Submission for Algorithm Theory Exercise 3

Students: Anton Merlin Geburek (5348694), Matthias Zumkeller (5115157),
Kai Malaka (5518467), Manuel Di Biase (5515822)

November 8, 2024

1 Covering Unit Intervals

a)

Counterexample:

Given a set of rational numbers $Y = \{0.0, 0.75, 1.0, 1.5, 1.75, 2.5\}$.

Algorithmn A would return solution $S_1 = \{[0.75, 1.75], [0.0, 1.0], [2.5, 3.5]\}$ but an optimal solution would be $S_2 = \{[0.0, 1.0], [1.5, 2.50]\}$ A is stuck in a local minimum, without considering future coverage intervalls.

b)

An optimal greedy algorithm would be:

1. Sort the points in X in ascending order.
2. Start with the leftmost uncovered element $x \in X$.
3. Append the interval $[x, x + 1]$ to S.
4. Remove all elements $x \in X$ that are covered by $[x, x + 1]$.
5. Repeat steps 2–4 until the set X is empty.

Proving Optimality with an Exchange Argument:

Assume there is a optimal solution S_1 and an solution S_2, which is produced by our greedy algorithm.

Exchange Argument: Starting from the leftmost element x in X, S_2 covers it with the interval $[x, x + 1]$, which covers all elements from x to $x + 1$.

If S_1 uses an interval that starts at x, it will be equivalent to the interval from S_2, so we can exchange them and proceed with the next element not included in this first interval. If S_1 uses an interval that covers x but does not start at x, we can replace it with the interval $[x, x + 1]$ from S_2. This replacement does not increase the total number of intervals, but it aligns S_1 more closely to S_2.

Repeat the process: Move to the next leftmost point beyond $x + 1$ and repeat the replacement process.

In the end, we will have replaced all intervals in S_1 with intervals from S_2, without increasing the number of intervals in S_1.

Therefore the solution produced from greedy algorithm is optimal.

2 Graph coloring

a)

Given a *k-degenerate* graph $G = (V, E)$, we show that G can be colored with at most $k+1$ colors. We show this by presenting a method that labels a *k-degenerate* graph with at most $k+1$ colors.

Algorithm *GetColoring(G, k)*:

- If $|V| = 1$:
  - Label the single vertex $v = V[0]$ with label 1
  - Return the labeling
- Else:
  1. Choose a vertex $v \in V$ with $deg(v) \leq k$
  2. Run *GetColoring(G', k)* recursively on the remaining subgraph G' where $G' = (V \backslash \{v\}, E')$ and E' contains all edges of E that are not adjacent to v. Store the labeling for G'.
     Note that by definition each subgraph of a k-degenerate graph is also k-degenerate.
  3. Label v with a label l so that $1 \leq l \leq k+1$ and l is the minimum label that is not used for any of the at most k neighbours in the result of the recursive step 2.
  4. Combine the labelings for G' and the label for v. This results in a labeling for G.
  5. Return the combined labeling for G.

  The presented method computes a labeling for *k-degenerate* graphs using at most $k+1$ colors. Reason for this is, that in the base case ($|V| = 1$) only 1 color is needed, which satisfies our condition. In all other cases there has to be at least one vertex v that has $deg(v) \leq k$ by definition of *k-degeneracy*. Therefore v can be labeled with some label in $1, \ldots, k+1$, because the neighbours can have at most k different labels. Also each subgraph of G is *k-degenerate* because if a subraph G' of G would not be $k-$*degenerate* this would mean that there is a subgraph G^* of G' that does not contain any node with at most degree k. But G^* is also a subgraph of G and thus in this case G also would not be *k-degenerate*.
  We can therefore in a recusive step fix a vertex v with $deg(v) \leq k$, then recursively label the remaining graph with our method and then label v and in use in all steps only at most $k+1$ labels without breaking the requirement that two neighbours cannot have the same label. Hence we have presented a method that solves the problem of the exercise.

b)

We show that every planar graph can be colored with at most 6 colors. This is done by showing that each planar graph is at most *5-degenerate*. For this there are two steps:

1. The average vertex degree of a planar graph is smaller than 6. Each edge is adjacent to two vertices. Therefore it increases the sum of all degrees in the graph by 2. Therefore the average degree is calculated by the formula: $\frac{2 \cdot |E|}{|V|}$.

$$
\begin{aligned}
&\frac{2 \cdot (3|V| - 6}{|V|}) \\
&\leq \frac{6|V| - 12}{|V|} = 6 - \frac{12}{|V|} \\
&< 6
\end{aligned}
$$

Since the average degree of the vertices in a planar graph is smaller then 6 there has to be at least one vertex v with $deg(v) < 6$ and therefore $deg(v) \leq 5$.

2. The fact that each planar graph has a vertex with degree at most 5 does not yet prove that each planar graph is *5-degenerate*. It also needs to be shown that each subgraph of a planar graph is also a planar graph. Because in this case each subgraph does also have at least one vertex with degree at most 5. We show that for a subgraph $G' = (V', E')$ of a planar graph, that is constructed by removing a vertex with $deg(v) \leq 5$ and all adjacent edges from the original graph, $|E'| \leq 3|V| - 6$ holds:

$$\begin{aligned} |E| &\leq |E| - 5 \text{ Since the degree of the removed vertex is at most 5} \\ &\leq 3 \cdot |V| - 11 \\ &< 3(|V| - 1) - 6 = 3(|V'|) - 6 \end{aligned}$$

