- Problem 2:

```python
def reverse_BFS(self, start, end):
    queue = deque([end])
    visited = set([end])
    parents = {end: None}

    while queue:
        vertex = queue.popleft()
        if vertex == start:
            path = []
            while vertex is not None:
                path.append(vertex)
                vertex = parents[vertex]
            return path[::-1]

        for neighbor in self.graph[vertex]:
            if neighbor[0] not in visited:
                visited.add(neighbor[0])
                parents[neighbor[0]] = vertex
                queue.append(neighbor[0])
    return None
```

➔ This function can be applied to find the shortest path between 2 nodes in reverse order (from end to start). We initialize a “queue”, which holds all the nodes that have to be checked, a “visited” set that holds the nodes that have already been checked, and “parents” where we have the first node as a key, and the one that it goes into as a value. The functionality runs as long as the queue isn’t empty. The “vertex” takes the value of the element that’s popped out of the queue. If it’s equal to the “start” node, we have finalized generating the shortest path, so we can return it. Otherwise, we check what other edges our initial vertex is a part of, and, if the found nodes are not yet visited, we add them to the “visited” set, and into the queue so the process repeats itself.