Seminar on Algorithms for Compressed Graphs Succinct Representation of Labeled Graphs

Matthias Dürksen

January 15, 2019

Abstract...

1 Introduction

Comment on the release of the interim version:

There are some gaps where explanations and extensions are still missing. Spelling needs to be improved. Chapter 3 will probably only touch on the topics superficially.

1.1 Motivation

1.2 Fundamentals

1.3 Header

2 Succinct Representation

The goal is to encode a graph as efficiently as possible and to be able to process query requests as efficiently as possible [1]. Therefore triangulated planar graphs are converted into three trees, which in turn are converted into parenthesis. The parenthesis for the three trees are then converted to a nested parenthesis. This bracketed representation is the efficient coding where requests can be answered efficiently.

2.1 Graph Model

A graph $G = (V, E)$ consists of a set of nodes V and a set of edges E. We consider undirected graphs so that the set of edges E is undirected. The succint representation is constructed so that the coding only works for a subset of graphs, the undirected planar graph. A planar graph is a G graph that can represent nodes in a two dimensional plane so that no edges intersect with another edge. For an example of a planar graph, see Figure 1.

In the first step, the planar graphs are converted to a triangulated planar graph. The graph from Figure 1 becomes a triangulated planar graph by the graph, see Figure 2. Therefore it can be assumed that the graph always has three vertices which are adjacent to each other. These edges form the convex hull of the graph, where we call the three nodes v_0, v_1 and v_{n-1}.

Explain how

Why always three vertices on convex hull

2.2 Schnyder realizer

The graph is divided into three trees for further compression. For this a Schnyder realizer [2] is used. This divides the graph into three trees T_0, T_1, T_2. Each inner edge is assigned to one of the

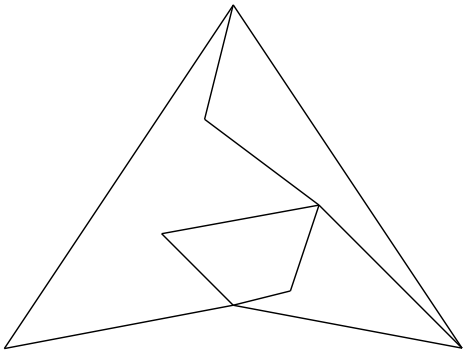

Figure 1: A planar graph

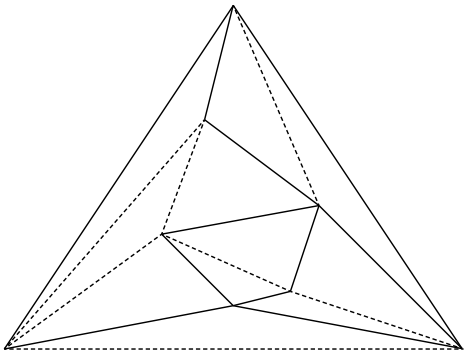

Figure 2: The triangulated graph from Figure 1

three trees. These are assigned so that each inner node has three outgoing edges, where each outgoing edge contains a different tree. In addition, the node can have incoming edges, where these must be arranged in a certain way. In Figure **??** we see the arrangement. The edges are arranged according to the pattern outgoing T_0, incoming T_1, outgoing T_2, incoming T_0, outgoing T_1, incoming T_2. The trees T_0, T_1, T_2 have as root the node v_0, v_1, v_{n-1}. Thus the graph, see Figure **??**, is divided into three trees, see Figure **??**.

Figure 3: Structure of the edges for an internal node in the graph.

Change directed edges

2.3 Traversal Orders

The graph is divided into three subtrees using a Schnyder realizer, but the edges lying on the convex hull of the graph are not yet contained in the trees. In addition we have to number the nodes for the following brackets. The two edges $(v_1, v_0), (v_{n-1}, v_0)$, which lie on the convex hull, are assigned to the tree T_0 and the edge (v_{n-1}, v_1) is assigned to the tree T_1. So T_0 is an canonical spanning tree.

To number the nodes in T_0 a depth search in the node v_0 is executed counterclockwise and the visiting nodes are numbered ascending, see Figure 5.

