Lab-8

Class Node: → method

FUNCTION __init__(state, parent=None, g=0, h=0):
    SET self.state = state
    SET self.parent = parent
    SET self.g = g
    SET self.f = g+h

→ Lexicographic priority

FUNCTION __lt__(other):
    RETURN self.f < other.f

FUNCTION findBlank(state):
    FOR i from 0 to 2
        FOR j from 0 to 2:
            if state[i][j] == 0:
                RETURN (i, j)

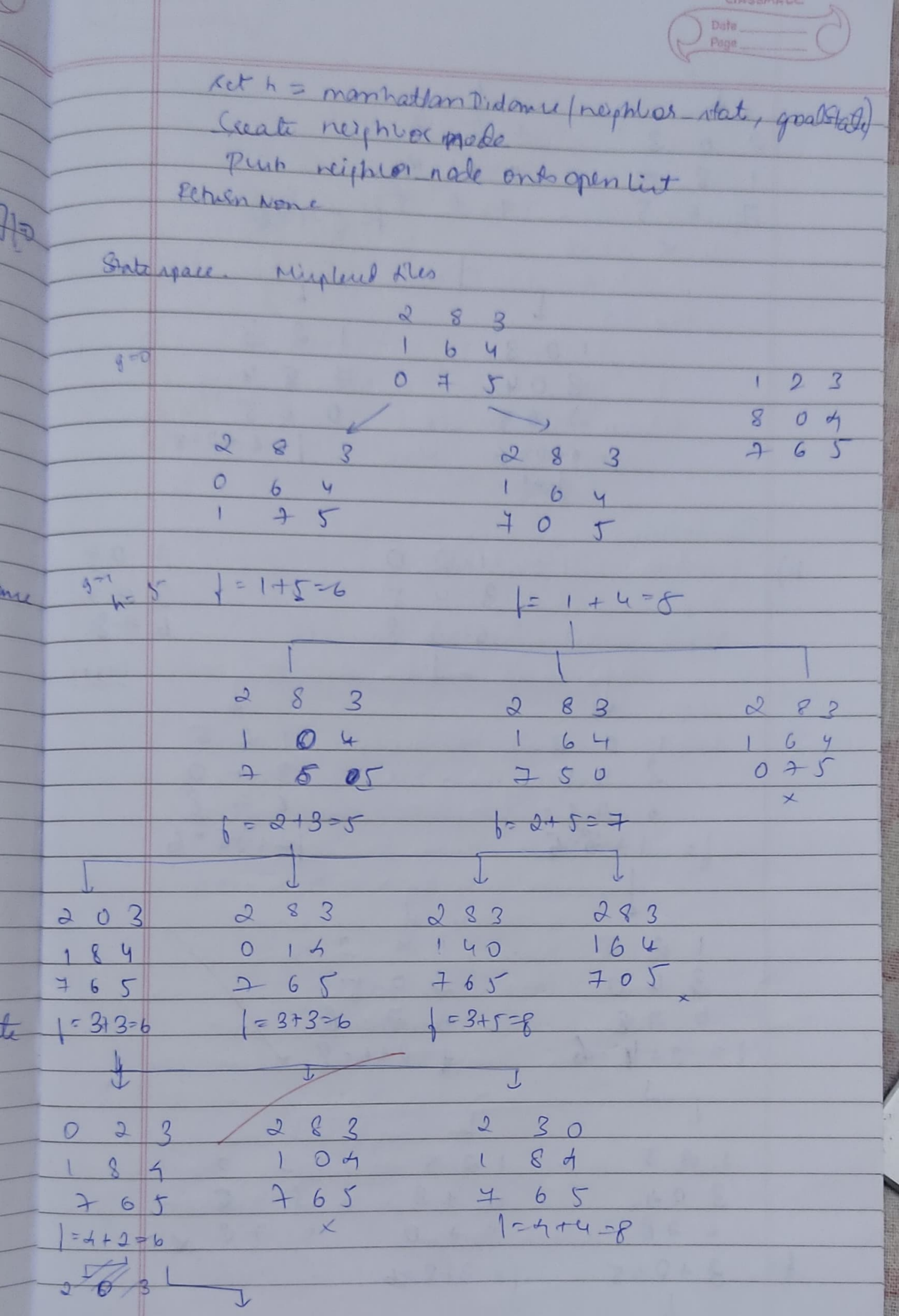
FUNCTION getNeighbours(state):
    SET (blankRow, blankCol) = findBlank(state)
    SET neighbours = [ ]
    SET possibleMoves = [(0,1), (0,-1), (1,0), (-1,0)]

