Informed Search Strategies

Today we'll discuss

- Informed Search Strategies
- Types
- Greedy Search
- A* Search

Informed Search Strategies

- One that uses problem-specific knowledge beyond the definition of the problem itself
- It can find solutions more efficiently than an uninformed strategy
- In uninformed search, we don’t try to evaluate which of the nodes on the frontier are most promising. We never “look-ahead” to the goal

Informed Search Strategies

- Best-first search is an instance of the general tree search or graph search algorithm in which a node is selected for expansion based on an evaluation function, f(n).
- The evaluation function is construed as a cost estimate, so the node with the lowest evaluation is expanded first

Informed Search Strategies

- Most best-first algorithms include as a component of f a heuristic function, denoted h(n)
- h(n) = estimated cost of the cheapest path from the state at node n to a goal state
- h(n) guesses the cost of getting to the goal from node n

Best First Search

- A node is selected for expansion based on evaluation function f(n)
- Node with lowest evaluation function is expanded first.
- The evaluation function must represent some estimate of the cost of the path from state to the closest goal state.

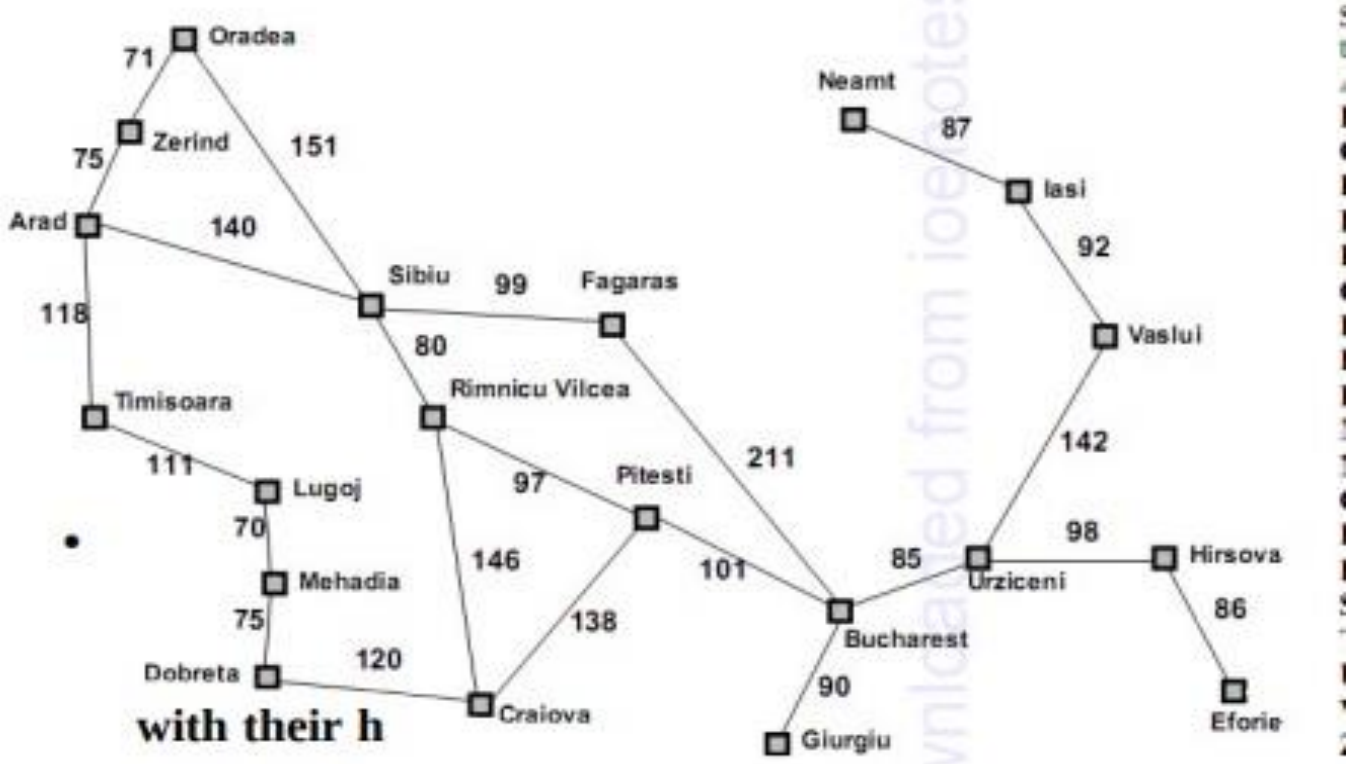
Types of Best First Search

- 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*

Greedy Best First Search



Greedy Best First Search

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Greedy Best First Search

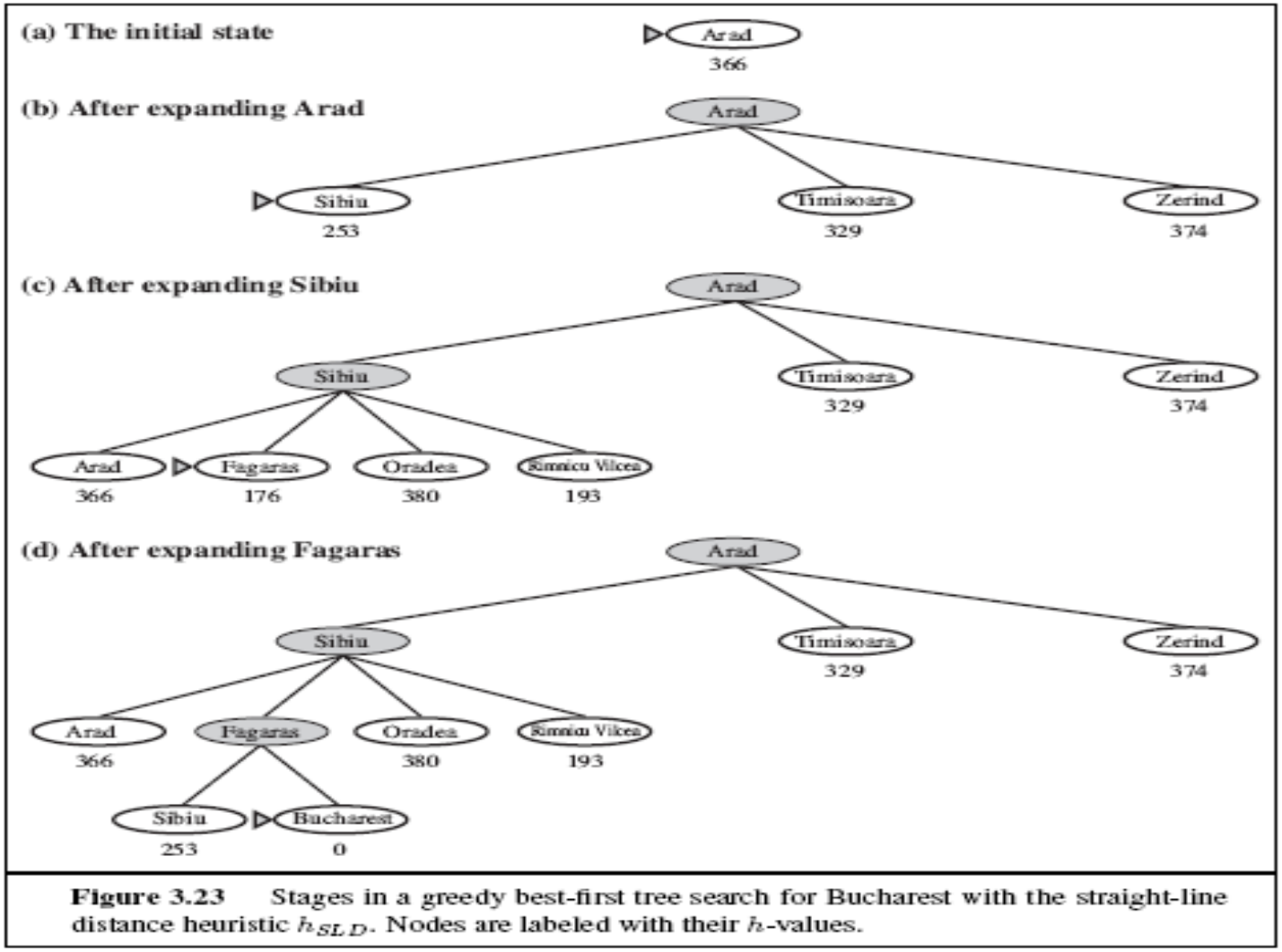
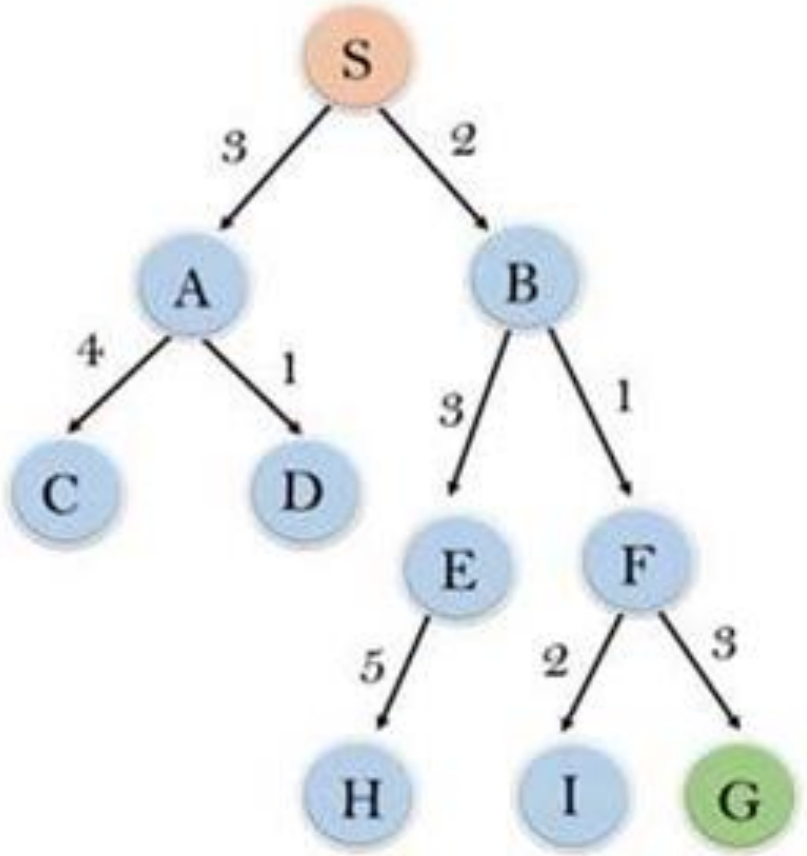


