TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Nikita Golovin 155419IAPB

VEEBIRAKENDUSE LOOMINE SUURIMA KLIKI JA VÄRVIMISE ALGORITMIDE TÖÖJÕUDLUSE KONTROLLIMISEKS

Bakalaureusetöö

Juhendaja: Deniss Kumlander
Doktorikraad

Tallinn 2018

TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technologies

Department of Software Science

Nikita Golovin 155419IAPB

WEB BASE RESEARCH TOOL FOR MAXIMUM CLIQUE AND VERTEX COLORING PROBLEMS

Bachelor’s thesis

Supervisor: Deniss Kumlander

Doctor degree

Tallinn 2018

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Nikita Golovin

21.05.2018

Annotatsioon

Infotehnoloogia valdkond maailmas areneb väga kiiresti. Tänapäeval on loodud sellised tehnoloogiad nagu tehisintellekt või arvuti loogika, mis võimaldavad lahendada palju probleeme. Enamik nendest probleemidest on seotud graafiteooriaga. Nagu me teame, *NP*-keerukuse probleemid on selle osaks. Sellised probleemid on väga tõsised, nii et teadlased loovad algoritme nende lahendamiseks alates XIX sajandist. Teadlased kasutavad graafe, et keskenduda probleemide põhjustel. Selle töö põhieesmärk on luua tööriist ehk veebirakendus maksimaalse kliki ja graafi värvimise probleemi analüüsimiseks.

Selle teesi alguses on probleemi kirjeldus. Kuna probleem on seotud graafi teooriga, siis selle põhiterminid ja nende definitsioonid on ka kirjutatud töö sissejuhatuses. Seejärel läheb maksimaalne kliki- ja värvimis probleemide kirjeldus ning kasutatud algoritmide ülevaade. Pärast ülevaadet läheb arendusvahendi valiku põhjendused, rakenduse vormi ja selles projektis rakendatavate kasulike funktsioonide kirjeldus koos näidetega. Veebirakenduse loomise protsess on kirjeldatud peatükis 4. Kokkuvõtet ja tulevaste uuringute võimalusi saaks leida viimases peatükis.

Teesi tulemuseks on veebirakendus suurima kliki ja värvimise algoritmide tööjõudluse uurimiseks. Veebirakendusel on *MVC* mudel, ning selle arendamisel on kasutatud: *front-end*: Angular5, Typescipt ja Bootstrap, aga algoritmid realiseeritud C# programmeerimiskeeles. Algoritmid olid juba ette antud, ning autor vajadusel tegi muutusi, selleks et adapteerida neid veebirakenduse jaoks.

Tulevikus võiks veebirakendust täita uute tehnoloogiate ja funktsioonidega, nii et selle tööjõudlus ja kiirus muutusid paremaks ning kiiremaks.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 40 leheküljel, 5 peatükki, 28 joonist.

Abstract

World of infotechnology is growing rapidly. Nowadays technologies such as artificial intellect or computer logic are resolving lots of problems. Mostly these problems are related the topic of graph theory. As we know *NP*-complete problems is a part of it. These type of problems still remain important, because scientists are finding solutions of them and creating the algorithms since 19th century. Graphs are used by researchers to focus on the root of the problems and show relations between the considered objects.

The main purpose of this thesis is to create web base research tool for maximum clique and vertex coloring problems. It is focused on the maximum clique and coloring algorithms and web application development.

First of all, thesis starts from the basics of graph theory introduction. Then goes maximum clique and coloring problems description and algorithms overview. After the overview author describes the main contribution – web application. Description consists of the development tool selection reasons, the form of the application and useful features applied in this project. Moreover, user interface aspects, which were taken to a count, were detailed described with the examples. After that, author describes web application development process with examples. At the end, the author makes analyze of the work and gives explanation whether this work achieved its goals.

The result of the thesis is a web application, which is a utilitarian tool for future researches. The application has *MVC* model and for its develepment were used programming languages: for front-end side: Typescript with Angular5 and Bootstrap library and algorithms are written with C#. In the future it could be developed with new technologies and features, so that it will have better performance and speed.

The thesis is in English and contains 40 pages of text, 5 chapters, 28 figures.

Abbreviations glossary

DIMACS	*The Center for Discrete Mathematics and Theoretical Computer Science*
DSatur	*Degree of Saturation*
IDO	*Incidence Degree Ordering*
JP	*Jones and Plassmann is referenced in case of JP algorithm*
LDO	*Largest Degree Ordering*
NP	*Nondeterministic polynomial time complexity class. Class of problems that can be solved with a polynomial amount of time by nondeterministic Turing machine.*
PSL	*Parallel Smallest-Last*
V2	*Version 2, used in Dsatur algorithm.*
V3	*Version 3, used in Largest-First algorithm.*
VColor-u	*Vertex Color unweighted Maximum clique algorithm created by D.Kumlander[1].*
VColor-BT-	*Vertex color with backtracking for unweighted cases. Exact maximum clique finding algorithm published by D. Kumlander [1] based on Östergård's algorithm. The main idea is to apply vertex coloring with backtracking for fastening maximum clique finding.*
VColor-u	*Vertex color for unweighted cases. Exact maximum clique finding algorithm published by D. Kumlander [1] based on Carraghan and Pardalos algorithm [14]. The main idea is to apply vertex coloring for fastening maximum clique finding.*
VRecolor-BT-u	*Vertex recolor with backtracking for unweighted cases. A new exact algorithm presented in the current thesis based on VColor-BT-u. The main idea is to apply additional in depth coloring (recoloring) to fasten maximum clique search.[12]*
RANDOM	*Randomly generated graph by the program.*
CBC	*Current best clique*
UI	*User interface*
MVC	*Model View Controller model*
GPS	*Global Positioning System*

API	*Application Program Interface*
JSON	*JavaScript Object Notation, datatransfer format*
HTML	*HyperText Markup Language*
CSS	*Cascading Style Sheets*
MIS	*Maximum Independent Set* - *the largest possible subset S in a graph.*

Table of Contents

List of figures

1 Introduction

The Swiss mathematician Leonard Euler is the founder of the graph theory. In one of his letters, he formulated and proposed a solution to the problem of seven Koenigsberg bridges, which later became one of the classical problems of graph theory. The term "graph" was firstly introduced by Sylvester J.J. in 1878 in his article in science magazine „Nature“ [21].

Graphs have been used for a long time to solve complex problems. The problem of seven Koenigsberg bridges is the eldest example, where graph theory was applied. The main point of this problem was whether it is possible to cross all the bridges with one walk without recrossing the same road multiple times. Other historic problem connected to the graph theory is the theorem of four colors. The main idea of it says that any map located on the sphere can be painted with no more than four different colors (paints) so that any two regions (areas) with a common border segment are painted in different colors. F. Guthrie from England formulated this theorem in 1852, but could not prove it for a long time.

Scientists from the University of Illinois K.Appel (8.10.1932 – 19.04.2013) and W.Haken (born 21.06.1928) in 1976 got success in proving the four-color theorem [22]. While proving this theorem scientist used computer, so that it was the first graph theory theorem proved by computer. Starting from 18th century, researchers resolving complex problems and creating new algorithms, which have better performance and speed of completion.

People have been using graph theory during long period of time in different fields of everyday life. Graph theory is used in social networks, *GPS* navigation, coordination of airport activity (aircrafts landing queue), telecommunication networks and etc. It helps people to resolve complete tasks and *NP*-complete tasks. Maximum clique and coloring problems are part of the *NP*-complete problems. *NP*-complete tasks are still remain actual, because scientists have not found an ideal algorithm for finding clique.

The aim of this work is to help scienists finding better algorithms and analysing their performance. Web application gives opportunity for portable analyze with visualization of the results, which could be understandable for every user. It helps writing research works, because results are available to download in Microsoft Excel file.

1.1 Graph and terms

This subchapter explains main terms for better undestanding of the processes in this thesis.
Graph is a mathematical object, which consists a set of vertices and a set of edges, connections/relations between pairs of vertices. For instance, number of vertices in graph

called the order of graph and size of the graph means number of relations(edges/pairs) in the graph. Strict science definition could be depicted by formula:

$G = (V,E)$

G – graph (pair of sets V and E), V – set of vertices, E – set of edges

Talking about independent sets, the problem related to them (Maximum independent set problem) is closely related to maximum clique problem. The main aim of these MIS problems is trying to find a subset $S \subseteq V$ so that no two vertices in S were adjacent to each other.
In essence subset S is an empty graph.

The important point in graph is the connections between the vertices, not their location on a diagram. For better explanation, Figure 1-1 is added below, which shows exactly the same graph, but the vertices have different location.

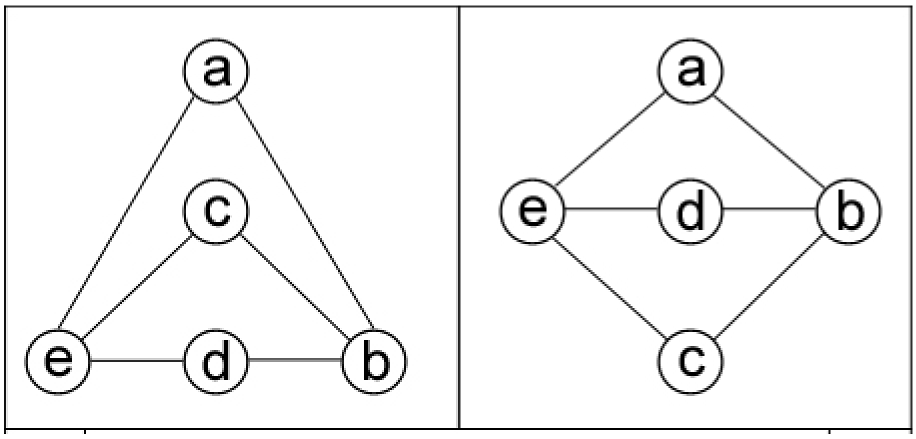


Figure 1-1 Different types of the same graph representation

Note: In this project author has used undirected, unweighted simple graphs.

