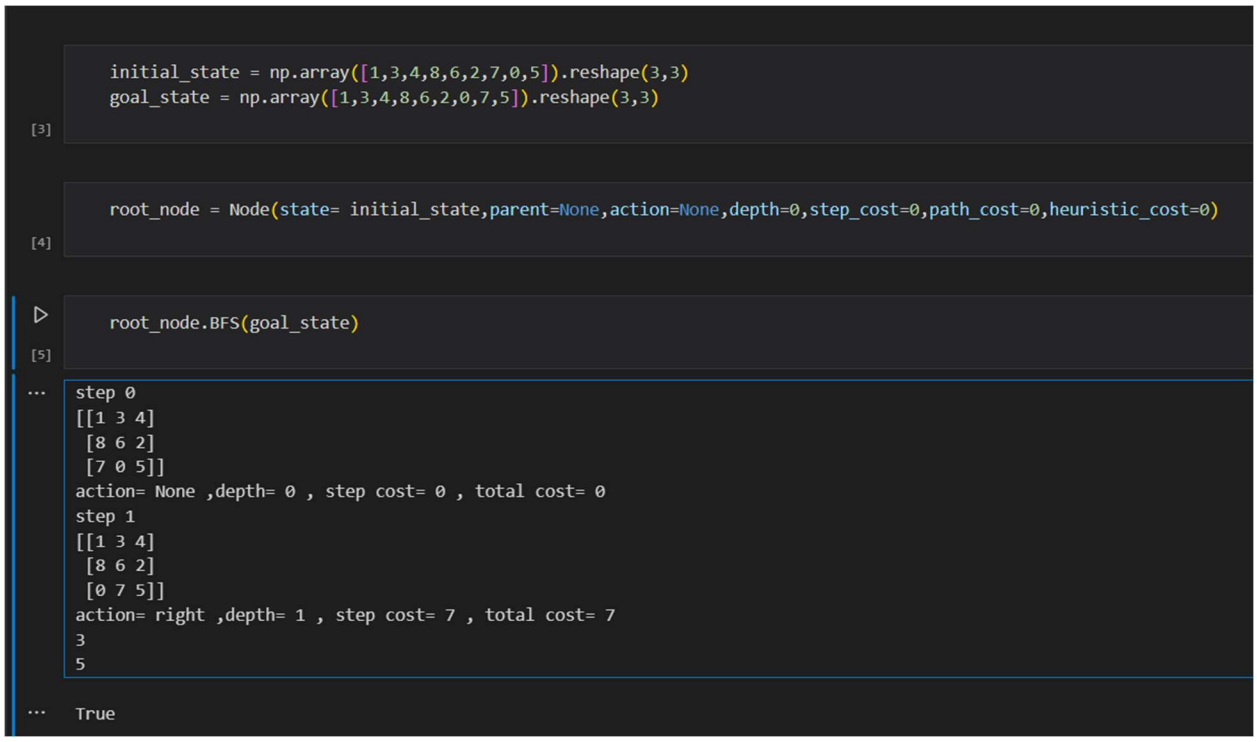
Assignment 1

Q1 Output:



```
[3]
initial_state = np.array([1,3,4,8,6,2,7,0,5]).reshape(3,3)
goal_state = np.array([1,3,4,8,6,2,0,7,5]).reshape(3,3)

[4]
root_node = Node(state= initial_state,parent=None,action=None,depth=0,step_cost=0,path_cost=0,heuristic_cost=0)

[5]
root_node.BFS(goal_state)

...
step 0
[[1 3 4]
 [8 6 2]
 [7 0 5]]
action= None ,depth= 0 , step cost= 0 , total cost= 0
step 1
[[1 3 4]
 [8 6 2]
 [0 7 5]]
action= right ,depth= 1 , step cost= 7 , total cost= 7
3
5

...
True
```

