Course 02249, DTU Compute
Carsten Witt

# Computationally Hard Problems – Fall 2024
# Assignment Project

**Date**: 08.10.2024, **Due date**: 04.11.2024, 21:00

**This project should be performed in groups consisting of three students. Please register your group on DTU Learn and state the division of labor within your group in your submission.**

The following exercise is **mandatory**:

**Exercise Project.1:** Consider the following problem.

---

**Problem:** [MirrorFriendlyMinimumSpanningTree (MFMST)]
**Input:** An undirected, weighted graph $G = (V, E, w)$, where $V = \{1, \dots, n\}$, $E = \{e_1, \dots, e_m\}$ and $w \colon E \to \mathbb{N}_0$, and a number $B \in \mathbb{N}$.
**Output:** YES if there is a spanning tree $T \subseteq E$ for $G$ such that

$$\max\left\{\sum_{e_i \in T} w(e_i), \sum_{e_i \in T} w(e_{m+1-i})\right\} \leq B, \tag{1}$$

and NO otherwise.

---

Note that we do not allow multigraphs as input to the problem.

**What you have to do:**

a) Read and understand the problem. Describe in colloquial terms what the problem is about and explain the main difference to the standard minimum spanning tree (MST) problem. Solve the problem for the input $(V, E, w, B)$, where $V = \{1, 2, 3\}$, $E = \{e_1 = \{1, 2\}, e_2 = \{2, 3\}, e_3 = \{1, 3\}\}$, $w(e_i) = i$ for $i \in \{1, 2, 3\}$ and $B = 4$.

b) Describe the formal language that we use in the `.uwg` file format (see Part h) below) to represent inputs to MFMST and describe how to solve the word problem for a word over the underlying alphabet.

c) Assume you are given an algorithm $A_d$ for the decision version of MFMST as described above. Show how to convert it into an algorithm $A_o$ for the optimization version of the problem, i. e., an algorithm that given the specified input, constructs and outputs a spanning tree $T$ for $G$ such that Inequality (1) from above is satisfied for a right-hand side $B$ being as small as possible, or answers NO if no such spanning tree exists. The algorithm $A_o$ has to run in polynomial time, assuming that a call to $A_d$ takes one computational step.

d) Show that MFMST is in $\mathcal{NP}$.

e) Show that MFMST is $\mathcal{NP}$-complete. As reference problem you must select a problem from the list of $\mathcal{NP}$-complete problems given below. Note that there may be many different approaches to prove $\mathcal{NP}$-completeness.

f) Design an algorithm which solves the optimization version of the MFMST problem as defined in Part c). The algorithm is allowed have exponential worst-case running time (i. e. time $O(2^{p(\|\mathbf{X}\|)})$, where $p$ is some polynomial and $\|\mathbf{X}\|$ the input size), but may use "smart"/"heuristic" techniques to deal faster with some instances. Describe in words how the algorithm works and prove its correctness.

g) Prove the worst-case running time of your algorithm.

h) Some problem instances are given on DTU Learn as text files in the following `.uwg` ("undirected weighted graph") format:

The file is an ASCII file consisting of lines separated by the line-feed (LF) symbol; besides the line-feed, the only allowed characters in the file are the digits $\{\texttt{0}, \texttt{1}, \ldots, \texttt{9}\}$ and the blank symbol.

The first line of the file contains the number $n$ of vertices and the second line the number $m$ of edges, both in decimal representation. The subsequent $m$ lines encode the edges as blank-separated triples of non-negative integers in decimal representation, where the first two numbers denote the vertices the edge is incident on and the third one its weight. Implicitly, the entry in line $i + 2$ represents edge number $i$. For example, the graph used in a) is stored as follows:

```
3
3
1 2 1
2 3 2
1 3 3
```

Implement the algorithm you developed in Part f) and run it on some `.uwg` instances according to the following conventions:

- The instance is read from standard input; no command-line arguments.
- The output is written to standard output; either `NO` on a single line (terminated by LF) if there is no solution, or a sequence of lines each containing a decimal number between 1 and $m$ to represent the edges in your solution, followed by a line containing the value of the solution (i. e., the left-hand side of (1)) as decimal number. All lines are terminated by LF.

Instead of developing your software from scratch, you may build upon existing software packages, provided all legal restrictions are obeyed.

You have to submit exactly three files to the corresponding assignment on DTU Learn. Replace `XX` with your group number.

- a PDF file called `report-group-XX.pdf` with your solutions to the theoretical (non-programming) parts of this project,
- the full source code of your implementation as a ZIP file called `code-group-XX.zip`; the top level of the zip file should contain the root of the project, i. e. not a single directory that contains the root of the project (hint: zip the files/directories in the root of the project – not the directory containing the whole project);

- and an instruction how to compile and run your implementation as a text file called `readme-group-XX.txt`.

