REPORT

DATA STRUCTURES & ALGORITHMS II SESSIONALS

CSE 208

OFFLINE 1

SUBMITTED BY:

MD. ZARIF UL ALAM

1705010

Runtime Analysis

Adjacency List Representation :

Number of Vertices	Number of Edges	Average time (micro seconds)
1000	1000	25
	2000	48
	4000	99
	8000	199.4
	16000	299.2
	32000	598.4
	64000	296.5
2000	2000	92.7
	4000	99.7
	8000	199.5
	16000	402.2
	32000	598.3
	64000	1298.7
	128000	2493.3
	256000	4787.3
4000	4000	298.9
	8000	399.2
	16000	401.1
	32000	900.2
	64000	1496.6
	128000	2493.3
	256000	4986.7
	512000	9773.6
	1024000	19547.7
8000	8000	1550.9
	16000	1561.2
	32000	1562.0
	64000	1562.2
	128000	3241.3
	256000	4686.4
	512000	9372.8
	1024000	19384.5
	2048000	39793.8
	4096000	77891.8

16000	16000	897.8
	32000	1396.2
	64000	2096.2
	128000	3291.2
	256000	5884.2
	512000	11269.9
	1024000	21043.8
	2048000	39693.9
	4096000	80385.3
	8192000	2.40956e+005
	16384000	4.04746e+005

Adjacency Matrix Representation :

Number of Vertices	Number of Edges	Average time (micro seconds)
1000	1000	1562.1
	2000	3112.6
	4000	3123.5
	8000	4562.0
	16000	5124.2
	32000	6124.3
	64000	8686.0
2000	2000	6485.2
	4000	8181.2
	8000	10172.8
	16000	9275.3
	32000	9372.5
	64000	10671.9
	128000	13763.2
	256000	18353.2
4000	4000	15647.7
	8000	29677.6
	16000	29680.6
	32000	29719.7
	64000	31246.1
	128000	37492.4
	256000	39083.7
	512000	48426.1
	1024000	67209.6

8000	8000	76566.2
	16000	1.20315e+005
	32000	1.18757e+005
	64000	1.18754e+005
	128000	1.25082e+005
	256000	1.31255e+005
	512000	1.3125e+005
	1024000	1.57576e+005
	2048000	1.89084e+005
	4096000	2.68788e+005
16000	16000	3.7847e+005
	32000	4.71962e+005
	64000	4.73495e+005
	128000	4.73537e+005
	256000	4.76628e+005
	512000	4.84841e+005
	1024000	5.04753e+005
	2048000	5.75487e+005
	4096000	6.68813e+005
	8192000	8.3451e+005
	16384000	1.20748e+006

Question – Answer

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 of BFS for adjacency list representation is O(V+E) . If |V| is unchanged and |E| is doubled the overall time complexity remains same but the number of computation is increased by almost double than what was before which means the runtime will be significantly large. From a time perspective , the difference is in some microseconds .
   If E becomes the dominating term , the complexity becomes O(E) .

   For example , from the table , for V = 2000 and E = 8000 , average time taken is 199.5 microseconds . When |E| is doubled , average time taken is 402.2 , So the difference is approximately 100 microseconds .

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 of BFS for adjacency list representation is O(V+E) . If |E| is unchanged and |V| is doubled the overall time complexity remains same but the number of computation is increased by almost double than what was before . From a time perspective , the algo has to traverse through doubled the vertices leading to more time in computation which means greater runtime .
   If V becomes the dominating term , the complexity becomes O(V) .

For example , from the table , for E = 8000 and V = 4000 , average time taken is 399.2 microseconds . When |V| is doubled , average time taken is 1550.9 microseconds . The difference is approximately 1000 microseconds .

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

   Answer :

   The runtime of BFS for adjacency matrix representation is O(V^2) . If |V| is unchanged and |E| is doubled the overall time complexity remains same as the time complexity depends solely on the number of vertices and it is directly proportional to the square of the number of vertices . That is why doubling the number of edge doesn't have a significant impact on the complexity but it will surely increase the runtime .

   We can also see that quantitatively from our measurement . For example , from the table , When V = 4000 and E = 16000 , average time taken is 29680.6 . When |E| is doubled , average time taken is 29719.7 . The difference is as expected , insignificant.

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 of BFS for adjacency matrix representation is O(V^2) . If |V| is unchanged and |E| is doubled , the number of computation is almost 4 times greater than that of earlier. Though the Big-Oh complexity remains same as it is differed only by a constant of 4 , but 4 times more computation leads to a huge difference in runtime.

   For example , from the table , for E = 8000 and V = 4000 , average time taken is 29677.6 microseconds . When |V| is doubled , average time taken is 76566.2 microseconds .The latter case has almost 3 times the runtime compared with the previous case.

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 :

For the same |E| and |V| , the runtime for adjacency list and adjacency matrix representation is different . Though the algorithm is same , runtime depends on the data structure used . The runtime here depends on how long does it take to iterate across the outgoing edges of a given node.
For adjacency list representation the time complexity is O(V+E) . All the adjacent vertices are stored in a list format and the total number of nodes in that case is the summation of degrees of all the vertices . From handshaking lemma , this value is 2* |E| . So the overall complexity is O(V+E). But if the graph is a dense graph (number of edges is significantly large compared to number of vertices) , then the E term dominates and the complexity becomes O(E).
For adjacency matrix representation the time complexity is O(V^2) . Here the edges are stored in a matrix .When adjacency needs to be checked during BFS , the algorithm has to traverse through all the vertices to get the adjacent vertices . So the time complexity becomes O(V^2).
From the table for V = 8000 and E = 16000 , average time for adjacency list representation is 1561.2 microseconds . And for the same V and E , average time for adjacency matrix representation is 1.20315e+005 microseconds.

So we can say that adjaecncy matrix representaion has significantly higher runtime .