USMAN INSTITUTE OF TECHNOLOGY

Department of Computer Science
CS321 Artificial Intelligence

Lab# 04
Optimal Search

Objective:

This experiment demonstrates the use of A* as an efficient Search Strategy. It also develops the concept of admissible and un-admissible heuristics.

Name of Student: _____

Roll No: _____ **Sec.** _____

Date of Experiment: _____

Marks Obtained/Remarks: _____

Signature: _____

Lab 05: Optimal and Adversarial Search

A* Search

A* is an informed search algorithm, meaning that it is formulated in terms of weighted graphs: starting from a specific starting node of a graph, it aims to find a path to the given goal node having the smallest cost (least distance travelled, shortest time, etc.). It does this by maintaining a tree of paths originating at the start node and extending those paths one edge at a time until its termination criterion is satisfied.

At each iteration of its main loop, A* needs to determine which of its paths to extend. It does so based on the cost of the path and an estimate of the cost required to extend the path all the way to the goal. Specifically, A* selects the path that minimizes $f(n) = g(n) + h(n)$, where n is the next node on the path, $g(n)$ is the cost of the path from the start node to n, and $h(n)$ is a heuristic function that estimates the cost of the cheapest path from n to the goal.

A* terminates when the path it chooses to extend is a path from start to goal or if there are no paths eligible to be extended. The heuristic function is problem-specific. If the heuristic function is admissible, meaning that it never overestimates the actual cost to get to the goal, A* is guaranteed to return a least-cost path from start to goal.

Admissibility Measures:

Using the evaluation function $f(n) = g(n) + h(n)$ that was introduced in the last section, we may characterize a class of admissible heuristic search strategies. If n is a node in the state space graph, $g(n)$ measures the depth at which that state has been found in the graph, and $h(n)$ is the heuristic estimate of the distance from n to a goal. In this sense $f(n)$ estimates the total cost of the path from the start state through n to the goal state. In determining the properties of admissible heuristics, we define an evaluation function f*:

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

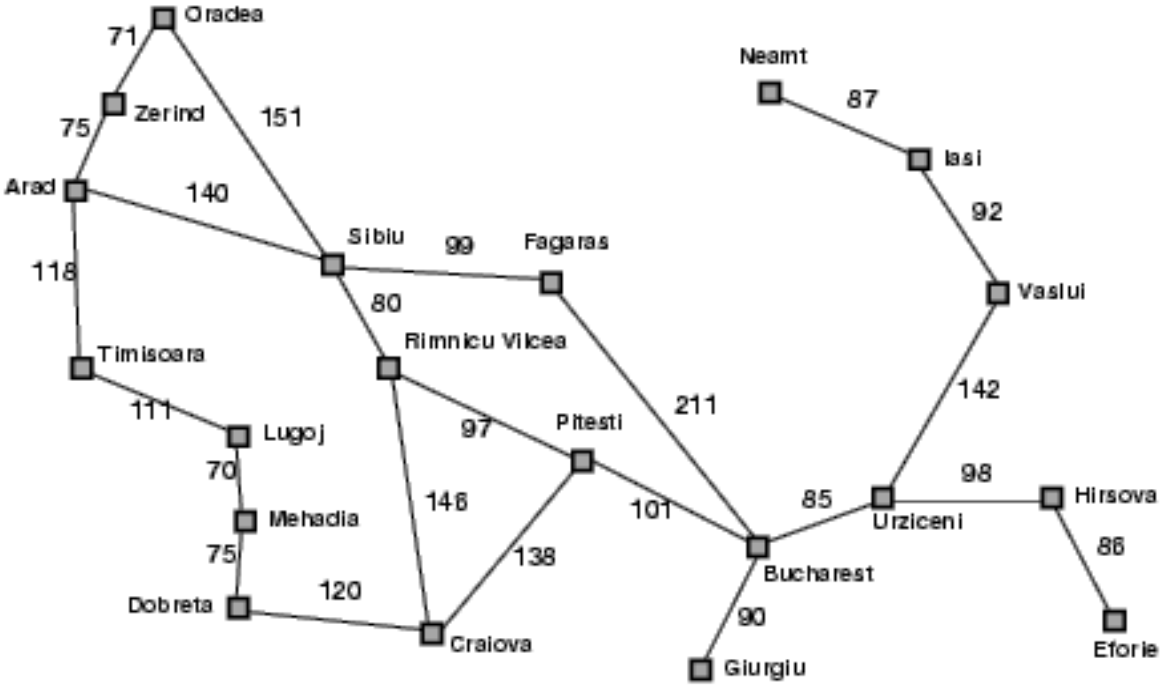
where $g^*(n)$ is the cost of the shortest path from the start to node n and h* returns the actual cost of the shortest path from n to the goal. It follows that $f^*(n)$ is the actual cost of the optimal path from a start node to a goal node that passes through node n. The resulting search strategy is admissible. Although oracles such as f* do not exist for most real problems, we would like the evaluation function f to be a close estimate of f*. In algorithm A, $g(n)$, the cost of the current path to state n, is a reasonable estimate of g*, but they may not be equal: $g(n) \geq g^*(n)$. These are equal only if the graph search has discovered the optimal path to state n.

