Course 02249, DTU Compute
Carsten Witt

Computationally Hard Problems – Fall 2013
Assignment Project

Date: 08.10.2013, **Due date**: 04.11.2013, 23:59

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 $w\colon E \to \mathbb{N}_0$, and a number $B \in \mathbb{N}$.
Output: YES if there is a spanning tree $T \subseteq E$ for G such that

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

and NO otherwise.

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) Find 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 and B is 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 Campusnet 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 symbol; besides the line-feed, the only allowed characters in the file are numbers $\{\texttt{0}, \texttt{1}, \ldots, \texttt{9}\}$ 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 natural numbers in decimal representation, where the first two numbers denote the vertices the edge is incident on and the third one its weight. 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. 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 and an instruction how to execute it on an UWG instance file has to be delivered to the teaching assistant. Accepted programming languages are Java, C, C++, C#. Other languages have to be agreed upon with the teaching assistant.
Your programs will be run on some UWG files.

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 $V_1, V_2, \ldots, V_k$ 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 Σ and a natural number B.
Output: YES if there is a string $\boldsymbol{x}$ over Σ 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, $1 \leq j \leq \ell_x$, 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 1