**CSE 208**

**Graph OFFLINE**

**Runtime Analysis of BFS**

**Submitted by:**

**Saem Hasan**

**L-2/T-2**

**1705027**

**CSE, BUET**

**Machine Configuration:**

Processor: Intel® Core(TM) i7-7500U CPU @ 2.70GHz 2.90GHz

RAM : 8.00 GB (7.80 GB usable)

Operating System : 64 bit Windows 10 pro

**Runtime of analysis of BFS:**

**For adjacency list representation:**

|  |  |  |
| --- | --- | --- |
| Num. of Vertices | Num. of Edges | Time(microseconds) |
| 1000 | 1000 | 55.6 |
| 2000 | 99.8 |
| 4000 | 99.8 |
| 8000 | 99.8 |
| 16000 | 99.8 |
| 32000 | 99.8 |
| 64000 | 99.8 |
| 2000 | 2000 | 99.7 |
| 4000 | 199.4 |
| 8000 | 299.2 |
| 16000 | 299.2 |
| 32000 | 299.2 |
| 64000 | 299.2 |
| 128000 | 398.7 |
| 256000 | 398.7 |
| 4000 | 4000 | 99.7 |
| 8000 | 199.5 |
| 16000 | 299.2 |
| 32000 | 299.2 |
| 64000 | 299.2 |
| 128000 | 299.2 |
| 256000 | 299.2 |
| 512000 | 299.2 |
| 1024000 | 498.8 |
| 8000 | 8000 | 399.2 |
| 16000 | 801.1 |
| 32000 | 1000.6 |
| 64000 | 1000.6 |
| 128000 | 1000.6 |
| 256000 | 1399.6 |
| 512000 | 1698.8 |
| 1024000 | 2496.7 |
| 2048000 | 3291.5 |
| 4096000 | 3990.2 |
| 16000 | 16000 | 797.8 |
| 32000 | 1399 |
| 64000 | 2293.4 |
| 128000 | 3091.2 |
| 256000 | 4285.1 |
| 512000 | 5781.1 |
| 1024000 | 9970 |
| 2048000 | 14457.8 |
| 4096000 | 25428.4 |
| 8192000 | 61535.3 |
| 16384000 | 121079 |

Table no: 01

**For adjacency matrix representation:**

|  |  |  |
| --- | --- | --- |
| Num. of Vertices | Num. of Edges | Time(microseconds) |
| 1000 | 1000 | 1695.8 |
| 1000 | 2000 | 2294.3 |
| 1000 | 4000 | 2892.7 |
| 1000 | 8000 | 3391.4 |
| 1000 | 16000 | 3391.4 |
| 1000 | 32000 | 3590.9 |
| 1000 | 64000 | 3790.4 |
| 2000 | 2000 | 4687.4 |
| 2000 | 4000 | 7180.8 |
| 2000 | 8000 | 8078.6 |
| 2000 | 16000 | 9774.1 |
| 2000 | 32000 | 9774.1 |
| 2000 | 64000 | 9774.1 |
| 2000 | 128000 | 9774.1 |
| 2000 | 256000 | 9774.1 |
| 4000 | 4000 | 21040.6 |
| 4000 | 8000 | 35601.8 |
| 4000 | 16000 | 42882.4 |
| 4000 | 32000 | 46472.8 |
| 4000 | 64000 | 50063.3 |
| 4000 | 128000 | 57643.1 |
| 4000 | 256000 | 61134.2 |
| 4000 | 512000 | 61134.2 |
| 4000 | 1024000 | 61134.2 |
| 8000 | 8000 | 85973.7 |
| 8000 | 16000 | 149204 |
| 8000 | 32000 | 190893 |
| 8000 | 64000 | 217725 |
| 8000 | 128000 | 258914 |
| 8000 | 256000 | 286245 |
| 8000 | 512000 | 326338 |
| 8000 | 1024000 | 409212 |
| 8000 | 2048000 | 435937 |
| 8000 | 4096000 | 464262 |
| 16000 | 16000 | 362235 |
| 16000 | 32000 | 684969 |
| 16000 | 64000 | 960034 |
| 16000 | 128000 | 1.18E+06 |
| 16000 | 256000 | 1.46E+06 |
| 16000 | 512000 | 1.84E+06 |
| 16000 | 1024000 | 2.11E+06 |
| 16000 | 2048000 | 2.54E+06 |
| 16000 | 4096000 | 2.87E+06 |
| 16000 | 8192000 | 3.11E+06 |
| 16000 | 16384000 | 3.31E+06 |

