**LAB – 4**

1. Implement stimulated annealing algorithm to solve N- queens problem.

import random

import math

def calculate\_cost(state):

"""Calculate number of attacking queen pairs."""

cost = 0

n = len(state)

for i in range(n):

for j in range(i + 1, n):

if state[i] == state[j] or abs(state[i] - state[j]) == abs(i - j):

cost += 1

return cost

def generate\_random\_neighbor(state):

"""Generate a neighbor by moving one queen to a different row."""

neighbor = state.copy()

n = len(state)

col = random.randint(0, n - 1)

row = random.randint(0, n - 1)

while row == neighbor[col]:

row = random.randint(0, n - 1)

neighbor[col] = row

return neighbor

def simulated\_annealing(n, initial\_temp=100, cooling\_rate=0.95, max\_iter=1000):

current\_state = [random.randint(0, n - 1) for \_ in range(n)]

current\_cost = calculate\_cost(current\_state)

temperature = initial\_temp

iteration = 0

while temperature > 0 and iteration < max\_iter:

neighbor = generate\_random\_neighbor(current\_state)

neighbor\_cost = calculate\_cost(neighbor)

delta = neighbor\_cost - current\_cost

if delta < 0 or random.random() < math.exp(-delta / temperature):

current\_state = neighbor

current\_cost = neighbor\_cost

if current\_cost == 0:

break

temperature \*= cooling\_rate

iteration += 1

return current\_state, current\_cost

def print\_board(state):

n = len(state)

for row in range(n):

line = ""

for col in range(n):

if state[col] == row:

line += " Q "

else:

line += " . "

print(line)

print()

# Example usage:

N = 8

solution, cost = simulated\_annealing(N)

if cost == 0:

print("Solution found:")

else:

print(f"Stopped at cost {cost}:")

print\_board(solution)

**OUTPUT**:

