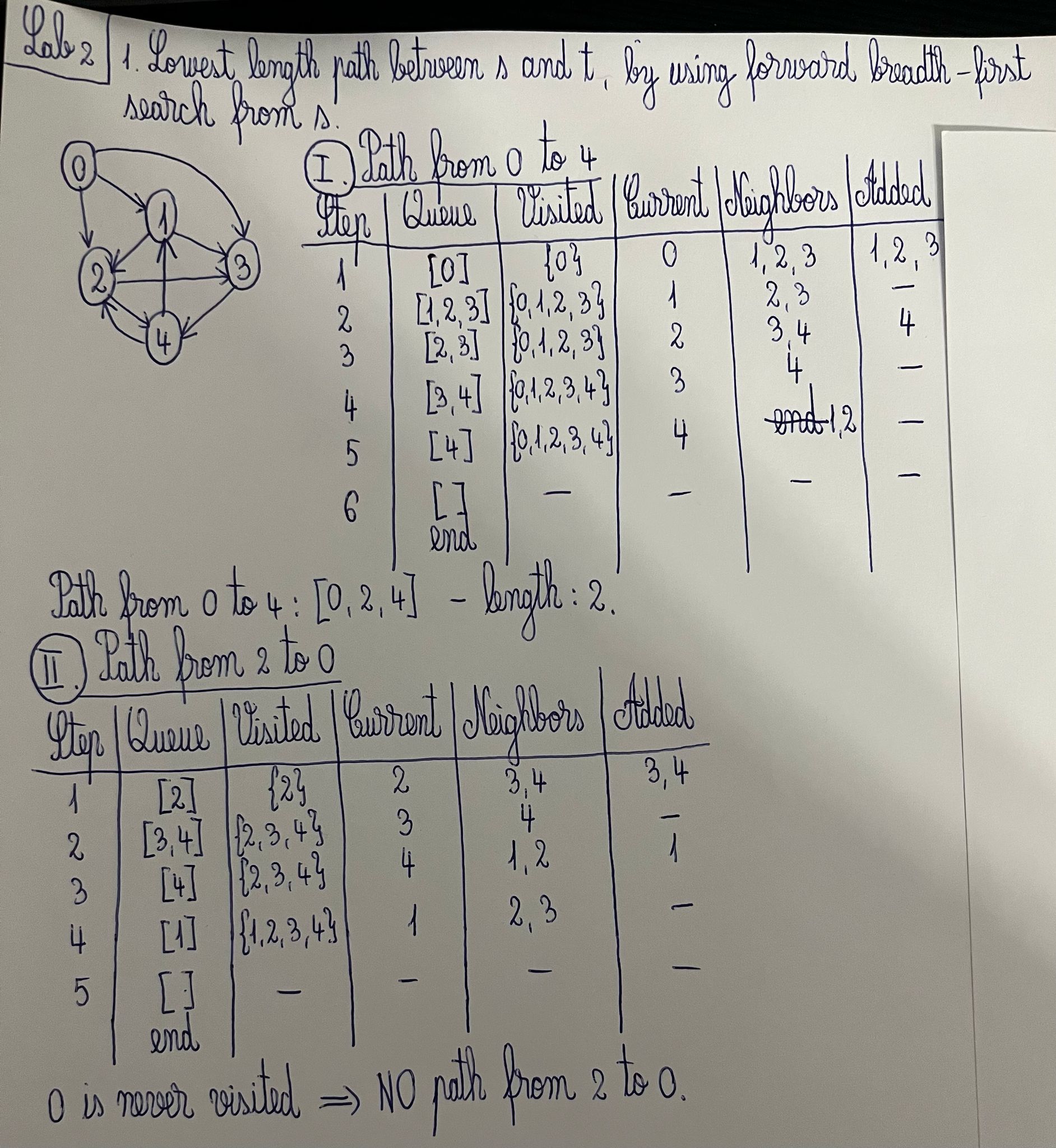
def bfs(self, start\_vertex, goal\_vertex):  
 *"""  
 A function that searches for the path between two given vertices and returns it if it exists.  
 :param start\_vertex: The starting vertex  
 :param goal\_vertex: The vertex we want to reach  
 :return: The path between the two vertices or None if there is no such path  
 """* queue = deque([start\_vertex])  
 parent = {start\_vertex: None}  
  
 while queue:  
 current\_vertex = queue.popleft()  
 if current\_vertex == goal\_vertex:  
 path = []  
 while current\_vertex is not None:  
 path.append(current\_vertex)  
 current\_vertex = parent[current\_vertex]  
 return path[::-1]  
  
 for neighbor in self.\_\_outgoing\_edges.get(current\_vertex, []):  
 if neighbor not in parent:  
 queue.append(neighbor)  
 parent[neighbor] = current\_vertex  
 return None



graph1k.txt:

The shortest path from 1 to 100 has the length = 6 and is as follows:

[1, 5, 487, 175, 699, 624, 100].

The shortest path from 100 to 1 has the length = 5 and is as follows:

[100, 416, 354, 865, 109, 1].

graph10k.txt:

The shortest path from 1 to 100 has the length = 8 and is as follows:

[1, 3300, 2607, 523, 6311, 5359, 9794, 5173, 100].

The shortest path from 100 to 1 has the length = 7 and is as follows:

[100, 2398, 3054, 5232, 8217, 2478, 7151, 1].

graph100k.txt:

The shortest path from 1 to 100 has the length = 8 and is as follows:

[1, 17024, 27471, 14969, 3075, 4156, 32753, 14973, 100].

The shortest path from 100 to 1 has the length = 8 and is as follows:

[100, 44340, 54527, 6606, 53263, 95930, 98655, 58288, 1].