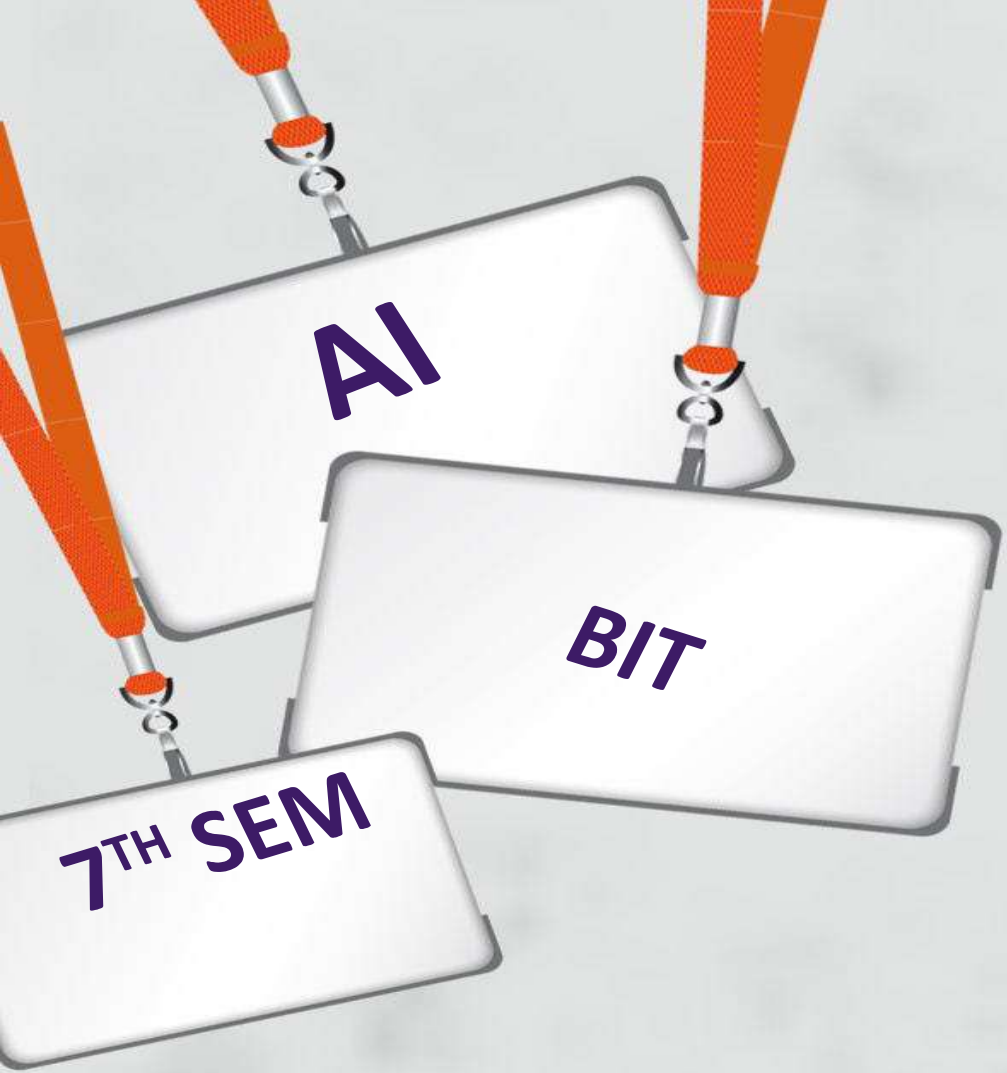


Chapter 3

PROBLEM SOLVING

Compiled by Er. Mohan Bhandari

Planning

- Planning is the process of deciding in detail how to do something before you actually start to do it.
- It is the task of coming up with a sequence of actions that will achieve a goal.
- **Planner** is viewed as the producer or generator of the solution.
- Eg: STRIPS (STandford Research Institute Problem Solver)

- **Classical Planning**

  Planning for which environments that are fully observable, deterministic, finite, static (change happens only when the agent acts), and discrete (in time, action, objects, and effects)

- **Non Classical Planning**

  Planning for which environments for partially observable and involves a different set of algorithms and agent designs

Problems

- A problem is a situation (difficult/easy) experienced by an agent.
- Problem is solved by a sequence of actions that reduce the difference between the initial situation and the goal.

Problem Solving

- Problem solving, particularly in artificial intelligence, may be characterized as a systematic search through a range of possible actions in order to reach some predefined goal or solution.
- Problem-solving methods divide into special purpose and general purpose.
- A special-purpose method is tailor-made for a particular problem and often exploits very specific features of the situation in which the problem is embedded.
- In contrast, a general-purpose method is applicable to a wide variety of problems.

Four general steps in problem solving:

- Goal formulation
  - What are the successful world states
- Problem formulation
  - What actions and states to consider given the goal
- Search
  - Determine the possible sequence of actions that lead to the states of known values and then choosing the best sequence.
- Execute
  - Give the solution perform the actions.

Problem formulation:

A problem is defined by:

- An initial state: State from which agent start
- Successor function: Description of possible actions available to the agent.
- Goal test: Determine whether the given state is goal state or not
- Path cost: Sum of cost of each path from initial state to the given state.

A solution is a sequence of actions from initial to goal state. Optimal solution has the lowest path cost.

Defining problem as a state space search

A set of initial state, actions and the goal state for a given problem is known as state space for that problem. A state space represents a problem in terms of states and operators that changes the states. A state space essentially consists of a set of nodes representing each state of the problem, arcs between nodes representing the legal moves from one state to another, an initial state and a goal state

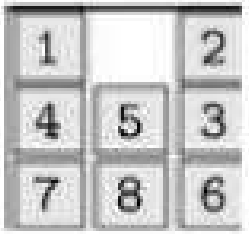
The major components of state space representation are:

- Initial state
- Goal state, and
- Operator or legal moves.

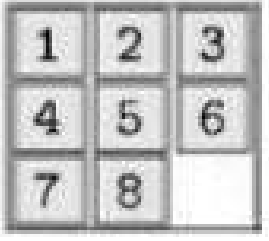
Q. Explain problem as a state space with an example.

To explain problem as a state space representation in more detailed manner, let us consider a problem of 8puzzle game. The puzzle consists of an 8-square frames and an empty slot. The tiles are numbered from 1-8. It is possible to move the tiles in the square field by moving tile into the empty slot. The objective is to get square in a numeric order.

- **Initial State**



- **Operators**
  - UP: If the empty slot is not touching the up-frame, move it up.
  - DOWN: If the empty slot is not touching down-frame, move it down.
  - LEFT: If the empty slot is not touching left-frame, move it left.
  - RIGHT: If the empty slot is not touching right-frame, move it right.
- **Final State**



Planning	Problem Solving
The task of coming up with a sequence of actions that will achieve a goal is called planning	Problem Solving is the systematic search through a range of possible actions in order to reach some predefined goal or solution
Planning is a generic term of problem solving	Problem solving comprises standard search process
Planning is involved in plan generation	Problem solving is typically more concerned with plan execution
Eg: STRIPS	Eg: A* Search

Q. Vacuum World Problem

The world has only two *locations*

Each location may or may not contain *dirt*

The agent may be in one location or the other

8 possible *world states*

Three possible actions: *Left, Right, Suck*

Goal: clean up all the dirt

Here,

- **Initial States**:

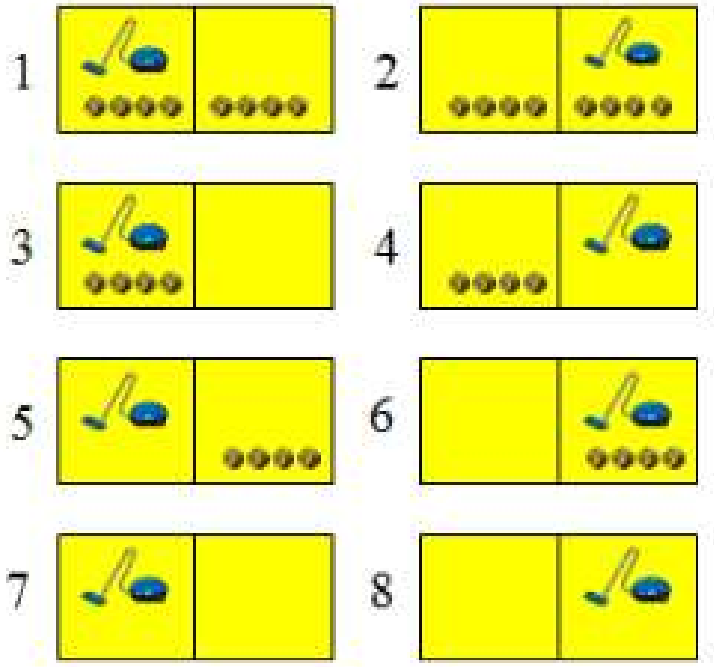
  One of the 8 states

- **Operators:**

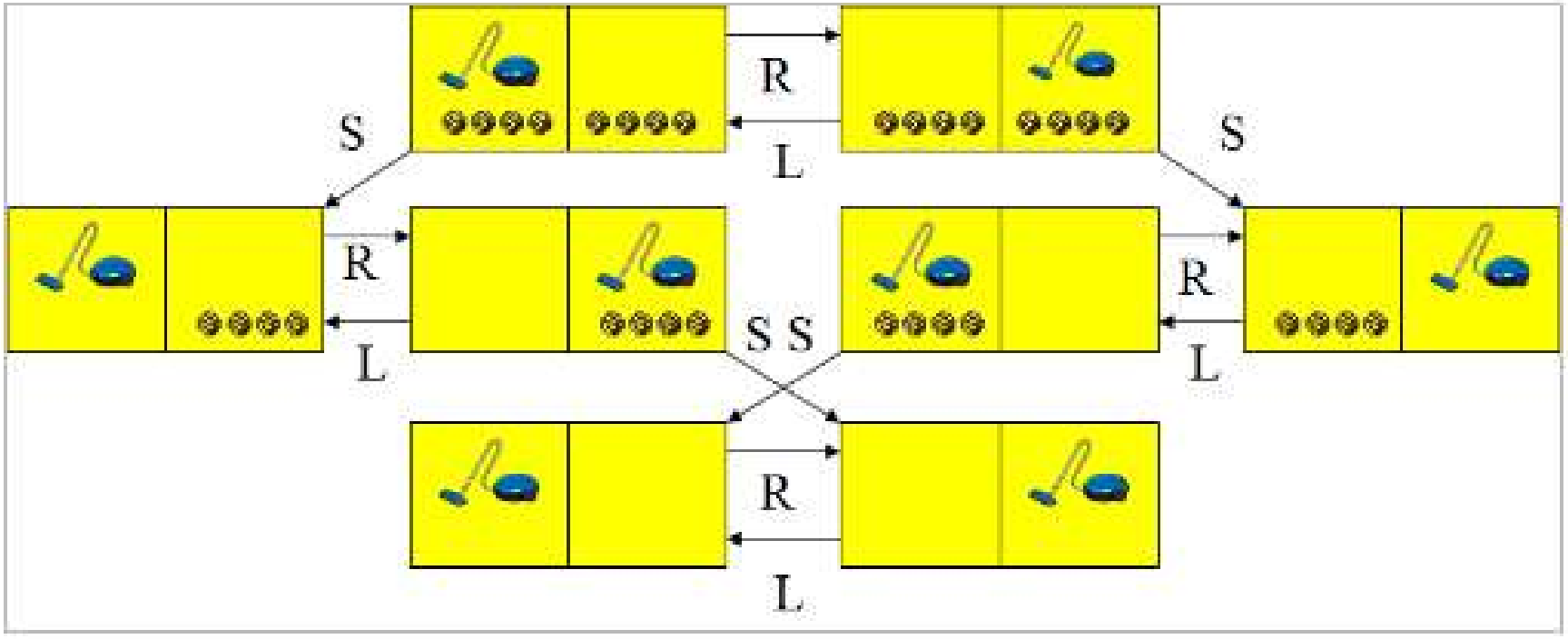
  Move left, Move right, Suck

- **Final State**:

  No dirt left in any square



- **Solution**



Q. You are given two jars of a six gallons and eight gallons capacity. There is no marking on the jars. There is a water tap, which can be used to fill jar. Your goal is to have exactly four gallons of water in eight gallons jar without taking any other jar or measuring device.

Q. River Crossing Problem

In a distinct land, bigamy is common. There are six people who want to cross a river in this land. This group consists of two men, each two wives. No man can tolerate any of his wives being in the company of another man unless yet least he or his next wife is present in the boat or in next land. There is a boat that holds two people to be used for crossing the river. How is the trip possible?

