TECNOLÓGICO
NACIONAL DE MÉXICO

ALUMNA:

BARRON ZAZUETA MARIA GUADALUPE

UNIDAD II

PROBLEMA 8 REINAS ALGORITMO RECOCIDO SIMULADO

24/03/2025

C: > Users > maria > Downloads > N reynas recocido.py > SimulatedAnnealing

```python
import random
import math
import time

def get_manual_input():
    manual_input = "3,6,2,7,1,4,0,5"  # Entrada predeterminada
    return [int(x) for x in manual_input.split(",")]

class SimulatedAnnealing:
    def __init__(self, n=8, initial_temp=1000, cooling_rate=0.95, max_iterations=1000):
        self.n = n
        self.initial_temp = initial_temp
        self.cooling_rate = cooling_rate
        self.max_iterations = max_iterations

    def generate_initial_solution(self, manual_input=None):
        if manual_input:
            return manual_input
        return [random.randint(0, self.n - 1) for _ in range(self.n)]

    def calculate_conflicts(self, state):
        conflicts = 0
        for i in range(self.n):
            for j in range(i + 1, self.n):
                if state[i] == state[j] or abs(state[i] - state[j]) == abs(i - j):
                    conflicts += 1
        return conflicts
```

C: > Users > maria > Downloads > N reynas recocido.py > SimulatedAnnealing

```python
class SimulatedAnnealing:
    def get_neighbor(self, state):
        new_state = list(state)
        col = random.randint(0, self.n - 1)
        row = random.randint(0, self.n - 1)
        new_state[col] = row
        return new_state

    def simulated_annealing(self, initial_state=None):
        current_state = self.generate_initial_solution(initial_state)
        best_state = list(current_state)
        best_conflicts = self.calculate_conflicts(best_state)
        temperature = self.initial_temp
        iteration = 0
        start_time = time.time()

        while temperature > 1 and iteration < self.max_iterations and best_conflicts > 0:
            neighbor = self.get_neighbor(current_state)
            current_conflicts = self.calculate_conflicts(current_state)
            neighbor_conflicts = self.calculate_conflicts(neighbor)
            delta = neighbor_conflicts - current_conflicts

            if delta < 0 or random.uniform(0, 1) < math.exp(-delta / temperature):
                current_state = neighbor
                if neighbor_conflicts < best_conflicts:
                    best_state = neighbor
                    best_conflicts = neighbor_conflicts

            temperature *= self.cooling_rate
```

```python
        end_time = time.time()
        return best_state, best_conflicts, iteration, end_time - start_time

if __name__ == "__main__":
    initial_state = get_manual_input()
    sa_solver = SimulatedAnnealing()
    solution, conflicts, moves, exec_time = sa_solver.simulated_annealing(initial_state)
    print(f"Mejor solución encontrada: {solution}")
    print(f"Conflictos restantes: {conflicts}")
    print(f"Movimientos realizados: {moves}")
    print(f"Tiempo de ejecución: {exec_time:.4f} segundos")
```

CORRIDA:

```
PS C:\Users\maria\Downloads>  & 'c:\Python\python.exe' 'c:\Users\maria\.vscode\extensions\ms-python.de
debugpy\launcher' '63986' '--' 'C:\Users\maria\Downloads\N reynas recocido.py'
Mejor solución encontrada: [3, 6, 2, 7, 1, 4, 0, 5]
Conflictos restantes: 0
Movimientos realizados: 0
Tiempo de ejecución: 0.0000 segundos
PS C:\Users\maria\Downloads>
```