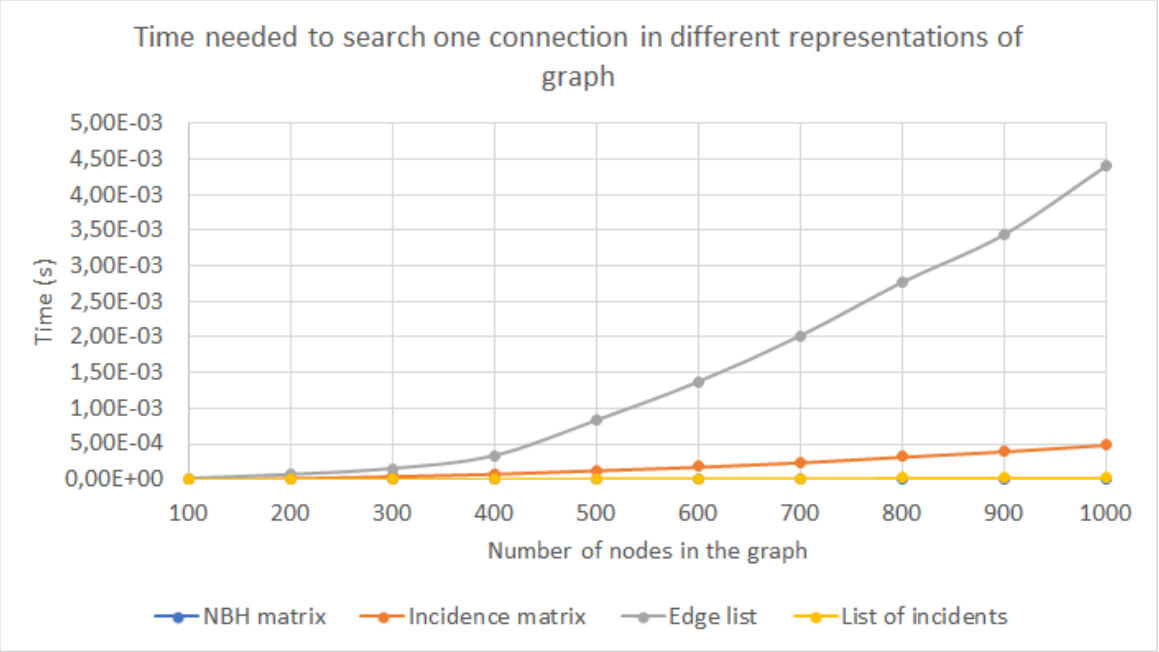
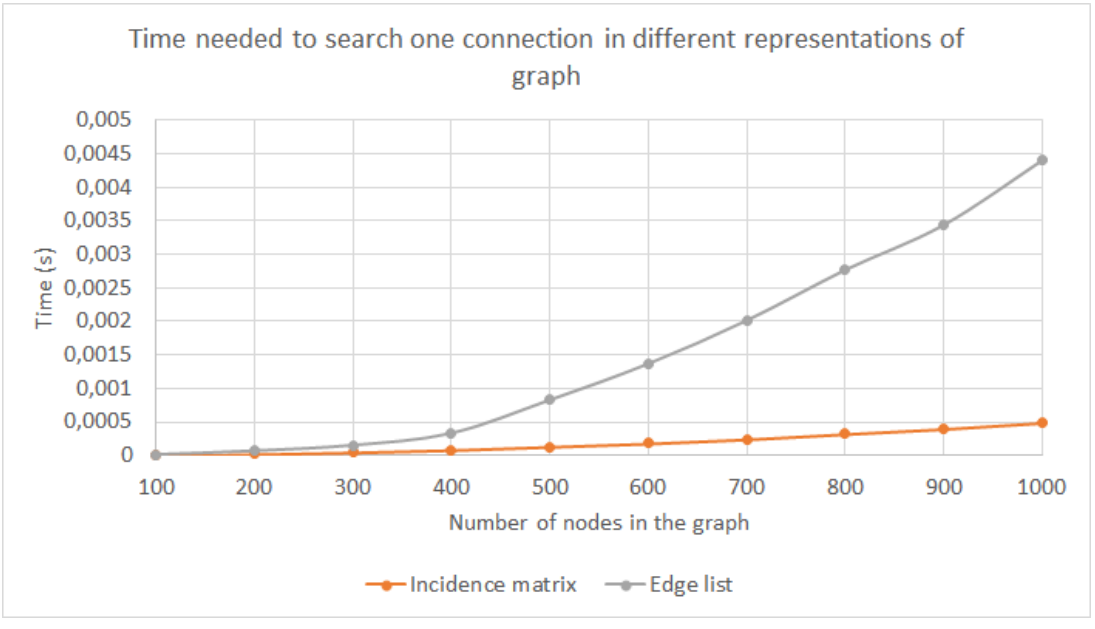
Mateusz Małecki: 148265
Nikodem Seidel: 148269