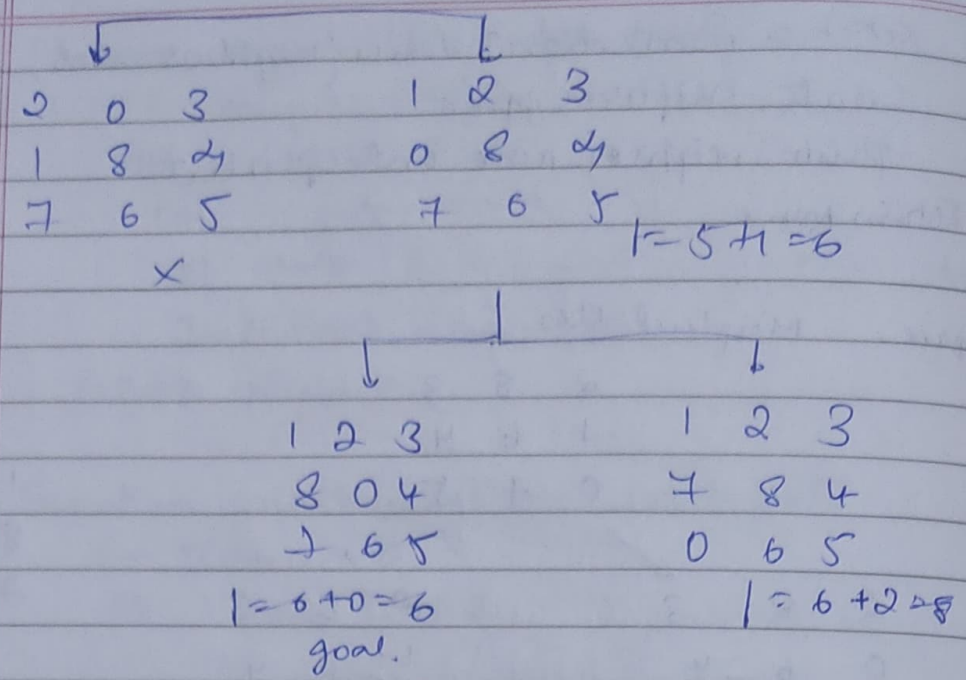
FOR (dr, dc) in possibleMoves:
    SET (newRow, newCol) = (blankRow + dr, blankCol + dc)
    If newRow And newCol are within bounds:
        CREATE newState by copying state
        SWAP blank tile with neighbour
        APPEND newState to neighbours.
    RETURN neighbours

```
Function misplacedTiles(state, goal):
    SET misplaced = 0:
    For i from 0 to 2:
        For j from 0 to 2
        If state[i][j] != goal[i][j] AND state[i][j]
        Increment misplaced
    Return misplaced

Function manhattanDistance(state, goal):
    Set Distance = 0
    For i from 0 to 2:
        if state[i][j] != 0
        find (goalRow, goalCol) in goal
        ADD abs(i - goalRow) + abs(j - goalCol) to
                                                distance

    Return distance

FUNCTION solvePuzzle(initialState, goalState):
    Initialize openList as priority queue
    Push initialState onto openList
    Initialize closedSet as empty set

While openList is not empty:
    Set currNode = pop node from openList

    If currNode.state == goalState
        Return path from currNode to initialState

    Add currNode.state to closed_set

    For each neighbor state in getNeighbors
                            (currNode.state);
    If neighbor state not in closed_set:
        Set g = currNode.g + 1
```

