Lab 8

Introduction

A* Search algorithm is one of the best and popular technique used in path-finding and graph traversals. The idea is avoid expanding nodes which are expansive.

1. Objective of the Experiment

After completing this lab, the student should be able to:

- Clearly understand difference between uninformed searches and informed searches.
- Better understanding of heuristic function.
- Understand search mechanism where cost is involved.

2. Concept Map

A search using domain-specific knowledge. Suppose that we have a way to estimate how close a state is to the goal, with an evaluation function.

3.1 A* Search

Idea behind A* search is to avoid expanding paths that are already expensive. Cost of node **"n"** is calculated using following function:

Evaluation function: f(n) = g(n)+h(n).

Where g(n) is the actual cost to reach **n** node from initial state, h(n) is estimated cost from n to goal. h(n) is also called heuristic function. f(n) is the estimated cost of path through initial to goal through n. So A* has f(n) = g(n) + h(n), whereas greedy best first search has f(n) = h(n) and uniform cost search has f(n) = g(n).

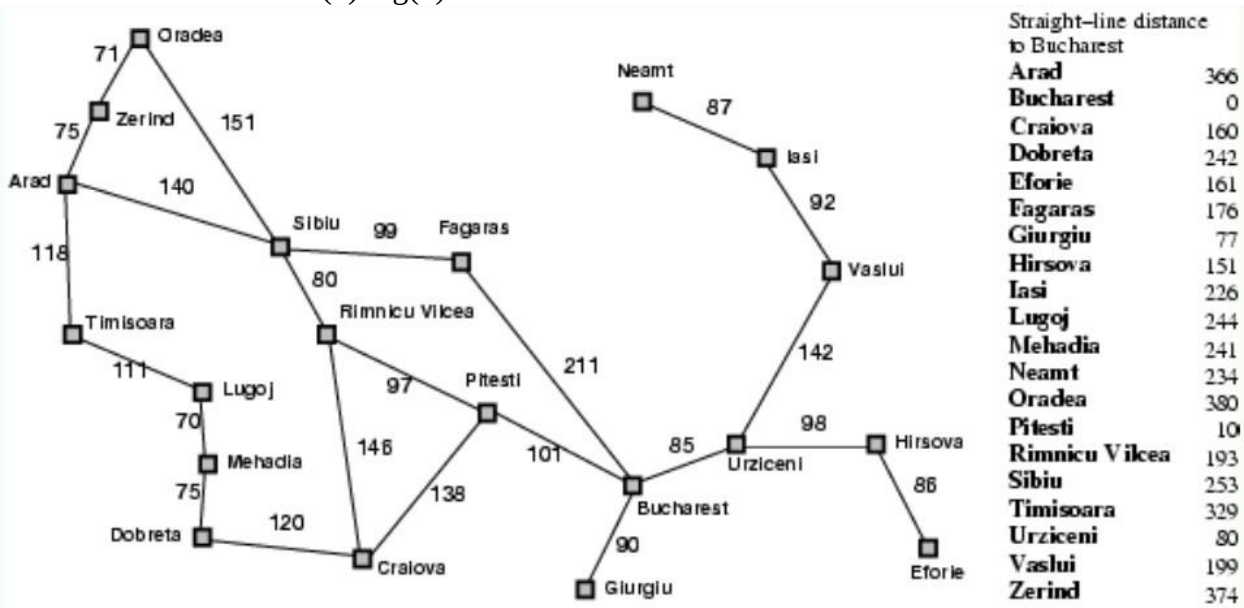
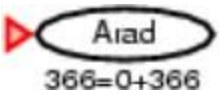
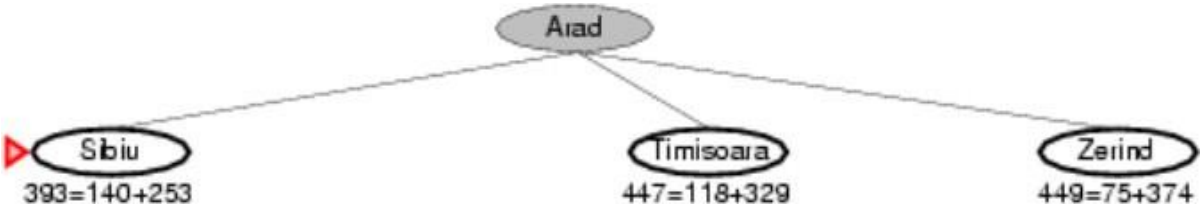


