|  |
| --- |
| Department of Software Engineering  Mehran University of Engineering and Technology, Jamshoro |

|  |  |  |  |
| --- | --- | --- | --- |
| Course: SWE – Artificial intelligence concepts and techniques | | | |
| Instructor | Eman Shahid | **Practical/Lab No.** | 04 |
| Date |  | **CLOs** | 03 |
| Signature |  | **Assessment Score** |  |

|  |  |
| --- | --- |
| Topic | To become familiar with informed searched algorithms |
| Objectives |  |

|  |
| --- |
| Lab Discussion: Theoretical concepts and Procedural steps |

**Informed Search Strategies**

**(Heuristic Search)**

* Best-first search
* Greedy Search
* Beam search
* Algorithm A
* Algorithm A\*
* Hill climbing

**Heuristic function**

[dictionary]*“A rule of thumb, simplification, or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood.”*

* *h(n)* = estimated cost of the cheapest path from node *n* to goal node.
* If *n* is goal then *h(n)=0*

**A\* search**

* Best-known form of best-first search.
* Idea: avoid expanding paths that are already expensive.
* Evaluation function *f(n)=g(n) + h(n)*
  + *g(n)* the cost (so far) to reach the node.
  + *h(n)* estimated cost to get from the node to the goal.
  + *f(n)* estimated total cost of path through *n* to goal.
* A\* search uses an admissible heuristic
* A heuristic is admissible if it *never overestimates* the cost to reach the goal
* It is optimistic

Formally:

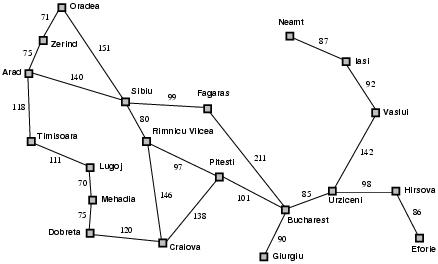
*h(n) <= h\*(n)* where *h\*(n)* is the true cost from *n*

*h(n) >= 0* so *h(G)>= 0* for any goal *G*.

e.g. *hSLD(n)* never overestimates the actual road distance

**Romania with step costs in km**



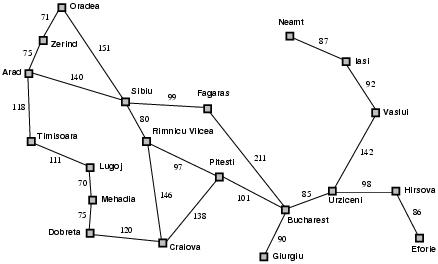


* *hSLD*=straight-line distance.
* *hSLD* can **NOT** be computed from the problem description itself

In this example *f(n)=h(n)*

Expand node that is closest to goal

**Romania example**

****

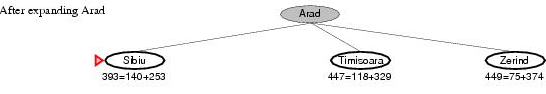
**A\* search example**

**Find Bucharest starting at Arad**

****

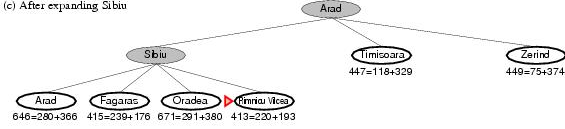
f(Arad) = g(??,Arad)+h(Arad)=0+366=366

**Expand Arrad and determine *f(n)* for each node**



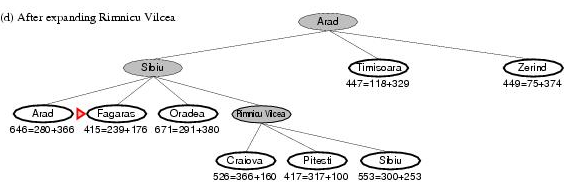
* f(Sibiu)=g(Arad,Sibiu)+h(Sibiu)=140+253=393
* f(Timisoara)=g(Arad,Timisoara)+h(Timisoara)=118+329=447
* f(Zerind)=g(Arad,Zerind)+h(Zerind)=75+374=449
* Best choice is Sibiu

