Date 15/10/24
Page 15

15/10/24

Iterative Deepening search (IDS)

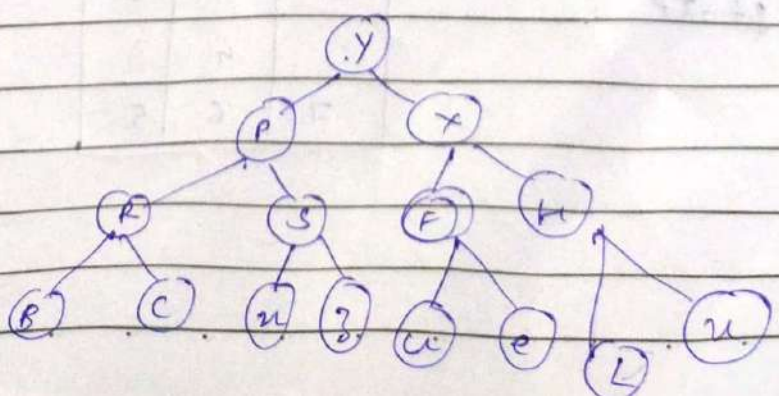
Algorithm:

1) The function Deapth limit search performs dfs (depth first search) till given max limit.

2) Call DLS function (1, max limit)
declare a global var. goal.

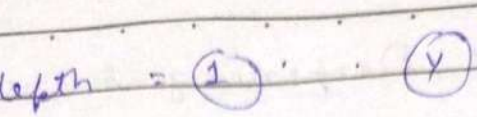
```
function IDS( graph, limit, start)
    for depth → 0 to limit:
        result = DFS( ~~, start depth~~ )
        if result
            return result
        else
            return None

function DFS ( root, ~~depth~~ limit)
    if root == goal
        return ~~goal~~
    if root = limit return
    for child in children
        (call recursive DFS function)
```

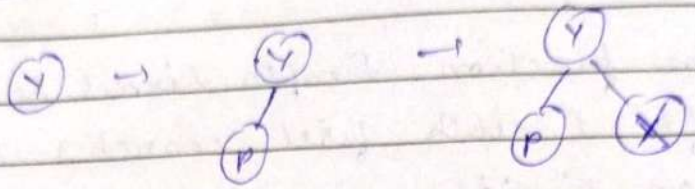


Date __/__/__
Page _____

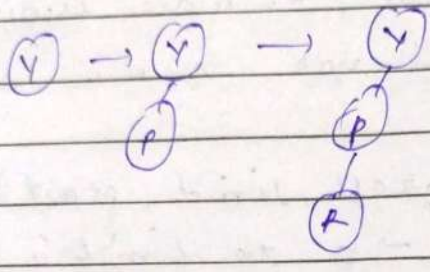
max depth = 1

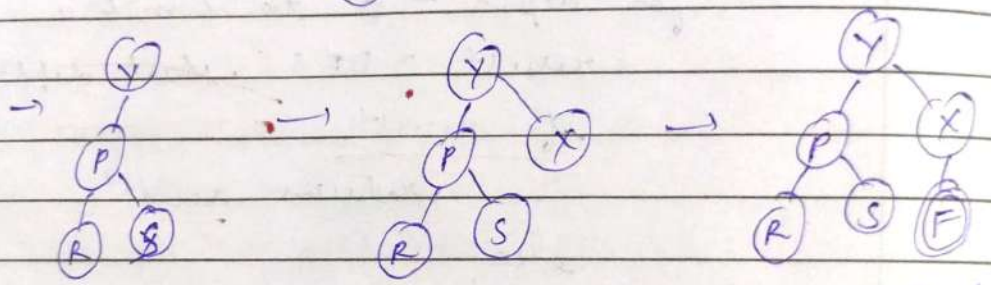


max depth = 2



max depth = 3





8-puzzle (Using A* algorithm)

Initial state :

1	2	3
8		4
7	6	5

Final state :

2	8	1
	4	3
7	6	5

Date __/__/__
Page _____

Algorithm:

1. Initialize a list which stores goal state of the puzzle $\begin{bmatrix} [2,8,1] \\ [0,4,7] \\ [6,5,5] \end{bmatrix}$

2. Initialize a priority queue() and take initial state as input from user.

3. Calculate manhattan distance and g_score
g-score → no of moves and store it in a variable.

$$f(n) = g(n) + md$$

f(n) → f score

manhattan distance.

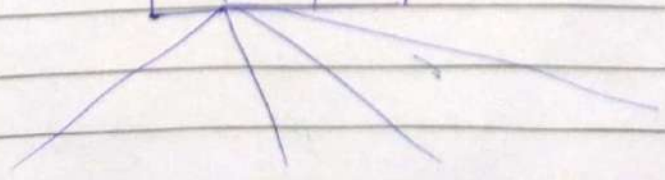
md = abs(curr_x - goal_x) + abs(curr_y - goal_y)

4. Using recursive DFS function and backtracking calculate f score and compare each iteration (which has less f-score)

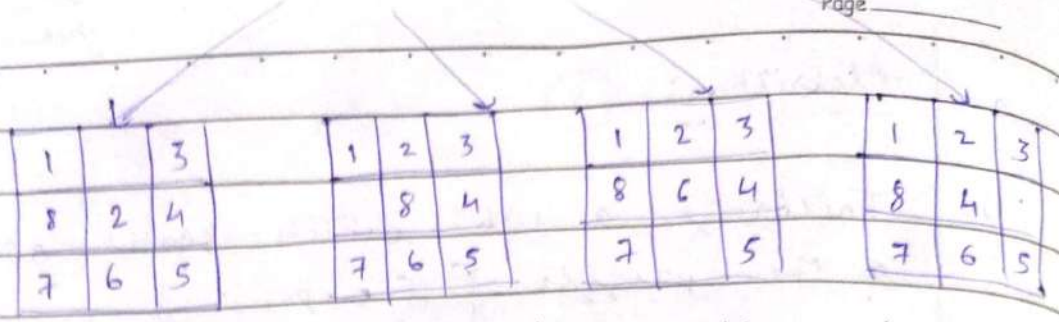
5. Traverse until the goal state is reached.

Ex:

1	2	3
8		4
7	6	5



Date __/__/__
Page _____



$f(n) = 1 + md$ ✗ $f(n) = 1 + md$ ✗ $f(n) = 1 +$ ✗ $f(n) = 1 + md$ ✓

$f(n)$ = no of moves + manhattan distance

Prakash
15/10/24