Graphs can be divided into directed and undirected. A directed graph (demostrated on the right at Figure 1-2 has directed edge, which is called arc. Therefore, it san be seen that vertex v1 have relation to vertex *v2,* but there might not be relation from *v2* to *v1*.

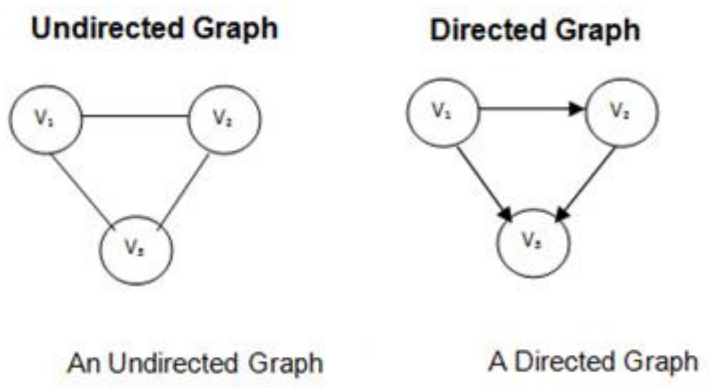


Figure 1-2 Undirected and directed graphs

Talking about other graphs' characteristics, graphs are divided to weighted and unweighted. Weight is a number, usually it is non-negative value, which is assigned to each edge or vertex. This number gives us additional information. For example, it could be length of a route(in case of *GPS*), cost of the flight tickets or other sort of property in case of the problem core. On the right diagram of the Figure 1-3 is the unweighted graph. It does not have any additional values (weights). It means that all weights are equal to one.

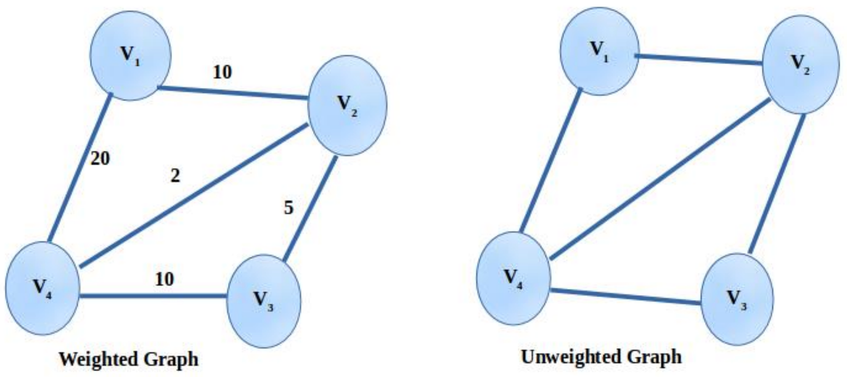


Figure 1-3 Weighted and unweighted graphs

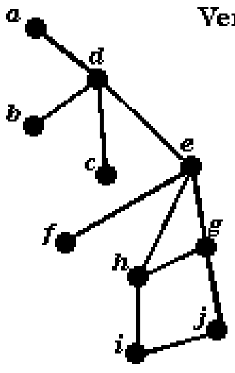
Loop is an edge that connects a vertex to itself. **Simple graph** is an undirected graph that does not contain any loops and there is no more than one edge connecting two vertices [24].
An undirected graph is called complete, when all the vertices are adjacent to each other. Talking about opposite situation, if a graph has no edges, it is called edgeless (no two vertices are adjacent to each other).

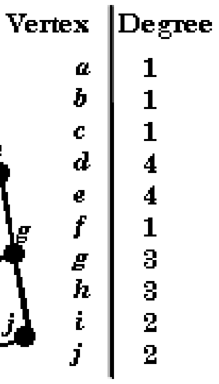
Vertex degree *deg(v)* - number of adjacent vertices or neighbours of a vertex. Vertex can be even or odd depending on the vertex degree number, whether it is even or odd.

If there is given graph *G*, it means that ∆ *G* - maximum vertex degree of a graph *G*. Vertex support value and maximum vertex degree value are different things. Vertex support value is a sum of degrees of all neighbours of a given vertex. For for better explanation, all calculations and diagram (Figure 1-4) are given below.

Degree of vertex *e* is 4. It means that e has 4 connections*: e – d, e – f, e – g, e – h*. But, talking about support of the vertex *e,* it is equal to 11.

Support of *e* equals: *4 (vertex d) + 1 (vertex f) + 3 (vertex g) + 3 (vertex h) = 11.*





Vertex	Degree
a	1
b	1
c	1
d	4
e	4
f	1
g	3
h	3
i	2
j	2

Figure 1-4 Graph diagram

The density *g (G)* of the graph is ratio of the graph’s vertices number *n* = |*V*| to the number of the graph’s edges *m* = |*E*|. Formula for finding density [5]:

$$g(G) = \frac{2 * m}{n * (n - 1)}$$

Figure 1-5 Graph diagram [5]

Complement graph $\overline{G}$ for graph G has edges (connections) only between these pairs of vertices, which graph G does not. Figure 1-6 demonstrates the difference. If we unite both graphs (G and $\overline{G}$) in one diagram, we will get complete graph, because each pair of vertices will have 1 edge between each other [2].

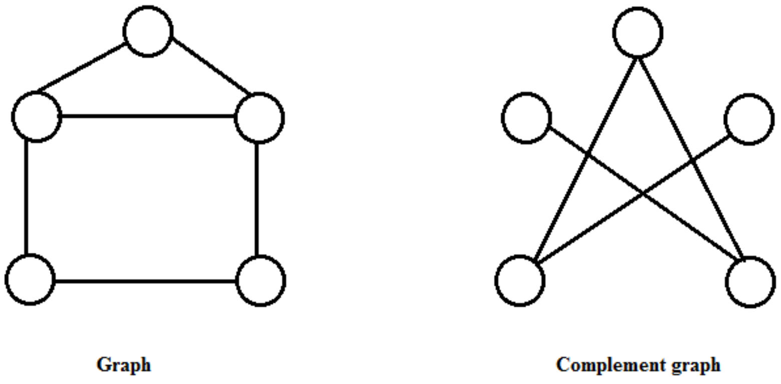


Figure 1-6 Complement graph diagram [5]

In order to use some terms later in the Chapter 2, author gives explanation of terms here, which are directly connected to the maximum clique terms. The clique is a complete subgraph of the graph. Subgraph is a subset formed by vertices with corresponding edges. It is not obligatory fact, that subgraph should have all edges, which have graph. Vertices can have relations in a graph, but in case of subgraph they might not have any edges between each other.

A complete subgraph is a subset of vertices of the graph, where each vertex is connected by edges to other vertices of this subgraph [5].

Clique is called maximum clique, if there is no more other cliques bigger in the same graph. In other words, if considering clique is not a part of the other clique.

Heuristic solution is a type of techniques for solving a problem in case when classic methods are too slow or fail in finding any exact solution. This type of techniques finds more quickly, but approximate solution.

For example, we have some problems, which classic methods are not able to resolve in acceptable time and get acceptable solution. We can use heuristic method such as arbitary choices in order to get not bad solution much faster.

Graph coloing – distributing nodes of the graph by color assingment. It means that nodes of the same color can not have relations between each other. Color is just identification of the node.

1.2 DIMACS and Random graphs

In many researches scientists are testing maximum clique and coloring algorithms on *DIMACS* and *RANDOM* graphs. Web application created in this project gives opportunity to test algoritms' performance on the both types of graphs.

The Center for Discrete Mathematics and Theoretical Computer Science (*DIMACS*) is a collaboration between Rutgers University, Princeton University, and the research firms AT&T, Bell Labs, Applied Communication Sciences and NEC. It was founded in 1989 with money from the National Science Foundation [18].

One of the *DIMACS* Challenge purposes is to ease the effort required to test and compare algorithms and heuristics. Center for Discrete Mathematics and Theoretical Computer Science provides a pack of benchmarks instances, which represent different graphs, constructed on the real life problem basis. These instances can be used for testing maximum clique algorithms performance. Talking about other the *DIMACS* center's activities, scientists also test many assumptions about implementation methods and data structures. As it was mentioned before, *RANDOM* graphs can also be tested with web application. User gives the input parameters and application generates the graph by itself. Code is demonstrated on Figure 1-7. As input parameters it has number of nodes and density of the graph.

```
public static Graph GenerateGraph(int nodes, double density)
        {
            if (density < 0 || density > 1)
                throw new Exception("0 <= density <= 1");
            int numberOfEdges = Convert.ToInt32(Math.Round(nodes * (nodes -
1) * density / 2, 0));
            var graph = new Graph
            {
                Values = new bool[nodes, nodes],
                Edges = numberOfEdges
            };
            var random = new Random();
            Thread.Sleep(40);
            var random2 = new Random();
            int x, y;
            for (int i = 0; i < numberOfEdges; i++)
            {
                do
                {
                    x = random.Next(0, nodes);
                    y = random2.Next(0, nodes);
                } while (x == y || graph.Values[x, y]);
                graph.Values[x, y] = true;
                graph.Values[y, x] = true;
            }
            return graph;
        }
```

Figure 1-7 Graph geretion code

1.3 Complexity and NP-complexity

„How fast or slow does particular algorithm perform?“ - this type of question belongs to algorithmic complexity topic. We define complexity as a numerical function *T(n)* - time versus the input size *n*.

