Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Spanning trees in graphs

Bachelor Thesis

2022
Terézia Strišovská

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Spanning trees in graphs

Bachelor Thesis

Study Programme:	Applied Informatics
Field of Study:	Applied Informatics
Department:	Department of Applied informatics
Supervisor:	doc. RNDr. Tatiana Jajcayová, PhD.
Consultant:	Mgr. Dominika Mihálová

Bratislava, 2022
Terézia Strišovská

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Terézia Strišovská
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Spanning trees in graphs
Kostry v grafoch

Anotácia: Cieľom práce je navrhnť a zbehnúť experimenty a určiť ohraničenia pre počet kostier v rôznych triedach grafov. Napríklad výsledok Nogu Alona tvrdí, že pre k-regulárne jednoduché grafy s k>0, s n vrcholmi existuje aspoň 2^(n\2) kostier.

Cieľ: Cieľom práce je navrhnť a zbehnúť experimenty a určiť ohraničenia pre počet kostier v rôznych triedach grafov.

Vedúci: doc. RNDr. Tatiana Jajcayová, PhD.
Konzultant: Mgr. Dominika Mihálová
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 01.06.2022

Dátum schválenia: 19.09.2022

doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Terézia Strišovská
Study programme:	Applied Computer Science (Single degree study, bachelor I. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Bachelor´s thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Spanning trees in graphs

Annotation: The aim of the theses is to run experiments and give bounds for number of spanning trees in variety of classes of graphs. For isntance a result of Noga Alon claims that for k-regular simple graphs with k>0, with n vertices there are at least 2^(n\2) spanning trees.

Aim: The aim of the theses is to run experiments and give bounds for number of spanning trees in different classes of graphs.

Supervisor:	doc. RNDr. Tatiana Jajcayová, PhD.
Consultant:	Mgr. Dominika Mihálová
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	prof. Ing. Igor Farkaš, Dr.
Assigned:	01.06.2022
Approved:	19.09.2022

doc. RNDr. Damas Gruska, PhD.
Guarantor of Study Programme

.....
Student

.....
Supervisor

Acknowledgments: Tu môžete poďakovať školiteľovi, prípadne ďalším osobám, ktoré vám s prácou nejako pomohli, poradili, poskytli dáta a podobne.

Abstrakt

Slovenský abstrakt v rozsahu 100-500 slov, jeden odstavec. Abstrakt stručne sumarizuje výsledky práce. Mal by byť pochopiteľný pre bežného informatika. Nemal by teda využívať skratky, termíny alebo označenie zavedené v práci, okrem tých, ktoré sú všeobecne známe.

Kľúčové slová: jedno, druhé, tretie (prípadne štvrté, piate)

Abstract

Abstract in the English language (translation of the abstract in the Slovak language).

Keywords:

Contents

List of Figures

Introduction

Cieľom tejto práce je poskytnúť študentom posledného ročníka bakalárskeho štúdia informatiky kostru práce v systéme LaTeX a ukážku užitočných príkazov, ktoré pri písaní práce môžu potrebovať. Začneme stručnou charakteristikou úvodu práce podľa smernice o záverečných prácach [**?**], ktorú uvádzame ako doslovný citát.

> Úvod je prvou komplexnou informáciou o práci, jej cieli, obsahu a štruktúre. Úvod sa vzťahuje na spracovanú tému konkrétne, obsahuje stručný a výstižný opis problematiky, charakterizuje stav poznania alebo praxe v oblasti, ktorá je predmetom školského diela a oboznamuje s významom, cieľmi a zámermi školského diela. Autor v úvode zdôrazňuje, prečo je práca dôležitá a prečo sa rozhodol spracovať danú tému. Úvod ako názov kapitoly sa nečísluje a jeho rozsah je spravidla 1 až 2 strany.

V nasledujúcej kapitole nájdete ukážku členenia kapitoly na menšie časti a v kapitole **??** nájdete príkazy na prácu s tabuľkami, obrázkami a matematickými výrazmi. V kapitole **??** uvádzame klasický text Lorem Ipsum a na koniec sa budeme venovať záležitostiam záveru bakalárskej práce.

