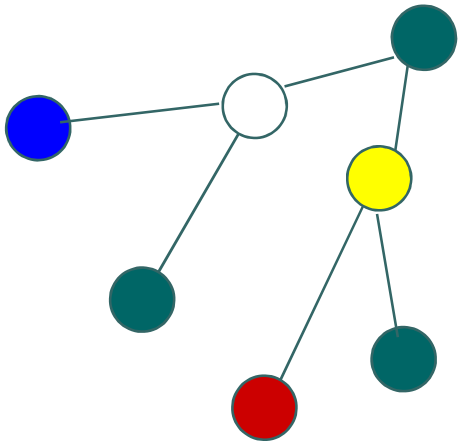
AI Searching Techniques

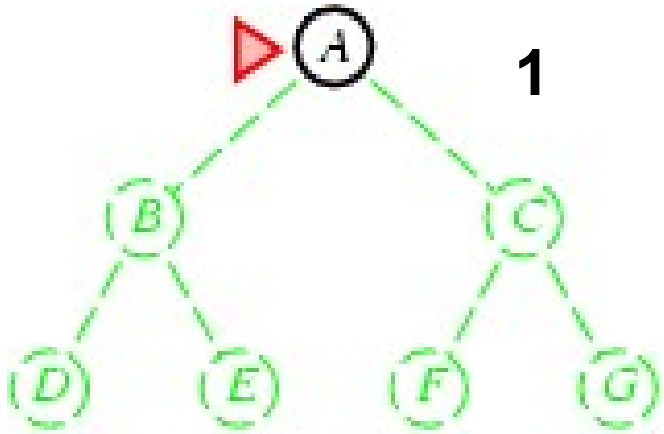


Dr. Ganesh Bhutkar

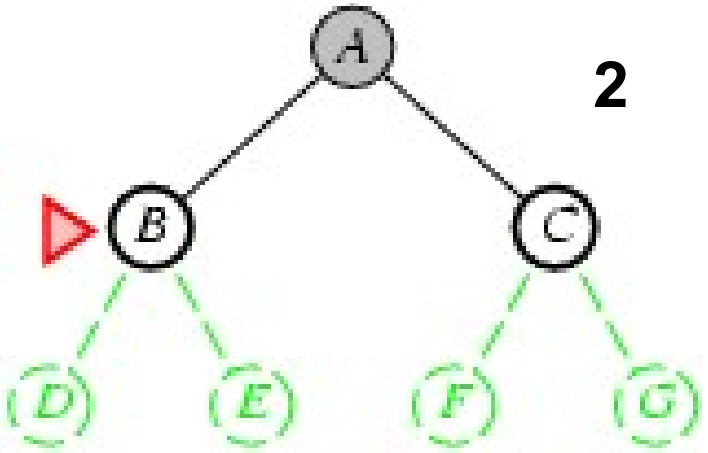
ganesh.bhutkar@vit.edu

TY - 2020-21

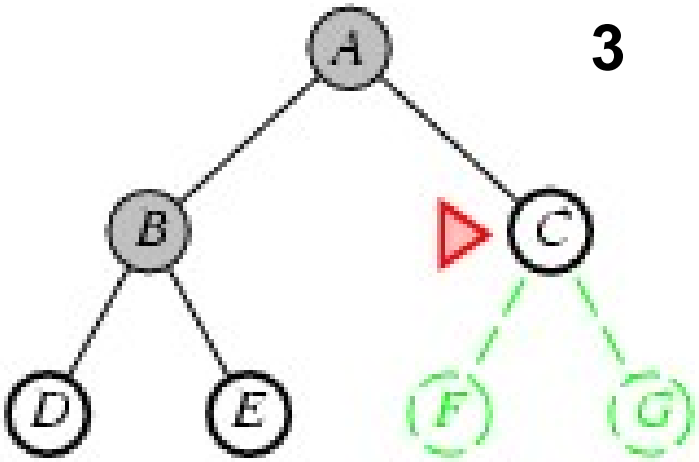
Blind Search : Breadth First Search (BFS)



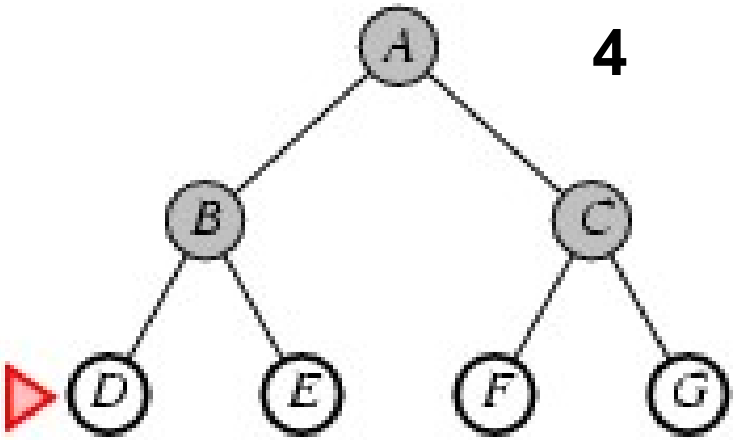
1



2

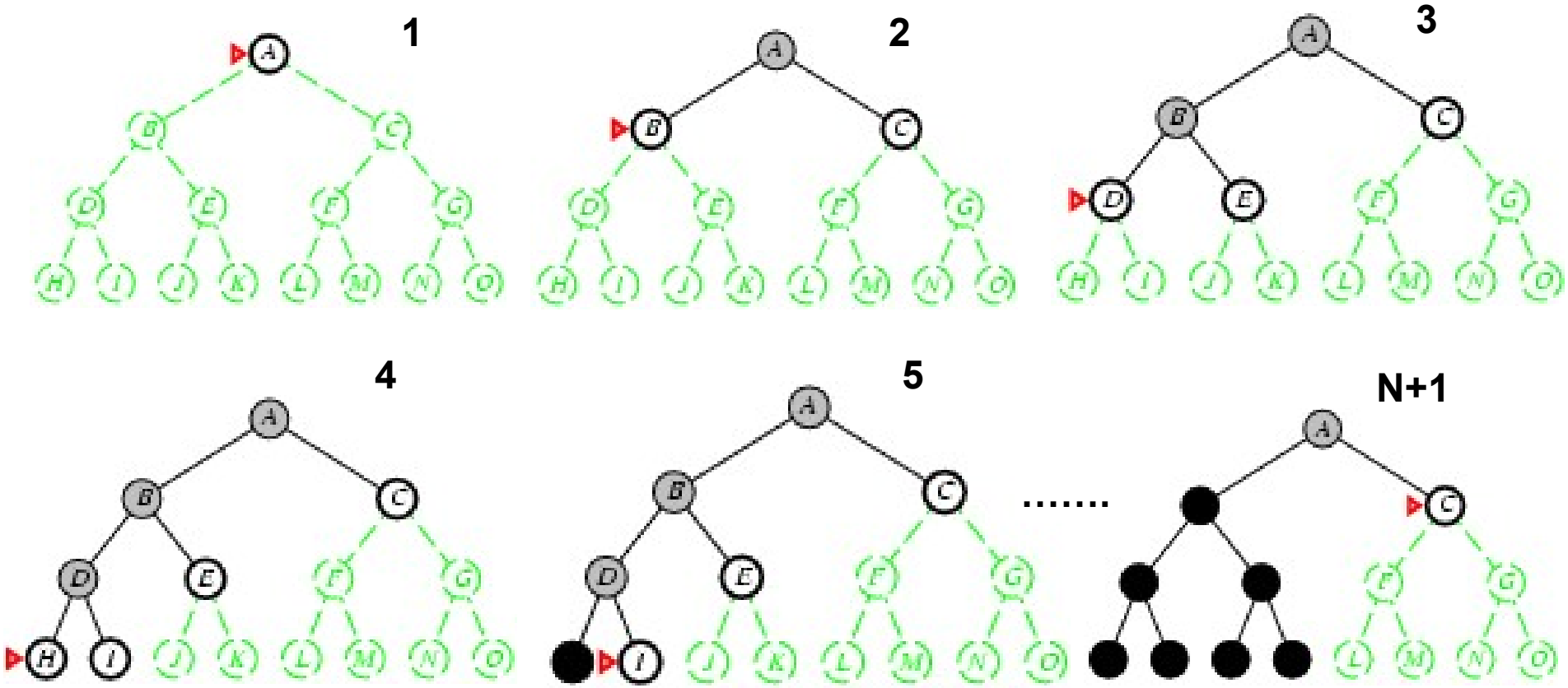


3

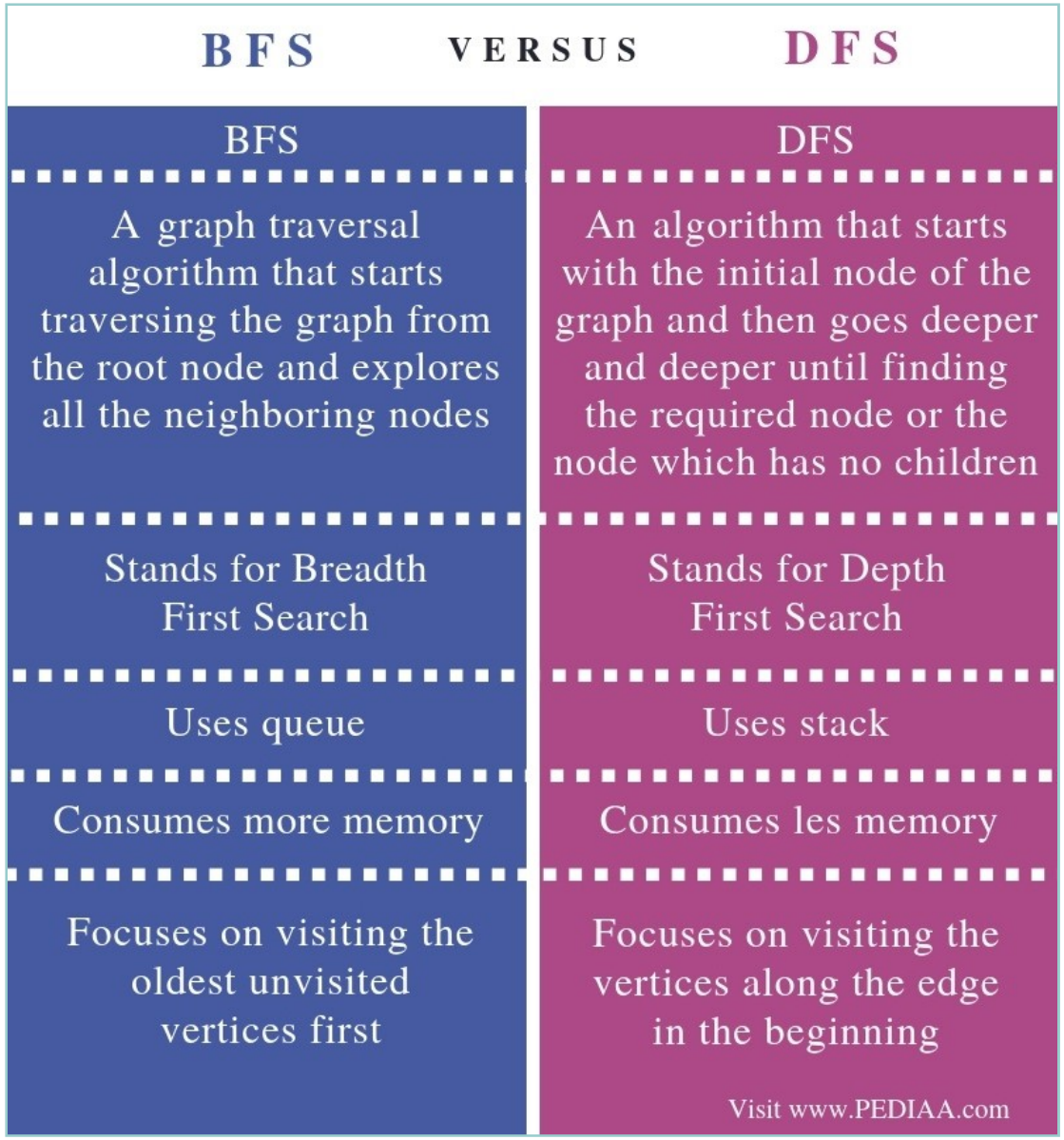


4

Blind Search : Depth First Search (DFS)



Difference between DFS & BFS



Difference between DFS & BFS

- BFS traverses level-wise; whereas DFS traverses depth-wise.
- No backtracking required in BFS.
- DFS may be trapped into Infinite loop – Blind Alley.

Problem as a State Space Search

- To build as system to solve a particular problem, we need:
  - **Define the problem**: must include precise specifications ~ initial solution & final solution.
  - **Analyze the problem**: select the most important features that can have an immense impact.
  - **Isolate and represent** : convert these important features into knowledge representation.
  - **Problem solving technique(s)**: choose the best technique and apply it to particular problem.

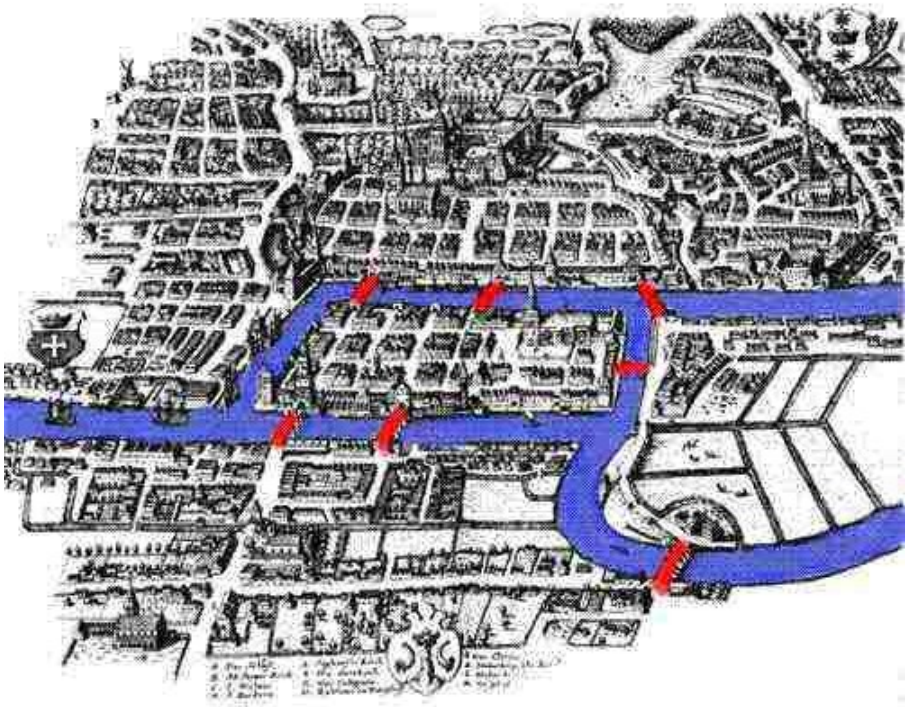
Important Questions…

- **Typical questions** that need to be answered:
  - Is the problem solver **guaranteed** to find a solution?
  - Will the system always terminate or caught in a **infinite loop**?
  - If the solution is found, it is **optimal**?
  - What is the **complexity** of searching process?
  - How it can effectively utilize the **representation** paradigm?

