Karlsruher Institut für Technologie

Solution to Exercise Sheet 3

Submission due by June 4th, 2026

Problem 1: Odd Subgraph

4 + 8 = 12 points

Given a graph G and a parameter k. The problem Odd Subgraph asks whether there is a subgraph of G with k edges in which all vertices have odd degree. Note that a subgraph does not necessarily have to be vertex-induced.

Part (a) Show that if the maximum degree in G is at most k and G contains at least $k \cdot (2k-1)$ edges, then there is a matching in G with k edges.

Solution 1a:

We determine a matching greedily by choosing an arbitrary edge for the matching and deleting the edge itself as well as all adjacent edges. Since the vertex degree is at most k, each edge is adjacent to at most $2 \cdot (k-1)$ other edges. Thus, in each step, we delete at most $2k-1$ edges. Because $|E(G)| \geq k \cdot (2k-1)$, we will select at least k edges for the matching.

Part (b) Show that for Odd Subgraph, a kernel of size $O(k^2)$ is computable in polynomial time.

Hint: How do you solve the cases not covered by part (a)?

Solution 1b:

First, we delete all isolated vertices. If G contains fewer than $k \cdot (2k-1)$ edges, it is already a kernel of size $O(k^2)$. If G has at least $k \cdot (2k-1)$ edges, we can determine a solution in linear time. We distinguish two cases:

Case 1: There is a vertex v with degree $> k$.
If G is a star and k is even, there is no solution. If G is a star and k is odd, then a star with k edges is a valid subgraph.

Assume that G is no star, then there is an edge e that is not incident to v. Since $\deg(v) > k$, there are still $k-1$ other neighbors of v that are not incident to e. The edges from v to these neighbors, together with the edge e form a solution of size k. Such a solution can be determined in linear time.

Case 2: All vertices have degree $\leq k$.
As G contains at least $k \cdot (2k-1)$ edges, and by part (a), there is a matching in G with k edges. We can determine this in linear time. These k matching edges form a valid solution for Odd Subgraph.

Problem 2: Edge Clique Cover **16 points**

The parameterized problem Edge Clique Cover is defined as follows. Given a graph G and a parameter k. The goal is to find a set of at most k cliques such that every edge of G is contained in at least one of the cliques.

Provide reduction rules that compute a kernel with at most 2^k vertices for Edge Clique Cover. Argue why these reduction rules are safe and explain why the resulting kernel has at most 2^k vertices.

Hint: Two vertices u and v are called *true twins* if $N(u) \cup \{u\} = N(v) \cup \{v\}$, where $N(u)$ denotes the neighbors of u. Can you get rid of such true twins?

Solution 2:

Reduction Rule 1. Delete vertices without edges and keep k the same. This is a safe reduction because these vertices cannot appear in any clique that covers an edge.

Reduction Rule 2. If vertices a and b are *true twins* delete b (we write $G - b$ for the resulting graph). If (a,b) formed and isolated edge, also reduce k by 1.

To see that this reduction rule is safe consider the following. If $G - b$ can be covered by k cliques, then we can also cover G with k cliques (with $k+1$ if (a,b) formed an isolated edge). For this, take the solution for $G - b$ and add b to all cliques that contain a. Because the neighborhoods of a and b are identical, adding b will form valid cliques and all its edges will be covered as well. In the case that (a,b) formed an isolated edge, a was contained in no clique, and we add $\{a,b\}$ as a clique and increase k by 1 to cover this edge. On the other hand, if G can be covered by k cliques, then we can also cover $G - b$ with k cliques by simply removing b from all cliques in the solution for G (cover it with $k-1$ if (a, b) formed an isolated edge).

We will now show that after exhaustively applying these two reduction rules, the resulting kernel has at most 2^k vertices. To simplify this, we first show the following lemma:

Lemma 1. *Let a and b be any two distinct vertices in the kernel. The cliques in which a occurs are not the same as those in which b occurs.*

Proof. Assume that the lemma does not hold, and two distinct vertices a and b are contained in the exact same subset of cliques. Then they would have the same neighborhood in each of the cliques, and consequently, the entire neighborhood of both vertices would be identical, making them *true twins*. However, this is excluded by Reduction Rule 2, which is a contradiction. □

Now we can show that the kernel is bounded.

Theorem 2. *After repeatedly applying the two reduction rules to a solvable Edge Clique Cover instance with k cliques, the kernel is at most 2^k vertices large.*

Proof. As there are k cliques in the solution, there are at most 2^k different combinations of which cliques a vertex is contained in. Since these combinations of cliques are pairwise different for vertices according to our lemma, there can be at most 2^k vertices in the kernel. □

Problem 3: Vertex Cover on bipartite graphs $4 + 4 + 4 = 12$ points

The *Application of Lenstra* exercise from the previous version was moved to sheet 4 and replaced with this new exercise.

Part (a) Show that the LP relaxation of the ILP for Vertex Cover has an optimal integer solution if the graph is bipartite (i.e. that there is an optimal solution in which all variables are set to 0 or 1).

