**CSE 207**

**DATA STRUCTURES AND ALGORITHM 2**

**SESSIONALS**

****

**OFFLINE NO : One**

**TITTLE : Report of BFS runtime for two type of implementation**

**SUBMITTED BY :**

**FAHMID - AL-RIFAT**

**STUDENT NO : 1705080**

LEVEL-2 ,TERM-2

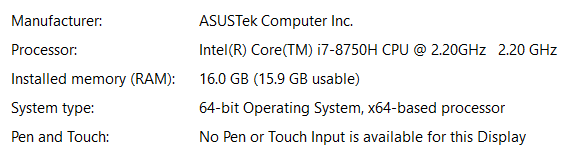
DEPARTMENT OF CSE

BUET

**OBJECTIVE:** The objective of offline is to get deep understanding about “ Graph Data Structure “ and it’s different type of implementation . One type of implementation is Adjacent matrix and other type is Adjacent list . In both type of implementation they have some advantages and disadvantages . Depanding on situation and according to how much dense the graph is the efficiency of type is determined .

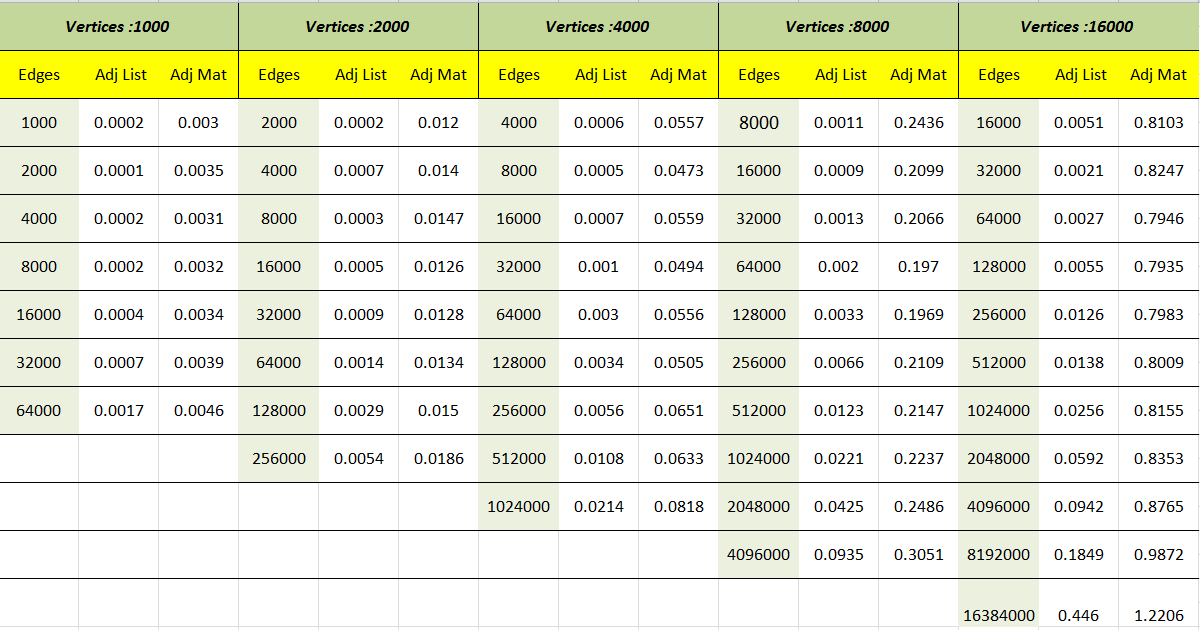
**MACHINE CONFIGURATION:**

Code::Blocks 17.12

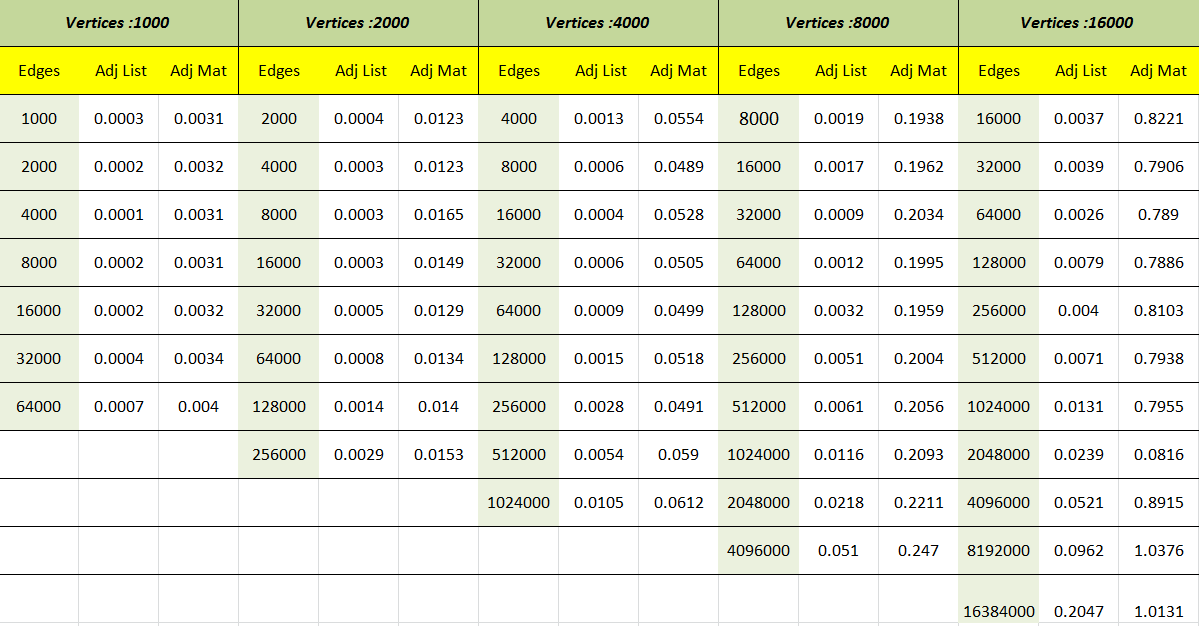


**DATA SET :**

**Table of BFS average running time in millisecond with respect to vertices and edges Undirected Graph :**

****

**Table of BFS average running time in millisecond with respect to vertices and edges Undirected Graph :**

****

**Question and Ans :**

**● What is the impact on runtime if we keep |V| unchanged and double |E| for**

**adjacency list? Why is it so?**

**Ans: if we double |E| keeping |V| constantant in adjacency list the runtime becomes almost or more than double . It’s a linear relationship as doubling the edges increases search in adjacent list as we have to search through double entity now . In term of complexity it increases to O( |V| +2 \*|E|) from O (|V|+|E|).**

**● What is the impact on runtime if we keep |E| unchanged and double |V| for**

**adjacency list? Why is it so?**

**Ans: if we double |V| keeping |E| constant in adjacent list the runtime becomes almost or more than double . It’s a linear relationship .Doubling the edges don’t increases search in adjacent list but in BFS algorithm we have traverse though double vertices and needed double time hence. In term of complexity it increases to O( 2\*|V| +|E|) from O (|V|+|E|)**

**● What is the impact on runtime if we keep |V| unchanged and double |E| for**

**adjacency matrix? Why is it so?