- **Initial State**

  W { H1, W1, W1' and H2, W2, W2'}, E {ϕ}

- **Operation**

  - No man can tolerate any of his wives being in the company of another man.
  - Crossing the river

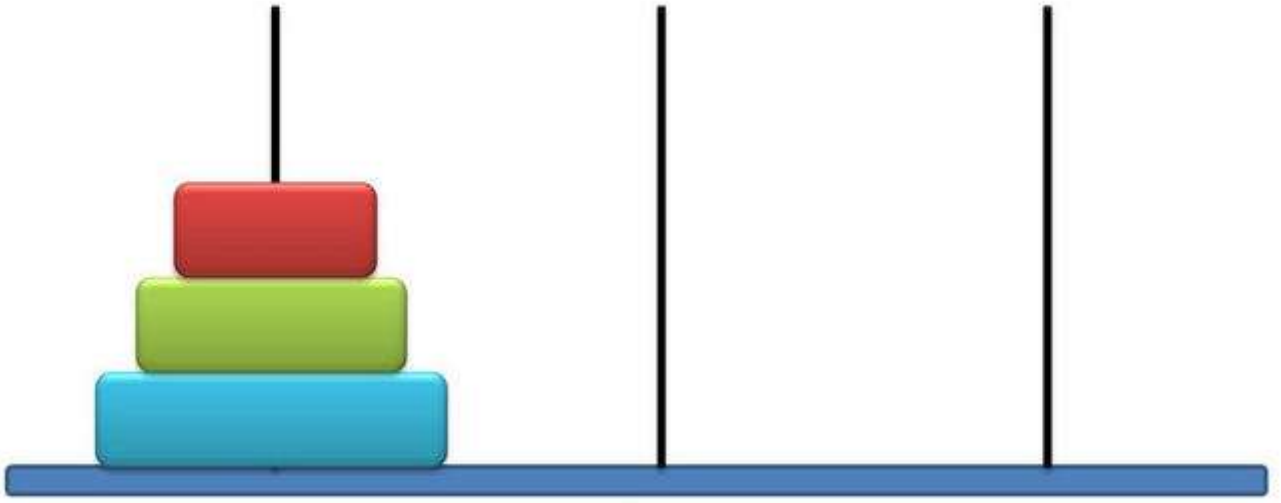
- **Final State**

  W { ϕ}, E { H1, W1, W1' and H2, W2, W2'}

- **Solution**

S. N	West (W)	River		East (E)
0.	H1,W1,W1', H2,W2,W2'			(ϕ)
1.	H1,H2,W2,W2'	W1, W1'		W1,W1'
2.	H1,W1,H2,W2,W2'	W1,		W1'
3.	H1,W1,H2,	W2, W2'		W1',W2,W2'
4.	H1,W1,H2,W2	W2		W1',W2'
5.	H1, H2,	W1, W2		W1,W1',W2,W2'
6.	H1, H2, W2	W2		W1,W1', W2'
7.	H1,	H2, W2		W1,W1', H2,W2, W2'
8.	H1,H2,	H2,		W2
9.	(ϕ)	H1, H2,		H1, W1,W1', H2,W2, W2'

Q. Pegs and Disks problem



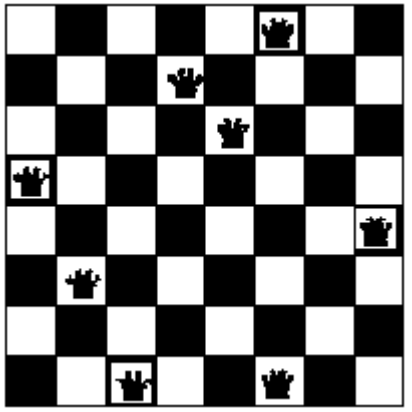
Q.8 queen’s problem

The problem is to place 8 queens on a chessboard so that no two queens are in the same row, column or diagonal

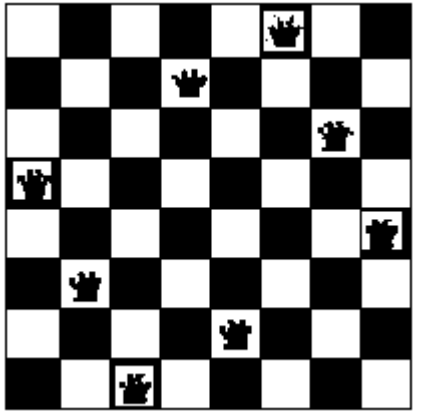
Here,

- **Initial State**
- **Operation**

  Add a queen in any square



- **Final State**



SEARCHING

- Searching is the process of finding the required states or nodes.
- Searching is to be performed through the state space.
- Search process is carried out by constructing a search tree.
- Search is a universal problem-solving technique.

SEARCH TERMINOLOGY

- **Problem Space**: Environment in which the search takes place.
- **Problem Instance:** It is Initial state + Goal state
- **Problem Space Graph:** It represents problem state. States are shown by nodes and operators are shown by edges.
- **Depth of a problem:** Length of a shortest path or shortest sequence of operators from Initial State to goal state.

- **Space Complexity:** The maximum number of nodes that are stored in memory.
- **Time Complexity:** The maximum number of nodes that are created.
- **Admissibility:** A property of an algorithm to always find an optimal solution.
- **Branching Factor:** The average number of child nodes in the problem space graph.

CLASSIFICATION

Uninformed Search (Blind Search/Brute force search)

The search algorithms that do not use any extra information regarding the problem

- ✓ Depth First Search
- ✓ Breath First Search
- ✓ Depth Limit Search
- ✓ Iterative deepening
- ✓ Uniform Cost
- ✓ Bidirectional Search

Informed search or Heuristic search

Informed search have problem specific knowledge apart from problem definition

- ✓ Hill climbing Search
- ✓ Best first Search
  - Greedy Best First Search
  - A* Search
- ✓ Simulated Annealing

DEPTH FIRST SEARCH (DFS)

- Proceeds down a single branch of the tree at a time
- Expands the root node, then the leftmost child of the root node
- Always expands a node at the deepest level of the tree
- Only when the search hits a dead end (a partial solution which can't be extended), the search backtrack and expand nodes at higher levels.

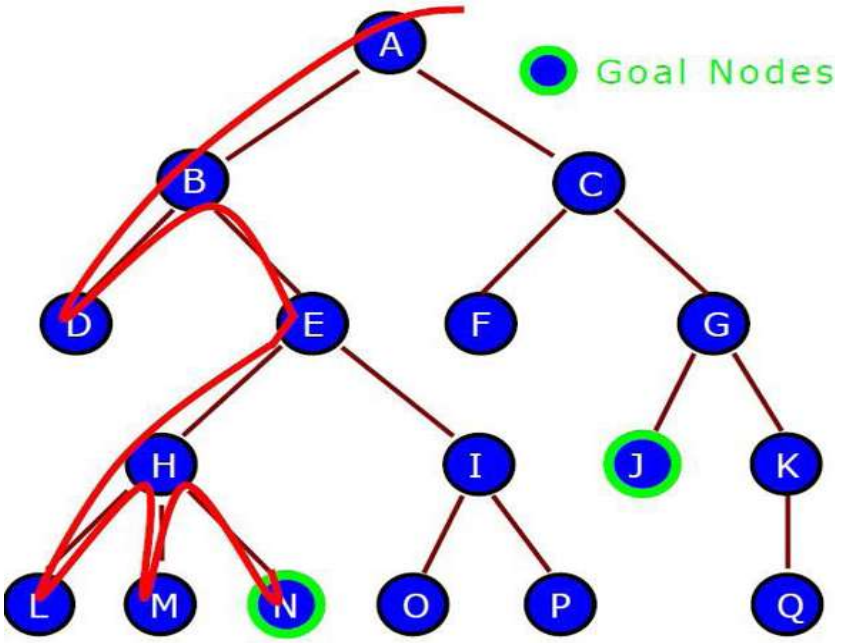


Fig. Depth-first search (DFS)

DEPTH FIRST SEARCH (DFS)

- ***Completeness***: Incomplete as it may get stuck going down an infinite branch that doesn't leads to solution.
- ***Optimality***: The first solution found by the DFS may not be shortest.
- ***Space complexity***: b as branching factor and d as tree depth level, Space complexity = $\mathbf{O(b^{d+1})}$
- ***Time Complexity***:

$$\mathbf{b + b^2 + b^3 + \ldots\ldots\ldots\ldots\ldots + b^d + b^{d+1} = O(b^{d+1})}$$

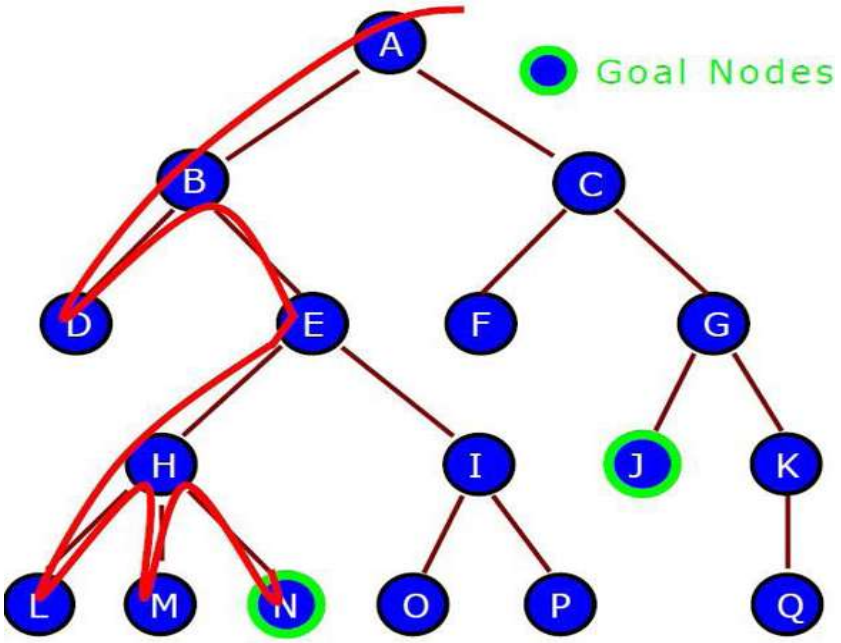
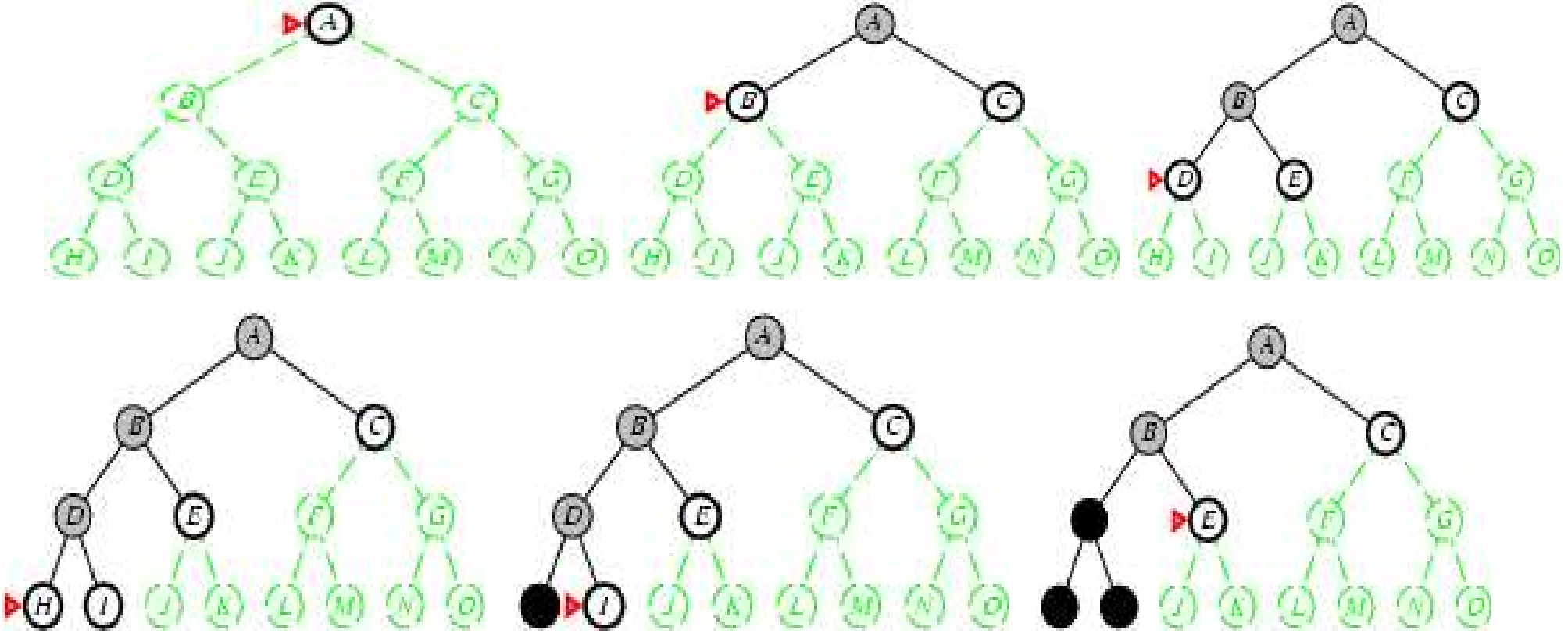


Fig. Depth-first search (DFS)

DFS example (find path from A to E)



Breadth-First Search (BFS)

- Proceeds level by level down the search tree
- Starting from the root node (initial state) explores all children of the root node, left to right
- If no solution is found, expands the first (leftmost) child of the root node, then expands the second node and so on

