Celina Strasser

Intersection of longest paths in connected graphs

BACHELORARBEIT

STUDIUM
Technische Mathematik

Alpen-Adria-Universität Klagenfurt
Fakultät für Technische Wissenschaften

ADVISOR
Dipl.-Ing. Dr. Benjamin Hackl
Institut für Mathematik
Alpen-Adria-Universität Klagenfurt

Klagenfurt, 5. Juli 2021

Abstract

This Bachelor's thesis is about the intersection of all longest paths in a connected graph. First the idea of longest paths and their intersection is introduced and a motivating example is given. Then the graph classes, whose longest-path-intersection is guaranteed to be nonempty, and additionally the shape of the intersection of longest paths in trees is considered more closely. Finally, SageMath implementations of methods for identifying and visualizing the intersection of all longest paths in connected graphs are discussed.

Contents

1 Introduction

1.1 Motivating Example - A sightseeing tour in Klagenfurt

There are three different tourist companies in Klagenfurt and each of them offers a longest possible sight tour. They advertise them with, "We show you the most possible sights, without showing you any sight twice." The three tourists Michaela, Benjamin and Angelika, who each took a different tour this day, meet for a drink at "Uniwirt". They share their experiences of the day and start to realize that they saw some pretty different sights. Benjamin starts to wonder, "Is there a sight that was visited by all of us?" After comparing their maps they find out, that everyone visited the "Lindwurm". Had their routes been positioned in such a way that there was no common visited sight, they would have a part of the answer this Bachelor's thesis is concerned with.

It is unkown up to now, if Michaela, Benjamin and Angelika necessarily saw a common sight. If this question is rephrased mathematically it would be "Do three longest paths in a connected graph have a common vertex?"

Before going deeper into this question we will recall some basic definitions from graph theory.

1.2 Definitions

▶ Definition 1. A **graph** is an ordered pair $G = (V, E)$, where V is a set of **vertices** (also called nodes or points) and E is a set of edges, which are unordered pairs of vertices, defined by $E \subseteq \{\{x, y\} \mid x, y \in V \text{ and } x \neq y\}$. ◀

An example would be the well-known **Petersen graph**, where the set of vertices is $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and the set of edges for the graph is $E = \{\{1,3\}, \{1,4\}, \{1,6\}, \{2,4\}, \{2,5\}, \{2,7\}, \{3,5\}, \{3,8\}, \{4,9\}, \{5,10\}, \{6,7\}, \{6,10\}, \{7,8\}, \{8,9\}, \{9,10\}\}$. Graphs are visually represented with nodes for the vertices and lines between these nodes for the edges between two vertices. In our case the nodes in the figure are labeled, this is not always the case, only if we are interested in individual vertices.

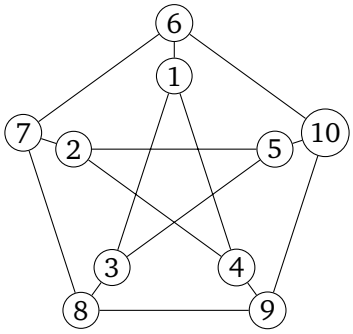


Figure 1.1: Petersen graph

► Definition 2. A **walk** in a graph is a finite sequence of alternating incident vertices and edges. An edge $e \in E$ and a vertice $v \in V$ are **incident** if $v \in e$. A walk where all vertices are different is called a **path**. A **longest path** in a graph is a path of maximum length. ◄

As an example we can take a look at two of the longest paths of the Petersen graph. On the left side of Figure 1.2 the red path P_1 can be written as $P_1 = (1, \{1,3\}, 3, \{3,8\}, 8, \{8,7\}, 7, \{7,6\}, 6, \{6,10\}, 10, \{10,9\}, 9, \{9,4\}, 4, \{4,2\}, 2, \{2,5\}, 5)$ whereas the blue path P_2 on the right side can be portrayed as $P_2 = (1, \{1,4\}, 4, \{4,9\}, 9, \{9,10\}, 10, \{10,6\}, 6, \{6,7\}, 7, \{7,8\}, 8, \{8,3\}, 3, \{3,5\}, 5, \{5,2\}, 2)$. Both have length 9.

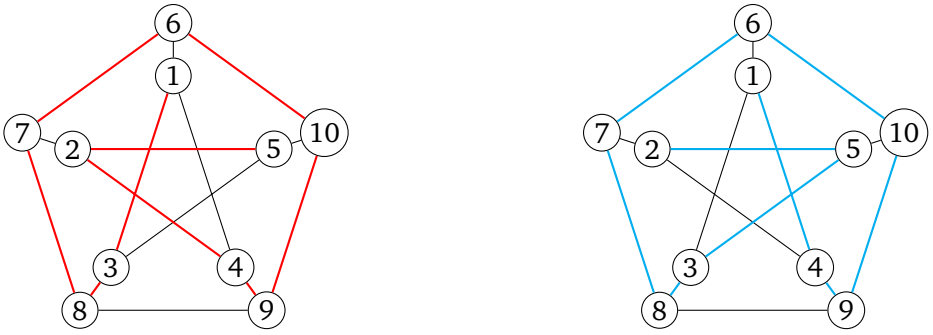


Figure 1.2: Two longest paths in the Petersen graph

Remark. A finite graph (i.e., a graph whose vertex set is finite) always has a longest path.

