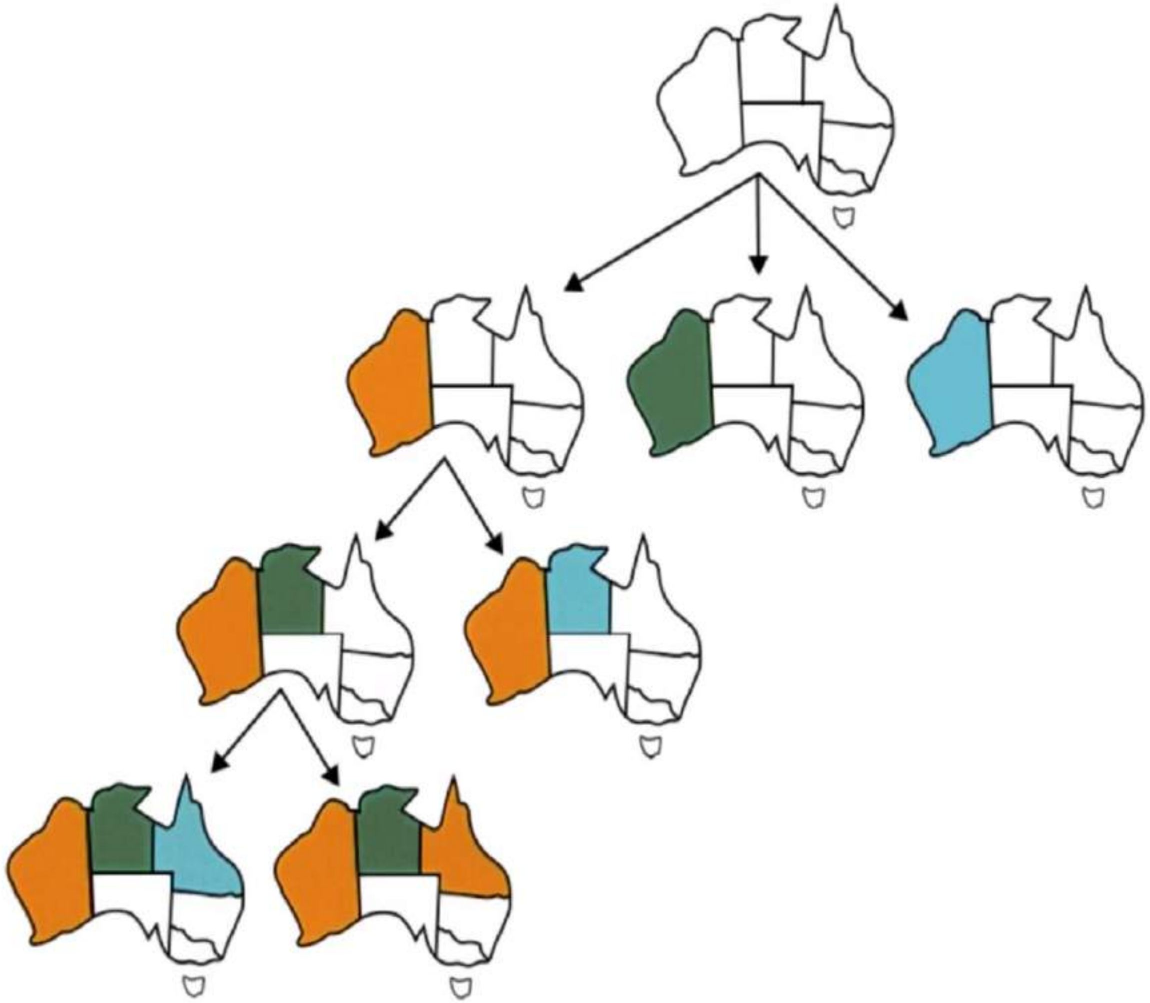
Constraint Graph



- Color each region either red,green or blue
- No adjacent region can have the same color

- $X = \{SA, NSW, NT, Q, WA, V, T\}$
- $D = \{red, blue, green\} foreach X_i \in X\}$
- $C = \{\langle(\forall X_i, X_j such\ that\ X_i\ touches\ X_j), (Color(X_i) \neq Color(X_j))\rangle\}$

