Name:- Krishna Mundada

Roll NO:- 45

Section:- AI-ML

8 puzzle using A* algorithm

In [ ]:
```
# initial state
boardMain = [['2','8','3'],['1','6','4'],['7','-','5']]
finalBoard = [['1','2','3'],['8','-','4'],['7','6','5']]
print(boardMain)
```

```
[['2', '8', '3'], ['1', '6', '4'], ['7', '-', '5']]
```

In [ ]:
```
def displayBoard(board):
  for i in board:
    for j in i:
      print(j,end=" ")
    print("")
  print("")
displayBoard(boardMain)
```

```
2 8 3
1 6 4
7 - 5
```

In [ ]:
```
def isGameOver(board,final):
  for i in range(0,len(board)):
    for j in range(0,len(board[0])):
      if(board[i][j] != final[i][j]):
        return False
  return True
isGameOver(boardMain,finalBoard)
```

Out[ ]:
```
False
```

Transistion functions

In [ ]:
```
# transistion functions
def findVoid(board):
  loc = [-1,-1]
  for i in range(0,len(board)):
    for j in range(0,len(board[0])):
      if(board[i][j] == '-'):
        loc[0] = i
        loc[1] = j
        return loc;
  return loc
findVoid(boardMain)

def up(board):
  loc = findVoid(board)
  ls = [x[:] for x in board]
  if(loc[0] - 1 >= 0):
    ls[loc[0]][loc[1]] = ls[loc[0] - 1][loc[1]]
    ls[loc[0]-1][loc[1]] = '-'
```

```
  return ls
displayBoard(up(boardMain))

def down(board):
  loc = findVoid(board)
  ls = [x[:] for x in board]
  if(loc[0] + 1 < len(ls)):
    ls[loc[0]][loc[1]] = ls[loc[0] + 1][loc[1]]
    ls[loc[0] + 1][loc[1]] = '-'
  return ls
displayBoard(down(boardMain))

def left(board):
  loc = findVoid(board)
  ls = [x[:] for x in board]
  if(loc[1] - 1 >= 0):
    ls[loc[0]][loc[1]] = ls[loc[0]][loc[1] - 1]
    ls[loc[0]][loc[1] - 1] = '-'
  return ls
displayBoard(left(boardMain))

def right(board):
  loc = findVoid(board)
  ls = [x[:] for x in board]
  if(loc[1] + 1 < len(ls)):
    ls[loc[0]][loc[1]] = ls[loc[0]][loc[1] + 1]
    ls[loc[0]][loc[1] + 1] = '-'
  return ls
displayBoard(right(boardMain))
```

```
2 8 3
1 - 4
7 6 5

2 8 3
1 6 4
7 - 5

2 8 3
1 6 4
- 7 5

2 8 3
1 6 4
7 5 -
```

Heuristics

In [ ]:

```
# h1 = misplaced tiles
def h1(b1,b2):
  count = 0
  for i in range(0,3):
    for j in range(0,3):
      if b1[i][j] != b2[i][j]:
        count += 1
  return count
h1(boardMain,finalBoard)
```

Out[ ]:

```
5
```

In [ ]:

```
# h2 = manhattan distance
```

```
def manDist(s,board):
  x,y = 0,0
  for i in range(0,3):
    for j in range(0,3):
      if(board[i][j] == s):
        x,y = i,j
  return (x,y)
def h2(b1,b2):
  count = 0
  for i in range(0,3):
    for j in range(0,3):
      loc = manDist(b1[i][j],b2)
      count += abs(loc[0] - i) + abs(loc[1]-j)
  return count
h2(boardMain,finalBoard)
```