Important Terms

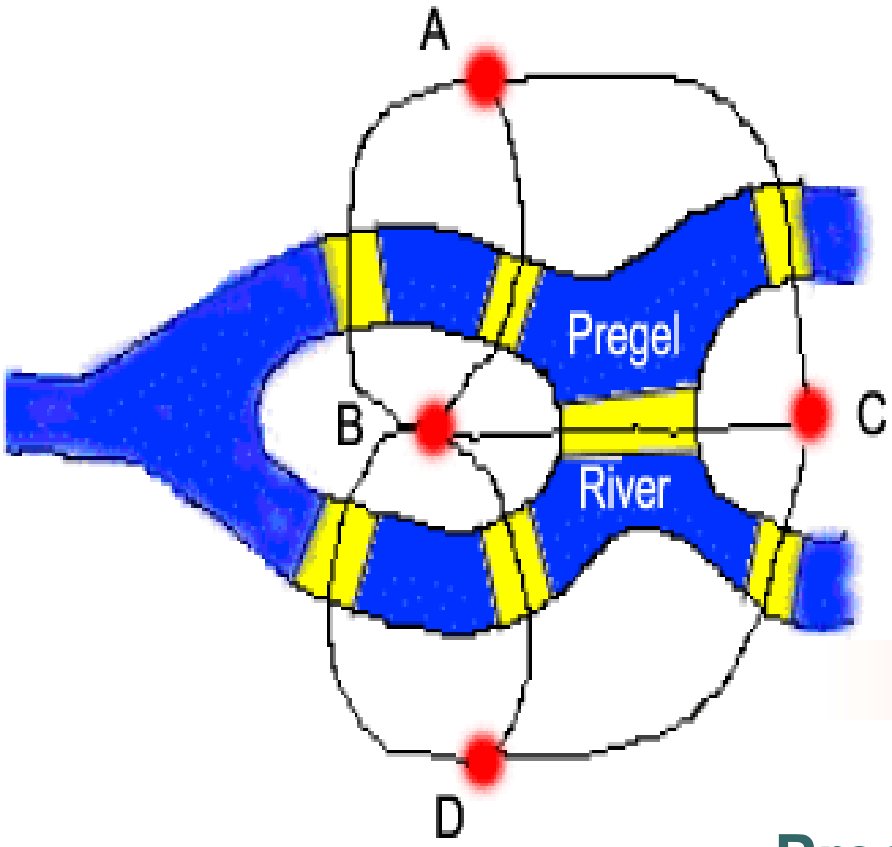
- **Search space** → possible states, conditions and solutions.
- **Initial state** → state where the searching process started.
- **Goal state** → the ultimate aim of searching process.
- **Problem space** → “what to solve”
- **Searching strategy** →strategy for controlling the search.
- **Search tree / graph** → tree / graph representation of search space, showing possible solutions from initial state.

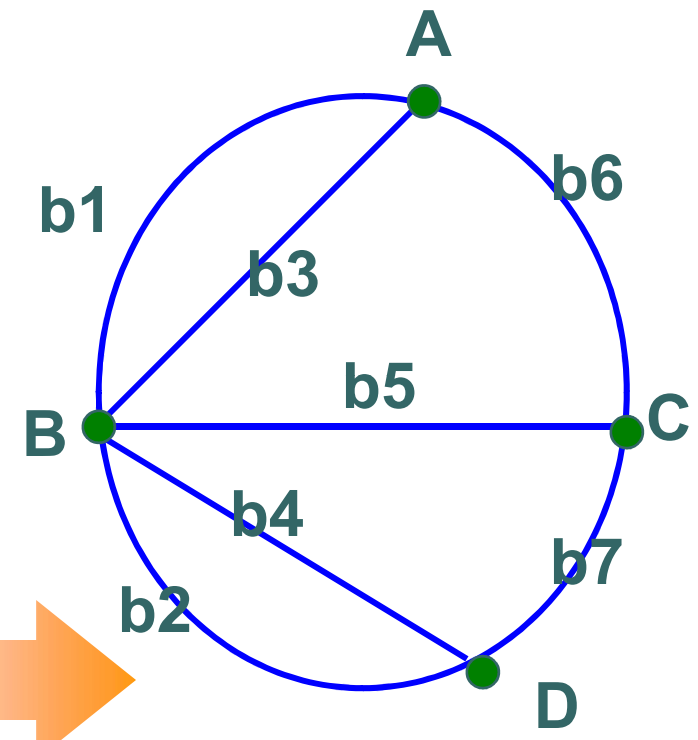
Example: The Bridges of Konigsberg Problem



- Classical graph application
- Introduced by **Leonhard Euler**
- Problem: **Can a person walk around the city crossing each bridge exactly once?**

Example: The Bridges of Konigsberg Problem (Cont…)





Predicates: Connect (B, C, b5)

Searching Strategies

- **Uninformed Search**

- **Blind search** → traversing the search space until the goal nodes is found (might be doing exhaustive search).

- *Techniques* : **Breadth First Depth first, Interactive Deepening search.**

- Higher Guarantee

- **Informed Search**

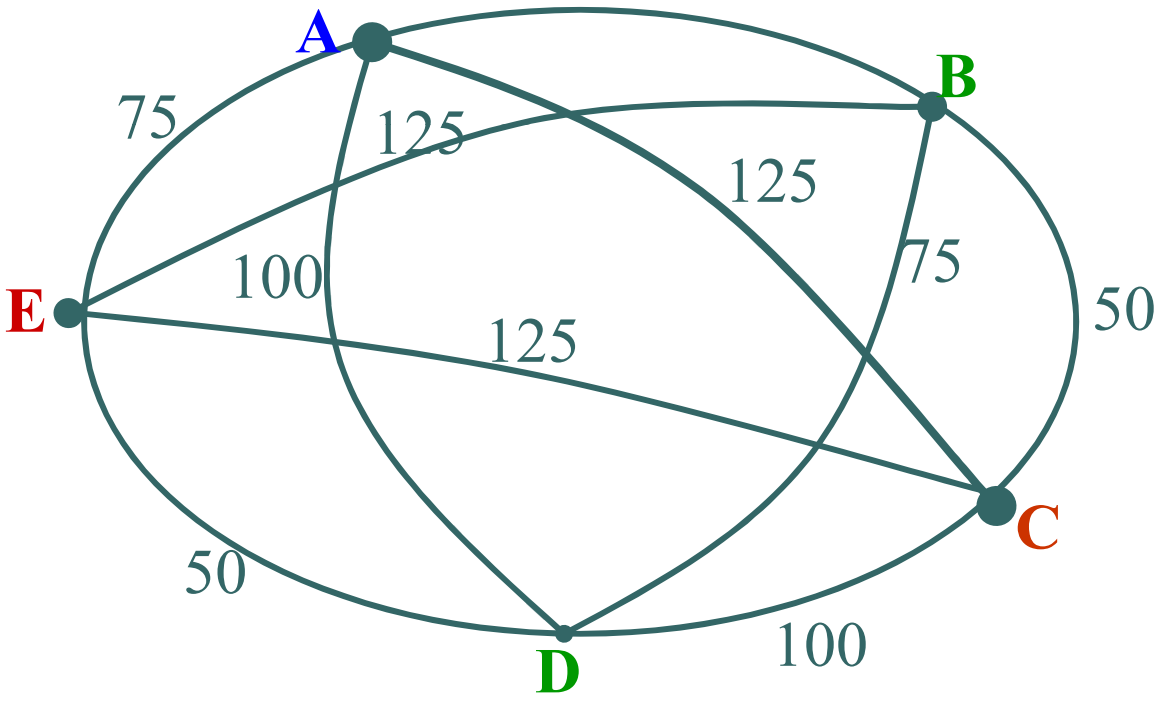
- **Heuristic search** → search process takes place by traversing search space with applied rules (information).

- T*echniques*: **Greedy Best First Search, A* Algo, AO* Algo.**

- There is no guarantee that solution is found.

Example: Traveling Salesman Problem

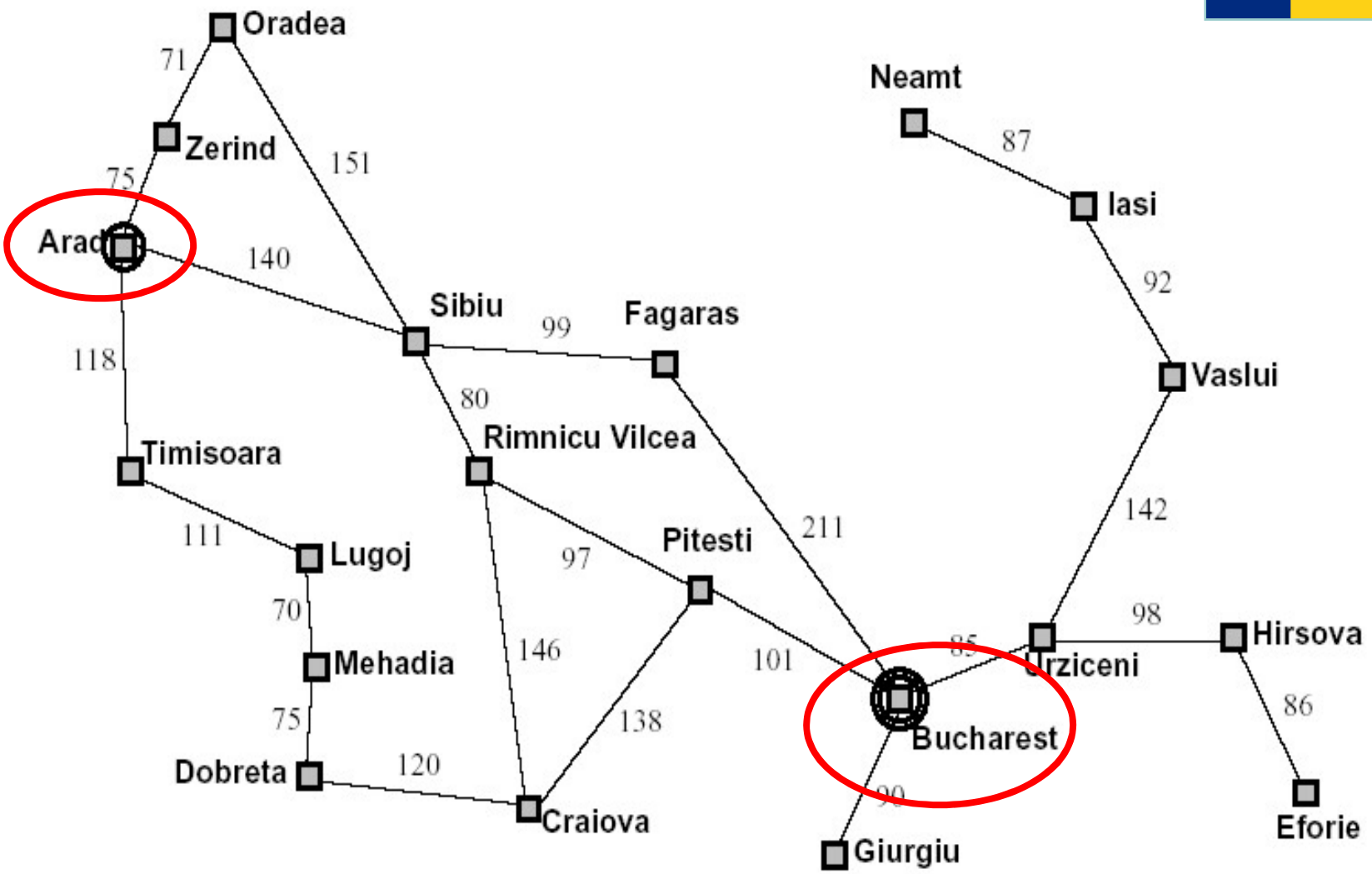
- Suppose, a salesman has five cities to visit and them must return home.
- **Goal** → Find the **Shortest Path** to travel.



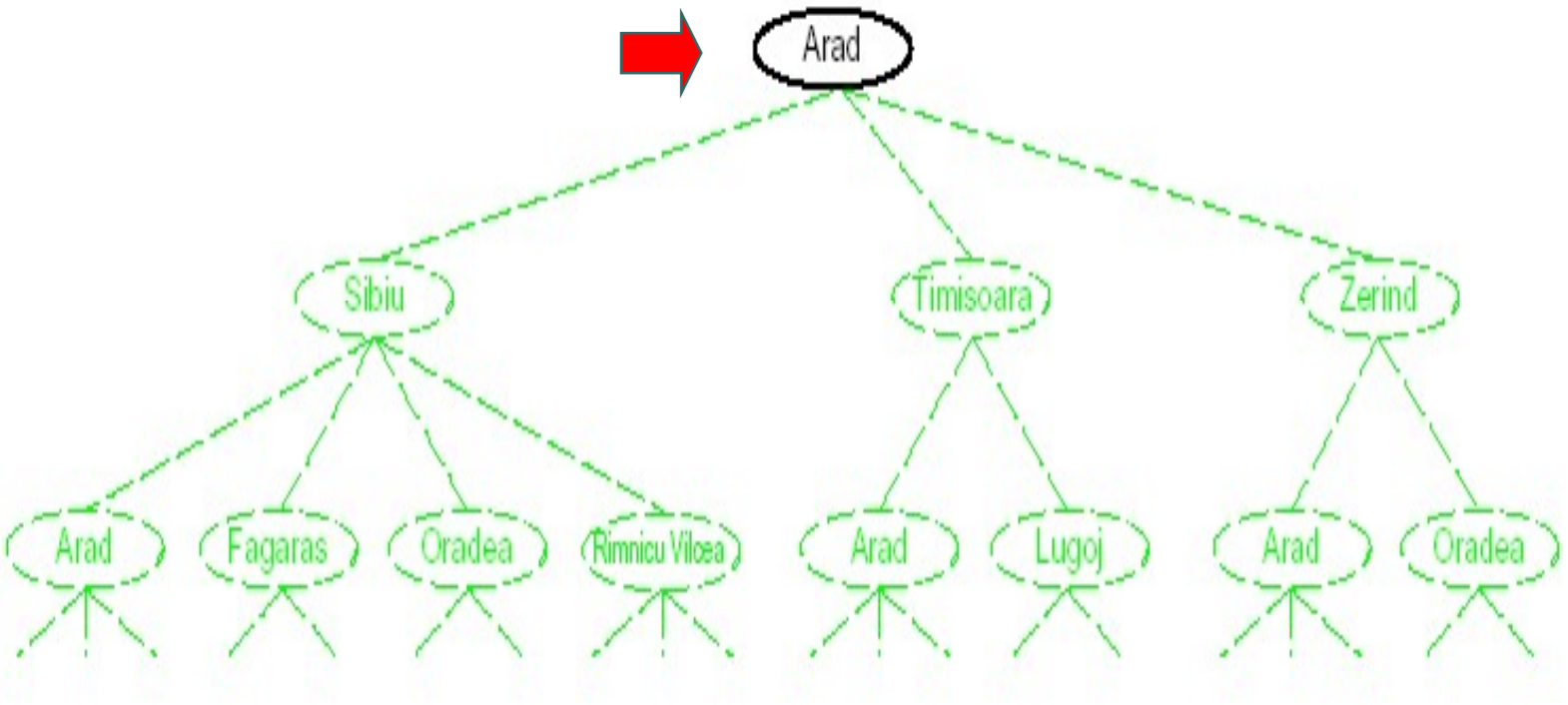
Searching for Solution

- Search through state space (explicitly using searching tree)
- **Node expansion :** Generating new node related to previous nodes
- **Concepts:**
  - **State :** Conditions in which the node corresponds
  - **Parent node :** The superior node
  - **Path cost :** The cost, from initial to goal state
  - **Depth:** Number of steps along the path from initial state