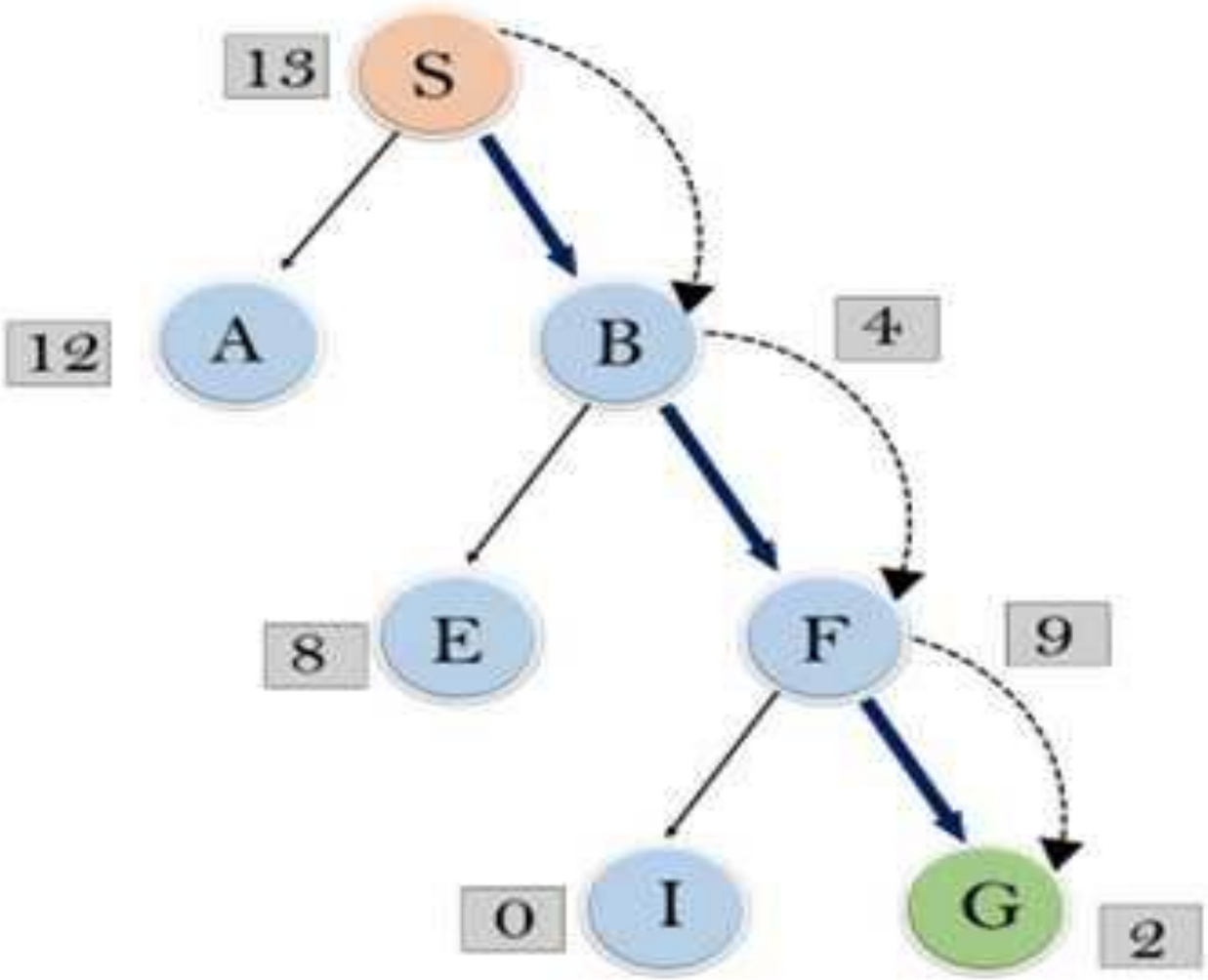
Figure 3.23 Stages in a greedy best-first tree search for Bucharest with the straight-line distance heuristic h_{SLD}. Nodes are labeled with their h-values.