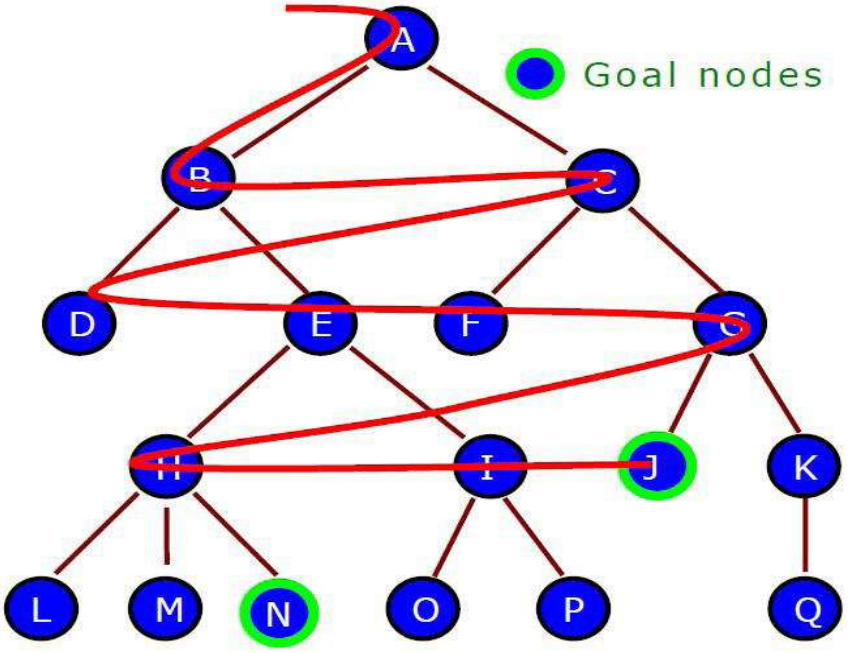


Fig. Breadth-first search (BFS)

Breadth-First Search (BFS)

- Completeness: Complete if the goal node is at finite depth
- Optimality: It is guaranteed to find the shortest path
- Time Complexity: $b + b^2 + b^3 + \ldots\ldots\ldots\ldots\ldots b^m = O(b^m)$
- Space Complexity: (1+ b + b + b + ... b) m times = O(b m)

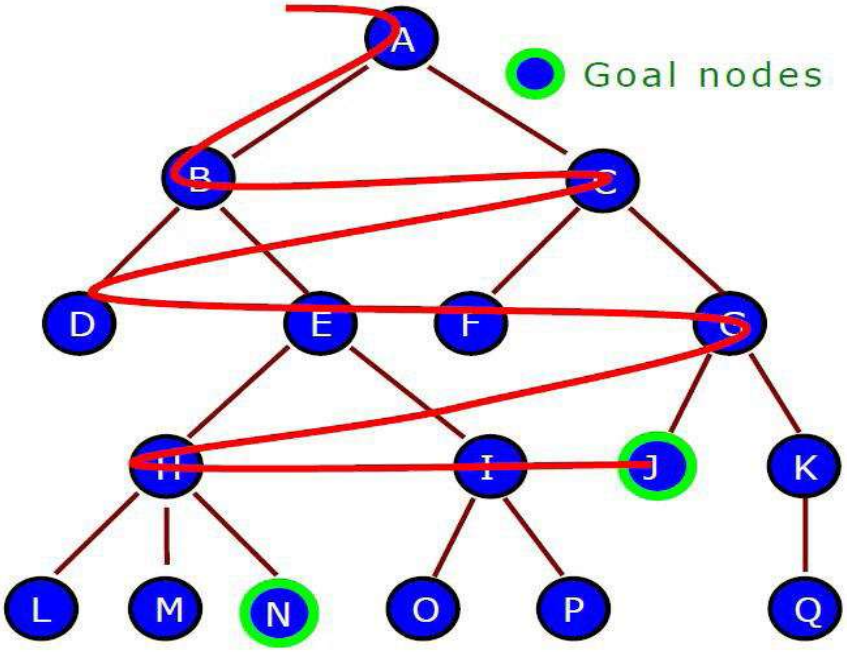
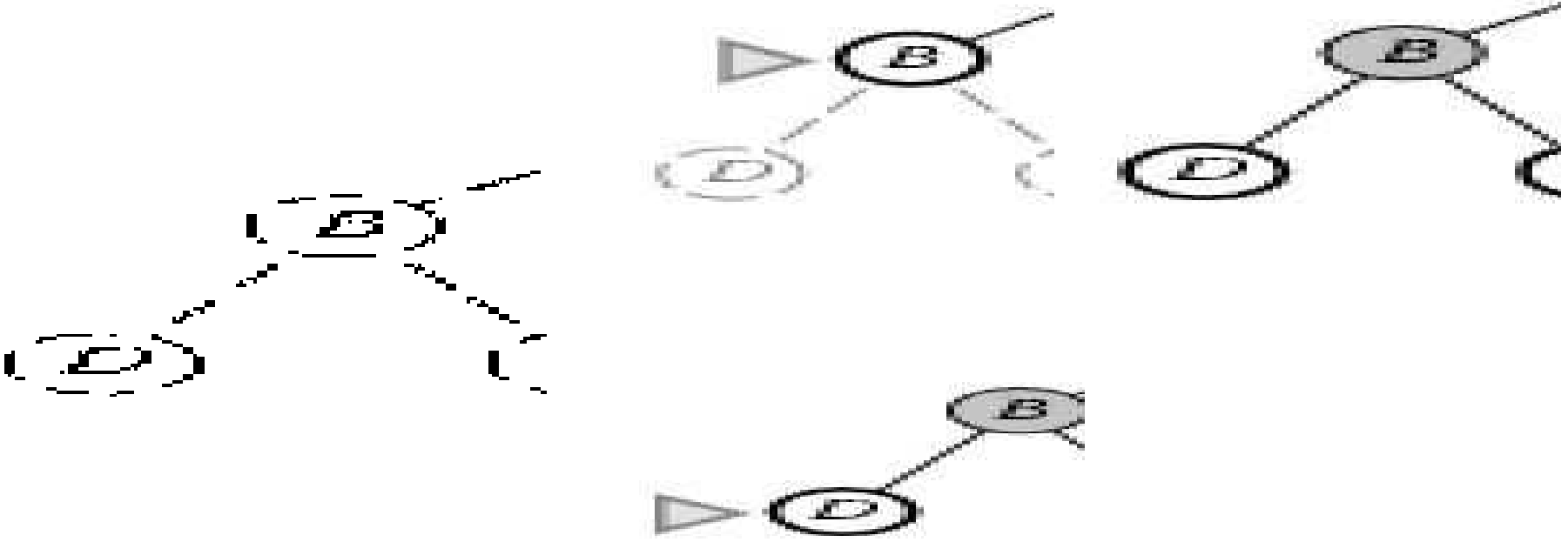


Fig. Breadth-first search (BFS)

BFS example (Find path from A to D)

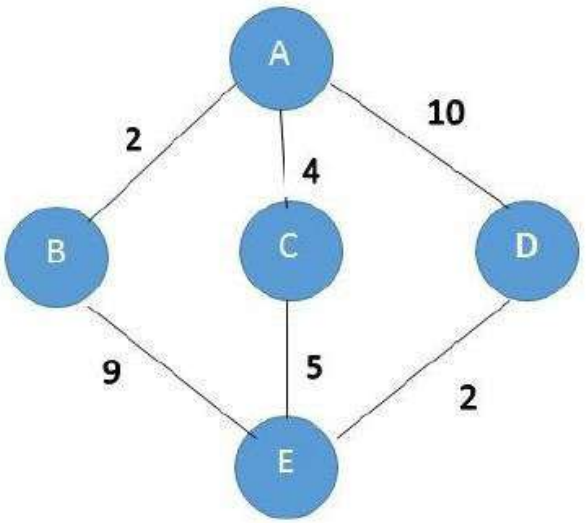


Uniform-Cost Search

Uniform-cost is guided by path cost rather than path length like in BFS, the algorithms starts by expanding the root, then expanding the node with the lowest cost from the root, the search continues in this manner for all nodes.

- **Completeness:**
  Complete if the cost of each step exceeds some small positive integer, this to prevent infinite loops.
- **Optimality:**
  Optimal in the sense that the node that it always expands is the node with the least path cost.
- **Time Complexity:** $O(b^{C/e})$.
- **Space Complexity:** $O(b^{C/e})$

UCS example (Find path from A to E)



- Expand A to B, C, and D.
- The path to B is the cheapest one with path cost 2.
- Expand B to E
- Total path cost = 2+9 =11
- This might not be the optimal solution since the path AC as path cost 4 (less than 11)

- Expand C to E
- Total path cost = 4+5 =9
- Path cost from A to D is 10 (greater than path cost, 9)
- Hence optimal path is ACE.

Depth Limit Search

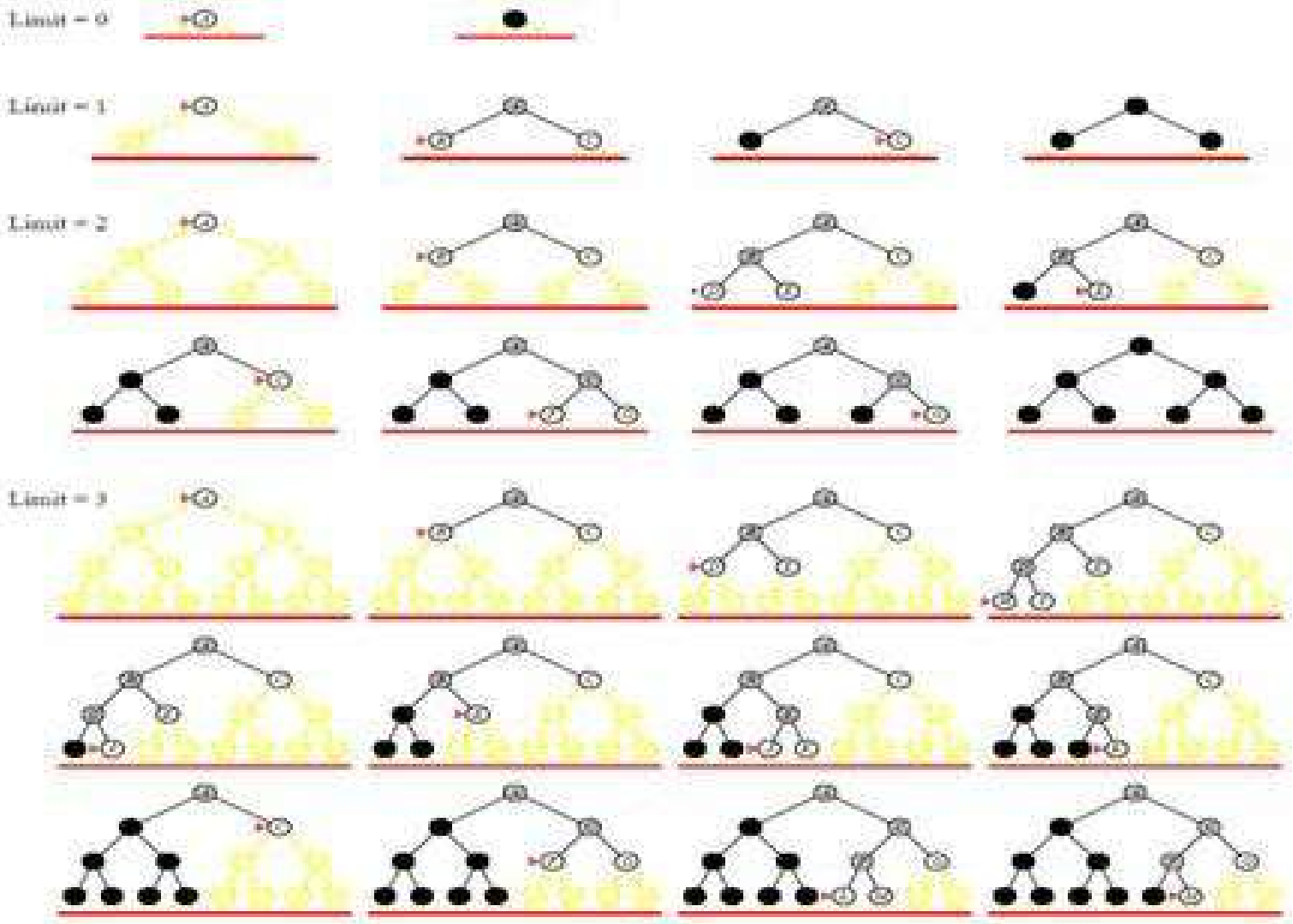
- Depth-first search will not find a goal if it searches down a path that has infinite length. So, in general, depth-first search is not guaranteed to find a solution, so it is not complete.
- This problem is eliminated by limiting the depth of the search to some value L. However, this introduces another way of preventing depth-first search from finding the goal: if the goal is deeper than L it will not be found.
- Perform depth first search but only to a pre-specified depth limit L.
- No node on a path that is more than L steps from the initial state

Depth Limit Search

- ***Completeness***: Incomplete as solution may be beyond specified depth level.
- ***Optimality***: not optimal
- ***Space complexity***: b as branching factor and L as tree depth level, O (b.L)
- ***Time Complexity***: O(bL)