Similarly, we replace h*(n) with h(n), a heuristic estimate of the minimal cost to a goal state. Although we usually may not compute h*, it is often possible to determine whether or not the heuristic estimate, h(n), is bounded from above by h*(n), i.e., is always less than or equal to the actual cost of a minimal path. If algorithm A uses an evaluation function f in which h(n) h*(n), it is called algorithm A*.

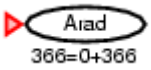
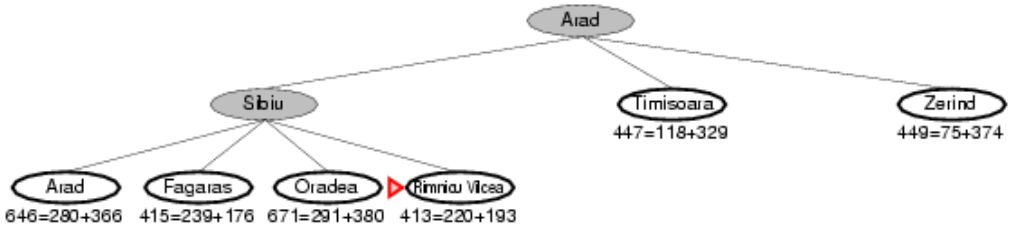
Typical implementations of A* use a priority queue to perform the repeated selection of minimum (estimated) cost nodes to expand. This priority queue is known as the open set or fringe. At each step of the algorithm, the node with the lowest f(x) value is removed from the queue, the f and g values of its neighbors are updated accordingly, and these neighbors are added to the queue. The algorithm continues until a goal node has a lower f value than any node in the queue (or until the queue is empty).