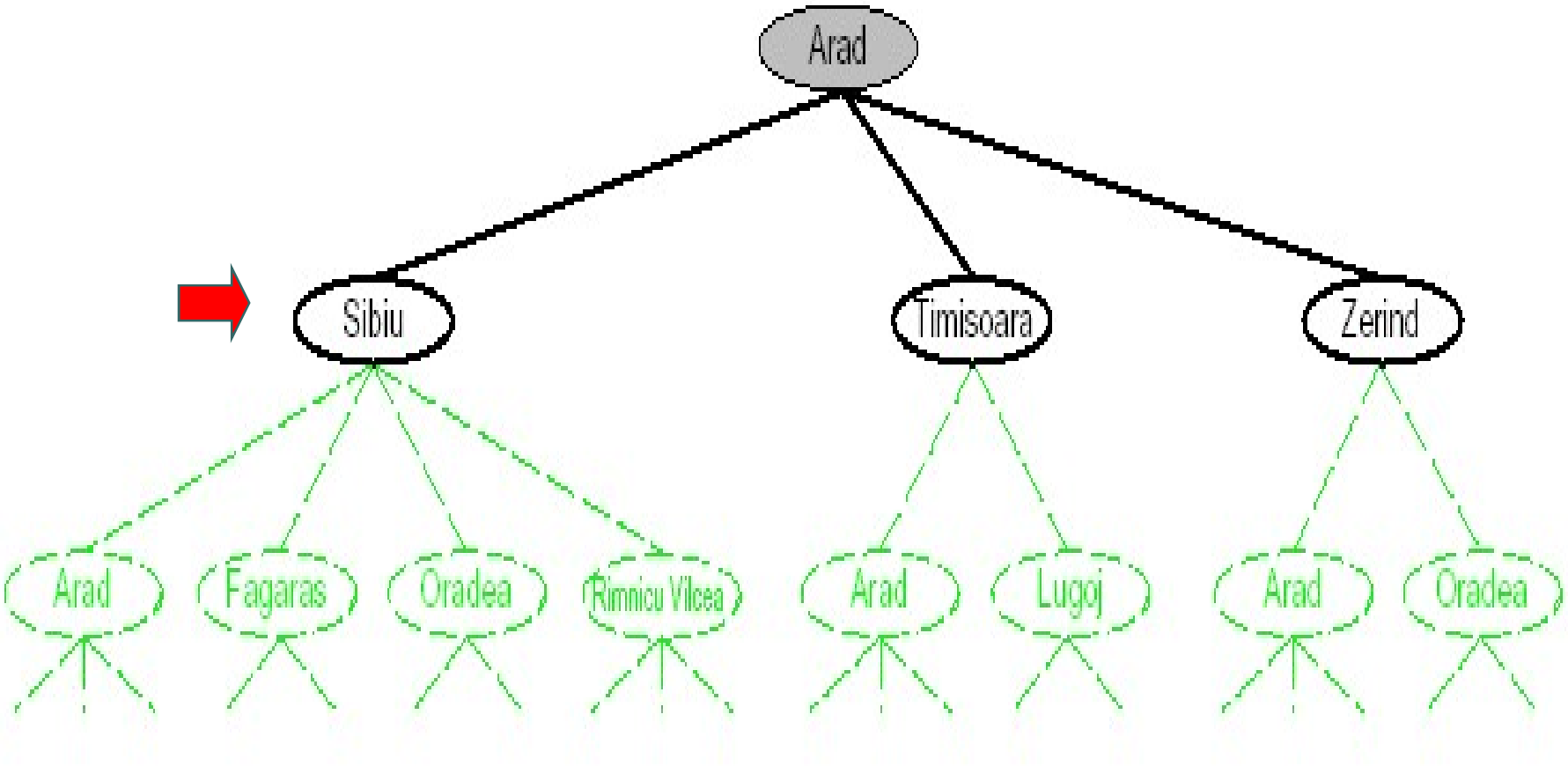
Node Expansion



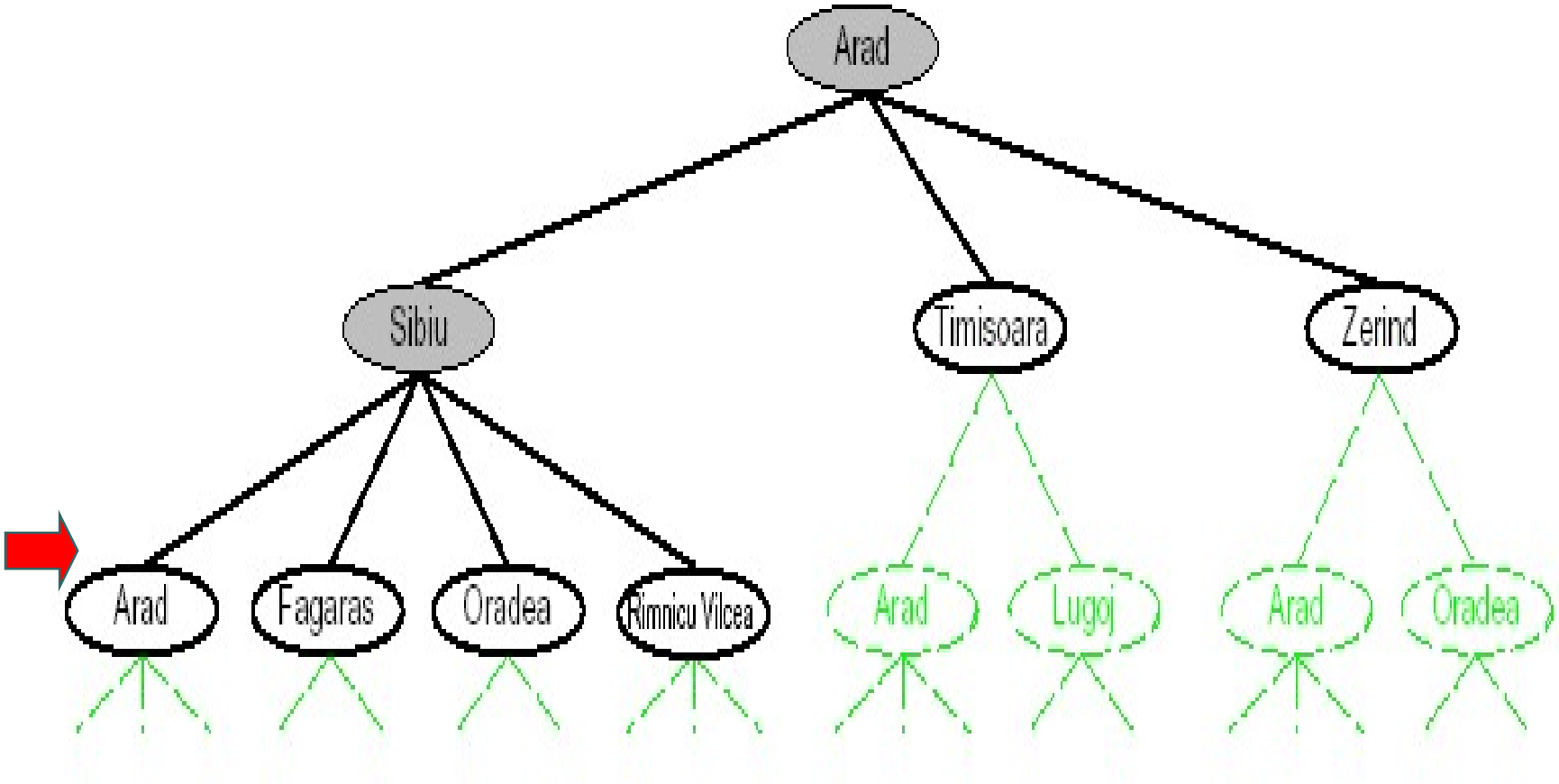
Node Expansion (initial)



Node Expansion (Expanding Arad)

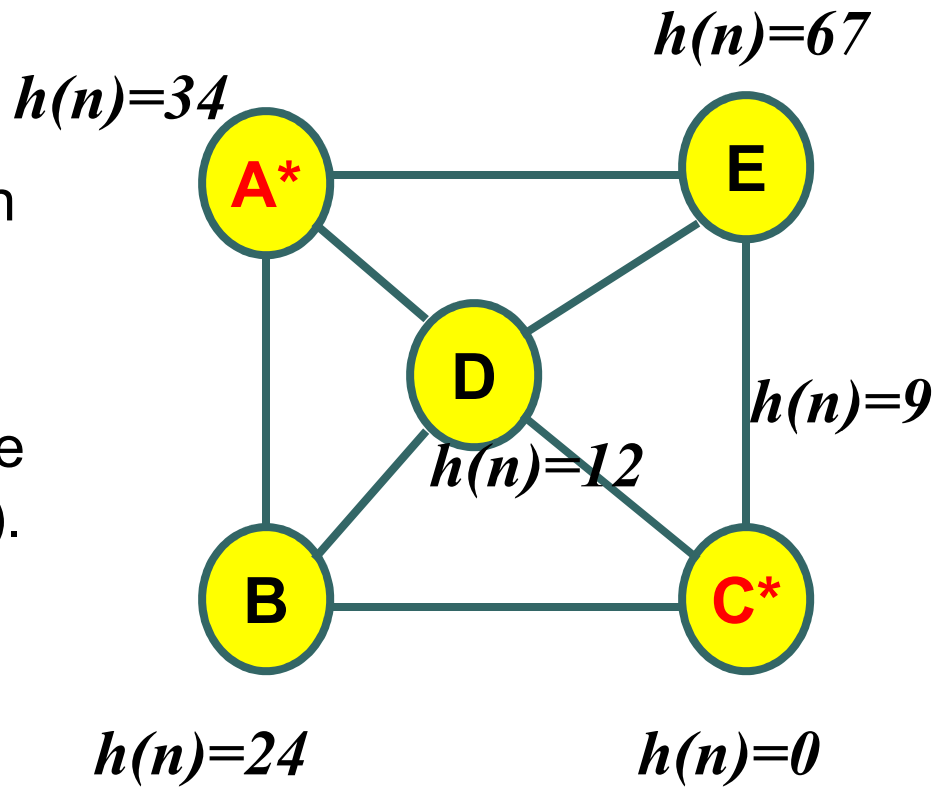


Node Expansion (Expanding Sibiu)



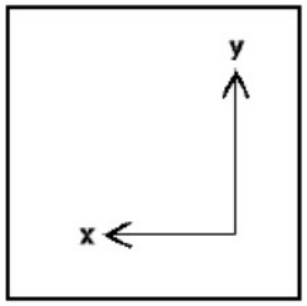
Heuristic Search :

- Important aspect: Formation of heuristic function *(h(n))*.
- Heuristic function → Additional knowledge to guide searching strategy (short cut).
- Distance: Heuristic function can be straight line distance (SLD)
- Eucledian & Manhattan Distance

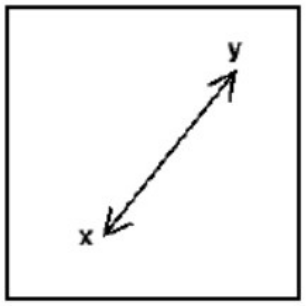


Eucledian & Manhattan Distances

EUCLIDEAN VS. MANHATTAN DISTANCE



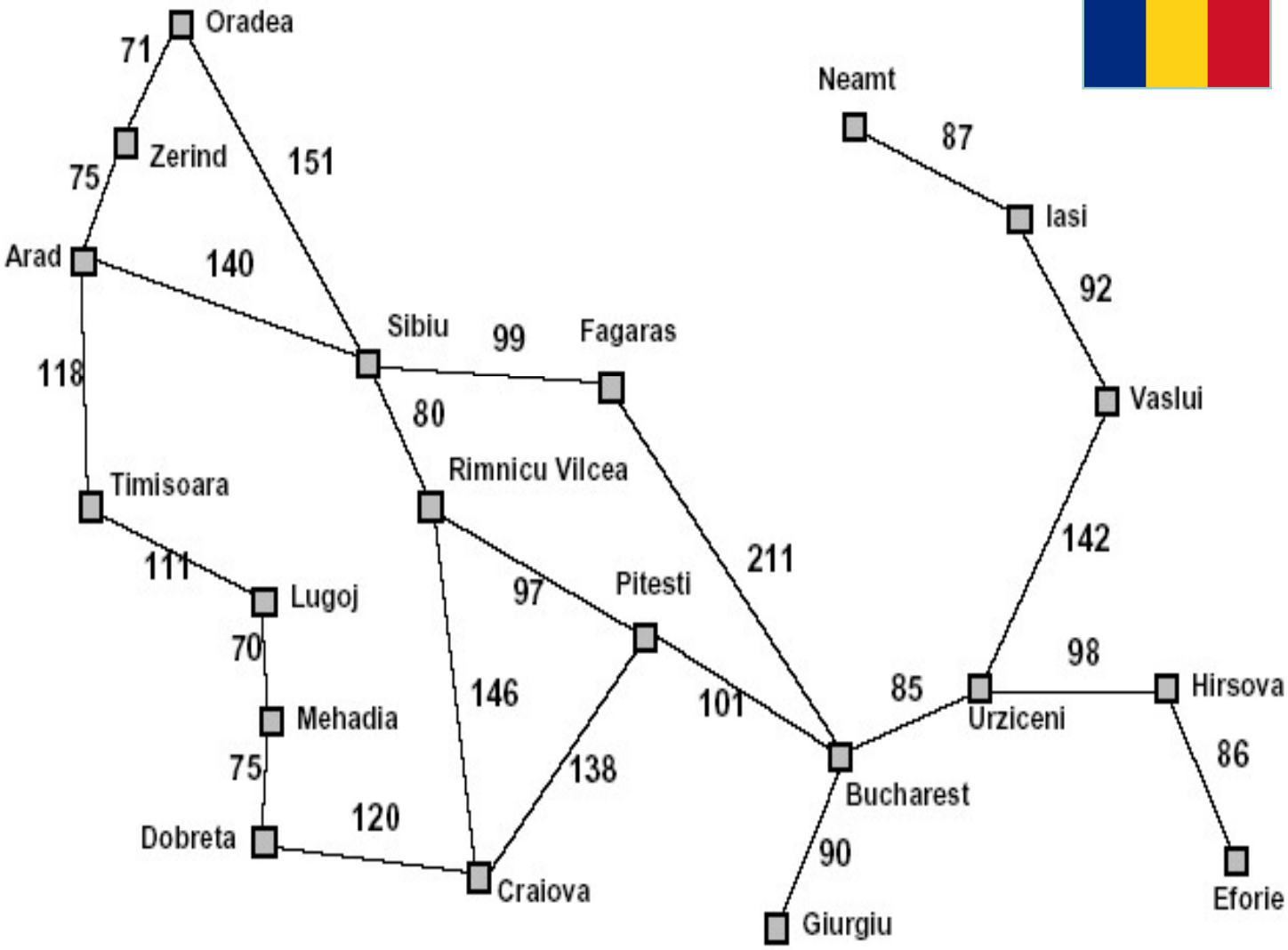
Manhattan



Euclidean

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Heuristic Search : Heuristic Function