Backtracking Search



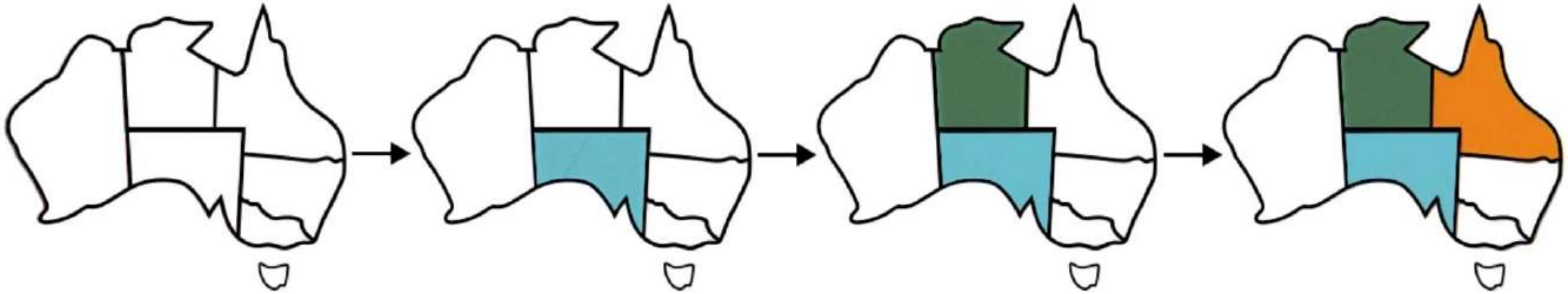
Optimizing Backtracking

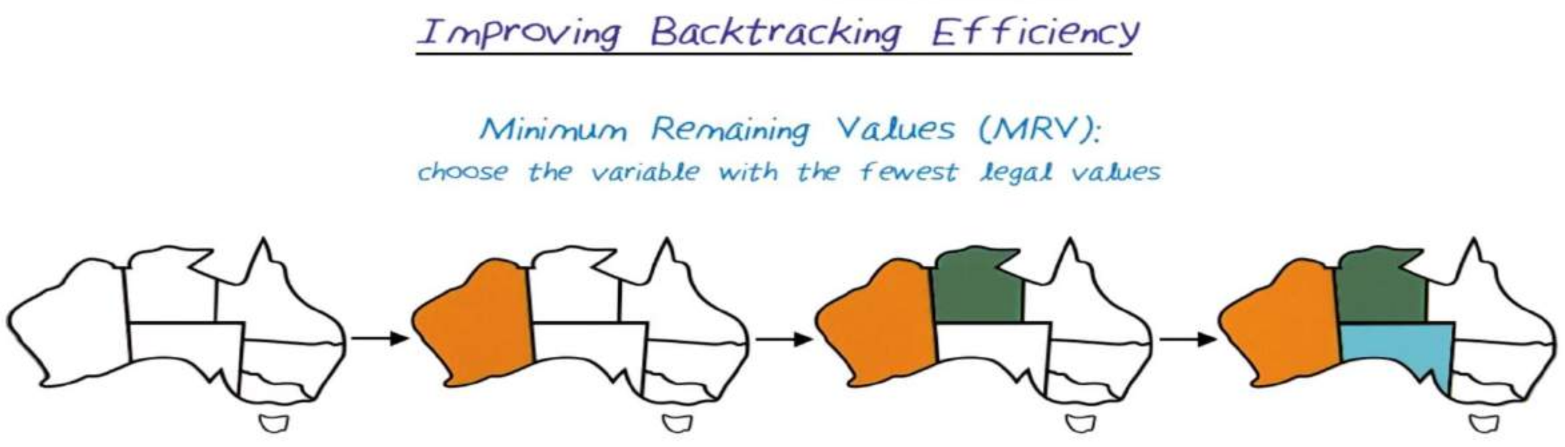
- How to choose next variables:
  - **Degree heuristic**: assign a value to the variable that is involved in the largest number of constraints on other unassigned variables.
  - **Minimum remaining values (MRV):** choose the variable with the fewest possible values.
  - **Least-constraining value heuristic**: choose a value that rules out the smallest number of values in variables connected to the current variable by constraints.
  - **Forward Checking:** Keep track of all permissible values for variables