1. Searching for information about connection of two nodes:

We started with generating 10 different undirected graphs of sizes 100, 200, …, 1000 represented by the edge list with saturation 0.6. Therefore the number of connections in each graph is equal to $\frac{n*(n-1)}{2}$ * $0,6$ where n is the number of nodes.
We picked edge list, because it was the easiest to create, understand and convert into different representations.
Bellow is the chart, that shows comparison of all four representations



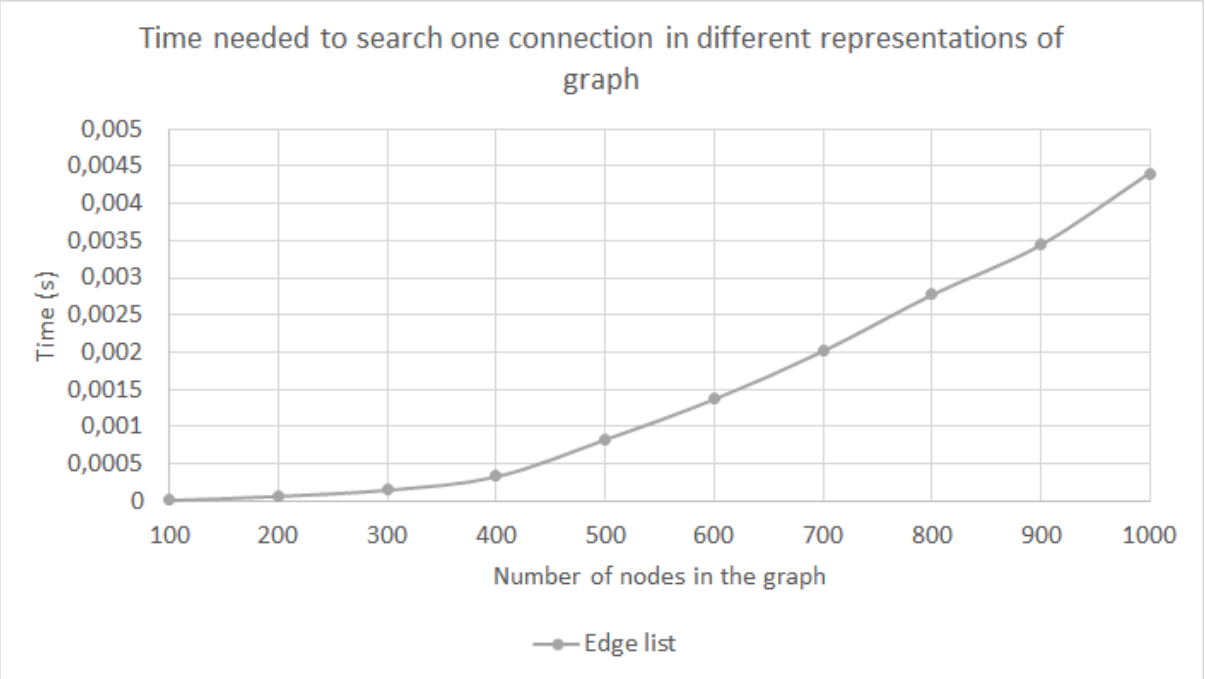

What can be easily noticed, is the fact, that In the Edge list is the slowest to search through. But should it be that slow? On the next chart, we can see a comparison of EL and the Incidence matrix. In theory their complexity is the same and equal $O(n^2)$, since to search for the precise edge between two selected nodes in the worst case scenario we have to look through all possible edges. Thus since number of edges in graph with saturation = s is equal to $\frac{n*(n-1)}{2}$ * $s$, so the big O notation is equal to $O(n^2)$.



