**Lab 05**

**Search Problems in Artificial Intelligence**

**Informed Searches**

So far we have talked about the uninformed search algorithms which looked through search space for all possible solutions of the problem without having any additional knowledge about search space. But informed search algorithm **contains an array of knowledge** such as how far we are from the goal, path cost, how to reach to goal node, etc. This knowledge help agents to explore less to the search space and find more efficiently the goal node.

The informed search algorithm is more useful for large search space. Informed search algorithm uses the idea of heuristic, so it is also called Heuristic search.

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 agent is from the goal. The heuristic method, however, might not always give the best solution, but it guaranteed to find a good solution in reasonable time. Heuristic function estimates how close a state is to the goal. 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.

Admissibility of the heuristic function is given as:

**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.

Pure Heuristic Search:

Pure heuristic search is the simplest form of heuristic search algorithms. It expands nodes based on their heuristic value h(n). It maintains two lists, OPEN and CLOSED list. In the CLOSED list, it places those nodes which have already expanded and in the OPEN list, it places nodes which have yet not been expanded.

On each iteration, each node n with the lowest heuristic value is expanded and generates all its successors and n is placed to the closed list. The algorithm continues unit a goal state is found.

In the informed search we will discuss two main algorithms which are given below:

* Best First Search Algorithm(Greedy search)
* A\* Search Algorithm

### 1) Greedy Search

In greedy search, we expand the node closest to the goal node. The “closeness” is estimated by a heuristic h(x) .

Heuristic: A heuristic h is defined as-  
h(x) = Estimate of distance of node x from the goal node.  
Lower the value of h(x), closer is the node from the goal.

Strategy: Expand the node closest to the goal state, i.e. expand the node with lower h value.

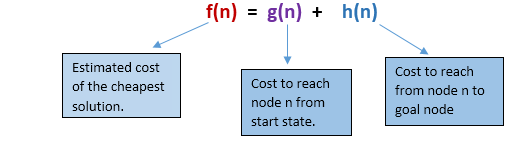
**Algorithm**

* 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.

### 2.) A\* Search Algorithm:

A\* search is the most commonly known form of best-first search. It uses heuristic function h(n), and cost to reach the node n from the start state g(n). It has combined features of UCS and greedy best-first search, by which it solve the problem efficiently. A\* search algorithm finds the shortest path through the search space using the heuristic function. This search algorithm expands less search tree and provides optimal result faster. A\* algorithm is similar to UCS except that it uses g(n)+h(n) instead of g(n).

In A\* search algorithm, we use search heuristic as well as the cost to reach the node. Hence we can combine both costs as following, and this sum is called as a fitness number.



### 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

**Now Let’s see the Examples of these Algorithms.. ( On Slides).**

**Lab Task 1**

**Problem**

Find an Optimal Path at lowest cost of time as well as path cost on **Map of Romania.**

**Code Provided:**

Provided code file named A\_Star.ipynb contains skeleton code which Connect the different cities of graph along with their cost

**self.graph\_dict** contains cost of each edge traversal of (u,v)

**Tasks to Perform:**

1**)** Create All Connections of Romania Map in Code with following Function

**graph.connect( Arad, Zerind, 75)**

2) Create and store all heuristics (Straight Line Distance To **Bucharest** in this case) Provided in Example in **Heuristics** Dictionary defined in code in following Fashion

**heuristics['Arad'] = 366**

3) Implement A-Star function on Graph (Map of Romania) to find optimal path from source to destination with the help of Heuristics provided.

4) Run the code given at end to see all paths and shortest path find by your A\_Start

**Lab Task 2**

Solve 8 puzzle problem using greedy best first search.