Chapter 1

Preliminaries

This chapter will be devoted to explaining terms and notations from graph theory, introducing concepts, algorithms and previous research in the field of the topic of our thesis.

1.1 Basic terminology and definitions

In this section, we will introduce basic definitions from graph theory that are relevant to our work. Each subsection is dedicated to one or two related terms, providing their definition, further explanation and in some cases illustrated examples.

1.1.1 Simple graph

Graph is a structure, that is present around us in many forms, even though we may not realize it's a graph. Transportation network, family trees and electircity network are just a few examples. We will look at this term from a formal side, defining one specific type of graphs - simple graphs.

Definition. A simple graph is an ordered pair of sets $G = (V, E)$, where V is a non-empty set of vertices (or nodes) of G, and E, the set of edges of G, is a set of two-element pairs(2-combinations) of vertices. Thus, each edge of G can be expressed as $\{u, v\}$, where u and v are distinct vertices, i.e., $u, v \in V, u \neq v$. The vertices u and v determining an edge $\{u, v\}$ are called edpoints of the edge. The edge $\{u, v\}$ is said to join u and v, and the edge is said to be incident to either of its endpoints. Any two vertices in G that are joined bz an edge are said to be adjacent, and are called neighbors. A vertex with no neighbors is called isolated. [3, p. 497]

In other words, simple graph is a graph without loops(edges with equal endpoints) and multiple edges(edges with the same pair of endpoints).

In our work, we will consider only undirected simple graphs, which means that edges are unordered pairs of vertices and thus each edge $\{u, v\}$ can be traversed in both directions - from u to v and from v to u.

1.1.2 Regular graph

Definition. If v is a vertex of a graph G, then the degree of v, denoted $deg(v)$ (or $deg_G(V)$), if we wish to emphasize the dependence on G), is the number of edges incident to v, with any self-loops counted twice. A simple graph in which all vertices have the same degree k is called a regular graph or, more precisely, a k-regular graph. [3, p. 499]

This means that each vertex in a k-regular graph has exactly k neighbors. k may range from 0 to $|V(G)| - 1$. For $k = 0$, the edge set is empty, 1-regular graph consists of disconnected edges and 2-regular grahs contains one cycle or more disjoint cycles. For $k \geq 2$, any k-regular graph contains one or more cycle. Every simple k-regular graph on n vertices has exactly $\frac{n \cdot k}{2}$ edges. Since the sum of degrees of vertices in a simple undirected graphs must be an even number(every edge is counted twice - once in both possible directions), k-regular graphs where k is odd exist only for even number of vertices.

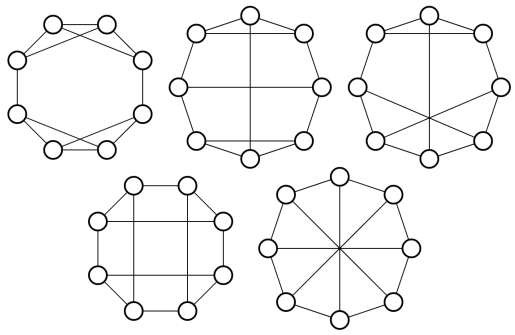

Figure 1.1: 3-regular graphs on 8 vertices

1.1.3 Paths and cycles

Definition. Suppose that $G = (V, E)$ is a graph, and $v, w \in V$ are a pair of vertices. A path in G from v to w is an alternating sequence of vertices and edges: $P =< v_0, e_1, v_1, e_2, v_2, ..., v_{k-1}, e_k, v_k >$, such that the endpoints of edge e_i are the vertices $\{v_{i-1}, v_i\}$, for $1 \leq i \leq k$, $v_0 = v$, and $v_k = w$. We say that path P passes through the vertices $v_0, v_1, v_2, ..., v_{k-1}, v_k$, and traverses the edges $e_1, e_2, ..., e_k$, and that the path has length k, since it traverses k edges. [3, p. 540]

There may exist several different paths from v to w, as we can see in figure 1.2, where there are 4 different paths from 2 to 4 and those are: $2, \{2, 0\}, 0, \{0, 1\}, 1, \{1, 4\}, 4$; $2, \{2, 0\}, 0, \{0, 1\}, 1, \{1, 3\}, 3, \{3, 4\}, 4$; $2, \{2, 3\}, 3, \{3, 4\}, 4$ and $2, \{2, 3\}, 3, \{3, 1\}, 1, \{1, 4\}, 4$.

