15/10/24

Iterative Deepening Search Algorithm

1. Take user input of graph (adjacency matrix) and max-depth of graph.

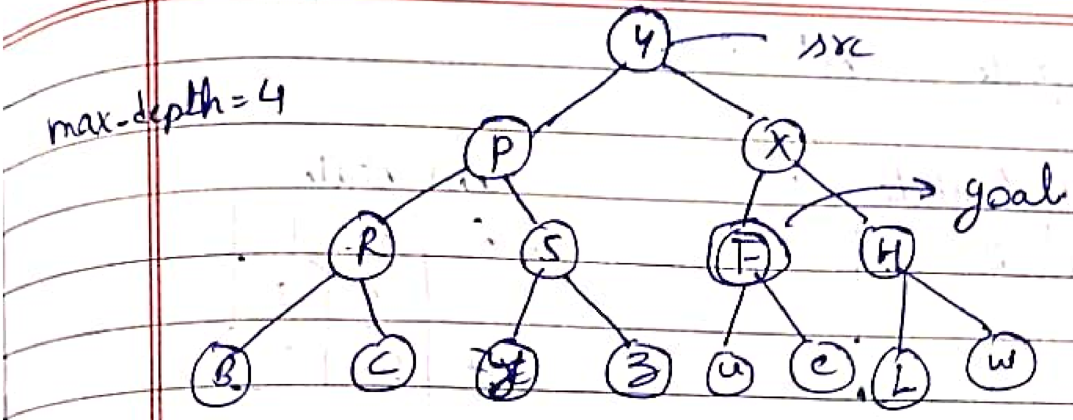
2. Take the initial node and goal node as input.

3. //IDDFS function to return whether we have reached goal node or not.

```
~~bool~~
bool IDDFS (src, target, ~~limit~~ max-depth)
    for limit from 0 to max-depth
        if D~~F~~LS (src, target, limit) == true
            return true
    return false
```

4. // Depth limit search function to check whether we have reached goal in the respective level.

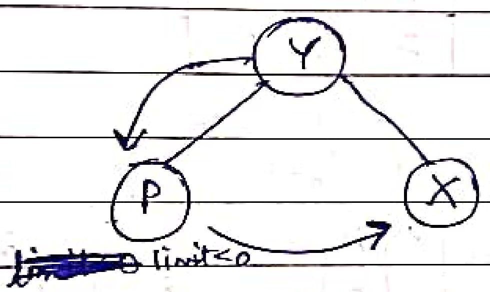
```
bool DLS(src , target , limit)
    if (src == target)
        return true
    if (limit <= 0)
        return false
    for each adjacent i of src
        if (DLS (i, target , limit - 1))
            return true
    return false.
```



~~Graph:~~

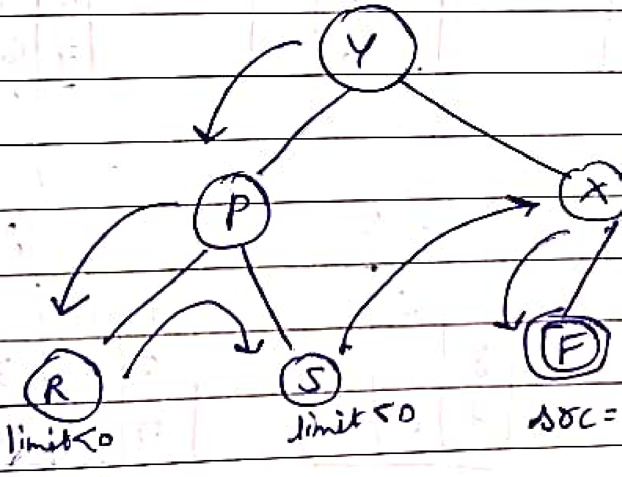
Level 0 Y limit=0

Level 1 limit = 1



YPX

Level 2 limit = 2



src = target => true

YPRSXF

∴ Goal is reached with path Y→P→R→S→X→F.