Complexity of the algorithm is a function *f: N—›N* [11].

It can be divided on time complexity and memory complexity. Time complexity is an algorithm search performance dependance from the time comsumed and memory complexity is a dependence from the used memory for finding the result.
Every algorithm has a logic for finding solution of different kind of problems. Talking about maximum clique problems, these kind of tasks does not have guaranteed result. There is no algorithm, which could completely solve maximum clique problem. It is more complicated type of the problem, which called *NP*-complete problems. These problems are hard to solve and nowadays they do not have guaranteed solution.
Alan Turing in 1936 brought to the science world first serious result in algorithms complexity field. Alan invented an abstract computer model called Turing machine[20].
P class problems can be solved with a polynomial time on a deterministic Turing machine. *NP* class problems are solvable by non-deterministic Turing machine in polynomial time.

1.4 Goals of study

The graph theory and projects related to this topic are extensive.The goals were set up based on the assigned task: Web application for maximum clique and coloring algorithms analyze.

1. Create utility WEB application in order to help researchers test maximum clique and coloring algorithms
2. Study modern algorithms for finding maximum clique
3. Visualize maximum clique and coloring algorithms results in order to help researchers quickly analyze differences between algorithms’ performance
4. Create opportunity for researchers to test maximum clique and coloring algorithms on *DIMACS* graphs and on generated *RANDOM* graphs
5. Build a foundation for exploring maximum clique for end-user research

1.5 Work overview

In the chapter 1 of this thesis is the problem introduction and main terms related to this topic. There is an overview on the types of graphs and *NP*-complexety. After that, goals of study are represented. Chapter 2 describes maximum clique and coloring problems. This chapter contains short overview of the maximum clique finding algorithms and other heuristic algorithms like coloring used in the web application. Firstly, there are described basic algorithms, which later became fundamental. Chapter 3 represents web application development tools and explaines author’s choice of the tools and desicions made during development based on the requirements. There is a description of application model and used technologies. The main focus of the chapter 4 is to describe the development process such as front-end and required back-end. Moreover, it describes the ways, how application could have high discoverability. Finally, Chapter 5 contains a summary of the study. Possible opportunities for future studies are also noted there.

2 Maximum Clique and Coloring algorithms

This chapter starts with an overview of maximum clique and coloring problem. Afterwards, it describes basic algorithms for solving these problems such as: greedy, branch and bound algorithms. After, in the next subchapter will be described modern algorithms and will be noticed that they are based on basic ones. The topic of this thesis consists of maximum clique and coloring algorithms, so that later will be described algorithms for resolving these type of problems.

2.1 Maximum Clique and Coloring problem

The root of the maximum clique problem is finding the biggest complete subgraph. As it was mentioned before, this type of problem is called *NP*-complete, because it is very complicated to find solution with usual methods and techniques. The first and the most famous algorithms, later became the basis of modern algorithms, were created by Randy Carraghan and Panos M. Pardalos in 1990 and in 2002 by Patrick R.J. Östergård. The purpose of existing algorithms modification is to invent algorithm with better quality and performance. In other words, find the best solution in a shorter period of time. If we compare modern and basic algorithms logic, the difference will be clearly seen: old algorithms are focused more on adjacent vertices checking vertices that are not adjacent, while modern algorithms depend on heuristics, more precisely on coloring.

The problem of coloring the graph is defined as the task of assigning colors to the vertices of the graph in a way that no pairs of adjacent vertices has not got the same colors. The number of colors used in the process should remain as small as possible. As smaller the number of colors as better algorithm, but we always should take to the count time consumption.

The coloring of the graph allows you to build a number of independent sets and to use the additional characteristics of the information gathered, and the rational use of heuristic solutions does not increase the time significantly.

2.2 Basic algorithms

These algorithms are Carraghan and Pardalos algorithm, Östergård algorithm and greedy algorithms. They are a basis for the modern algorithms, so that their description is more detailed in this subchapter.

2.2.1 Carraghan and Pardalos algorithm

In 1990 was published branch and bound algorithm in article “An exact algorithm for the maximum clique problem” written by Randy Carraghan and Panos M. Pardalos [14]. It still remains simple and efficient. But it’s efficiency could be noticed only on lower density graphs. The algorithm itself is a good and simple example of finding clique. As an example, below is added pseudo code of this algorithm (Figure 2-1).

```
function Main
     CBC := 0 // the maximum clique's size
     clique (V, 0)
     return CBC
end function

function clique(V, depth)
     if |V| = 0 then
          if depth > CBC then
               New record - save it.
               CBC := depth
          end if
          return
     end if
     i := 0
     while i < |V| do
            if depth + |V| - i ≤ CBC then // prune
                 return
            i := i + 1
            // form a new depth. N(v_i) denotes a neighborhood of v_i

            clique (N(v_i) | ∀v_j : j > i, j ≤ |V|, depth + 1)
       end while
       return
  end function
```

Figure 2-1 Carraghan Pardalos algorithm. Pseudo code [1]

Carraghan and Pardalos algorithm's concept is creating branches of the vertices. In other words, it has depths (levels). First of all, it expands vertex "*ver1*" from the first depth. Then at second depth it considers only these vertices, which are adjacent to "*ver1*". Then expand suitable vertex "*ver2*" from the second depth. Next step of the solution is to build-up third depth from the vertices, which are adjacent to "*ver1*" and "*ver2*". Every set of depths is one branch in the solution [1].

CBC is a variable, where we store the biggest depth number found by far. It needs to be stored for prunning the branches. Formula of prunning is pretty simple:

$$depth + num - cur \leq CBC$$

depth - current depth
num - number of vertices in a current depth
cur - currently expanding vertex
This formula can be described like this: if current biggest possible clique size is not bigger than *CBC*, it means that we can prune this branch.

2.2.2 Östergård algorithm

After twelve years since Carraghan and Pardalos article was published, in 2002 Patrick R.J. Östergård published “A fast algorithm for the maximum clique problem” article [15]. This algorithm is a reversed variant of Carraghan and Pardalos algorithm, because it starts from the end. In other words, it starts with n-th depth subgraph, where only vertex located and clique size is 1 initially. Clique sizes of subgraphs are stored in cache. Next step is to consider (*n-1*)th subgraph containing 2 vertices. New prunning formula could be used in this algorithm:

$$depth + cache[i] \leq CBC$$

depth – current depth
i – vertex currently being expanded
CBC – currently biggest clique

But more important point is condition, which stops further search.

$$depth + cache[i] > CBC$$

Potentially *CBC* value could be increased only by one, because only one vertex added to a new *n*-th depth subgraph compared to (*n+1*)th depth subgraph.

2.2.3 Greedy algorithm

The most productive and the simplest heustic coloring algorithm is Greedy algorithm. This algorithm is a mix of speed and performance [1]. Moreover, it is easy to implement. However, number of color classes sometimes is not close to chromatic number. We have list of colors, our purpose to color all vertices and use as less color classes as possible. First step is coloring the first vertex „*ver1*“ in color „*color1*“. Number of used colors equals 1. Then it takes other uncolored vertices and tries to color them, so that there will not be any adjacent vertex to „*ver1*“ with the „*color1*“ color. Important condition: If it is not possible to color a vertex into any of existing colors, a new color must be created and assigned, so that number of used colors will be increased. And algorithms repeats steps two and three until final result - all the vertices should be colored. Because of its simplicity and productivity, lots of modern algorithms are based on greedy algorithm.

```
// n - number of vertices, k - number of colors on each step
k = 1; Color $v_1$ with $C_1$ ($C_k$)
For i := 2 to n
     Try to color $v_i$ with color $C_j$, where j = min (1, ... , k)
     If none color was used to color $v_i$ then
          k := k+1 [Produce a new color]
          Color $v_i$ with $C_k$
     End if
       Next
```

Figure 2-2 Greedy algorithm pseudo code

2.3 Modern algorithms

This subchapter describes modern algorithms used in this project. Some of them are based on basic algorithms, which where described before. But, some of algorihms described in this subchapter are the improved variations of the modern algorithms with another approach.

2.3.1 Vcolor-U algorithm

“Some Practical Algorithms to Solve The Maximum Clique Problem” thesis Deniss Kumlander published in 2005 [1]. VColor-u, the whole name is Vertex Color unweighted algorithm, was introduced in this thesis. The author of this algorithm wanted to demonstrate efficiency of using independent sets within clique finding algorithms.
The difference between previous described algorithms and VColor-u is that Vertex Color unweighted algorithm demonstrates fine performance on high densities. This algorithm is based on Carraghan and Pardalos approach.
D. Kumlander noted that there is no any other optimizations than vertex coloring. It is done to evaluate influence of coloring on overall performance purely [1].

2.3.2 Vcolor-BTU

The idea of the second algorithm introduced in the same article was to apply initial vertex coloring on the bases of the Östergård’s algorithm. The main idea is to apply additional in depth coloring (recoloring) to fasten maximum clique search. The only distiction of the algorithm is that it operates with independent sets instead of single vertices.