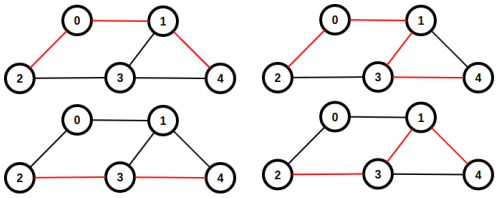

Figure 1.2: Paths in a graph

Definition. A cycle of a graph G, also called a circuit if the first vertex is not specified, is a subset of the edge set of G that forms a path such that the first node of the path corresponds to the last. A graph containing no cycles of any length is known as an acyclic graph. [9]

If we see a cycle of length n as a path that begins and ends in the same vertex and no other vertices are repeated, it is not important in which vertex it starts, since it could start in any of its n vertices and it would still represent the same cycle.

A graph where each vertex is of degree at least 2, this graph must contain a cycle. ucenbica 4

1.1.4 Connected graph

Definition. A graph is connected if it has a u, v-path for each pair u, $v \in V(G)$. [1, p. 5]

This means that each vertex of G is reachable from any of its vertices. The number of components(maximal connected subgraphs) in connected graph with $|V(G)| \geq 1$ is exactly 1, and this component contains all the vertices from its vertex set.

1.1.5 Subgraph and spanning subgraph

Definition. A subgraph of a graph G is a graph H such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$; we write $H \subseteq G$ and say that "G contains H". [1, p. 3]

Definition. A spanning subgraph of G is a subgraph with vertex set $V(G)$. [1, p. 51]

Basically, we can create a subgraph of G by deleting any of its vertices and their incident edges and any of remaining edges. However, in case of spanning subgraph, we may delete only edges, as vertex set must be preserved.

We can see this diffenece in figure 1.3. Both G_2 and G_3 are subgraphs of G_1, but only G_3 is its spanning subgraphs, since vertex set of G_2 is $V(G_1) \setminus \{4\}$.

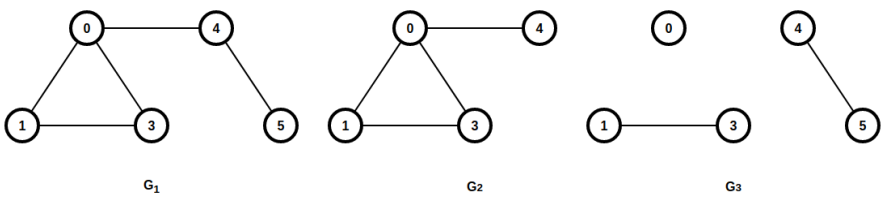

Figure 1.3: Examples of subgraphs and spanning subgraph

1.1.6 Tree

There exist several characterizations of trees and each of them is equivalent. This means that in case we want to prove that a graph is a tree, we can choose any of the characterizations and verify that the graph satisfies it.

The following are examples of characteristics of a tree G on n vertices.

a) G is connected and has no cycles

b) G is connected and has $n-1$ edges

c) G has $n-1$ edges and no cycles

d) for $u, v \in V(G)$, G has exactly one u,v-path [1, p. 52]

Since there is only one path between each pair of vertices, if we delete any edge from $E(G)$, the graph becomes disconnected. This means that every edge of a tree is a bridge. [3, p. 573]

When speaking of trees, a vertex of degree 1 is called a leaf, the rest of the vertices are internal vertices. [3, p. 572]

Another important properties of trees are that $|E(G)| = |V(G)| - 1$ and that by adding a new edge to G between two of it vertices which weren't adjacent originally, we create a new graph with exactly one cycle.

