LAB 5
Simulates annealing

```python
import random
import math
N = 4

def cost(state):
    """Compute number of attacking queen pairs."""
    conflicts = 0
    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):
                conflicts += 1
    return conflicts

def random_neighbor(state):
    """Generate a neighbor by moving one queen to another row."""
    neighbor = state.copy()
    col = random.randrange(N)
    new_row = random.randrange(N-1)
    if new_row >= neighbor[col]:
        new_row += 1
    neighbor[col] = new_row
    return neighbor

def simulated_annealing(
    T0=5.0, alpha=0.995, Tmin=1e-6, max_iters=50000
):
    state = [random.randrange(N) for _ in range(N)]
    current_cost = cost(state)
    T = T0
    it = 0

    while T > Tmin and it < max_iters and current_cost != 0:
        neighbor = random_neighbor(state)
        neighbor_cost = cost(neighbor)
        delta = neighbor_cost - current_cost

        if delta <= 0 or random.random() < math.exp(-delta / T):
            state, current_cost = neighbor, neighbor_cost

        T *= alpha
        it += 1

    return state, current_cost
```

```
    state = [random.randrange(N) for _ in range(N)]
    current_cost = cost(state)
    T = T0
    it = 0

    while T > Tmin and it < max_iters and current_cost != 0:
        neighbor = random_neighbor(state)
        neighbor_cost = cost(neighbor)
        delta = neighbor_cost - current_cost

        if delta <= 0 or random.random() < math.exp(-delta / T):
            state, current_cost = neighbor, neighbor_cost

        T *= alpha
        it += 1

    return state, current_cost

def print_board(state):
    """Pretty-print the board."""
    for r in range(N):
        row = ""
        for c in range(N):
            row += "Q " if state[c] == r else ". "
        print(row)
    print()
solution, c = simulated_annealing()

print("Final state (col -> row):", solution)
print("Cost:", c)
print("\nBoard:")
print_board(solution)
```

```
Final state (col -> row): [1, 3, 0, 2]
Cost: 0

Board:
. . Q .
Q . . .
. . . Q
. Q . .
```