Table no: 02

|  |  |  |  |
| --- | --- | --- | --- |
| Num. of Vertices | Number of Edges | Time for list(microseconds) | Time for matrix(microseconds) |
| 1000 | 1000 | 55.6 | 1695.8 |
| 1000 | 2000 | 99.8 | 2294.3 |
| 1000 | 4000 | 99.8 | 2892.7 |
| 1000 | 8000 | 99.8 | 3391.4 |
| 1000 | 16000 | 99.8 | 3391.4 |
| 1000 | 32000 | 99.8 | 3590.9 |
| 1000 | 64000 | 99.8 | 3790.4 |
| 2000 | 2000 | 99.7 | 4687.4 |
| 2000 | 4000 | 199.4 | 7180.8 |
| 2000 | 8000 | 299.2 | 8078.6 |
| 2000 | 16000 | 299.2 | 9774.1 |
| 2000 | 32000 | 299.2 | 9774.1 |
| 2000 | 64000 | 299.2 | 9774.1 |
| 2000 | 128000 | 398.7 | 9774.1 |
| 2000 | 256000 | 398.7 | 9774.1 |
| 4000 | 4000 | 99.7 | 21040.6 |
| 4000 | 8000 | 199.5 | 35601.8 |
| 4000 | 16000 | 299.2 | 42882.4 |
| 4000 | 32000 | 299.2 | 46472.8 |
| 4000 | 64000 | 299.2 | 50063.3 |
| 4000 | 128000 | 299.2 | 57643.1 |
| 4000 | 256000 | 299.2 | 61134.2 |
| 4000 | 512000 | 299.2 | 61134.2 |
| 4000 | 1024000 | 498.8 | 61134.2 |
| 8000 | 8000 | 399.2 | 85973.7 |
| 8000 | 16000 | 801.1 | 149204 |
| 8000 | 32000 | 1000.6 | 190893 |
| 8000 | 64000 | 1000.6 | 217725 |
| 8000 | 128000 | 1000.6 | 258914 |
| 8000 | 256000 | 1399.6 | 286245 |
| 8000 | 512000 | 1698.8 | 326338 |
| 8000 | 1024000 | 2496.7 | 409212 |
| 8000 | 2048000 | 3291.5 | 435937 |
| 8000 | 4096000 | 3990.2 | 464262 |
| 16000 | 16000 | 797.8 | 362235 |
| 16000 | 32000 | 1399 | 684969 |
| 16000 | 64000 | 2293.4 | 960034 |
| 16000 | 128000 | 3091.2 | 1.18E+06 |
| 16000 | 256000 | 4285.1 | 1.46E+06 |
| 16000 | 512000 | 5781.1 | 1.84E+06 |
| 16000 | 1024000 | 9970 | 2.11E+06 |
| 16000 | 2048000 | 14457.8 | 2.54E+06 |
| 16000 | 4096000 | 25428.4 | 2.87E+06 |
| 16000 | 8192000 | 61535.3 | 3.11E+06 |
| 16000 | 16384000 | 121079 | 3.31E+06 |