Solution 3a:

Based on the lecture, we know that the LP relaxation has a half-integral solution, i.e., a solution where all variables are in $\{0, \frac{1}{2}, 1\}$. To obtain a new solution where all variables are either 0 or 1, we only change variables with value $\frac{1}{2}$. We have to ensure that all edges in the new solution are still covered and that the size does not increase. Edges that are incident to a 1 variable are covered in both solutions. Edges between two 0s or a 0 and $\frac{1}{2}$ variable cannot exist in the old solution, as they would not be covered. Therefore, we only have to ensure that edges between two $\frac{1}{2}$ variables are still covered. For this, consider the subgraph of all edges where both endpoints have value $\frac{1}{2}$. This graph is bipartite, as it is a subgraph of a bipartite graph. We set the variables in smaller bipartite component to 1 and the other variables to 0 (or do this for every connected component). This ensures that all edges are covered and additionally, that the solution size does not increase.

Part (b) *This subexercise builds on lecture 6 (June 1).*

Use the duality theorem to prove Kőnig's theorem. Kőnig's theorem states that in a bipartite graph the number of vertices in a minimum vertex cover equals the number of edges in a maximum matching.

You can use (without proof) that the LP relaxation of the ILP for Maximum Matching has an optimal integer solution if the graph is bipartite.

Solution 3b:

We will show that the LP relaxations of the ILPs for Maximum Matching and Minimum Vertex Cover are dual to each other. Using the duality theorem, we can then conclude that the optimal values of these two LPs are the same. As both LP relaxations have optimal integer solutions in bipartite graphs, we can conclude that the size of a maximum matching equals the size of a

minimum vertex cover in bipartite graphs, which is exactly Kőnig's theorem.

Consider the LP-relaxation of Minimum Vertex Cover. For this, let $A \in \mathbb{R}^{n \times m}$ be the $n \times m$ incidence matrix of the graph, i.e., $a_{ij} = 1$ if vertex i is incident to edge j and 0 otherwise. Let $y \in \mathbb{R}^n$ be the vector of variables for the vertex cover, where y_i corresponds to vertex i. The LP can be written as follows:

$$\text{minimize } \sum_{i=1}^{n} y_i \qquad \text{with } A^T y \geq \mathbf{1} \qquad \text{and } y \geq \mathbf{0}$$

For the LP-relaxation of Maximum Matching, let $x \in \mathbb{R}^m$ be the vector of variables for the matching, where x_j corresponds to edge j. The LP can be written as follows:

$$\text{maximize } \sum_{j=1}^{m} x_j \qquad \text{with } Ax \leq \mathbf{1} \qquad \text{and } x \geq \mathbf{0}$$

For Minimum Vertex Cover the constraint $A^T y \geq \mathbf{1}$ ensures that for every edge, at least one of its endpoints is selected in the vertex cover. For Maximum Matching the constraint $Ax \leq \mathbf{1}$ ensures that for every vertex, at most one incident edge is selected in the matching.

These two LPs are dual to each other, and they clearly both have a feasible solution. According to the duality theorem, the optimal values of these two LPs are the same.

Part (c) Prove that we can find a minimum vertex cover in a bipartite graph in time $\mathcal{O}(m\sqrt{n})$.

You should use part (b) and the fact that a maximum matching can be computed in time $\mathcal{O}(m\sqrt{n})$ on bipartite graphs (you don't need to prove this).

Solution 3c:

According to Kőnig's theorem, the size of a minimum vertex cover equals the size of a maximum matching in bipartite graphs. Therefore, we can first compute a maximum matching in time $\mathcal{O}(m\sqrt{n})$ and then compute a vertex cover of the same size in linear time.

To compute the vertex cover from the maximum matching, observe the for every matched edge, exactly one of its endpoints must be in the vertex cover. This allows us to formulate the problem as a 2-SAT instance. We introduce a variable for every vertex, which is true if the vertex is in the vertex cover and false otherwise. For every matched edge uv, we add the clauses $(u \vee v)$ and $(\neg u \vee \neg v)$, which ensure that exactly one of its endpoints is in the vertex cover. For every unmatched edge uv, we add the clause $(u \vee v)$, which ensures that at least one of its endpoints is in the vertex cover. Additionally, for every vertex v that is not incident to any matched edge, we add the clause $(\neg v)$, which ensures that only vertices belonging to a matching edge can be selected.

A solution to this 2-SAT instance clearly gives us a vertex cover. It is also of the same size as the maximum matching as for every matched edge, exactly one incident vertex is selected. The SAT

instance can be solved in linear time, and the resulting assignment gives us a vertex cover of the same size.

Problem 4: Closest String **10 bonus points**

In this task, you should implement a program (using any programming language you prefer) that solves the Closest String problem. Given is a set of strings. The task is to find a minimum k and a string s such that s has a Hamming distance of at most k to each given string.

Use the algorithm from Active Session 1. However, you are welcome to implement further rules and make optimizations.

Describe in the PDF submission which procedures you have implemented and what else you have optimized. Also, in the PDF submission, state for each instance the smallest k for which you found a solution. Additionally, submit the source code and your found solutions (in the format described below) as a ZIP file. You can get 2 points for each solved instance.

Input format: The first line contains n, the number of strings. The next n lines contain one string each. All of these strings have the same length.

Output format: One line with a string that has the minimum Hamming distance to the given strings.

Note: Using the file `validator.py`, you can determine the Hamming distance of your solution to the given strings. It outputs the maximum Hamming-Distance over all given strings.