Zuzanna Gawrysiak ID: 148255
Agata Żywot ID: 148258

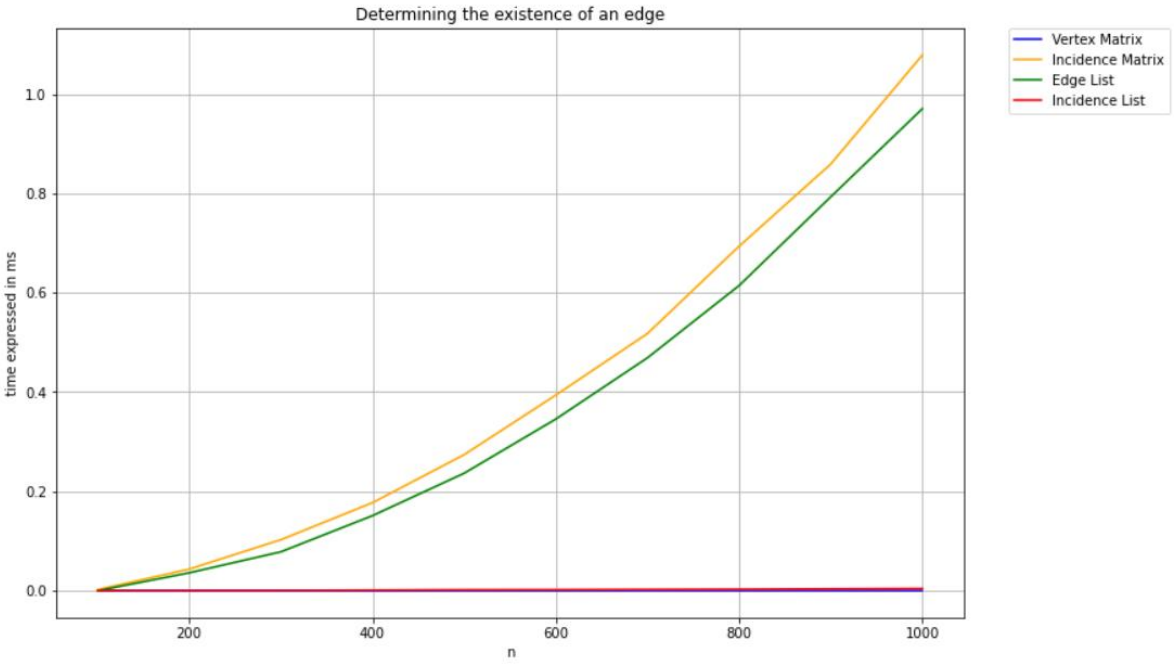
TASK 1

Method:

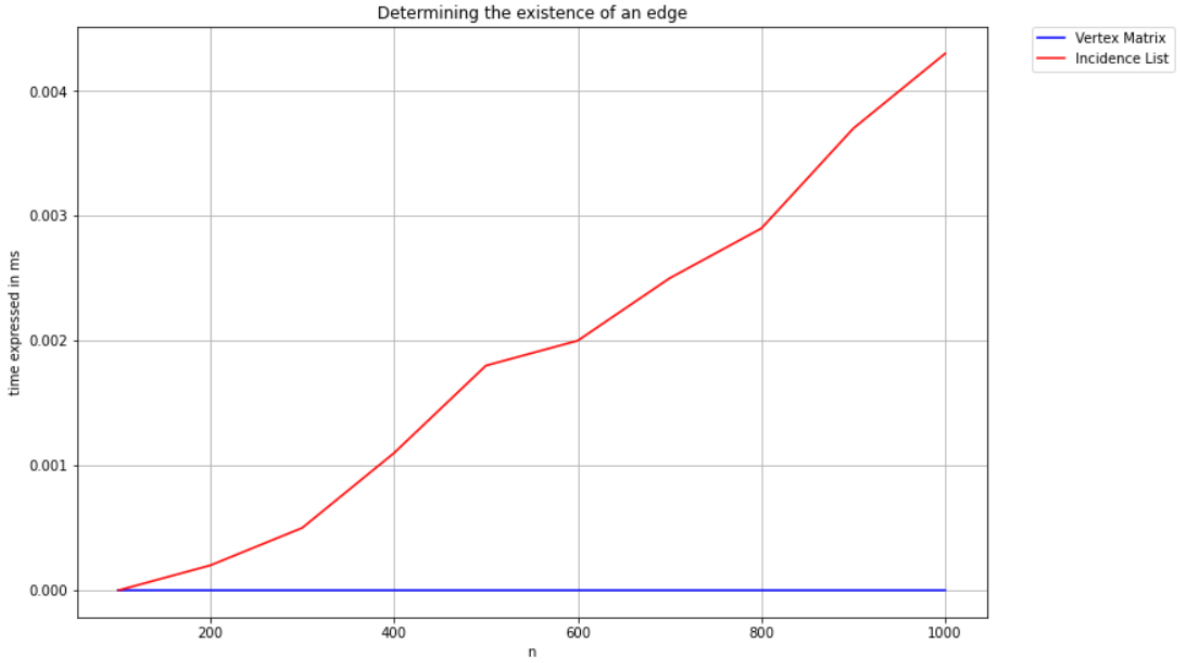
In the experiment a random undirected graph was generated by firstly creating a set of all possible edges and selecting from it only a desired number of edges (a product of the number of all edges possible multiplied by saturation value, in this task equal to 0.6). Subsequent time measurements (expressed in milliseconds) were taken on graphs of the number of vertices equal to n, starting from n = 100 up to n = 1000. Every outcome is averaged over 10 independent runs (i.e. 10 sets of random generated undirected graphs of different sizes).

