Question 3:

Here,

We are given that the graph is unweighted and undirected. In this case, we can consider the weight of each edge to be 1 (i.e., every connection is of 1 weight). This means that the shortest path is always the path that goes through the least number of vertices.

This means we can utilize a breadth first search where the first iteration is of weight 1, second iteration is of weight 2 and so on.

To do this, we also need to keep track of visited and unvisited vertices while storing the result for the shortest path somewhere. We will utilize an unordered map with <int, int> for it. The key is an int representing each vertex and value is an int representing the distance from the given vertex.

We start by adding all the vertices into the map with a value = -1. We then make a vertices list which will contain all vertices in the map and when all vertices are added, this means we have discovered all vertices.

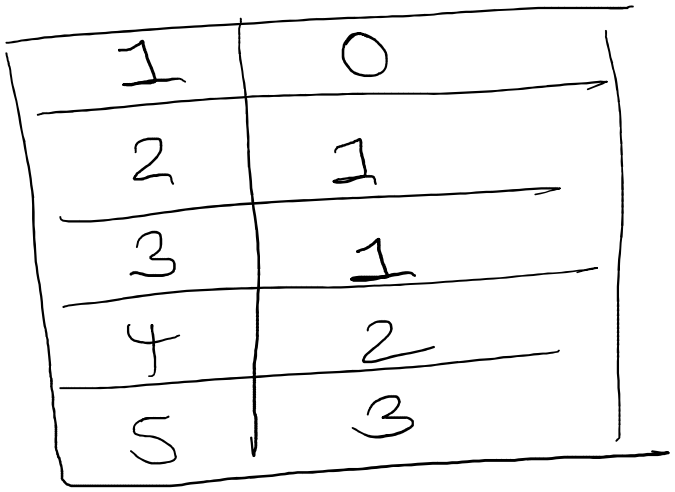
We also have an integer called step that increments by 1 for every big iteration of the BFS.

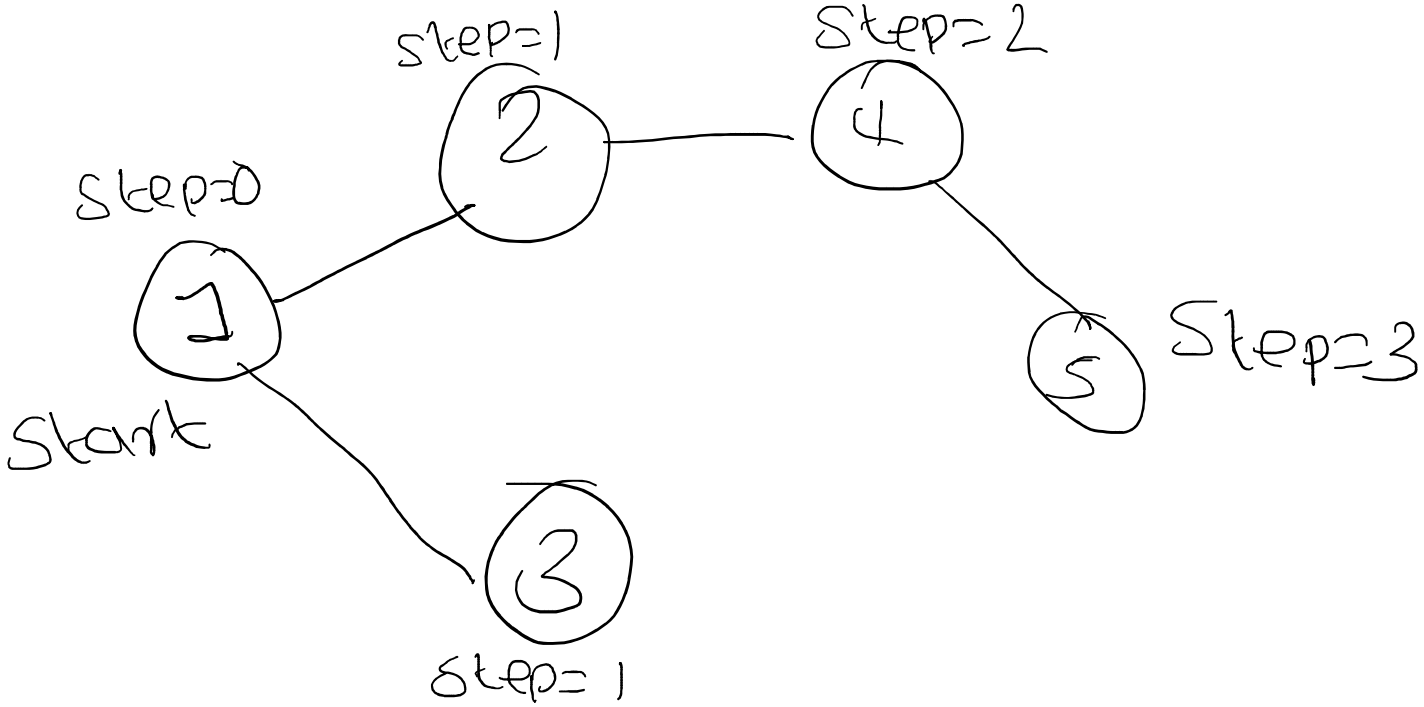
Pseudocode:

```
Int startVertex = vertex where we start from;
Unordered_map<int, int> gr;
Add all vertices to the map with value = -1;
VerticesList = empty list of vertices;
AdjVertices = adjacent vertices to startVertex;
Int step = 0;
Q = Queue
Put startVertex into Q
Put startVertex into gr with value = step = 0 ;
Int count = 0;
while (Q is non-empty)
{
auto it = gr.find(head of queue);
        it->second = step;
        remove the head u of Q
        Enqueue all (unvisited) neighbors of u and when enqueuing add into the map gr with a value of
        step + 1 and count the number of items enqueued as count
        Use count to do step = step + 1;
}
```

Breadth first search takes O(V+E) time

Table gr




step=1
Step=2
step=0
1
2
4
5
Step=3
Start
3
step=1

QUESTION 4

The average is 58.3634

The average degree of all of the adjacent vertices is double of the average degree over all the vertices in the map.