Homework III, Theory of Computation 2025

Submission: **The deadline for Homework 3 is 23:59 on 22 May.** Please submit your solutions on Moodle. Typing your solutions using a typesetting system such as LaTeX is strongly encouraged! If you must handwrite your solutions, write cleanly and with a pen. Messy and unreadable homeworks will not be graded. No late homeworks will be accepted.

Writing: Please be precise, concise and (reasonably) formal. Keep in mind that many of the problems ask you to provide a proof of a statement (as opposed to, say, just to provide an example). Therefore, make sure that your reasoning is correct and there are no holes in it. A solution that is hard/impossible to decipher/follow might not get full credit (even if it is in principle correct). You do not need to reprove anything that was shown in the class—just state clearly what was proved and where.

Collaboration: These problem sets are meant to be worked on in groups of 2–4 students. Please submit only one writeup per team—it should contain the names of all the students. You are strongly encouraged to solve these problems by yourself. If you must, you may use books or online resources to help solve homework problems, but you must credit all such sources in your writeup and you must never copy material verbatim. Even though only one writeup is submitted, it is expected that each one of the team members is able to fully explain the solutions if requested to do so.

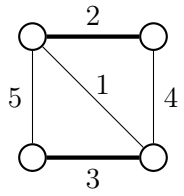
Grading: Each of the two problems will be graded on a scale from 0 to 5.

Warning: Your attention is drawn to the EPFL policy on academic dishonesty. In particular, you should be aware that copying solutions, in whole or in part, from other students in the class or any other source (e.g., ChatGPT) without acknowledgement constitutes cheating. Any student found to be cheating risks automatically failing the class and being referred to the appropriate office.

Homework 3

In the following problems, you may assume the **NP**-completeness of any of the problems discussed in the lectures: SAT, Independent-Set, Clique, Vertex-Cover, Set-Cover, Subset-Sum, etc. Make sure to prove that your reductions are correct!

1 Let $G = (V, E, w)$ be an undirected graph with edge-weights $w\colon E \to \mathbb{Z}$. A subset $M \subseteq E$ of the edges is a *perfect matching* if every vertex is incident to exactly one edge in M. In other words, the edges in M pair up all the vertices. The weight of a matching M is the total weight of its edges, that is, $\sum_{e \in M} w(e)$. An example is depicted below. The graph consists of 4 vertices, 5 edges with integer weights, and the thick edges indicate a perfect matching of weight $2 + 3 = 5$.



Show that the following problem is **NP**-complete:

$$\textsc{ExactMatch} = \{\langle G = (V, E, w), k\rangle : G \text{ has a perfect matching of weight exactly } k\}.$$

1 Solution to Question 1

We first need to show that ExactMatch $\in$ NP.

Given an instance $\langle G = (V, E, w), k\rangle$ and a certificate consisting of edges $M \subseteq E$, the verifier first checks if the sum of weights of edges in M is k and then checks that every vertex v occurs in exactly one edge in M. The first operation takes $|E|$ time and the second operation takes $|V| \cdot |E|$ time. Hence the verifier takes time polynomial in the size of the input and so ExactMatch $\in$ NP.

To show that ExactMatch is NP-hard, we perform the following reduction: SUBSET-SUM $\leq_p$ ExactMatch. Given an instance of SUBSET-SUM $\langle S = \{s_1, \cdots, s_n\}, t\rangle$, we construct an instance of ExactMatch as follows: For each number s_i, create 4 vertices $v_{i,1}, v_{i,2}, v_{i,3}, v_{i,4}$. Connect $(v_{i,1}, v_{i,2})$ with an edge of weight s_i; connect $(v_{i,2}, v_{i,3})$ with an edge of weight 0; connect $(v_{i,3}, v_{i,4})$ with an edge of weight 0; connect $(v_{i,4}, v_{i,1})$ with an edge of weight 0. Let G denote the new graph. Then $\langle G, t\rangle$ is the constructed instance of ExactMatch. For each s_i, we add 4 vertices and 4 edges to the graph; hence the construction is polynomial time in the size of the input.