Improving Backtracking Efficiency

Degree Heuristic:

choose the variable with the most constraints on remaining variables



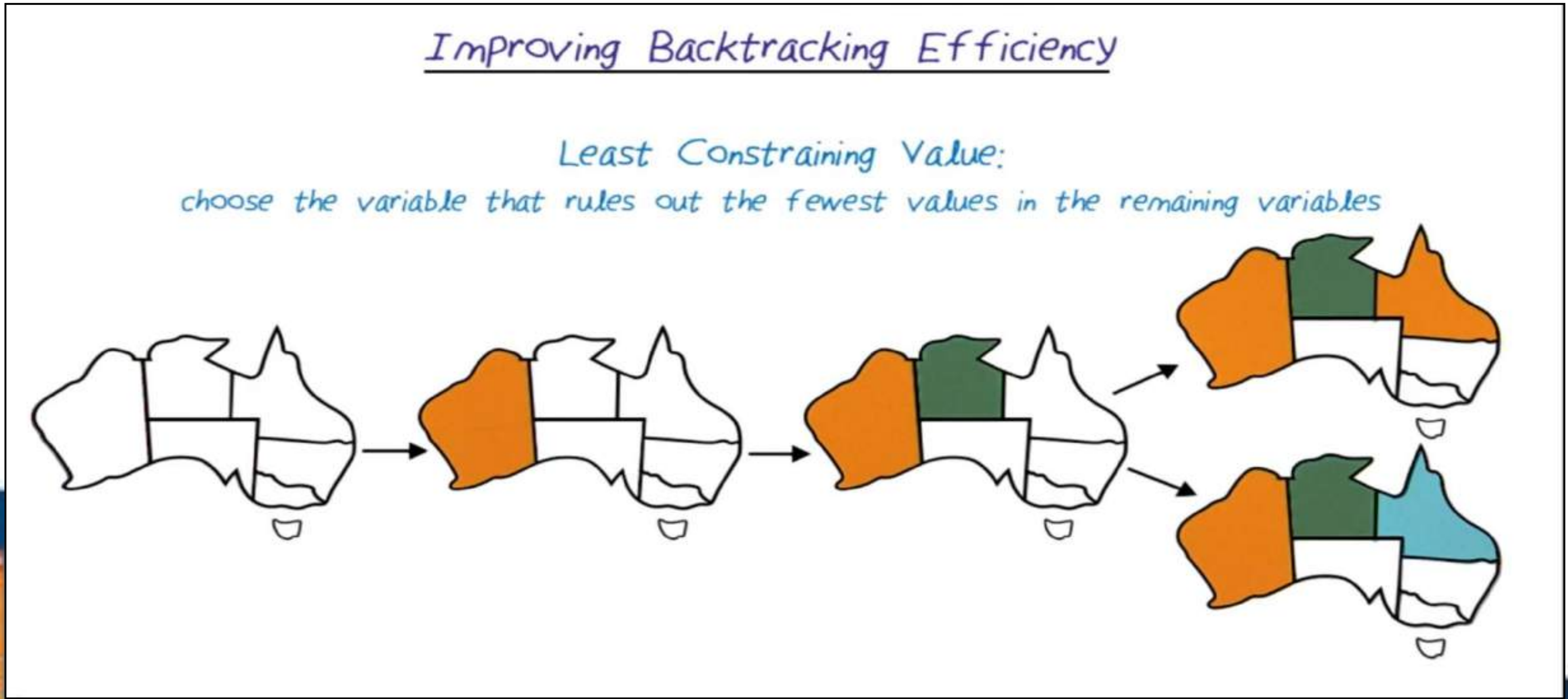


Improving Backtracking Efficiency

Minimum Remaining Values (MRV):
choose the variable with the fewest legal values

In order to determine the order of the variable to be assigned, minimum remaining-value heuristic is adopted. It means that we choose the most constrained variable, i.e, choose the variable with the fewest legal values.

least-constraining-value (LCV) is used to assign value for a variable. Specically, we choose the value that rules out the fewest values in the remaining variables, that is to say, leave more space for the rest variables. In general, the heuristic is trying to leave the maximum exibility for subsequent variable assignments.



Improving Backtracking Efficiency

Least Constraining Value:
choose the variable that rules out the fewest values in the remaining variables

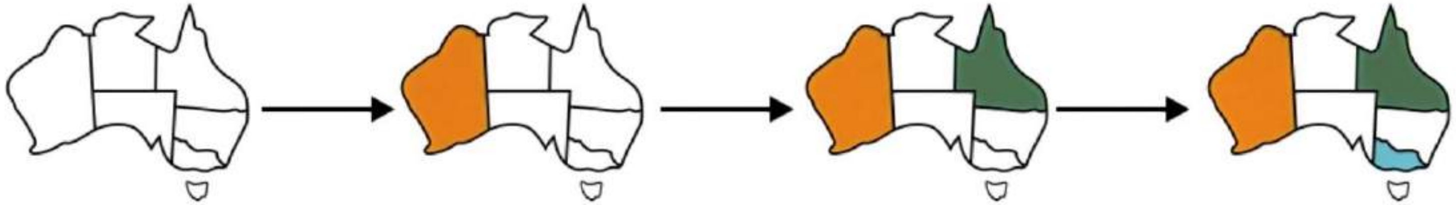
PRESIDENCY UNIVERSITY

LCV H.