1.1.7 Spanning tree

Now we are getting to the core topic of our thesis, spanning trees, which are special cases of trees.

Definition. A spanning tree is a spanning subgraph that is a tree. [1, p. 51]

So when we have a graph G, any of its spanning trees T is a graph such that $V(T) = V(G)$ and T is a maximum posiible tree in G, which means that adding an edge to $E(T)$ renders T cyclic.

From the earlier definitions, we can see that a diconnected graph can't have any spanning tree and therefore we will focus on connected graphs in our work.

In figure 1.4, we can see a graph G and all of its spanning trees.

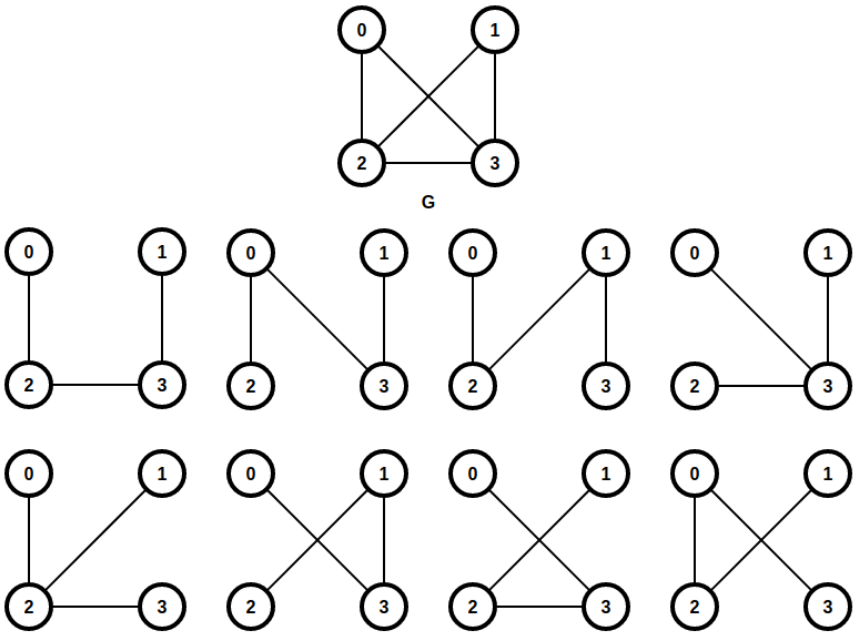

Figure 1.4: All spanning trees of graph G

Also spanning trees have many applications, for example in network, where we want to cover and connect all nodes, but also to minimize the cost of connections between those nodes. A special case of spanning tree is minimum spanning tree, in weighted graphs it's a spanning trees with minimum possible weight of the contained edges. Minimum spanning trees can be used in approximating solution of complex mathematical problems, for example the Traveling Salesman Problem.

1.1.8 Graph isomorphism

Definition. An isomorphism from G to H is a bijection $f : V(G) \to V(H)$ such that $uv \in E(G)$ if and only if $f(u)f(v) \in E(H)$. As a result, if two graphs are isomorphic, they must have the same degree sequence. However, the same degree sequence doesn't implies that two graphs have to be isomorphic. We say "G is isomorphic to H", written $G \cong H$, if there is an isomorphism from G to H. [1, p. 7]

Basically, two graphs, G and H, are isomorphic if they have the same structure, they vertex and edge sets might be different. Isomorphism is then a mapping from vertex set of V to vertex set of H, which renames the vertices of V, creating G' such that it has the same vertex and edge set as H.

Definition. An isomorphism from a graph G to itself is called an automorphism of G. [10, p. 4]

Figure 1.5 shows two isomorphic graphs G_1 and G_2, where one of the possible isomorphisms betwen them is $\{(0,6),(1,8),(2,7),(3,5),(4,9)\}$.