Do not duplicate files, in particular, do not include the report and readme file in the zip archive. Accepted programming languages are Java, C, C++ and Python. If you prefer other languages, you have to contact the teachers. Note that the latest version submitted before the deadline will be evaluated.
It is mandatory for the program to solve the first 5 instances published on DTU Learn to be eligible for the top score in this assignment. We reserve the right to deduce points if you do not comply with the above guidelines, including conventions for file types, file structures and names, which are case sensitive.

The three blocks [a)–e)], [f),g)], and [h)] have approximately weights of 50 %, 25 %, and 25 %, respectively, in the grading. Please state in your report the **division of labor**, i. e., describe the contributions of the individual group members.

---

**List of $\mathcal{NP}$-complete problems to choose from.**

---

**Problem:** [MinimumDegreeSpanningTree]
**Input:** An undirected graph $G = (V, E)$ and a natural number $k$.
**Output:** YES if there is a spanning tree $T$ in which every node has degree at most $k$; NO otherwise.

---

---

**Problem:** [MinimumCliqueCover]
**Input:** An undirected graph $G = (V, E)$ and a natural number $k$.
**Output:** YES if there is clique cover for $G$ of size at most $k$. That is, a collection $V_1, V_2, \dots, V_k$ of not necessarily disjoint subsets of $V$, such that each $V_i$ induces a complete subgraph of $G$ and such that for each edge $\{u, v\} \in E$ there is some $V_i$ that contains both $u$ and $v$. NO otherwise.

---

---

**Problem:** [Graph-3-Coloring]
**Input:** An undirected graph $G = (V, E)$.
**Output:** YES if there is a 3-coloring of $G$ and NO otherwise. A 3-*coloring* assigns every vertex one of 3 colors such that adjacent vertices have different colors.

---

---

**Problem:** [Maximum Common Induced Subgraph]
**Input:** Undirected graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ and a natural number $k$.
**Output:** YES if there is a common induced subgraph of cardinality $k$, i. e., subsets $V_1{}' \subseteq V_1$ and $V_2{}' \subseteq V_2$ such that $|V_1'| = |V_2'| = k$, and the subgraph of $G_1$ induced by $V_1'$ and the subgraph of $G_2$ induced by $V_2'$ are isomorphic. NO otherwise.

---

**Problem:** [LONGEST-COMMON-SUBSEQUENCE]
**Input:** A sequence $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n$ of strings over an alphabet $\Sigma$ and a natural number $B$.
**Output:** YES if there is a string $\boldsymbol{x}$ over $\Sigma$ of length $B$ which is a subsequence of all $\boldsymbol{w}_i$. The answer is NO otherwise.

Formally, we say that $\boldsymbol{x} = x_1 x_2 \cdots x_{\ell_x}$ is a *subsequence* of $\boldsymbol{w} = w_1 w_2 \cdots w_{\ell_w}$ if there is a strictly increasing sequence of indices $i_j$, $1 \le j \le \ell_x$, such that for all $j = 1, 2, \ldots, \ell_x$ we have $x_j = w_{i_j}$.

---

**Problem:** [MINIMUMRECTANGLETILING]
**Input:** An $n \times n$ array $A$ of non-negative integers, natural numbers $k$ and $B$.
**Output:** YES if there is a partition of $A$ into $k$ non-overlapping rectangular sub-arrays such that the sum of the entries every sub-array is at most $B$. NO otherwise.

---

**Problem:** [PARTITIONBYPAIRS]
**Input:** A sequence $S = (s_1, s_2, \ldots, s_{2n})$ of natural numbers.
**Output:** YES if there is a subset $A \subseteq \{1, \ldots, 2n\}$ choosing exactly one from each pair $(2i-1, 2i)$, where $i \in \{1, \ldots, n\}$, such that $\sum_{i \in A} s_i = \sum_{i \in \{1,\ldots,2n\} \setminus A} s_i$, and NO otherwise.

---

**Problem:** [1-IN-3-SATISFIABILITY]
**Input:** A set of clauses $C = \{c_1, \ldots, c_k\}$ over $n$ boolean variables $x_1, \ldots, x_n$, where every clause contains exactly three literals.
**Output:** YES is there is a satisfying assignment such that every clause has exactly one true literal, i. e., if there is an assignment

$$a\colon \{x_1, \ldots, x_n\} \to \{0, 1\}$$

such that every clause $c_j$ is satisfied and no clause has two or three satisfied literals, and NO otherwise.

---

End of Exercise Project.1