Compulsory 3

Tree and Graph

Link to repository:
https://github.com/Zartok89/Compulsory-3.git

Description of the two traversal algorithms (Breadth and Depth), and why you chose to implement either one of them for graphs and trees

- I chose both because I wanted to test them and to learn how they function. Logically in my mind I would go depth first since I want to see where it ends before I start reaching into another branch.
  -Depth first is when you search one and one branch until you hit the bottom of the tree.
  Example path:

	1	
2	3	4
5		7
6		

You would go 1 → 2 → 5 → 6, then 1 → 3, then 1 → 4 → 7

-Breadth first is when you explore every child at the next level in your tree before moving onto then next level.
Example path:

	1	
2	3	4
5		7
6		

You would go 1, then 2 → 3 → 4, then 5 → 7, then 6

Reason for choosing to implement an adjacency list or an adjacency matrix for graphs.

- I chose to go for adjacency list rather than a matrix. For me it is visually easier to see where the vertex pathing is going. It also makes sense if I were to represent it as a map of train stations or so to have shown which vertices are connect through the edges rather than a list of 0 and 1s in a matrix grid