It prefers the value that rules out the fewest choic for the neighbouring variables in the Constraint graph

(prefers flexibility for the Future)

Forward Checking

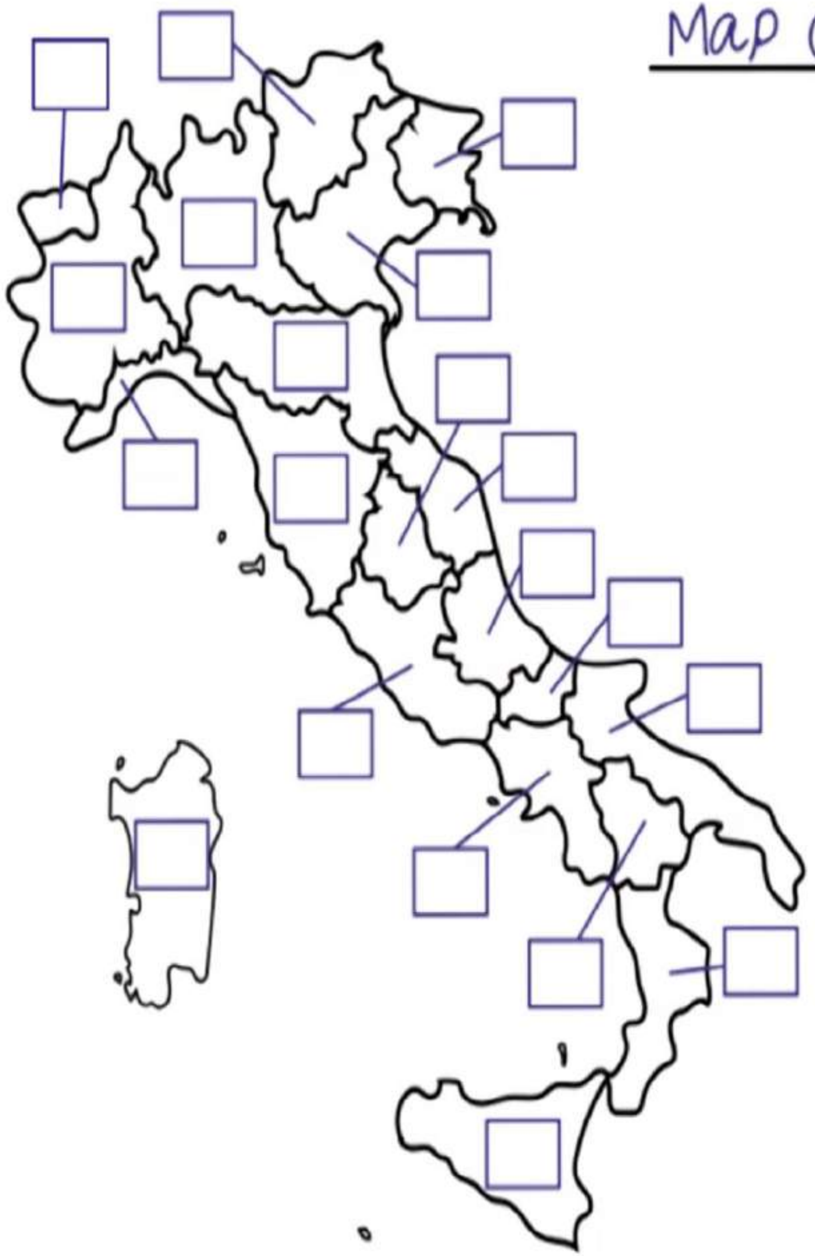


WA	NT	Q	NSW	V	SA	T
🟧🟩🟦	🟧🟩🟦	🟧🟩🟦	🟧🟩🟦	🟧🟩🟦	🟧🟩🟦	🟧🟩🟦
🟧	🟩🟦	🟧🟩🟦	🟧🟩🟦	🟧🟩🟦	🟩🟦	🟧🟩🟦
🟧	🟦	🟩	🟧🟦	🟧🟩🟦	🟦	🟧🟩🟦
🟧	🟦	🟩	🟧	🟦		🟧🟩🟦

Keep track of remaining legal values for unassigned variables

 Terminate search when any variable has no legal values

Map Coloring Quiz

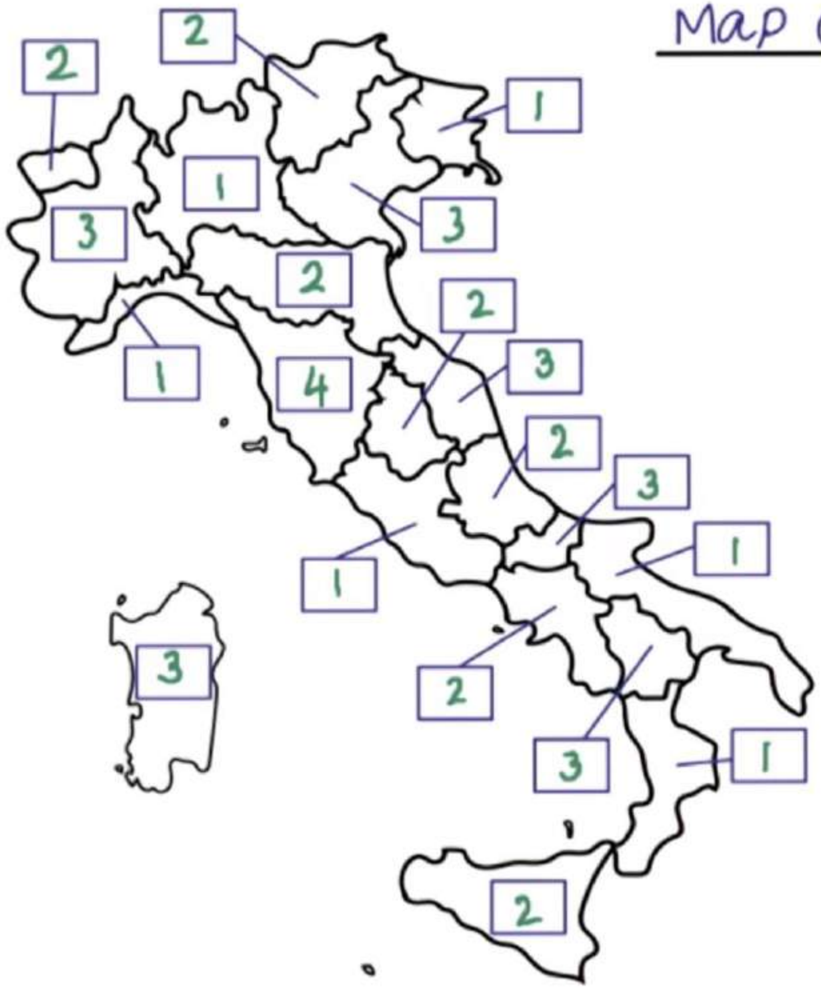


'Color' each region by typing in a number.
(e.g. 1, 2, 3, .. etc.)

What is the minimum number of colors needed for this map?

□

Map Coloring Quiz



'Color' each region by typing in a number.