2.3.3 Vrecolor-BTU

This algorithm is based on VColor-BT-u was introduced in master’s thesis made by A. Porošin [12]. As the author of the algorithm noted, the idea of this one is to gather and combine all the gained knowledge to fasten maximum clique finding even more. It can be clearly seen from the modern algorithms that almost all of them are focused on Carraghan and Pardalos approach and only VColor-BT-u implements Östergård’s idea.

2.3.4 DSatur-LDO

DSatur-LDO is a coloring algorithm, with combination of two algorithms. Basically this algorithm works in the same way as DSatur, but when there is a conflict with finding solution, it uses logic of the Largest-First algorithm and resolves the problem. The Largest-First algorithm orders the vertices by the number of neighbors and then the Greedy coloring begins. The basic idea of the Largest-First algorithm is that vertices with more neighbors are colored first of all, because the most conflicts are caused by them.

2.3.5 Largest FirstV3

First of all, the Largest-Frist algorithm orders the vertices by the number of neighbors and then starts Greedy coloring. Largest-First V2 is an upgraded version of the Largest-First algorithm. In this algorithm, it can be colored more than one vertex in every iteration. For example, after coloring of the vertex with the biggest number of neighbors, the algorithm gives the same color for all other vertices, which have unpainted neighbors [5]. Finally, these vertices are removed from the graph.
Largest-FirstV3 is the third version of the Largest-First algorithm. Generally it is the same as the Largest-First V2, but in each iteration reordering is occured. Removing a painted vertex from a graph causes a decrease of the edges number between the removed vertex and neighbors [5].

2.3.6 DsaturV2

The DSaturi algorithm orders the vertices by decreasing their saturation levels. In other words, on the first place is a vertex whose neighbors have used the largest number of different colors. Reordering happens in each iteration. In the case of conflict(if the number of different colors is the equal), algorithm selects vertex according to the largest number of non-colored neighbors [1]. DSatur V2 is DSatur's second version.
In the second version, the first step is to find clique of the graph, and then determines colors fo each vertex in the clique. Finally, these colored vertices are removed from the graph and the work continues starts again the same way as in the original version of DSatur.

2.3.7 DSatur-IDO-LDO

Again we deal with an algorithm consisting of combination of three. In this algorithm, the conflicts are firstly solved by the IDO algorithm, and then the remaining conflicts are solved by the LDO algorithm [13]. IDO is an advanced DSatur algorithm. Vertices are ordered by the number of their painted neighbors and if there are two vertices that have the same number of colored neighbors, it uses random numbers to determine the order of the vertices. For the coloring it uses greedy algorithm.

2.3.8 LDO-IDO

The LDO-IDO depends to a group of coloring algorithm with combination of other algorithms. The main heuristics of this algorithm is the Largest-First algorithm. In case of conflicts, IDO heuristics decides which vertex to choose. Generally, this algorithm is similar to the Largest-First V3 algorithm, but it has the IDO function, which means that the first ordering is made by the largest number of neighbors and only then by the largest number of colored neighbors [5].

2.3.9 Parallel Largest-First

The Jones and Plassmann algorithm is a basis for the Parallel Largest-First algorithm, but the heuristic part is the Largest-First algorithm. The basic idea of a JP algorithm is to create a unique set of weights at the beginning of the algorithm that is used throughout the algorithm. For example, it uses random integers. The numbers of the vertices will be used for the same random numbers in the collection. In each iteration, the JP algorithm finds an independent group in a graph, and then assigns colors to these vertices using the Greedy algorithm. All activities are carried out in parallel [5].

The Parallel Largest-First algorithm finds the maximum degree for each vertex. Random values are also used to help solving a problem where two vertices have the same number of neighbors.

3 Web application

Purpose of this project is to creat application, which users (researchers) can use to analyze and visualize maximum clique and coloring algorithms. Web application should have simple for user to understand interface and do implementations with high performance. It means that the application should give user opportunity to make analyze of algorithms' performance and compare algorithms' with each other. Moreover, application should not use a lot of resources from the user's device, but at the same time it should be pretty fast. Besides visualisation of the results, application gives user opportunity to download Excel file with results table. User can choose what algorithms will be tested (Maximum Clique or Coloring) and implement algorithms on *DIMACS* graph by downloading it from his device or create *RANDOM* graph by filling the inputs in the web application. The main points of maximum clique and coloring algorithms research are:

1) Number of colours
2) Maximum clique value
3) Time consumed in order to find solution by algorithm

Application visualize results in tables and charts, so that user could use them in his/her research.

3.1 Difference between Web and Desktop applications

The purpose of thesis is to create application. In order to achieve portability and platform independence (can be used on different devices) was set the list of requirements after discussion with the supervisor. As the best solution was chosen web application. **Web application** is a client–server computer program. The client part implements the user interface, generates requests (client-side logic) to the server and processes the responses from it. Despite having a complicated structure, application interface is brought down to a level of a regular user. The client is the browser, the server is the web server. Communication is implemented via network (Internet connection). Web application consists primarily of pages with partially or completely undefined content. The final content of web pages will be formed when a particular user sends a request. In this project user should fill the inputs, choose the algorithms from the list and run the application. Then input information will be formed as a request to the server-side.

The server part receives a request from the client, performs calculations, then forms a web page and sends it to the client over the network.
In other words, this is an Internet application with a client-server architecture. For better understanding of its working principe, the basic elements of such architecture are named below.

From the server side, different technologies and any programming languages can be used for developing web applications. Moreover, it does not matter which OS is configured on a user device, so that Internet applications can be considered as universal cross-platform services.
The standards for any web development products are generally common. Functional aspects are based on the implementation of functions needed to solve user tasks. Other positive points of choosing web application instead of desktop app are developing, portability and sharing. Updating and developing is done on the server, the user does not care how it is executed by developers on the server. It means, that user does not need to download any programs of his/her device. Cross-platform usage, in other words, location of the user generally is not important, browser is the only thing needed. Tools, software on the server are also cross-platform.

Functionality is the most important requirement for any software product. Historically, desktop applications are considered to be more functional and ergonomic. But over the years, different interface libraries have been developed significantly. As far as author's application concerned, for *UI* and visualization of the results were used Bootstrap and Chart.js libraries. Nowadays developers can use lots of features to improve functionality of the web application:

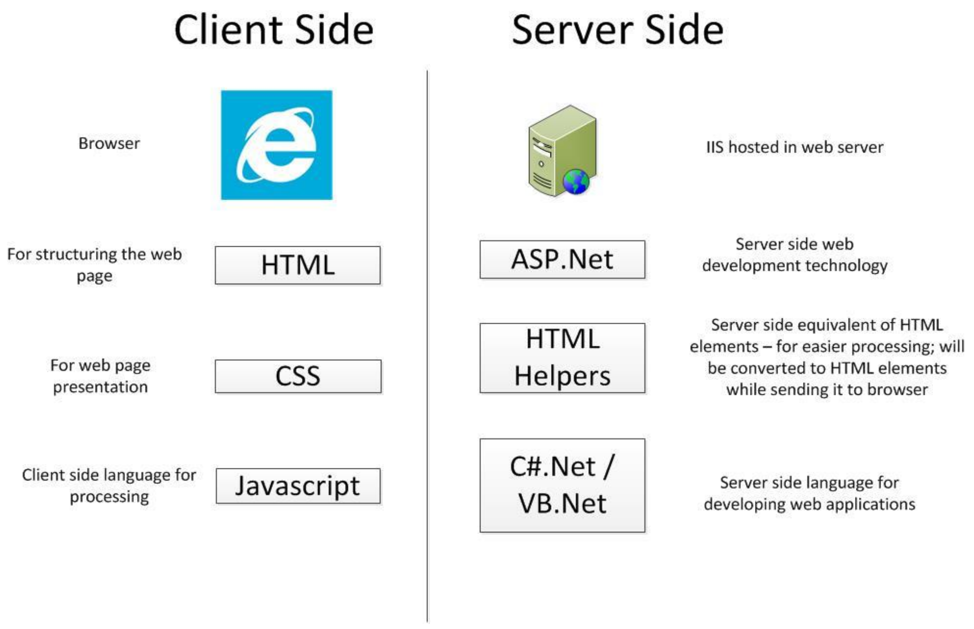
- hierarchical lists with the ability to move columns and apply filters;
- drag & drop with any visual effects;
- interactive panels, business graphics and charts;
- audio and video players (Flash player);
- and others.

On the other hand, speed and performance of the web application depends on Internet connection and poor connectivity can cause performance issues with web application. Desktop applications are standalone, that's why they do not face any hindrances from Internet connectivity. Moreover, as web applications are internet dependent, they cost more bandwidth usage than desktop applications [17].

Security and reliability usually are a very serious issues. Some organizations fundamentally do not want and do not provide the opportunity to work in corporate systems outside their domain. The need for the usage of cryptographic information protection and electronic signature has not been proven to anyone for a long time. To use these technologies, application needs to access third-party libraries, but not all web applications can do this and often have some limitations. The stability of the browsers themselves is also potentially a weakness, and it can not always be affected by the developer of a business application [16]. Offline work, objectively, more often and more easily implemented by using desktop applications. In case of individual organizations, they are still frightened and working in the browser very carefully. However, considering application for finding maximum clique and coloring algorithms security aspect is not important, because there is no personal information or even databases used and can not cause any serious issues.

3.2 MVC model

The *MVC* design pattern involves splitted application data into three separate components: Model, View (the user interface), and Controller (control logic),- so that each component can be modified independently. The view is responsible for the user's visible display of this data (in case of web, it forms *HTML / CSS* given from the server to user browser) and the Controller manages all processes [19].



