**CSE 207**

**DATA STRUCTURES & ALGORITHMS II SESSIONAL**

**OFFLINE: 01**

**TITLE:** IMPLEMENTATION OF GRAPH DATA STRUCTURE, IN ADJACENT MATRIX AND ADJACENT LIST FORMAT

**SUBMITTED BY:**

Kawshik Kumar Paul

STUDENT NO: 1705043

LEVEL – 2, TERM – 2

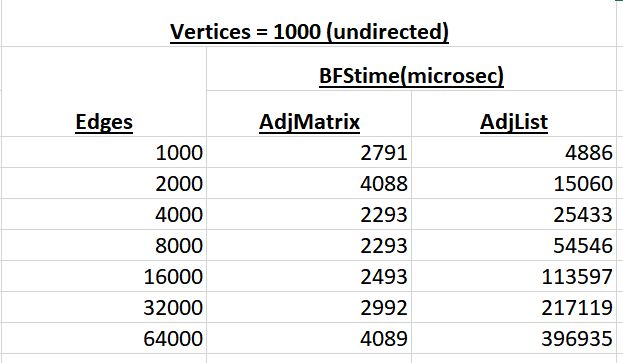
DEPARTMENT OF CSE

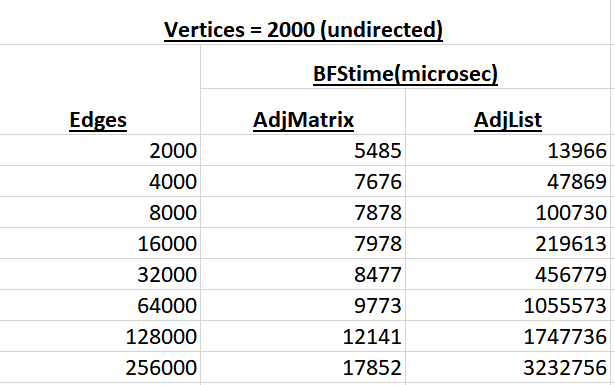
BUET

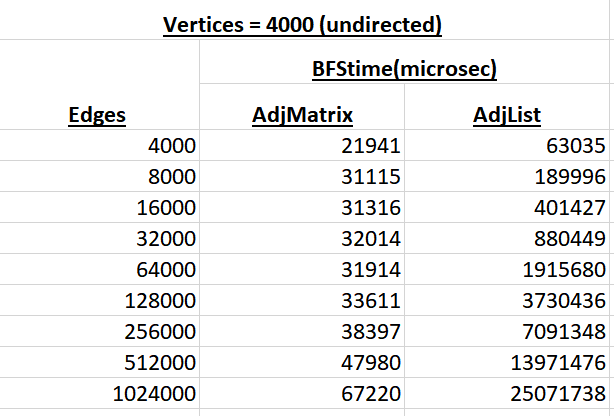
**Machine Configuration:**  
 Processor: Intel Core i7 (8th Gen)

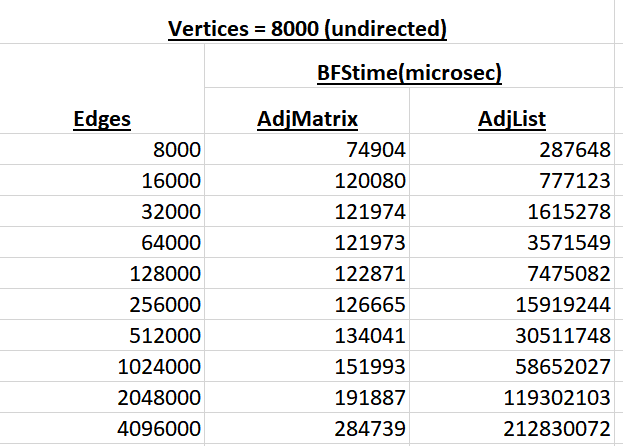
Ram: 8GB  
 OS: Windows 10 64Bit  
 Compiler: C++14

