Comparison of Methods of Graph representation and Topological Sorting Procedure.

Mateusz Tabaszewski

Introduction and Methods of Research

This paper is meant to illustrate the difference between 4 methods of graph representation in a computer. These methods will be tested based on the time necessary to find if a connection exists between two randomly chosen vertices. Programs responsible for creating graphs and performing the experiments have been written using the C++ language and time was measured using GetProcessTimes(), GetCurrentProcess() functions included via <processthreadapi.h> header. The graphs tested in the first experiment were created randomly while making sure that no two edges were the same. The saturation of the graph with edges is equal to 0.6. Results of the experiment are shown through graphs, with two graphs representing values divided by the number of searched pairs of points (to get the average time per pair of points) and by n (number of vertices of the graph). The two remaining graphs show time divided only by the number of search operations (pairs of points).

The following part of the experiment required creating a DAG (Directed Acyclic Graph). This was accomplished by connecting all nodes in a way that would not create a cycle-each node would have out-edges connected to the following nodes (but never the reverse). Then random nodes would be deleted to fill the graph with an edge fill factor equal to 0.3. The reason behind DAG's creation was to test the time necessary to perform the topological sorting procedure. The results of the second part of the experiment have been shown on a single graph. The values on the graph are the average of 1000 000 repetitions of a topological sort, such a high number of repetitions was meant to minimize the noise and make the time more easily measurable.

Experiment 1

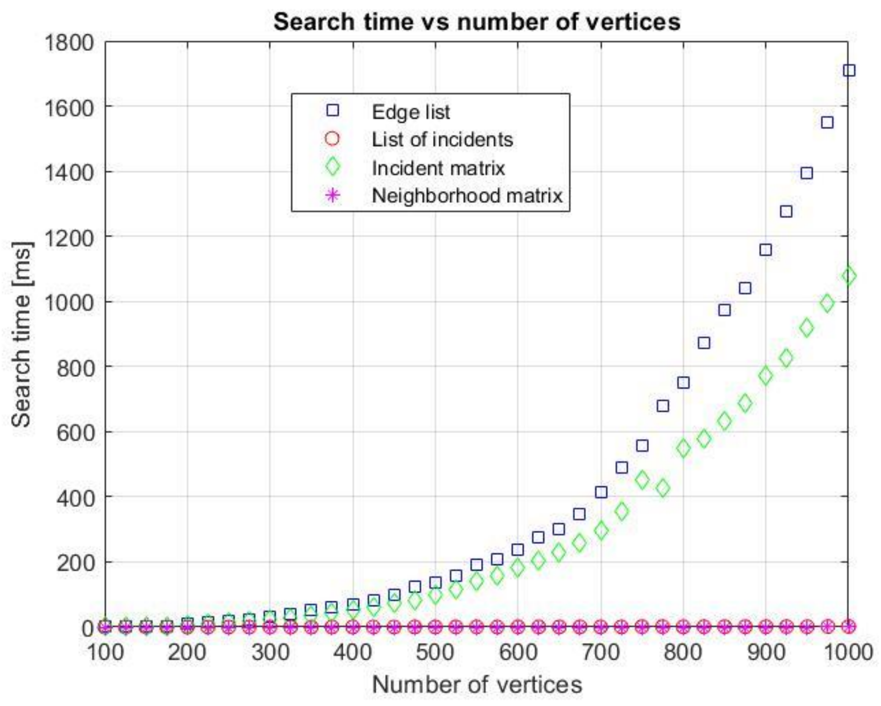

Fig. 1. Graph showing time necessary for finding a connection between a par of vertices in n-sized graph(not divided by n).

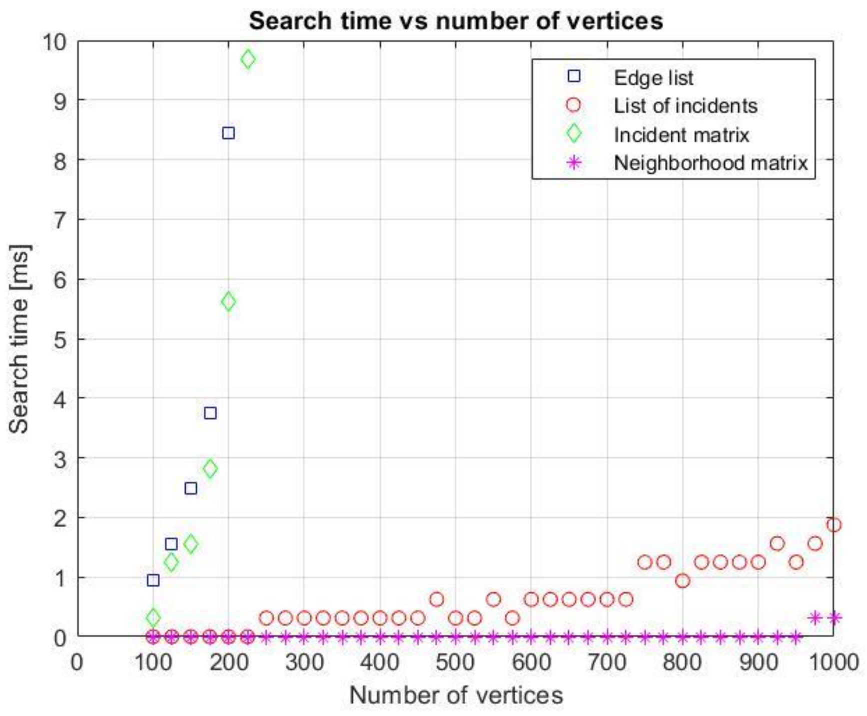

Fig. 2. Graph from Figure 1 with Y –axis cut-off point of 10 ms.