**

**Ans: if we double |E| keeping |V| constantant in adjacentcy list the runtime becomes slightly greater and sometimes constant . Actually doubling the edges doesn't bring significant change in the case or adjacent matrix as we have to search though the same number as before . So the run time doesn’t rise so much as adjacency list representation . In term of complexity remain constant at O(|V|^2)**

**● What is the impact on runtime if we keep |E| unchanged and double |V| for**

**adjacency matrix? Why is it so?**

**Ans:if we double |V| keeping |E| constantant in adjacent list the runtime becomes about or more than 4 times of the one before . The reason behind this is in the adjacent matrix list we have search through the whole matrix . Doubling the the number of vertices generate new 2D matrix of which is 4 times of before and as well the running time of the algorithm becomes 4 times of the one before too.In term of complexity it increases to 4\*O( |V|^2|) from O (|V|^2)**

**● For the same |E| and |V|, why are the runtimes for adjacency list and adjacency**

**matrix representation different? Which one is higher and why?**

**Ans:In the case of a lower number of edges or sperce graph the running time in adjacent list representation is much lower than adjacent matrix representation . because in the lower dense graph in adjacency list implementation we have to search lower as we don't have to check all combinations of vertices . One the other hand when the dense graph case appear the running time of adjacent matrix slightly improve as both of the structure have about same number of operation than.**

**DISCUSSION :**

From the above statistics we can see that the density determine which Graph Data Structure is better .In the undirected graph usually we needed much time as we have to visit both direction of edges max cases .The changes in the number of edges and vertices make change in the running time of the BFS algorithm. When the graph is sparse then Adjacent List implementation out perform the Adjacent Matrix implementation .In real life case maximum graph are sparsed and so adjacent list is a good choice. On the other hand for dense graph adjacent matrix is good choice .

I