**Expand Sibiu and determine *f(n)* for each node**

****

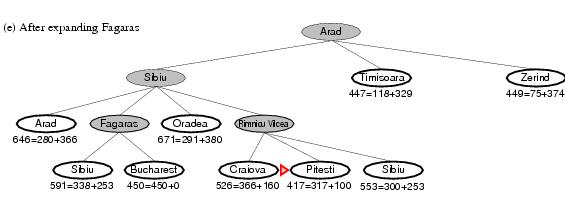
* f(Arad)=g(Sibiu,Arad)+h(Arad)=280+366=646
* f(Fagaras)=g(Sibiu,Fagaras)+h(Fagaras)=339+176=415
* f(Oradea)=g(Sibiu,Oradea)+h(Oradea)=291+380=671
* f(Rimnicu Vilcea)=g(Sibiu,Rimnicu Vilcea)+h(Rimnicu Vilcea)=220+192=413
* Best choice is Rimnicu Vilcea

**Expand Rimnicu Vilcea and determine *f(n)* for each node**

****

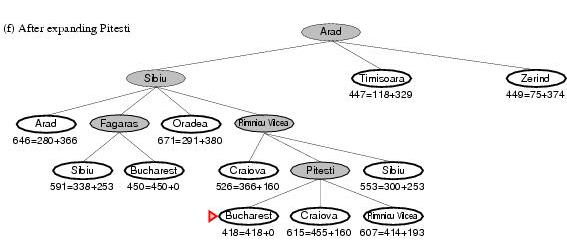
* f(Craiova)=c(Rimnicu Vilcea, Craiova)+h(Craiova)=360+160=526
* f(Pitesti)=c(Rimnicu Vilcea, Pitesti)+h(Pitesti)=317+100=417
* f(Sibiu)=c(Rimnicu Vilcea,Sibiu)+h(Sibiu)=300+253=553
* Best choice is Fagaras

**Expand Fagaras and determine *f(n)* for each node**

****

* f(Sibiu)=c(Fagaras, Sibiu)+h(Sibiu)=338+253=591
* *f(Bucharest)=c(Fagaras,Bucharest)+h(Bucharest)=450+0=450*
* Best choice is Pitesti !!!

**Expand Pitesti and determine *f(n)* for each node**

*f(Bucharest)=c(Pitesti,Bucharest)+h(Bucharest)=418+0=418*

* Best choice is Bucharest !!!
* Optimal solution (only if *h(n)* is admissable)
* Note values along optimal path !!
* The heuristics that we are using here is the straight-line distance from the city to the goal(Here, Bucharest). Note that, this straight line distance is obtained only by knowing the map coordinates of the 2 cities.
* Input
  + Input is taken from the file
  + Graph.txt
* Each line in the input is of the form
  + city1 city2 dis
* Heuristic
  + city dis