1.3 Common vertices in longest paths

Before considering the intersection of all longest paths we are going to start small and begin with the intersection of two longest paths.

Theorem 3. In a connected graph, any two longest paths have at least one vertex in common.

Proof. Let G be a connected graph and P, Q two longest paths of length k. Let $q \in V(Q)$ and $p \in V(P)$ such that there exists a path from p to q, which is assured, as G is connected. The paths P and Q can be seperated into two subpaths P_1 and P_2 as well as Q_1 and Q_2, see Figure 1.3. It follows that one subpath in every path has length $\geq \frac{k}{2}$. Without loss of generality let $|Q_1| \geq \frac{k}{2}$ and $|P_2| \geq \frac{k}{2}$. Let a_1 be the first vertex of the path Q_1, which ends with vertex q and b_{k+1} the last vertex of the path P_2, where this path starts with p. It is now possible to create a path M which starts in a_1, goes to q, then to p and ends in b_{k+1}. It holds that length$(M) \geq \frac{k}{2} + 1 + \frac{k}{2} = k + 1 > k$ which is a contradiction to P and Q being longest paths. □

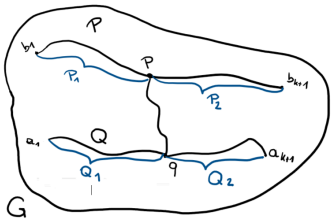


Figure 1.3: Sketch of the proof

The question "Does every connected graph have a common vertex for all its longest paths?" was first posed by Tibor Gallai in 1966. Since 1969 we know that the answer to this question is negative, as Walther [7] found a counterexample on 25 vertices. Later Walther [8] and Zamfirescu [9] independently also exhibited a counterexample on 12 vertices, see Figure 1.4.

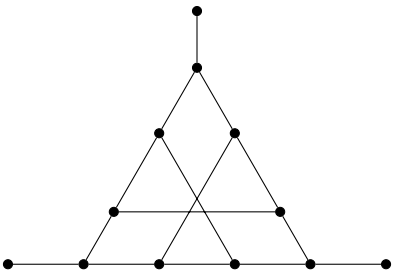


Figure 1.4: Graph of Walther and Zamfirescu

Proposition 4. None of the vertices in Figure 1.4 is contained in all of the graphs longest paths.

Proof. First, let us consider the paths in Figure 1.5.

Figure 1.5: Two longest paths in the counterexample graph

As we can see there are always two vertices which are not included in the path. Due to symmetric properties of the graph we can see that we are always able to find a longest path, so that a specific vertex is not part of this path. Therefore the intersection of all longest paths is empty. □

A first example with a nonempty intersection would be the Petersen graph. If we look back at the two longest paths in Figure 1.2, we see that the length of these are 9, which means, that they contain all vertices of the graph. Therefore all longest paths contain all 10 vertices of the graph and it follows that the intersection is $\{1, 2, \ldots, 10\}$.

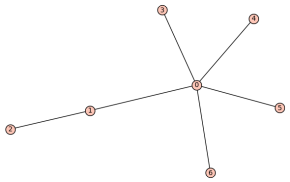
1.3.1 Graph classes with common vertices in their longest paths

We know that there are some graph classes with a common vertex for all its longest paths. Before diving deeper into these we are going to need some definitions in order to know which graph types we are talking about.

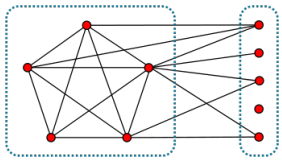
▶ Definition 5. Let $G = (V, E)$ be a graph. G is called a

- *tree*, if the graph is acyclic and undirected, see Figure 1.6a.
- *split graph*, if V can be partitioned into a clique and an independent set, see Figure 1.6b.
- *cactus graph*, if any two simple cycles have at most one shared vertex, see Figure 1.6c.
- *outerplanar graph*, if there exists a planar embedding, where all vertices belong to the outer face of the embedding, see Figure 1.7a.
- *circular arc graph*, if it is an intersection graph of arcs of a circle. That means each arc in the set represents a vertex and if there are two arcs, which intersect, then we have an edge between the corresponding vertices, see Figure 1.7b.
- $2K_2$*-free graph*, if it does not contain two independent edges as an induced subgraph, see Figure 1.7c.

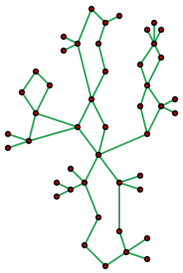
◀



(a) Tree

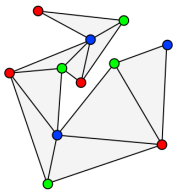


(b) Split graph

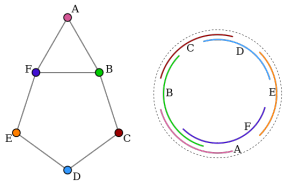


(c) Cactus graph

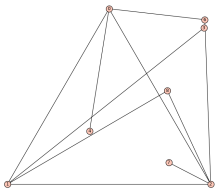
Figure 1.6: Graph classes



(a) Outerplanar graph



(b) Circular arc graph



(c) $2K_2$-free graph

Figure 1.7: Graph classes

Now, with the relevent graph classes from Definition 5, we present a series of known results from literature. We omit the proofs here, as they can be found in the corresponding references.

