---

## Exercise 4 – Informed Search Strategies: Maze Application

**Date:** 15/09/2022

### Aim:

Problem statement: Initial state: (1, 1) – A, Goal state: (3, 3) – H Assume that the location (1,3) is not used, since a pit is available in that location. There is a possibility that the Robot may fall into the pit during navigation.

To compute and implement the level-by-level search using Greedy Best First Search and A* search from the given initial state to goal state.

### Code:

```
# Maze - Greedy and A* search strategies
h={}
goal=(3,3)
m=3
n=3

# (1,1)|(1,2)|(1,3)
# (2,1)|(2,2)|(2,3)
# (3,1)|(3,2)|(3,3)<---Goal

def man_dist(goal,node):
  d= abs(goal[0]-node[0]) + abs(goal[1] - node[1])
  return d

for i in range(m):
  for j in range(n):
    if (i,j)==(0,2):
      continue
    h[(i+1,j+1)]=man_dist(goal,(i+1,j+1))

print("Manhattan Distance")

for k,v in h.items():
  print(k," : ",v)
```

```
graph={
   (1,1,1,2):9,
   (1,2,1,1):9,
   (1,1,2,1):6,
   (2,1,1,1):6,
   (1,2,2,2):5,
   (2,2,1,2):5,
   (2,1,2,2):8,
   (2,2,2,1):8,
   (2,1,3,1):5,
   (3,1,2,1):5,
   (2,2,3,2):6,
   (3,2,2,2):6,
   (2,2,2,3):7,
   (2,3,2,2):7,
   (2,3,3,3):4,
   (3,3,2,3):4,
   (3,1,3,2):7,
   (3,2,3,1):7,
   (3,2,3,3):8,
   (3,3,3,2):8,

}

dirs=[(0,-1),(-1,0),(0,1),(1,0)]

def greedyBest():

  v=[(1,1)]

  state=(1,1)
  pathcost=0
  while(1):

    print(state,end=" ")

    if state==goal:
      break


    minh=100
    min_node=(1,1)
    for dx,dy in dirs:
```

```
    xx,yy=state[0]+dx,state[1]+dy
    if 1<=xx<=m and 1<=yy<=n and (xx,yy) not in v and (xx,yy)!=(1,3):


      if h[(xx,yy)]<minh:
        minh=h[(xx,yy)]
        min_node=(xx,yy)



  pathcost+=graph[state[0],state[1],min_node[0],min_node[1]]
  v.append(min_node)
  state=min_node

 print("\nPath Cost : ",pathcost)

def astar():


 v=[(1,1)]

 state=(1,1)
 pathcost=0
 while(1):

  print(state,end=" ")

  if state==goal:
    break


  minhg=100
  min_node=(1,1)
  for dx,dy in dirs:

    xx,yy=state[0]+dx,state[1]+dy
    if 1<=xx<=m and 1<=yy<=n and (xx,yy) not in v and (xx,yy)!=(1,3):

      gh= graph[(state[0],state[1],xx,yy)] + h[(xx,yy)]
      if gh<=minhg:
        minhg=gh
        min_node=(xx,yy)
```

```
    pathcost+=graph[state[0],state[1],min_node[0],min_node[1]]
    v.append(min_node)
    state=min_node

  print("\nPath Cost : ",pathcost)


#main

print("\nGreedy: ")
greedyBest()

print("\nA*: ")
astar()
```

**Output:**

```
~/AIwork$ python3 ex4maze.py
Manhattan Distance
(1, 1)  :  4
(1, 2)  :  3
(2, 1)  :  3
(2, 2)  :  2
(2, 3)  :  1
(3, 1)  :  2
(3, 2)  :  1
(3, 3)  :  0

Greedy:
(1, 1) (1, 2) (2, 2) (2, 3) (3, 3)
Path Cost :  25

A*:
(1, 1) (2, 1) (3, 1) (3, 2) (3, 3)
Path Cost :  26
~/AIwork$
```