## horizontal line

CSE 208

**DATA STRUCTURES & ALGORITHMS (II)**

**─  
  
Offline Report on Graph Algorithm (BFS)**

**Submitted by:  
  
Ataf Fazledin Ahamed**

**1705066**

**Department of CSE, BUET**

# Objective

This offline experiment was given to us so that we can understand how basic Graph algorithms, such as - BFS work. In addition to that, a data representation has been asked to show the runtime for BFS algorithm with respect to different numbers of Vertices and Edges.

# Goals

1. Generate run time for BFS algorithm using <chrono> library.
2. Plot the time found for different values of vertices and edges in a table.

# Device Specification

* Processor: Intel i3 8th Gen, 3.4 GHz
* RAM: DDR3 4GB
* Operating System: Windows 10 Ultimate
* Compiler Used: MinGW

# 

# 

# Data Table (List)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **GraphAdjList (Time in Microseconds)** | | | | | |
| **E/V** | **1000** | **2000** | **4000** | **8000** | **16000** |
| **|V|** | 21 | 40 | 51 | 96 | 171 |
| **2|V|** | 22 | 44 | 67 | 97 | 194 |
| **4|V|** | 34 | 44 | 55 | 95 | 130 |
| **8|V|** | 39 | 33 | 56 | 68 | 116 |
| **16|V|** | 27 | 29 | 30 | 61 | 117 |
| **32|V|** | 24 | 24 | 33 | 61 | 107 |
| **64|V|** | 36 | 27 | 47 | 52 | 114 |
| **128|V|** | - | 20 | 43 | 55 | 132 |
| **256|V|** | - | 25 | 32 | 40 | 108 |
| **512|V|** | - | - | - | 52 | 155 |
| **1024|V|** | - | - | - | - | 138 |

# Data Table (Matrix)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **GraphAdjMatrix (Time in Microseconds)** | | | | | |
| **E/V** | **1000** | **2000** | **4000** | **8000** | **16000** |
| **|V|** | 2206 | 6899 | 27235 | 97885 | 474614 |
| **2|V|** | 2627 | 8625 | 33679 | 135153 | 557421 |
| **4|V|** | 2791 | 8831 | 34617 | 151994 | 567377 |
| **8|V|** | 2429 | 9127 | 36922 | 137731 | 555453 |
| **16|V|** | 3065 | 9198 | 35080 | 144486 | 553845 |
| **32|V|** | 3823 | 9735 | 36109 | 154479 | 555411 |
| **64|V|** | 3573 | 11009 | 38176 | 155853 | 586170 |
| **128|V|** | - | 14022 | 43319 | 173678 | 608466 |
| **256|V|** | - | 20594 | 55129 | 220323 | 612954 |
| **512|V|** | - | - | - | 220891 | 712599 |
| **1024|V|** | - | - | - | - | 874160 |

# Data Table (Matrix)

**1. What is the impact on runtime if we keep |V| unchanged and double |E| for adjacency list? Why is it so?**

Answer:

The runtime for the algorithm increases by a little in the beginning. But with time, the runtime decreases a bit and after a certain period, it increases back. The runtime doesn’t follow a linear relation with the sample size.   
  
It happens because the edges were generated randomly. Since the datasets are huge, there is a high chance of having a disconnected graph. And since the source was also chosen randomly, the BFS algorithm had only traversed a small part of the graph.

**2. What is the impact on runtime if we keep |E| unchanged and double |V| for adjacency list? Why is it so?**

Answer:

The runtime follows a linear relation here. Since there are less number of edges here and more vertices, the number of paths between two vertices becomes less and less. And when we run a BFS algorithm, the traversing takes more time because there are less paths and more vertices. Sometimes, it has to traverse the whole graph to complete the search. Which results in getting longer time.

**3. What is the impact on runtime if we keep |V| unchanged and double |E| for adjacency matrix? Why is it so?**

Answer:

If we double |V|, at lower vertices number, the time becomes a little lower than double, but as the edges are higher in number, the runtime hits a certain limit which doesn’t ends in a linear relation all the way.

In BFS, the algorithm searches the adjacent matrix in a constant time having complexity of O(1). And when the edges are doubled, the values for vertices increase. Thus, searching the matrix takes more time than the previous one.

**4. What is the impact on runtime if we keep |E| unchanged and double |V| for adjacency matrix? Why is it so?**

Answer:

The runtime increases in an exponential relation for keeping |E| unchanged and making |V| doubled.

As the vertices double, the path decreases with respect to the whole graph size. Since there are less paths, the algorithm takes more time to search through the graph than usual. Also, the algorithm has to iterate over a large number of vertices in the adjacency matrix, so it results in an increasing runtime.

**5. For the same |E| and |V|, why are the runtimes for adjacency list and adjacency matrix representation different? Which one is higher and why?**

Answer:

From the tables, it is clear that the runtime for the adjacency matrix is much, much higher than the adjacency list. The ratio follows an exponential relation in between.

The reason for this is the implementation of the adjacency matrix and list. For the adjacency matrix, it is slower to iterate over the neighboring edges. The complexity becomes O(n).   
  
In the adjacency list, it takes less time to iterate over the neighboring edges because it maintains a dynamic list system. Which takes less time than adjacency matrix. When the size of the graph increases, the overall runtime of BFS increases on a larger scale.