Proposition 6 (Intersection for trees 2). Let G be a tree. Then the intersection of all longest paths in G is nonempty.

Atleast the center, which is the middle vertex of every longest path in a tree, is a part of the intersection. Continuative information will be treated in Chapter 2.

Proposition 7 (Intersection for split graphs [5]). Let G be a split graph. Then the intersection of all longest paths in G is nonempty.

Proposition 8 (Intersection for cactus graphs [5]). Let G be a cactus graph. Then the intersection of all longest paths in G is nonempty.

Proposition 9 (Intersection for outerplanar graphs [2]). Let G be an outerplanar graph. Then the intersection of all longest paths in G is nonempty.

Proposition 10 (Intersection for series-parallel graphs [1]). Let G be a series–parallel graph. Then the intersection of all longest paths in G is nonempty.

Proposition 11 (Intersection for dually chordal graphs [4]). Let G be a dually chordal graph. Then the intersection of all longest paths in G is nonempty.

Proposition 12 (Intersection for $2K_2$-free graphs [3]). Let G be a nonempty $2K_2$-free graph. Then every vertex of maximum degree is common to all longest paths.

2 Intersection of longest paths in trees

Since we know that the intersection of all longest paths in a tree is nonempty, we will take a closer look at the shape of it.

Theorem 13. The intersection of all longest paths in a tree is again a path.

Proof. Let $G = (V, E)$ be a tree. Suppose the intersection is not a path. We first show, that the subgraph induced by the vertices of the intersection (blue vertices) cannot contain a vertex of degree ≥ 3. Assume that the subgraph induced by the intersection has a vertex w with degree > 2, see Figure 2.1. Thus, all longest paths contain w, as well as the neighbours of w. As paths are sequences of (distinct) vertices and edges, this is not possible without using a cycle, and there are no cycles in trees.

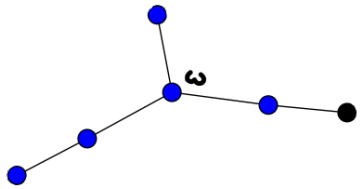


Figure 2.1: Subgraph with degree > 2

If we suppose that the intersection is not a path, then we have vertices $v_1, \ldots, v_n$, which are in the intersection and at least one vertex w, which interrupts the path.

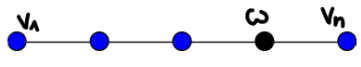


Figure 2.2: Path with interrupting vertex

Because v_1 and v_n are in the intersection, they are also in all longest paths. It holds that in a tree there is always only one explicit path between two vertices.

That means that there is also an explicit path between v_1 and v_n, which holds the disrupting vertex w as well. Because of that, w is also part of all longest paths and therefore an element of the intersection. This is a contradiction to the assumption that w is not part of the intersection. It follows that the intersection of all longest paths in a tree is again a path. □

Theorem 14. The path, which emerges after iteratively cutting away leaves from a tree until a path remains, is a part of the intersection of all longest paths.

As an example we can take a look at the following tree. If we remove all the leaves (red vertices) in the left graph, we have done one round of "leaf cutting" and get the graph on the right side.

Figure 2.3: leaf cutting

Proof. Let $G = (V, E)$ be a tree. Assume that P is a longest path in G. Then P has to start and end with a leaf, otherwise P could be extended and would not be a longest path. After one round of cutting away leaves of P we get a shorter path $\tilde{P}$ with $\text{length}(\tilde{P}) = \text{length}(P) - 2$.

Now we are going to assume, that $\tilde{P}$ is not a longest path in the reduced tree $\tilde{G}$. That means there must exist a longest path $\tilde{Q}$ in $\tilde{G}$ with $\text{length}(\tilde{Q}) > \text{length}(\tilde{P})$. It holds, that $\tilde{Q}$ has to span from leaf to leaf as well, because otherwise it could be extended as mentioned before.

By reversing the leaf cutting we can use the path $\tilde{Q}$ to construct a path Q in the original tree G. In order to execute this prolongation at least one new leaf at every side of $\tilde{Q}$ has to grow back and therefore it follows that $\text{length}(Q) = \text{length}(\tilde{Q}) + 2$.

As P is a longest path in G, it holds that $\text{length}(Q) \leq \text{length}(P)$. If we now use the connection with the leaf-cutted paths, we get

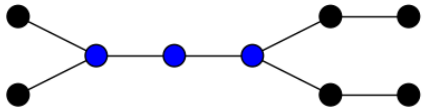
$$\text{length}(Q) \leq \text{length}(P) \iff \text{length}(\tilde{Q}) + 2 \leq \text{length}(\tilde{P}) + 2 \iff \text{length}(\tilde{Q}) \leq \text{length}(\tilde{P}).$$

This is a contradiction to $\text{length}(\tilde{Q}) > \text{length}(\tilde{P})$, which we have already established earlier.