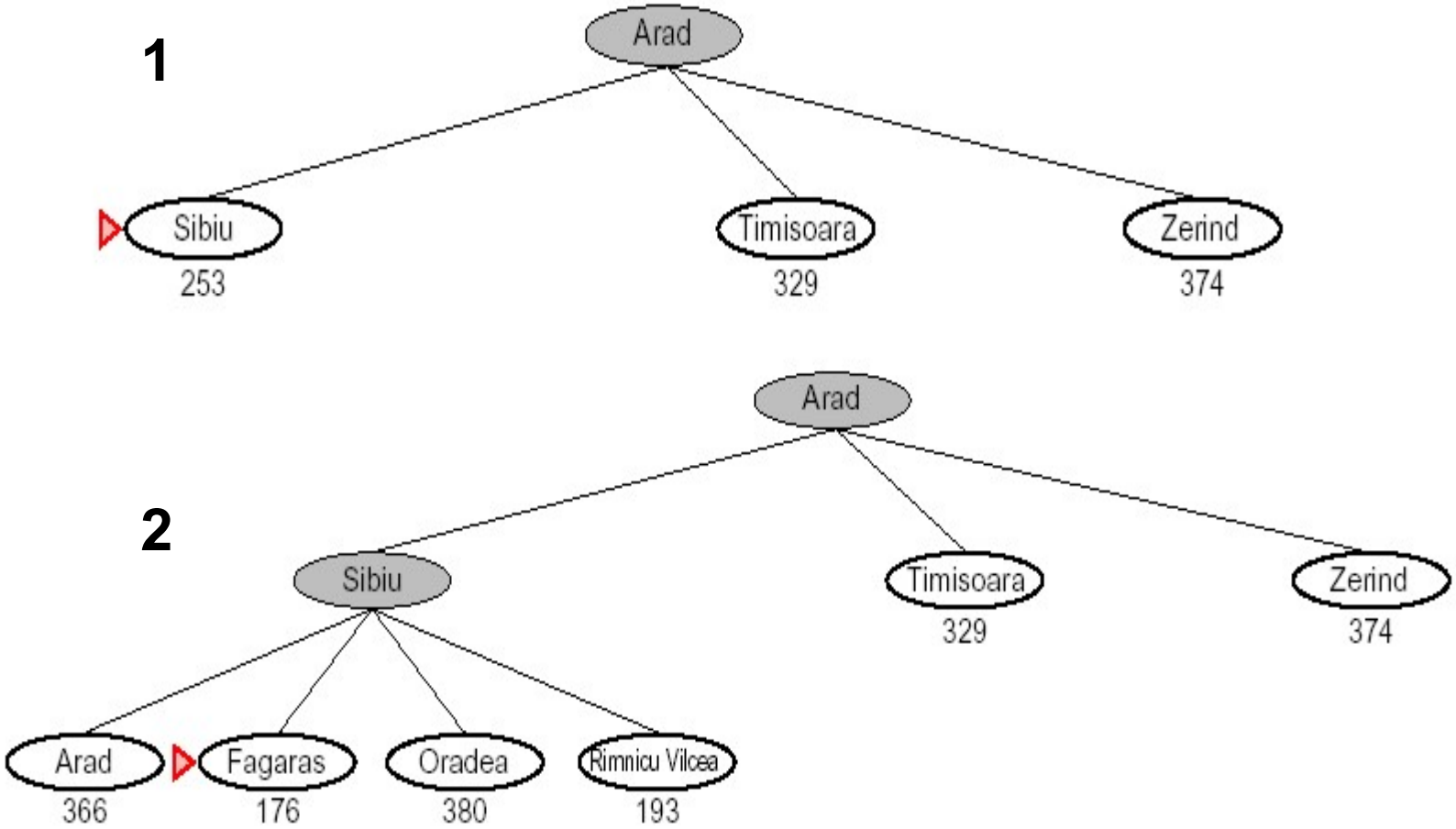


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

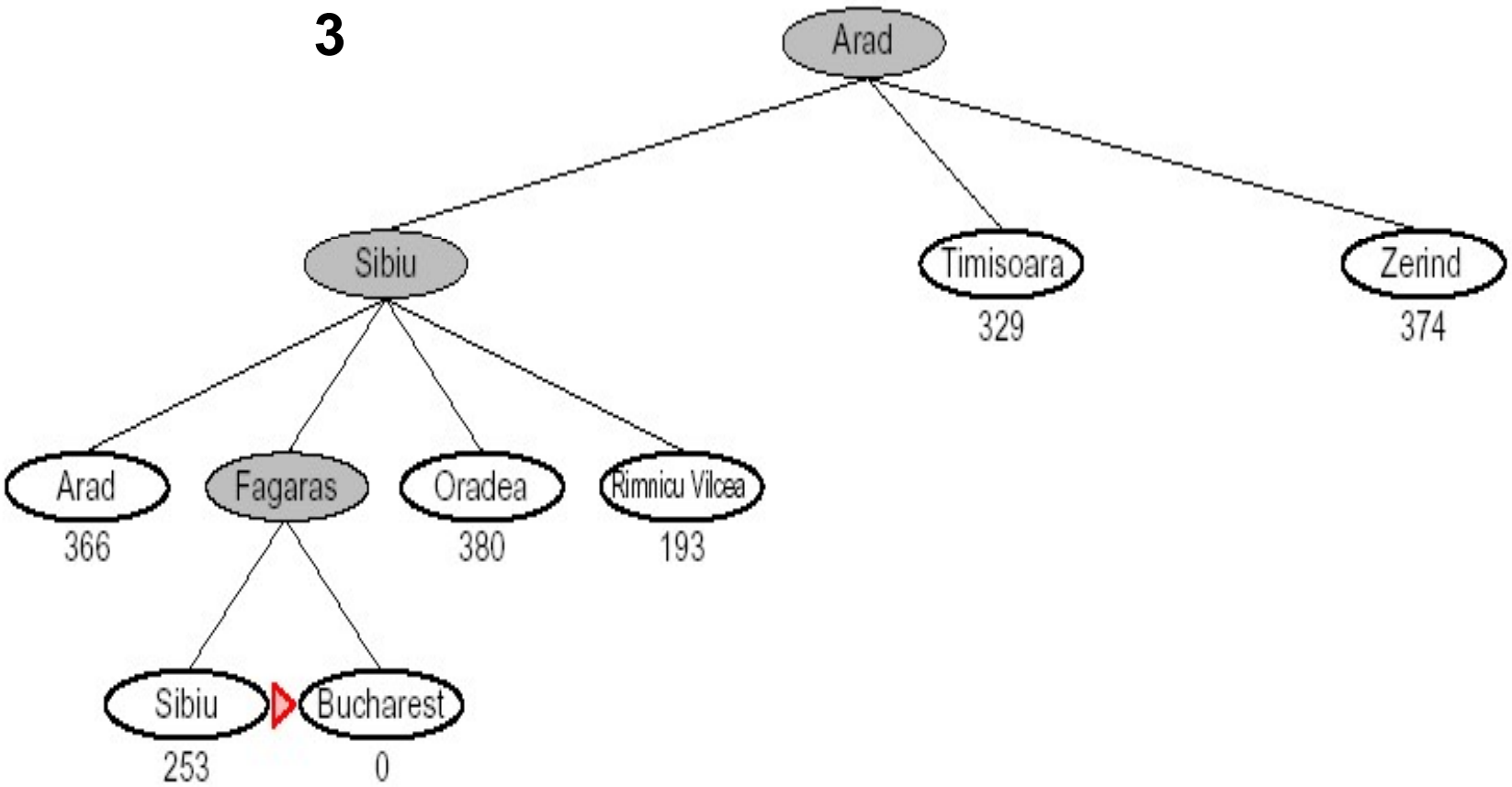
Heuristic Search :Greedy-Best Search

- Tries to expand the node that is closest to the goal.
- Evaluates using only heuristic function : $f(n) = h(n)$
- Possibly lead to the solution very fast.
- Problem ? ~ can end up in sub-optimal solutions (doesn't take notice of the distance it travels).
- Complexity and time $O(b^m)$
- Complete & optimal ? : No (stuck in infinite loop)

Heuristic Search :Greedy-Best Search



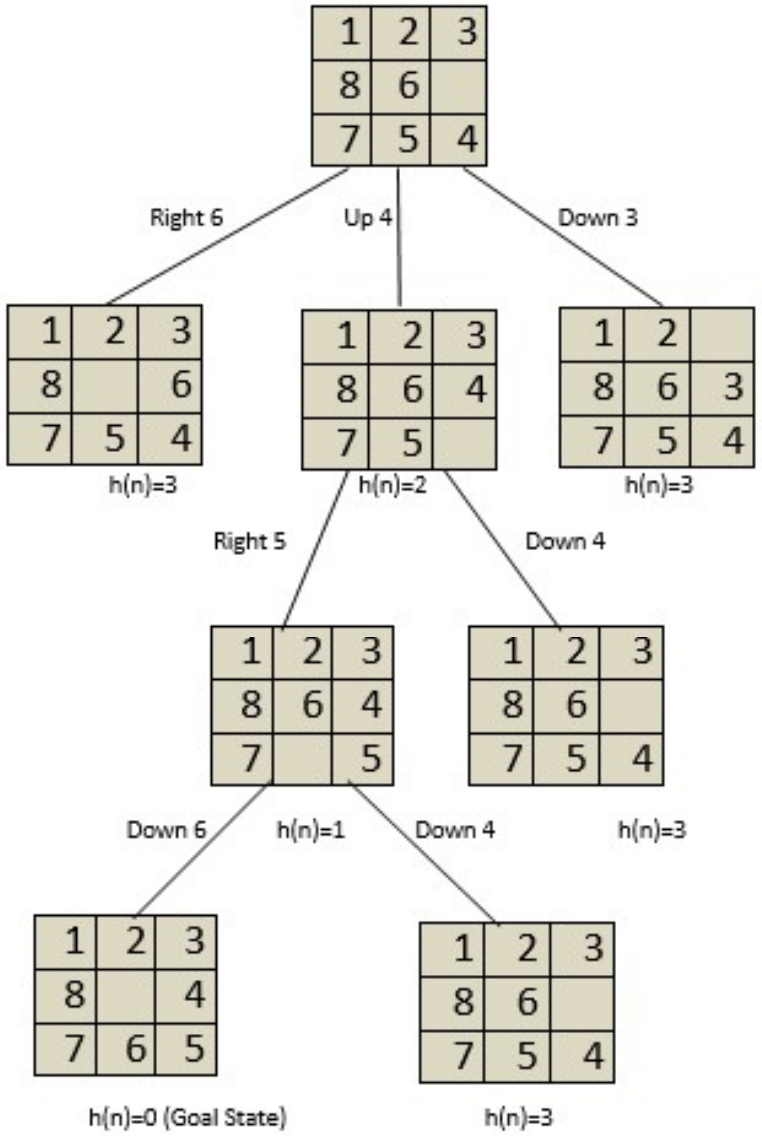
Heuristic Search :Greedy-Best Search



Heuristic Functions

Problem / Puzzle / Game	Heuristic Function	Remark
Travelling Salesman Problem	Eucledian Distance	$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$
8 Puzzle	Misplaced Tiles & Manhattan Distance	-
FCGW Problem	-	Farmer Cabbage Goat Wolf
Tik-Tac-Toe Game	Max Win Lines	-
8 Queen Problem	Number of pairs in attack position	-
Tower of Hanoi	-	Recursion
Monkey & Banana problem	-	-
Water Jug Problem	-	-
Snake Game	-	-
RPS Game	-	Rock Paper Scissor

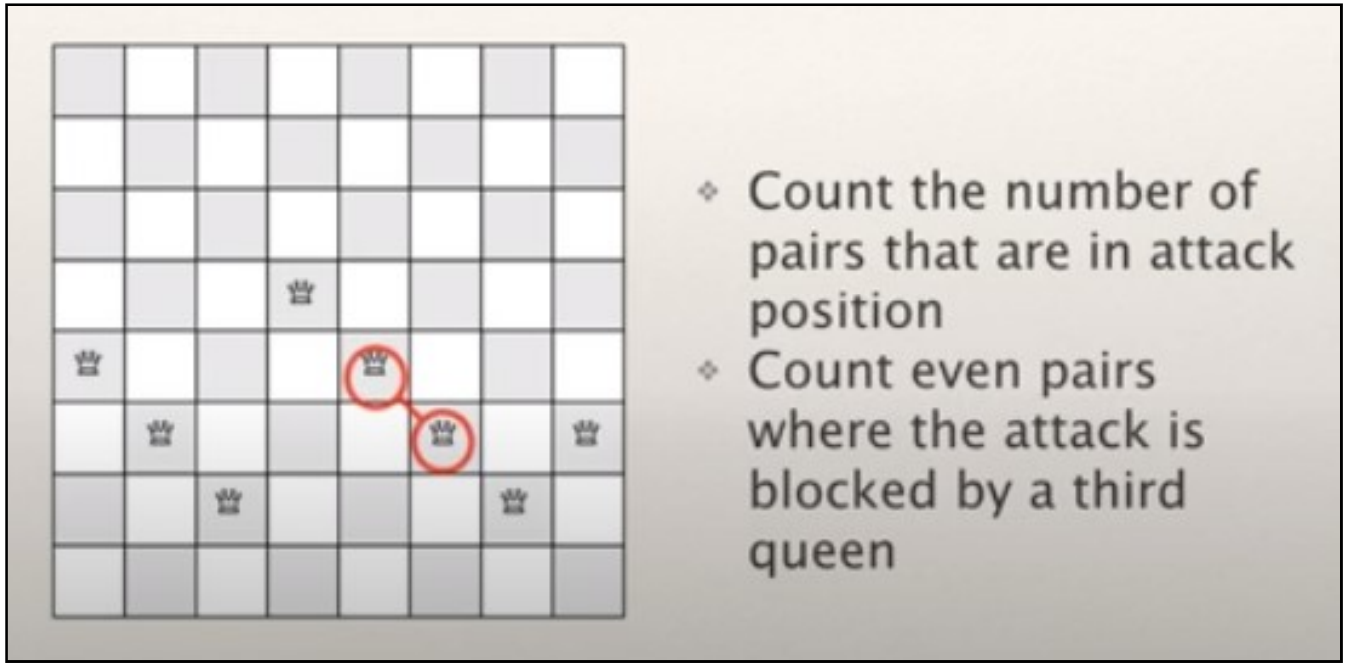
H(n) – 8 Puzzle



Misplaced Tiles

H(n) – Tic-Tac- Toe

H(n) – 8 Queen Problem

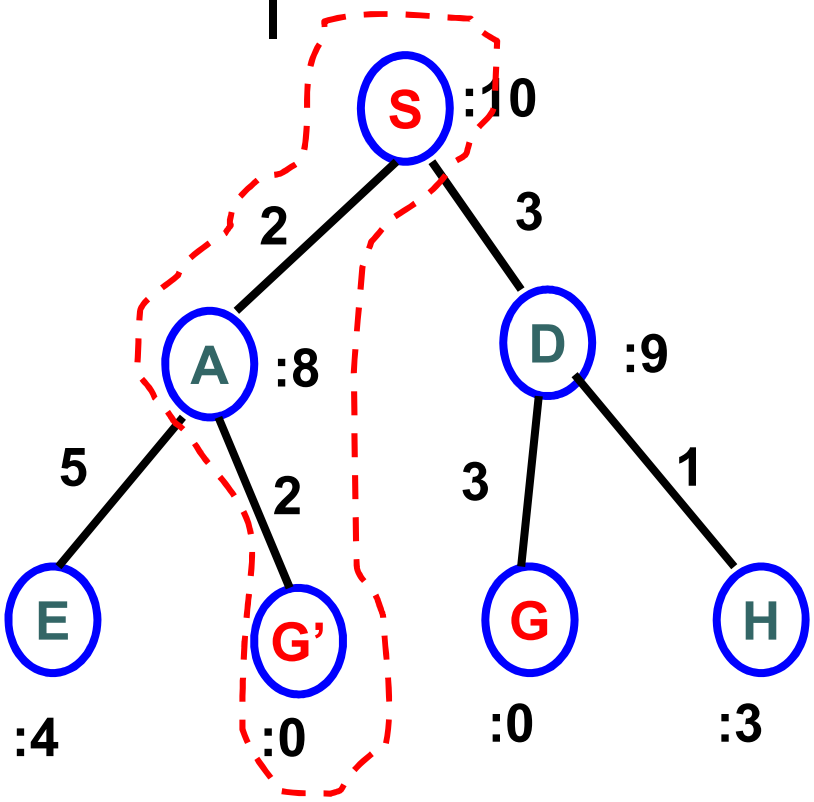


- Count the number of pairs that are in attack position
- Count even pairs where the attack is blocked by a third queen

Heuristic Search : A* Algorithm

- Widely known algorithm – (pronounced as '**A Star**' search).
- Evaluates nodes by combining g(n) "cost to reach the node" and h(n) "cost to get to the goal"
- $f(n) = g(n) + h(n)$, $f(n)$ → estimated cost of the cheapest solution.
- Complete and optimal ~ since evaluates all paths.
- Time ? ~ a bit time consuming
- Space ? ~ lot of it!

Heuristic Search : A* Algorithm



* Path S-A-G is chosen
= Lowest cost

Path cost for S-D-G

$f(S) = g(S) + h(S)$
$= 0 + 10 \rightarrow \mathbf{10}$
$f(D) = (0+3) + 9 \rightarrow \mathbf{12}$
$f(G) = (0+3+3) + 0 \rightarrow \mathbf{6}$

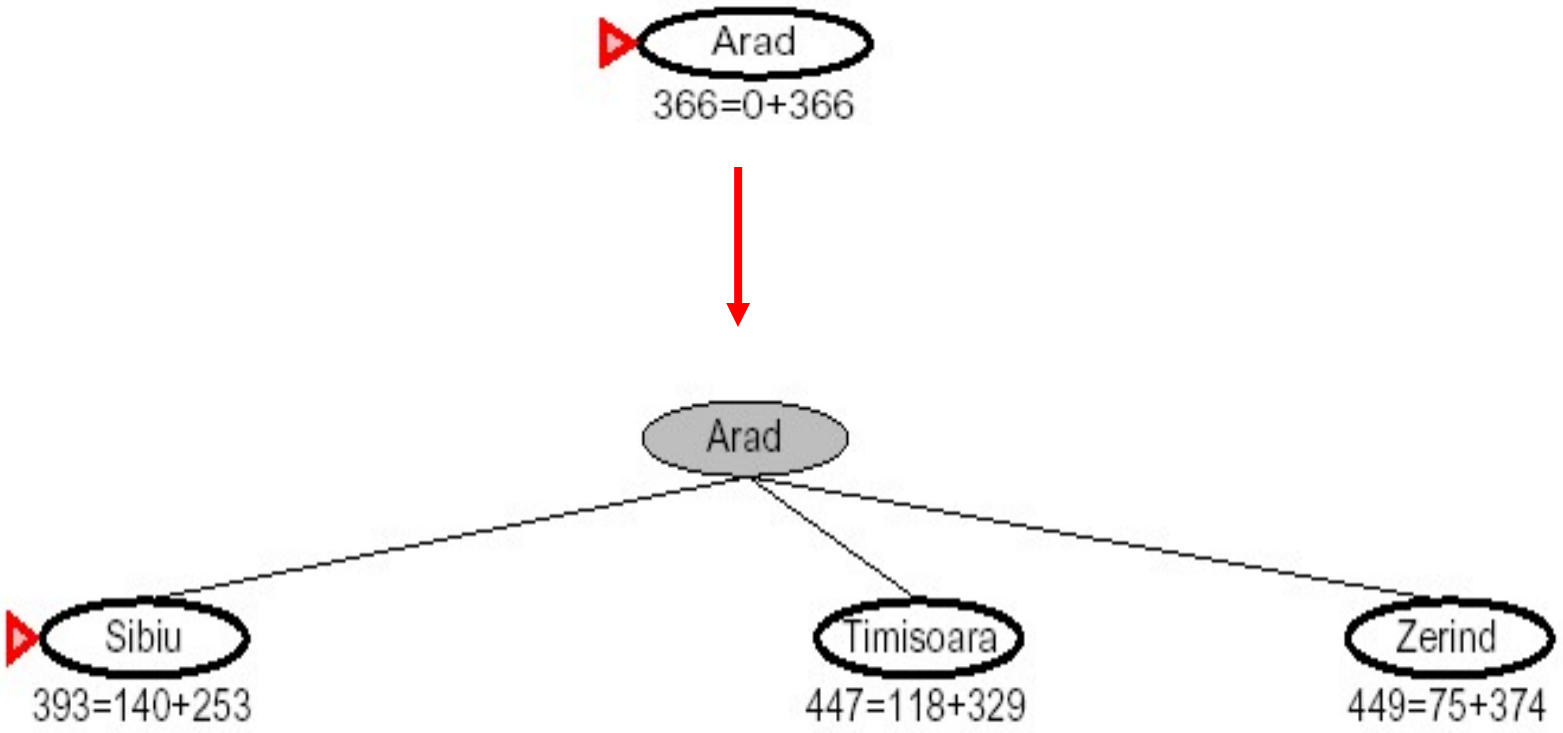
Total path cost = $f(S)+f(D)+f(G) \rightarrow \mathbf{28}$