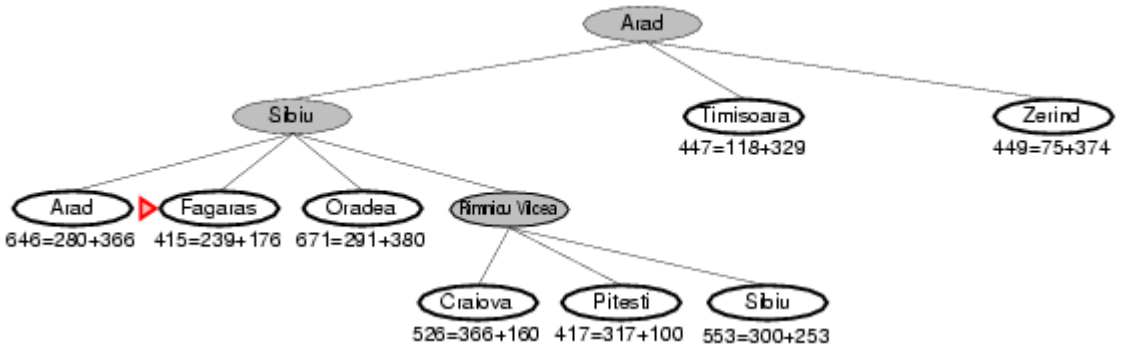
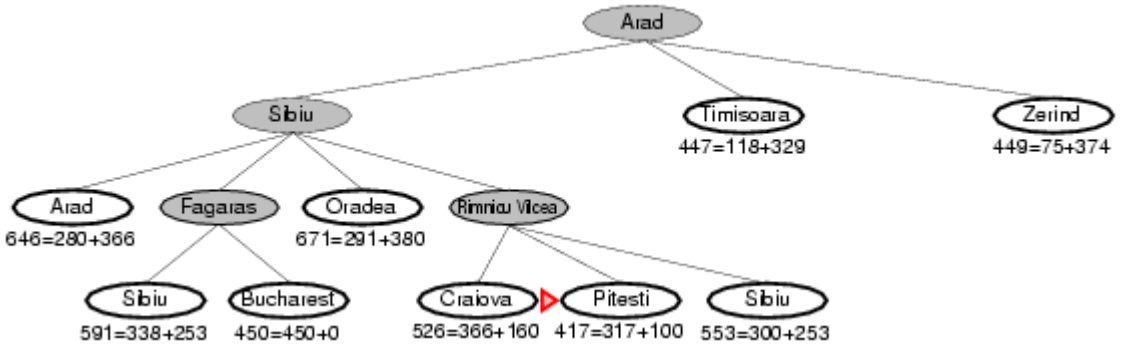
Pseudo Code for A*

- Input: – QUEUE: Path only containing root
- Algorithm: –
    - WHILE (QUEUE not empty && first path not reach goal) DO
        - Remove first path from QUEUE
        - Create paths to all children
        - Reject paths with loops
        - Add paths and sort QUEUE (by f = cost + heuristic)
        - IF QUEUE contains paths: P, Q
            - AND P ends in node Ni && Q contains node Ni AND
            - cost_P ≥ cost_Q
        - THEN remove P
    - IF goal reached THEN success ELSE failure



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

Step #	Fringe	Explored
1.	**Arad**	
2.	**1.Sibiu 2.Timisoara 3.Zerind**	
3.	**1. Rimnicu Vilcea 2. Fagaras 3.Timisoara 4.Zerind 5. Arad 6. Oradea**	

4.	1. Fagaras 2. Pitesti 3.Timisoara 4.Zerind 5. Craiova 6. Sibiu 7. Arad 8. Oradea	
5.	1. Pitesti 2.Timisoara 3.Zerind 4. Bucharest 5. Craiova 6. Sibiu(553) 7. Sibiu(591) 8. Arad 9. Oradea	

Student Exercise

Task 1

For the following graph and heuristic values program the variants of search algorithms mentioned below.

Graph	Heuristic
'A': {'B':9, 'C':4, 'D':7}	'A':21
'B': {'A':9, 'E':11}	'B':14
'C': {'A':4, 'E':17, 'F':12}	'C':18
'D': {'A':7, 'F':14}	'D':18
'E': {'B':11, 'C':17, 'Z': 5}	'E':5
'F': {'C':12, 'D':14, 'Z': 9}	'F':8
	'Z':0

- **Uniform Cost Search**
- **Best First Search**
- **A* Search**

Task 2

Develop code to implement the A* algorithm in order to find the optimal path in the Travel in Romania problem. Use the heuristic given in the text above. Furthermore, propose an admissible heuristic of your own and compare the two heuristics utilized. Suggest which of the two heuristics is a better choice for the travel in Romania problem and why?