Out[ ]:
```
6
```

2. A* using heuristic 1

In [ ]:
```
func = [up,down,left,right]
mem = {}
queue = []
queue.append((h1(boardMain,finalBoard),boardMain,0))
while(len(queue) > 0):
  print("Frontier list: ",end="")
  for i in queue:
    print(["( h: " + str(i[0]) + ")" + str(i[1])])
  h = queue[0][0]
  node = queue[0][1]
  d = queue[0][2]
  queue.pop(0)
  mem[str(node)] = 1
  print("Current node: ")
  displayBoard(node)
  if(isGameOver(node,finalBoard)):
    print("Goal State found\nAt depth: " + str(d))
    break
  for f in func:
    temp = f(node)
    if str(temp) in mem.keys():
      continue
    queue.append((h1(temp,finalBoard) + d + 1,temp,d + 1))
  queue.sort()
```

```
Frontier list: ["( h: 5)[['2', '8', '3'], ['1', '6', '4'], ['7', '-', '5']]"]
Current node:
2 8 3
1 6 4
7 - 5

Frontier list: ["( h: 4)[['2', '8', '3'], ['1', '-', '4'], ['7', '6', '5']]"]
["( h: 7)[['2', '8', '3'], ['1', '6', '4'], ['-', '7', '5']]"]
["( h: 7)[['2', '8', '3'], ['1', '6', '4'], ['7', '5', '-']]"]
Current node:
2 8 3
1 - 4
7 6 5

Frontier list: ["( h: 6)[['2', '-', '3'], ['1', '8', '4'], ['7', '6', '5']]"]
["( h: 6)[['2', '8', '3'], ['-', '1', '4'], ['7', '6', '5']]"]
["( h: 7)[['2', '8', '3'], ['1', '4', '-'], ['7', '6', '5']]"]
```

```
["( h: 7)[['2', '8', '3'], ['1', '6', '4'], ['-', '7', '5']]"]
["( h: 7)[['2', '8', '3'], ['1', '6', '4'], ['7', '5', '-']]"]
Current node:
2 - 3
1 8 4
7 6 5

Frontier list: ["( h: 6)[['-', '2', '3'], ['1', '8', '4'], ['7', '6', '5']]"]
["( h: 6)[['2', '8', '3'], ['-', '1', '4'], ['7', '6', '5']]"]
["( h: 7)[['2', '8', '3'], ['1', '4', '-'], ['7', '6', '5']]"]
["( h: 7)[['2', '8', '3'], ['1', '6', '4'], ['-', '7', '5']]"]
["( h: 7)[['2', '8', '3'], ['1', '6', '4'], ['7', '5', '-']]"]
["( h: 8)[['2', '3', '-'], ['1', '8', '4'], ['7', '6', '5']]"]
Current node:
- 2 3
1 8 4
7 6 5

Frontier list: ["( h: 6)[['1', '2', '3'], ['-', '8', '4'], ['7', '6', '5']]"]
["( h: 6)[['2', '8', '3'], ['-', '1', '4'], ['7', '6', '5']]"]
["( h: 7)[['2', '8', '3'], ['1', '4', '-'], ['7', '6', '5']]"]
["( h: 7)[['2', '8', '3'], ['1', '6', '4'], ['-', '7', '5']]"]
["( h: 7)[['2', '8', '3'], ['1', '6', '4'], ['7', '5', '-']]"]
["( h: 8)[['2', '3', '-'], ['1', '8', '4'], ['7', '6', '5']]"]
Current node:
1 2 3
- 8 4
7 6 5

Frontier list: ["( h: 5)[['1', '2', '3'], ['8', '-', '4'], ['7', '6', '5']]"]
["( h: 6)[['2', '8', '3'], ['-', '1', '4'], ['7', '6', '5']]"]
["( h: 7)[['2', '8', '3'], ['1', '4', '-'], ['7', '6', '5']]"]
["( h: 7)[['2', '8', '3'], ['1', '6', '4'], ['-', '7', '5']]"]
["( h: 7)[['2', '8', '3'], ['1', '6', '4'], ['7', '5', '-']]"]
["( h: 8)[['1', '2', '3'], ['7', '8', '4'], ['-', '6', '5']]"]
["( h: 8)[['2', '3', '-'], ['1', '8', '4'], ['7', '6', '5']]"]
Current node:
1 2 3
8 - 4
7 6 5

Goal State found
At depth: 5
```

