A* Search Algorithm

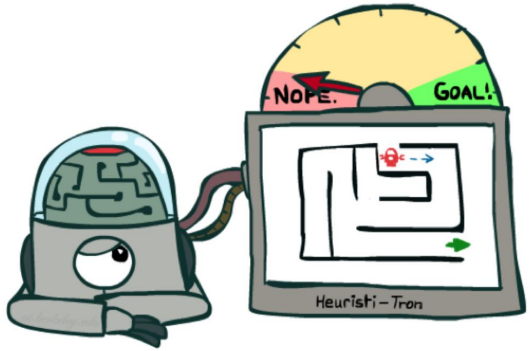
Dr. Nasima Begum
Assistant Professor
Dept. of CSE, UAP

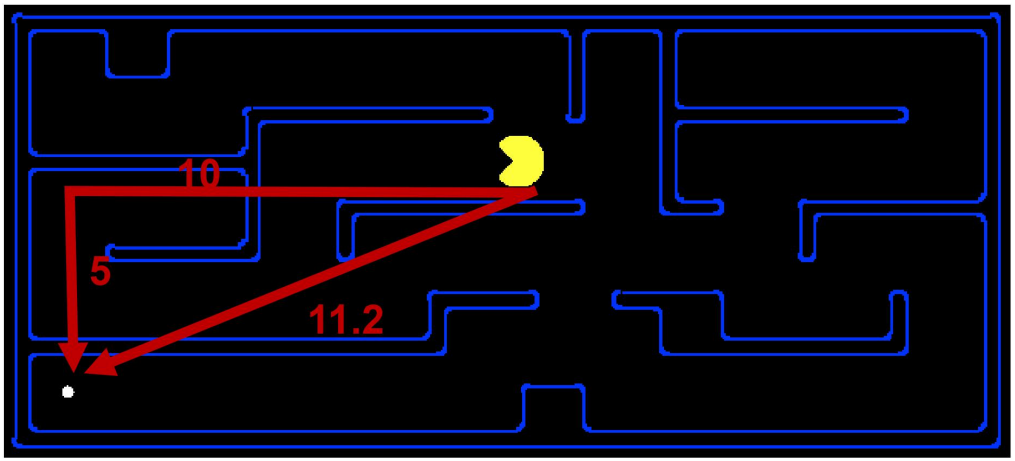
Heuristics Function

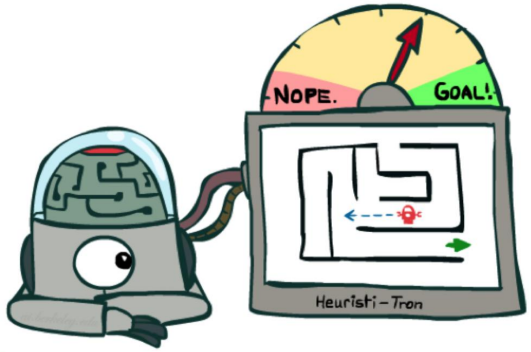
- Heuristic is a function which is used in Informed Search, and **it finds the most promising path**.
- It takes the current state of the agent as its input and produces the estimation of **how close the agent** is to the goal.
- Heuristic function estimates how close a state is to the goal state.
- It is represented by **h(n)**, and it calculates the cost of an optimal path between the pair of states. The value of the heuristic function is always positive.