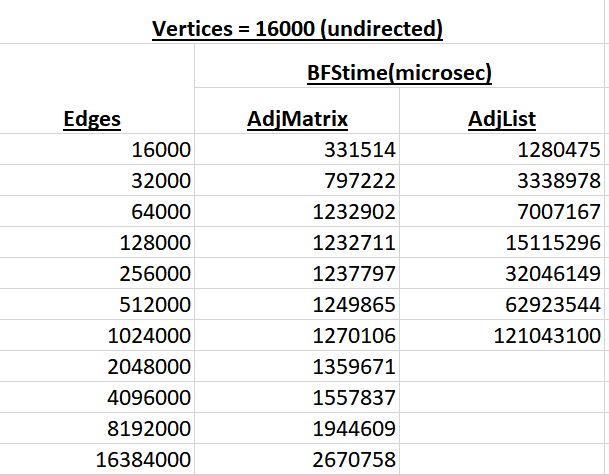
**Data Table:**  Here are examples for undirected graph











**QUESTION & ANSWER:**

**QN1: What is the impact on runtime if we keep |V| unchanged and double |E| for adjacency list? Why is it so?**  
**Ans:** If we keep |V| unchanged and double |E| for adjacency list, The runtime will be almost double. Because in adjacency list, searching an edge occurs in complexity O(n), so increasing of edges causes runtime to increase. If we use double quantity edges, searching time will be doubled and running time will also be doubled.   
  
**QN2: What is the impact on runtime if we keep |E| unchanged and double |V| for adjacency list? Why is it so?**  
**Ans:** If we keep |E| unchanged and double |V| for adjacency list, the runtime will be double. In this case searching time of edge from a vertex will be same , but the loop by the increased quantity of vertices will be bigger and it will make the runtime almost double.   
  
**QN3: What is the impact on runtime if we keep |V| unchanged and double |E| for adjacency matrix? Why is it so?**  
**Ans:** If we keep |V| unchanged and double |E| for adjacency matrix, the runtime will also become about double. This is because, when we double the number of vertices, but still keep a fixed number of edges, the searchItem function for the arrayList does not take more time, as the number of edges is fixed. But, as we increase the number of vertices, the complexity increases. Because, if BFS operation, we check if each vertex has an edge with each vertex we’re operating on. As the number of vertex increases, this loop becomes bigger. So, as we double the number of vertices, the time required also doubles, leading to almost double the time in double vertices scenarios.  
  
**QN4: What is the impact on runtime if we keep |E| unchanged and double |V| for adjacency matrix? Why is it so?  
Ans:** Running time becomes almost 4times higher.

This is because, when the number of vertices are doubled, the BFS function runs higher. For each vertex the function visits, it runs a loop up to the total number of vertices to check if there was an edge between them. So, as the number of vertices increase, this loop runs at almost double the time. And, as this is an adjacent matrix, if the number of vertices are doubled, the size of the matrix becomes 4 times the previous size. This means, that we have to check 4 times more cells than the previous number of vertices. This means, that the runtime must also increase by 4times the previous time, which is exactly the case. And for undirected graph, we have to check both ways to check if there is an edge between two vertices. So, slightly more time is needed.

**QN5. For the same |E| and |V|, why are the runtimes for adjacency list and adjacency matrix representation different? Which one is higher and why?**  
**Ans:** For the same |E| and |V|, the bfs running time of adjacency list and adjacency matrix is different. Adjacency list requires more time than Adjacency matrix. This is because :

In adjacency matrix representation, we do this search by just checking specific cells of the matrix we created and stored this information in. Accessing a specific cell in a matrix is a constant time (O(1)) operation.

But for adjacency list representation, we do this search by running the searchItem function, which is a function which checks the whole arrayList that we are working with at that moment to check if a specific edge exists. So, this function has to traverse the whole list to find the edge, which is a linear time (O(n)) operation.  
So BFS running time of adjacency list is higher than adjacency matrix.