

**CSE 208: Data Structures and Algorithms Sessional II**

Offline 1

**RUNTIME ANALYSIS OF BFS**

Submitted by:

Name: Jayanta Sadhu

Student ID: 1705047

Department: Computer Science and Engineering(CSE)

Report on:

Running Time of BFS algorithm on Adjacent List and Adjacent Matrix Graph Representation

|  |  |  |  |
| --- | --- | --- | --- |
| **NO. OF VERTICES** | **NO. OF EDGES** | **TIME FOR ADJ. LIST(nanosecond)** | **TIME FOR ADJ. MATRIX(nanosecond)** |
| 1000 | 1000 | 199400 | 1.4963e+006 |
| 2000 | 99800 | 2.0896e+006 |
| 4000 | 176400 | 2.2952e+006 |
| 8000 | 199600 | 2.2943e+006 |
| 16000 | 537100 | 2.3936e+006 |
| 32000 | 603200 | 4.2885e+006 |
| 64000 | 1.2966e+006 | 6.3791e+006 |
| 2000 | 2000 | 138600 | 1.1068e+007 |
| 4000 | 204200 | 1.13694e+007 |
| 8000 | 309900 | 1.17685e+007 |
| 16000 | 504300 | 1.25698e+007 |
| 32000 | 703400 | 1.38629e+007 |
| 64000 | 2.5794e+006 | 1.57578e+007 |
| 128000 | 2.8205e+006 | 1.95477e+007 |
| 256000 | 8.3892e+006 | 3.36099e+007 |
| 4000 | 4000 | 299100 | 4.65753e+007 |
| 8000 | 1.0973e+006 | 4.77771e+007 |
| 16000 | 1.0274e+006 | 4.77771e+007 |
| 32000 | 1.5104e+006 | 4.65717e+007 |
| 64000 | 2.5065e+006 | 4.87693e+007 |
| 128000 | 4.5192e+006 | 5.45541e+007 |
| 256000 | 8.5144e+006 | 5.92415e+007 |
| 512000 | 1.67349e+007 | 6.34303e+007 |
| 1024000 | 1.92522e+007 | 7.59967e+007 |
| 8000 | 8000 | 598100 | 1.27458e+008 |
| 16000 | 993400 | 1.33341e+008 |
| 32000 | 1.4958e+006 | 1.53286e+008 |
| 64000 | 1.8951e+006 | 1.84211e+008 |
| 128000 | 3.0854e+006 | 1.96375e+008 |
| 256000 | 5.5305e+006 | 1.83011e+008 |
| 512000 | 1.06025e+007 | 1.83206e+008 |
| 1024000 | 2.07729e+007 | 1.91382e+008 |
| 2048000 | 4.1544e+007 | 2.48934e+008 |
| 4096000 | 1.26838e+008 | 2.55516e+008 |
| 16000 | 16000 | 2.2947e+006 | 4.69241e+008 |
| 32000 | 3.4889e+006 | 6.3281e+008 |
| 64000 | 4.8859e+006 | 6.45572e+008 |
| 128000 | 6.7556e+006 | 6.64725e+008 |
| 256000 | 1.15369e+007 | 7.04515e+008 |
| 512000 | 2.10223e+007 | 6.49462e+008 |
| 1024000 | 7.16766e+007 | 6.8387e+008 |
| 2048000 | 7.75807e+007 | 6.72201e+008 |
| 4096000 | 1.83042e+008 | 7.16682e+008 |
| 8192000 | 2.69284e+008 | 8.31874e+008 |
| 16384000 | 5.22351e+008 | 1.06316e+009 |

**Questions and Answers:**

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

**Answer:**

If we keep |V| unchanged and double the edges, then the graph will turn into a dense graph from a sparse graph. The complexity of BFS is O (V+E). For adjacency list representation O(E) is tends to O(V) for sparse graph and O(V2) for dense graph. As the graph will turn into dense graph by doubling the edges, the running time will be changed to O(V2).

From the table, we can see that the running time is increasing when we are doubling the edges (keeping vertex number fixed)

VERTEX: 8000

Time for 16000 edges: 993400 nanoseconds

“ “ 32000 “ : 1.4958e+006 nanoseconds

“ “ 64000 “ : 1.8951e+006 nanoseconds

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

**Answer:**

The running time of BFS is O (V+E). If |V| is doubled by keeping the edge number constant then running time complexity of BFS will be O(V), since the graph will become more and more sparse.

Example: (from table)

EDGES: 8000

Time for 1000 vertices: 199600 nanoseconds

“ “ 2000 “ : 309900 nanoseconds

“ “ 4000 “ : 1.0973e+006 nanoseconds

Here we can see that the complexity doubles if V is doubled.

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

**Answer:**

There will not be much impact on the running time of BFS if we double the edges keeping the number vertices fixed. Because O(E) tends to O(V2) in the adjacency matrix representation. So the complexity of BFS is O(V2). If vertex is fixed then it does not matter whether we increase the edges or not.

From the report, we can see that if vertex is fixed for adjacency matrix, the running time is almost same though we are doubling the edges.

VERTEX: 8000

Time for 16000 edges: 1.33341e+008 nanoseconds

“ “ 32000 “ : 1.53286e+008 nanoseconds

“ “ 64000 “ : 1.84211e+008 nanoseconds

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

**Answer:**

The running time will increase rapidly if we keep edge unchanged and keep doubling the vertices. The complexity of BFS using adjacency matrix is O(V2). As we are doubling the vertex, the runtime will increase. (will be 4 times if we double vertices)

For example:

EDGES: 4000

Time for 1000 vertices: 2.2952e+006 nanoseconds

“ “ 2000 “ : 1.13694e+007 nanoseconds

“ “ 4000 “ : 4.65753e+007 nanoseconds

So it is evident that for same number of edges, the time becomes almost 4 times if we double the number of vertices.

* **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:**

The runtimes of BFS algorithm depends on the data structure used. Typically, the running time of BFS is O(V+E). But in adjacency list algorithm, the edges require O(|V|) time to traverse in sparse graph and O(|V|2) in dense graph. But the complexity is the same for both sparse and dense graph in the case of adjacency matrix, which is O(V|2). So on average the run time for BFS in the case of adjacency list is less than that of adjacency matrix. This is quite evident from the data in the table.