(e.g. 1, 2, 3, .. etc.)

What is the minimum number of colors needed for this map?

$\boxed{4}$

Types of Constraints

- Types:
  - Unary
  - Binary
  - Constraints involving 3 or more variables

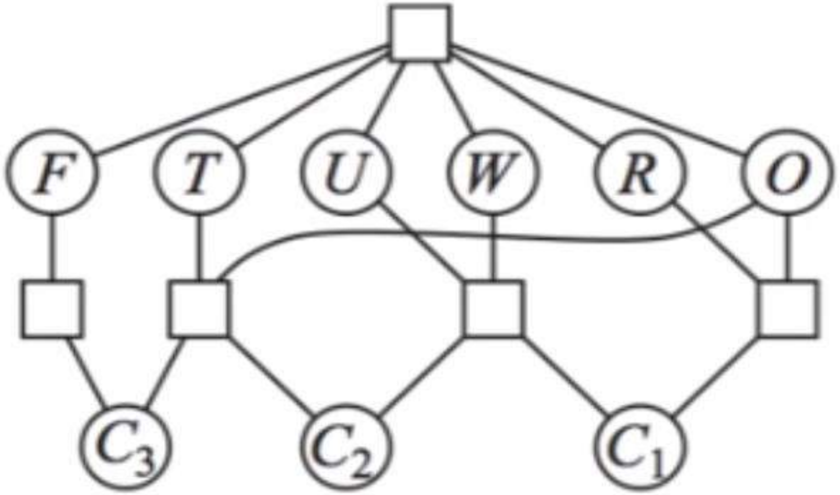
Cryptarithmetic Puzzle

- Cryptarithmetic Problem is a type of constraint satisfaction problem where the game is about digits and its unique replacement either with alphabets or other symbols.
- In **cryptarithmetic problem,** the digits (0-9) get substituted by some possible alphabets or symbols.
- The task in cryptarithmetic problem is to substitute each digit with an alphabet to get the result arithmetically correct.

