

## STIMULATED ANNEALING

CODE:-

```
print(f"SHREYASGOWDA C (1BM23CS319)")
import random
import math

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()

def calculate_cost(state):
    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]) == j - i:
                cost += 1
    return cost

def get_neighbor(state):
    n = len(state)
    neighbor = list(state)
    i, j = random.sample(range(n), 2)
    neighbor[i], neighbor[j] = neighbor[j], neighbor[i]
    return tuple(neighbor), (i, j)

def simulated_annealing(initial_state, initial_temp=1000, cooling_rate=0.95, min_temp=1e-3,
max_iter=1000):
    current_state = initial_state
    current_cost = calculate_cost(current_state)
    temperature = initial_temp
    path = [(current_state, current_cost, None)]

    print("Initial State:")
    print_board(current_state)
    print(f"Cost: {current_cost}\n")
```

## STIMULATED ANNEALING

```
iteration = 0
while temperature > min_temp and current_cost > 0 and iteration < max_iter:
    neighbor, swap = get_neighbor(current_state)
    neighbor_cost = calculate_cost(neighbor)

    cost_diff = neighbor_cost - current_cost

    if cost_diff < 0 or math.exp(-cost_diff / temperature) > random.random():
        current_state, current_cost = neighbor, neighbor_cost
        path.append((current_state, current_cost, swap))
        print(f"Iteration {iteration}: Swap columns {swap}")
        print_board(current_state)
        print(f"Cost: {current_cost}, Temperature: {temperature:.4f}\n")

    temperature *= cooling_rate
    iteration += 1

print("Terminated.")
return path

def get_initial_state():
    print("Enter the initial positions of the 4 queens (row for each column, 0-indexed):")
    positions = []
    for col in range(4):
        while True:
            try:
                pos = int(input(f"Column {col}: "))
                if 0 <= pos < 4:
                    positions.append(pos)
                    break
            else:
                print("Invalid input. Enter a number between 0 and 3.")
        except ValueError:
            print("Invalid input. Please enter an integer.")
    return tuple(positions)

initial_state = get_initial_state()
solution_path = simulated_annealing(initial_state)

print("Final path:")
for i, (state, cost, swap) in enumerate(solution_path):
```

## STIMULATED ANNEALING

```
print(f"Step {i}:")
print_board(state)
print(f"Cost: {cost}")
if swap is not None:
    print(f"Swap columns: {swap}")
    print("-----")
```

OUTPUT:-

```
IDLE Shell 3.13.5
File Edit Shell Debug Options Window Help
Python 3.13.5 (tags/v3.13.5:6cb20a2, Jun 11 2025, 16:15:46) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
== RESTART: C:/Users/student/AppData/Local/Programs/Python/Python313/319/5B.py =
SHREYASGOWDA C (1BM23CS319)
Enter the initial positions of the 4 queens (row for each column, 0-indexed):
Column 0: 1
Column 1: 2
Column 2: 3
Column 3: 4
Invalid input. Enter a number between 0 and 3.
Column 3: 0
Initial State:
. . . Q
Q . . .
. Q . .
. . Q .

Cost: 4

Iteration 0: Swap columns (3, 2)
. . Q .
Q . . .
. Q . . |
. . . Q

Cost: 1, Temperature: 1000.0000

Iteration 1: Swap columns (1, 0)
. . Q .
. Q . .
Q . . .
. . . Q

Cost: 4, Temperature: 950.0000

Iteration 2: Swap columns (1, 2)
. Q . .
. . Q .
Q . . .
. . . Q

Cost: 1, Temperature: 902.5000

Iteration 3: Swap columns (0, 1)
Q . . .
. . Q .
. Q . .
. . . Q

Cost: 2, Temperature: 857.3750

Iteration 4: Swap columns (0, 3)
. . . Q
. . Q .
. Q . .
Q . . .

Cost: 6, Temperature: 814.5062
```

# STIMULATED ANNEALING

```
IDLE Shell 3.13.5
File Edit Shell Debug Options Window Help

. . . Q

Cost: 4
Swap columns: (1, 0)
-----
Step 3:
. Q . .
. . Q .
Q . . .
. . . Q

Cost: 1
Swap columns: (1, 2)
-----
Step 4:
Q . . .
. . Q .
. Q . .
. . . Q

Cost: 2
Swap columns: (0, 1)
-----
Step 5:
. . . Q
. . Q .
. Q . .
Q . . .

Cost: 6
Swap columns: (0, 3)
-----
Step 6:
. . . Q
. . Q .
Q . . .
. Q . .

Cost: 2
Swap columns: (1, 0)
-----
Step 7:
. . Q .
. . . Q
Q . . .
. Q . .

Cost: 4
Swap columns: (2, 3)
-----
Step 8:
. . Q .
Q . . .
. . . Q
. Q . .

Cost: 0
Swap columns: (3, 0)
-----
>>>
```