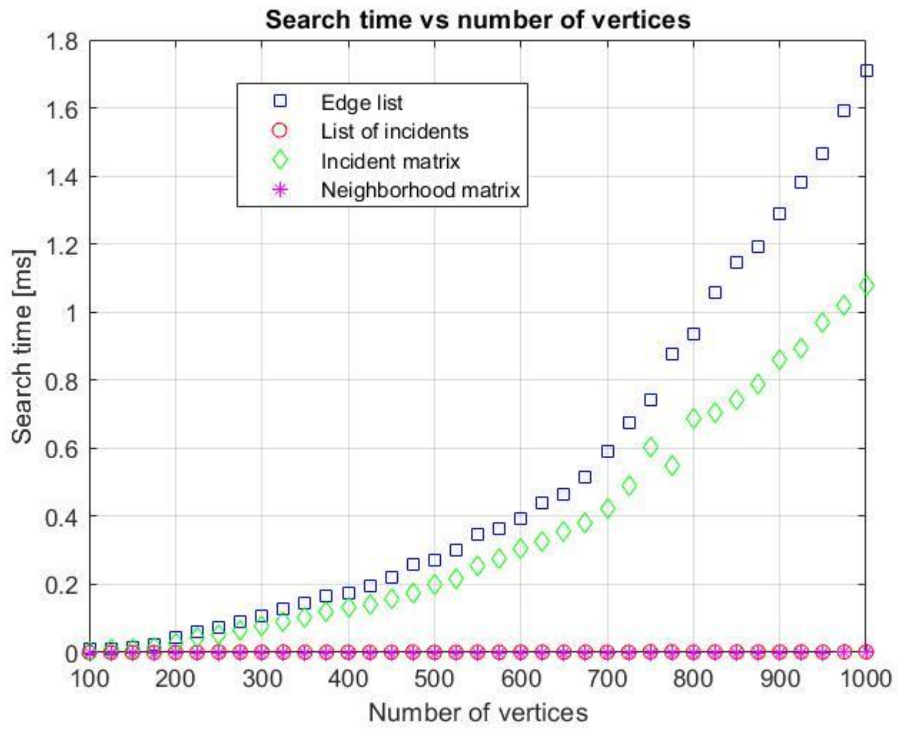

Fig. 3. Graph showing the time necessary for finding a connection between a pair of elements in n-sized graph divided by n.

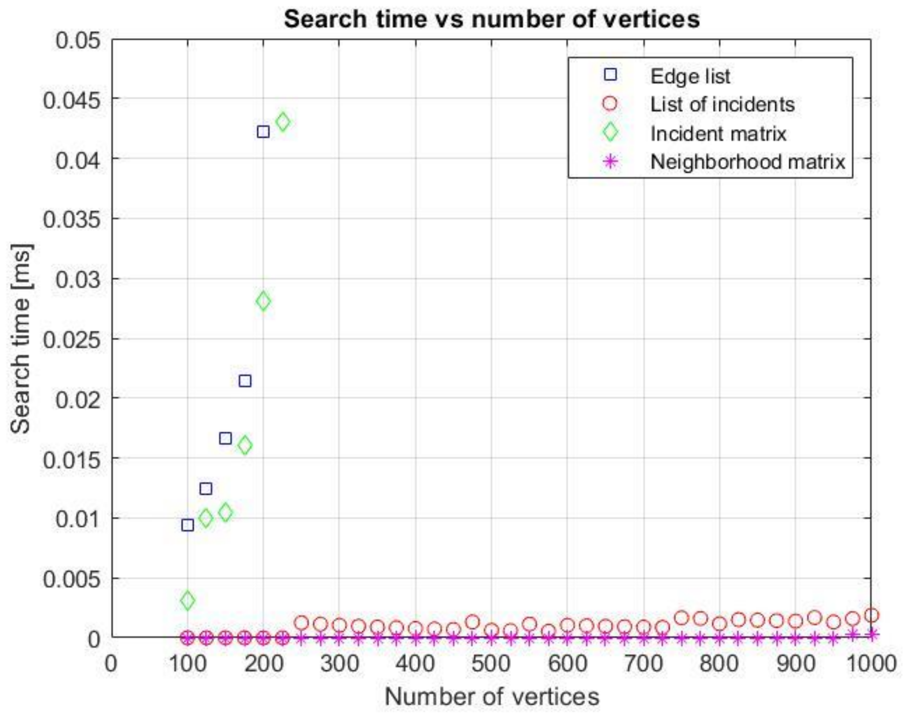

Fig. 4. Graph from figure 3 with cut-off point for Y-axis equal to 0.05 ms.