Mateusz Małecki: 148265
Nikodem Seidel: 148269

But what can be seen is the fact that IM is much faster. This is caused by our way of implementation. It is much faster in C language to iterate over the 2D matrix than through the linked list. The process is just less complex, therefore IM is that faster.

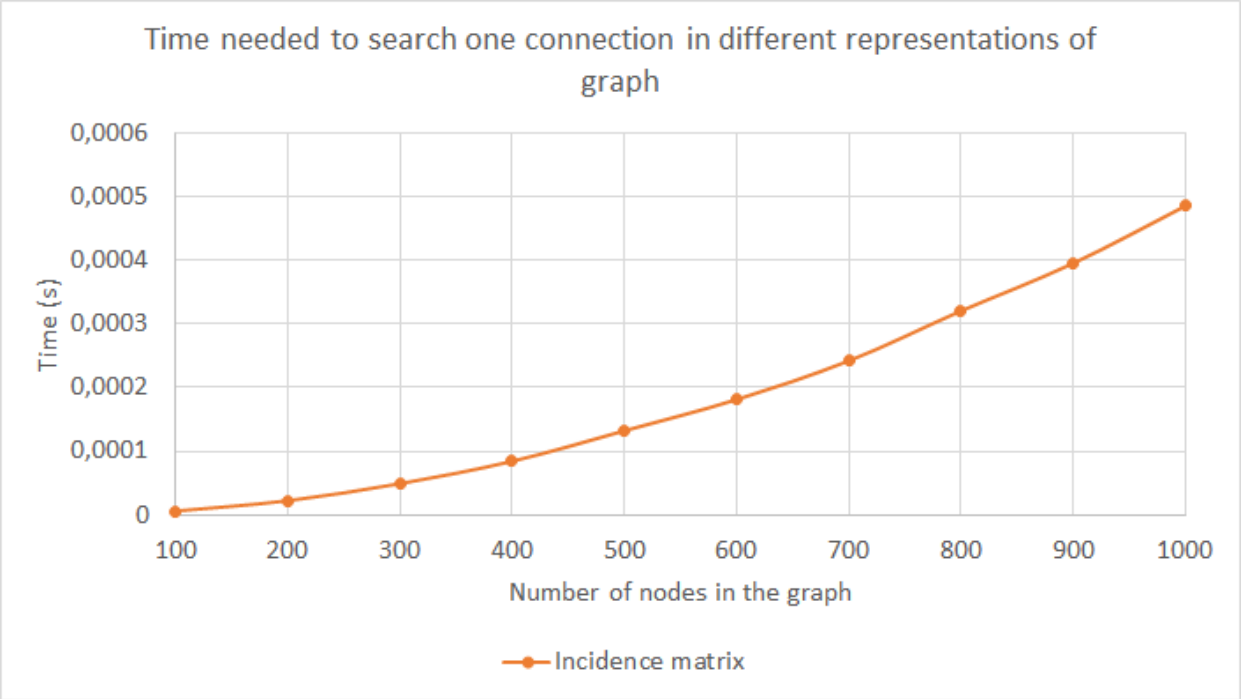
- Edge list




Procedure of searching for a specific edge is quite simple, but not optimal. Algorithm is iterating over the list of all edges and is checking if any pair of nodes is the same as searched until the edge is found. If such an edge does not exist the algorithm is iterating to the end of the list. Thus in the graph with n nodes time complexity of such procedure is equal to $O(n*(n-1)/2) = O(n^2)$.
This representation is keeping information about every edge separately, thus memory complexity is equal to $O(n^2)$. This representation is not very useful.