Best first search algorithm:

- Step 1: Place the starting node into the OPEN list.
- Step 2: If the OPEN list is empty, Stop and return failure.
- Step 3: Remove the node n, from the OPEN list which has the lowest value of h(n), and places it in the CLOSED list.
- Step 4: Expand the node n, and generate the successors of node n.
- Step 5: Check each successor of node n, and find whether any node is a goal node or not. If any successor node is goal node, then return success and terminate the search, else proceed to Step 6.
- Step 6: For each successor node, algorithm checks for evaluation function f(n), and then check if the node has been in either OPEN or CLOSED list. If the node has not been in both list, then add it to the OPEN list.
- Step 7: Return to Step 2.



node	H (n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0


13
S
12
A
B
4
8
E
F
9
0
I
G
2

A* search

- Greedy best first search analyses nodes with lowest cost of node n to reach goal. and it may get stuck in search of goal
- The most widely known form A of best-first search is called A$*$ search
- It evaluates nodes by combining g(n), the cost to reach the node, and h(n), the cost to get from the node to the goal

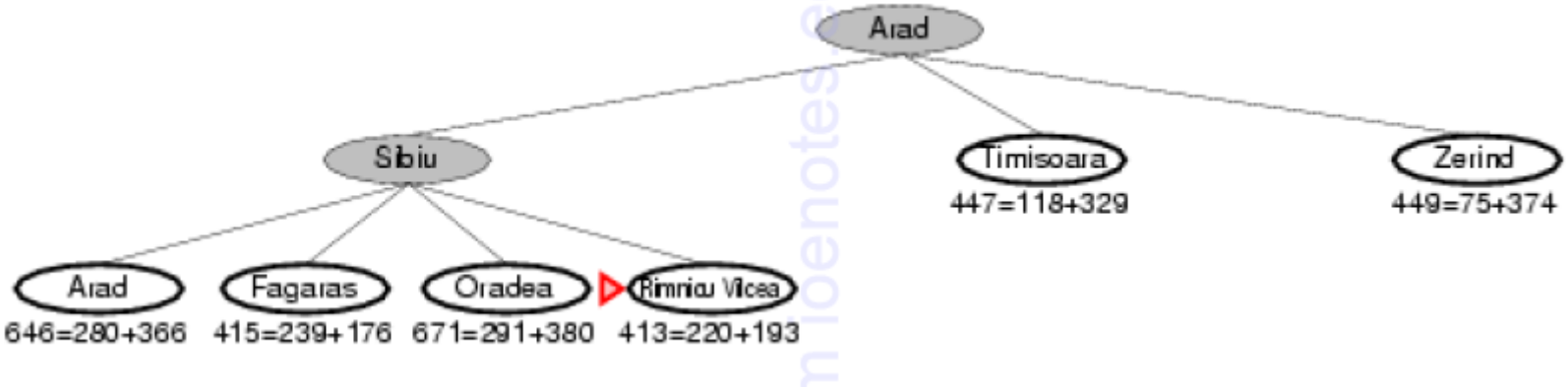
A* search

- f(n) = g(n) + h(n) .
- Since g(n) gives the path cost from the start node to node n, and h(n) is the estimated cost of the cheapest path from n to the goal, we have
- f(n) = estimated cost of the cheapest solution through n

