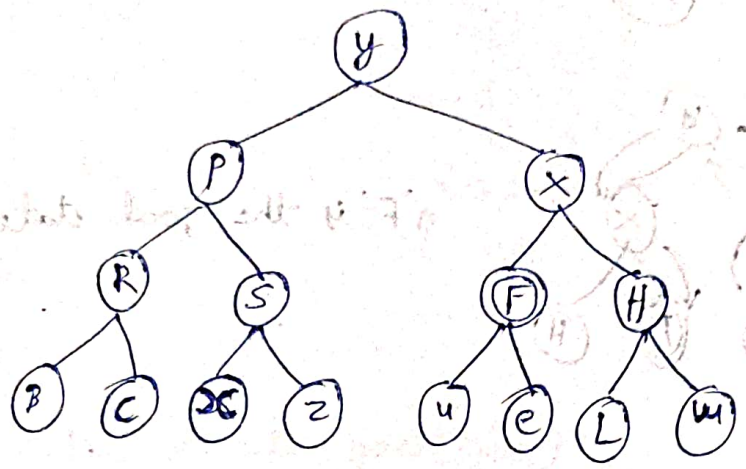
Lab - 04

* Write an algorithm and code for Iterative deepening depth first search and solve 8 puzzle using A* algorithm.

→ Iterative deepening DFS

(1)



⇒ Iterative deepening DFS is a combination of DFS and BFS. It goes to each level and traverses each level. If the goal state is not

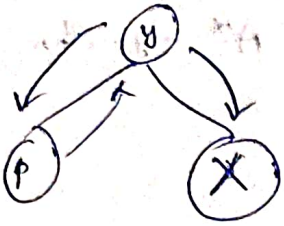
Step 1: Call the limit search function from range (1, max-size)

goal ⇒ Given as an i/p

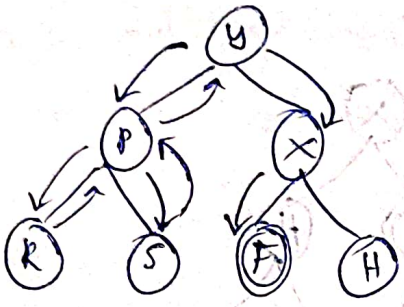
```
def IDDFS(graph, limit, start):
    for depth = 0 to limit:
        result = DFS(start, limit, depth)
        if result:
            return result
        else:
            return None

def DFS(root, limit, depth):
    if root == goal:
        return root
    if depth == limit + 1 : return
    for child in root:
        DFS(child, limit, depth)
```

Depth = 1

Depth = 2



Depth = 3



⇒ 'F' is the goal state

② 8 puzzle using A* algorithm

1	2	3
8		4
7	6	5

Initial state

2	8	1
	4	3
7	6	5

Goal state

1		3
8	2	4
7	6	5

g(n)=1

h(n) = 8

f(n) = 9

1	2	3
	8	4
7	6	5

h(n) = 6

f(n) = 7

1	2	3
8	4	
7	6	5

h(n) = 6

f(n) = 7

1	2	3
8	6	4
7		5

h(n) = 8

f(n) = 9

	2	3
1	8	4
7	6	5

h(n) = 7

f(n) = 9

1	2	3
8		4
7	6	5

h(n) = 7

f(n) = 9

1	2	3
7	8	4
	6	5

h(n) = 7

f(n) = 9

1	2	
8	4	3
7	6	5

h(n) = 5

f(n) = 7

1	2	3
8		4
7	6	5

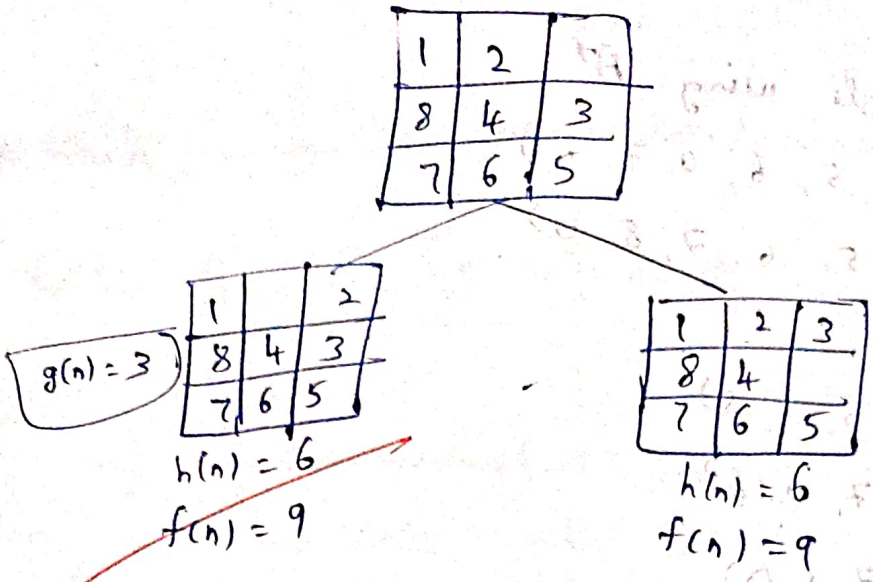
h(n) = 7

f(n) = 9

1	2	3
8	4	5
7	6	

h(n) = 7

f(n) = 9



Algorithm :-

```
def Astar(prestate, goalstate):
    for i in range(1, maxDepth):
    cost = 0
    if prestate != visited[]:
        states[] = generateMoves(prestate)
        for j in states:
            f[j] = cost[j] + manhattan(prestate, goalState)
count++;
mini = min(f)
visited.append(state)
Astar(state, goalstate)
```

Output :-

1) Iterative deepening DFS

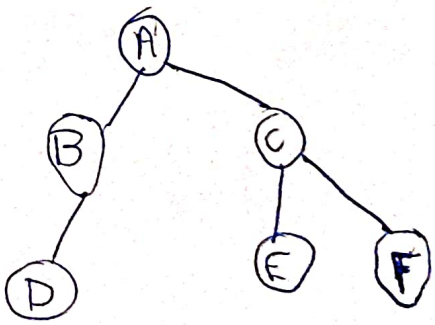
Iteration 1

A → B → C →

Iteration 2

A → B → D → C → E →

Target node E found!



2) Output for 8 puzzle using A*

start = (1, 2, 3, 4, 5, 6, 0, 7, 8)

goal = (1, 2, 3, 4, 5, 6, 7, 8, 0)

=> (1, 2, 3, 4, 5, 6, 0, 7, 8)

(1, 2, 3, 4, 5, 6, 7, 0, 8)

(1, 2, 3, 4, 5, 6, 7, 8, 0)