Mateusz Małecki: 148265
Nikodem Seidel: 148269

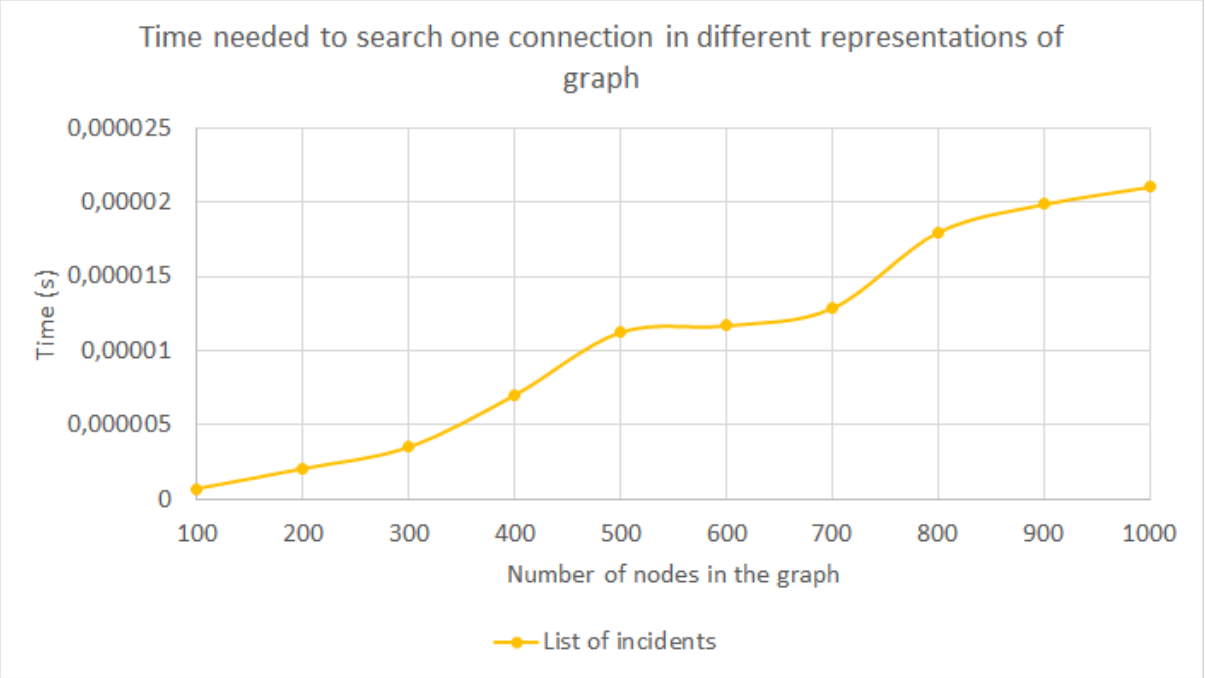
- Incidence matrix




This representation of the graph keeps information about which edges are connected with which nodes. For every node, there is an information if the particular edge is connected to the node or not. Therefore memory complexity of this representation is equal to number of nodes times number of edges. $O(n*n^2) = O(n^3)$.
Operation of checking if particular nodes are connected include iterating over all edges for one of the nodes, and when we get information that the specific edge is connected to this node, we check if the same edge is also connected to the second node. In this way, in the worst case scenario, we get complexity equal to $O(n^2)$. In our test, the only practical advantage over EL is the fact that iterating over the matrix is faster than over the linked list (less complex).