Results:

n	Vertex Matrix	Incidence Matrix	Edge List	Incidence List
100	0,0000	0,0020	0,0000	0,0000
200	0,0000	0,0435	0,0360	0,0002
300	0,0000	0,1027	0,0783	0,0005
400	0,0000	0,1773	0,1512	0,0011
500	0,0000	0,2740	0,2366	0,0018
600	0,0000	0,3940	0,3457	0,0020
700	0,0000	0,5180	0,4691	0,0025
800	0,0000	0,6938	0,6146	0,0029
900	0,0000	0,8593	0,7933	0,0037
1000	0,0000	1,0790	0,9708	0,0043



Zuzanna Gawrysiak ID: 148255
Agata Żywot ID: 148258



The charts and table provided above present the average time needed to determine the existence of an edge between a pair of random vertices for four different graph representations, namely: **Edge List (EL), Incidence Matrix (IM), Incidence List (IL) and Vertex Matrix (VM)**.

Analysis of representations:

Considering an **Edge List**, this is the most primitive way to represent a graph – it is simply a list of pairs of connected vertices without any specified order. Therefore, to state whether a random edge exists it is necessary to check the list one by one and compare elements to the searched one. Consequently, the complexity of this operation is of O(E), where E is the number of edges in a graph. The memory complexity is also O(E) since E is the number of EL's elements.

The concept of **Incidence Matrix** is to create a VxE matrix (V – the number of vertices). The value 1 is placed in the intersection of row v and column e if an edge e includes a vertex v. For this reason, the complexity of determining the existence of an edge is, as in the previous case, of O(E) – one has to check the columns (edges) of two fixed rows (vertices) one by one. However, the space complexity is O(V*E) as the size of IM which undoubtedly makes this representation very inefficient (in each column, the value of all but two elements is equal to 0).
e.g. 1000(1000-1)*0,5*0,6 = 299700 elements in the case of a graph containing 1000 vertices.*

The other way to represent a graph is **Incidence List** in which an element (vertex) points to a list of all its adjacent vertices. The complexity of searching of an edge is O(V) since every vertex can have at most V-1 neighbours (although on average the number of adjacent vertices of a vertex is equal to E/V) and in the worst case all neighbours will have to be checked. The memory complexity of IL is of O(E).

Taking **Vertex Matrix** into consideration, it is very efficient in terms of stating whether an edge exists or not. This is because the only operation that has to be performed is to check the value of M[x][y], where M is a VxV matrix and x, y are vertices that a given edge is supposed to link (1 if an the edge exists, 0 – otherwise). Therefore, the complexity of checking the existence of a random edge is O(1) regardless of the value n. However, a disadvantage of this particular representation is its space complexity of O(V^2) as the matrix has the dimensions VxV. In the case of a sparse graph, VM is filled mostly with 0's and uses a lot of space to represent only 'a few' edges.

Zuzanna Gawrysiak ID: 148255
Agata Żywot ID: 148258

Observations:

One can immediately notice the performance of IM and EL is significantly worse compared to IL and VM. This is the result of IM's and EL's complexity of searching an edge equal to O(E) which is further explained above. To have a clearer view on the IL and VM, the remaining representations were removed from the second chart. The results show the independence of VM from the number of vertices (n) and the complexity of searching an edge in IL increasing with the value V.