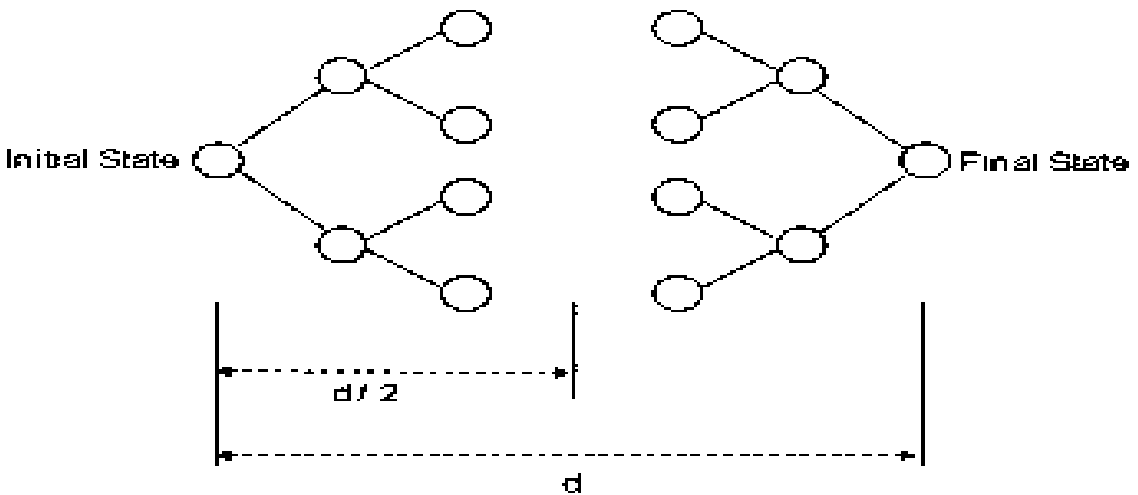
Iterative Deepening Search (IDS)

- Iterative deepening search is a strategy that sidesteps the issue of choosing the best depth limit by trying all possible depth limits: first depth 0, then depth 1, then depth 2, and so on.
- In effect, iterative deepening combines the benefits of depth-first and breadth-first search.
- It is optimal and complete, like breadth-first search, but has only the modest memory requirements of depth-first search.


Limit = 0
Limit = 1
Limit = 2
Limit = 3

Bidirectional Search

- As the name suggests, bidirectional search suggests to run 2 simultaneous searches
- One from the initial state and the other from the Final state
- Those 2 searches stop when they meet each other at some point in the middle of the graph.



Bidirectional Search

- Completeness:

  Bidirectional search is complete when we use BFS in both searches

- Optimality:

  Like the completeness, bidirectional search is optimal when BFS is used.

- Time/Space Complexity : $O(b^{d/2})$

Informed Search (Heuristic Search)

- Informed search have problem specific knowledge apart from problem definition.
- They use experimental algorithm which improves efficiency of search process.
- The idea is to develop a domain specific heuristic function h(n) where h(n) guesses the cost of getting to the goal from node n.

Heuristic Function

The heuristic function is a way to inform the search about the direction to a goal. It provides an informed way to guess which neighbor of a node will lead to a goal.

Best-First Search (BFS)

- Best-First search is a graph-based heuristic search algorithm
- The name “best-first” refers to the method of exploring the node with the best “score” first.
- An evaluation function is used to assign a score to each candidate node. The evaluation function must represent some estimate of the cost of the path from state to the closest goal state

Algorithm

1. Put the initial node on a list START
2. If START = GOAL or START = EMPTY, then terminate search
3. Assign the next node as START and call this node-A
4. If A = GOAL, terminate the search with success
5. Else-if, node has successor and generate all of them. Find out how far they are from the GOAL node.
6. Sort all the children generated so far by remaining distance from the goal. Name the list as START-1. Replace START = START-1
7. Go to step-2

Types of BFS

- Greedy Best First Search
- A* Search

Greedy Best First Search

- It tries to get as close as it can to the goal.
- It expands the node that appears to be closest to the goal
- It evaluates the node by using heuristic function only.
- Evaluation function f(n) = h(n)
  - heuristic= (estimate of cost from n to goal)
  - h(n) = 0 for goal state

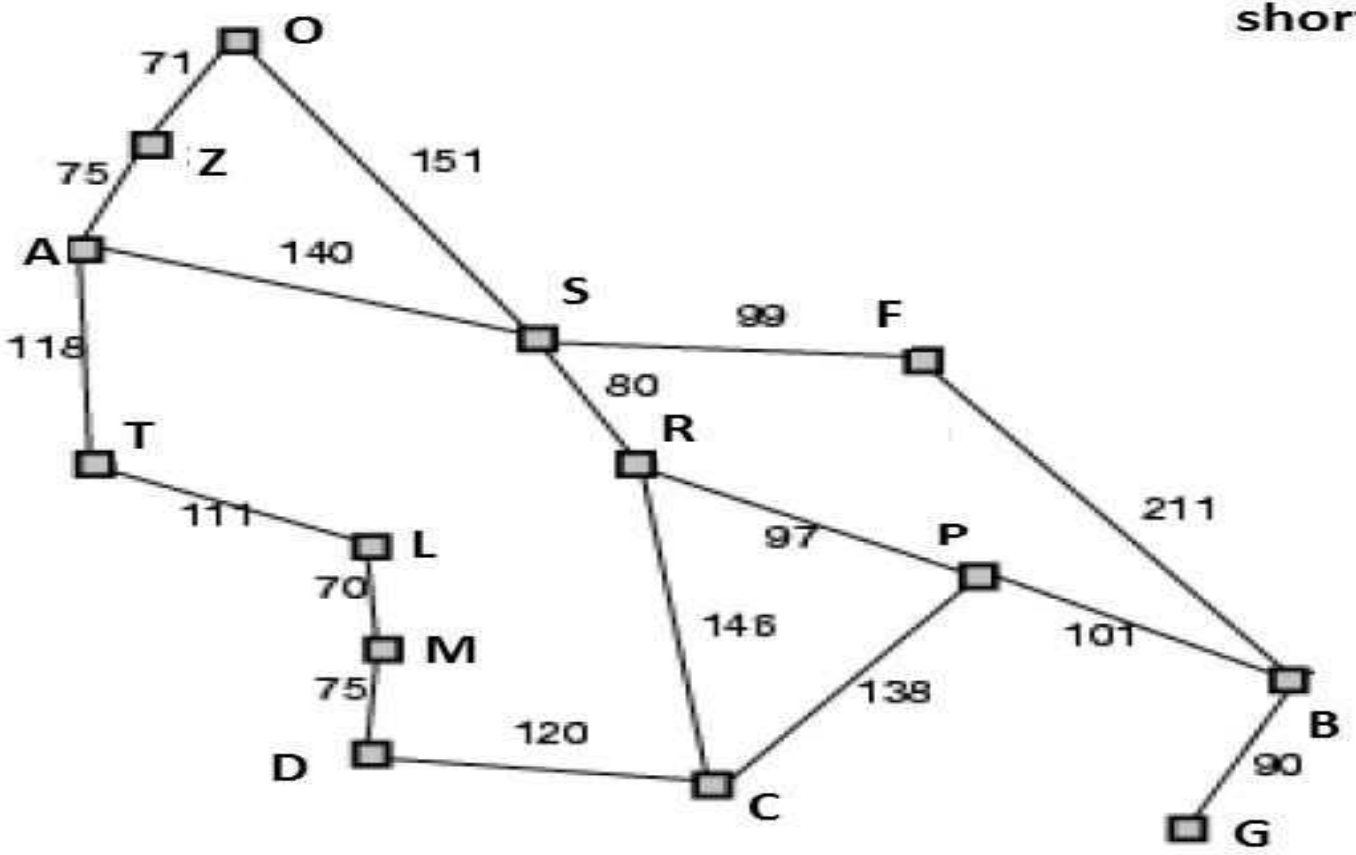
Properties

- ***Completeness:*** No – can get stuck in loops
- ***Time Complexity***: O (b^m), but a good heuristic can give dramatic improvement
- ***Space Complexity***: O (b^m), keeps all nodes in memory
- ***Optimality***: No

Applications:

This algorithm is used in Huffman encoding, minimum spanning tree, Dijkstra's algorithm etc.

Given following graph of cities, starting at City "A", problem is to reach to the "B"

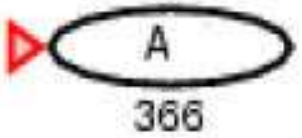


shortest line distance to B

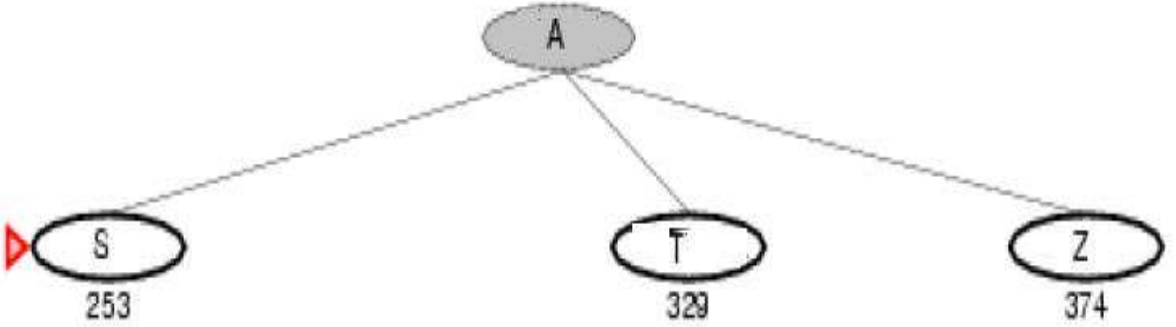
A	366
B	0
C	160
D	242
F	176
G	77
L	244
M	241
O	380
P	10
R	193
S	253
T	329
Z	374

Solution using greedy best first can be as below:

