Data Structures and Algorithms II

Assignment 4

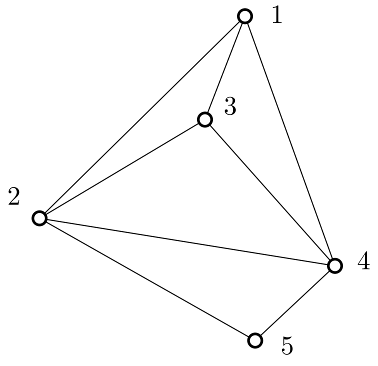
Wachmann Elias

January 22, 2022

1 Task Description

Triangulation 3-coloring
Your are given a triangulation of a point set. Your task is to design an efficient algorithm that constructs a valid 3-coloring of the points of the triangulation or determines that such a 3-coloring does not exist. A 3-coloring of the points is valid if any two points that are connected with an edge have different colors. The n points of the triangulation are labeled with the integers $\{1, \ldots, n\}$. The triangulation is given by a list of edges with additional triangle points (see Figure 1 for an example):

- Every edge is given by the labels of its two end points (first the smaller point label, then the larger one).
- For every edge, the labels of the point(s) with which the edge forms a triangle (a bounded triangular face) in the triangulation is given (two labels for interior edges and one label for edges on the boundary of the convex hull).



edges	**triangle points**
$1-2$	3
$2-4$	$3, 5$
$3-4$	$1, 2$
$1-3$	$2, 4$
$4-5$	2
$2-5$	4
$2-3$	$1, 4$
$1-4$	3

Figure 1: Example of a triangulation and a list of its edges with triangle points.

Explain and describe your algorithm in detail, analyze its runtime and memory requirements, and give reasons for the correctness of your solution.

2 Description of algorithm

It is assumed from the example, that the edges are always given with the lower vertex number in the first place e.g. 1-5 instead of 5-1. If this is not a given, the edge representation could be changed to conform to the above constraint in linear time, thus not increasing the asymptotic runtime.
Description

1. Setup: First get number of Points, which is the maximum in triangle points. Use this number to create an array with this size for the colors of the vertices. Store

edges (keys) and triangle points (values) in a hashmap for fast access. Setup counter which tracks how many edges need to be checked. Setup empty stack.

2. Start: Choose arbitrary start edge as current edge: e.g. first edge in list and color it with c_1 and c_2.
3. Loop: Check if triangle vertex of current edge is colored. If so, check if color is not one of the current edge colors, if this also applies –> stop no coloring exists. Else color triangle vertex with color which is not in current edge.
   Choose next Edge as follows: Pop Edge from stack if stack is not empty. Otherwise choose an edge from the most recent colored triangle vertices to one of the current points, such that

Jede Edge abchecken, damit man kein Dreieck übersieht.

1) Wähle beliebigen Startpunkt -> Färbe initial erste Edge Loop:
2) checke triangle points von current edge -> färbe falls keine farbe -) wenn triangle points farben falsch –> ABBRUCH
3) Auswahlregel für nächste Edge: Aus dem Stack nehmen wenn Stack nicht leer. Sonst Minimalste Edge nehmen z.B.: 2-4 3 (2-3) wählen restliche Edges in einen Stack falls mehrere möglich sind (Somit kann gewährleistet werden, dass jede Edge angeschaut wird und somit kein Dreieck übersehen wird.)

3 Korrektheit des Algorithmus

TODO