Conclusions:

Having analyzed the results, the representations of a graph that seem to be the most efficient are Incidence List and Vertex Matrix. Edge List is definitely time consuming in terms of searching as well as Incidence Matrix which space complexity is the worst of all aforementioned representations. Incidence List appears to be the best choice since it does not require as much memory as a vertex matrix and performs the operation quite effectively. However, in case of a dense graph it is worth considering Vertex Matrix representation since a matrix will not 'waste' much memory to store 0's. Therefore, the best choice of graph representation surely depends on a problem specification.

TASK 2

Method:

Random DAGs were generated by firstly calculating the desired number of edges with given saturation = 0.3, and then iteratively adding random edges. After each addition, a new graph was checked whether it was cyclic – if yes, the new edge was deleted. The process was repeated until the expected number of edges was reached. Surely it is not the optimal solution, as recursive checking for cycles consumes vast amounts of memory and time, but it ensures correctness of a graph and obtains the exact number of edges.

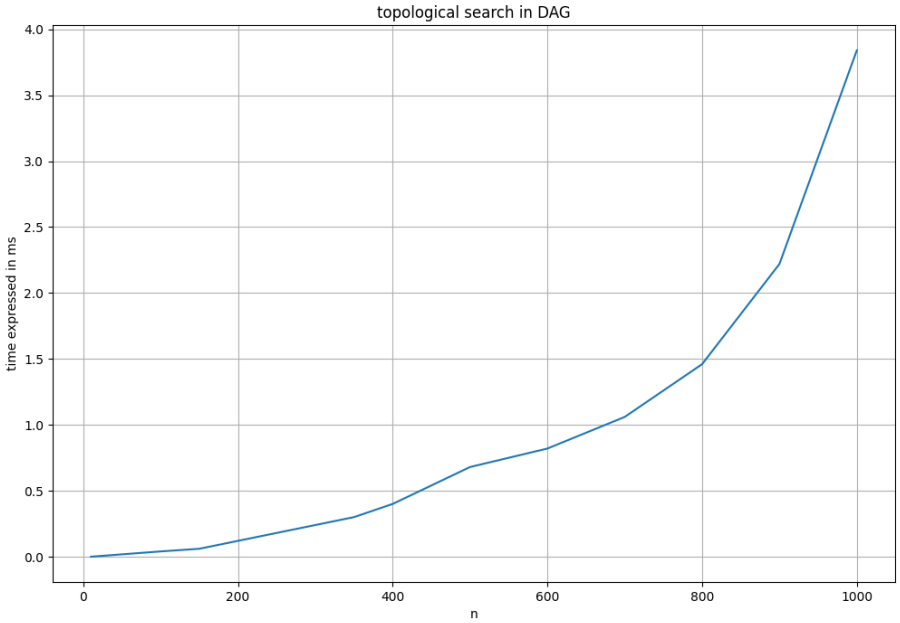
Topological Search was performed using Kahn's algorithm which is based on counting in-degrees of vertices and the chosen graph representation is the incidence list.

Outcomes are averaged over 50 repetitions of the search operation and the time is expressed in milliseconds.

Zuzanna Gawrysiak ID: 148255
Agata Żywot ID: 148258

Results:

n	time [ms]
10	0,00
100	0,04
150	0,06
200	0,12
250	0,18
300	0,24
350	0,30
400	0,40
500	0,68
600	0,82
700	1,06
800	1,46
900	2,22
1000	3,84



Observations and conclusions:

Time complexity of the implemented algorithm is O(V+E), since the whole adjacency list needs to be traversed to fill in-degrees of vertices, i.e. iterate over each of V vertices and E edges.

The chosen graph representation is incidence list (IL), since it has one main advantage – simplicity of implementation. Topological search requires visiting adjacent vertices and IL gives immediate [O(1)] access to the neighbours of the checked vertex. It is much better than other considered representations - searching for neighbours in AM, EL, and IM has the time complexity respectively O(V), O(E) and O(E). Another advantage of IL is its memory complexity – O(E), which is the minimum possible value for reviewed graph representations. There are no significant drawbacks of IL concerning topological search.

Sources:

http://www.cs.put.poznan.pl/arybarczyk/GrafReprezentacje.htm

https://www.geeksforgeeks.org/graph-and-its-representations/

https://www.geeksforgeeks.org/topological-sorting/