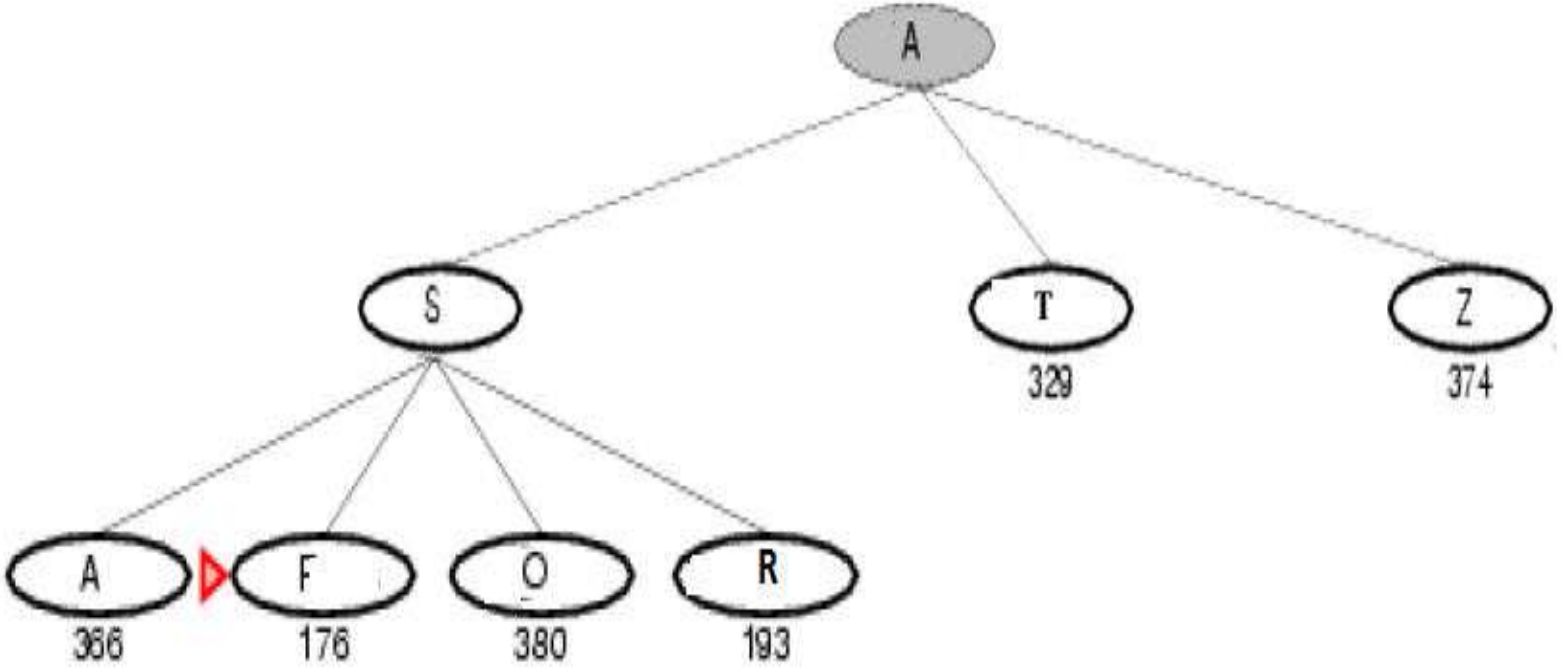
Step 1: Initial State



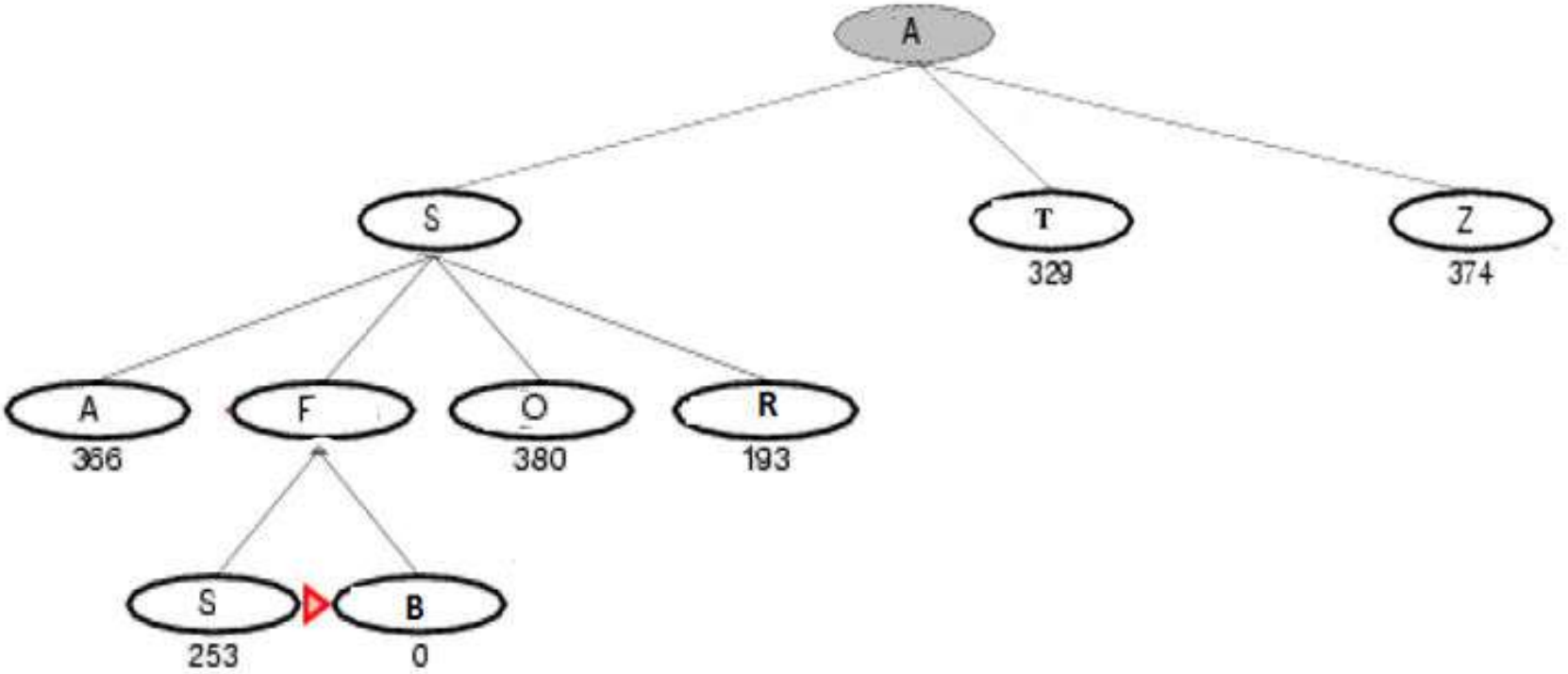
Step 2: After expanding A



Step 3: After expanding S



Step 3: After expanding F



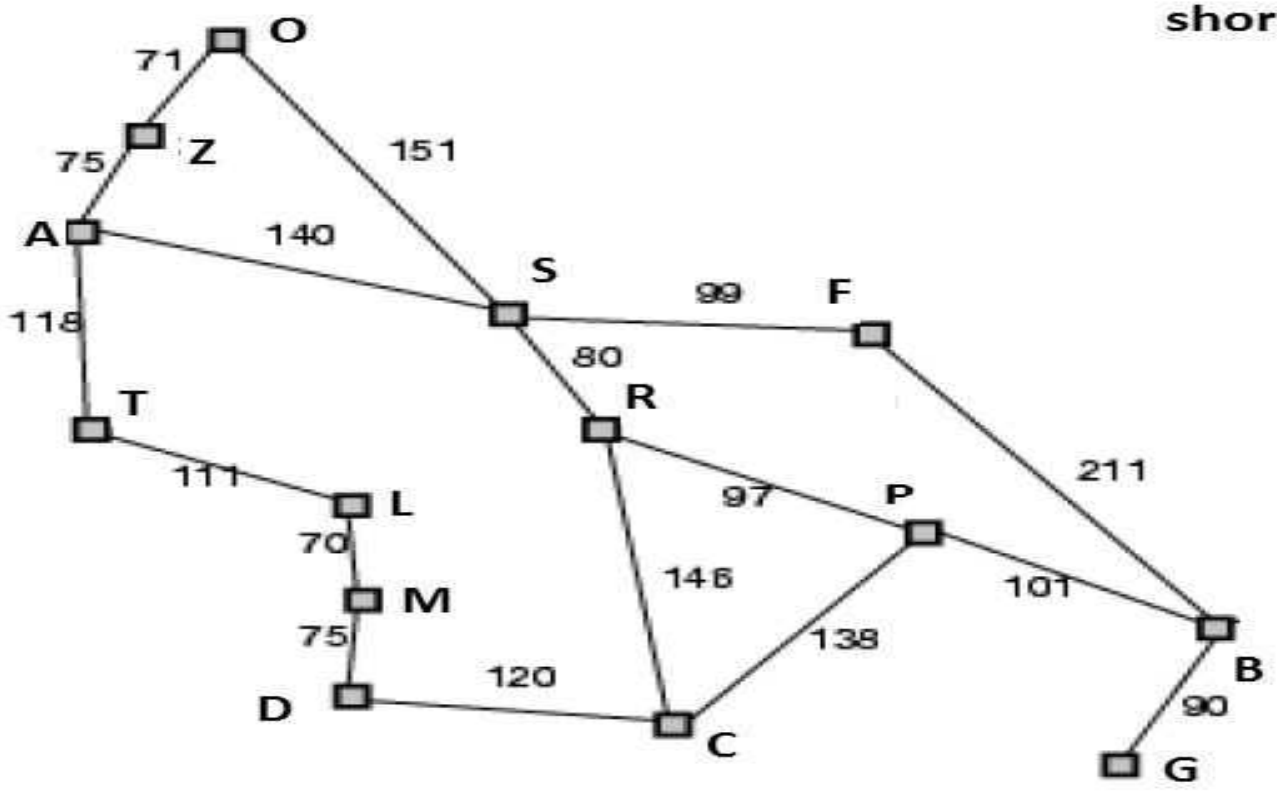
A* Search

- It finds a minimal cost-path joining the start node and a goal node for node n.
- Evaluation function: $f(n) = g(n) + h(n)$

  Where,
  - $g(n)$ = cost so far to reach n from root
  - $h(n)$ = estimated cost to goal from n

  Thus, $f(n)$ estimates always the lowest total cost of any solution path going through node n.
- Avoid expanding paths that are already expensive
- The main drawback of A* algorithm and indeed of any best-first search is its memory requirement.

A* search example (Find path from A to B)



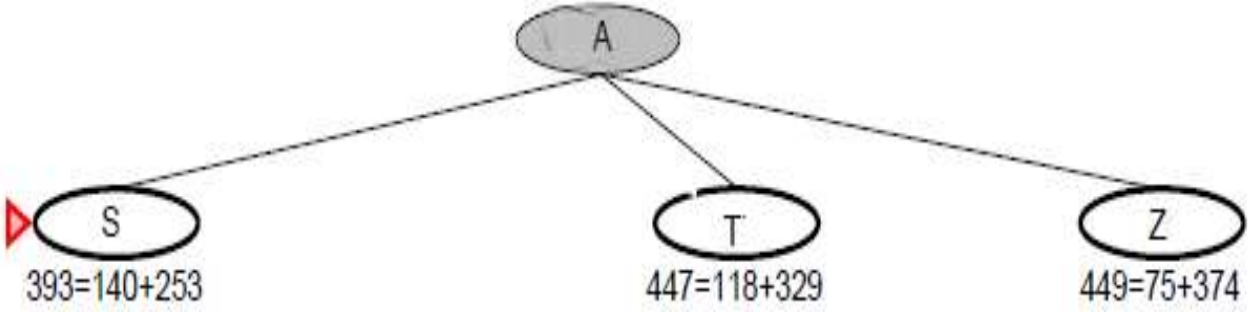
shortest line distance to B

A	366
B	0
C	160
D	242
F	176
G	77
L	244
M	241
O	380
P	10
R	193
S	253
T	329
Z	374

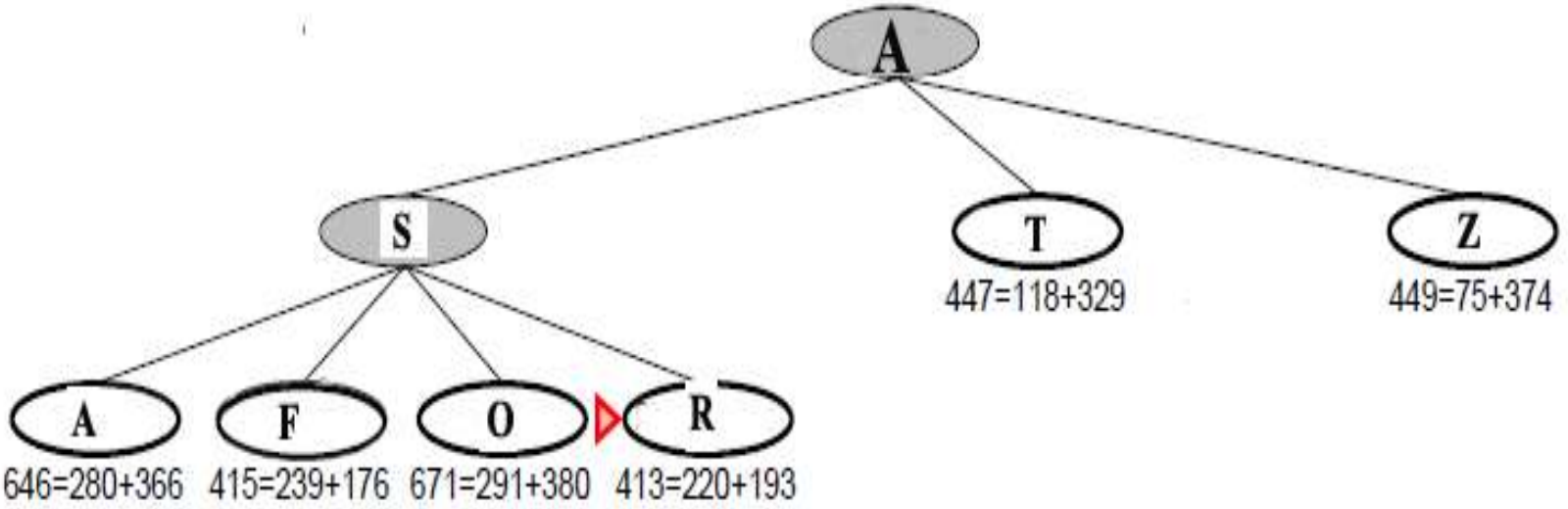
Here, evaluate nodes connected to source. Evaluate f(n)=g(n)+h(n) for each node. Select node with lowest f(n) value.