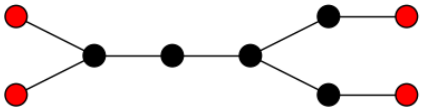
That means that $\tilde{P}$ is a longest path. If we now iterate the leaf cutting in the tree G until only a path remains, we will still have a longest path which starts and ends with a leaf. This means that this path is a part of the intersection of all longest paths of the original tree. □

We are going to introduce two examples in order to show that Theorem 14 is sharp in some cases. First we will consider a tree with 9 vertices, where the leaf cutting only yields a subset of the intersection.

► Example. In Figure 2.4 the blue coloured vertices are the intersection of all longest paths and the red coloured vertices are leaves in the original tree.



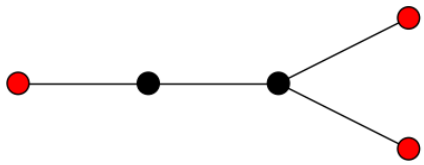
(a) Intersection of all longest paths



(b) Graph before one round of leaf cutting

Figure 2.4: Tree on 9 vertices

After one round of leaf cutting we get the graph on the left side of Figure 2.5, where the leaves are already coloured red again. Furthermore we can see that the graph on the left does not resemble a path. After another round of leaf cutting, we get the path on the right side.



(a) Graph after one round of leaf cutting

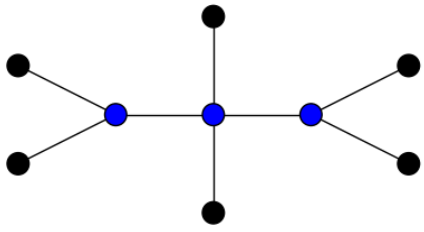
(b) Graph after two rounds of leaf cutting

Figure 2.5: Tree on 9 vertices

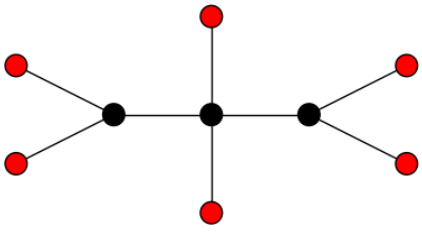
The leaf cutting resulted in a path with 2 vertices, which are in the intersection . However, these vertices are only a subset of the actual intersection containing 3 vertices, see Figure 2.4a. ◄

Next, we will analyse a tree which has also 9 vertices but where the leaf cutting method gives us the whole intersection of all longest paths.

► Example. In Figure 2.6 the blue vertices resemble the intersection and the red vertices the leaves of the original tree.



(a) Intersection of all longest paths



(b) Graph before one round of leaf cutting

Figure 2.6: Tree on 9 vertices

We can see in Figure 2.7 that after only one round of leaf cutting we are left with a path with 3 vertices, which is exactly the intersection of all longest paths.

Figure 2.7: Graph after one round of leaf cutting

◀

We conclude that there exist trees, where the Theorem 14 is sharp, i.e. we get the whole intersection with our leaf cutting method.

3 Implementations in SageMath

SageMath contains a lot of objects and methods to create and work with graphs. In connection with the intersection of longest paths in connected graphs there already exists the `longest_path()` method which gives us one longest path in a graph. For example,

```
g = graphs.PetersenGraph()
g.longest_path()
```

returns one longest path of the Petersen graph, as depicted in Figure 3.1.

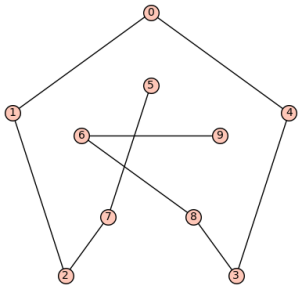


Figure 3.1: Longest path in Petersen graph

As we are looking for the intersection of **all** longest paths, we need to implement a function, which returns not only one longest path, but all of them. In order to reach this goal, we are going to consider permutations.

3.1 Permutations

The basic idea is that we utilize the already existing method `longest_path()` to identify the length n of the longest path. Next we produce a list of length-n partial permutations of the vertex set V, that represents paths. If we have the permutations $[0,1,3,4]$ and $[4,3,1,0]$ we just add one of them to our list, because both of them correspond to the same path. For example, if $n = 3$ and the order of the graph is 4, then we have the list $[[0,1,2],[1,2,0],[2,0,1],[0,1,3],[1,3,0],[3,0,1],[1,2,3],[2,3,1],[3,1,2],[0,2,3],[2,3,0],[3,0,2]]$. The next step is to take each permutation and check

for edges between consecutive vertices in the list. If we mathematically rephrased it, then we would check for edges between the i-th and $(i+1)$-th vertex for $i \in \{1,\dots,n-1\}$. For example when looking at the permutation $[0,3,1]$, we would first test if there is an edge between the vertex 0 and the vertex 3 and then we would check for an edge between the vertices 3 and 1. If all edges between the vertices exist, we have found a longest path. By checking all of the permutations, we get a list of all longest paths. We can say that this approach is a brute force method. This could be implemented in SageMath as follows.

```
def all_longestpaths_perm(G):
        L = []
        n = len(G.longest_path())
        P = [perm for perm in Permutations(G.vertices(), n)
                if perm[0] < perm[-1]]
        for p in P:
                c = 0
                for i in range(n-1):
                        if not G.has_edge(p[i], p[i+1]):
                                c=1
                if c == 0:
                        L.append(p)
        return L
```