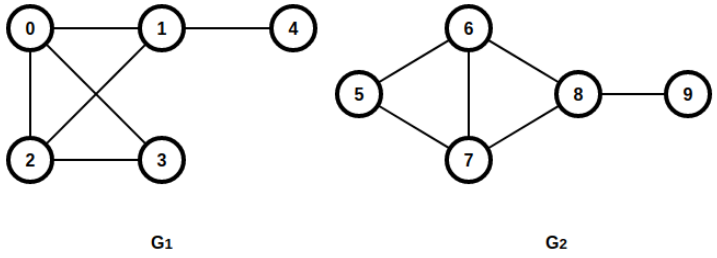

Figure 1.5: Isomorphic graphs

1.2 Graph representation

There exist several ways to represent a graph, and each way may be more suitable for a different situation, depending on the operations to be performed on graphs, the information about the graphs we want to preserve or memory criteria. We will list a few methods, some of which are also used in our work. Suppose we have a graph G with vertex set $V = \{1, 2, 3...n\}$ and edge set E.

First is adjacency matrix, which is a $|V| \times |V|$ matrix where element in i-th row and j-th column holds information about whether vertex i is adjacent to vertex j. This means that in case of undirected graphs, where $ij \in E \rightarrow ji \in E$, the matix is diagonally symmetric. When working with unweighted graph, the value of the element would be set to 1 if $ij \in E$, otherwise to 0. For weighted graphs, it represents the weight of edge from i to j.

Edge list is an array with $|E|$ pairs, each for one edge from E. This method is memory efficient in case of sparse graphs - graphs with only a few edges, however, we may need to store an extra record for the number of vertices in the graph, since there might be isolated vertices and edge sets holds no information about those.

Another representation is adjacency list, which is a collection of lists, one for each vertex of V. List for a given vertex v contains all vertices adjacent to v.

1.3 Tree isomorphism

Generally speaking, the task of determining whether two graphs are isomorphic is non-trivial. There are some cases, when we can exclude the existence of isomorphism easily, for example when the two graphs are not on the same number of vertices or when they have different degree sequence. The problem doesn't have a broadly appliable known solution with polynomial time, neither is it known to be a NP-complete problem, therefore it may be of intermediate complexity. [11, p. 3304] For some classes of graphs, however, there exist solutions with polynomial complexity.

Algorithms for tree isomorphism work with rooted trees. Although the graphs in our thesis are undirected, simple trees can by rooted by finding central vertex such that the longest path to leaf is minimum over all vertices in the graph.

We will be using AHU algorithm, described more in detail in chapter **??**, which solves the problem in linear time proportional to the number of vertices in the trees. The difinition of tree isomorphism it uses if the following:

Definition. Two trees are said to be isomorphic if we can map one tree into the other by permuting the order of the sons of vertices [5, p. 84]

Each vertex is then assigned a tuple representing the structure of its subtree.

1.4 Spanning tree enumeration

Finding an arbitrary spanning tree of a graph, or even minimum spanning tree in edge-weighted graphs, is a well known problem with several possible solutions that work in polynomial time. The main factor affecting computation time is the number of edges and vertices in a graph. There are, however, cases when we would like to generate all spanning trees for a given graph. The number of spanning trees in a graph might be exponential to its size, so the time taken for generating all spanning trees is no longer polynoial.

There are a few different approaches to spanning tree enumeration, varying also in efficiency. [13] One of them would be generating combinations of graph's edges of size $n - 1$, candidates for spanning trees, and testing them for acyclicity. Another method starts with initial spanning tree, constructed for example by breadth or depth first search. We can then replace one edge of the current spanning tree by an edge outside of it, making sure no cycle is formed. The resulting graph is a spanning tree as well. We need to employ some policy to ensure no duplicates are generated, but on the contrary with the previous one, this method produces only spanning trees.

For the purposes of our work, we will be implementing algorithm from the second described gropup, one that was proposed by S. Kapoor and H. Ramesh. [14]

1.5 Matrix Tree Theorem

Although spanning tree enumeration can serve as a method of counting spanning trees, when we are interested only in the number of spanning trees their listing and time taken for their enumeration becomes redundant. In these cases, we can use Matrix Tree Theorem, also know as Kirchhoff's theorem, named after Gustav Kirchohoff, which computes this number using determinant of a matrix Q of size $n \times n$ for a graph G

with vertex set $\{v_1, v_2, ..., v_n\}$.It works for loopless graphs, so graphs in our work satisfy this condition. Multiple edges are allowed, but since we are working with simple graphs, we can suppose that the number of edges $\{u, v\}$ for any vetrex u and v in the graph is always 0 or 1.

