EXPERIMENT 5

PROBLEM TITLE

Solve the 8-Puzzle Problem Using Best-First Search.

CODE

```
import heapq

class EightPuzzle:
    def __init__(self, board):
        self.board = board
        self.goal = [1, 2, 3, 4, 5, 6, 7, 8, 0]

    def heuristic(self, state):
        return sum(abs((val-1) % 3 - i % 3) + abs((val-1) // 3 - i // 3)
                   for i, val in enumerate(state) if val != 0)

    def get_neighbors(self, state):
        def swap(state, i, j):
            state = list(state)
            state[i], state[j] = state[j], state[i]
            return state

        neighbors = []
        zero_index = state.index(0)
        x, y = zero_index % 3, zero_index // 3
        for dx, dy in [(-1, 0), (1, 0), (0, -1), (0, 1)]:
            nx, ny = x + dx, y + dy
            if 0 <= nx < 3 and 0 <= ny < 3:
                neighbor = swap(state, zero_index, ny * 3 + nx)
                neighbors.append(neighbor)
        return neighbors

    def best_first_search(self):
        open_list = []
        heapq.heappush(open_list, (self.heuristic(self.board), self.board))
        visited = set()

        while open_list:
            _, current = heapq.heappop(open_list)
            if current == self.goal:
                return current

            visited.add(tuple(current))
            for neighbor in self.get_neighbors(current):
                if tuple(neighbor) not in visited:
                    heapq.heappush(open_list, (self.heuristic(neighbor), neighbor))

        return None  # No solution found

# Example board: 0 represents the empty space
board = [1, 2, 3, 4, 5, 6, 0, 7, 8]
puzzle = EightPuzzle(board)
solution = puzzle.best_first_search()
print("Solved 8-Puzzle:", solution)
```

EXPECTED OUTPUT

```
Solved 8-Puzzle: [1, 2, 3, 4, 5, 6, 7, 8, 0]
```