3.2 All paths

The next idea is to consider all paths in the graph by constructing all paths between all pairs of vertices and picking all of those whose length is equal to the length of a longest path (which we know from the `longest_path` method). This could be implemented as follows.

```
def all_longestpaths_all(G):
        n = len(G.longest_path())
        L = []
        for (i, j) in Set(G.vertices()).subsets(size=2):
                for path in G.all_paths(i,j):
                        if len(path) == n:
                                L.append(path)
        return L
```

3.3 Comparing the execution time for the Petersen graph

As we now have two different methods for receiving a list of the longest paths in a graph we are interested in the computational performance. For the test we are going to take the Petersen graph as an example.

```
%%timeit
all_longestpaths_perm(graphs.PetersenGraph())
all_longestpaths_all(graphs.PetersenGraph())
```

The permutations method takes 45.2 s while the `all_paths` method finishes after only 14.7 ms. From now on we are going to use the second method, given that it is a lot more efficient.

3.4 Intersection of the longest paths

Until now we only have a list of longest paths. As we are interested in the intersection of those we are going to use the following method, which returns a set of the vertices of the intersection.

```
def intersec_all_longestpaths(G):
        L = all_longestpaths_all(G)
        a = set.intersection(*[set(path) for path in L])
        if len(a) == 0:
                print ("The intersection of all its longest paths is empty.")
        return a
```

For visualization we are also going to use methods which draw one of the longest paths and another that colours the vertices in the intersection.

```
import random

def draw_path(G, L, n, **kwargs):
        if n > len(L):
                print("Cannot find the path you are looking for")
        Ln = L[n]
        edges = []
        for i in srange(len(Ln)-1):
                edges.append((Ln[i],Ln[i+1]))
        return G.graphplot(edge_colors={(1, 0, 0): edges}, **kwargs)

def draw_rand_path(G, **kwargs):
        L = all_longestpaths_all(G)
        n = len(L)
        a = random.randint(0, n-1)
        B = draw_path(G, L, a, **kwargs)
        B.show()

def show_intersec(G, **kwargs):
        L = intersec_all_longestpaths(G)
        B = G.graphplot(vertex_colors={(0, 0, 1): L}, **kwargs)
        B.show()
```

3.5 Graph classes with nonempty intersections

As already discussed in Chapter 1.3.1 there are some graph classes, where we know that they have atleast one common vertex in their intersection. We are going to take an example out of each and apply the functions discussed in the previous section.

3.5.1 Trees

The first graph class we are going to illustrate is trees. For our example we consider a tree with 7 vertices, which we are going to construct as following:

```
T = list(graphs.trees(7))
t = T[9]
t.plot()
```

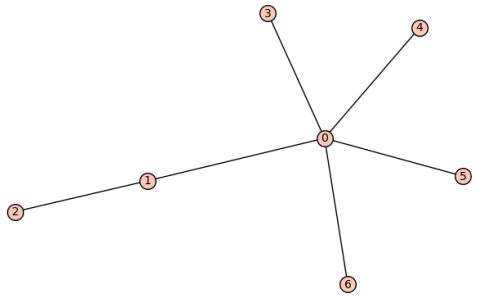


Figure 3.2: Tree with 7 vertices

Next we are going to draw a random longest path and get the set of the vertices in the intersection, in our example $\{0, 1, 2\}$, which is visualized in Figure 3.3.

```
draw_rand_path(t)
intersec_all_longestpaths(t)
show_intersec(t)
```

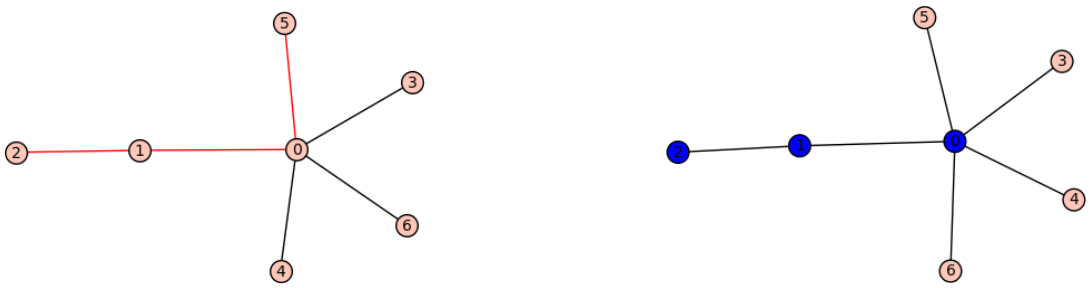


Figure 3.3: Visualization of a longest path and the intersection of all longest paths in a tree