Client Side
Server Side
Browser
IIS hosted in web server
For structuring the web page
HTML
ASP.Net
Server side web development technology
HTML Helpers
Server side equivalent of HTML elements – for easier processing; will be converted to HTML elements while sending it to browser
For web page presentation
CSS
Client side language for processing
Javascript
C#.Net / VB.Net
Server side language for developing web applications

Figure 3-1 MVC model

Figure 3-1 shows the basic components and programming languages of the *MVC*. User fill the inputs with information. This information is formed into the request. Data transfer is usually realised by *JSON*. On a server-side controller receives request and run the algorithms with given from request parameters. The results are formed to the response, which *JSON* sends back to the client-side. The client-side gets the response and performs the visualization of the results to the user. Despite the fact that all these parts are independent for developing, they exchange the information to reach the final goal of the application.

3.3 Tools

Before starting application development were discussed with superviser list of tools, which were used in order to achieve the best result. The main factors, which were taken to the count, are: programming languages, flexibility of usage, powerful built-in tools and features and worldwide popularity. This part describes used for developing tools and explanation of choice.

3.3.1 Typescript+Angular5+Bootstrap

TypeScript is a typed superset of JavaScript that compiles to plain JavaScript [4]. JavaScript is most common and well-known programming language for web pages. It is also used on many platforms. JavaScript runs on the client-side on the web and its function is to check the behavior and functionalit of the website. In other words, JavaScript is a parent of TypeScript. The autor has chosen it, because it is rather modern and offers support for the latest and evolving JavaScript features, including those from ECMAScript 2015 and future proposals, like async functions and decorators, to help build robust components [4].
Angular 5 is the latest famework for developing applications with JavaScript especially TypeScript.
Angular turns templates into code that's highly optimized for today's JavaScript virtual machines, giving you all the benefits of hand-written code with the productivity of a framework [9]. There is also lots of tutorial videos and documentation available in the Internet. It was decided to use Angular 5, because comparing it with Vue or React, it shows good performance and speed and it decrease its reliability. Moreover, it has lots of good educational videos and documentations, which were easily found.
Bootstrap is a library for the front-end development. It has extensive prebuilt components, and powerful plugins built on jQuery. It helps to perform finctional *UI* with high discoverability.
Bootstrap is an open source toolkit for developing with HTML, CSS, and JavaScript [8].

3.3.2 Webstorm

Webstorm is product of JetBrains company. During last 15 years they have been developing the strongest, most effective developer tools. As a proof of their quility, JetBrains has won over 50 international industry awards in the past 15 years.
„By automating routine checks and corrections, our tools speed up production, freeing developers to grow, discover and create. We make professional software development a more productive and enjoyable experience.“ – this is a slogan (idea) of the company [23].
Webstorm was chosen because of its smart coding assistance for JavaScript and compiled-to-JavaScript languages, Node.js, *HTML* and *CSS*. Moreover, it gives opportunity for future development of the web application, because it provides advanced coding assistance for Angular, React, Vue.js and Meteor. Nowadays mobile development industry is growing rapidly and Webstorm supports for React Native, PhoneGap, Cordova and Ionic development.
Being a student of Tallinn university of technology, author have got official license for using advanced version of Webstorm given by the university. For the front-end development was chosen Javascript, so that the main positive points of Webstorm are more compact and lightweight documentation popup and Typescript support.
It uses a more simple and consistent format to present developer the available information about parameters in methods, their type and type of return values in JavaScript and TypeScript.

3.3.3 Visual Studio

Visual Studio is a Microsoft product. The supervisor provided author with maximum clique and coloring algorithms made with C# programming language. In this case Visual Studio is the most suitable development enviroment, which helps developer write program code and run the project to see the result offline. For developing web application was used Visual Studio Community 2017, the latest version from any available.

3.3.4 IIS Manager

IIS Manager(previously known as the Web Management Tool) is a internet information service tool, which is a part of every Windows OS. Windows server is used for implementing application in order to get it available for regular internet users. Moreover, Visual Studio gives opportunity to test the application on local machine with running IIS Manager insade of the platform.

3.3.5 Chart.js

One of the most important aspects of the work is to visualize results for the user. After discussion we have decided, that the best way for visualization is a charts. For Javascript the best solution is Chart.js, because it has huge documantation and tutorials. Moreover, it is useful for future development, because it has a lot of plugins and developers can get them, for example, from Github releases. It has MIT license, in the other words, it is free software license. As an example, color chart code is given below:

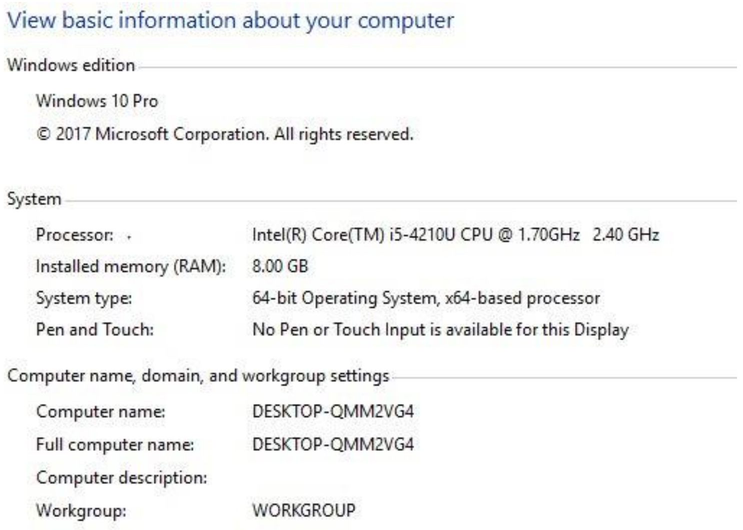
```
//Color Chart
          this.chartColor = new Object();
          var chartColorName = 'chartColor' + this.chartsCanvas[i];

          this.chartColor = new Chart(chartColorName, {
            type: 'line',
            data: {
              labels: node,
              datasets: DatasetColor,
            },
            options: {
              title: {
                display: true,
                text: objRandomResultViewModel.Density + "%",
                fontSize: 20,

              },
              scales: {
                yAxes: [{
                  scaleLabel: {
                    display: true,
                    labelString: 'Color'
                  }
                }],
                xAxes: [{
                  scaleLabel: {
                    display: true,
                    labelString: 'Nodes'
                  }
                }]
              }
            }
    });
```

Figure 3-2 Color chart code adapted from Chart.js tutorial [7]

Note: Charts were done with Chart.js official website documentation [6] and tutorial from the Coursetro website [7].

3.3.6 PC technical characteristics

In this thesis was used for developing and testing author’s personal computer. All decisions results were made depending on its performance and productivity. In next chapters, which describe the development process, it should be taken to the count.



View basic information about your computer

Windows edition

Windows 10 Pro
© 2017 Microsoft Corporation. All rights reserved.

System

Processor:	Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz 2.40 GHz
Installed memory (RAM):	8.00 GB
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	No Pen or Touch Input is available for this Display

Computer name, domain, and workgroup settings

Computer name:	DESKTOP-QMM2VG4
Full computer name:	DESKTOP-QMM2VG4
Computer description:	
Workgroup:	WORKGROUP

Figure 3-3 Used in this project PC information

3.3.7 Visual Studio and Web Api

A server-side web application is a programmatic interface consisting of one or more publicly exposed endpoints –response message system, which are needed to define request, in current project in expressed in JSON, which is exposed via the web. As far as all algorithms are written in C# - Visual Studio Community, which is also open-source tool, is the best solution.

4 Development process

4.1 Requirements analyze

The development process started from the web application requirements analyze. In other words, what should the web application be able to do. After the analyze author set up list of requirements:

- Ability to run algorithms, developed with C# programming language
- Foundation for future development
- 4 types of testing (Maximum Clique or Coloring algorithms with *DIMACS* or *RANDOM* graphs)
- Give opportunity to user upload DIMACS graphs
- Give opportunity to user input parameters for RANDOM graph research
- Modern and understandable *UI* design
- Visualization of the results (tables and charts)

Taking to a count all requirements, was chosen web application form with *MVC* model. As it was mentioned in Chapter 3, *MVC* gives opportunity for development separatly front-end and back-end. This fact is very important, because given to author algorithms are made with C# programming language. Talking about logical structure of web application, it can be devided into 2 components:

1. Maximum clique algorithms testing
2. Coloring algorithms testing

Regardless the type of the graph, for example, the maximum clique algorithms used in testing *DIMACS* graph, are the same, as in testing *RANDOM* graphs. The only difference is the input parameters. The same logic with coloring algorithms.

4.2 Data transfer in the web application

As it was mentioned before, this web application has got *MVC* software pattern. Client-side, in other words front-end, was developed with Typescript, Angular5 framework and Bootstrap library. Client-side is responsible for getting input data from the user.

In case of DIMACS graphs, user upload the file with the graph. The maximum clique with DIMACS graph code is taken as an example:

```
 saveFiles(fileList) {
    this.errors = []; // Clear error
    let formData: FormData = new FormData();
    for (var j = 0; j < fileList.length; j++) {
      formData.append("file[]", fileList[j], fileList[j].name);
    }
```

Figure 4-1 File save code

The file, which was checked by the list of errors. It should be of the correct extension, file size should not more than 10mb.

