

Computationally Hard Problems – Fall 2020 Assignment Project

Date: 06.10.2020, **Due date:** 02.11.2020, 21:00

This project counts for three weekly assignments. It should be performed in groups consisting of either two or three students (3 is a hard maximum).

The following exercise is **mandatory**:

Exercise Project.1: Consider the following problem.

Problem: [MIRRORFRIENDLYMINIMUMSPANNINGTREE (MFMST)]

Input: An undirected, connected weighted graph $G = (V, E, w)$, where $V = \{1, \dots, n\}$, $E = \{e_1, \dots, e_m\}$ and $w: E \rightarrow \mathbb{N}_0$, and a number $B \in \mathbb{N}$.

Output: YES if there is a spanning tree $T \subseteq E$ for G such that

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

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) Show that MFMST is in \mathcal{NP} .
- c) 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.
- d) Design an algorithm which solves the optimizing version of the problem, i.e., given an undirected, connected weighted graph $G = (V, E, w)$, it 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. The algorithm is allowed have exponential worst-case running time but may use “smart”/“heuristic” techniques to deal faster with some instances. Describe in words how the algorithm works.

- e) Prove the worst-case running time of your algorithm.
- f) Some problem instances are given on DTU Learn as text files in the following UWG (“undirected weighted graphs”) 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 numbers $\{0, 1, \dots, 9\}$ 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 d) 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.

The program including the source code has to be submitted to the corresponding assignment on DTU Learn. Moreover, the code must also be uploaded to the CodeJudge system: <https://dtu.codejudge.net/02249-e20/assignments/view>. Accepted programming languages are Java, Python, C, C++ and C#. Other languages, as far as available on CodeJudge, have to be agreed upon with the teaching assistants.

To be eligible for the top score, your program must pass all tests that CodeJudge runs. Note that in addition to the mandatory assignment, there is a voluntary competition on CodeJudge that is not relevant for your score.

The three blocks [a),b),c)], [d),e)], and [f)] have approximately equal weights in the grading.

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 $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_n$ of strings over an alphabet Σ and a natural number B .

Output: YES if there is a string \mathbf{x} over Σ of length B which is a subsequence of all \mathbf{w}_i . The answer is NO otherwise.

Formally, we say that $\mathbf{x} = x_1x_2 \cdots x_{\ell_x}$ is a *subsequence* of $\mathbf{w} = w_1w_2 \cdots w_{\ell_w}$ if there is a strictly increasing sequence of indices i_j , $1 \leq j \leq \ell_x$, such that for all $j = 1, 2, \dots, \ell_x$ 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 $S = (s_1, s_2, \dots, s_{2n})$ of natural numbers.

Output: YES if there is a subset $A \subseteq \{1, \dots, 2n\}$ choosing exactly one from each pair $(2i - 1, 2i)$, where $i \in \{1, \dots, n\}$, such that $\sum_{i \in A} s_i = \sum_{i \in \{1, \dots, 2n\} \setminus A} s_i$, and NO otherwise.

Problem: [1-IN-3-SATISFIABILITY]

Input: A set of clauses $C = \{c_1, \dots, c_k\}$ over n boolean variables x_1, \dots, 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

$$a: \{x_1, \dots, x_n\} \rightarrow \{0, 1\}$$

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

End of Exercise 1