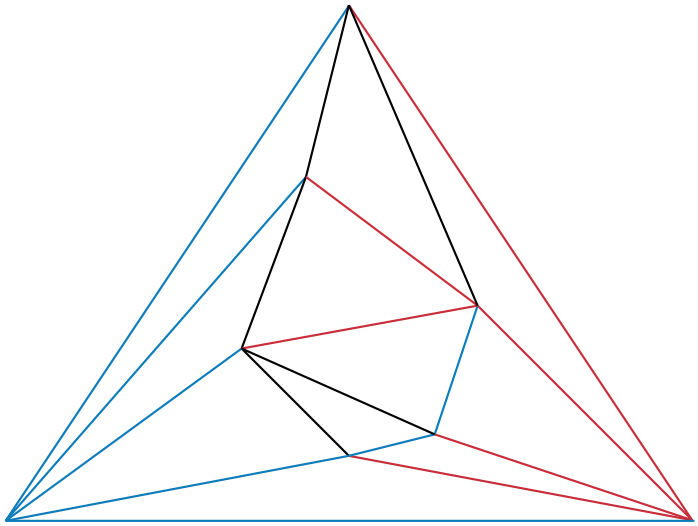

Figure 4: The trees in the graph which are formed on the basis of the Schnyder realizer

2.4 Parenthesis

The aim is to combine the trees with the help of brackets to a common brackets representation. To do this, we first generate the parenthesis for the individual trees and then combine them. For each of the trees T_0, T_1, T_2 the parenthesis is generated differently. We designate a pair of parenthesis as an opening parenthesis followed by a closing parenthesis.

Let's first look at the parenthesis for the tree T_0. Starting from the root v_0 this parenthesis is created. To encode the root we use pairs of brackets. For each of the children of depth 1 we create another pair of brackets in the pair of brackets. If a child a of depth 1 has children of its own, its children are displayed as a pair of parenthesis in the pair of parenthesis for the node a. The children are included in the parenthesis on a level countering the clockwise direction. Thus the following recursive definition follows:

For all nodes v of depth i applies:

$$i = 0 \rightarrow \text{pair of parenthesis}$$

i>0 $\rightarrow$ insert pair of parenthesis in $parent(v)(1)$

Arrange correctly

Figure 6 zeigt wie eine solche Klammerdarstellung für den Baum T_0 anhand unseres Beispiel aussieht.

For the tree T_1 the parenthesis representation is generated differently. A pair of parenthesis is created for each child of the root v_1, but the pairs of parenthesis are not written one after the other on the same level, but nested together. Here the edges are passed counterclockwise again, so that the child a, which is visited first, is on the outside and the other children are written into the brace pair of a. If a node has v children, the children are also nested into each other and packed in the place of the parenthesis, which comes directly after the closing parenthesis of a. The parenthesis for the tree T_1 from our example is shown in Figure 7.

The parenthesis for the tree T_2 is created similar to the parenthesis for T_1. Again the children are nested again, but the children are passed clockwise. If a node has v children, these nested children are now written in front of the opening parenthesis of v. This procedure is illustrated by the example in Figure **??**c).

The three parenthesis must now be combined so that no information about the graph is lost. The parenthesis of a tree T_i is called S_i. In order to be able to distinguish the parts of the