After uploading the file, user should choose the algorithms. It is needed to put tick in a checkbox of the algorithm.

```
onAlgoCheckChange(algName, event) {

    var curIndex = this.selectedAlgorithms.indexOf(algName);
    if (event.target.checked && curIndex < 0) {
      this.selectedAlgorithms.push(algName);
    }
    else if (!event.target.checked && curIndex > -1) {
      this.selectedAlgorithms.splice(curIndex, 1);
    }
    this.checkedAll = this.selectedAlgorithms.length ==
this.checkAlllst.length;
  }
```

Figure 4-2 Check and uncheck the algorithms with checkboxes

After the filling the inputs, all data should be checked in order to perform proper data transfer. The input information control is represented on the Figure 4-3.

```
uploadData(event) {
    this.elestyle = { 'display': 'block' };
    this.errors = []; // Clear error
    this.validateFiles(this.fileList);
    if (this.fileList == undefined || this.fileList == null ||
this.fileList.length < 1)
      this.errors.push("Please upload file");
    if (this.selectedAlgorithms.length < 1)
      this.errors.push("Please select atleast one algorithm ");
    if (this.errors.length > 0) {
      this.elestyle = { 'display': 'none' };
      return this.errors;
    }
    this.saveFiles(this.fileList);
  }
```

Figure 4-3 Input data control

If the data is correct, it is perfromed to a request and then JSON sends data request to the server-side:

```
     formData.append("CustomFormData",
       JSON.stringify({ Algorithms: this.selectedAlgorithms, Density:
null,SubTests:10 }));
   var parameters = {};
```

Figure 4-4 JSON request

The request is received by MaximumClique Api Controller. It converts the recieved data to a proper format in order to use the information for algorithms running. One of the aims of adapting algorithms for web application was to save the logic of the algorithms and their performance.
After the execution the results are sent to the ResultViewModel. As it was mentioned Model is a part of the *MVC*, which directly manages the data, logic and rules of the application. Then *JSON* sends the formed response back to the client-side in order to visualize the results.
The algorithms, which were given to the author also could create Excel file with the results, so that the author has left this part and end-user can also download the file by clicking on the link.

Talking about *RANDOM* algorithms testing, there are other input parameters such as minimum and maximum density and numbers of the minimum and maximum vertices for each density. The logic of the data transfer remains the same.

4.3 The changes made in the logic of testing algorithms

The main changes were made by author in implementation of the *RANDOM* graphs. The logic explanation is pretty simple:

1) The user inputs the minumum and maximum densities in per cents. The system automatically calculates certain number of researching densities. The step between densities is always 10%. For better understanding, the example is given below:

   Minimum density equals 30% and the maximum equals 60%, it means that research will be done on 4 types of densities: 30%, 40%, 50% and 60%.

2) Each density has got minimum and maximum number of vertices. In order to visualize results on a line chart and show algorithms' performance more detailed, number of vertices also divided into 4 intervals.

```
decimal diff = maxVertices - minVertices;
var div = Math.Ceiling(diff / 4); //divide the vertices
int d = Convert.ToInt32(div);
int row = 1;
int counter = 1;
```

Figure 4-5 Intervals division

3) Next step is start of the algorithms' work with the calculated values.
4) In order to get more accurate result, algorithms are executed 10 times. The number of execution in the code is a Subtest variable. To visualize the average result of the Subtests, was implemented change in the code:

```
//calculate the average
sumColor = sumColor + alg.ColorNumber;
sumTime = sumTime + alg.Elapsed;
lstobj.Add(objRandomColorBenchmarkAlgorithmSubTestDetail);
numberOfSubTest++;
avgColor = sumColor / numberOfSubTest;//average of Color
avgTime = sumTime / numberOfSubTest;//average of Time
```

Figure 4-6 Average value calculation

5) Saving average results in Excel sheets.

```
private void PrintResult(int testNumber, int numberOfSubTest,
int algorithmNumber, int row, double avg, double colorNumber,
int step)
        {
            const int initialRow = 2;
            row = row - 1;// set the average of color and time
value in sheet
            int column = algorithmNumber + 1 + testNumber *
(NumberOfAlgorithms * 2 + 1) + step;
            using (var doc = new ExcelPackage(new
FileInfo(CurrentFile)))
            {
                var wb = doc.Workbook.Worksheets[1];
                wb.Cells[row + initialRow + numberOfSubTest,
column].Value = avg;
                wb.Cells[row + initialRow + numberOfSubTest,
column + 1].Value = colorNumber;
                doc.Save();
            }
        }
```

Figure 4-7 Excel file generation code

4.4 User interface issues and limits

Even though maximum clique and coloring problems are a part of graph theory, which is complicated by itself, one of the purposes of the application is to have high discoverability and be understandable for regular user. In order to achieve this goal, there were made some front-end decisions and limitations.

When the user opens the web application page, there is a menu for choosing what type on analyse is needed. User can choose 4 types on testing:

1) Maximum clique algorithms testing with *DIMACS* graphs
2) Coloring algorithms testing with *DIMACS* graphs
3) Maximum clique algorithms testing with *RANDOM* graphs
4) Coloring algorithms testing with *RANDOM* graphs

In the first and second type of the analyze user should upload file with *DIMACS* graph. As far as file uploading concerned, there were limitations applied such as file extension and size.
After the testing the *DIMACS* graph compilation time, was set up the limit of the file size. Otherwise application uploads it more than hour and it causes web api timeout request.

```
function FileUploadComponent(http, fileservice) {
        this.http = http;
        this.fileservice = fileservice;
        this.errors = [];
        this.fileExt1 = "CLQ";
        this.maxFiles = 2;
        this.maxSize = 10; // 10MB
```

Figure 4-8 Limitations

User interface should give feedback to the client that the uploaded file is wrong. The Figure 4-9 shows how it is implemented.

```
// isValidFileExtension use to check file extention
    ColorFileuploadComponent.prototype.isValidFileExtension = function
(fileList) {
        // Make array of file extensions
        var extensions = (this.fileExt1.split(','))
            .map(function (x) { return x.toLocaleUpperCase().trim(); });
        for (var i = 0; i < fileList.length; i++) {
            // Get file extension
            var ext = fileList[i].name.toUpperCase().split('.').pop() ||
fileList[i].name;
            var exists = extensions.includes(ext);
            if (!exists) {
                this.errors.push("you can not upload file : " +
fileList[i].name);
            }
            this.isValidFileSize(fileList[i]);
        }
```

Figure 4-9 File extension check

Web application has got list of extensions and if the uploaded file extention is wrong, it throws error message. Talking about size, user also gets the error message, if the size limit exceeded.

```
// isValidFileSize use to check file size in MB
    ColorFileuploadComponent.prototype.isValidFileSize = function
(fileList) {
        var fileSizeinMB = fileList.size / (1024 * 1000);
        var size = Math.round(fileSizeinMB * 100) / 100; // convert upto 2
decimal place
        if (size > this.maxSize)
            this.errors.push("Error (File Size): " + fileList.name + ":
exceed file size limit of " + this.maxSize + "MB ( " + size + "MB )");
    };
```

Figure 4-10 File size check

Talking about discoverability of the user interface, it is easy to understand. User can choose the algorithms just by putting a tick in the checkbox.

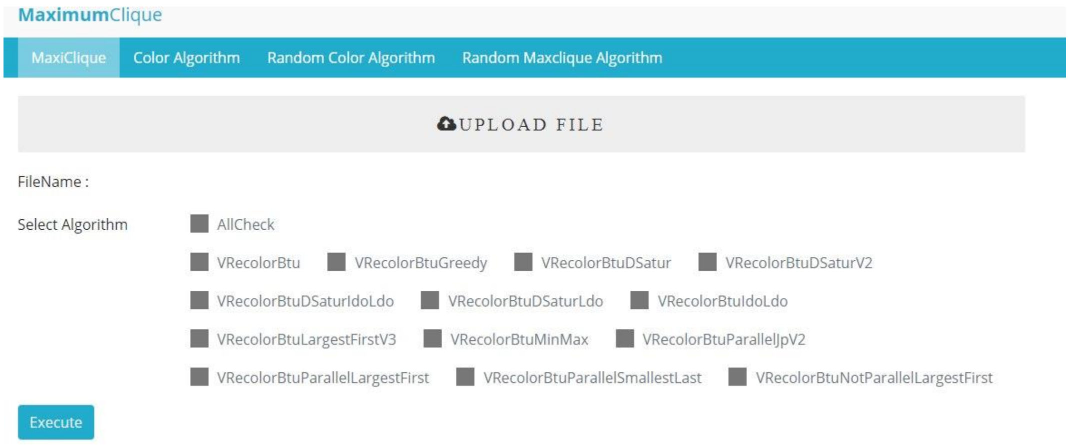


Figure 4-11 UI example

There is no *UI* conflicts with the functionality. The result is shown as a table with the names of the alforithms and charts.