Step 1:

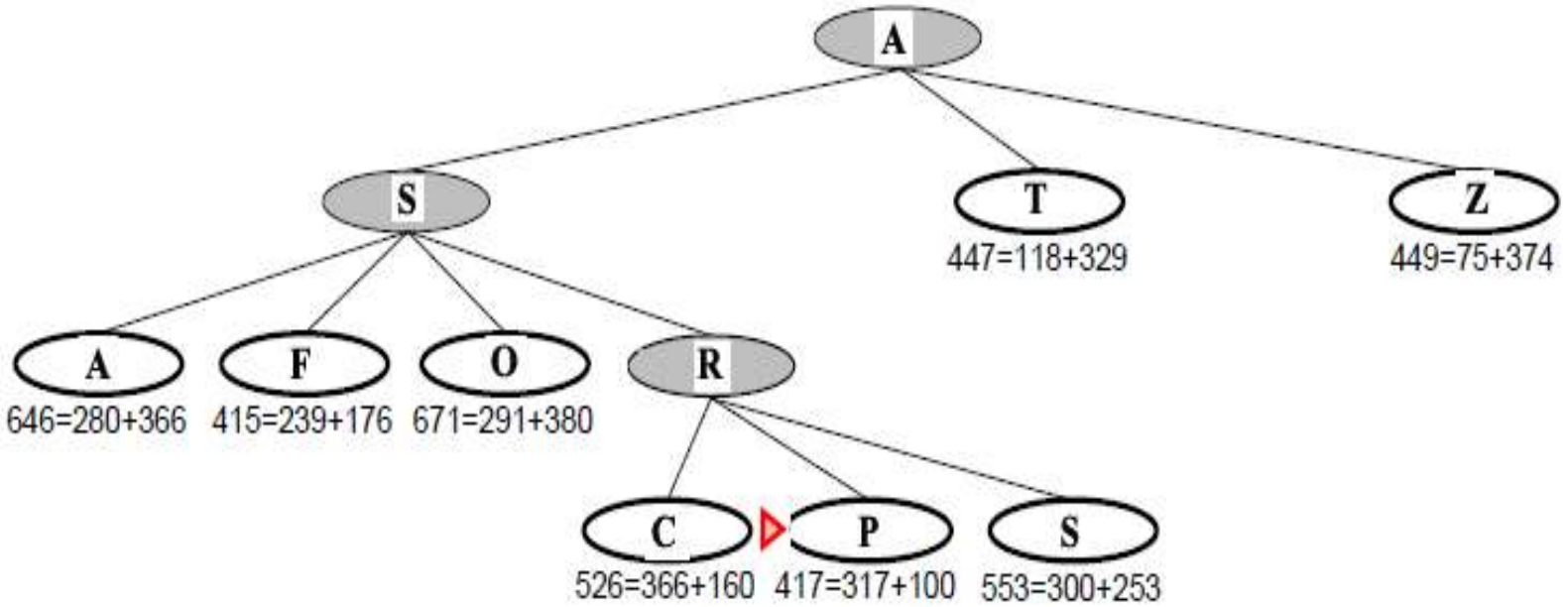
Step 2



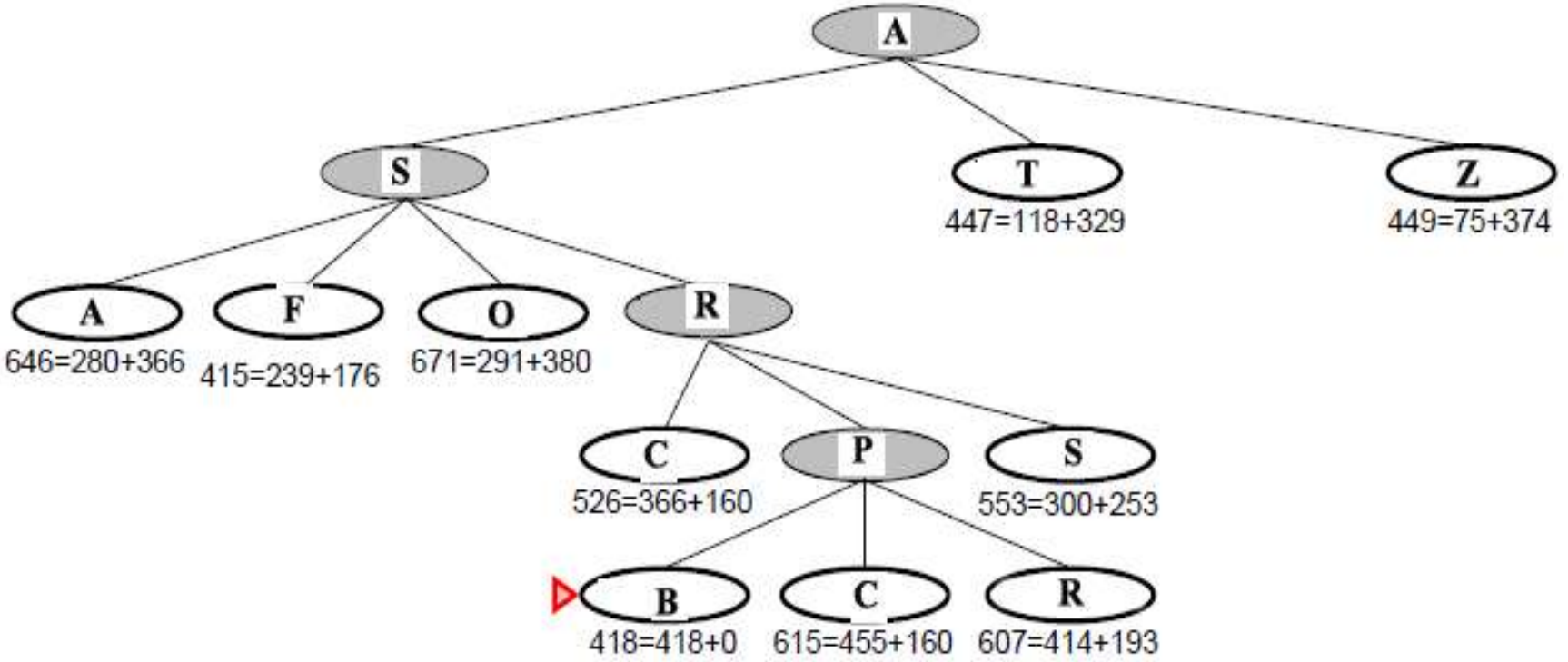
Step 3:


A
S
T
447=118+329
Z
449=75+374
A
646=280+366
F
415=239+176
O
671=291+380
R
413=220+193

Step 4:

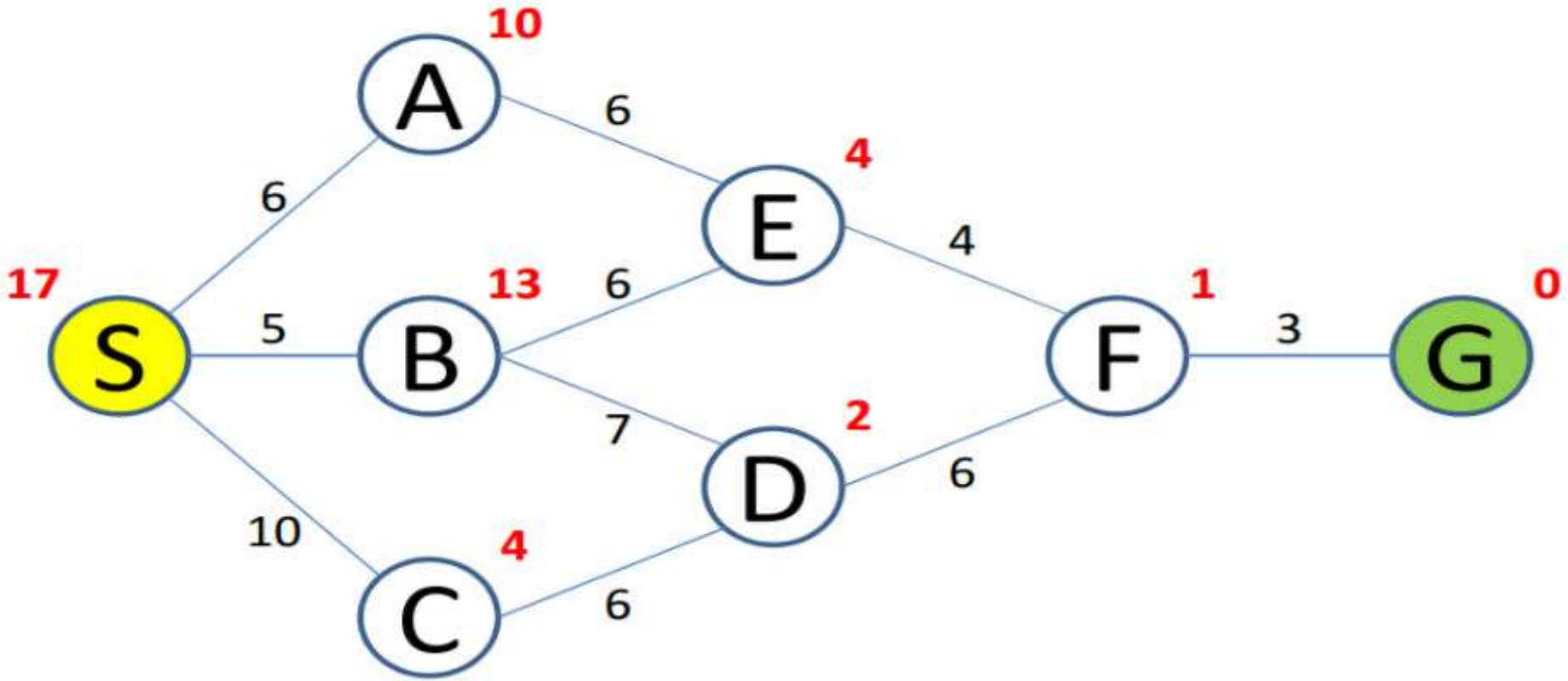

A
S
T
447=118+329
Z
449=75+374
A
646=280+366
F
415=239+176
O
671=291+380
R
C
526=366+160
P
417=317+100
S
553=300+253

Step 5:


A
S
T
447=118+329
Z
449=75+374
A
646=280+366
F
415=239+176
O
671=291+380
R
C
526=366+160
P
S
553=300+253
B
418=418+0
C
615=455+160
R
607=414+193

Assignment

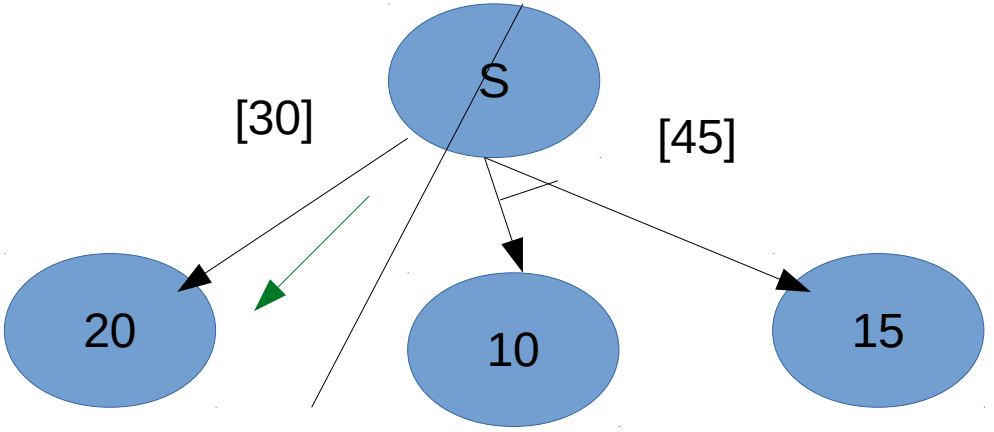
Perform A* Search



AO* Searching

- AO* Search is a type of heuristic search algorithm .
- AO* Search is used when problems can be divided into sub parts and which can be combined
- AO* in artificial intelligence is represented using AND OR graph or AND OR tree
- AO* have one or more and arc in it

Edge = 10



Hill Climbing Search

- Hill climbing is an extension of depth-first search which uses some knowledge such as estimates of the distance of each node from the goal to improve the search
- It is simply a loop that continually moves in the direction of increasing value—that is, uphill. It terminates when it reaches a “peak” where no neighbor has a higher value.
- Hill climbing is sometimes called greedy local search because it grabs a good neighbor state without thinking ahead about where to go next

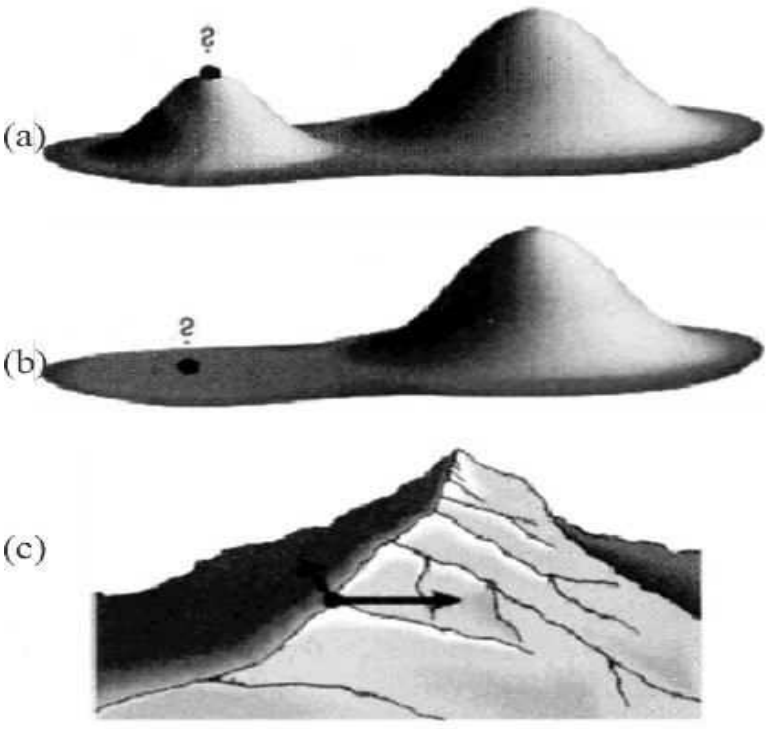


Figure 5.9 Local maxima, Plateaus and ridge situation for Hill Climbing

Hill Climbing Search

- Drawbacks
  - Local Maxima

    A local maximum is a peak that is higher than each of its neighboring states but lower than the global maximum.
  - Plateaus: A plateau is an area of the search space where evaluation function is flat, thus requiring random walk
  - Ridges: Where there are steep slopes and the search direction is not towards the top but towards the side

