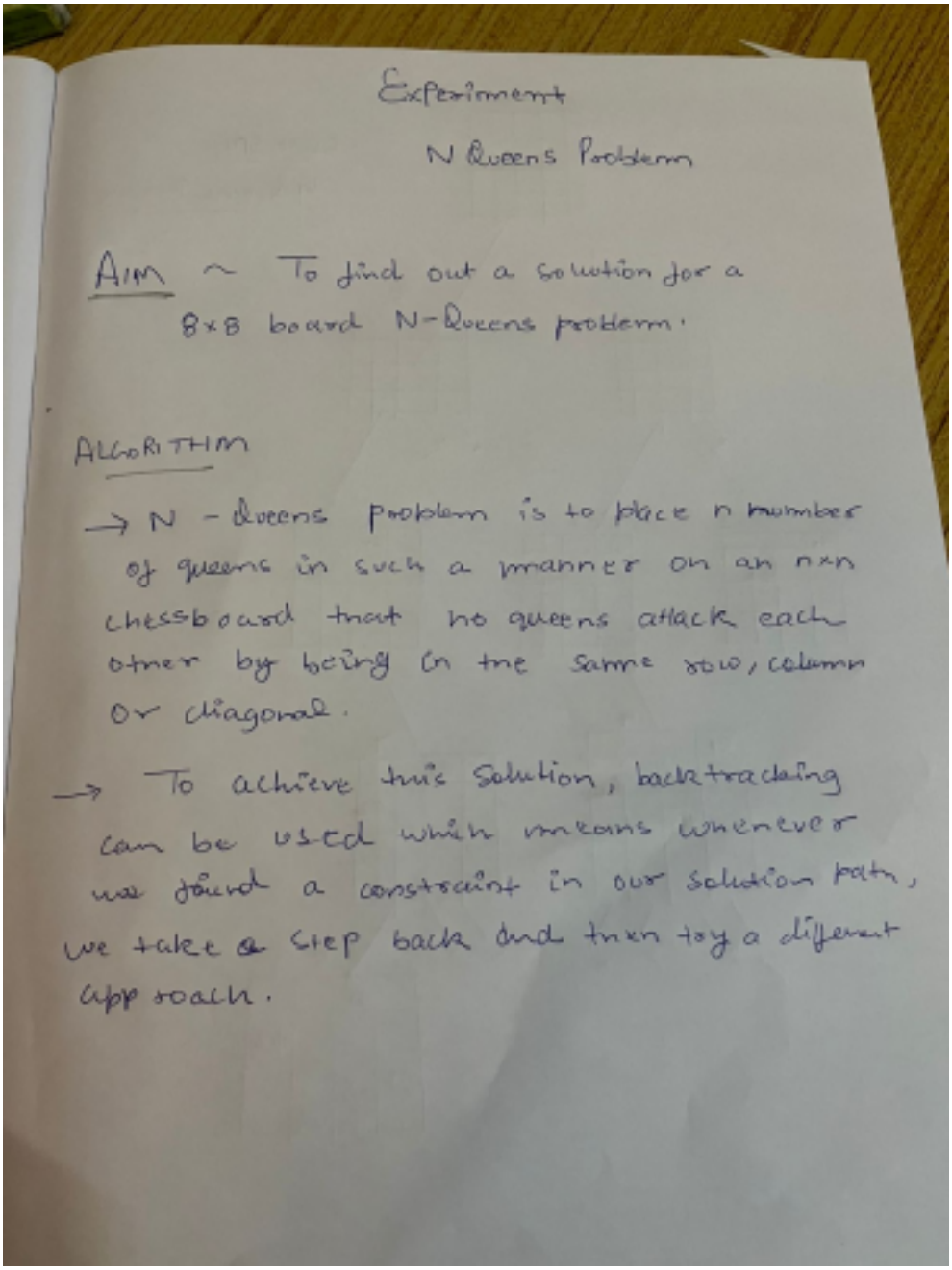
EX 1
Toy Problem

Harsh Saxena
RA1911033010060



Experiment

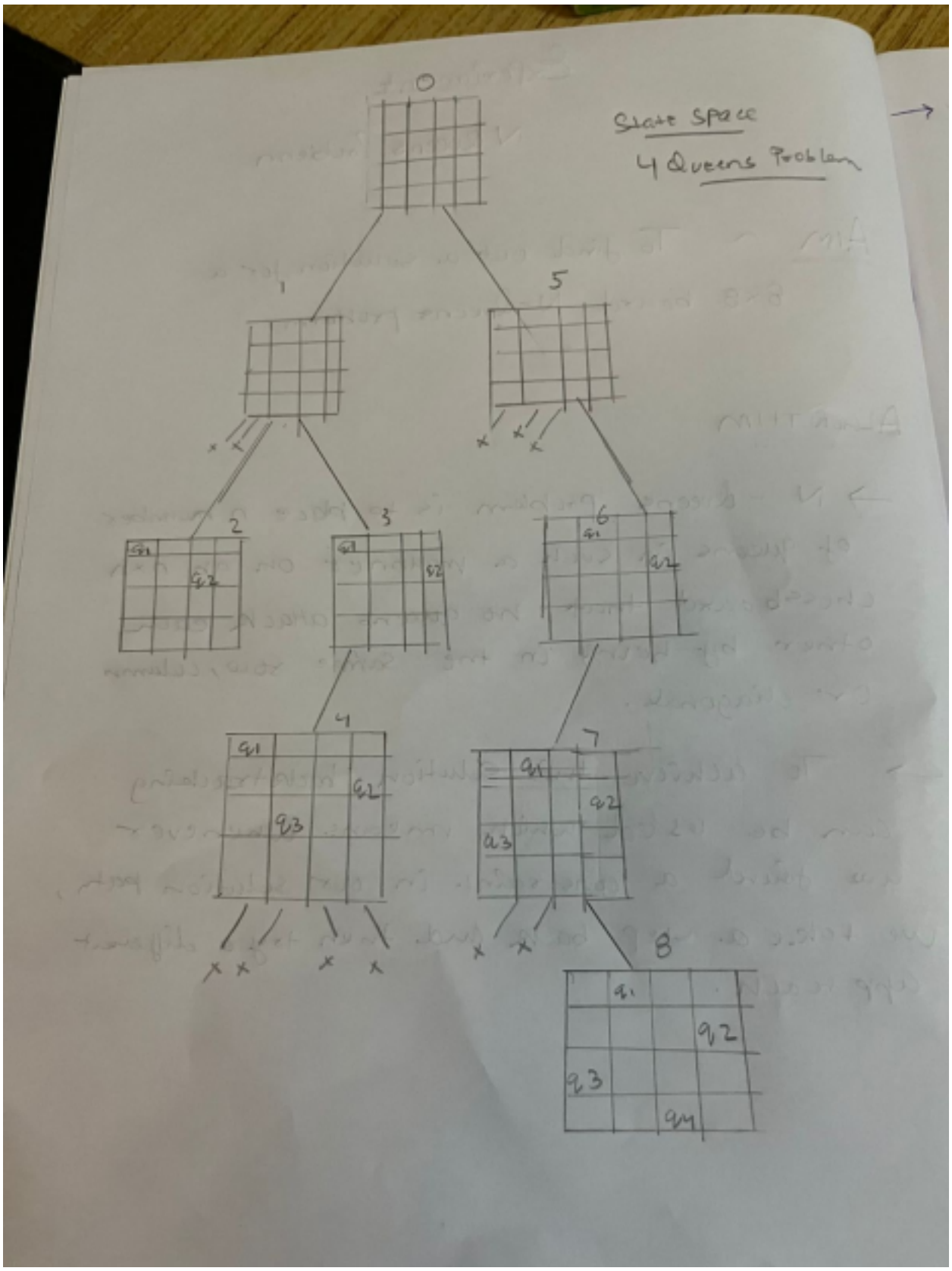
N Queens Problem

AIM ~ To find out a solution for a 8×8 board N-Queens problem.

ALGORITHM

→ N - Queens problem is to place n number of queens in such a manner on an n×n chessboard that no queens attack each other by being in the same row, column or diagonal.

→ To achieve this solution, backtracking can be used which means whenever we found a constraint in our solution path, we take a step back and then try a different approach.


State Space
4 Queens Problem
0
1
5
2
3
6
4
7
8
q1
q2
q3
q4

→ Here

```
nqueens (k,n)
{
  for i in range (1,n)
      if Place (k,i) then
      {
        x[k] ← i
        if (k==n) then
            Print(x[1-n]);
        else
            nqueens (k+1, n);
      }
}
```

1) start in the leftmost column
2) If all queens are placed
   return true
3) Try all rows in current column
   a) If queen can be placed, mark this as part of soln
   b) If placing queen here leads to soln, return t
   c) If not, go back to (a) and try again; ba
4) If all rows have been tried and false, then
   backtrack and try again

N Queen problem for a 8x8 chess board.

```python
# Taking number of queens as input from user
print ("Enter the number of queens")
N = int(input())
# here we create a chessboard
# NxN matrix with all elements set to 0
board = [[0]*N for _ in range(N)]
def attack(i, j):
    #checking vertically and horizontally
    for k in range(0,N):
        if board[i][k]==1 or board[k][j]==1:
            return True
    #checking diagonally