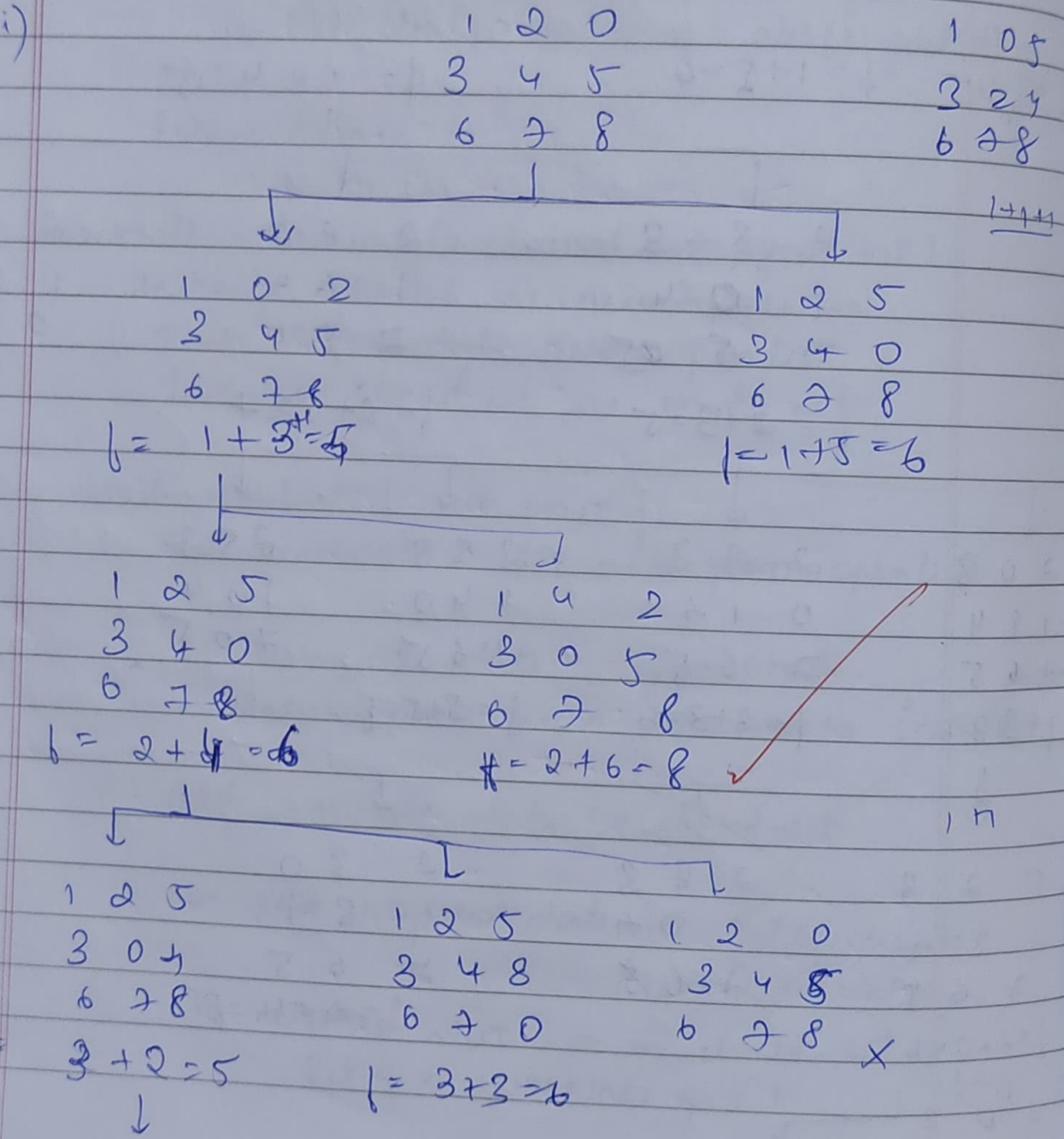
Set h = manhattanDistance(neighbor_state, goalState)
Create neighbor node
Push neighbor node onto open list
Return None

State space. Misplaced tiles




2 0 3
1 8 4
7 6 5
1 2 3
0 8 4
7 6 5
h=5+1=6
1 2 3
8 0 4
7 6 5
h=6+0=6
goal.
1 2 3
7 8 4
0 6 5
h=6+2=8

ii)


1 2 0
3 4 5
6 7 8
1 0 5
3 2 4
6 7 8
1 0 2
3 4 5
6 7 8
h= 1+3=4
1 2 5
3 4 0
6 7 8
h=1+5=6
1 2 5
3 4 0
6 7 8
h= 2+4=6
1 4 2
3 0 5
6 7 8
h=2+6=8
1 2 5
3 0 4
6 7 8
h= 3+2=5
1 2 5
3 4 8
6 7 0
h=3+3=6
1 2 0
3 4 5
6 7 8
X

1 0 5
3 2 4
6 7 8

goal

$f(n)$?