Path cost for S-A-G'

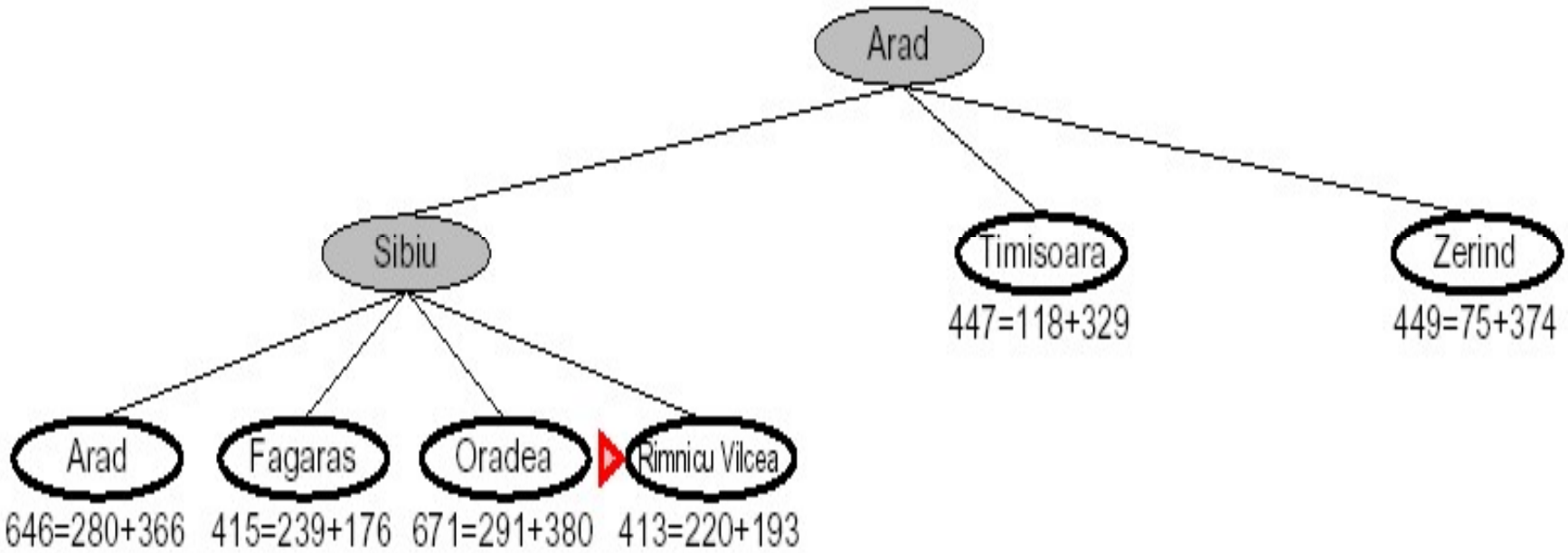
$f(S) = 0 + 10 \rightarrow \mathbf{10}$
$f(A) = (0+2) + 8 \rightarrow \mathbf{10}$
$f(G') = (0+2+2) + 0 \rightarrow \mathbf{4}$

