Zuzanna Gawrysiak ID: 148255
Agata Żywot ID: 148258

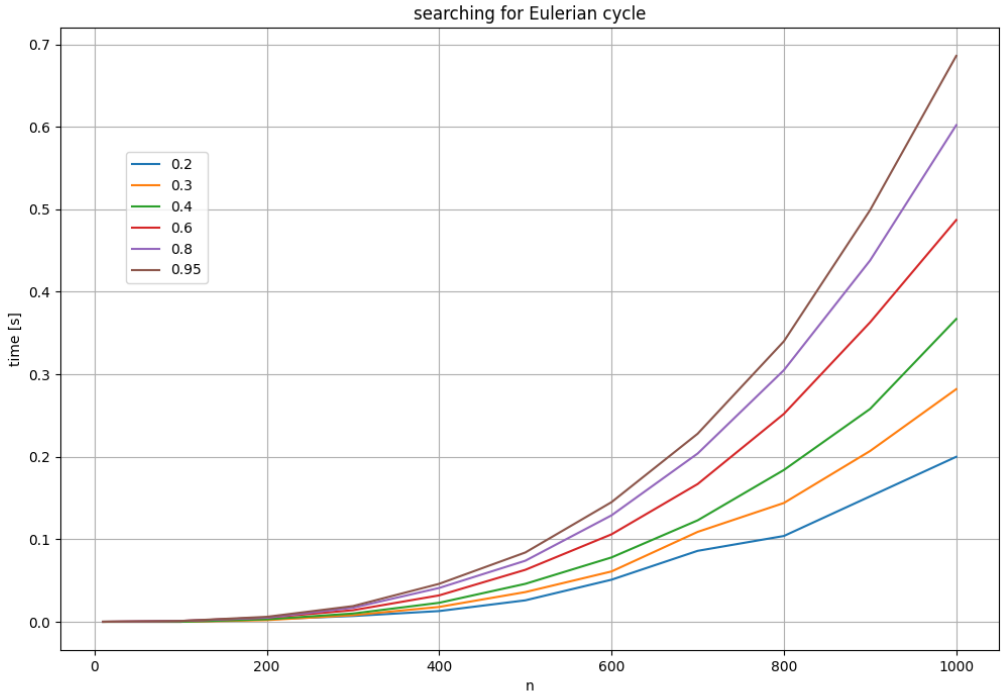
Report #4 - Backtracking Algorithms

1. Euler Cycle

Method

In the experiment random connected graphs were generated with given saturations [0.2, 0.3, 0.4, 0.6, 0.8, 0.95] and modified to obtain eulerian graphs. Searching for Eulerian Cycle (EC) was implemented with algorithm using recursive DFS procedure and chosen graph representation is vertex matrix. Subsequent time measurements (expressed in seconds) were taken on graphs of the number of vertices equal to n, starting from n = 10 up to n = 1000.

Results



Chosen representation

The graph representation chosen for EC searching problem is vertex matrix. It allows to access edges and adjacent vertices easily, which is required for search algorithm. Another advantage is the simplicity of implementation. On the other side, VM has one noticeable disadvantage, which is memory complexity of $O(|V^2|)$. It is worse than incidence list with $O(|E|)$, but for graphs with high saturation the difference is not tremendous. IL could be a good alternative, thanks to the easy access to incident vertices and uncomplicated implementation. Other representations are fairly impractical. Both incidence matrix and edge list have terrible ($O(|E|)$) time complexity of checking the existence of one edge and traversing adjacent edges. IM has also memory complexity $O(|V|*|E|)$ what makes it the worst candidate for the problem considered.