A* search

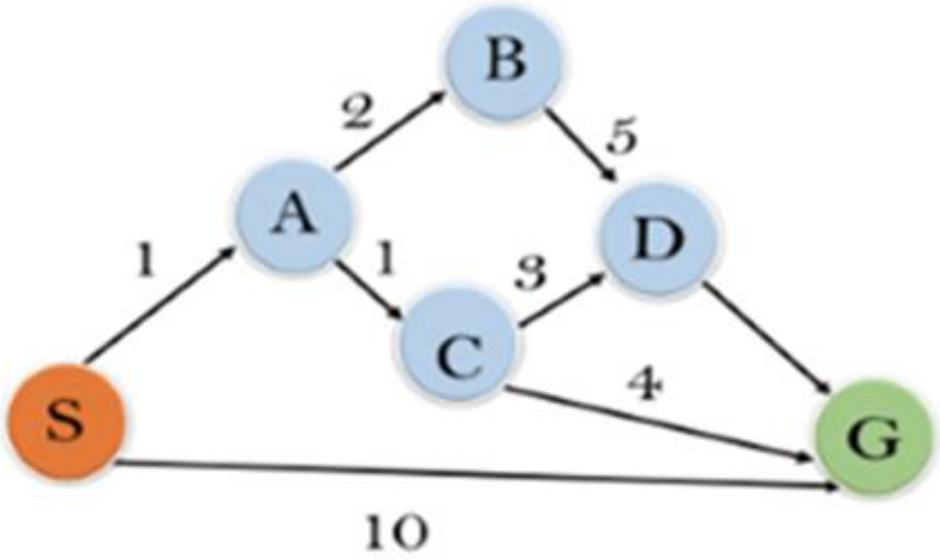
Start with source node

A* search


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vicea
413=220+193

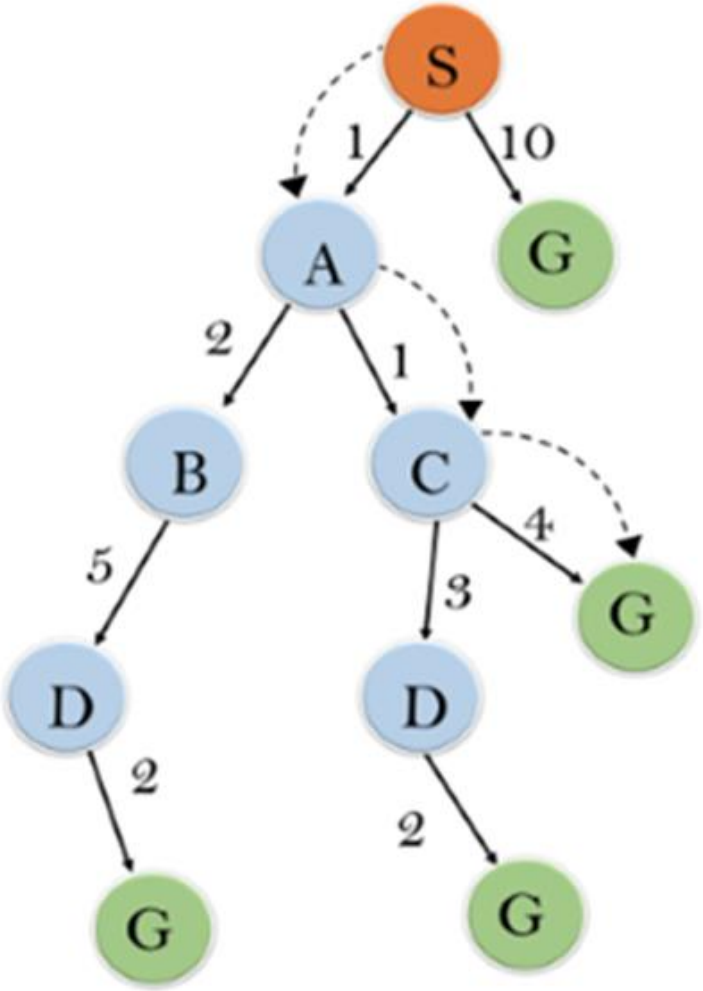
Algorithm of A* search:

- **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**.

