MACHINE CONFIGURATION:

Processor: Intel(R) Core(TM) i3-7100U CPU

2.40GHz

RAM: 4.00 GB DDR4

Operating System: Windows 8 64-bit

**RUNTIME ANALYSIS OF BFS:**

**For |V| = 1000:**

|  |  |  |
| --- | --- | --- |
| *Number of Edges (|E|)* | *Average Runtime for Adj. Matrix (micro second)* | *Average runtime for Adj. List(micro second)* |
| 1000 | 0 | 0 |
| 2000 | 3128.7 | 0 |
| 4000 | 12501.3 | 0 |
| 8000 | 23823.4 | 1562.8 |
| 16000 | 48933 | 4688.1 |
| 32000 | 103291.6 | 10941.6 |
| 64000 | 229600.5 | 41226.4 |

**For |V| = 2000:**

|  |  |  |
| --- | --- | --- |
| *Number of Edges (|E|)* | *Average Runtime for Adj. Matrix (micro second)* | *Average runtime for Adj. List(micro second)* |
| 2000 | 0 | 0 |
| 4000 | 18797.5 | 0 |
| 8000 | 48803 | 799.9 |
| 16000 | 94540.8 | 2802.5 |
| 32000 | 194008 | 7203.2 |
| 64000 | 401192.3 | 23603.9 |
| 128000 | 823164.9 | 81767 |

**For |V| = 4000:**

|  |  |  |
| --- | --- | --- |
| *Number of Edges (|E|)* | *Average Runtime for Adj. Matrix (micro second)* | *Average runtime for Adj. List(micro second)* |
| 4000 | 0 | 0 |
| 8000 | 46797.3 | 0 |
| 16000 | 188928 | 1565.4 |
| 32000 | 382717.6 | 6251 |
| 64000 | 765923.4 | 14067.3 |
| 128000 | 1533901.7 | 48678.6 |

**For |V| = 8000:**

|  |  |  |
| --- | --- | --- |
| *Number of Edges (|E|)* | *Average Runtime for Adj. Matrix (micro second)* | *Average runtime for Adj. List(micro second)* |
| 8000 | 1559.7 | 0 |
| 16000 | 279492.5 | 1560 |
| 32000 | 683398.2 | 3127.8 |
| 64000 | 1546814.4 | 10936 |
| 128000 | 3073508.5 | 31250.2 |

**For |V| = 16000:**

|  |  |  |
| --- | --- | --- |
| *Number of Edges (|E|)* | *Average Runtime for Adj. Matrix (micro second)* | *Average runtime for Adj. List(micro second)* |
| 16000 | 3200 | 1547.8 |
| 32000 | 1156189 | 1565.7 |
| 64000 | 3007406.6 | 9379.4 |
| 128000 | 6185702.9 | 23440.5 |

**Questions and Answers:**

**Q1.** What is the impact on runtime if we keep |V| unchanged and double |E| for adjacency list? Why is it so?

Answer: We know, the running time of BFS for adjacency list representation is O(|V|+|E|) . So, if we keep |V| unchanged and double |E| the running time will increase.

If we look at |V|=8000 for,

|E|=64000 running time is: 10936 microseconds

|E|=128000 running time is: 31250.2 microseconds

At |V|=16000 for,

|E|=64000 running time is: 9379.4 microseconds

|E|=128000 running time is: 23440.5 microseconds

Here it is seen in the collected data.

**Q2.** What is the impact on runtime if we keep |E| unchanged and double |V| for adjacency list? Why is it so?

Answer: We know, the running time of BFS for adjacency list representation is O(|V|+|E|) . So, if we keep |E| unchanged and double |V| the running time will increase.

If we look at |E|=64000 for,

|V|=8000 running time is: 10936 microseconds

|V|=16000 running time is: 9379.4 microseconds

Maybe for some technical flaw the collected data is showing the opposite behavior. I ran my code on another device and got the expected result.

**Q3.** What is the impact on runtime if we keep |V| unchanged and double |E| for adjacency matrix? Why is it so?

Answer: We know, the running time of BFS for adjacency matrix representation is O(|V|2) . Here the cardinality of vertices theoretically has no effect on the running time. But there may have been some technical flaws in the library function. So, the running time kept increasing. One of my friend used QueryPerformenceFrequency and got the expected result.

If we look at |V|=4000 for,

|E|=32000 running time is: 382717.6 microseconds

|E|=64000 running time is: 765923.4 microseconds

**Q4.** What is the impact on runtime if we keep |E| unchanged and double |V| for adjacency matrix? Why is it so?

Answer: We know, the running time of BFS for adjacency matrix representation is O(|V|2) . In our experiment the runtime did not increase in a quadratic manner but |V|2 is surely an upper bound of it.

If we look at |E|=64000 for,

|V|=8000 running time is: 1546814.4 microseconds

|V|=16000 running time is: 3007406.6 microseconds

So, the running time will increase very fast.

**Q5.** 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: For running BFS in adjacency list representation, we have to first initialize all the vertices and then have to run a loop which covers all the edges. So, it’s runtime is O(|V| + |E|).

On the other hand, when it comes to adjacency matrix representation to run a BFS we have to first initialize all the vertices as before but on the next step we have to cover all the edges. For this reason we have to iterate over the 2D |V| x |V| matrix that is used to store the information about the edges. So, the second step takes O(|V|2) runtime alone. Thus the total runtime in this case is O( |V| + |V|2) or O( |V|2).

So, adjacency list representation takes less time.

For example, from the tables,

At |V|=8000 and |E|=64000

Adjacency Matrix representation takes 1546814.4 microseconds.

Adjacency list representation takes 10936 microseconds.

**Report prepared by:**

**Rasman Mubtasim Swargo**

**Student ID: 1705051**