Q2 Output:

```
print("Enter a initial state(with spaces)::{eg:- 1 2 3 4 5 6 7 8 0}")
initial_state=[int(i) for i in input().split()]
initial_state= np.array([initial_state]).reshape(3,3)
print("Given input is::")
print(initial_state)
a=int(input("Enter 1 for up, 2 for down, 3 for left and 4 for right:: "))
print("Output:")
if a==1:
  print(up(initial_state))
elif a==2:
  print(down(initial_state))
elif a==3:
  print(left(initial_state))
elif a==4:
  print(right(initial_state))
else:
  print("give a valid input")
```

```
Enter a initial state(with spaces)::{eg:- 1 2 3 4 5 6 7 8 0}
1 2 3 4 5 6 7 8 0
Given input is::
[[1 2 3]
 [4 5 6]
 [7 8 0]]
Enter 1 for up, 2 for down, 3 for left and 4 for right:: 3
Output:
[[1 2 3]
 [4 5 6]
 [7 0 8]]
```

Q3 Output:

```
[[1 2 3]
 [6 8 4]
 [7 0 5]]
action= right ,depth= 8 , step cost= 7 , total cost= 51
step 9
[[1 2 3]
 [6 0 4]
 [7 8 5]]
action= up ,depth= 9 , step cost= 8 , total cost= 59
step 10
[[1 2 3]
 [6 4 0]
 [7 8 5]]
action= right ,depth= 10 , step cost= 4 , total cost= 63
step 11
[[1 2 3]
 [6 4 5]
 [7 8 0]]
action= down ,depth= 11 , step cost= 5 , total cost= 68
step 12
[[1 2 3]
 [6 4 5]
 [7 0 8]]
action= left ,depth= 12 , step cost= 8 , total cost= 76
step 13
[[1 2 3]
 [6 4 5]
 [0 7 8]]
action= left ,depth= 13 , step cost= 7 , total cost= 83
step 14
[[1 2 3]
 [0 4 5]
 [6 7 8]]
action= up ,depth= 14 , step cost= 6 , total cost= 89
step 15
[[1 2 3]
 [4 0 5]
 [6 7 8]]
action= right ,depth= 15 , step cost= 4 , total cost= 93
step 16
[[1 2 3]
 [4 5 0]
 [6 7 8]]
action= right ,depth= 16 , step cost= 5 , total cost= 98
A Goal State is achieved
```

Q4 Output:

```
step 19
[[3 2 5]
 [6 1 0]
 [7 4 8]]
action= right ,depth= 19 , step cost= 1 , total cost= 84
step 20
[[3 2 0]
 [6 1 5]
 [7 4 8]]
action= up ,depth= 20 , step cost= 5 , total cost= 89
step 21
[[3 0 2]
 [6 1 5]
 [7 4 8]]
action= left ,depth= 21 , step cost= 2 , total cost= 91
step 22
[[3 1 2]
 [6 0 5]
 [7 4 8]]
action= down ,depth= 22 , step cost= 1 , total cost= 92
step 23
[[3 1 2]
 [6 4 5]
 [7 0 8]]
action= down ,depth= 23 , step cost= 4 , total cost= 96
step 24
[[3 1 2]
 [6 4 5]
 [0 7 8]]
action= left ,depth= 24 , step cost= 7 , total cost= 103
step 25
[[3 1 2]
 [0 4 5]
 [6 7 8]]
action= up ,depth= 25 , step cost= 6 , total cost= 109
step 26
[[0 1 2]
 [3 4 5]
 [6 7 8]]
action= up ,depth= 26 , step cost= 3 , total cost= 112
True
```