Then the elements of the matrix Q are defined as

$$Q_{i,j} := \begin{cases} deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \\ 0 & otherwise \end{cases}$$

Basically, we take a $n \times n$ matrix with vertex degrees on the diagonal, other elements are set to 0, and subtract the adjacency matrix of G.

After this step, we delete any row s and column t of Q, obtaining Q^*. Once we have computed determinant of Q^*, $detQ^*$, the number of spanning trees in G equals $(-1)^{s+t} detQ^*$. [1, p. 67]

1.6 Graph classes and their bounds for number of spanning trees

While the question of number of spanning trees in a particular graph is straightforward - it can be easily answered with use of above described Matrix Tree Theorem, there is another question and that is generalisation of number of spanning trees for a specific class of graphs. In this case, we may not always be searching for one concrete number, but rather for example for a relationship between the number of vertices in a graph and its number of spanning trees, which we could use to determine the number after being given only the number of vertices. Even more general approach would be to estimate upper or lower bounds for the number of spanning trees in a particular class in terms of number of vertices.

We will go through some common classes of graphs and discuss the available information about their number of spanning trees.

1.6.1 Trees

Since any tree T is an acyclic connected graph, by removing any of its edges, it will become disconnected. This means that is has only one spanning subgraph, which is T itself. We already stated, that T is a tree, combined with te fact that T is its own spanning subgraph(this applies to all the graphs, not only trees) it meets the definition of a spanning tree.

Therefore, T has exactly one spanning tree, T, and one isomorphism class containing solely T.

1.6.2 Cycles

C_n are connceted graphs with all of its vertices in a single cycle of a length n. If we remove any of its edges, we will get acyclic subgraph G with $|V(G)| = n-1$ and $n-1$ edges, which is by definintion a tree. The resulting graph will remain connected, with the same vertex set as C_n, meaning that it is spanning subgraph as well. This makes G a spanning tree of C_n.

C_n has n edges, so the number of its different spanning trees is n, each one of them is obrained by removing a different edge from the edge set of C_n.

However, the number of isomorphism classes of spanning trees in C_n is 1 - all of them are path graphs on n vertices.

1.6.3 Complete graphs

Complete graphs, K_n, have maximum possible number of edges in a graph on n vertices, exactly $\frac{n\cdot(n-1)}{2}$. Each vertex $v \in V$ is adjacent to all the vertices $u \in V(K_n) \setminus \{v\}$, so while looking for a spanning trees of K_n, we can instead see this problem as constructing all possible labeled trees on n vertices. These trees cover all the vertices in vertex set of K_n whihch makes them spanning trees of K_n.

The number of labeled trees on n vertices is defined by Cayley's formula, n^{n-2}.

In 1918, Prüfer found proof for this theorem, using function that would assign each tree a unique code, a sequence of length $n-2$ with entries from $[n]$, which is a set of natural numbers $\{1, ..., n\}$, where $n \in N$. There is a bijection between the set of trees on n vertices and the set of above mentioned sequences, hence both sets have the same cardinality, n^{n-2}.

$f(T)$, which computes Prüfer sequence for a labeled tree T, is defined iteratively. In each step, we delete leaf with the smallest label and add label of its only neighbour to the end of the sequence. This way, we perform $n-2$ iterations, producing a sequence of length $n-2$ and leaving one edge.