Experiment 2

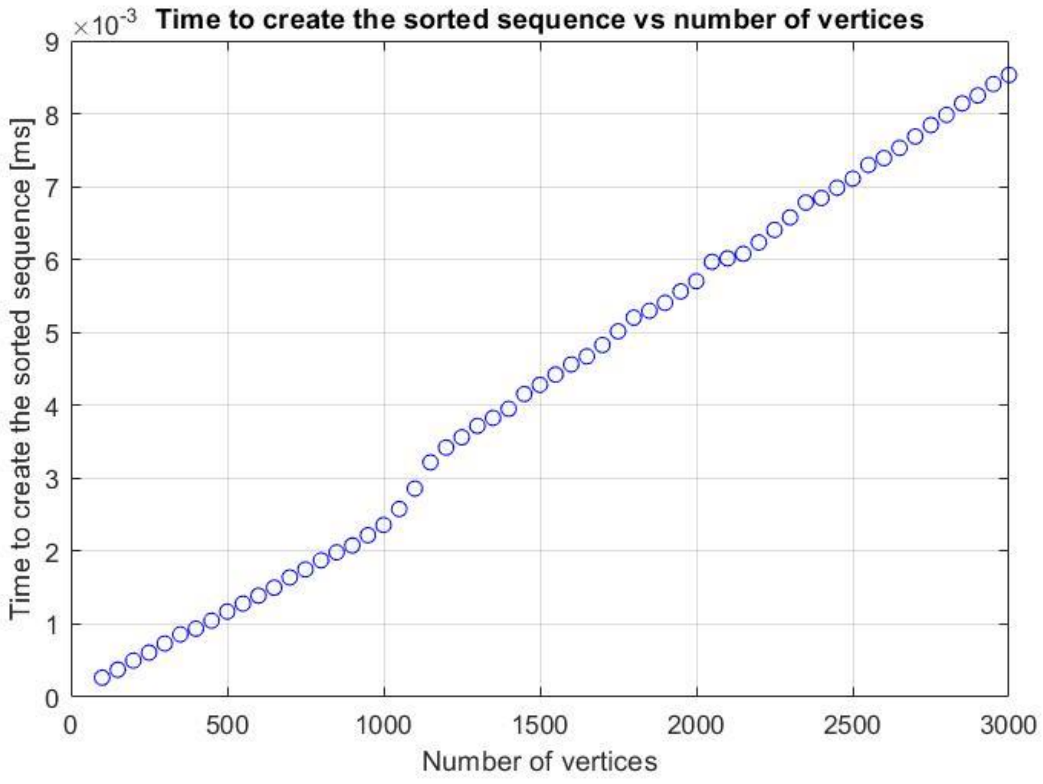

Fig. 5. Graph showing the time necessary to finish the topological sort algorithms for DAG with number of nodes shown on X-axis.

Conclusions

Judging by the results of the first experiment both Edge List and List of Incidents turned out to be ineffective methods of representation for this task, while List of Incidents and Neighborhood Matrix(Vertex Matrix) turned out to be highly effective. Edge List had the longest time out of all tested methods of representation for finding a connection. This is most likely caused by the fact that the time necessary to find a connection is connected with a number of edges in the graph- O(|E|). This is also the case with the second slowest, for this experiment, method of representation- List of Incidents. As the time is also dependent on a number of edges. However, in this case, Incident Matrix works faster than Edge List, most likely due to the nature of implementation of Edge List and the fact that at least 2 compare operations are performed, while in Incident Matrix the second compare operation is only performed when an edge connected to one of the tested vertices is detected. List of Incidents is the second fastest of presented methods for this problem as it only depends on a number of vertices, a more specific number of remaining vertices connected to one of looked for vertices. Although complexity is still regarded as O(|V|) in reality it is mostly going to be much lower. The fastest representation is the Neighborhood Matrix(Vertex Matrix) which has the complexity of O(1) as it only takes the parameters of chosen vertices and checks if at the chosen point of the matrix there is a 1 or a 0 which means that the method is near-instant and does not depend on the size of the graph.

For the second experiment, a representation in the form of a List of Incidents was chosen thanks to it being a fitting choice for a recursive algorithm and giving quick access to all nodes further requiring investigation from the starting node. Neighborhood Matrix representation was rejected due to being more problematic when it came to implementation. Incident Matrix was rejected due to problematic representation of neighbors. This would cause operations to take longer since more data would need to be compared before accurately judging which nodes are neighbors-for each vertex. Edge List was also rejected due to lacking a direct representation of which nodes were connected-only a pair would be represented at once, which would greatly increase the searching time for neighboring nodes. The time necessary to perform a topological search also increases with number of nodes as it depends on the number of vertices as well as the number of connections between given vertices. The algorithm is recursive and seems to fit the problem quite well and the time necessary to finish calculations is (subjectively) quite fast.