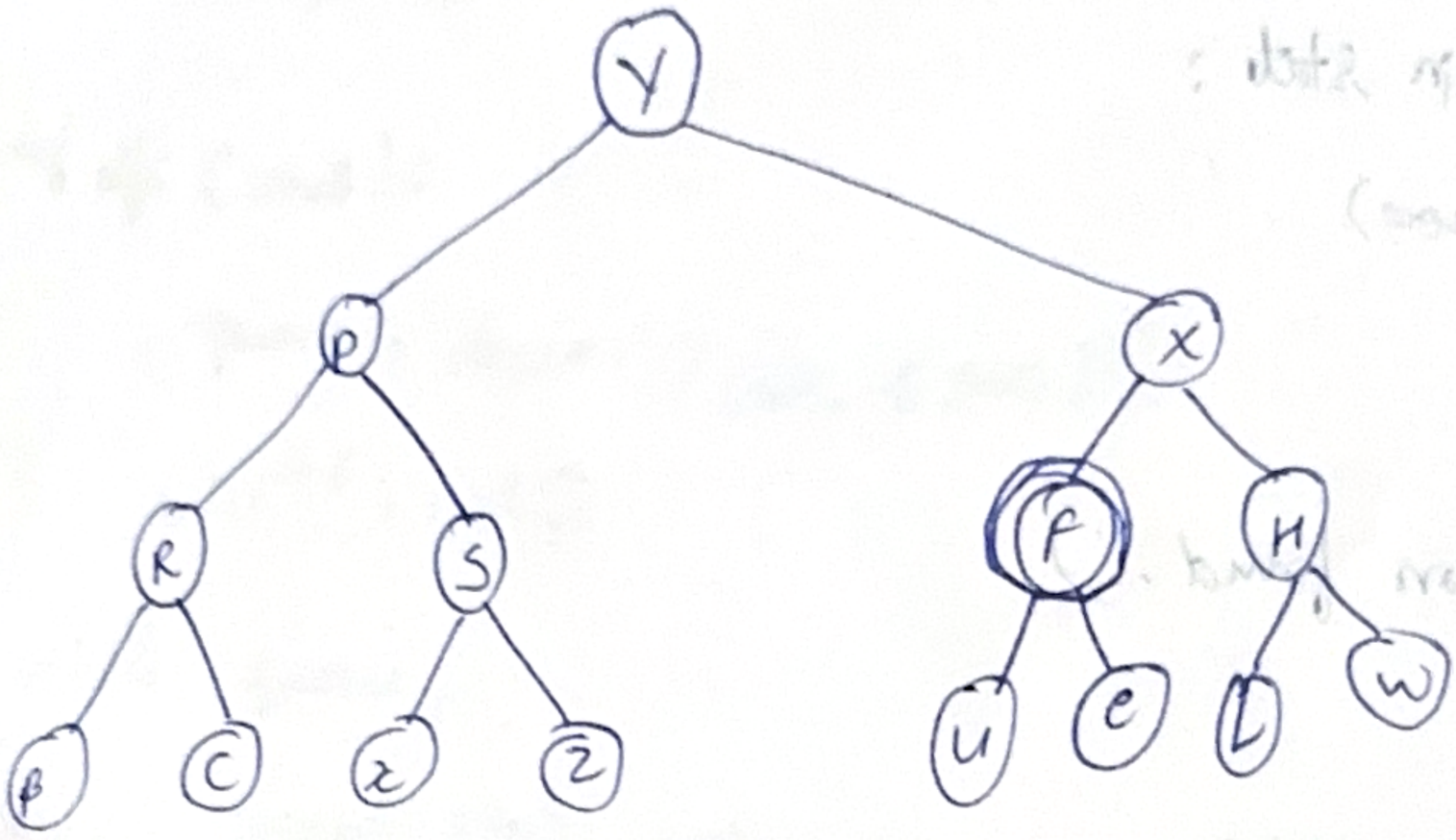
Lab-4

① Iterative deepening



⇒ Iterative DFS is combined of both dfs & bfs.

limit

Call the depth search function from range (0, max size)