3.5.2 Cactus graphs

Next in line are the cactus graphs. One sample would be the following:

```
Cactus = Graph({
        0: [1, 4], 1: [2], 2: [3], 4: [5, 6],
        7: [0, 3, 8, 10], 9: [8, 10, 11, 12, 13] })
plot(Cactus)
```

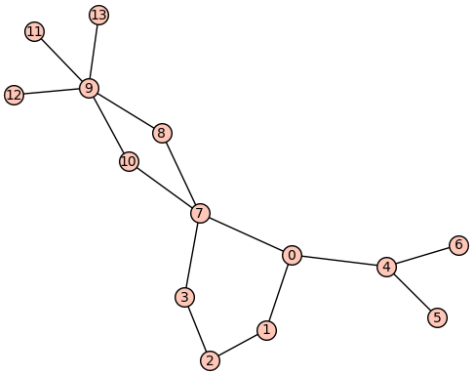


Figure 3.4: Cactus graph with 14 vertices

As before we are going to draw a random longest path and apply the method for the intersection of the longest paths, which gives us the set $\{0, 1, 2, 3, 4, 7, 9\}$, which we can see in Figure 3.5.

```
draw_rand_path(Cactus)
intersec_all_longestpaths(Cactus)
show_intersec(Cactus)
```

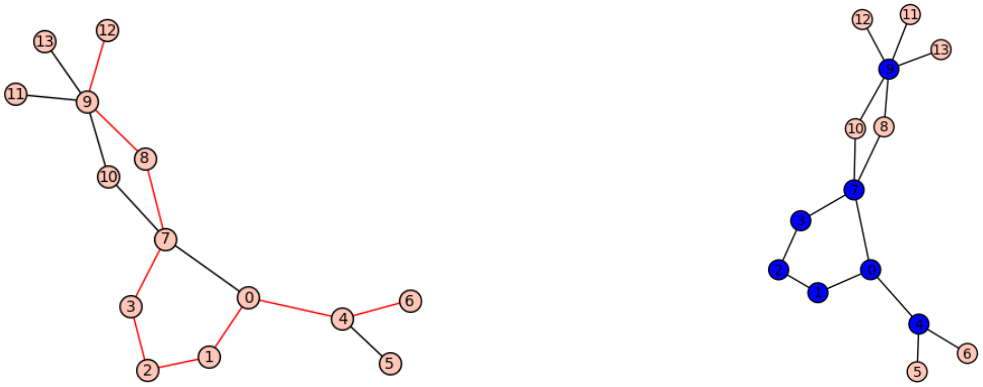


Figure 3.5: Visualization of a longest path and the intersection of all longest paths in a cactus graph

3.5.3 Split graphs

The third graph class we are going to look at is split graphs. We can construct one example with the following code:

```
split = Graph({0:[1, 2, 3, 4, 5], 1:[2, 3, 4, 5],
        2:[3, 4, 5, 6, 7, 9], 4:[3, 7, 8]})
posi = {
        0: (0, 1.0), 1: (-0.9510565162951535, 0.3090169943749475),
        2: (-0.5877852522924732, -0.8090169943749473),
        3: (0.5877852522924729, -0.8090169943749476),
        4: (0.9510565162951536, 0.3090169943749472), 5: (1.5, 1.0),
        6: (1.5, 0.54775), 7: (1.5, 0.0955), 8: (1.5, -0.35675), 9: (1.5, -1)}
split.graphplot(pos=posi).plot()
```

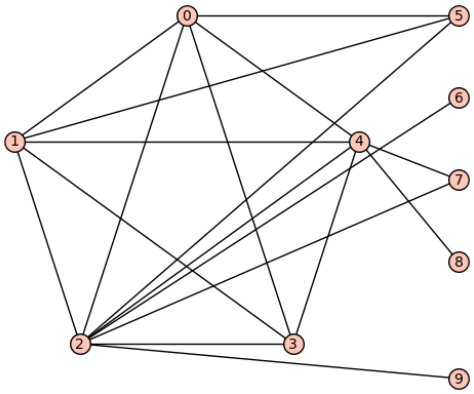


Figure 3.6: Split graph with 10 vertices

The next step is to draw a random longest path and to get the set of vertices in the intersection, in our example the set $\{0, 1, 2, 3, 4, 5\}$. The results are visualized in Figure 3.7.

```
draw_rand_path(split, pos=posi)
intersec_all_longestpaths(split)
show_intersec(split)
```