Total path cost = $f(S)+f(A)+f(G') \rightarrow \mathbf{24}$

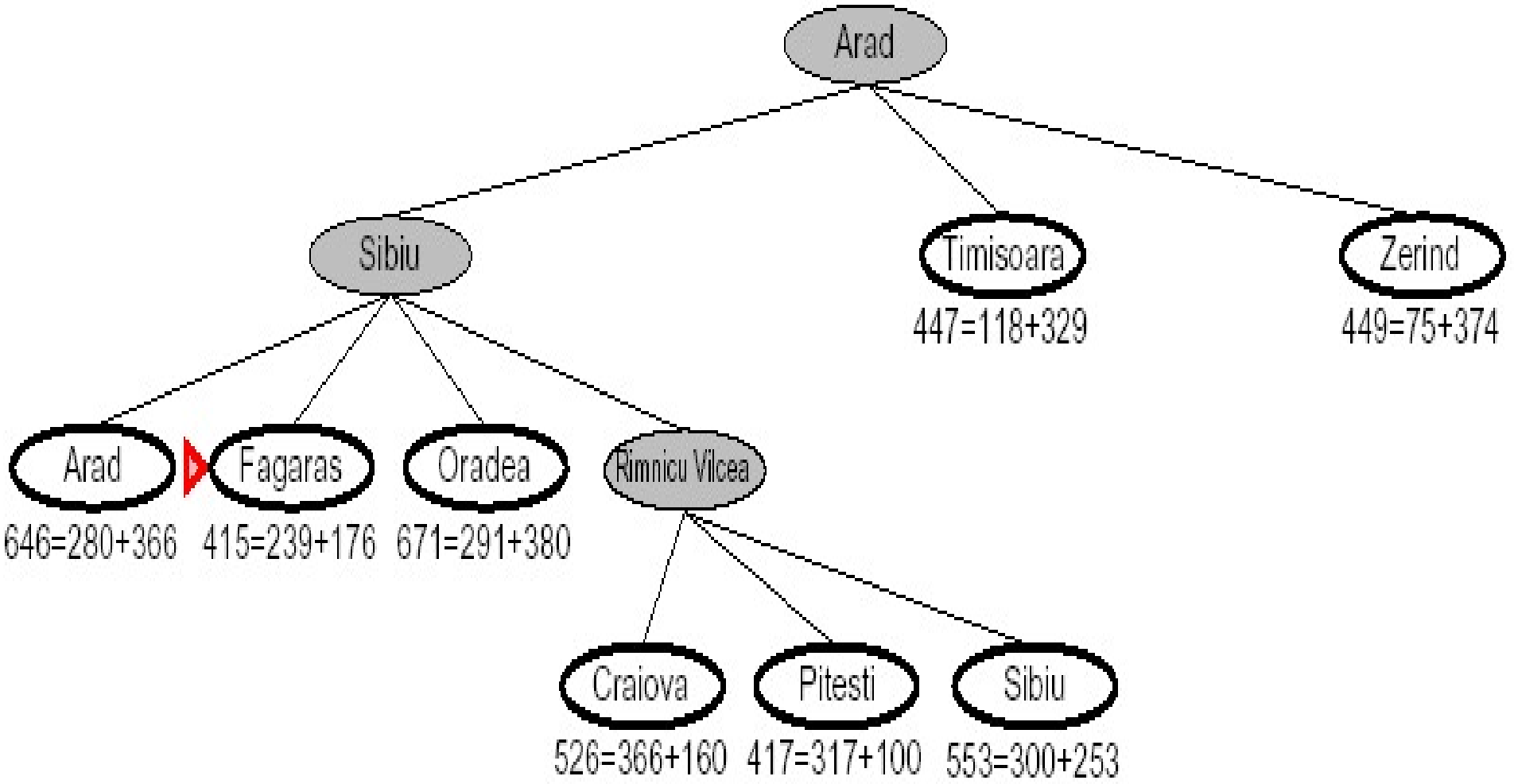
Heuristic Search : A* Algorithm



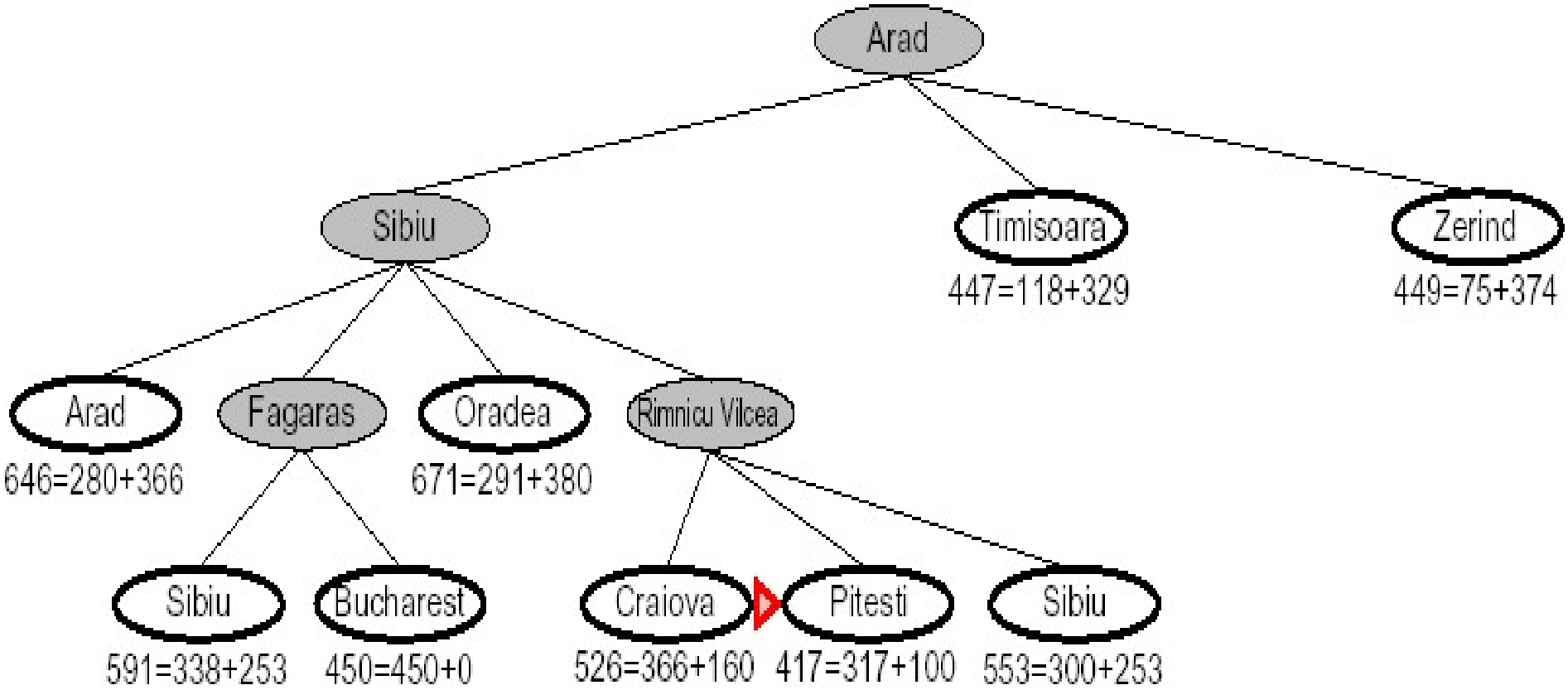
Heuristic Search : A* Algorithm



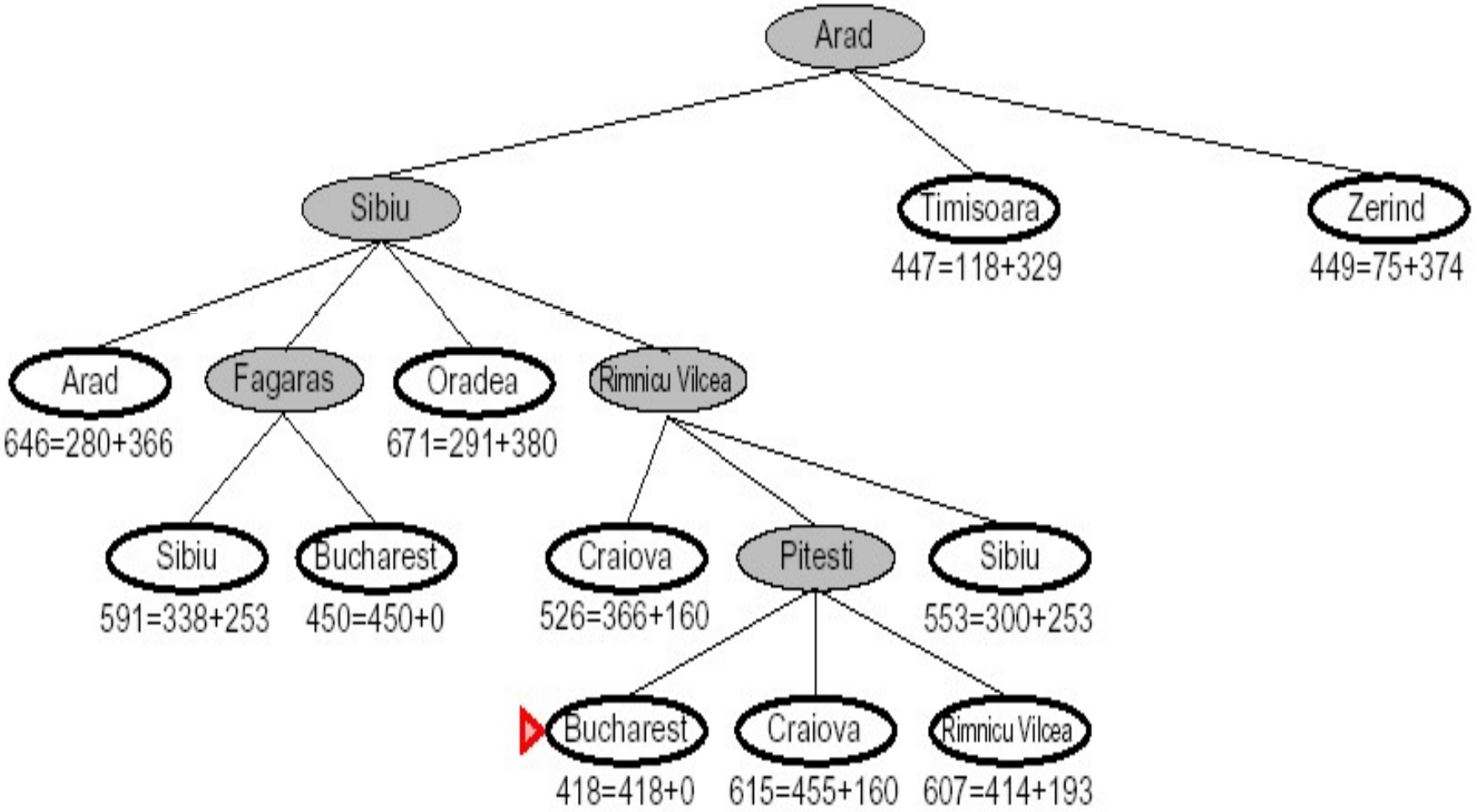
Heuristic Search : A* Algorithm



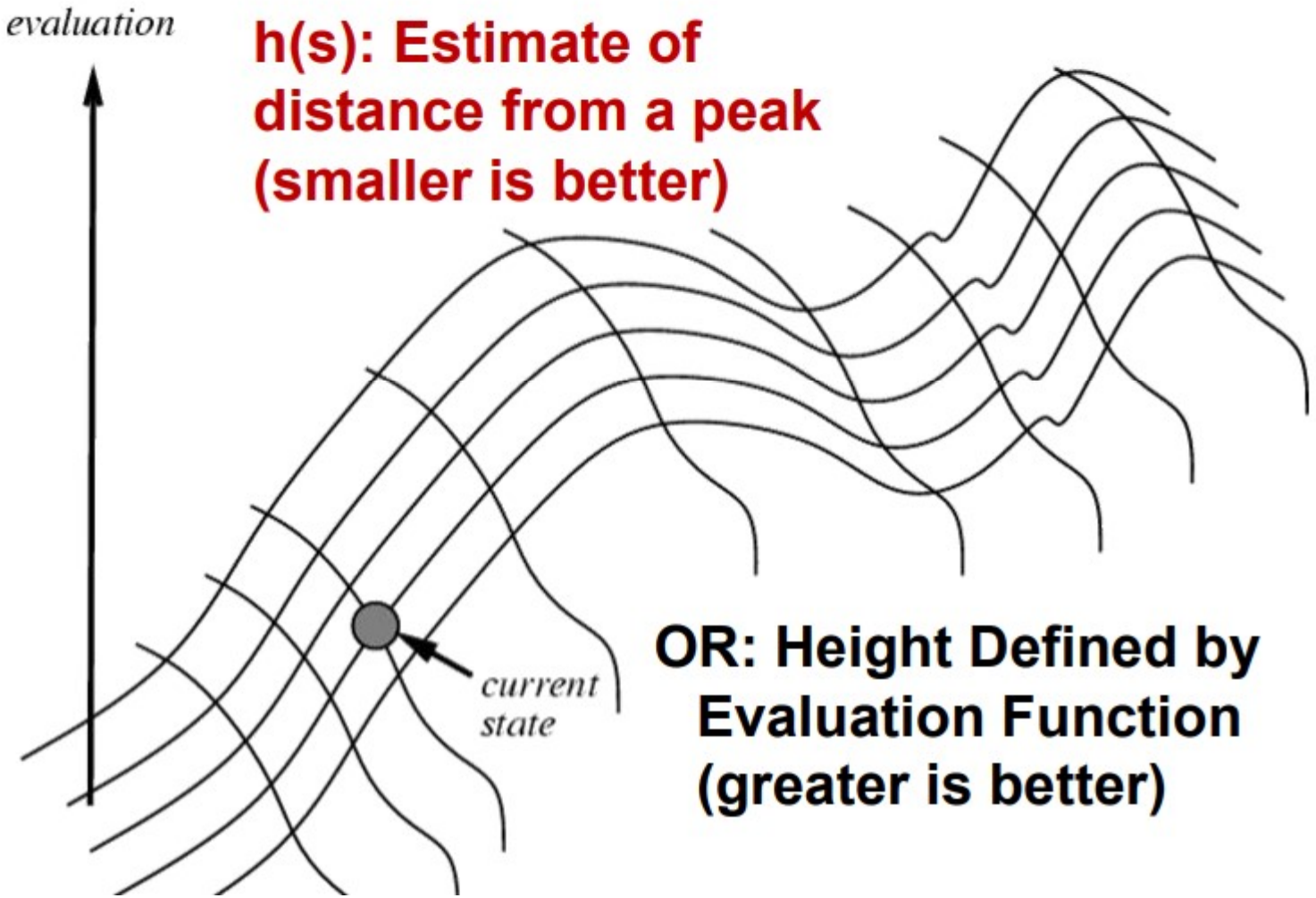
Heuristic Search : A* Algorithm



Heuristic Search : A* Algorithm



Hill-Climbing Algorithm



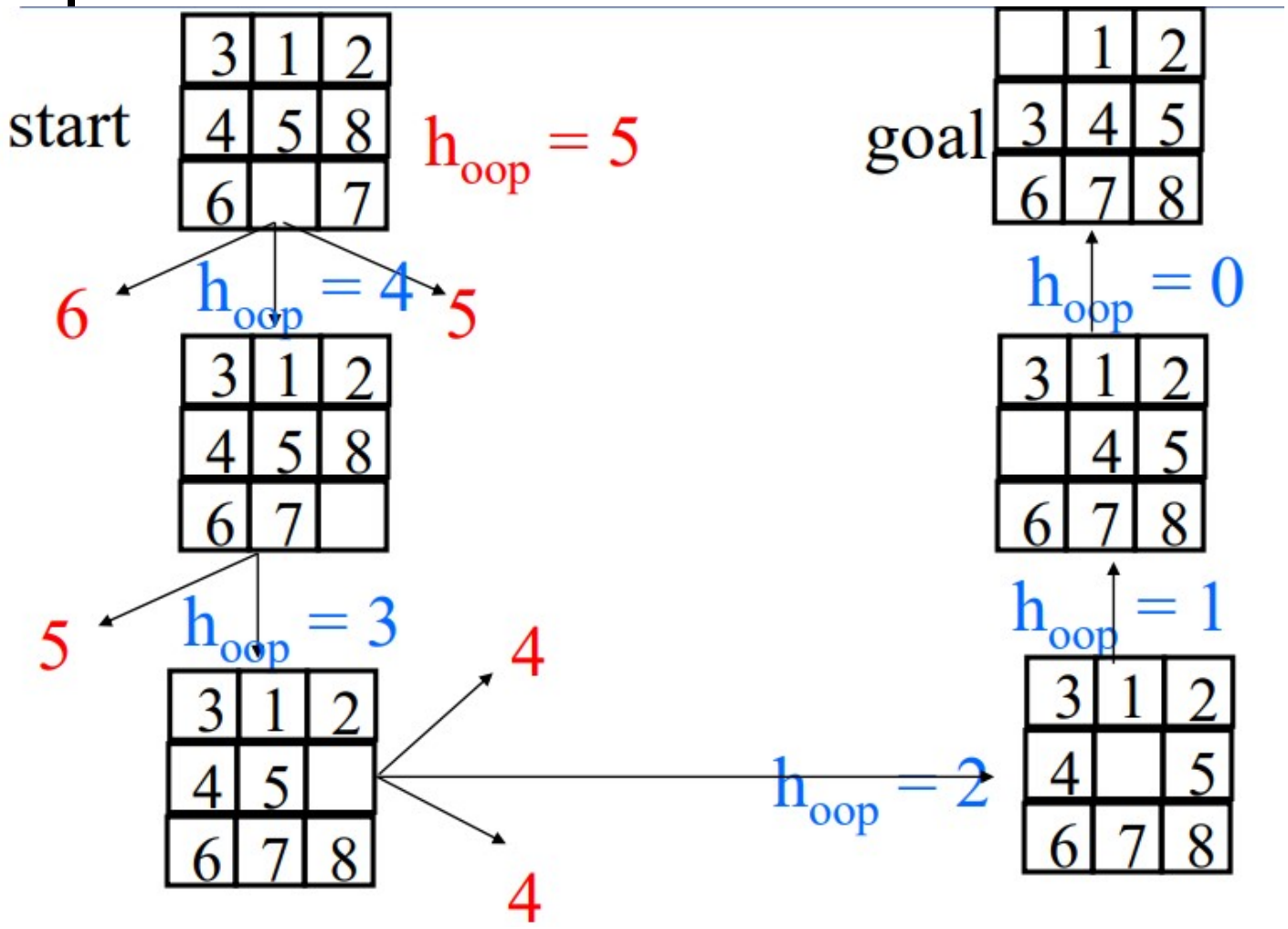
Hill-Climbing Algorithm

I. **While (∃ uphill points):**
   - Move in the direction of increasing evaluation function ***f***

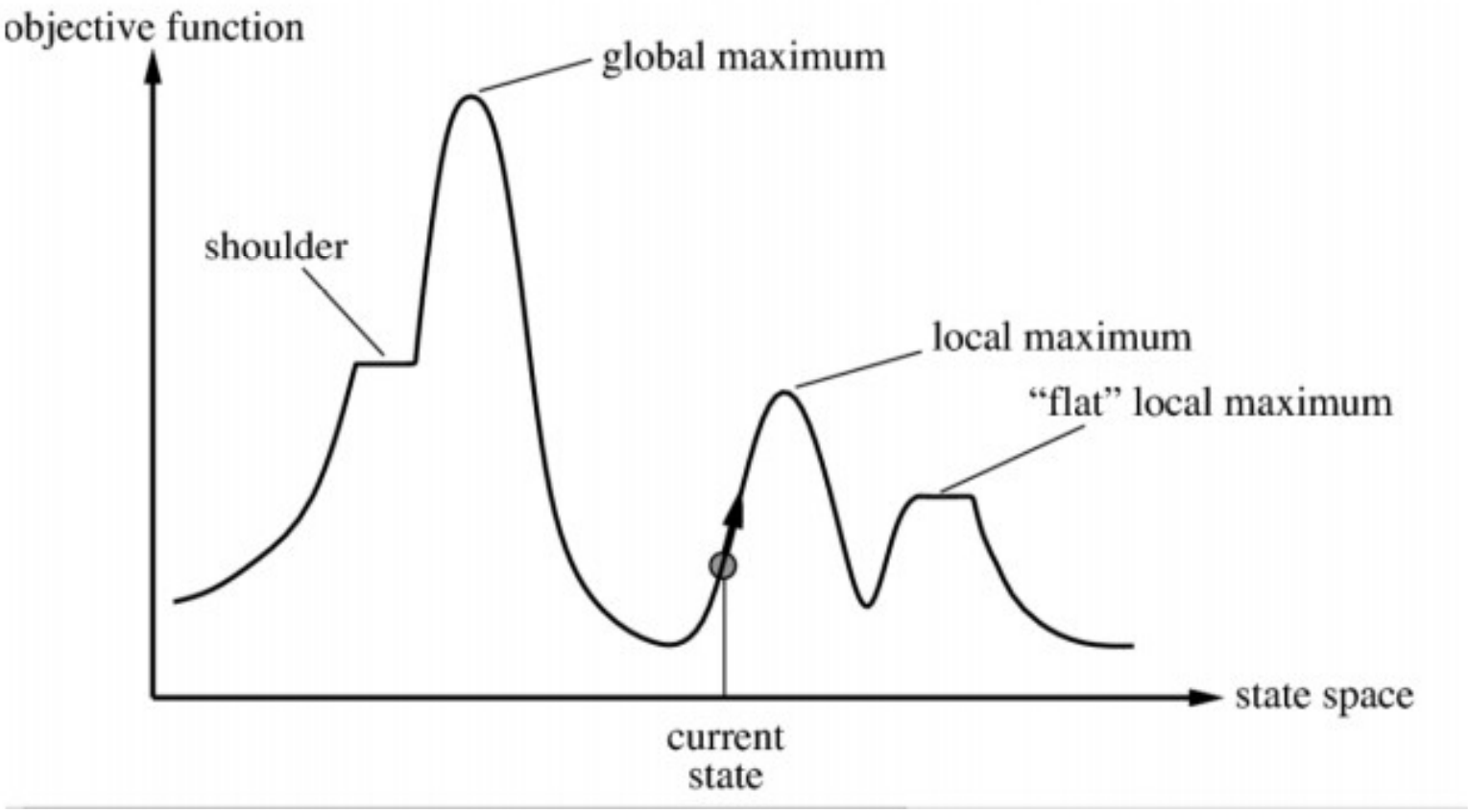
II. **Let** $s_{next} = \arg\max_{s} f(s)$ **, *s* a successor state to the current state n**
   - If ***f(n)*** < ***f(s)*** then move to ***s***
   - Otherwise halt at ***n***

- **Properties:**
  - Terminates when a peak is reached.
  - Does not look ahead of the immediate neighbors of the current state.
  - Chooses randomly among the set of best successors, if there is more than one.
  - Doesn't *backtrack*, since it doesn't remember where it's been
- **a.k.a. *greedy local search***

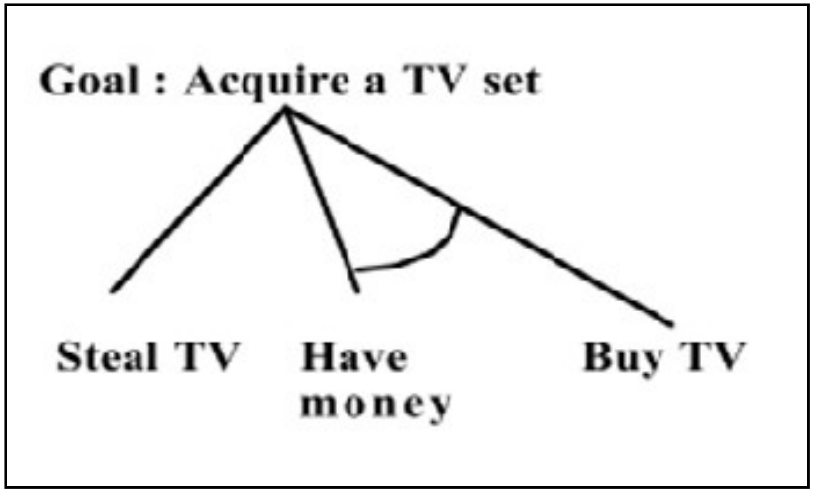
Hill-Climbing Alg - Example

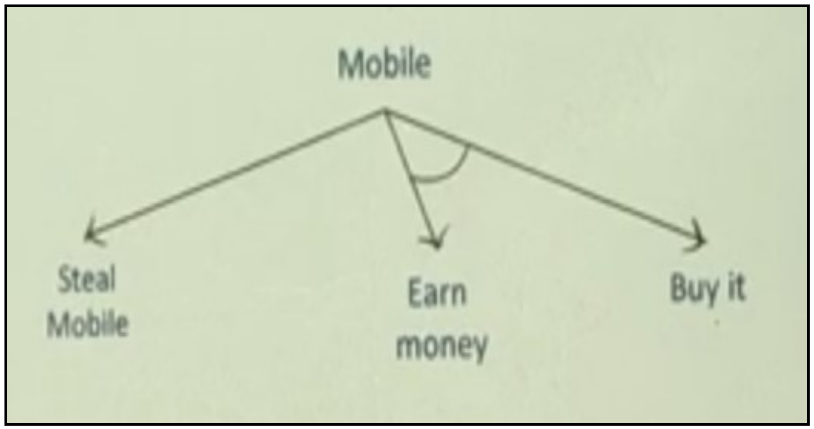


Hill-Climbing – Search Space Features



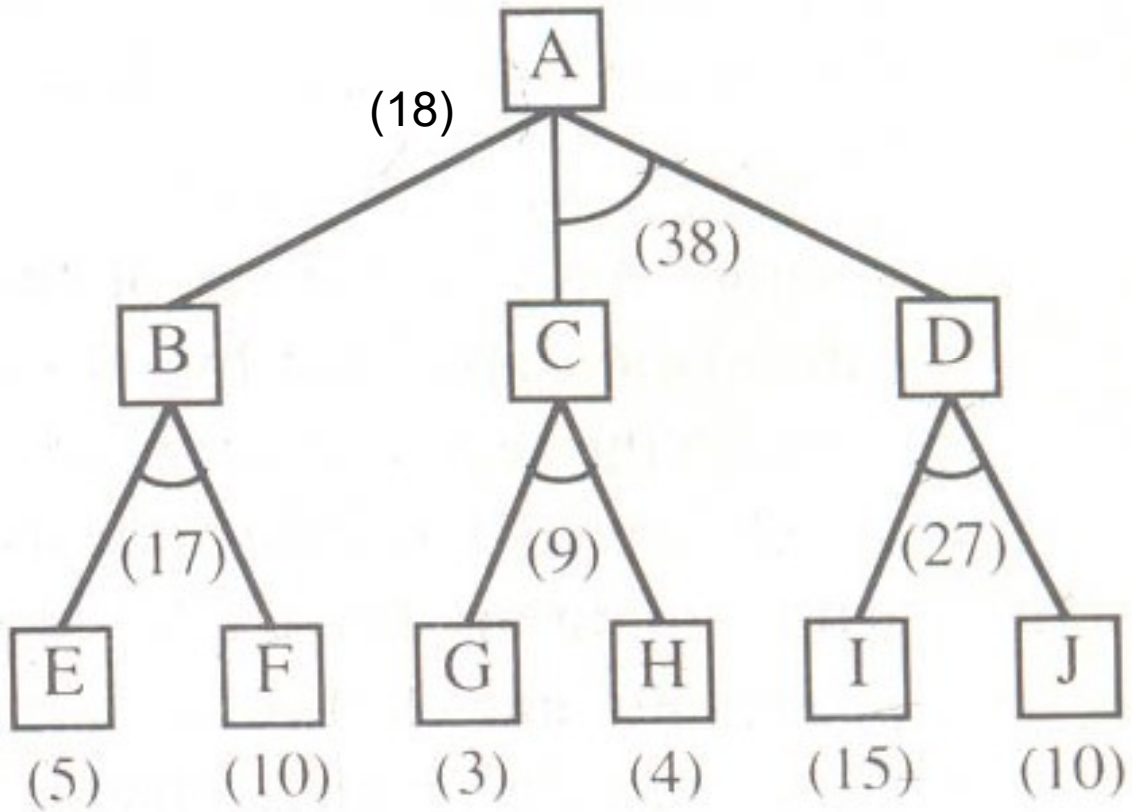
AO* Algo – AND / OR Graph



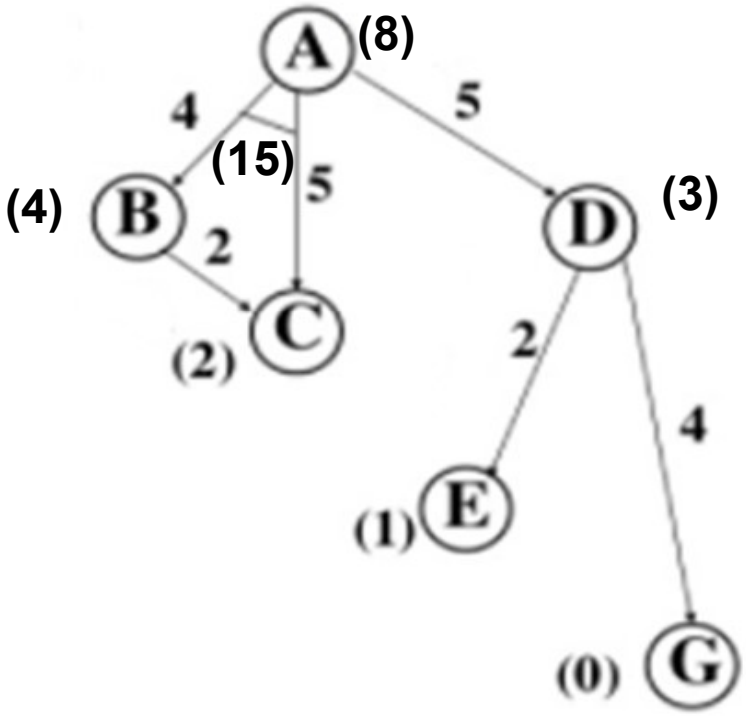


- **OR Graphs** - Generally used for data driven approach
- **AND OR Graphs** - Used for Goal driven approach
  - Problems solvable by decomposing into sub problems some of which is to be solved
  - **OR** - the node has to be solved
  - **AND** - all the nodes in the arc has to be solved