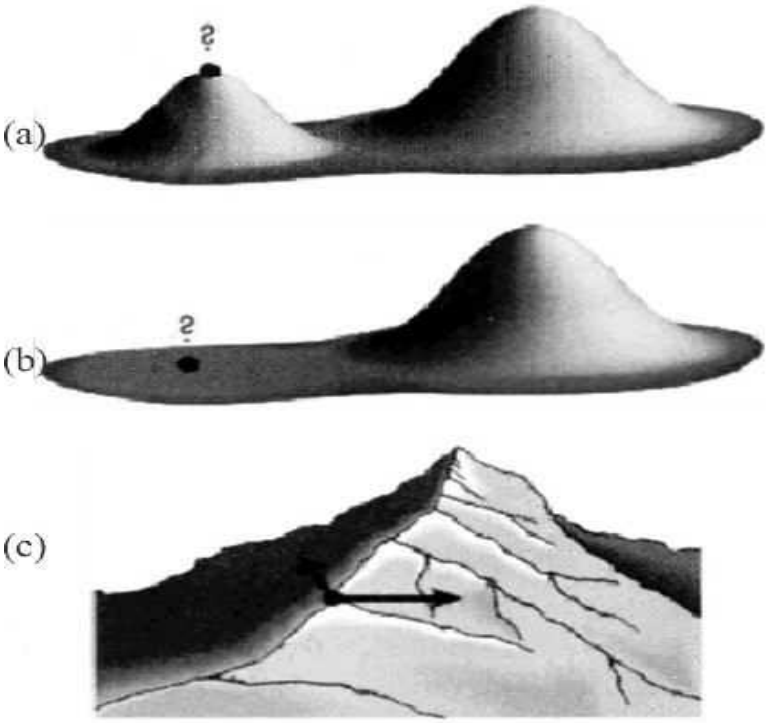


Figure 5.9 Local maxima, Plateaus and ridge situation for Hill Climbing

Hill Climbing Search

- **Remedies**
  - **Back tracking for local maximum:**
    The back tracking help in undoing what is been done so far and permit to try totally different part to attain the global peak.
  - **Big Jump:**
    A big jump is the solution to escape from plateaus because all neighbors' points have same value using the greedy approach
  - **Random restart:**
    Keep restarting the search from random locations until a goal is found

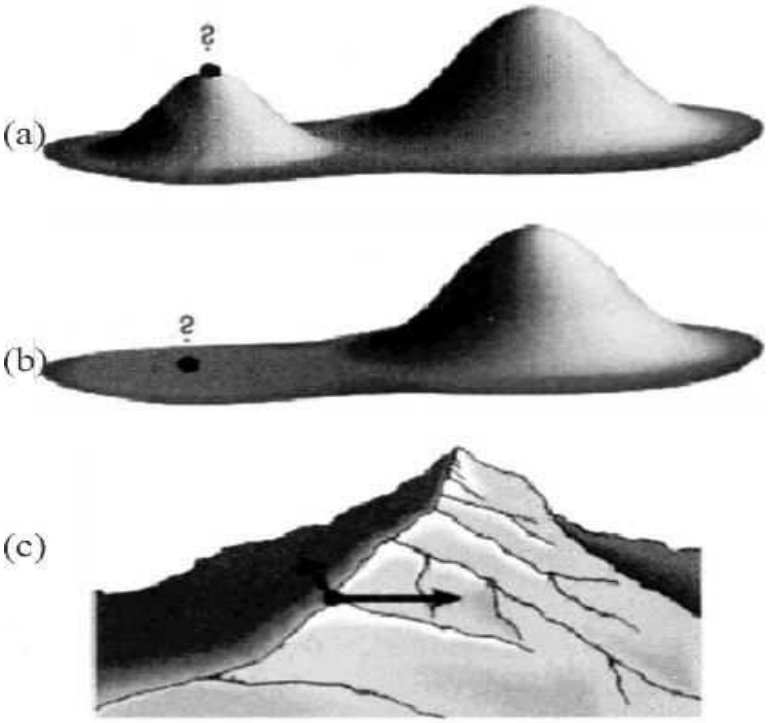
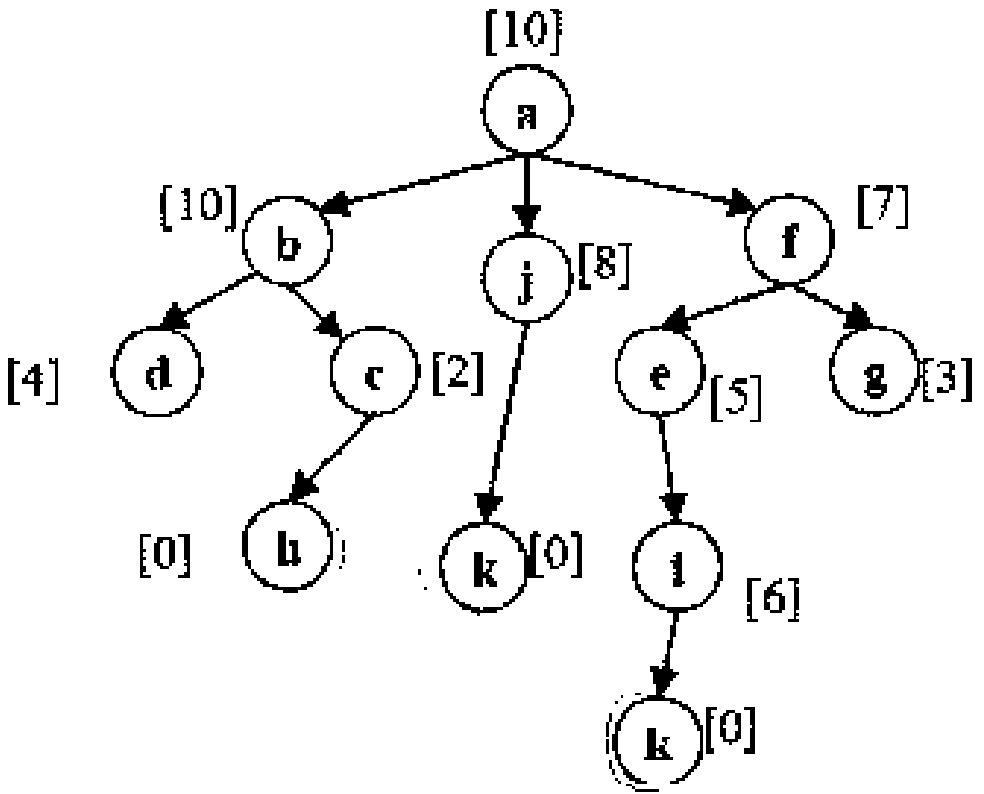


Figure 5.9 Local maxima, Plateaus and ridge situation for Hill Climbing

Why Hill Climbing is not Complete?

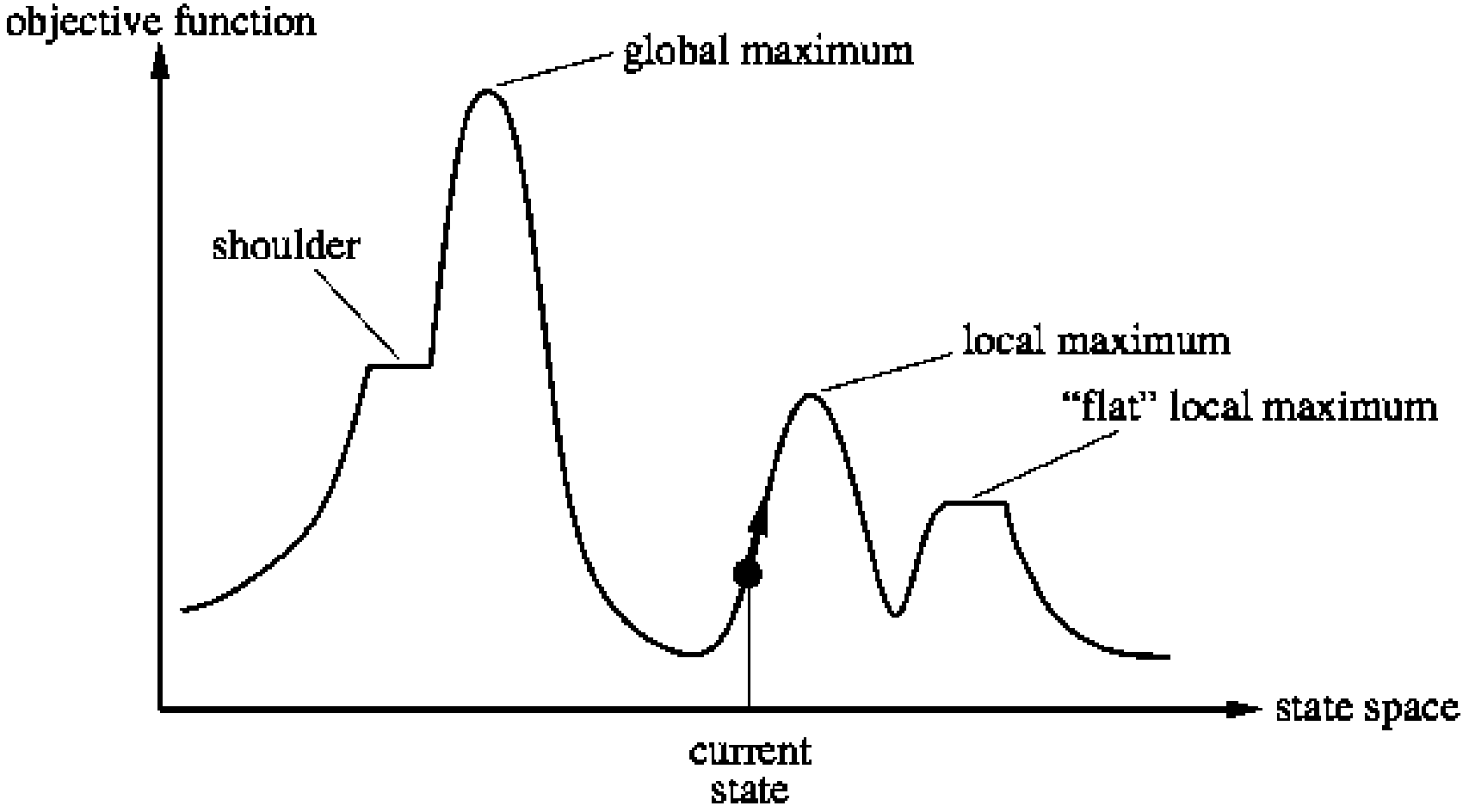
- Hill-climbing always attempts to make changes that improve the current state.
- The main problem that hill climbing can encounter is that of local maxima. This occurs when the algorithm stops making progress towards an optimal solution; mainly due to the lack of immediate improvement in adjacent states.

Problems in Hill Climbing



Here, "a" is initial and h and k are final states

- We start a-> f-> g and then what ??finish(without result)
- A common way to avoid getting stuck in local maxima with Hill Climbing is to use random restarts. In your example if G is a local maxima, the algorithm would stop there and then pick another random node to restart from. So if J or C were picked (or possibly A, B, or D), this would find the global maxima in H or K


objective function
global maximum
shoulder
local maximum
"flat" local maximum
state space
current
state

Simulated Annealing

- Simulated Annealing escapes local maxima by allowing some "bad" moves but gradually decrease their frequency.
- Instead of restarting from a random point, we allow the search to take some downhill steps to try to escape local maxima

Means Ends Analysis

- Means ends analysis is the problem solving techniques used commonly in Artificial intelligence for limiting search in AI program.
- The MEA technique as a problem-solving strategy was first introduced in 1961 by Allen Newell and Herbert A. Simon in their computer problem-solving program General Problem Solver (GPS).In that implementation, the correspondence between differences and actions, also called operators, is provided a priori as knowledge in the system.

- How MEA works?
  The MEA technique is a strategy to control search in problem-solving. Given a current state and a goal state, an action is chosen which will reduce the difference between the two. The action is performed on the current state to produce a new state, and the process is recursively applied to this new state and the goal state. Note that, in order for MEA to be effective, the goal-seeking system must have a means of associating to any kind of detectable difference those actions that are relevant to reducing that difference. It must also have means for detecting the progress it is making (the changes in the differences between the actual and the desired state), as some attempted sequences of actions may fail and, hence, some alternate sequences may be tried. When knowledge is available concerning the importance of differences, the most important difference is selected first to further improve the average performance of MEA over other brute-force search strategies. However, even without the ordering of differences according to importance, MEA improves over other search heuristics (again in the average case) by focusing the problem solving on the actual differences between the current state and that of the goal

Forward chaining

- Forward chaining is one of the two main methods of reasoning when using inference rules
- An inference engine, using forward chaining, searches the inference rules until it finds one where the antecedent (If clause) is known to be true. When such a rule is found, the engine can conclude, or infer, the consequent (Then clause), resulting in the addition of new information to its data.
- Forward chaining is a popular implementation strategy for expert systems, business and production rule systems.
- Forward chaining starts with the available data and uses inference rules to extract more data until a goal is reached.