- **The rules or constraints on a cryptarithmetic problem are as follows:**
- There should be a unique digit to be replaced with a unique alphabet.
- The result should satisfy the predefined arithmetic rules, i.e., 2+2 =4, nothing else.
- Digits should be from **0-9** only.
- There should be only one carry forward, while performing the addition operation on a problem.
- The problem can be solved from both sides, i.e., **lefthand side (L.H.S), or righthand side (R.H.S)**

Cryptarithmetic Puzzle

$$\begin{array}{r} T\;W\;O \\ +\;T\;W\;O \\ \hline F\;O\;U\;R \end{array}$$



“Reasoning”?

- Reasoning is an act of deriving a conclusion from certain premises using a given methodology.
- Reasoning is a process of thinking; logically arguing; drawing inference.
- When a system is required to do something, that it has not been explicitly told how to do, it must reason. It must figure out what it needs to know from what it already knows.
- Many types of reasoning have been identified and recognized, but many questions regarding their logical and computational properties still remained under controversy.
- Popular methods of Reasoning include: Abduction, Induction, model- based, explanation and confirmation. All of them are intimately related to problems of belief revision and theory development, knowledge assimilation, discovery and learning.

- When a system is required to do something, that it has not been explicitly told how to do, it must reason. It must figure out what it needs to know from what it already knows.

  Example: Fact-1 : Robins are birds, Fact-2 : All birds have wings.

Then we can ask: DO ROBINS HAVE WINGS?

- Reasoning is an act of deriving a conclusion from certain premises using a given methodology.
  - Any knowledge system must reason, if it is required to do something which has not being told explicitly.
  - For reasoning system must find out, what it needs to know from what it already knows.

Hence to answer the above question- some reasoning must go.

Human reasoning capabilities

- Broadly it is being divided into three areas:
  - Mathematical Reasoning- *axioms, definitions, theorems, proofs*
  - Logical Reasoning- *deductive, inductive, abductive*
  - Non-logical Reasoning- linguistic, language
- Above three mentioned are in every human being, but the ability level depends on education, genetics and environment.
- Intelligent Quotient (IQ)= mathematical + logical reasoning, whereas, Emotional Quotient mostly depends on non-logical reasoning capabilities.
- Logical Reasoning is our major concern in AI.

Logical Reasoning

- **Logic** is a language of reasoning. It is a collection of rules called logic arguments, we use when doing logical reasoning.
- **Logical Reasoning** is a process of drawing conclusions from premises using rule of inference.
- The study of is divided into two: **formal and informal logic.**
- The formal logic is sometimes called symbolic logic.
- **Symbolic logic** is the study of symbolic abstraction (construct) that capture the formal features of logical inference by a formal system.
- **Formal system** consists of two components, a formal language and a set of inference rules. The formal system has axioms.
- **Axiom** is a sentence that is always true within the system.
- **Sentences** are derived using the system's axioms and rules of derivation are called theorems.

Formal Vs Informal Logic

Formal Logic

- The formal logic is the study of inference with purely formal content, i.e. where content is made explicit. Eg: Propositional logic and Predicate Logic.
- Here the logical arguments are set of rules for manipulating symbols. The rules are of two types:
  - Syntax rules: how to build meaningful expressions.
  - Inference rules: how to obtain true formulas from other true formulas.
- Logic also need **semantics,** which says how to assign meaning to expressions.

Informal Logic

- The informal logic is study of natural language arguments.
- The analysis of argument structures in ordinary language is part of informal logic
- The focus lies in distinguishing good arguments (valid) and bad arguments (invalid).