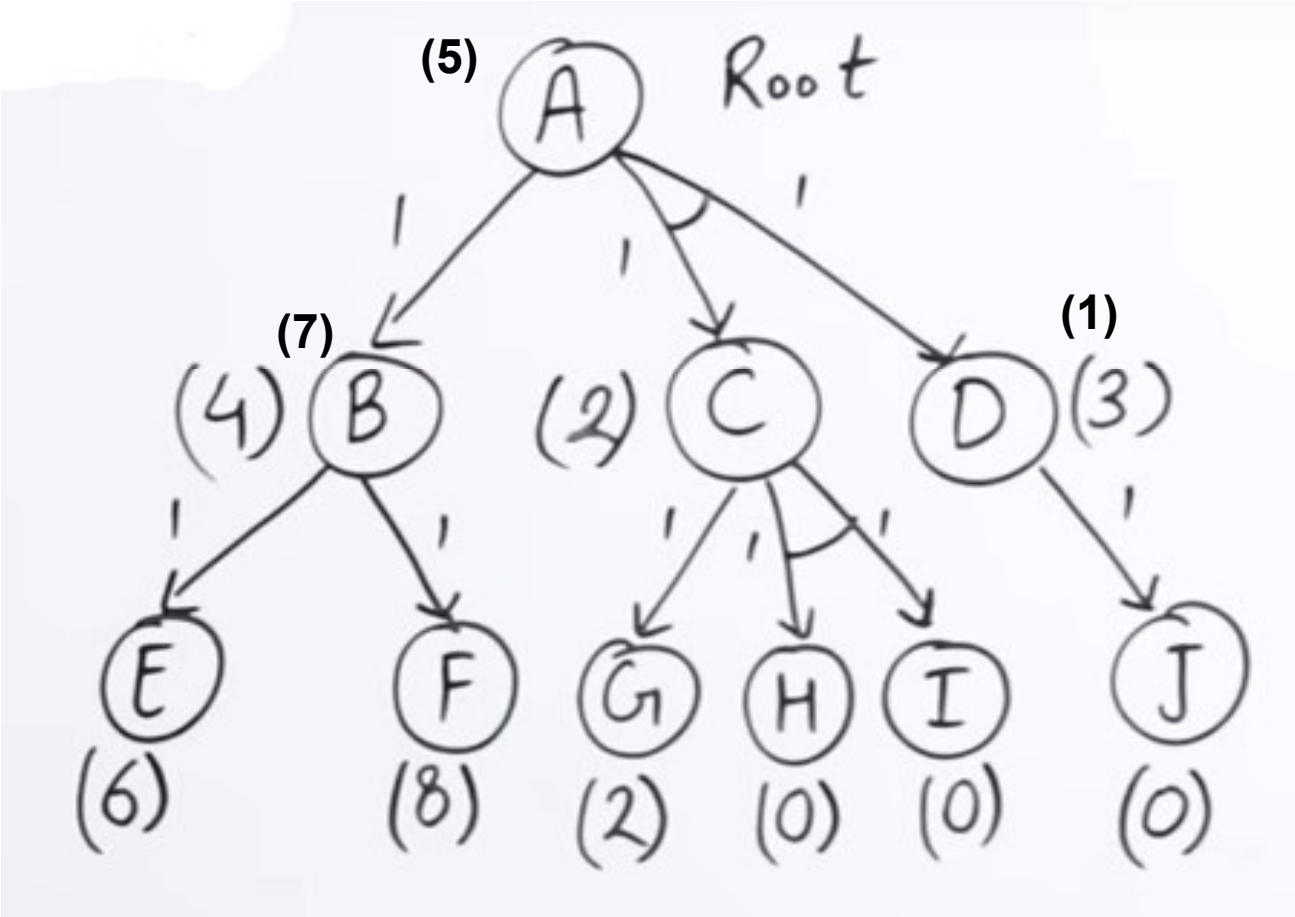
AO* Algo



AO* Algo



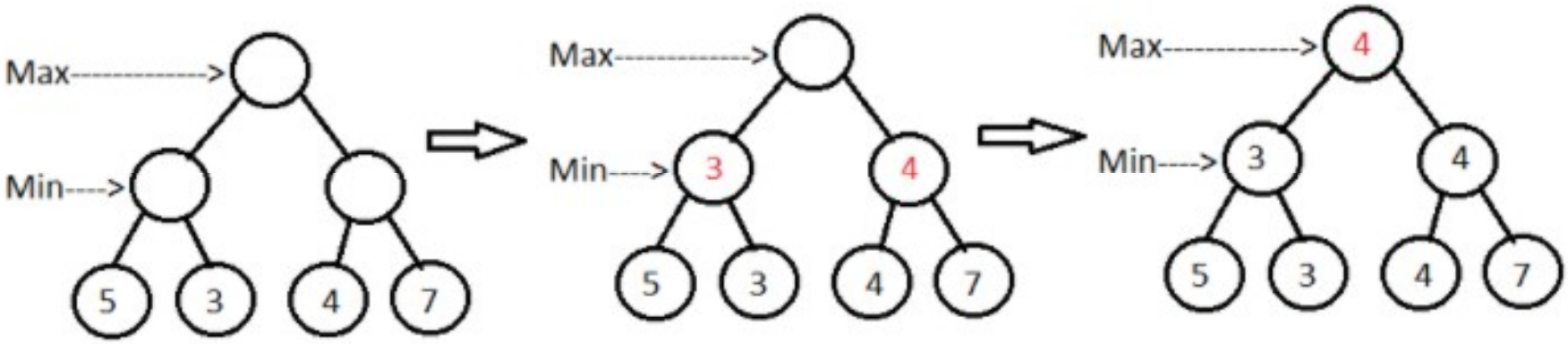
AO* Algo



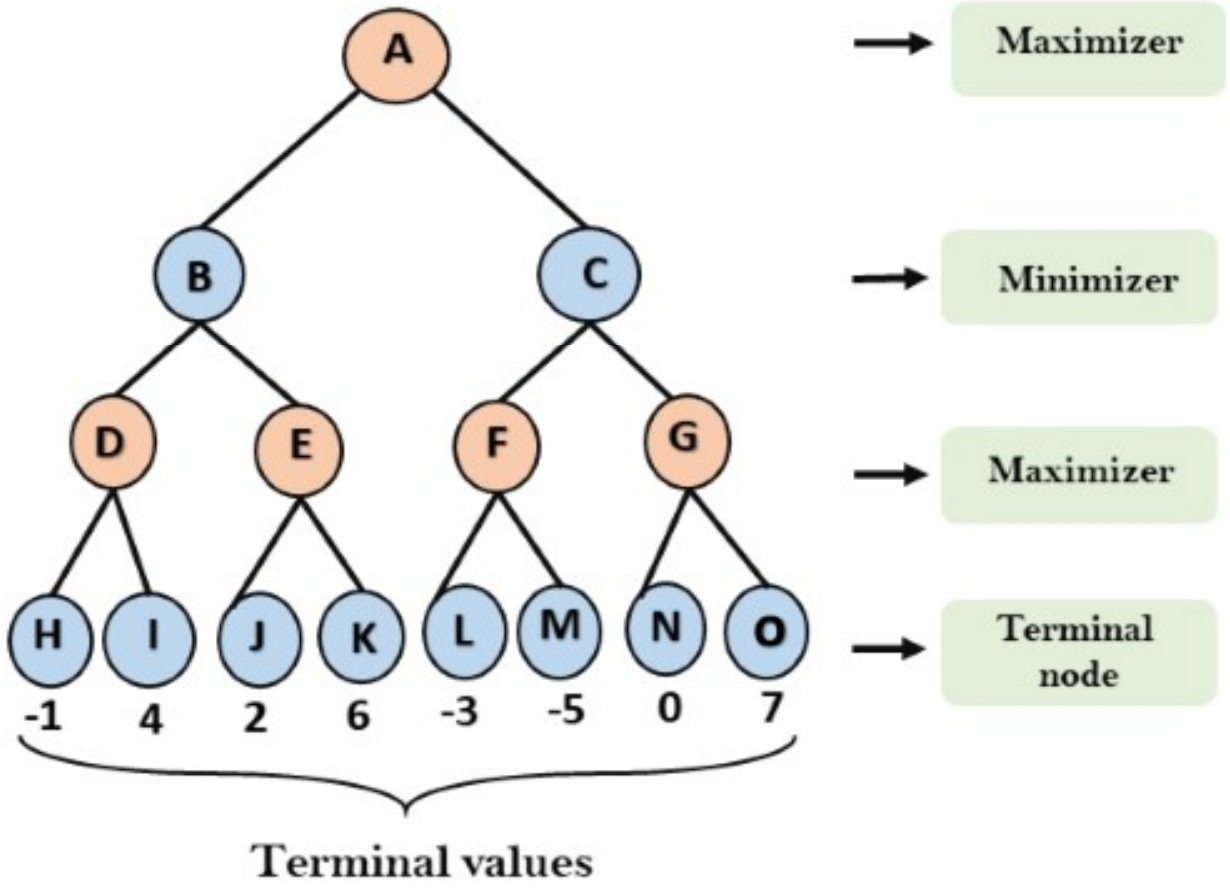
MinMax Algo - Games

- Consider games with two players (**MAX, MIN**)
- Backtracking algo
- **Initial State:**
  - Board position and identifies the player to move
- **Successor Function:**
  - Returns a list of (move, state) pairs; each a legal move and resulting state
- **Terminal Test:**
  - Determines if the game is over (at terminal states)
- **Utility Function:**
  - Objective function, payoff function, a numeric value for the terminal states (**+1, 0, -1**) or (**+10, 0, -10**)