Figure 3 Bucharest Map

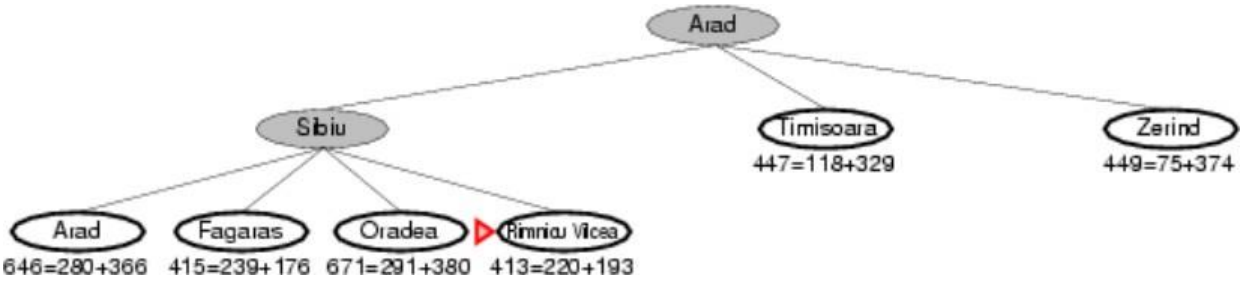
Let's run A* algorithm on above problem.



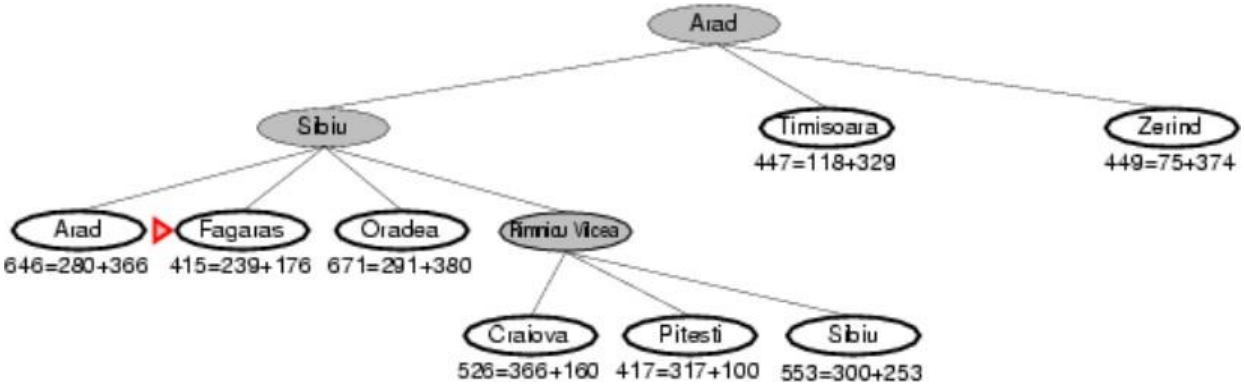
Add Arad in close list. As Arad is not the goal state, expand its childs.



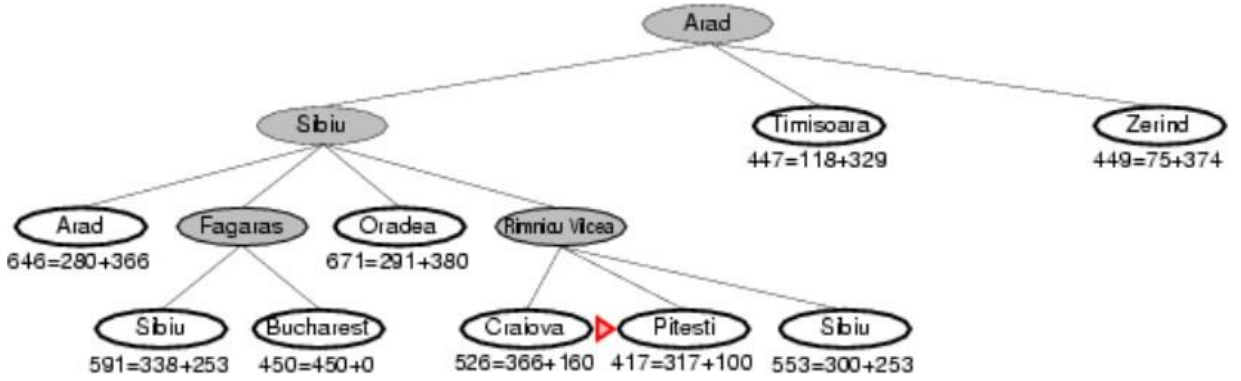
Continue the process until goal node is reached.



