1. Write a python program to trace out how the searching process works for the given graph using bidirectional Search.

Program :

```
from collections import deque


def bidirectional_search(graph, start, goal):
    forward_queue = deque([(start, [start])])
    backward_queue = deque([(goal, [goal])])

    forward_visited = set([start])
    backward_visited = set([goal])

    while forward_queue and backward_queue:
        forward_node, forward_path = forward_queue.popleft()
        backward_node, backward_path = backward_queue.popleft()

        if forward_node in backward_visited:
            intersection_node = forward_node
            intersection_path = forward_path + backward_path[::-1]
            return intersection_path
```

```
        for neighbor in graph[forward_node]:
            if neighbor not in forward_visited:
                forward_visited.add(neighbor)
                forward_queue.append((neighbor, forward_path + [neighbor]))

        if backward_node in forward_visited:
            intersection_node = backward_node
            intersection_path = forward_path + backward_path[::-1]
            return intersection_path

        for neighbor in graph[backward_node]:
            if neighbor not in backward_visited:
                backward_visited.add(neighbor)
                backward_queue.append((neighbor, backward_path + [neighbor]))

    return None  # No intersection found

def main():
    # Example graph represented as an adjacency list
    graph = {
        'A': ['B', 'C'],
        'B': ['A', 'D', 'E'],
        'C': ['A', 'F', 'G'],
        'D': ['B'],
```

```
        'E': ['B', 'H'],
        'F': ['C', 'I'],
        'G': ['C'],
        'H': ['E', 'J'],
        'I': ['F'],
        'J': ['H']
    }

    start_node = 'A'
    goal_node = 'J'

    result_path = bidirectional_search(graph, start_node, goal_node)

    if result_path:
        print(f"Path from {start_node} to {goal_node}: {result_path}")
    else:
        print("No path found between the start and goal nodes.")

if __name__ == "__main__":
    main()
```

Output :

Path from A to J: ['A', 'C', 'E', 'H', 'J']