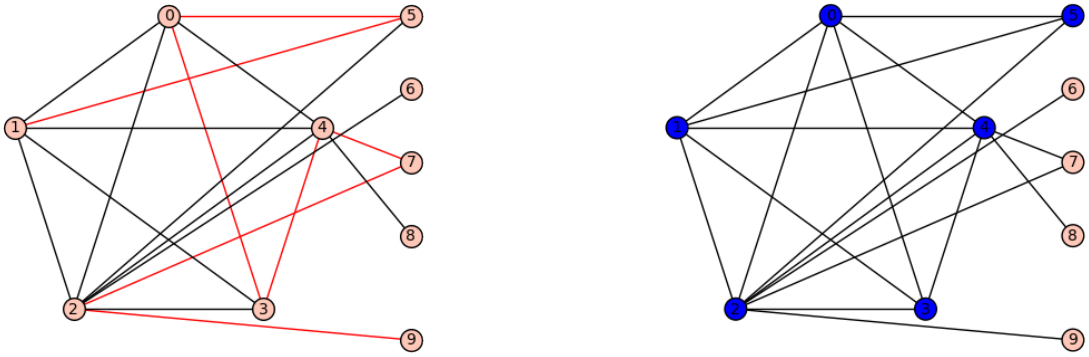


Figure 3.7: Visualization of a longest path and the intersection of all longest paths in a split graph

3.5.4 Circular arc graphs

Last but not least we are going to use our methods on circular arc graphs, one example would be the following:

```
circ = Graph({1: [0, 2, 5], 5: [0, 4], 3: [2, 4]})
plot(circ)
```

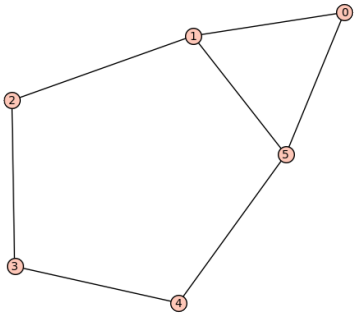


Figure 3.8: Circular arc graph with 6 vertices

Now we are again going to draw a random longest path and visualize in Figure 3.9 the vertices in the intersection, which is in our case the set $\{0, 1, 2, 3, 4, 5\}$.

```
draw_rand_path(circ)
intersec_all_longestpaths(circ)
show_intersec(circ)
```

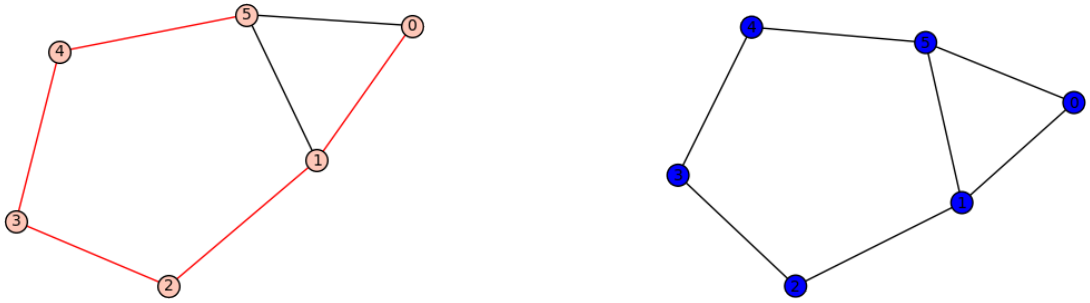


Figure 3.9: Visualization of a longest path and the intersection of all longest paths in a circular arc graph

3.5.5 Counterexample

As we already discussed in Chapter 1.3 there exists the counterexample of Walther [8] and Zamfirescu [9]. Now we want to apply our methods to check, if we really get an empty intersection. First we have to insert the graph:

```
counter = Graph({
        1: [0, 2, 3], 2: [8, 4], 3: [7, 5], 4: [5],
        6: [4, 7, 10], 7: [3, 8], 9: [5, 8, 11] })
posit = {
        0: (0, 1.2), 1: (0, 1.0),
        2: (-0.10, 0.666666666), 3: (0.10, 0.666666666),
        4: (-0.23, 0.33333333333333333), 5: (0.23, 0.33333333333333333),
        6: (-0.375, 0), 7: (-0.125, 0),
        8: (0.125, 0), 9: (0.375,0),
        10: (-0.625, 0), 11: (0.625, 0) }
counter.graphplot(pos=posit).plot()
```

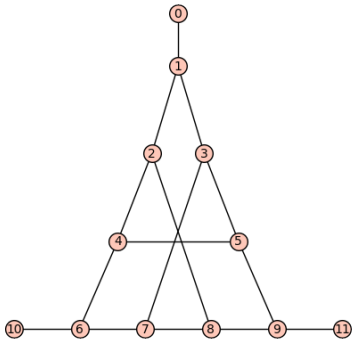


Figure 3.10: Counterexample

We can draw a random longest path into the graph. Contrary to the earlier examples, now the function returns ”The intersection of all its longest paths is empty.“ and the empty set with the method `intersec_all_longestpaths(counter)`. If we try to visualize the intersection we can also see, that no vertex is coloured, as it should be because of the empty intersection.

```
draw_rand_path(counter, pos=posit)
show_intersec(counter, pos=posit)
```

