**This function can be applied on a directed graph to find the path between two given vertices.**

**def is\_reachable(self, destination\_vertex, start\_vertex):**

*"""*

*performs a backwards breath-first search to check if the start vertex is reachable from the destination vertex*

*and returns the path is true*

*:param destination\_vertex: integer*

*:param start\_vertex: integer*

*:return:*

*"""*

# at start no vertices are visited

visited = [None] \* (self.count\_number\_of\_vertices())

# declare an empty queue which will be the BFS

queue = []

#the search begings from the last node (for us it will be the starting vertex)

queue.append(start\_vertex)

found = False

while queue:

# take the vertex on top of the queue

current\_vertex = queue.pop(0)

# if it is the destination node, then found = True

if current\_vertex == destination\_vertex:

found = True

break

# otherwise, BFS keeps running

for vertex in self.\_inbound\_neighbours[current\_vertex]:

if visited[vertex] is None:

queue.append(vertex)

visited[vertex] = current\_vertex

if not found:

return False

path = []

current\_vertex = destination\_vertex

while current\_vertex is not None and current\_vertex != start\_vertex:

path.append(current\_vertex)

current\_vertex = visited[current\_vertex]

if len(path) > 0:

path.append(start\_vertex)

return path

**In the UI part of the program, if the returned list of the BFS is not None, then it displays the length and the path. Otherwise a remark that there is no available path.**

**def bfs\_algorithm(self):**

start\_vertex = -10000000

end\_vertex = -100000000

while not self.\_\_graph.is\_vertex(start\_vertex):

start\_vertex = int(input("Starting vertex: "))

while not self.\_\_graph.is\_vertex(end\_vertex):

end\_vertex = int(input("Ending vertex: "))

bfs = self.\_\_graph.is\_reachable(start\_vertex, end\_vertex)

if bfs:

print(end\_vertex, "is reachable from", start\_vertex)

print("The length is:", len(bfs))

print(self.\_\_graph.is\_reachable(start\_vertex, end\_vertex))

else:

print(end\_vertex, "not reachable from", start\_vertex)

**The following cases resulted for the path from vertex 1 to vertex 100:**

* Graph1k.txt
  + 100 is reachable from 1
  + The length is: 7
  + [1, 5, 649, 453, 107, 647, 100]
* Graph10k.txt
  + 100 is reachable from 1
  + The length is: 9
  + [1, 7317, 4118, 2404, 690, 1494, 739, 4722, 100]
* Graph100k.txt
  + 100 is reachable from 1
  + The length is: 9
  + [1, 17024, 27471, 14969, 3075, 70733, 85480, 14973, 100]