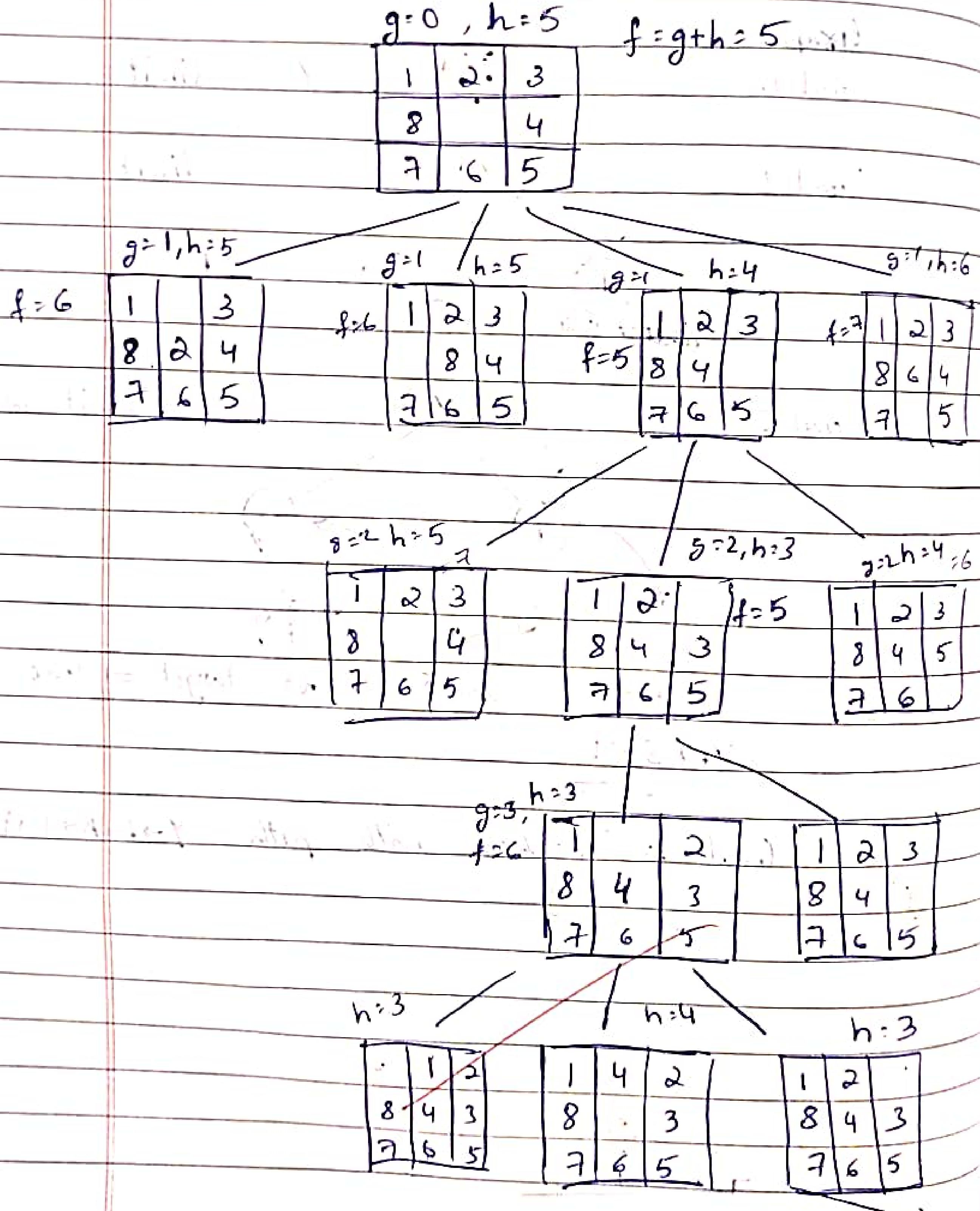
8 Puzzle A* :

Initial state

1	2	3
8		4
7	6	5

Goal state

2	8	1
	4	3
7	6	5



Algorithm :

1. Take the input of initial state as 3×3 matrix.

2. Define the goal state.

3. Heiuristic function (h) = count of misplaced tiles ~~that~~ from goal state.

4. Cost function (g) = count of current state from start state. It's the number of moves taken.

5. $f = g + h$
Whichever state has minimum f value that will be chosen for further expansion And the steps 3, 4, 5 are repeated until we reach the goal state.

6. If $h == 0$ → return goalstate.

14/10/24