Inverse function to f is a function that produces a tree T from each sequence, where $f(T) = s$. Beginning with a forest with all the vertices from $[n]$ and empty set of finished vertices, in $i-$th iteration, edge xy is added and vertex y is marked finished. x is the label of a vertex in i-th position of s and y is the smallest label not yet included in finished vertices and not appearing in later positions of s. After $n-2$ steps, we remain with two unfinished vertices, which are then joined by an edge. [1, p. 63]

Similarily, the number of non-isomorphic spanning trees of K_n is the number of non-isomorphic trees on n vertices. Every complete graph has a spanning tree that is a path, a star(even though in $K_n, n \leq 3$, there is only one spanning tree that is a path and at the same time a star) and the number of isomorphism clases grows for higher values of n. The sequence of these values for $n \geq 1$ is $1, 1, 1, 2, 3, 6, 11, 23, 47, 106, 235, 551, 1301, 3159, ...$,

it can be found to a greater extent in The On-line Encyclopedia of Integer Sequences. [12]

Here is an example for K_5, which has three isomorphism classes. We are showing one spanning tree from each class. After we count cardinality of each isomorphism class, we reach number $125 = 5^{5-2}$, which respresents the number of all different spanning trees of K_5.

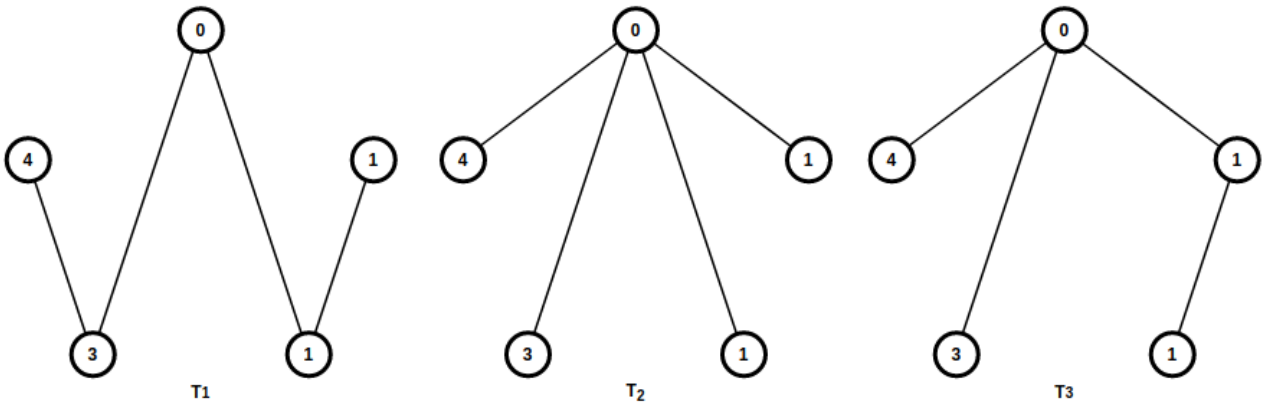

Figure 1.6: Representants of each isomorphism class of K_5's spanning trees

1.6.4 Regular graphs

Regular graphs don't necessary have to be connected, but for the purposes of determining the bounds for number of spanning trees, we will take into account only those with one component. Any disconnected regular graph has no spanning trees, which is information that wouldn't help us specify the relationship between the number of vertices and spanning trees.

Since the papers our theseis is based on suggest bounds only for k-regular graphs where $k \geq 3$, we will first separatedly discuss 0, 1 and 2-regular graphs. For illustration, figure 1.7 shows all possible unlabeled k-regular graphs for $k = 0, 1, 2$ with $|V(G)| \geq 5$ which are connected.

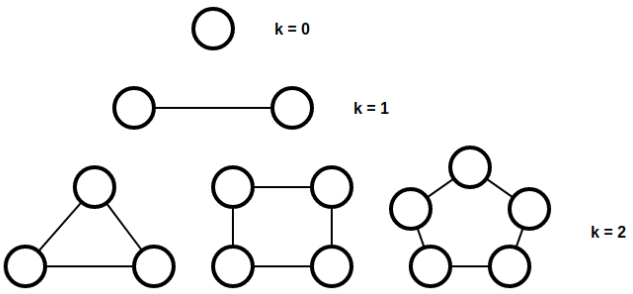

Figure 1.7: All connected unlabeled 0, 1 and 2-regular graphs on up to 5 vertrices

$k = 0$ 0-regular graphs contain only isolated vertices and therefore their number of spanning trees is always 0 once their vertex set consists of more than one vertex. And although 0-regular graph on 1 vertex has empy edge set, it is a tree by definition, which means that it is its own spanning tree.