Search Heuristics

- A heuristic is:
  - A function that *estimates* how close a state is to a goal
  - Designed for a particular search problem
  - Examples: Manhattan distance, Euclidean distance for pathing





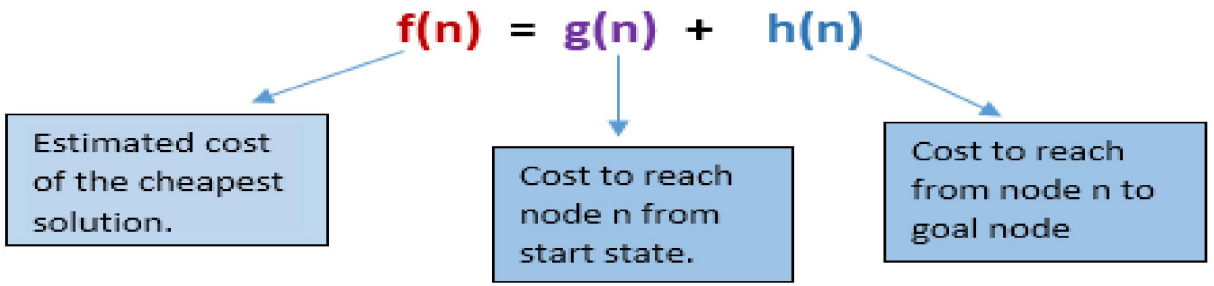


Heuristics Function

- Search Heuristics: In an informed search, a heuristic is a function that estimates how close a state is to the goal state.
- For examples – Manhattan distance, Euclidean distance, etc. (lesser the distance, closer the goal).
- Admissibility of the heuristic function is given as:
  0 <= h(n) <= h*(n)
- Here **h(n) is heuristic cost**, and **h*(n) is the estimated cost**. Hence heuristic cost should be less than or equal to the estimated cost (later).

A* Search

- It combines the strengths of UCS and greedy best-first search, by which it solve the problem efficiently.
- Here, the heuristic is the summation of the cost in UCS, denoted by g(n), and the cost in greedy search, denoted by h(n). The summed cost is denoted by f(n).
- Hence we can combine both costs as following, and this sum is called as a fitness number.



- A* search algorithm expands less search tree and provides optimal result faster.

A* Search

- Idea: avoid expanding paths that are already expensive, but expands most promising paths first.

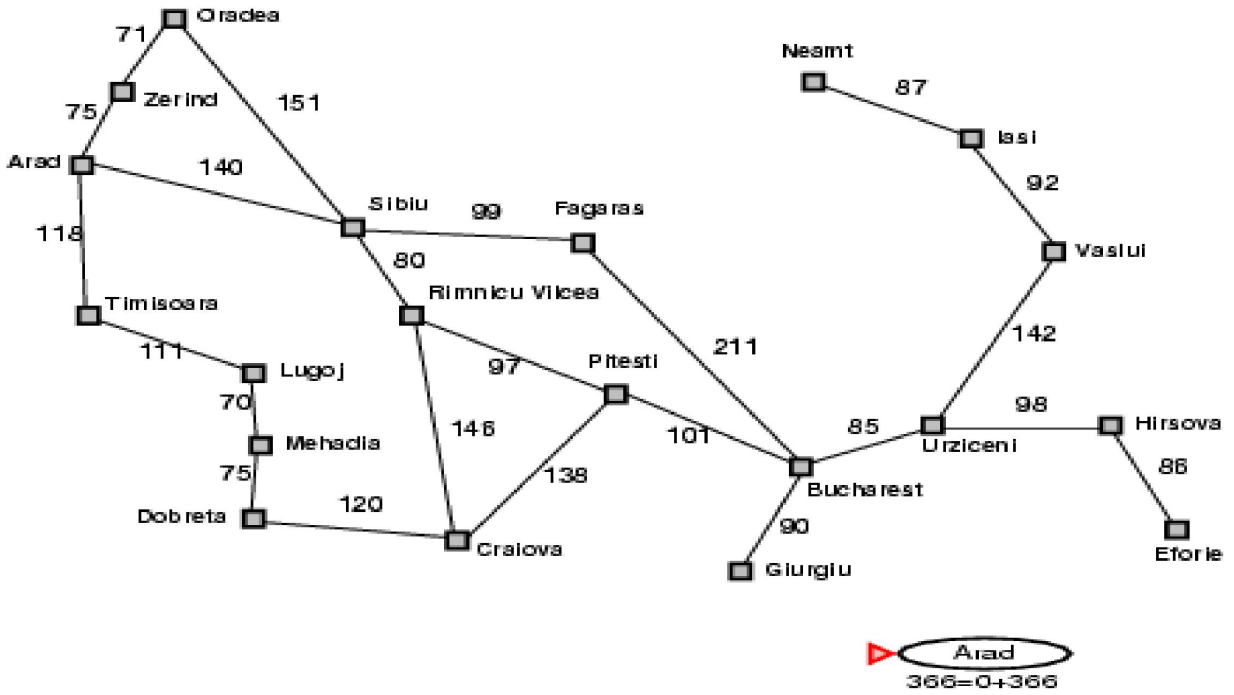
- Evaluation function $f(n) = g(n) + h(n)$

