BFS WITHOUT HEURISTIC

```
from collections import deque
 def get_neighbors(state):     neighbors = []     idx =
state.index("0")     moves = [(-1, 0), (1, 0), (0, -1), (0, 1)]  # up,
down, left, right     x, y = divmod(idx, 3)
     for dx, dy in
moves:
        nx, ny = x + dx, y + dy
if 0 <= nx < 3 and 0 <= ny < 3:
            new_idx = nx * 3 + ny             state_list = list(state)
state_list[idx], state_list[new_idx] = state_list[new_idx],
state_list[idx]             neighbors.append("".join(state_list))
return neighbors
 def bfs(start_state,
goal_state):     queue =
deque([start_state])     visited
= set([start_state])     parent =
{start_state: None}
     while
queue:
        current = queue.popleft()
if current == goal_state:
            path = []
while current:
                path.append(current)
current = parent[current]             return
path[::-1]         for neighbor in
get_neighbors(current):
            if neighbor not in visited:
visited.add(neighbor)
parent[neighbor] = current
queue.append(neighbor)     return None

# Get input from the user row by row print ("1BM23CS333")
print("Enter the initial state (enter 3 digits per row, separated by
spaces, 0 for empty):") initial_state_rows = [] for i in range(3):
    row = input(f"Row {i+1}: ").split()
```

```
    initial_state_rows.extend(row) initial_state
= "".join(initial_state_rows)
 print("\nEnter the goal state (enter 3 digits per row, separated
by spaces, 0 for empty):") goal_state_rows = [] for i in range(3):
    row = input(f"Row {i+1}: ").split()
goal_state_rows.extend(row) goal_state =
"".join(goal_state_rows)


solution = bfs(initial_state, goal_state)
 if
solution:
    print("\nBFS solution path:")
for s in solution:
print(s[:3])
print(s[3:6])
print(s[6:])          print()
else:
    print("\nNo solution found.")
```

OUTPUT:

```
1BM23CS333
Enter the initial state (enter 3 digits per row, separated by spaces, 0 for empty):
Row 1: 2 8 3
Row 2: 1 6 4
Row 3: 7 0 5

Enter the goal state (enter 3 digits per row, separated by spaces, 0 for empty):
Row 1: 1 2 3
Row 2: 8 0 4
Row 3: 7 6 5

BFS solution path:
283
164
705

283
104
765

203
184
765

023
184
765

123
084
765

123
804
765
```