Mateusz Małecki: 148265
Nikodem Seidel: 148269

- List of incidents

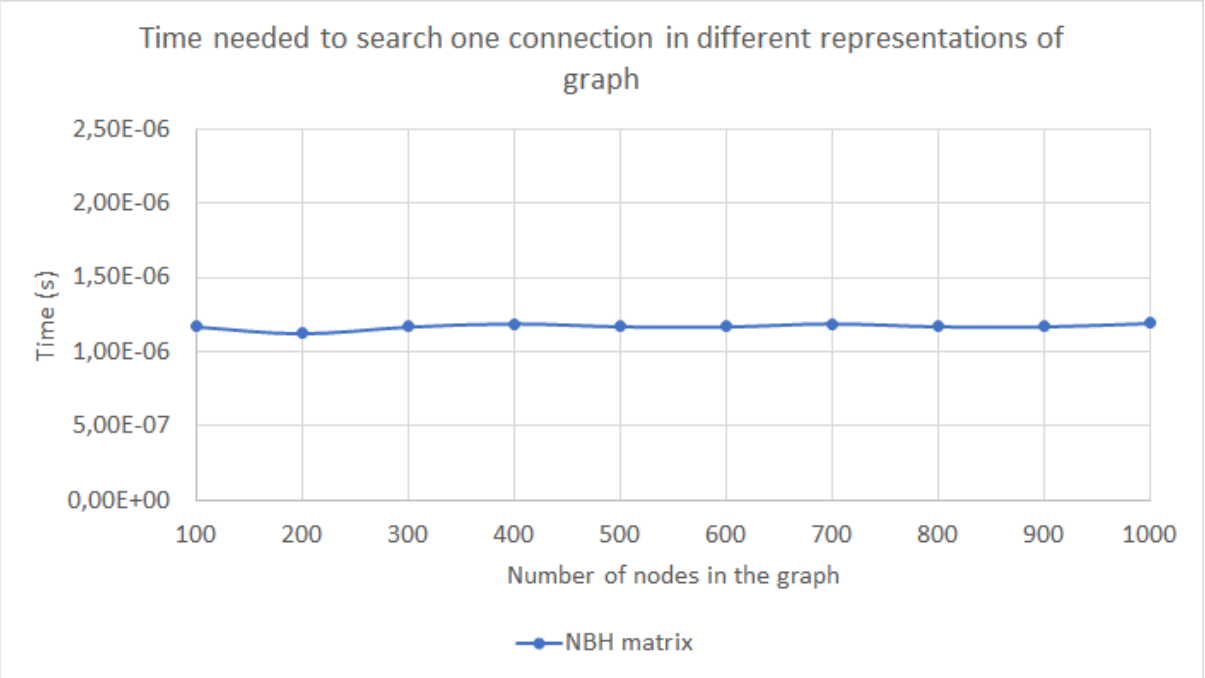



Incidence list is an array of lists. This representation holds information about which vertices are connected to the particular vertex. One to know if there exists connection between nodes A and B has to search B node in lists which is stored in an array which has index A (array[A]) or search A in list with address array[B]. As well as NBH matrix list of incidents has simple implementation. On the graph above there is small hesitation on 600 and 700, but we can clearly conclude that the complexity of searching for the existence edge between two particular vertices in this representation is linear O(n).

Mateusz Małecki: 148265
Nikodem Seidel: 148269

- Neighbourhood matrix

This representation is one of the easiest to understand and implement. It's also the fastest one in our test as shown on the chart below.




This representation for every node keeps information to which node it is connected thus the operation of checking the connection between two nodes is very simple. It's just checking the value of the 2D array for two determined indices. What can be easily concluded is the fact, that time complexity of such operation is constant i.e. O(1).
To keep this representation in memory, we need space for bool variables for every possible connection, so memory complexity of this implementation is equal to $O(n^2)$.
While considering undirected graphs, this representation might be a little memory wasting, since the matrix will always be symmetric, and connection from node "x" to node "y" will be the same as from node "y" to node "x".

