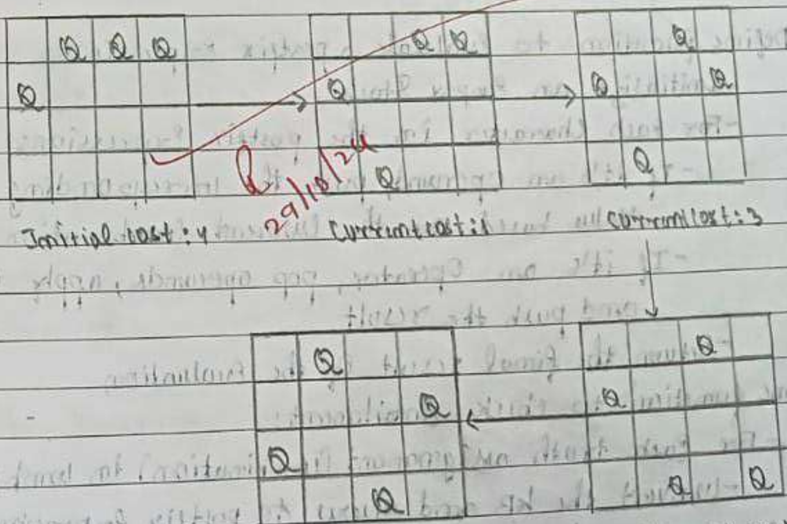
Date 29.10.24

① N Queens Using Simulates Annealing

```
function SimulatedAnnealing(N, initialTemp, coolingRate, maxIterations):

    CurrentSolution = randomConfiguration(N)
    current Cost = evaluateSolution(current Solution)
    bestSolution = currentSolution
    bestCost = currentCost
    temperature = initialTemp

    While Temperature > 0 and not terminationCondition():
        for i=0 to maxIterations:
            newSolution = generateNeighbor(currentSolution)
            newCost = evaluateSolution(newSolution)
            if new Cost < currentCost
                currentSolution = newSolution
                currentCost = newCost
            Else:
                delta = currentcost-newCost
                acceptanceProbability = exp(-delta/temperature)
                if random() < acceptanceProbability:
                    CurrentSolution = newSolution
                    currentCost = newCost
            if currentCost < bestCost:
                bestSolution = currentSolution
                bestCost = current Cost
        temperature = temperature * coolingRate
return bestSolution
```

```
function randomConfiguration(N):
    return [random(column(i) for i in range(N)]

function generateNeighbor(currentSolution):
    newSolution = currentSolution.copy()
    row = random() % len(newSolution)
    newColumn = random() % len(newSolution)
    newSolution[row] = newColumn
    return newSolution

function EvaluteSolution(solution):
    attacking Pairs = 0
    for i=0 to len(solution)-1:
        for j=i+1 to len(solution):
            if solution[i] == Solution[j] or abs(solution[i]-
                                     Solution[j]) == j-i:
                attackingPairs += 1
    return attackingPairs
```



Initial cost: 4

Current cost: 1

Current cost: 3

Current cost: 1

Best cost: 0

29/10/24