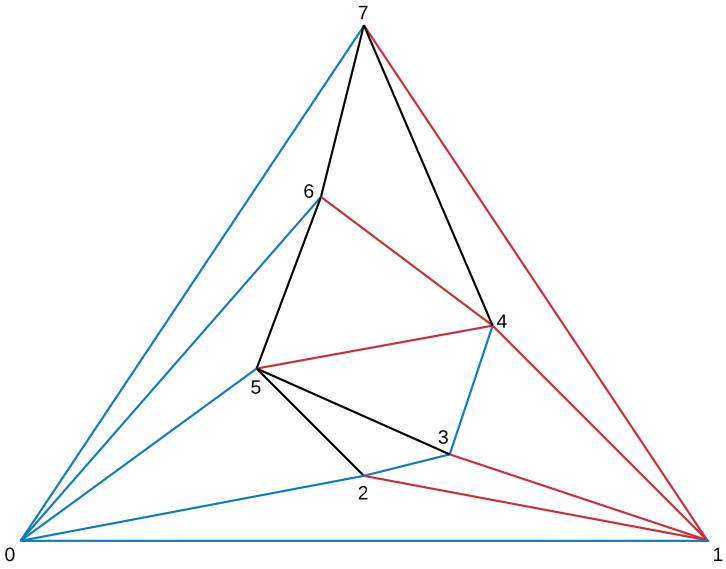

Figure 5: The partitioned graph with the numbering by T_0

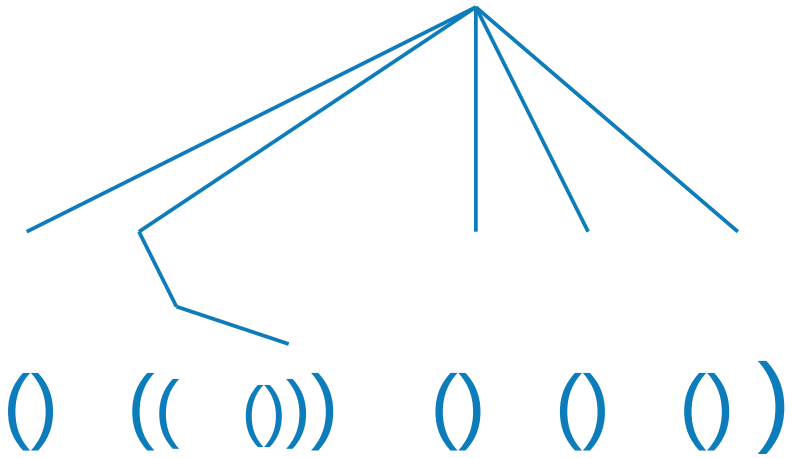

Figure 6: The parenthesis of the tree T_0

different parenthesis according to the combination, we use different parenthesis symbols for each parenthesis. The parenthesis S_0 serves as the basic structure. The other two parenthesis are included in this. The parenthesis are created as follows.

- $\forall e = (v_i, v_j) \in T_1 :$ insert $'['$ before $parenthesis(v_i) \in S_0$ closed.
- $\forall e = (v_i, v_j) \in T_1 :$ insert $']'$ after $parenthesis(v_i) \in S_0$ opened.
- $\forall e = (v_i, v_j) \in T_2 :$ insert $']'$ after $parenthesis(v_i) \in S_0$ opened.
- $\forall e = (v_i, v_j) \in T_2 :$ insert $'['$ before $parenthesis(v_i) \in S_0$ closed.

In the example, the three parenthesis become the combined representation in Figure 8. This creates a parenthesis that contains three types of parenthesis, each parenthesis still being a valid parenthesis.

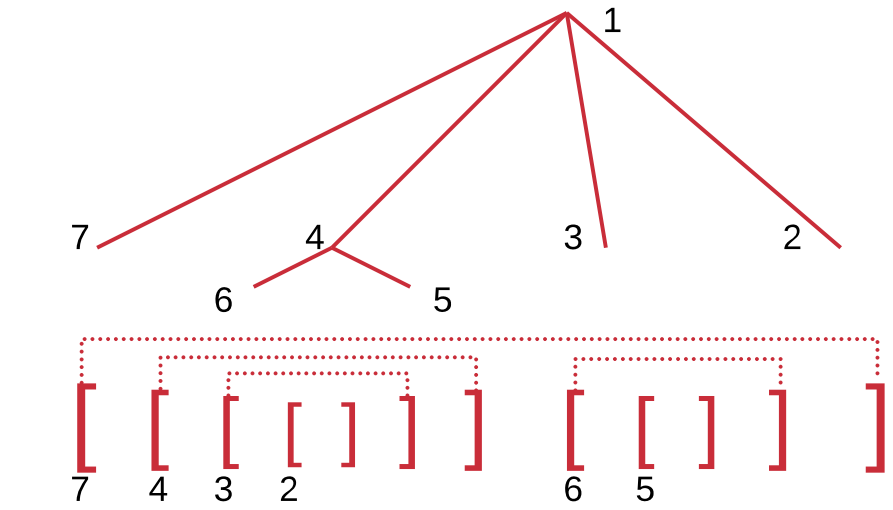

Figure 7: The parenthesis of the tree T_1

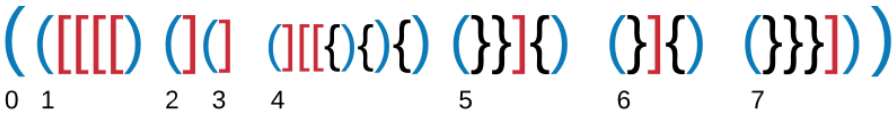

Figure 8: The parenthesis of the merges trees

2.5 Querys

2.6 Size complexity

The parenthesis representation consists of three different types of parenthesis, whereby 2 symbols are required for each type (opening and closing parenthesis). This means that a total of 6 different characters are required. Each edge of the graph is coded exactly once and one opening and one closing bracket are required per edge. Nodes do not have to be encoded separately, since the information indirectly contains the edges. Thus $2m$ symbols are needed for the representation, at m the number of edges is in G.

2.7 Reconstruction of the graph

3 Extensions

node/edge labels

k-page Graphs

4 Conclusion & Future Work

References

[1] Jérémy Barbay et al. "Succinct Representation of Labeled Graphs". In: *Algorithmica* 62.1 (Feb. 2012), pp. 224–257. ISSN: 1432-0541. DOI: 10.1007/s00453-010-9452-7. URL: https://doi.org/10.1007/s00453-010-9452-7.

[2] Walter Schnyder. "Embedding planar graphs on the grid". In: *SODA '90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*. San Francisco, California, United States: Society for Industrial and Applied Mathematics, 1990, pp. 138–148. ISBN: 0-89871-251-3. URL: http://portal.acm.org/citation.cfm?id=320191.