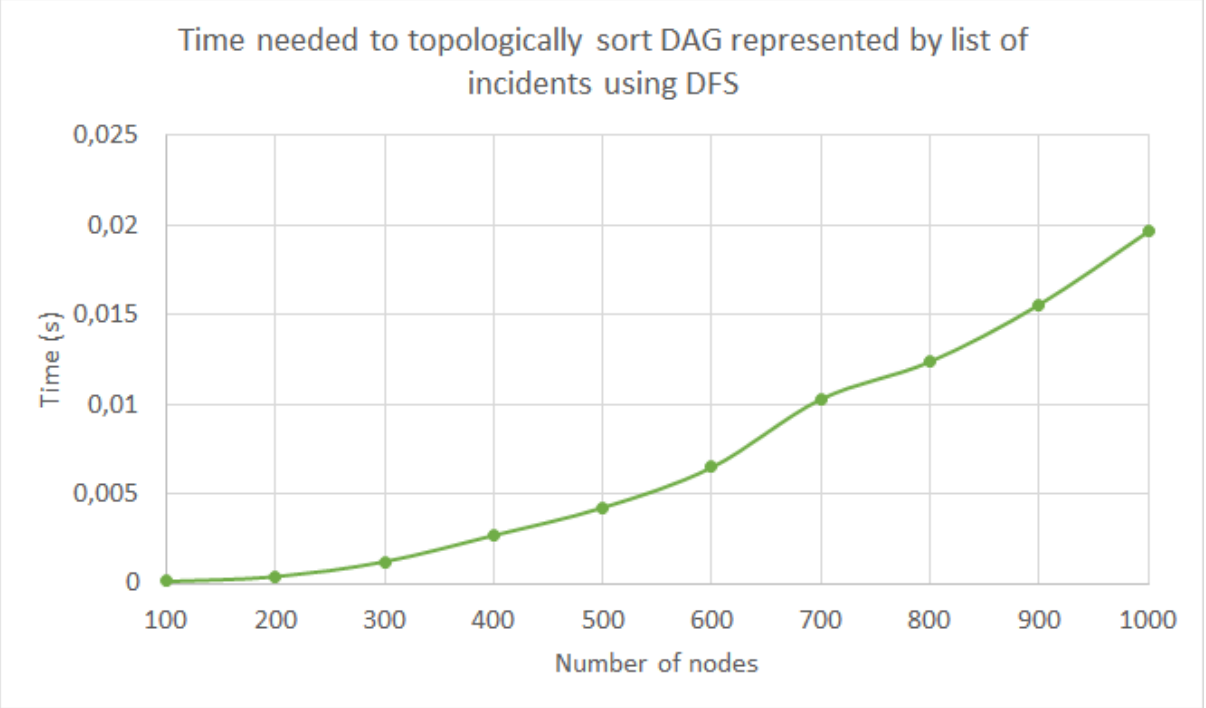
To sum up, for our test NBH matrix occurs to be uncomparably faster than the other representations. One could say that this implementation is made for such operations. List of incidents have very big potential. In our implementation this structure was created as the matrix of linked lists, but to maximize the performance for considered test, a good idea would be to make a list of binary search trees. That would lower the time complexity to O(logn), where n is a number of all nodes. Incident matrix (IM) as well as Edge list performs purely, and in case of the IM the implementation, operations and understandment are not that easy.

Mateusz Małecki: 148265
Nikodem Seidel: 148269

| type of representation | time complexity for our test | memory complexity |
|---|---|---|
| neighborhood matrix | O(1) | $O(V^2)$ |
| incident matrix | $O(E) = O(V^2)$ | $O(V*E) = O(V^3)$ |
| edge list | $O(E) = O(V^2)$ | $O(E) = O(V^2)$ |
| list of incidents | O(V) | $O(V+E) = O(V^2)$ |

*V = number of vertices
**E = number of edges = V * (V-1) * level of saturation /2

- Topological sorting




Different graph representations can be used to topologically sort Directed Acyclic Graph. We've chosen a list of incidents because of how "deep first search" algorithm works. DFS uses information about which vertices are connected to the particular vertex so the list of incidents is perfectly suited for this purpose, because among all representations ADJ list gives needed data in the shortest time. Only Neighborhood matrix has comparable time needed to perform this operation, but consecutive nodes in the list are read one by one in contrast to the matrix, where we have to skip some empty cells to find the next node. Advantage of ADJ list over NBH matrix is the greater the smaller is the level of saturation, so in this case, where saturation is relatively small the list has better performance. Also neighborhood matrix in graphs with low saturation wastes much memory on filling an array with zeros. Another big advantage of list of incidents is easy implementation in DFS algorithm. We've rejected Edge list and incident matrix because of the longer time necessary to obtain essential data. Additionally Incident matrix has worse memory complexity than list

of incidents. The last disadvantage of these representations is that they are much more complicated in implementation.

The List of incidents is not the best choice in every case. Neighborhood matrix has better performance when saturation is big.

The same approach would be used in breadth first search because of the same reasons.