Zuzanna Gawrysiak ID: 148255
Agata Żywot ID: 148258

Observations and conclusions

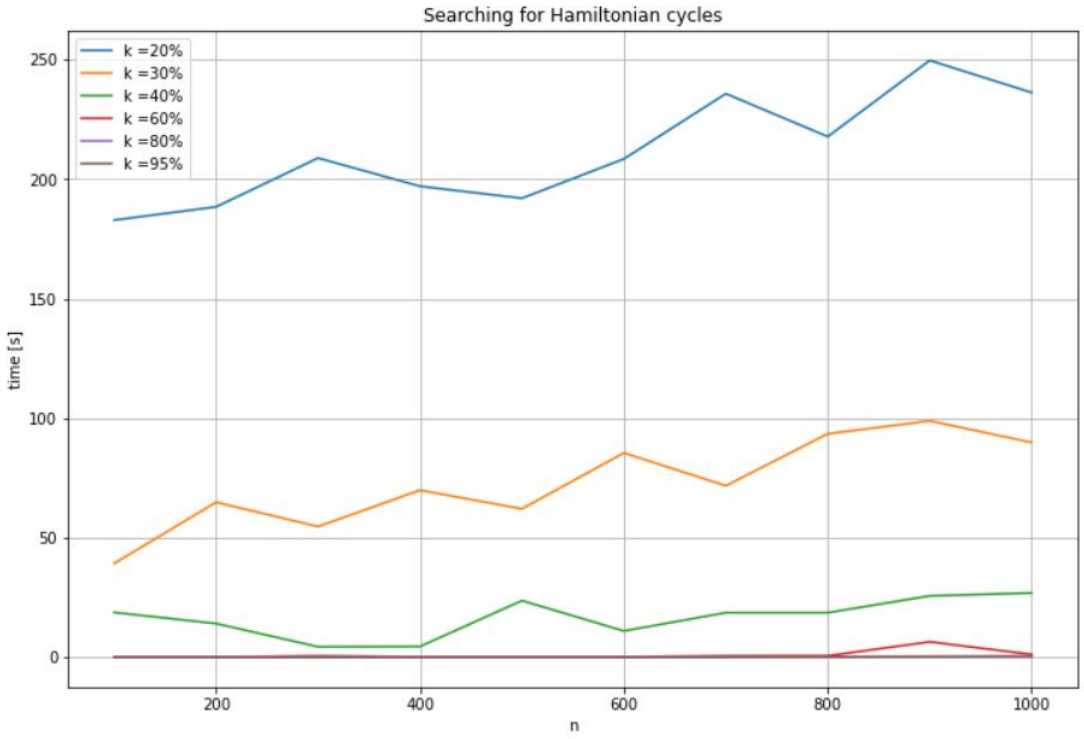
Time complexity for an undirected eulerian graph is O(|E|), since every edge has to be traversed. It is obviously the best possible result, because Euler Cycle has to include every edge. For big, dense graphs operation of searching is computationally heavy – recurrence procedure may cause stack overflow. It is also worth noticing, that chosen algorithm requires deleting edges of a graph, therefore one must create a copy of the instance if further operations on the graph are desired. Moreover, using vertex matrix representation with this algorithm allows to search for Eulerian Cycle also in multigraphs.

2. Hamiltonian Cycle (HC)

Method

In this experiment sets of random connected graphs were, as previously, created in accordance with the following saturation values: [0.2, 0.3, 0.4, 0.6, 0.8, 0.95] and represented by means of vertex matrix. The process of searching for Hamiltonian Cycle was implemented using DFS and backtracking. Subsequent time measurements (expressed in seconds) were taken on graphs of the number of vertices equal to n, starting from n = 100 up to n = 1000; however, in case of searching time reaching the value 300s, the examination of a graph was stopped. Every outcome is averaged over 50 independent runs (i.e. 50 sets of random generated undirected and connected graphs of different sizes).

Results



Observations and conclusions

Finding Hamiltonian Cycle in a graph is a Strongly NP-Complete problem. The algorithm implemented in the experiment searches for HC using DFS and backtracking and its complexity is at worst O(n!). The worst cases can be easily found in the chart – these are graphs of the lowest saturation value (20%) which average searching time is over 200s for every n (and probably would be even larger if it were not for stopping condition). It is the result of sparse graphs having low probability of HC existing. If a graph does not have HC, the algorithm tries different vertices as a next candidate on each recursive level until the absence of HC is confirmed or time limit is reached (the latter situation occurs repeatedly). The time needed to find HC in a graph decreases with increasing number of edges as one can see in the figure – in dense graphs HC is often found within just a few seconds.
Remark:
Despite time measurements being averaged over 50 independent runs, there are still some deviations visible in the chart which is caused by the randomness of generated graphs.

Chosen representation of graph

The graph representation chosen for finding HC is, same as before, vertex matrix. The justification as well as advantages and disadvantages of such a choice have been further explained in the first task.

Sources

https://eduinf.waw.pl/inf/alg/001_search/0135.php

https://www.geeksforgeeks.org/hamiltonian-cycle-backtracking-6/