3. A* using heuristic 2

In [ ]:

```
func = [up,down,left,right]
mem = {}
queue = []
queue.append((h2(boardMain,finalBoard),boardMain,0))

while(len(queue) > 0):
  print("Frontier list: ",end="")
  for i in queue:
    print(["( h: " + str(i[0]) + ")" + str(i[1])])
    # print()

  h = queue[0][0]
  node = queue[0][1]
  d = queue[0][2]
  queue.pop(0)
  mem[str(node)] = 1
  print("Current node: ")
```

```
    displayBoard(node)

    if(isGameOver(node,finalBoard)):
      print("Goal State found \nAt depth: " + str(d))
      break
    for f in func:
      temp = f(node)
      if str(temp) in mem.keys():
        continue
      queue.append((h2(temp,finalBoard) + d + 1,temp,d + 1))
    queue.sort()
```

```
Frontier list: ["( h: 6)[['2', '8', '3'], ['1', '6', '4'], ['7', '-', '5']]"]
Current node:
2 8 3
1 6 4
7 - 5

Frontier list: ["( h: 5)[['2', '8', '3'], ['1', '-', '4'], ['7', '6', '5']]"]
["( h: 9)[['2', '8', '3'], ['1', '6', '4'], ['-', '7', '5']]"]
["( h: 9)[['2', '8', '3'], ['1', '6', '4'], ['7', '5', '-']]"]
Current node:
2 8 3
1 - 4
7 6 5

Frontier list: ["( h: 6)[['2', '-', '3'], ['1', '8', '4'], ['7', '6', '5']]"]
["( h: 8)[['2', '8', '3'], ['-', '1', '4'], ['7', '6', '5']]"]
["( h: 8)[['2', '8', '3'], ['1', '4', '-'], ['7', '6', '5']]"]
["( h: 9)[['2', '8', '3'], ['1', '6', '4'], ['-', '7', '5']]"]
["( h: 9)[['2', '8', '3'], ['1', '6', '4'], ['7', '5', '-']]"]
Current node:
2 - 3
1 8 4
7 6 5

Frontier list: ["( h: 7)[['-', '2', '3'], ['1', '8', '4'], ['7', '6', '5']]"]
["( h: 8)[['2', '8', '3'], ['-', '1', '4'], ['7', '6', '5']]"]
["( h: 8)[['2', '8', '3'], ['1', '4', '-'], ['7', '6', '5']]"]
["( h: 9)[['2', '3', '-'], ['1', '8', '4'], ['7', '6', '5']]"]
["( h: 9)[['2', '8', '3'], ['1', '6', '4'], ['-', '7', '5']]"]
["( h: 9)[['2', '8', '3'], ['1', '6', '4'], ['7', '5', '-']]"]
Current node:
- 2 3
1 8 4
7 6 5

Frontier list: ["( h: 6)[['1', '2', '3'], ['-', '8', '4'], ['7', '6', '5']]"]
["( h: 8)[['2', '8', '3'], ['-', '1', '4'], ['7', '6', '5']]"]
["( h: 8)[['2', '8', '3'], ['1', '4', '-'], ['7', '6', '5']]"]
["( h: 9)[['2', '3', '-'], ['1', '8', '4'], ['7', '6', '5']]"]
["( h: 9)[['2', '8', '3'], ['1', '6', '4'], ['-', '7', '5']]"]
["( h: 9)[['2', '8', '3'], ['1', '6', '4'], ['7', '5', '-']]"]
Current node:
1 2 3
- 8 4
7 6 5

Frontier list: ["( h: 5)[['1', '2', '3'], ['8', '-', '4'], ['7', '6', '5']]"]
["( h: 8)[['2', '8', '3'], ['-', '1', '4'], ['7', '6', '5']]"]
["( h: 8)[['2', '8', '3'], ['1', '4', '-'], ['7', '6', '5']]"]
["( h: 9)[['1', '2', '3'], ['7', '8', '4'], ['-', '6', '5']]"]
["( h: 9)[['2', '3', '-'], ['1', '8', '4'], ['7', '6', '5']]"]
["( h: 9)[['2', '8', '3'], ['1', '6', '4'], ['-', '7', '5']]"]
```

```
["( h: 9)[['2', '8', '3'], ['1', '6', '4'], ['7', '5', '-']]"]
Current node:
1 2 3
8 - 4
7 6 5

Goal State found
At depth: 5
```