- $g(n)$ = Actual cost to reach n
- $h(n)$ = Estimated cost from n to goal
- $f(n)$ = Estimated total cost of path through n to goal

Algorithm of A* Search

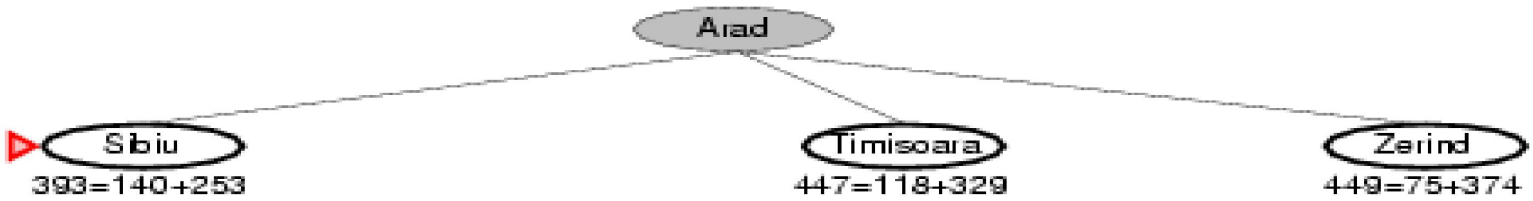
- Step1: Place the starting node in the OPEN list.
- Step 2: Check if the OPEN list is empty or not, if the list is empty then return failure and stops.
- Step 3: Select the node from the OPEN list which has the smallest value of evaluation function (g+h), if node n is goal node then return success and stop, otherwise
- Step 4: Expand node n and generate all of its successors, and put n into the closed list. For each successor n', check whether n' is already in the OPEN or CLOSED list, if not then compute evaluation function for n' and place into Open list.
- Step 5: Else if node n' is already in OPEN and CLOSED, then it should be attached to the back pointer which reflects the lowest g(n') value.
- Step 6: Return to Step 2.

A* Search Example



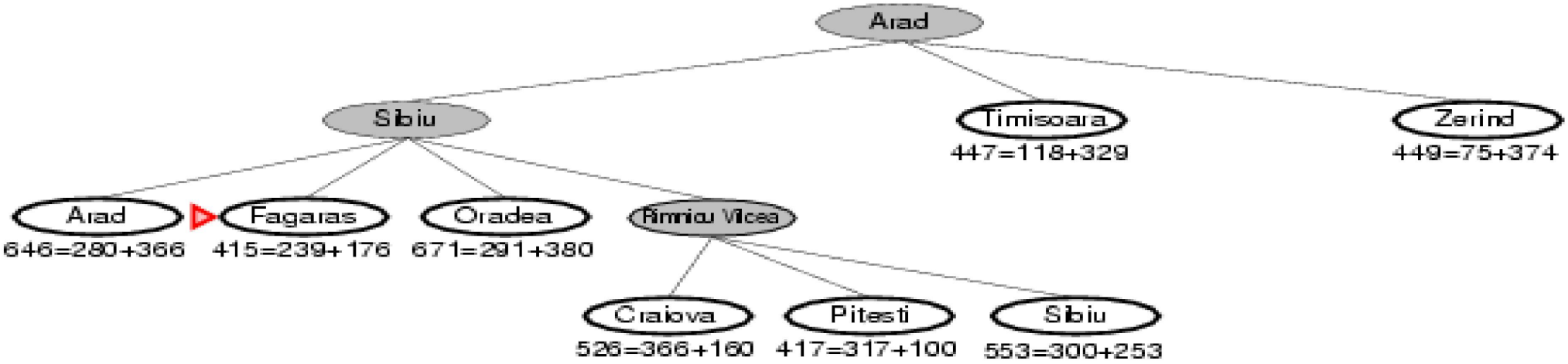
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* Search Example