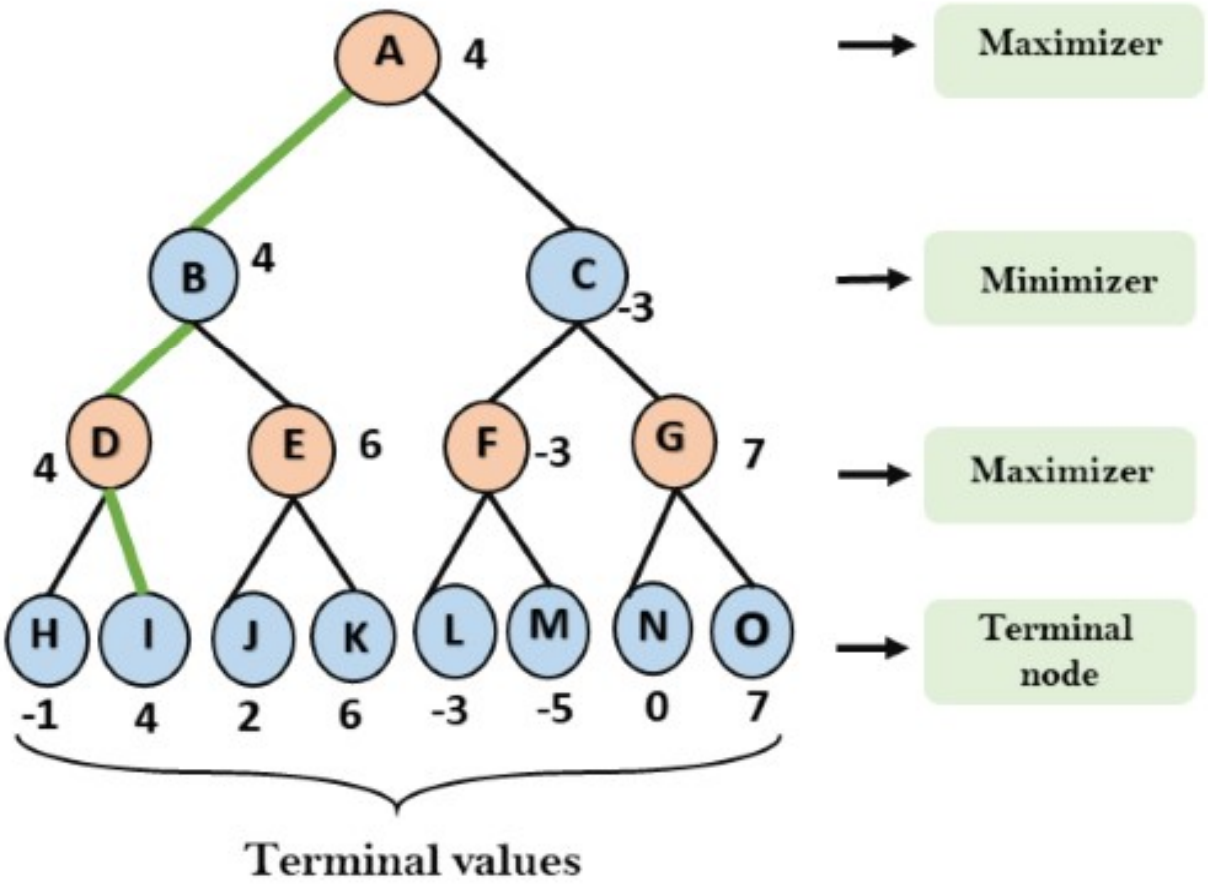
MinMax Algo - Example



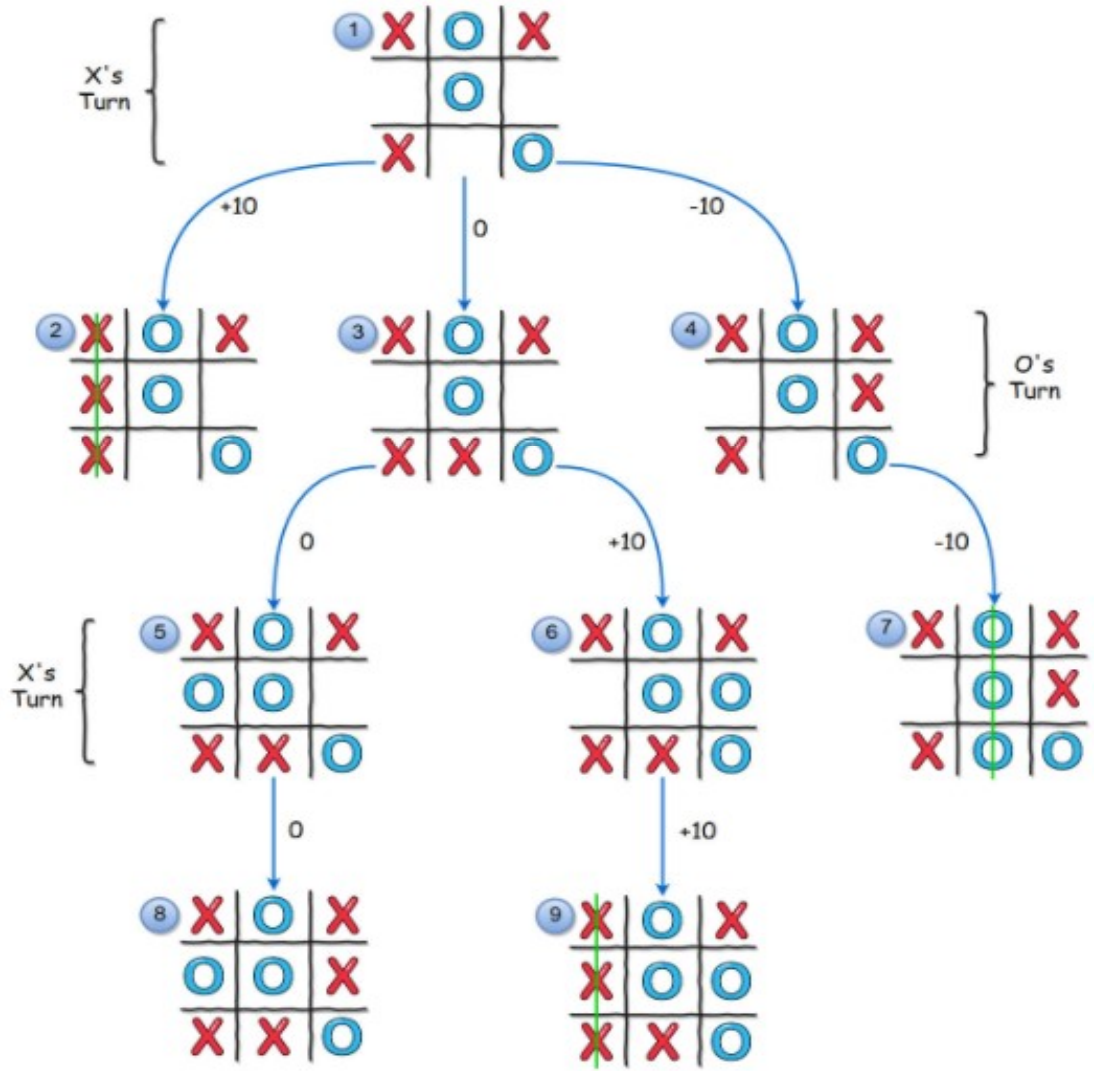
MinMax Algo - Game Tree



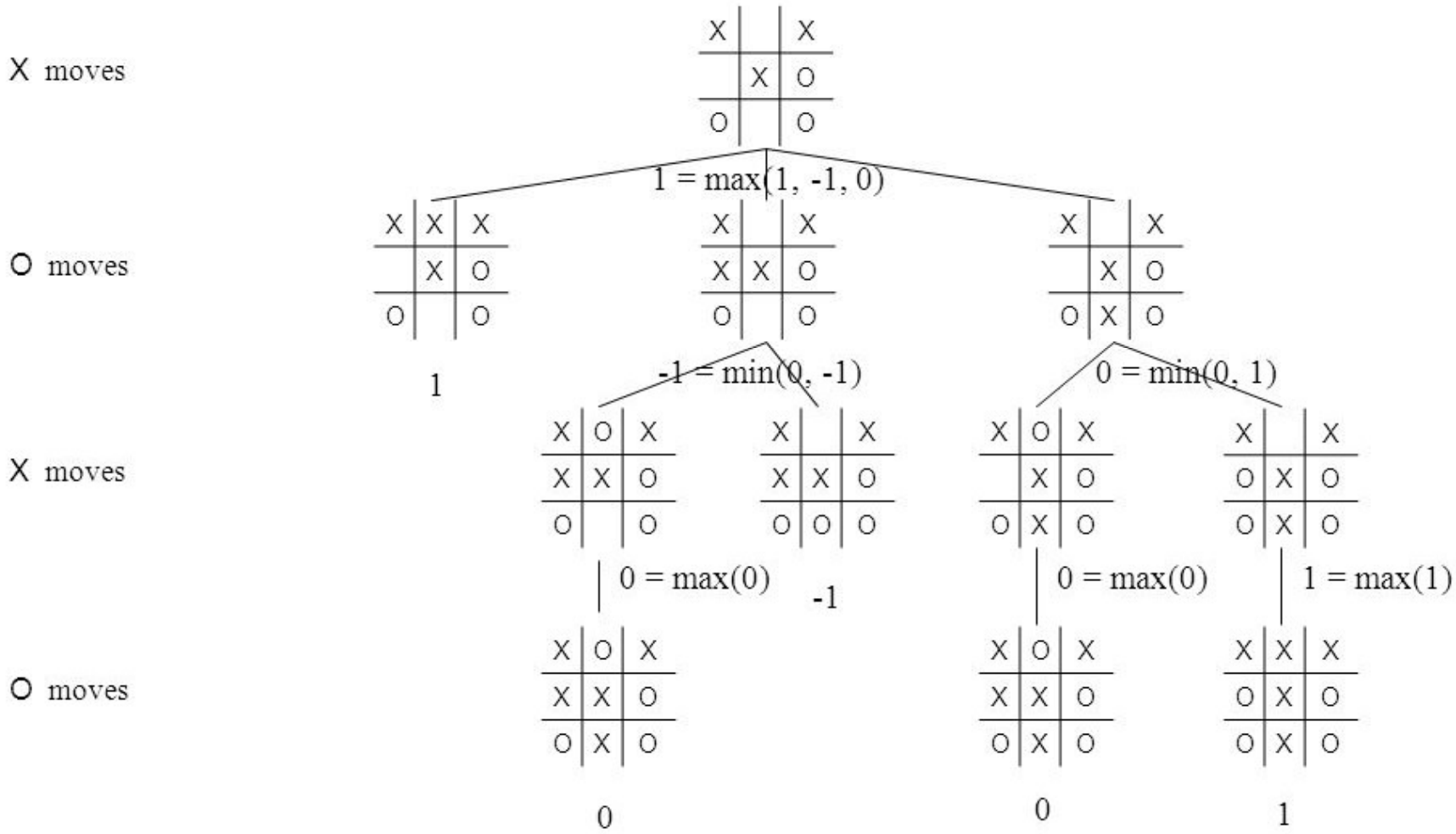
MinMax Algo - Solution



MinMax Algo - Tik Tak Toe



MinMax Algo - Tik Tak Toe (Contd…)



Constraint Satisfaction Problems (CSP)

- A **Constraint Satisfaction Problem** is a special kind of problem that satisfies some additional structural properties beyond the basic requirements for problems.

- In a CSP, the states are defined by the values of a set of **variables** and the goal test specifies a set of **constraints** that the values have to obey.

- **State Components** -
  1. **Variables**,
  2. **Domains** (possible values for the variables),
  3. **Constraints** (between variables)

- **Goal:** To find a state (a complete assignment of values to variables), which satisfies the constraints.

Constraint Satisfaction Problems (CSP)

- **Cryptarithmetic problems / puzzles**
- **Map coloring**
- **Crossword puzzles**
- **Others - Resources / Distribution / Location**

Cryptarithmetic Puzzles

- In 1924 Henry Dudeney published a popular number puzzle of the type known as a cryptarithm, in which letters are replaced with numbers.
- Dudeney's puzzle reads: SEND + MORE = MONEY.
- Cryptarithms are solved by deducing numerical values from the mathematical relationships indicated by the letter arrangements (i.e.) . S=9, E=5, N=6, M=1, O=0,….
- The only solution to Dudeney's problem: 9567 + 1085 = 10,652.

Cryptarithmetic Puzzle

- **Variables:** $F, T, U, W, R, O, X_1, X_2, X_3$
- **Domains:** $\{0,1,2,3,4,5,6,7,8,9\}$
- **Constraints**:
  - All diff (F,T,U,W,R,O)
  - $O + O = R + 10 \cdot X_1$
  - $X_1 + W + W = U + 10 \cdot X_2$
  - $X_2 + T + T = O + 10 \cdot X_3$
  - $X_3 = F$, $T \neq 0$, $F \neq 0$

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

Cryptarithmetic Puzzle - Solution

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

Setting F = 1, O = 4, R = 8,

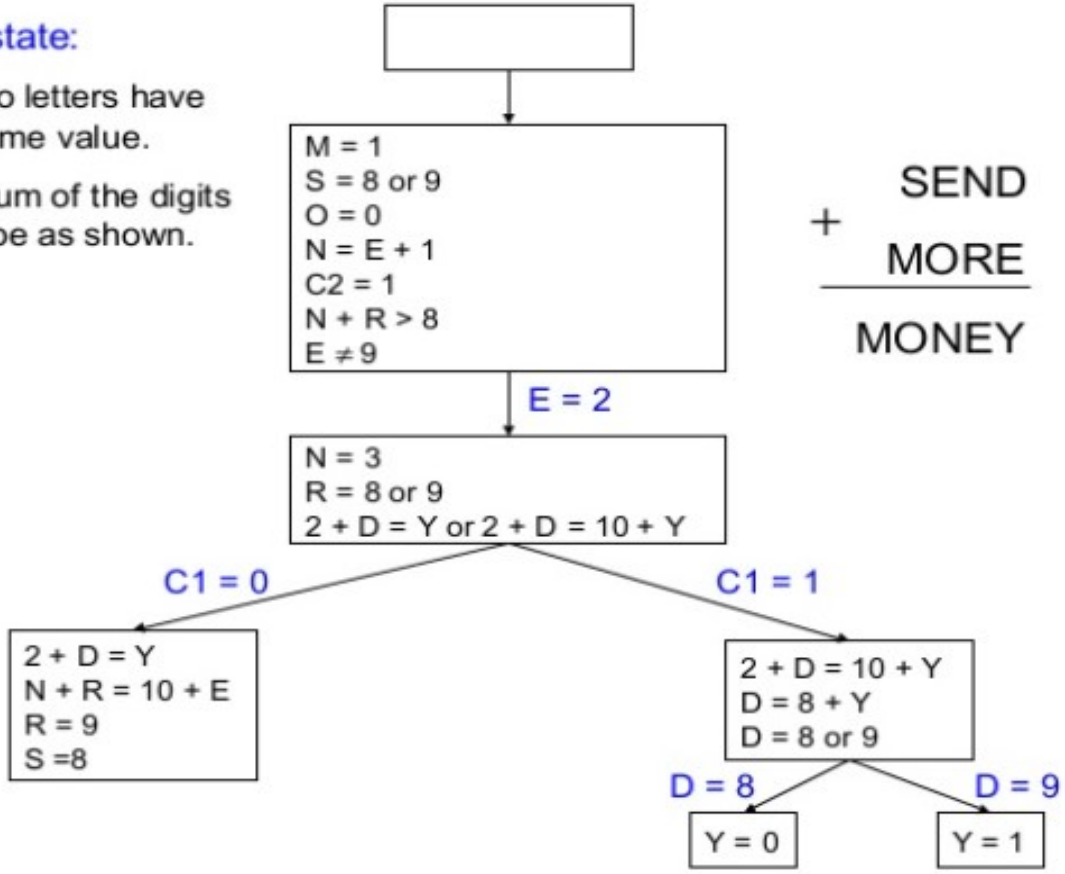
T = 7, W = 3, U = 6

Gives **734 + 734 = 1468**

Cryptarithmetic Puzzles

Initial state:

- No two letters have the same value.
- The sum of the digits must be as shown.




```
  SEND
+ MORE
------
 MONEY
```

S	9
E	5
N	6
D	7
M	1
O	0
R	8
y	2

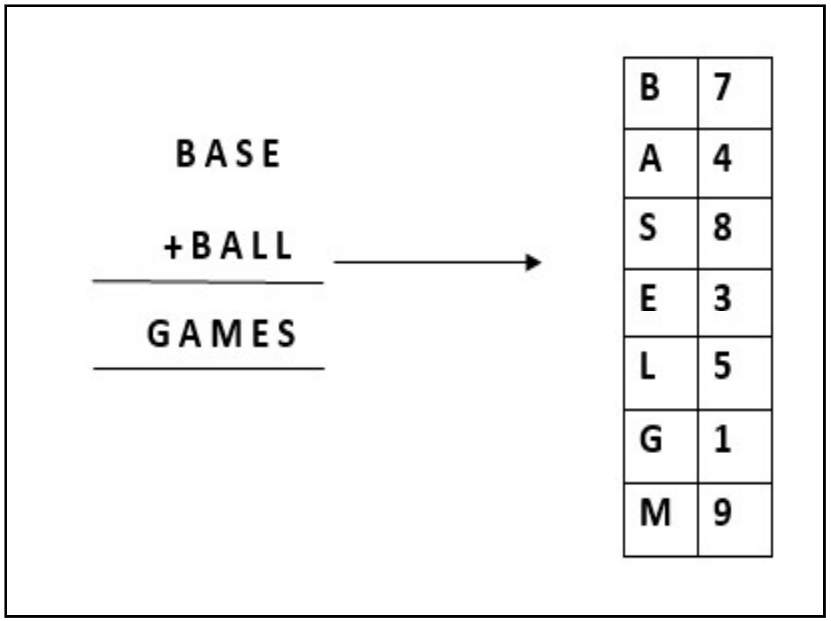
Cryptarithmetic Puzzles

```
  BASE
 +BALL
------
 GAMES
------
```

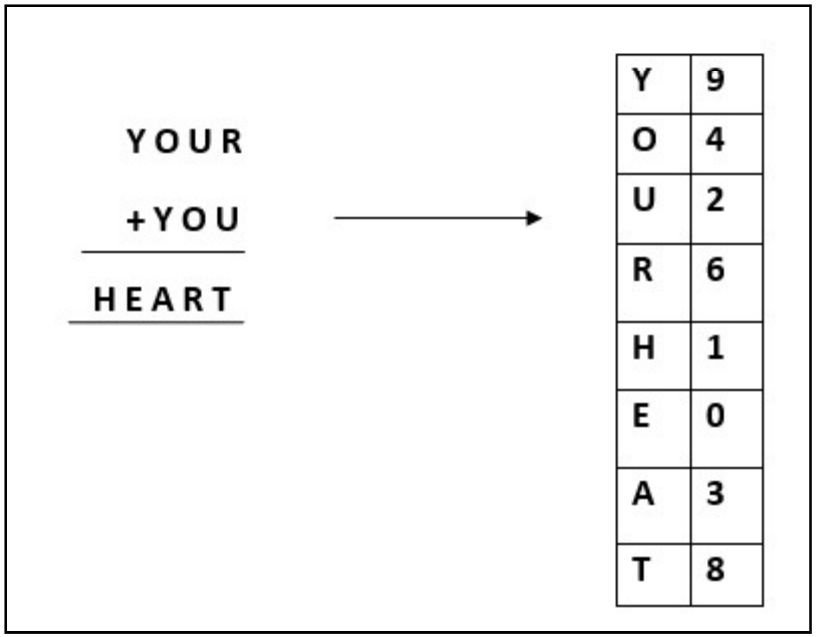
```
  YOUR
 +YOU
------
 HEART
------
```

```
  FORTY
 + TEN
 + TEN
---------
  SIXTY
```

Cryptarithmetic Puzzles



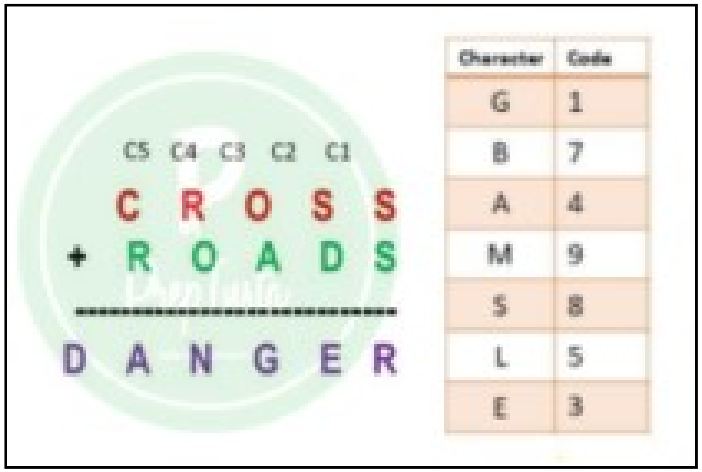
B	7
A	4
S	8
E	3
L	5
G	1
M	9



Y	9
O	4
U	2
R	6
H	1
E	0
A	3
T	8

Cryptarithmetic Puzzles

FORTY	Solution:	29786	F=2, O=9, R=7, etc.
+ TEN		850	
+ TEN		850	
---------		------	
SIXTY		31486	



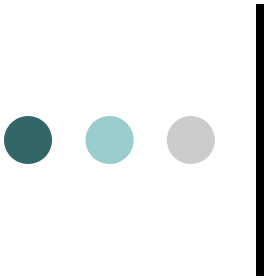
Character	Code
G	1
B	7
A	4
M	9
S	8
L	5
E	3

Measuring Searching Performance

•The output from problem-solving (searching) algorithm is either FAILURE or SOLUTION.

•Four ways:

- **Completeness** : is guaranteed to find a solution?
- **Optimality**: does it find optimal solution ?
- **Time complexity**: how long?
- **Space complexity**: how much memory?
  - Complexity : branching factor (*b*), depth (*d*), and max. depth (*m*)



Thank You !