Idea: fire any rule whose premises are satisfied in the *KB*,

- add its conclusion to the *KB*, until query is found

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

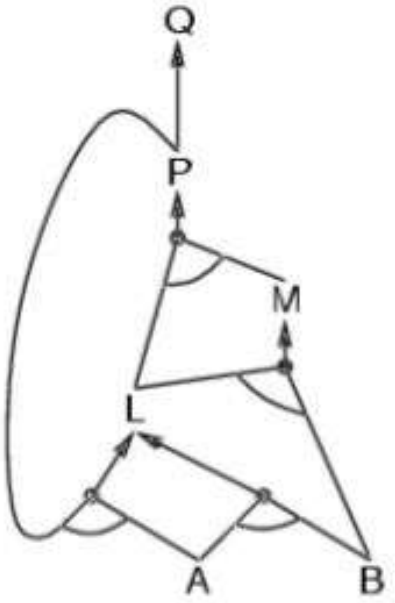
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

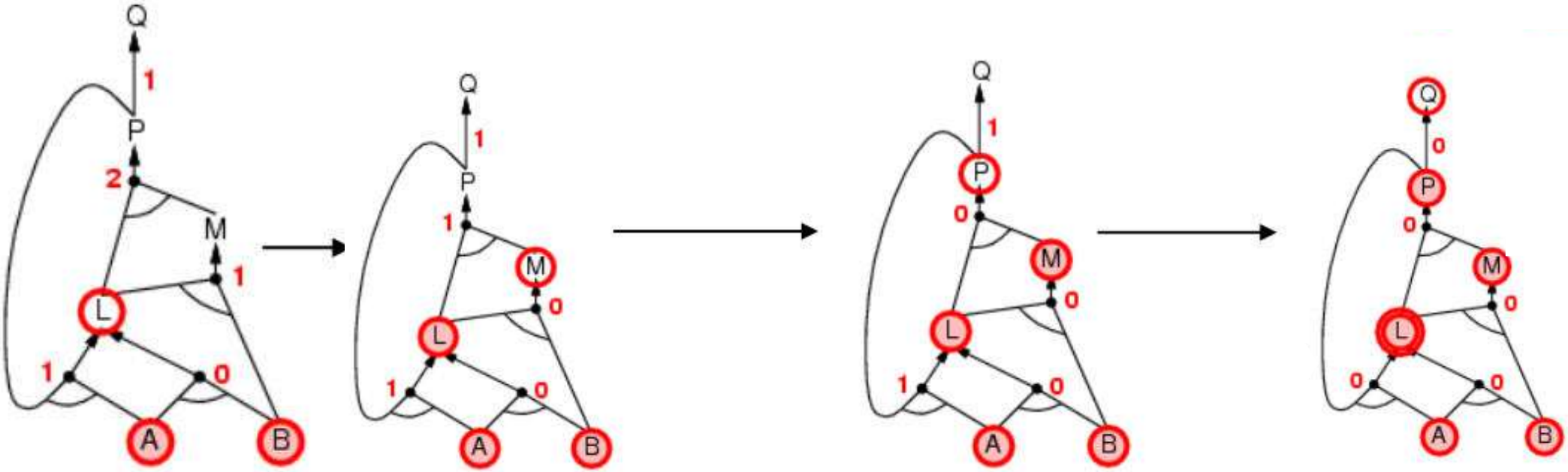
$A \wedge B \Rightarrow L$

A

B



Prove that Q can be inferred from above KB


Q
1
P
2
M
1
L
1
0
A
B
Q
1
P
1
M
0
L
1
0
A
B
Q
1
P
0
M
0
L
1
0
A
B
Q
0
P
0
M
0
L
0
0
A
B

Backward chaining

- Backward chaining (or backward reasoning) is an inference method that can be described as working backward from the goal(s).
- In game theory, its application to sub games in order to find a solution to the game is called backward induction.

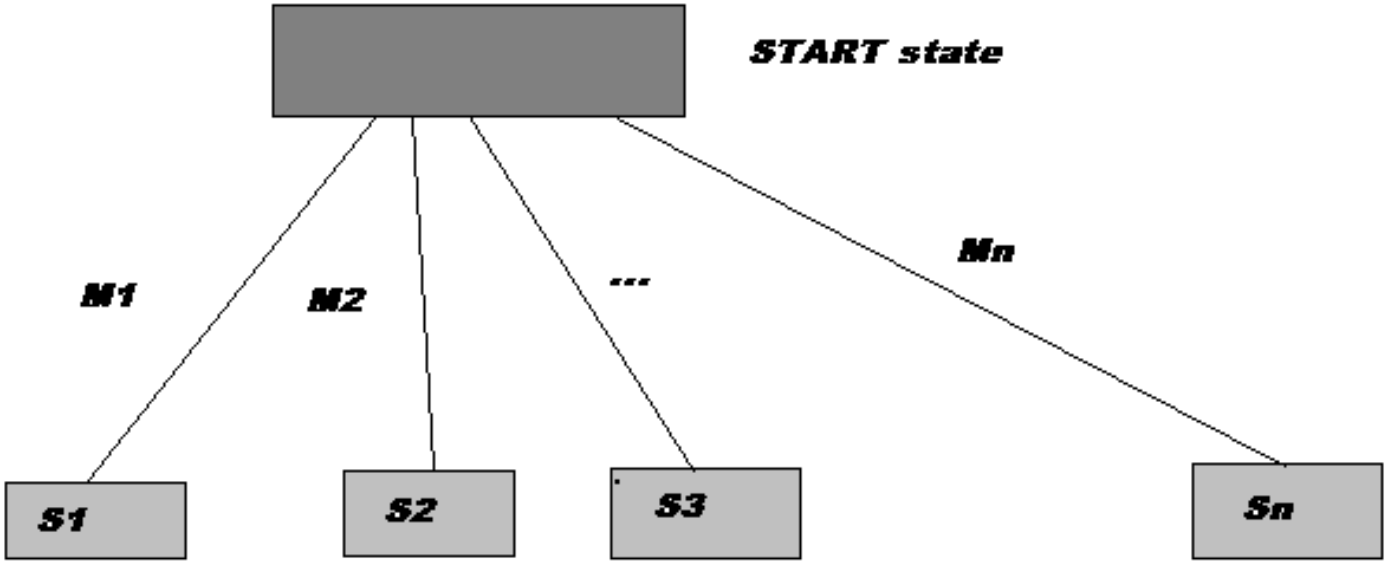
Game Playing

A game can be formally defined as a kind of search problem as below:

- **Initial state**: It includes the board position and identifies the player's to move.
- **Successor function**: It gives a list of (move, state) pairs each indicating a legal move and resulting state.
- **Terminal test**: This determines when the game is over. States where the game is ended are called terminal states.
- **Utility function**: It gives numerical value of terminal states. E.g. win (+1), loose (-1) and draw (0).

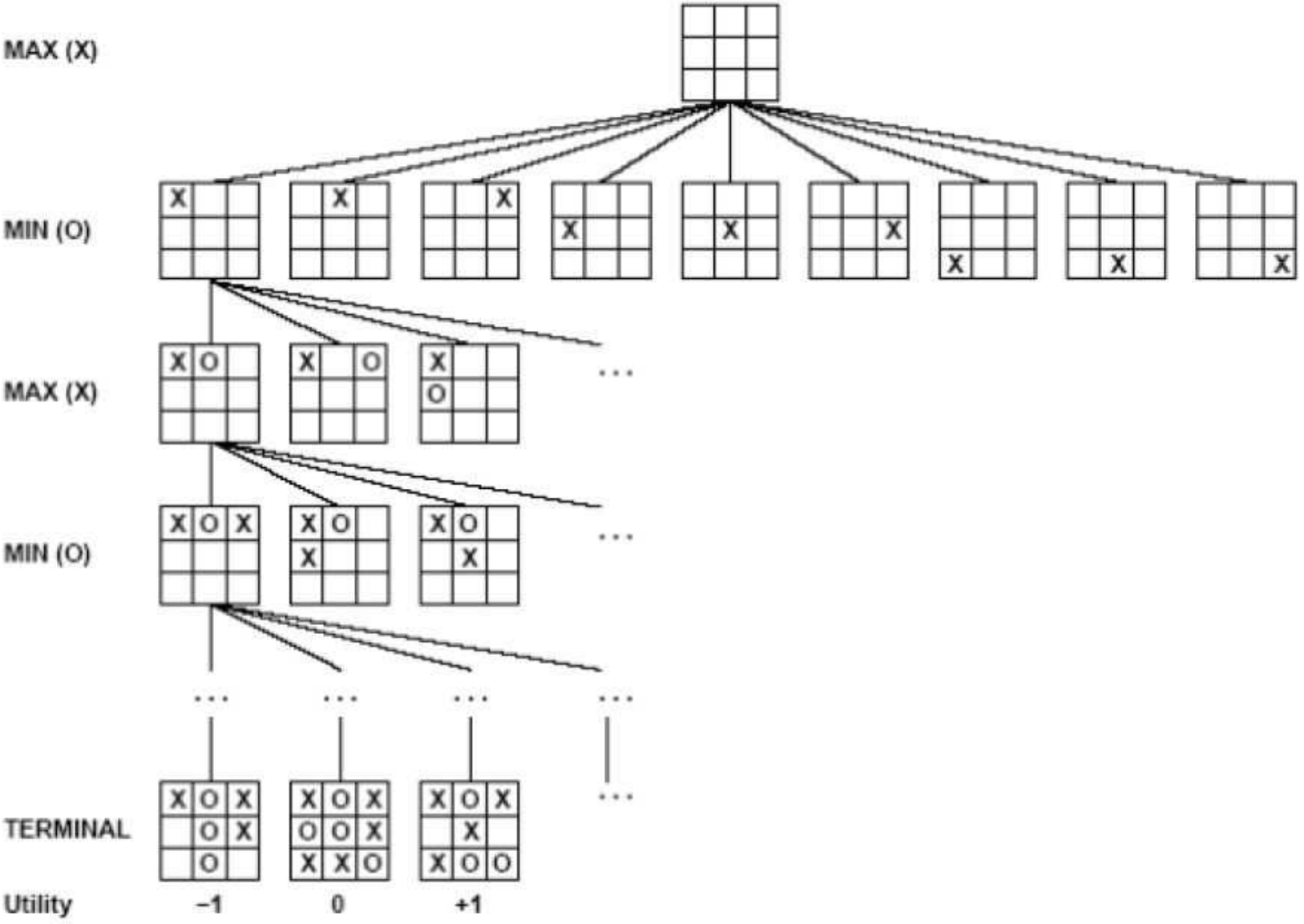
Game Trees

- We can represent all possible games(of a given type) by a directed graph often called a game tree.
- The nodes of the graph represent the states of the game. The arcs of the graph represent possible moves by the players (+ and -)



Example: Tic-tac-toe

There are two players denoted by X and O. They are alternatively writing their letter in one of the 9 cells of a 3 by 3 board. The winner is the one who succeeds in writing three letters in line.
The game begins with an empty board. It ends in a win for one player and a loss for the other, or possibly in a draw.
A complete tree is a representation of all the possible plays of the game. The root node is the initial state, in which it is the first player's turn to move (the player X).
The successors of the initial state are the states the player can reach in one move, their successors are the states resulting from the other player's possible replies, and so on.


MAX (X)
MIN (O)
MAX (X)
MIN (O)
TERMINAL
Utility
−1
0
+1

Assignment

Write about:

- Mini Max algorithm
- Alpha Beta pruning

Constraint Satisfaction Problems

- The process of finding a solution to set of variables Vi (V1, V2, …, Vn) and a set of constraint Ci (C1, C2, …, Cm)
- There is no any specified rule to define the procedure to solve the CSP.
- Eg: **8-queens problem**
- The CSP is a two-step process
  - ✗ Constraints are discovered and propagated as far as possible
  - ✗ There is still not found solution, then search begins

Crypto Arithmetic Problems

- Crypt-Arithmetic Problems are substitution problems where digits representing a mathematical operation are replaced by unique digits.
- Eg:

	A	B	C
+	D	E	F
	G	H	I

	0	2	4
+	5	8	9
	6	1	3

Solution:

1. $A \neq B \neq C \neq D \neq E \neq F \neq G \neq H \neq I$
2. $C + F = I$
   $C + F = 10 + I$ (I as carry)
3. $B + E = H$
   $B + E = 10 + H$
   $B + E + 1 = H$
   $B + E + 1 = 10 + H$ (H as carry)
4. $A + D = G$
   $A + D + 1 = G$

Step 1:

Domain of C = {1, 2, 3, 4, 5, 6, 7, 8, 9}
Domain of F = {1, 2, 3, 4, 5, 6, 7, 8, 9}
So,
Domain of I = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Select C = 4 & F = 9 Then I = 3 (Carry = 1)
So,

	A	B	4
+	D	E	9
	G	H	3

Step 2:

Domain of B = {1, 2, 5, 6, 7, 8}

Domain of E = {1, 2, 5, 6, 7, 8}

So,

Domain of H = {0, 1, 2, 5, 6, 7, 8}

Select B = 2 & E = 8

Then H = 10+1 {previous carry = 1 + (Carry = 1)}

So,

	A	2	4
+	D	8	9
	G	1	3

Step 3:

Domain of A = {0, 5, 6, 7}

Domain of D = {0, 5, 6, 7}

So, Domain of G = {5, 6}

Select A = 0 & D = 5 Then G = 6 (with addition of Carry)

So,

Hence, the required solutions are:

A = 0, B = 2, C = 4, D = 5, E = 8, F = 9, G = 6, H = 1, I = 3.

Assignment

Solve any 2 cryptoarithmetic problems.