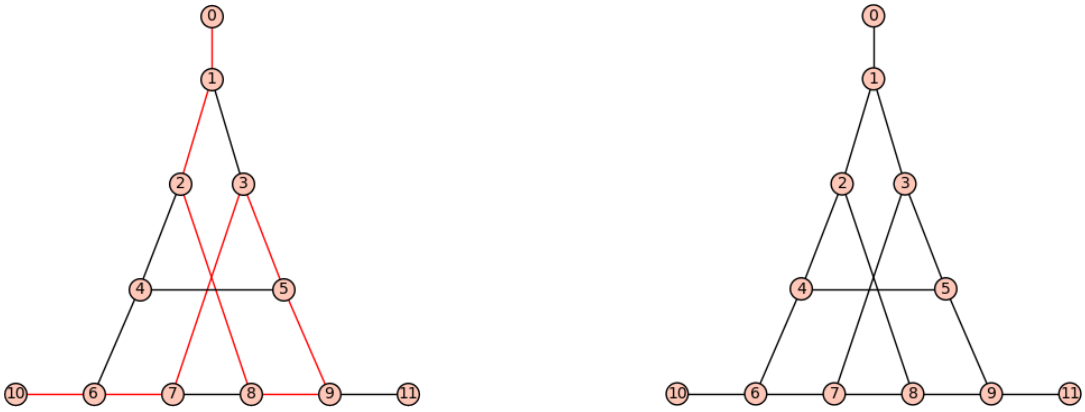


Figure 3.11: Visualization of a longest path and the nonempty intersection of all longest paths in the counterexample

3.6 Counterexample with fewer vertices

Since we know that there exists a counterexample on 12 vertices we are now interested if we can find a counterexample with a smaller order with our implemented methods. For this we use `graphs.nauty_geng("n -c")` [6] which allows us to create all existing connected graphs on n vertices. Next we check for all of these graphs, if they have an empty intersection and as soon we have found one we print ”Empty Intersection“. If we do not find any we print ”No empty intersections“.

```
B = []
for graph in graphs.nauty_geng("n -c"):
        a = intersec_all_longestpaths(graph)
        if len(a) == 0:
                print("Empty Intersection")
                B.append(graph)
                break
if len(B) == 0:
        print("No empty intersections")
```

We tested the code for $n \in \{3, 4, \cdots, 10\}$ and have not found a smaller counterexample. The runtimes of this code for the various graph orders can be found in Figure 3.12.

order n of the graph	runtime	number of graphes with n vertices
3	797 ms	2
4	800 ms	6
5	960 ms	21
6	1.17 s	112
7	5.12 s	853
8	131.75 s	11117
9	5039 s	261080
10	1098948.84 s $\approx$ 13 d	11716571

Figure 3.12: runtimes with various orders of graphs

As there exist 1006700565 graphs on 11 vertices – which are 85 times as many as graphs on 10 vertices – did we not calculate if there exists a counterexample on 11 vertices. Another aspect is that the runtime increased strongly when we went from order 9 to 10 and without a doubt would it go up drastically if we would calculate all connected graphs with 11 vertices.

4 Open questions

After engaging in the topic of this Bachelor's thesis and properly studying the intersection of longest paths in trees and the implementation of useful methods in SageMath there are still some open questions.

- Is there a counterexample on 11 vertices?
- Is there a counterexample on 12 vertices, that has fewer edges than Walther and Zamfirescu's counterexample in Figure 1.4?
- Can the leaf cutting characterization in Chapter 2 be improved such that we get the entire intersection all the time?
- Can the leaf cutting approach be used or modified for other graph classes as well?
- What is the probability that a random connected graph has an empty intersection of all longest paths?
- Are there other specific graph classes where the intersection is nonempty?
- In which graph classes do every three longest paths not have a common vertex?

Bibliography

[1] Guantao Chen, Julia Ehrenmüller, Cristina G Fernandes, Carl Georg Heise, Songling Shan, Ping Yang, and Amy N. Yates. Nonempty intersection of longest paths in series–parallel graphs. *Discrete Mathematics*, 340(3):287–304, 2017.

[2] Susanna F. de Rezende, Cristina G. Fernandes, Daniel M. Martin, and Yoshiko Wakabayashi. Intersecting longest paths. *Discrete Mathematics*, 313(12):1401–1408, 2013.

[3] Gili Golan and Songling Shan. Nonempty intersection of longest paths in $2K_2$-free graphs, 2016.

[4] Adam S. Jobson, André E. Kézdy, Jenő Lehel, and Susan C. White. Detour trees. *Discrete Applied Mathematics*, 206:73–80, 2016.

[5] Sandi Klavžar and Marko Petkovšek. Graphs with nonempty intersection of longest paths. *Ars Combin*, 29(43-52):4, 1990.

[6] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, 2014.

[7] Hansjoachim Walther. Über die Nichtexistenz eines Knotenpunktes, durch den alle längsten Wege eines Graphen gehen. *Journal of Combinatorial Theory*, 6(1):1–6, 1969.

[8] Hansjoachim Walther and Heinz-Jürgen Voss. *Über Kreise in Graphen*. VEB Deutscher Verlag der Wissenschaften, 1974.

[9] Tudor Zamfirescu. On longest paths and circuits in graphs. *Mathematica Scandinavica*, 38(2):211–239, 1976.