Formal systems

- Formal system can have following three properties:
  - **Consistency:** system's theorems do not contradict
  - **Soundness:** system's rules of derivation will never infer anything false, so long as start is with only true premises.
  - **Completeness:** there are no true sentences in the system that cannot be proved using derivation rules of the system.
- **Elements of formal systems:**
  - The finite set of symbols for constructing formulae.
  - A grammar, is the way of constructing well-formed formulae (wff).
  - A set of axioms; each axiom has to be a wff.
  - A set of inference rules.
  - A set of theorems.
- A well-formed formulae, wff, is any string generated by a grammar. Example: the sequence of symbols $((a \rightarrow b) \rightarrow (\sim b \rightarrow \sim a))$ is a WFF because its is grammatically correct in propositional logic.

Formal Language

- A formal language may be viewed as being analogous to a collection of words or a collection of sentences.
- In computer science, a formal language is defined as precise mathematical or machine process able formulas.
- A formal language Lis characterized as a set Fof finite length sequences of elements drawn from a specified finite set A of symbols.
- The mathematical theory that treats formal language in general is known as **formal language theory.**

Uncertainty in Reasoning

- The world is an uncertain places; often the knowledge is imperfect which causes uncertainty. Therefore reasoning must be able to operate under uncertainty.
- AI systems must have ability to reason under conditions of uncertainty.
  - Incompleteness knowledge→ compensate for lack of knowledge.
  - Inconsistencies knowledge→ Resolve ambiguities and contradictions
  - Changing knowledge→ update the knowledgebase over time.

Methods of reasoning

- To a certain extend this will depend on the chosen knowledge representation. Although a good knowledge representation scheme has to allow easy, natural, and plausible reasoning.
- Three types of logical reasoning:
  - Deduction
  - Induction
  - Abduction

Deductive Reasoning

- Deductive Reasoning is a process in which general premises are used to obtain a specific inference. Reasoning moves from a general principle to a specific conclusion.
- Example: Premise : I wash my car when the weather is good on weekends.
- Premise: Today is Sunday and the weather is hot
- Conclusion: Therefore, I will wash my car today.
- To use deductive reasoning the problem must generally be formatted in this way. Once the format has been achieved , the conclusion must be valid if the premises are true. The whole idea is to develop new knowledge from previously given knowledge.
- One of the basic rules of inference of deductive logic is the **modus ponens** rule.
- A formal English statement of this rule is : If X is true and if X being true implies Y is true then Y is true. $(X \land (X \rightarrow Y)) \rightarrow Y$
- Example: All cats are felines. Bosty is a cat. I can deduce that **Bosty is a feline**

Abduction Reasoning

- Abduction is a form of deductive logic which provides only a ‘plausible inference’. For instance:

If I read Smoking causes lung cancer and Frank died of lung cancer , I may infer that Frank was a smoker. Again this may or may not be true. Using statistics and probability theory , abduction may yield the most probable inference among many possible inferences.

- Abduction is heuristic in the sense that it provides a plausible conclusion consistent with available information, but one which may in fact be wrong. To illustrate how abduction works, consider following logical system consisting of a general rule and a specific proposition:

1)All successful , entrepreneurial industrialists are rich persons.

2)John is a rich person.

- If this was only information available, a plausible inference would be that **John was a successful , entrepreneurial industrialist.** This conclusion could also be false since there are other roads to riches such as inheritance , the lottery...If we had a table of the income distribution of wealthy persons along with their personal histories, we could refine our abduction inference with the probability of the inference being true .

Inductive Reasoning

- A principle of reasoning to a conclusion about all members of a class from examination of only a few members of the class; broadly, reasoning from the particular to the general.
- For example :
- In 1998, The best model of Turkey is from İzmir In 1999, The best model of Turkey is from İzmir In 2000, The best model of Turkey is from İzmir I would logically infer that all the girls from İzmir is beautiful.
- This may or may not be true. But it provides a useful generalization.
- Another example :
- Falcon can fly. Canary can fly. Gull can fly.
- Conclusion: Birds can fly.
- The outcome of the inductive reasoning process will frequently contain some measure of uncertainty because including all possible facts in the premises is usually impossible. Deductive or inductive approaches are used in logic, rule-based systems, and frames.

Other types of Reasoning

- **Analogical Reasoning:** Analogical reasoning assumes that when a question is asked, the answer can be derived by analogy.
- Example: Premise : All football teams gets 3 point when they win.

Question: How many points did GS take this weekend?

Conclusion: Because GS is a football team and won, hence, they took 3 points.

- Analogical reasoning is a type of verbalization of an internalized learning process. An individual uses processes that require an ability to recognize previously encountered experiences. The use of this approach has not been exploited yet in AI field. However, case-based reasoning is an attempt.