Claim 1.1. *If there is a subset S such that $\sum_{j\in S} s_j = t$, then G has a perfect matching of size t.*

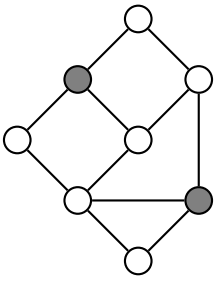
Proof. For every $i \in S$, choose the following edges as part of the matching: $(v_{i,1}, v_{i,2})$ and $(v_{i,3}, v_{i,4})$. If $i \notin S$, then choose $(v_{i,2}, v_{i,3})$ and $(v_{i,4}, v_{i,1})$ as part of the matching. It can be seen that this subset of edges is a perfect matching as it covers all the vertices. Furthermore, the only edges contributing to the weight of the matching are $(v_{j,1}, v_{j,2})$, for every $j \in S$. Therefore, the weight of the matching is t. □

Claim 1.2. *If G has a perfect matching of size t, then there exist a subset S such that $\sum_{j\in S} s_j = t$.*

Proof. Consider any connected square $\{v_{i,1}, v_{i,2}, v_{i,3}, v_{i,4}\}$. Since there are 4 edges in this square, two of these edges will be included in the perfect matching. It will either be $(v_{i,1}, v_{i,2})$ and $(v_{i,3}, v_{i,4})$, or $(v_{i,2}, v_{i,3})$ and $(v_{i,4}, v_{i,1})$. If $(v_{i,1}, v_{i,2})$ and $(v_{i,3}, v_{i,4})$ is part of the matching add i to S, otherwise don't add it. Since edge $(v_{i,1}, v_{i,2})$ has weight s_i, that means that adding i to S results in adding s_i weight to the subset S. Therefore, the matching of size k implies that the sum of weights of elements in S is also t. □

Grading scheme: 1.5 point for proof of **NP**-membership (0.5 for minor inaccuracies and 0.5 for missing running time analysis), 1.5 point for constructing a correct reduction f from an NP-complete problem P to ExactMatch, and 1 point for each direction of the proof that $f(x) \in$ ExactMatch iff $x \in P$.

2 Let $G = (V, E)$ be an undirected graph. We say that $S \subseteq V$ is a *blocking set* if it contains at least one node from each cycle in G. In other words, if we remove the nodes S (and all edges adjacent to S) from G, then we are left with a *forest* (graph with no cycles). For example, the highlighted nodes below form a blocking set of size 2.



Show that the following problem is **NP**-complete:

$$\textsc{Block} = \{\langle G, k\rangle : G \text{ is a graph that contains a blocking set of size } k\}.$$

2 Solution to Question 2

For **NP**-membership, let us describe a polynomial-time verifier. As a certificate, it takes a blocking set of size k. Verifier deletes all the vertices from blocking set with their adjacent edges from a graph G, and then checks that G does not contain any cycles (for example, with DFS or BFS).

For **NP**-hardness, let us describe a reduction from Vertex-Cover to Block. Given an instance of Vertex-Cover, $\langle G, k\rangle$, we map it to an instance of Block, $\langle G', k\rangle$.

G' is constructed from G as follows. For every edge $e = (v_1, v_2) \in G$ we have 3 vertices in G': v_1, v_2, v_e, and we connect each two of them with an edge such that they form a cycle of length 3.

Consider a vertex cover of size k in G. Its vertices exactly correspond to a blocking set in G'.

Let us note that it is enough for a blocking set to hit at least one vertex in every simple cycle, and from that it follows that it contains at least one node in every cycle.

Any cycle in G' is either a cycle in G or contains a newly introduced vertex v_e for some $e = (v_1, v_2)$. All cycles of the first kind are broken, as vertex cover in G is also a blocking set in G. All cycles of the second kind are also broken, since $v_e \in C \Rightarrow (v_1, v_e), (v_e, v_2) \in C$ for $(v_1, v_2) = e$. Since either v_1 or v_2 belongs to the vertex cover, C is broken.

For the other direction, consider a blocking set of size k in G'. Without loss of generality, we can assume that it does not contain any v_e for e an edge in G. If it does, simply swap this vertex for any of the two vertices v_1, v_2 such that $(v_1, v_2) = e \in G$.

After that, the blocking set in G' exactly corresponds to a vertex cover in G, as for every edge in contains at least one its endpoint.

Grading scheme: 1 point for proof of **NP**-membership (0.5 for minor inaccuracies) and 4 for **NP**-hardness ($3 - 3.5$ points for correct reductions with minor inaccuracies in the proof, $0 - 2.5$ points for more severe mistakes).