Table no: 03

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

|  |  |  |
| --- | --- | --- |
| Num. of Vertices | Num. of Edges | Time(microseconds) |
| 1000 | 1000 | 55.6 |
| 1000 | 2000 | 99.8 |
| 2000 | 2000 | 99.7 |
| 2000 | 4000 | 199.4 |
| 4000 | 4000 | 99.7 |
| 4000 | 8000 | 199.5 |
| 8000 | 8000 | 399.2 |
| 8000 | 16000 | 801.1 |
| 16000 | 16000 | 797.8 |
| 16000 | 32000 | 1399 |

If we keep V unchanged and double E for adjacency list, then the runtime will be almost 2 times. Using adjacency list for BFS, The time complexity is O (n+m). So if we increase the number of edges, the runtime will increase. As we double the number of edges, the runtime will be almost double. From the table we can see the same result. As we double the edges, the runtime increases by factor 2.

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

|  |  |  |
| --- | --- | --- |
| No of Vertices | No. of Edges | Time(microseconds) |
| 4000 | 32000 | 299.2 |
| 8000 | 32000 | 700.6 |
|  |  |  |
| 8000 | 256000 | 1399.6 |
| 16000 | 256000 | 2946.1 |

Using adjacency list for BFS implementation, the time complexity is O (n+m).

So if we keep E unchanged and double V for adjacency list, the runtime will be almost double. As the runtime depends on the no of vertices, so the runtime will be increased.

As the value of n is increased, the runtime also increases.

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

|  |  |  |
| --- | --- | --- |
| Num. of Vertices | Num. of Edges | Time(microseconds) |
| 1000 | 1000 | 1695.8 |
| 1000 | 2000 | 2294.3 |
| 2000 | 2000 | 4687.4 |
| 2000 | 4000 | 7180.8 |
| 4000 | 4000 | 21040.6 |
| 4000 | 8000 | 35601.8 |
| 8000 | 8000 | 85973.7 |
| 8000 | 16000 | 149204 |
| 16000 | 16000 | 362235 |
| 16000 | 32000 | 684969 |

Using adjacency matrix for BFS, the time complexity is O (n2). So In adjacency matrix representation the runtime will be almost same for any number of edges. So if we double edges for adjacency matrix, then the runtime should be almost same. But from the table we can see that the runtime is higher when the edges is doubled. As the no of edges is lower w.r.t. vertices, there will be a lot of disconnected graph. That’s why we can see difference in runtime. When the number of edges will increase enough, then the runtime will be almost same.

For example:

|  |  |  |
| --- | --- | --- |
| Num. of Vertices | Num. of Edges | Time(microseconds) |
| 8000 | 2048000 | 435937 |
| 8000 | 4096000 | 464262 |

From the table we can see that the runtime is almost same.

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

If we keep E unchanged and double V for adjacency matrix, the runtime will increase in a factor of 4.

We know, using adjacency matrix for BFS, time complexity is O (n2). The runtime is proportional to the square of the no of vertices. As we double the no of vertices, the runtime will be increased by factor 4.

|  |  |  |
| --- | --- | --- |
| Num. of Vertices | Num. of Edges | Time(microseconds) |
| 2000 | 32000 | 9774.1 |
| 4000 | 32000 | 46472.8 |
| 8000 | 32000 | 190893.2 |

From the table we can also see that the runtime is increased by factor 4 when the no of vertices is doubled.

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?

In BFS, we need all the adjacent vertices of a vertex. So to keep track of the adjacent vertices, we use two representation: 1. Adjacency list 2. Adjacency Matrix

In adjacency list, the no of operation needed to find all of the adjacent vertices is equal to the no of adjacent vertices. That’s why the time complexity using adjacency list is O (n + m).

In adjacency matrix, the no of operation needed to find all of the adjacent vertices is equal to the no of vertices of the graph. So we need to go to all of the vertices of the graph in adjacency matrix. That’s why the time complexity of BFS using adjacency matrix is O (n2).

The runtime is higher in matrix representation because in matrix representation we need to visit all the vertices to find the adjacent vertices. That’s why, in most of the cases the runtime in list representation is lower. We can see the same result in table no.03. In all cases the runtime in matrix representation is much higher than adjacency list representation.