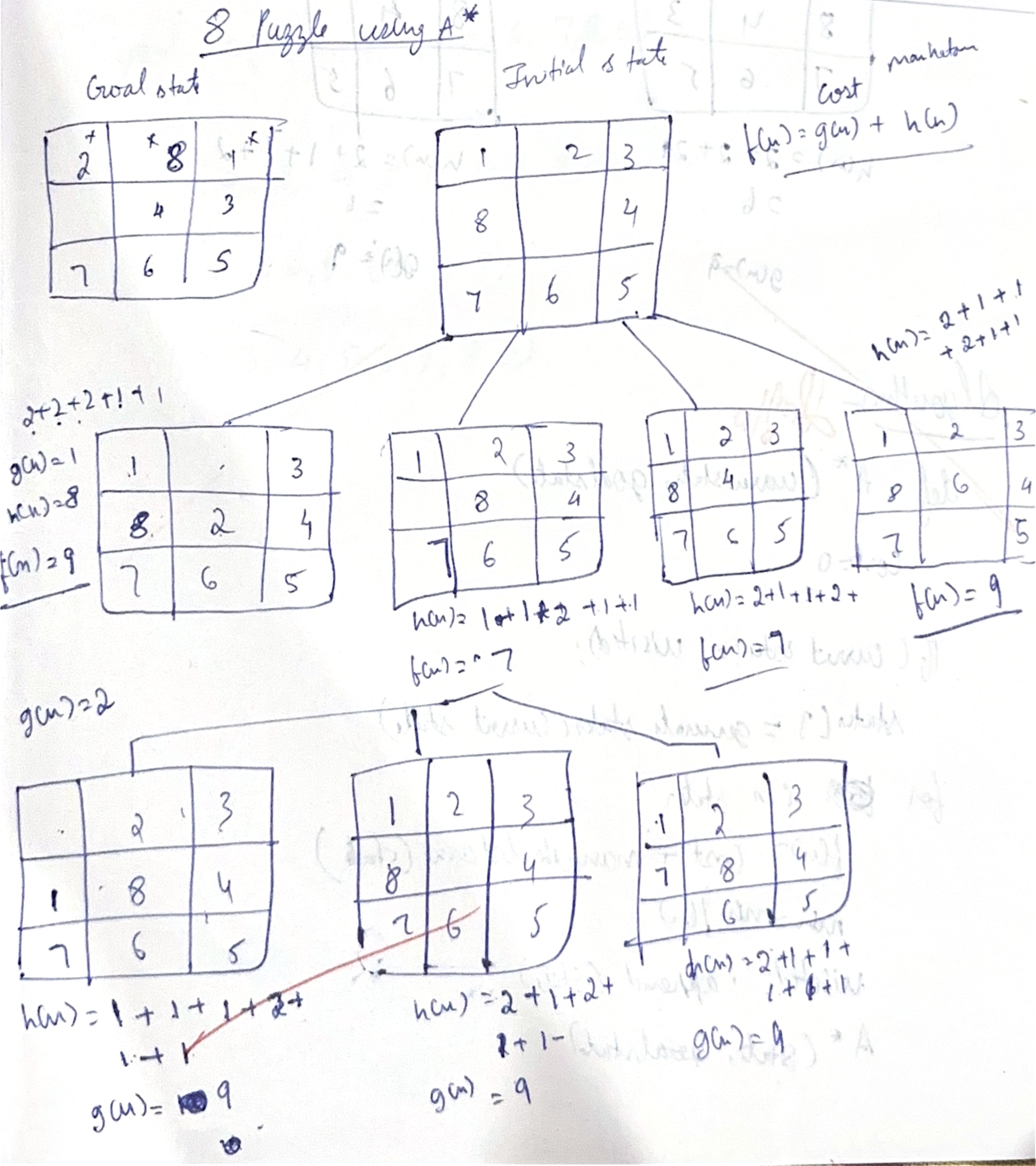
goal = given by user input.

```
func IDDFS (Graph , limit, start)

    for depth = 0 to limit;

        result = DFS (start, depth, limit):

        if result:
            return result

        else
            return : Null
```

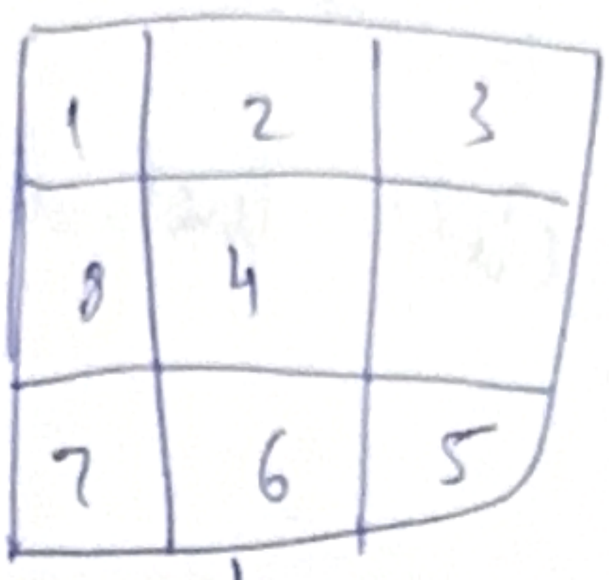
```
def DFS (root, limit, depth):
        if root = goal:
            return: root
    if depth == limit+1: return
```

