Documentation Practical Work no. 2

Code added

Repository

```
def shortest_path_between_two_vertices_forward_breath_first_search(self, start: int, end: int) -> list:
    '''
    This function calculates the shortest path between two vertices in a graph using a forward breadth-first search approach.
    It verifies the existence of the provided vertices within the graph and returns the starting vertex if both vertices are the same.
    To keep track of visited vertices, it utilizes a set, and employs a queue to maintain the current vertex and its associated path.
    Each vertex encountered during the search is marked as visited.
    The algorithm then explores the neighbors of the current vertex, checking if any neighbor matches the end vertex;
    in such case, it returns the path. If a neighbor has not been visited before, it's added to the queue and marked as visited.
    If the end vertex is not reached after exploring all possible paths, the function returns None.
    '''
    if not ((self.is_vertex(start) and self.is_vertex(end))):
        raise RepoError("\nVertices do not exist in the graph\n")
    if start == end:
        return [start]
    visited = set()
    queue = deque([(start, [start])])
    while queue:
        current, path = queue.popleft()
        visited.add(current)
        for neighbor in self.graph.dout[current]:
            if neighbor == end:
                return path + [neighbor]
            if neighbor not in visited:
                queue.append((neighbor, path + [neighbor]))
                visited.add(neighbor)
    return None
```

Service

```
def shortest_path_between_two_vertices_forward_breath_first_search(self, start: int, end: int) -> tuple:
    """
    Returns the shortest path between two vertices using forward breath first search
    Args:
        start : the starting vertex
        end : the ending vertex
    Preconditions:
        Both vertices must exist in the graph
    """
    return self.repo.shortest_path_between_two_vertices_forward_breath_first_search(start, end)
```

UI

```
if command == "1":
    start = int(input("Enter start vertex: "))
    end = int(input("Enter end vertex: "))
    path = self.service.shortest_path_between_two_vertices_forward_breath_first_search(start, end)
    if path is None:
        print("\nThere is no path between the two vertices\n")
    else:
        print("\nLength of the path: ", len(path) - 1) # length of the path is the number of edges
        print("Path: ", end="")
        for vertex in path[:-1]:
            print(vertex, end=" -> ")
        print(path[-1])
```

Results

Graph 1k

```
Enter start vertex: 1
Enter end vertex: 100

Length of the path:  6
Path: 1 -> 5 -> 487 -> 175 -> 699 -> 624 -> 100
```

```
Enter start vertex: 100
Enter end vertex: 1

Length of the path:  5
Path: 100 -> 416 -> 354 -> 865 -> 109 -> 1
```

Graph 10k

```
Enter start vertex: 1
Enter end vertex: 100

Length of the path:  8
Path: 1 -> 3300 -> 2607 -> 523 -> 6311 -> 5359 -> 9794 -> 5173 -> 100
```

```
Enter start vertex: 100
Enter end vertex: 1

Length of the path:  7
Path: 100 -> 2398 -> 3054 -> 5232 -> 8217 -> 2478 -> 7151 -> 1
```

Graph 100k

```
Enter start vertex: 1
Enter end vertex: 100

Length of the path:  8
Path: 1 -> 17024 -> 27471 -> 14969 -> 3075 -> 4156 -> 32753 -> 14973 -> 100
```

```
Enter start vertex: 100
Enter end vertex: 1

Length of the path:  8
Path: 100 -> 44340 -> 54527 -> 6606 -> 53263 -> 95930 -> 98655 -> 58288 -> 1
```