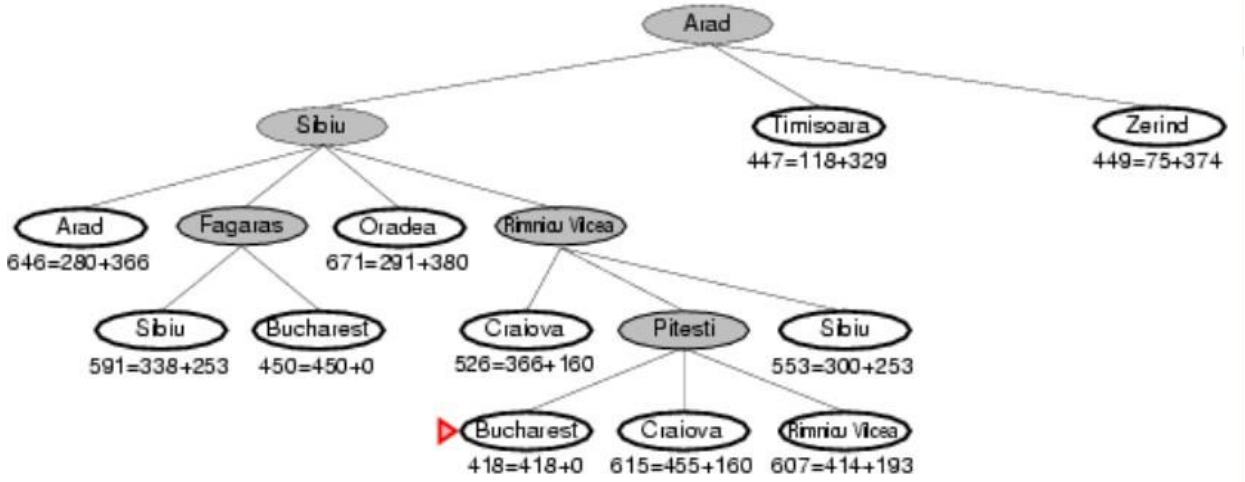
Goal state still not found, continue…



Now f(n) of Fagaras seems shorter than others, maybe we can find better solution through Fagarus.



Now Pitesti seems closer to goal.



Finally we reached the goal.

3. Procedure & Tools

4.1 Walkthrough Task

We maintain two lists: OPEN and CLOSE:
OPEN consists on nodes that have been visited but not expanded (meaning that sucessors have not been explored yet). This is the list of pending tasks.
CLOSE consists on nodes that have been visited and expanded (sucessors have been explored already and included in the open list, if this was the case).

Pseudocode of A* Search.

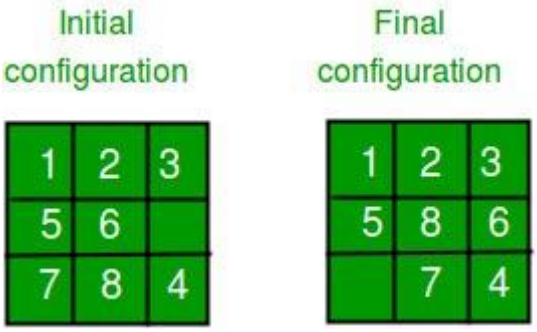
The goal node is denoted by node_goal and the source node is denoted by node_start.

```
1 Put node_start in the OPEN list with f(node_start) = h(node_start) (initialization)
 2 while the OPEN list is not empty {
      3 Take from the open list the node node_current with the lowest
      4 f(node_current) = g(node_current) + h(node_current)
      5 if node_current is node_goal we have found the solution; break
      6 Generate each state node_successor that come after node_current
          7 for each node_successor of node_current {
             8 Set successor_current_cost = g(node_current) + w(node_current, node_successor)
                    9 if node_successor is in the OPEN list {
                    10 if g(node_successor) ≤ successor_current_cost continue (to line 20) }
                    11  else if node_successor is in the CLOSED list {
                        12 if g(node_successor) ≤ successor_current_cost continue (to line 20)
                              13 Move node_successor from the CLOSED list to the OPEN list }
                              14 else {
                                        15 Add node_successor to the OPEN list
                                        16 Set h(node_successor) to be the heuristic distance to node_goal
                                        17 }
                                        18 Set g(node_successor) = successor_current_cost
                                        19 Set the parent of node_successor to node_current
                                        20 }
                                        21 Add node_current to the CLOSED list
                                        22 }
```

23 if(node_current != node_goal) exit with error (the OPEN list is empty)

8 Puzzle Problem

This problem can be solved by searching for a solution, which is a sequence of actions (tile moves) that leads from the initial state to the goal state. Two possible states of the 8-puzzle are shown in figure. The state on the right is a typical goal state.



Tasks

4.1 Task 1

Implement greedy best first search in Python.

4.2 Task 2

Implement the A* search in python.

4.3 Task 3

Implement the 8 Puzzle Problem using A* search in python.

The puzzle consists of an area divided into a grid, 3 by 3 for the 8-puzzle. On each grid square is a tile, expect for one square which remains empty. Thus, there are eight tiles in the 8-puzzle. A tile that is next to the empty grid square can be moved into the empty space, leaving its previous position empty in turn. Tiles are numbered, 1 thru 8 for the 8-puzzle, so that each tile can be uniquely identified. Heuristic for 8-puzzle problem is Number of Misplaced Tiles. You can design your own heuristic.

4. Further Readings

5.1 Books

- Artificial Intelligence: A modern approach 3rd Edition by Stuart Russell & Peter Norvig

5.2 Slides

The slides and reading material can be accessed from the folder of the class instructor.

5.3 Links

https://www.tutorialspoint.com/prolog_in_artificial_intelligence/index.asp

http://lpn.swi-prolog.org/lpnpage.php?pageid=online
https://www.geeksforgeeks.org/a-search-algorithm/
https://www.geeksforgeeks.org/best-first-search-informed-search/