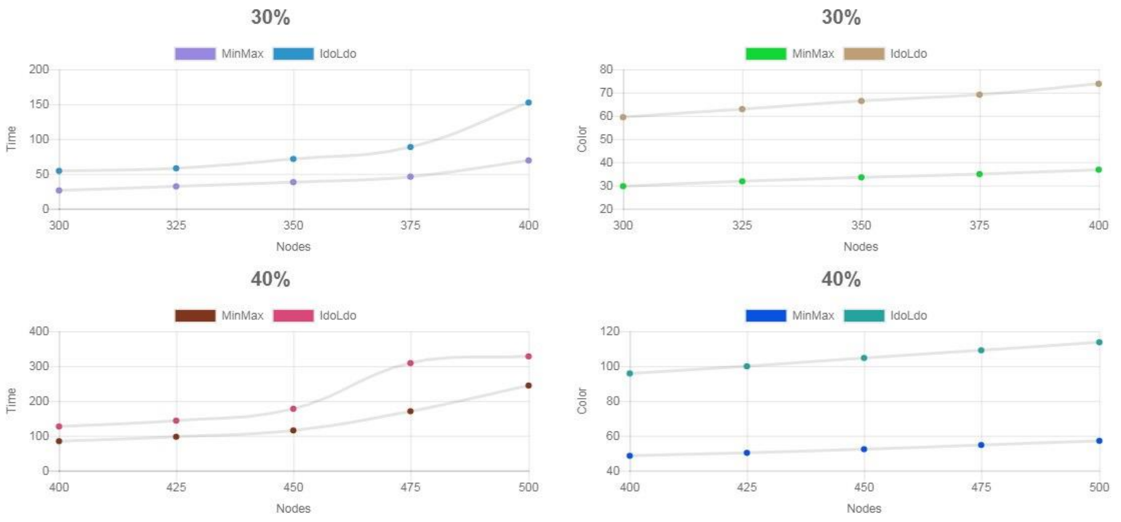
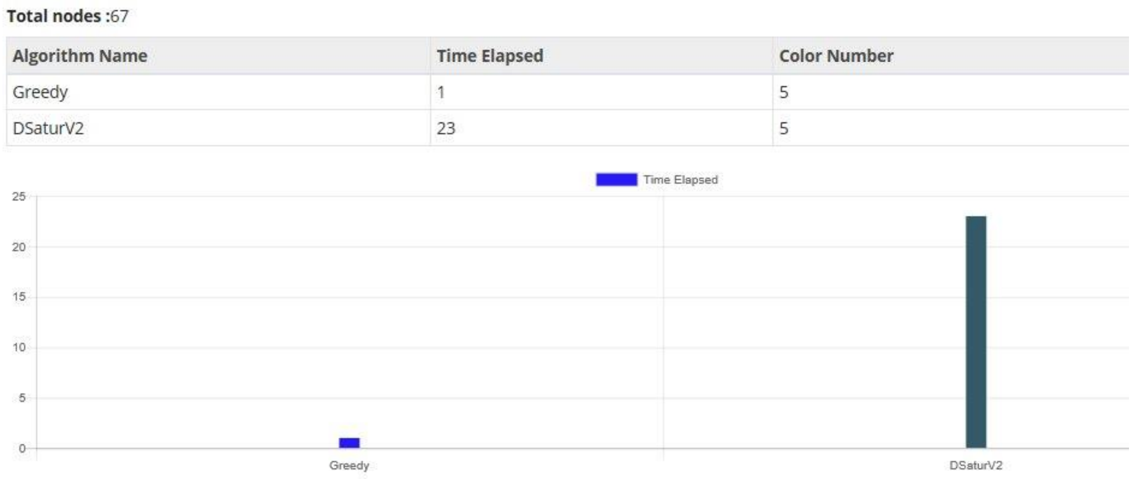


Figure 4-12 Charts example



Total nodes :67

Algorithm Name	Time Elapsed	Color Number
Greedy	1	5
DSaturV2	23	5

Figure 4-13 Results table example with coloring algorithms(Greedy and DsaturV2)

Other 2 types of analyze are directly connected with *RANDOM* graphs. In this part user should enter all needed parameters for graph generation and for the analyze. These parameters are density and number of vertices on each density. User chooses by himself, what test would be performed. In other words, application gives opportunity to put limits of test such as minimum and maximum density, because in the most cases, researchers are interested in testing specific densities. Talking about limitations, application gives opportunity to test maximum 5 algorithms, because otherwise it takes more than 50 minutes to get the result. Note: it was tested on author’s PC.

```
uploadData(event) {
this.elestyle = { 'display': 'block' };
this.errors = []; // Clear error
this.chartsCanvas = [];

if (this.mindensity == null) {
this.errors.push("Min density required");
}
```
Note: Gives error message, if the minimum density input is empty. The same control of the input in case of maximum density.

```
if (this.selectedAlgorithms.length < 1)
this.errors.push("Please select atleast one algorithm");
```

Note: Throws error message, if no algorithms chosen by the user.

```
if (this.selectedAlgorithms.length > 5)
this.errors.push("Please select algorithm between 1 to 5");
```

Note: Throws error message, if the user has chosen more than 5 algorithms. It is needed to avoid timeout problems.

```
for (var i = 0; i < this.minVertices.length; i++) {
if (this.minVertices[i] == '') {
this.errors.push("Min vertices can't be  null");
break;
}
```

Note: Throws error message, if the input of minimum number of vertices is not filled by the user. The same control of the input in case of maximum number of vertices.

```
if ((parseInt(this.minVertices[i])) >= (parseInt(this.maxVertices[i]))) {
this.errors.push("Min vertices must be less than max vertices ");
break;
}
}
```

Note: Throws error message, if the minimum number of vertices is bigger than maximum number.

Figure 4-14 Example of the limitations with Random graph

Because of necessity of all inputs fulfillment and avoiding exceptions, was created list of error massages. For example, if user forgets to choose any of algorithms, application throw notification: „Please select at least one algorithm“ and etc. Moreover, each input has a name, so that the input field will not cause misunderstanding from the user side.

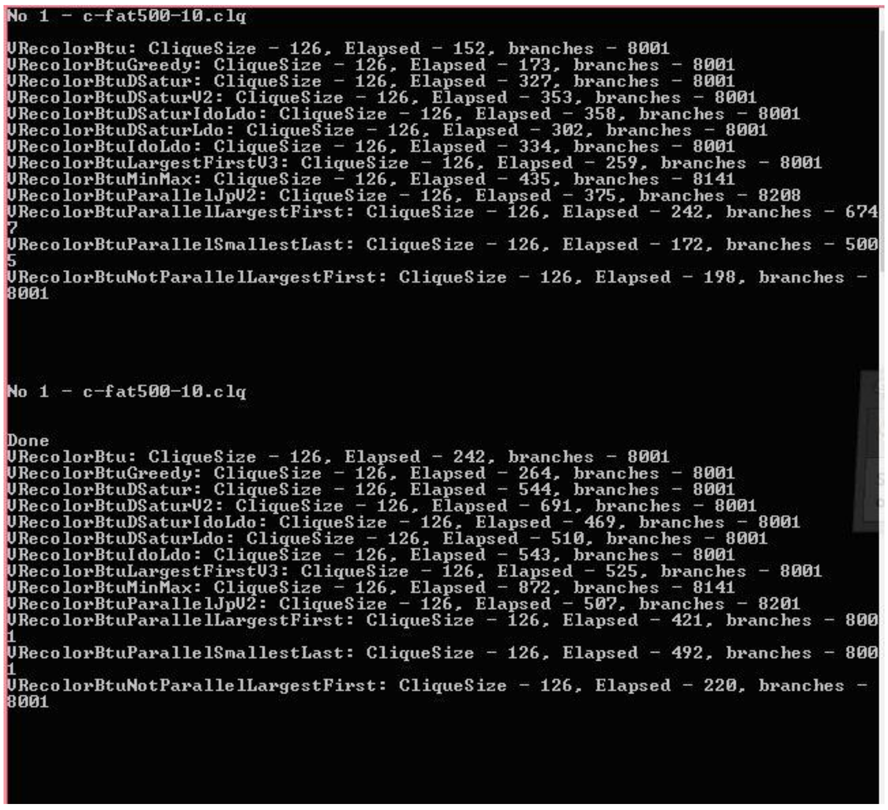
4.5 Performance control

During development process author tried several techniques of improving web application performance. The idea was to check if parallel processing of the algorithms will be faster than one by one. The web application works in different browsers such as: Chrome, Firefox or Microsoft Edge. The names of the function are presented on Figure 4-15.

```
// Nikita Golovin: function to start algorithm processing
public ResultViewModel StartNew(double? density)
        {
            ...
        }
        // Nikita Golovin: New function to be used parallel algorithm
processing
        public async Task<ResultViewModel> StartNewParallel(double?
density)
        {
      ...}
```

