$Implement\ Hill - Climb\ Searching\ Algorithm\ for\ n - queens\ problem$

```
import random

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

def hill_climbing(n):
    cost = 0
    while True:
        current_board = list(range(n))
        random.shuffle(current_board)
        current_conflicts = calculate_conflicts(current_board)

        while True:
            found_better = False
            for i in range(n):
                for j in range(n):
                    if j != current_board[i]:
                        neighbor_board = list(current_board)
                        neighbor_board[i] = j
                        neighbor_conflicts = calculate_conflicts(neighbor_board)
                        if neighbor_conflicts < current_conflicts:
                            print("Current Board:")
                            print_board(current_board)
                            print(f"Current Conflicts: {current_conflicts}")
                            print("Neighbor Board:")
                            print_board(neighbor_board)
                            print(f"Neighbor Conflicts: {neighbor_conflicts}")
                            current_board = neighbor_board
                            current_conflicts = neighbor_conflicts
                            cost += 1
                            found_better = True
                            break
                if found_better:
                    break

            if not found_better:
                break

        if current_conflicts == 0:
            return current_board, current_conflicts, cost

def print_board(board):
    n = len(board)
    for i in range(n):
        row = ['.'] * n
        row[board[i]] = 'Q'
        print(' '.join(row))
    print()

print("Output: 1BM22CS290")
n = 4
solution, conflicts, cost = hill_climbing(n)
print("Final Board Configuration:")
print_board(solution)
print("Number of Cost:", cost)
```

```
Output: 1BM22CS290
Current Board:
. Q . .
. . Q .
. . . Q
Q . . .

Current Conflicts: 4
Neighbor Board:
Q . . .
. . Q .
. . . Q
```

```
Q . . .

Neighbor Conflicts: 3
Current Board:
Q . . .
. . Q .
. . . Q
Q . . .

Current Conflicts: 3
Neighbor Board:
Q . . .
. Q . .
. . . Q
Q . . .

Neighbor Conflicts: 2
Current Board:
Q . . .
. Q . .
. . . Q
Q . . .

Current Conflicts: 2
Neighbor Board:
. . Q .
. Q . .
. . . Q
Q . . .

Neighbor Conflicts: 1
Final Board Configuration:
. . Q .
Q . . .
. . . Q
. Q . .

Number of Cost: 3
```