**Graph.txt**

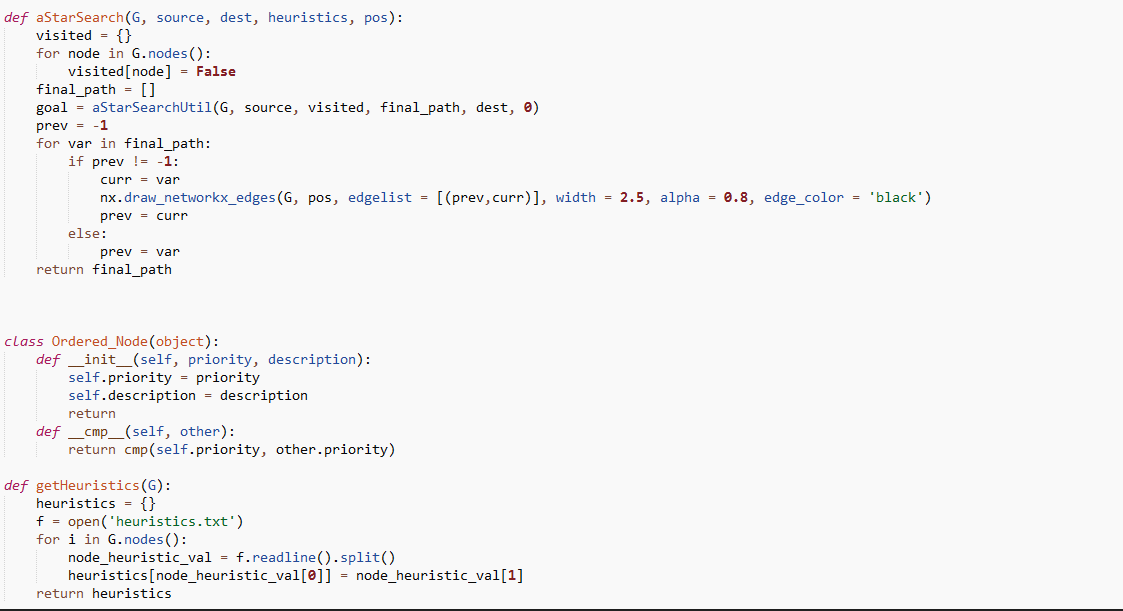


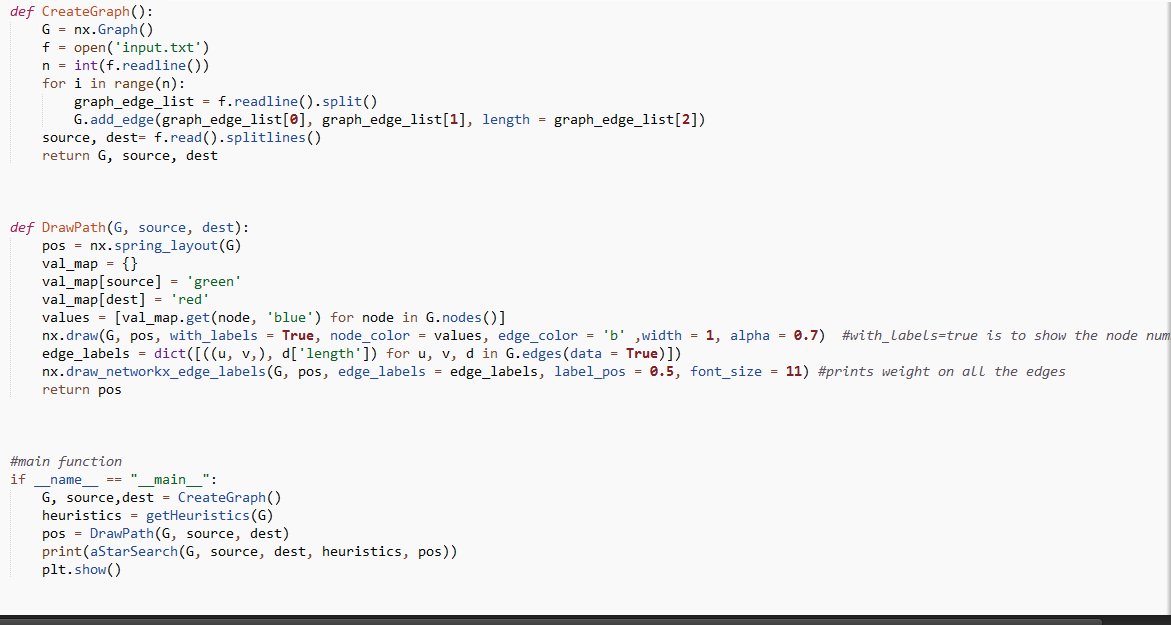
**Heuristic.txt**



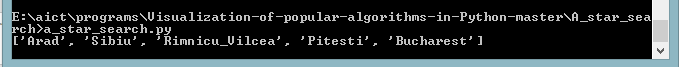
**Code:**

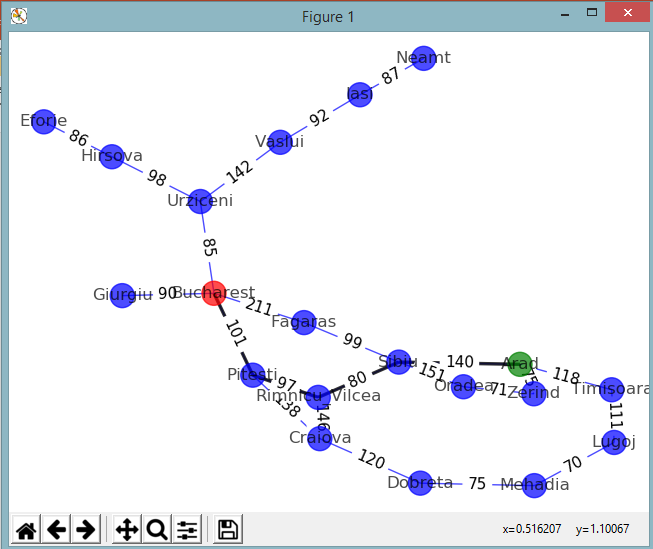






**Output**





|  |
| --- |
| Lab Tasks |
| Submission Date: 1st week after vacations |

Note: this code will be execute on python version 2.7

1. Execute the code for A\* provided in the handouts.
2. Execute the A\* algorithm on any map of your own choice.