Other types of Reasoning

- **Formal Reasoning:** Formal reasoning involves syntactic manipulation of data structures to deduce new facts. A typical example is the mathematical logic used in proving theorems in geometry.
- **Procedural Numeric Reasoning:** Procedural numeric reasoning uses mathematical models or simulation to solve problems. Model-based reasoning is an example of this approach.
- **Generalization and Abstraction:** Generalization and abstraction can be successfully used with both logical and semantic representation of knowledge.
- **Meta level Reasoning:** Meta level reasoning involves “knowledge about what you know”. Which approach to use, how successful the inference will be, depends to a great extent on which knowledge representation method is used. For example; reasoning by analogy can be more successful with semantic networks than with frames.

Reasoning with LOGIC

- We utilize various rules of inference to manipulate the logical expressions to create new expressions.
- **Modus Ponens**
- If there is a rule “if A, then B,” and if we know that A is true, then it is valid to conclude that B is also true. [A AND (A →B)] →B
- Example : A: It is rainy. B: I will stay at home. A→B : If is rainy, I will stay at home.
- **Modus Tollens**
- When B is known to be false, and if there is a rule "if A, then ,” it is valid to conclude that A is also false.
- **Resolution** is a method of discovering whether a new fact is valid, given a set of logical statements. It is a method of “theorem proving”. The resolution process, which can be computerized because of its well-formed structure, is applied to a pair of parent clauses to produce a derived new clause.

More about Reasoning

- **Monotonic Reasoning [Preferred in Certainty]-** Monotonic reasoning is static, in other words the truth of the statement doesn’t change when any new information is added.
- **Non-Monotonic Reasoning [Preferred in uncertainty]-** In real life we frequently deal with non-monotonic reasoning. In these situations the truth of astatement can change when a new information is added.
- **Shallow and Deep Reasoning.**
- **Inheritance Method**
- **Pattern- Matching Method**
- **Meta-Rules:** Meta-rules are the rules about rules. They are used for deciding about which rules to use next. Meta-rules make the knowledge base harder to read and understand because one meta-rule affects the sequence in which all other rules are called.
- **Conflict Resolution**

Alternative Reasoning Methods

- **Theorem Proving**
  - emphasis on mathematical proofs and correctness, not so much on performance and ease of use
- **Probabilistic Reasoning**
  - integrates probabilities into the reasoning process
- **Certainty Factors**
  - Express subjective assessment of truth of fact or rule
- **Fuzzy Reasoning**
  - allows the use of vaguely defined predicates and rules

Reasoning Types

- Model –based Reasoning
- Case-based Reasoning
- Rule-based Reasoning
- Blackboard Systems.
- Truth Maintenance systems.

Model Based Reasoning

- In artificial intelligence, **model-based reasoning** refers to an inference method used in expert systems based on a model of the physical world.
- With this approach, the main focus of application development is developing the model.
- Then at run time, an "engine" combines this model knowledge with observed data to derive conclusions such as a diagnosis or a prediction.
- In a model-based reasoning system knowledge can be represented using **causal rules**.

Case-Based Reasoning

- **Case-based reasoning** (CBR), broadly construed, is the process of solving new problems based on the solutions of similar past problems.
- An auto mechanic who fixes an engine by recalling another car that exhibited similar symptoms is using case-based reasoning.
- A lawyer who advocates a particular outcome in a trial based on legal precedents or a judge who creates case law is using case-based reasoning.

Rule-based Reasoning

- A particular type of reasoning which uses "if-then-else" rule statements.
- As mentioned above, rules are simply patterns and an inference engine searches for patterns in the rules that match patterns in the data.
- The "if" means "when the condition is true," the "then" means "take action A" and the "else" means "when the condition is not true take action B."
- Here is an example with the rule PROBABLE CAUSE:
- IF robbery is TRUE AND suspect witness identification is TRUE AND suspect physical evidence is TRUE AND suspect lacks alibi is TRUE THEN probable cause is TRUE ELSE round up usual suspects

Black-board Reasoning

- A **blackboard system** is an artificial intelligence application based on the blackboard architectural model, where a **common knowledge base,** the "blackboard", is iteratively updated by a diverse group of specialist knowledge sources, starting with a problem specification and ending with a solution.
- Each knowledge source updates the blackboard with a partial solution when its internal constraints match the blackboard state.
- In this way, the specialists work together to solve the problem.
- The blackboard model was originally designed as a way to handle complex, ill-defined problems, where the solution is the sum of its parts.