It took us 26 moves to reach to Goal state

Q5 Output:

```
Move_action= up ,depth= 97590 , step_cost= 3 , move_cost= 438987

step 97591
[[1 0 5]
 [3 2 4]
 [6 7 8]]
Move_action= right ,depth= 97591 , step_cost= 1 , move_cost= 438988

step 97592
[[1 2 5]
 [3 0 4]
 [6 7 8]]
Move_action= down ,depth= 97592 , step_cost= 2 , move_cost= 438990

step 97593
[[1 2 5]
 [3 4 0]
 [6 7 8]]
Move_action= right ,depth= 97593 , step_cost= 4 , move_cost= 438994

step 97594
[[1 2 0]
 [3 4 5]
 [6 7 8]]
Move_action= up ,depth= 97594 , step_cost= 5 , move_cost= 438999

step 97595
[[1 0 2]
 [3 4 5]
 [6 7 8]]
Move_action= left ,depth= 97595 , step_cost= 2 , move_cost= 439001

step 97596
[[0 1 2]
 [3 4 5]
 [6 7 8]]
Move_action= left ,depth= 97596 , step_cost= 1 , move_cost= 439002
```

From my observations Breadth First Search found the solution with fewer moves and DFS took more actions to reach the goal state

Q6 Output:

```
 [6 0 1]
 [7 8 3]]
action= right ,depth= 10 , step cost= 6 , total cost= 50
step 11
[[2 5 4]
 [6 1 0]
 [7 8 3]]
action= right ,depth= 11 , step cost= 1 , total cost= 51
step 12
[[2 5 4]
 [6 1 3]
 [7 8 0]]
action= down ,depth= 12 , step cost= 3 , total cost= 54
step 13
[[2 5 4]
 [6 1 3]
 [7 0 8]]
action= left ,depth= 13 , step cost= 8 , total cost= 62
step 14
[[2 5 4]
 [6 1 3]
 [0 7 8]]
action= left ,depth= 14 , step cost= 7 , total cost= 69
step 15
[[2 5 4]
 [0 1 3]
 [6 7 8]]
action= up ,depth= 15 , step cost= 6 , total cost= 75
step 16
[[2 5 4]
 [1 0 3]
 [6 7 8]]
action= right ,depth= 16 , step cost= 1 , total cost= 76
step 17
[[2 0 4]
 [1 5 3]
 [6 7 8]]
action= up ,depth= 17 , step cost= 5 , total cost= 81
step 18
[[2 4 0]
 [1 5 3]
 [6 7 8]]
```

Q7 Output:

```
step 0
[[1 3 4]
 [8 6 2]
 [7 0 5]]
action= None ,depth= 0 , step cost= 0  total cost= 0
step 1
[[1 3 4]
 [8 0 2]
 [7 6 5]]
action= up ,depth= 1 , step cost= 6  total cost= 7
step 2
[[1 3 4]
 [8 2 0]
 [7 6 5]]
action= right ,depth= 2 , step cost= 2  total cost= 11
step 3
[[1 3 0]
 [8 2 4]
 [7 6 5]]
action= up ,depth= 3 , step cost= 4  total cost= 16
step 4
[[1 0 3]
 [8 2 4]
 [7 6 5]]
action= left ,depth= 4 , step cost= 3  total cost= 19.5
step 5
[[1 2 3]
 [8 0 4]
 [7 6 5]]
action= down ,depth= 5 , step cost= 2  total cost= 22.5
27
19
True
```

```
step 0
[[1 3 4]
 [8 6 2]
 [7 0 5]]
action= None ,depth= 0 , step cost= 0  total cost= 0
step 1
[[1 3 4]
 [8 0 2]
 [7 6 5]]
action= up ,depth= 1 , step cost= 6  total cost= 7
step 2
[[1 3 4]
 [8 2 0]
 [7 6 5]]
action= right ,depth= 2 , step cost= 2  total cost= 10
step 3
[[1 3 0]
 [8 2 4]
 [7 6 5]]
action= up ,depth= 3 , step cost= 4  total cost= 15
step 4
[[1 0 3]
 [8 2 4]
 [7 6 5]]
action= left ,depth= 4 , step cost= 3  total cost= 19
step 5
[[1 2 3]
 [8 0 4]
 [7 6 5]]
action= down ,depth= 5 , step cost= 2  total cost= 22
27
19
True
```