$k = 1$ There is exactly one unlabelled connected 1-regular graph - two vertices connected by a single edge. We can see, that it satisfies one of the earlier mentioned characteristics of a tree. Therefore, it has exactly one spanning tree.

$k = 2$ As we stated in the information about 2-regular graphs, they consist of one or more disjoint cycles. In the cases when the number of disjoint cycles is greater than one, the graph is disconnected, so we will focus on those where only one cycle on n vertices is present. Again, this case case be answered with one of our previously discussed classes, cycle graphs. Then number of different spanning trees of any 2-regular connected graph is hence n.

$k \geq 3$ In 1983, Brendan McKay's article Spanning Trees in Regular Graphs was published. In the introduction, he stated a theorem setting upper bound for k-regular trees on n vertices where $k \geq 3$. According to the theorem, the number of spanning trees of such graph is at most $\frac{(\frac{n \cdot k}{n-1})^{n-1}}{n}$ [8, p. 149]. For 3-regular graphs, this means that their number of spanning trees can't be higher than 16, 100, 696, 5080, 38443... for $n = 4, 6, 8, 10, 12...$

Later, in 1990, Noga Alon went further into this problematics in his The Number of Spanning Trees in Regular Graphs [6] and proposed different set of bounds. One of the results presented there is that the number of spannng trees of the currently discussed k-regular graphs is always greater than $2^{\frac{n}{2}}$. Another observation introduced is that this number isn't greater than $(\frac{1}{n-1}) \cdot k^n$. McKay's theorem, however, presents slightly narrower and thus better bounds.

Bibliography

[1] WEST D. B. Introduction to Graph Theory. Prentice-Hall, Inc. 1996. 0-13-227828-6.

[2] CORMEN T. H., LEISERSON C. E., RIVEST R. L., STEIN C., Introduction to Algorithms, Third Edition. The MIT Press. 2009. 978-0-262-03384-8.

[3] STANOYEVITCH A., Discrete Structures with Contemporary Applications. Chapman and Hall/CRC Press. 2011. 978-1-4398-1768-1.

[4] MERINGER M., Fast Generation of Regular Graphs and Construction of Cages. Journal of Graph Theory 30, 137-146, 1999.

[5] AHO A., HOPCROFT J. and ULLMAN J., The Design and Analysis of Computer Algorithms. Addison-Wesley Publishing Co., Reading, MA, 1974, pp. 84-85.

[6] ALON N., The Number of Spanning Trees in Regular Graphs. Random Struct. Algorithms 1(2). 1990. 175-182.

[7] The On-line Encyclopedia of Integer Sequences, 2021, Connected regular graphs (with girth at least 3), From https://oeis.org/wiki/User:Jason_Kimberley/A068934/ 29.3.2023

[8] McKay B., Spanning Trees in Regular Graphs. Europ. J. Combinatorics (1983) 4. 1983. 149-160.

[9] Weisstein E. W., Graph Cycle. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/GraphCycle.html 24.3.2023

[10] GODSIL C., ROYLE G., Algebraic Graph Theory. Springer Science+Business Media, LLC, 2001.

[11] BABAI L., Group, graphs, algorithms: the Graph Isomorphism Problem. Proceedings of the International Congress of Mathematicians (ICM 2018), Rio de Janeiro, Vol. 3 (3303-3320). 2018. 3303-3320.

[12] The On-line Encyclopedia of Integer Sequences, A000055 Number of trees with n unlabeled nodes., From https://oeis.org/A000055 25.3.2023

[13] CHAKRABORTY M., CHOWDHURY S., CHAKRABORTY J. et al., Algorithms for generating all possible spanning trees of a simple undirected connected graph: an extensive review. Complex Intell. Syst. 5, 265–281 (2019). 2019

[14] KAPOOR S., RAMESH H., Algorithms for enumerating all spanning trees of undirected and weighted graphs. SIAM J Comput 24(2):247–265. 1995