Figure 4-15 Sequential and parallel processing

After applying these two techniques, author got the results. The time consumed for processing parallel technique was more than one by one. Probably, this results can be applied only to the author’s PC with its technical characteristics. The results of the processing are demostrated on the Figure 4-16.



```
No 1 - c-fat500-10.clq

URecolorBtu: CliqueSize - 126, Elapsed - 152, branches - 8001
URecolorBtuGreedy: CliqueSize - 126, Elapsed - 173, branches - 8001
URecolorBtuDSatur: CliqueSize - 126, Elapsed - 327, branches - 8001
URecolorBtuDSaturV2: CliqueSize - 126, Elapsed - 353, branches - 8001
URecolorBtuDSaturIdoLdo: CliqueSize - 126, Elapsed - 358, branches - 8001
URecolorBtuDSaturLdo: CliqueSize - 126, Elapsed - 302, branches - 8001
URecolorBtuIdoLdo: CliqueSize - 126, Elapsed - 334, branches - 8001
URecolorBtuLargestFirstV3: CliqueSize - 126, Elapsed - 259, branches - 8001
URecolorBtuMinMax: CliqueSize - 126, Elapsed - 435, branches - 8141
URecolorBtuParallelJpV2: CliqueSize - 126, Elapsed - 375, branches - 8208
URecolorBtuParallelLargestFirst: CliqueSize - 126, Elapsed - 242, branches - 674
7
URecolorBtuParallelSmallestLast: CliqueSize - 126, Elapsed - 172, branches - 500
5
URecolorBtuNotParallelLargestFirst: CliqueSize - 126, Elapsed - 198, branches -
8001




No 1 - c-fat500-10.clq


Done
URecolorBtu: CliqueSize - 126, Elapsed - 242, branches - 8001
URecolorBtuGreedy: CliqueSize - 126, Elapsed - 264, branches - 8001
URecolorBtuDSatur: CliqueSize - 126, Elapsed - 544, branches - 8001
URecolorBtuDSaturV2: CliqueSize - 126, Elapsed - 691, branches - 8001
URecolorBtuDSaturIdoLdo: CliqueSize - 126, Elapsed - 469, branches - 8001
URecolorBtuDSaturLdo: CliqueSize - 126, Elapsed - 510, branches - 8001
URecolorBtuIdoLdo: CliqueSize - 126, Elapsed - 543, branches - 8001
URecolorBtuLargestFirstV3: CliqueSize - 126, Elapsed - 525, branches - 8001
URecolorBtuMinMax: CliqueSize - 126, Elapsed - 872, branches - 8141
URecolorBtuParallelJpV2: CliqueSize - 126, Elapsed - 507, branches - 8201
URecolorBtuParallelLargestFirst: CliqueSize - 126, Elapsed - 421, branches - 800
1
URecolorBtuParallelSmallestLast: CliqueSize - 126, Elapsed - 492, branches - 800
1
URecolorBtuNotParallelLargestFirst: CliqueSize - 126, Elapsed - 220, branches -
8001
```

Figure 4-16 Result of parallel and sequential processing

5 Conclusion

5.1 Summary

The main goal of this bachelor’s thesis was to create the application for exploring the performance of the maximum clique and coloring algorithms. In order to achieve this, were analyzed the requirements and based on them were chosen the most appropriate software development technologies and solutions. As a result of the work, author has created web application that helps the user to perform analysis of algorithms performance, compare performance of algorithms between each other and visualize results. The author studied algorithms logic in order to implement them in the project and adapt them for web api. Web application as a form of software does not depend on the technical features of the user device and its location, but the only condition is the availability of the Internet connection. The application interface is made as user-friendly as possible and logically structured.

The web application allows user to explore the performance of the maximum clique and coloring algorithms with *DIMACS* or *RANDOM* graphs. It is known that the performance of some algorithms for a specific density is the best, so that the user can generate the *RANDOM* graph as it is needed for his research.

Talking the visualization of the results, the web application gives the end-user opportunity to compare the performance and efficiency of the maximum click and coloring algorithms using graphs and tables. Additionally, the user can download the results in Excel file. Link for downloading is available after launching the web application.

The aim of the bachelor thesis was successfully fulfilled. Web application is a real software that can be used on huge variety of devices, and thanks to its structure, it can be improved in the future. Web application is a foundation for researchers, so that in the future it could be easily developed to the end-user standards.

5.2 Future studies

Despite the fact that we have a working web application, it can be improved. The web application will remain utility tool for the researchers, because *NP*-complete problems are still being explored. Despite the fact that we have reached all goals, the speed of the web application could be improved by applying of new technologies. In this aspect, such as development, the selected *MVC* web application model is a right decision. Besides the speed, functionality of the application could be also improved. For example, the end-user could upload his own algorithm or compare multiple algorithms simultaneously with multiple graphs.

Kokkuvõte

Käesolevas bakalaureuse töö põhieesmärgiks oli realiseerida rakendus suurima kliki ja värvimise algoritmide tööjõudluse kontrollimiseks. Selle saavutamiseks analüüsiti esmalt nõudeid ja nende järgi valiti sobivamaid tarkvara arendamistehnoloogiaid ja lahendusi. Oli valitud veebirakenduse vorm. Töökäigus oli loodud tarkvara, mis aitab kasutajale teostada algoritmide tööjõudluse analüüsi, võrrelda algoritmide tööjõudlust ja visualiseerida tulemusi. Veebirakenduse kasutamise võimalus ei sõltu kasutaja seadme tehnilistest omadustest ning tema asukohast, vaid ainuke tingimus on Interneti ühenduse saadavus. Rakenduse kasutajaliides on tehtud võimalikult arusaadavalt lõppkasutajale ning loogiliselt üles ehitatud.

Veebirakendus võimaldab uurida suurima kliki ja värvimise algoritmide tööjõudlust nii *DIMACS*, kui ka *RANDOM* graafidega. Kuna on teada, et mõnede algoritmide tööjõudlus on parim konkreetse tiheduse juhul, kasutaja saab ise genereerida *RANDOM* graafi nii, nagu vaja tema uurimistöö jaoks.

Tulemuste visualiseerimisest rääkides, veebirakendus annab lõppkasutajale võimalust võrrelde suurima kliki ja värvimise algoritmide tööjõudlust ja effektiivsust graafikute ning tabelite abil. Peale selle, kasutaja saab alla laadida Excel faili tulemustega. Link allalaadimiseks on kättesaadav pärast veebirakenduse töö käivitamist.

Bakalaureusetöö eesmärk sai täidetud. Veebirakendus on reaalselt töötav tarkvara, mida saab kasutada erinevates seadmetes, ning tänu selle struktuuri, seda saab tulevikus edasi areneda.

Tulevased uuringud

Vaatamata sellele, et meil on töötav ja lõplik veebirakendus, seda saaks täiendada.Kuna tänapäeval jääb aktuaalseks, veebirakendus oma korral jääb vajalikuks uurimisvahendiks. Vaatamata sellele, arendamise käigus olid saavutatud kõik eesmärgid, veebirakenduse kiirust saaks parandada uute tehnoloogiate rakendamisel. Selles aspektis nagu arendamine hästi aitab autoriga valitud MVC veebirakenduse mudel. Peale kiirust, saaks lisada veebirakendusele ka funktsionaalsust. Näiteks, et lõpp-kasutaja saaks üles laadida oma algoritmi või võrrelda samaaegselt mitu algoritmi mitmede graafidega.

References

[1] D. Kumlander, Some Practical Algorithms to Solve The Maximum Clique Problem, Tallinn, 2005.

[2] W. Hasenplaugh, T. Kaler, T. B. Schardl ja C. E. Leiserson, Ordering Heuristics for Parallel Graph Coloring, 2014.

[3] I. Petuhhov, Graaf, [WWW]. Available: http://www.cs.tlu.ee/~inga/alg_andm_09/graph_2008.pdf (21.05.2018).

[4] TypeScript. [WWW]. Available: https://www.typescriptlang.org/ (21.05.2018)

[5] J. R. Allwright, R. Bordawekar, P. D. Coddington, K. Dincer ja C. L. Martin, A Comparison of Parallel Graph Coloring Algorithms, 1995.

[6] Chart.js documentation. [WWW]. Available: http://www.chartjs.org/docs/latest/ (21.05.2018)

[7] Chart.js Tutorial. [WWW]. Available: https://coursetro.com/posts/code/126/Let's-build-an-Angular-5-Chart.js-App---Tutorial (21.05.2018)

[8] Bootstrap. [WWW]. Available: https://getbootstrap.com/ (21.05.2018)

[9] Official Angular Website. [WWW]. Available: https://angular.io/ (21.05.2018)

[10] Angular Tutorial. [WWW]. Available: https://mdbootstrap.com/angular/angular-tutorial/ (21.05.2018)

[11] M. Koit, Graafiteooria põhimõisted, internet based by Hilja Afanasjeva [WWW]. Available: https://www.teaduskool.ut.ee/sites/default/files/teaduskool/oppetoo/mat_gymn_graafiteooria.pdf (21.05.2018)

[12] A. Porošin, Reversed Search Maximum Clique Algorithm Based on Recoloring, Tallinn, 2015.

[13] Soma Saha, Gyan Baboo, Rajeev Kumar, An Efficient EA with Multipoint Guided Crossover for Bi-objective Graph Coloring Problem, 2011.

[14] Carraghan R, Pardalos PM An exact algorithm for the maximum clique problem. Op. Research Letters 9, 1990, pp 375-382.

[15] Östergård PRJ, A fast algorithm for the maximum clique problem, Discrete Applied Mathematics 120, 2002, pp 197-207.

[16] Niga Atta, Desktop application VS. Web applicaiton [WWW]. Available: https://www.avestagroup.net/DetailsEN.aspx?PostID=1006&CataType=5&CataID=1006 (21.05.2018)

[17] Web and Desktop application comparison [WWW]. Available: http://lamp-dev.ru/desktop-vs-web-applications-340.html (21.05.2018)

[18] DIMACS. [WWW]. Available: http://dimacs.rutgers.edu/archive/Challenges/ (21.05.2018)

[19] MVC model. [WWW]. Available: https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/overview/understanding-models-views-and-controllers-cs (21.05.2018)

[20] University of Cambridge, Turing machine [WWW]. Available: https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/turing-machine/one.html (21.05.2018)

[21] Chartrand G (1985) The Königsberg Bridge Problem: An Introduction to Eulerian Graphs, Introductory Graph Theory. New York: Dover 3(1), pp 51-60

[22] Appel, K. and Haken, W. "The Solution of the Four-Color Map Problem." Sci. Amer. 237, 108-121, 1977.

[23] Jetbrains Co, Webstorm. [WWW]. Available: https://www.jetbrains.com/webstorm/ (21.05.2018)

[24] Simple graph [WWW]. Available: https://www.tutorialspoint.com/graph_theory/types_of_graphs.htm (21.05.2018)