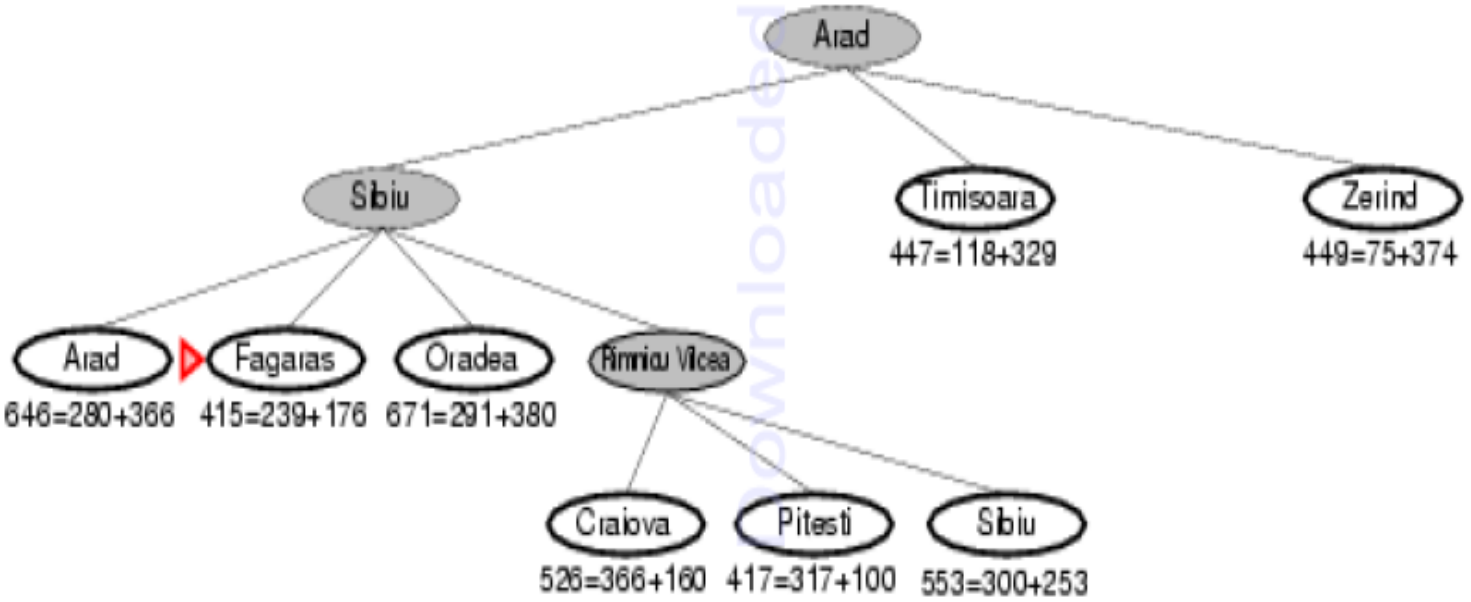


State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

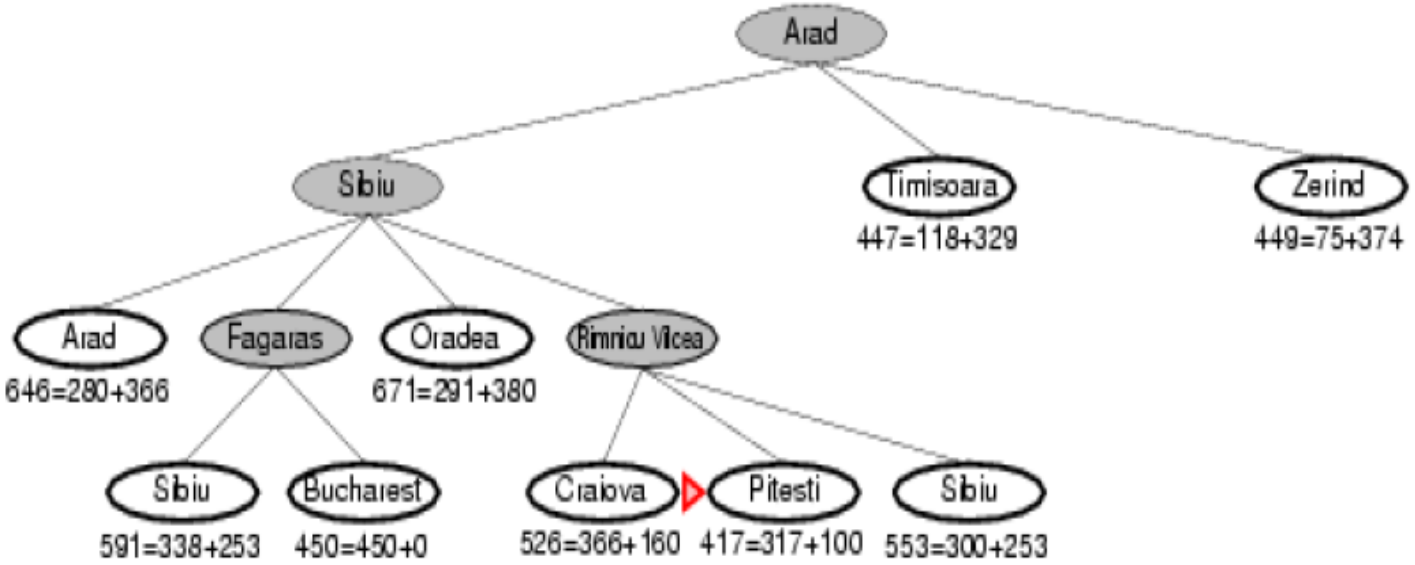
Solution



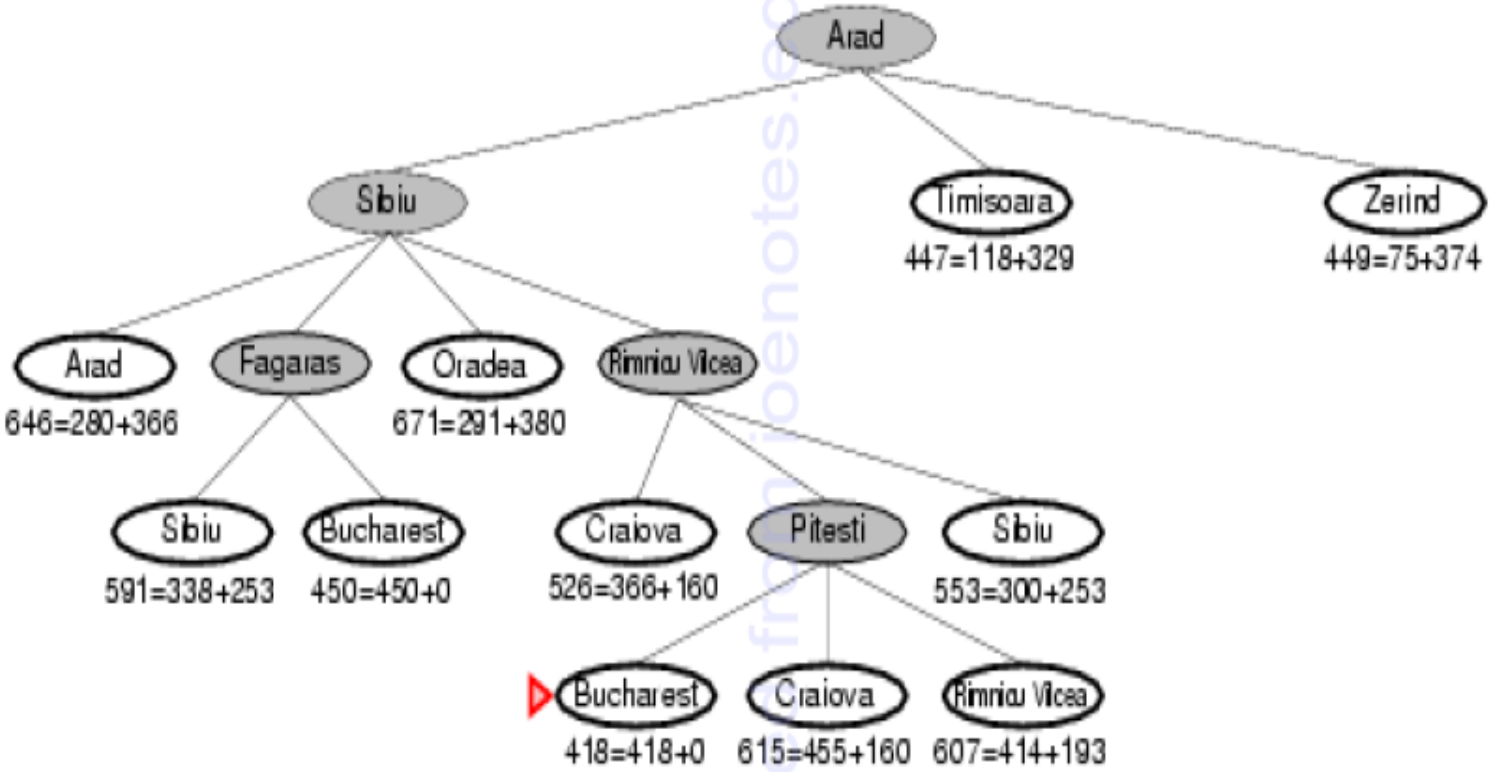
A* search


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vilcea
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253

A* search


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253

A* search


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
Sibiu
553=300+253
Bucharest
418=418+0
Craiova
615=455+160
Rimnicu Vilcea
607=414+193