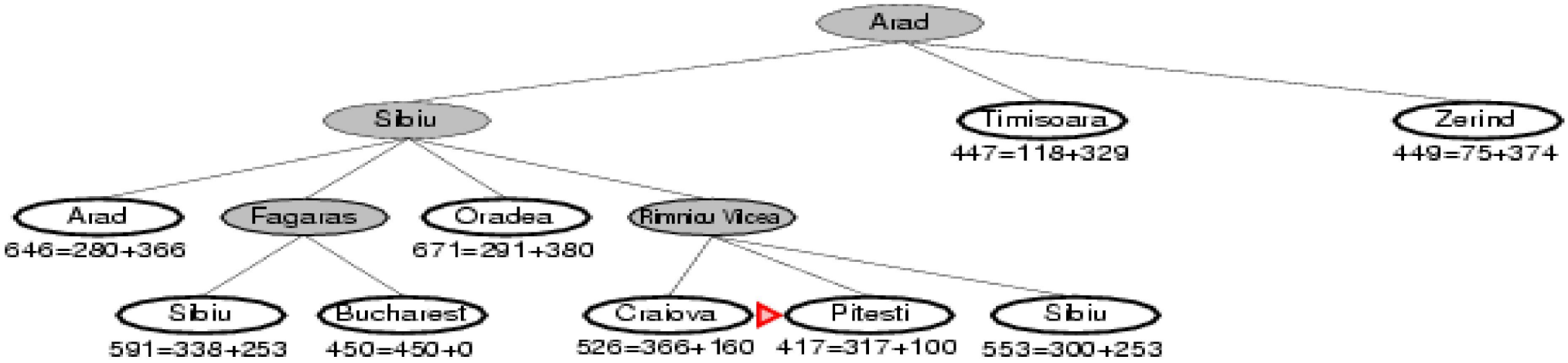


A* Search Example

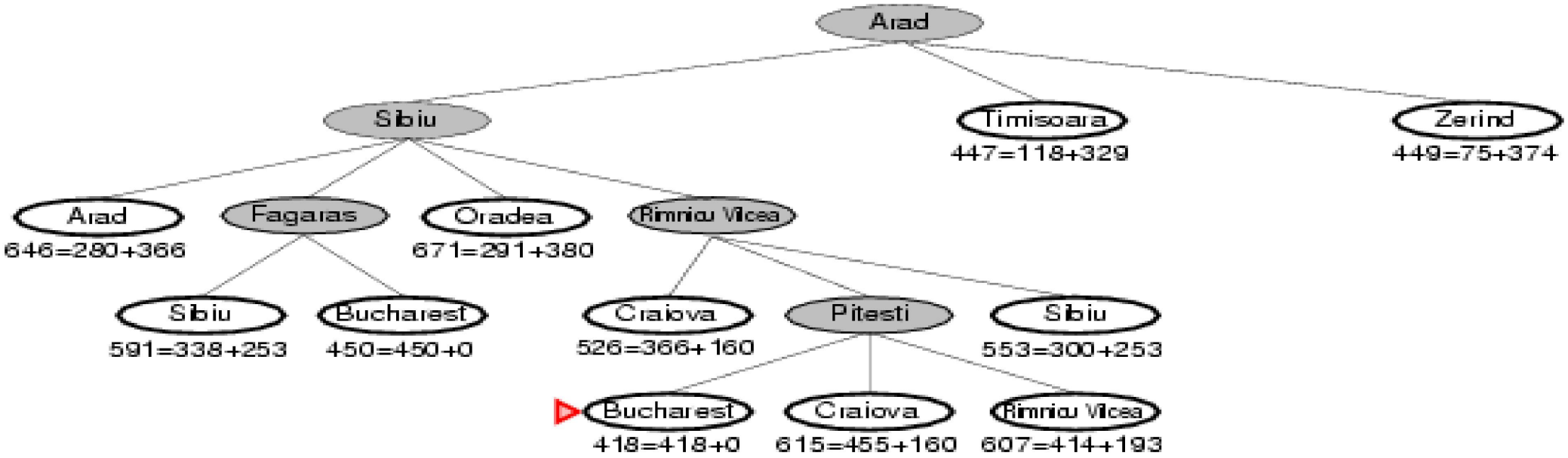
A* Search Example



A* Search Example

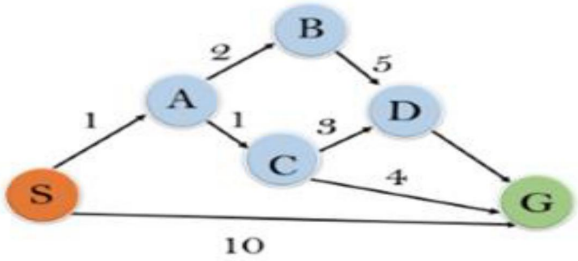


A* Search Example

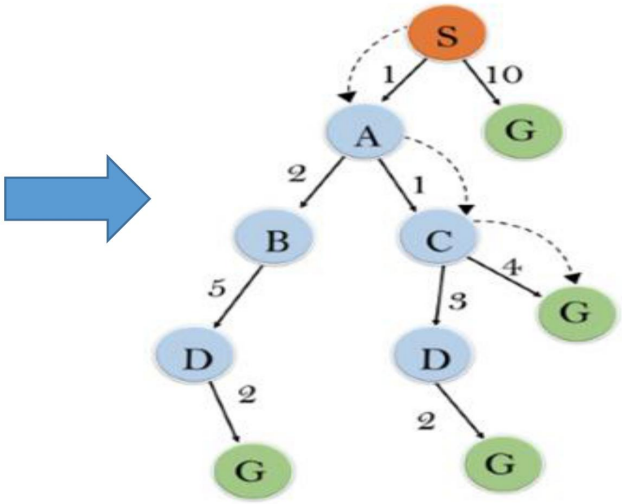


Finally return the path **A--->S--->R--->F--->P--->B**
it provides the optimal path with cost 2056.

A* Search Example



State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0



Initialization: {(S, 5)}
Iteration1: {(S--> A, 4), (S-->G, 10)}
Iteration2: {(S--> A-->C, 4), (S--> A-->B, 7), (S-->G, 10)}
Iteration3: {(S--> A-->C--->G, 6), (S--> A-->C--->D, 11), (S--> A-->B, 7),
(S-->G, 10)}
Iteration 4 will give the final result, as **S--->A--->C--->G** it provides the optimal path with cost 6.

Properties of A*

Points to remember:

-A* algorithm returns the path which occurred first, and it does not search for all remaining paths.

-The efficiency of A* algorithm depends on the quality of heuristic.

-A* algorithm expands all nodes which satisfy the condition f(n).

Complete? A* algorithm is complete as long as:

-Branching factor is finite.

-Cost at every action is fixed.

Optimal? Yes if it follows below two conditions:

-Admissible: h(n) should be an admissible heuristic for A* tree search. An admissible heuristic is optimistic in nature.

-Consistency: Second required condition is consistency for only A* graph-search.