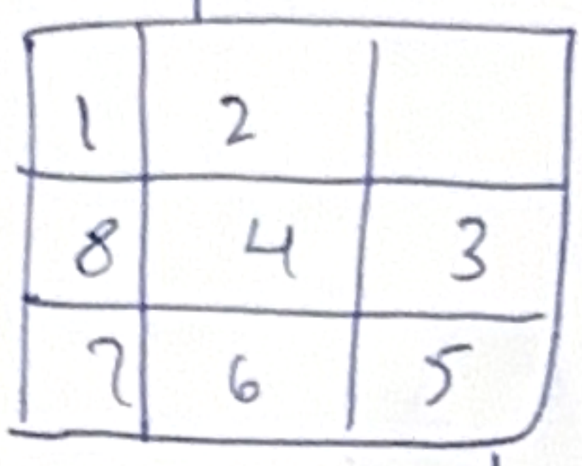
for child is root:

DFS (child, limit, depth)



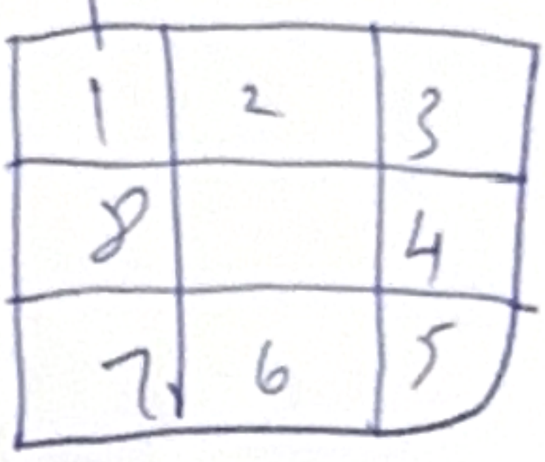
g(n)=9 g(n)=9 g(n)=9





h(n) = 5

g(n) = 9



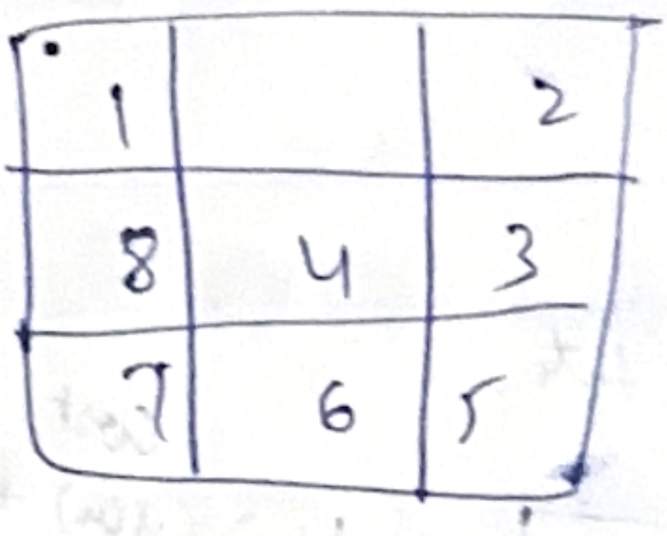
h(n)=7

g(n)=9

h(n)=7

g(n)=9

g(n)=3



h(n) = 2+2+2+

= 6

g(n)=9

h(n) = 2+1+1+2

= 6

g(n) = 9

Algorithm:- Prog 5/10

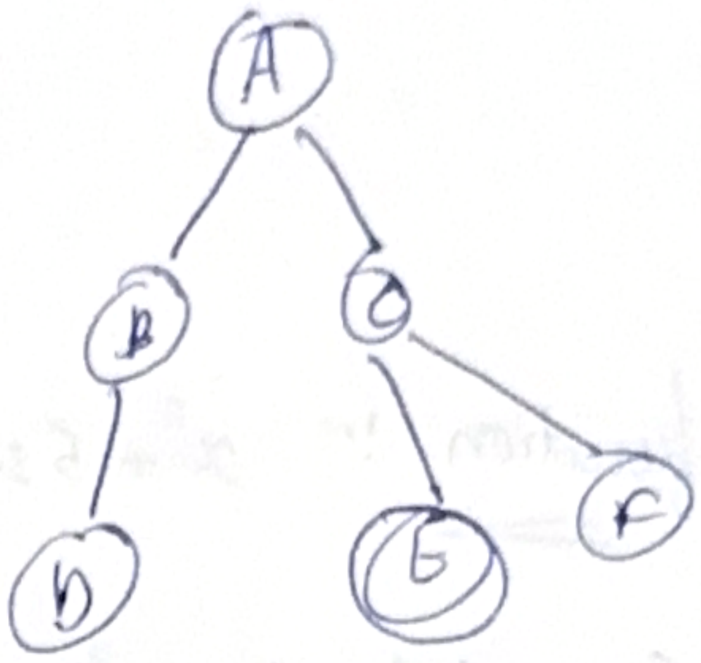
```
def A*(currentstate, goalstate)
    cost=0
If (current state != visited):
    states[] = generate states(current state)
for i in state:
    f(n) = cost + manhattandistance(state)
    min = min f(n)
    visited.append(state)
    A*(state, goalstate)
```

Output:

IDDFS:

Iteration 1:

A → B ⟷ C →



Iteration 2:

A → B → D → C → E →

Target node E found

⇒ A*

Output:

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)