    for k in range(0,N):
        for l in range(0,N):
            if (k+l==i+j) or (k-l==i-j):
                if board[k][l]==1:
                    return True
    return False
def N_queens(n):
    if n==0:
        return True
    for i in range(0,N):
        for j in range(0,N):
            if (not(attack(i,j))) and (board[i][j]!=1):
                board[i][j] = 1
                if N_queens(n-1)==True:
                    return True
                board[i][j] = 0
    return False
N_queens(N)
for i in board:
    print (i)
```

```
Enter the number of queens
8
[1, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0]
[0, 1, 0, 0, 0, 0, 0, 0]
```


```python
# Taking number of queens as input from user
print ("Enter the number of queens")
N = int(input())
# here we create a chessboard
# NxN matrix with all elements set to 0
board = [[0]*N for _ in range(N)]
```

```
def attack(i, j):
    #checking vertically and horizontally
    for k in range(0,N):
        if board[i][k]==1 or board[k][j]==1:
            return True
    #checking diagonally
    for k in range(0,N):
        for l in range(0,N):
            if (k+l==i+j) or (k-l==i-j):
                if board[k][l]==1:
                    return True
    return False
def N_queens(n):
    if n==0:
        return True
    for i in range(0,N):
        for j in range(0,N):
            if (not(attack(i,j))) and (board[i][j]!=1):
                board[i][j] = 1
                if N_queens(n-1)==True:
                    return True
                board[i][j] = 0
    return False
N_queens(N)
for i in board:
    print (i)
```