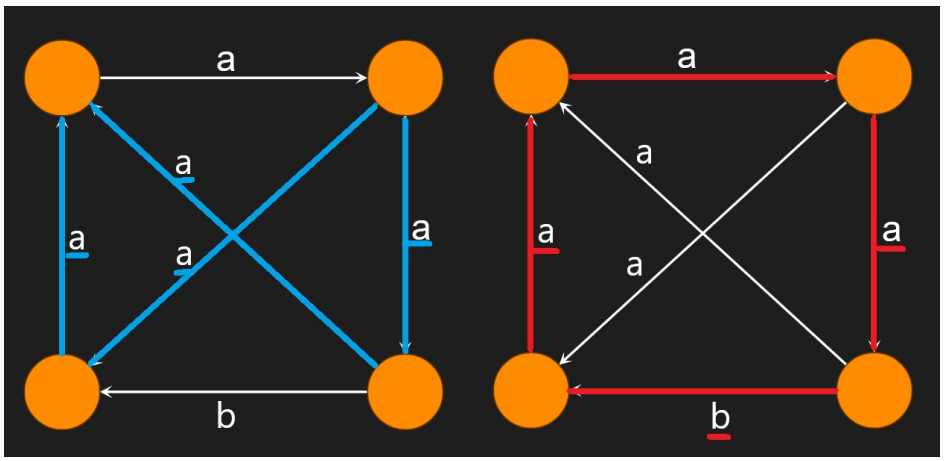
Thus each subgraph of a planar graph is also planar. This leads to the conclusion that each subgraph of a planar graph has to contain at least one vertex with degree at most 5. Therefore each planar graph is *5-degenerate*.

Applying our result from *exercise 2 a)* we conclude that each planar graph is (at most) *5-degenerate* and can therefore be colored with at most 6 colors.

3 Greedy TSP

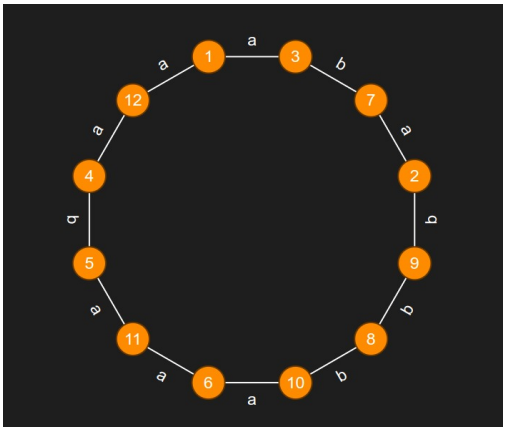
a)

In the following example, the greedy algorithm does not deterministically return the optimal solution, so it is not optimal. The optimal solution is marked in blue and a possible greedy solution in red.

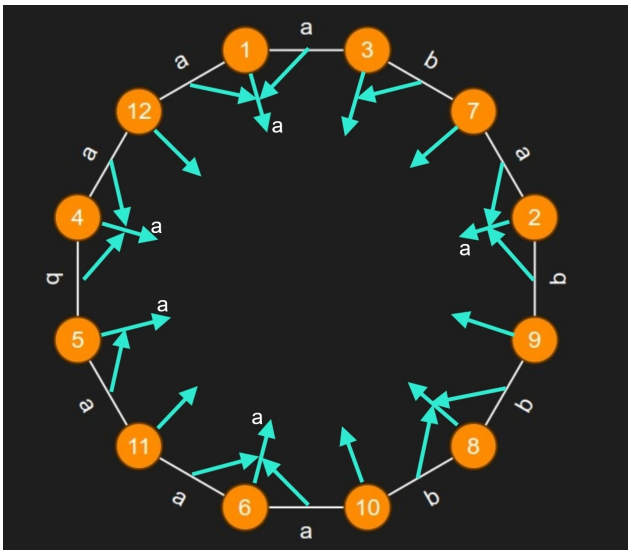


b)

Given any optimal solution with labeled edges. Label the vertices in the order, that the greedy algorithm visits them in.



Every edge e_o of the optimal solution is comparable to exactly one edge e_g in the greedy solution. Specifically the one connected to the vertex with the lower label. At some point the greedy algorithm reached that vertex and could have taken both edges e_o and e_g as it hasn't visited either one of the other two vertices connected to them. We can deduct, that e_g must have a smaller or equal weight then o_g, because otherwise the greedy algorithm would have taken e_o. Every greedy edge that is smaller or equal to an edge of the optimal solution with weight a must also have the weight a.
Notice, that every edge of the greedy solution can be mapped to a maximum of two edges of the optimal solution.



Every a in the optimal solution is now mapped to an a in the greedy solution. Assuming the optimal solution has k edges with weight a, the number of a's in the greedy solution can't get less then $\frac{k}{2}$

$$
\begin{aligned}
NN &= \frac{k}{2} \cdot a + (n - \frac{k}{2}) \cdot b \\
OPT &= k \cdot a + (n - k) \cdot b \\
\frac{NN}{OPT} &\leq \frac{\frac{k}{2} \cdot a + (n - \frac{k}{2}) \cdot b}{k \cdot a + (n - k) \cdot b} \qquad | \cdot 2 \\
&= \frac{k \cdot a + 2n \cdot b - k \cdot b}{2k \cdot a + 2n \cdot b - 2k \cdot b} \qquad | k \leq n \\
&\leq \frac{n \cdot a + 2n \cdot b - n \cdot b}{2n \cdot a + 2n \cdot b - 2n \cdot b} \\
&= \frac{n \cdot (a + b)}{n \cdot 2a} \\
&= \frac{a + b}{2 \cdot a}
\end{aligned}
$$