Course 02249, DTU Compute
Carsten Witt

Computationally Hard Problems – Fall 2019
Assignment Project

Date: 08.10.2019, **Due date**: 04.11.2019, 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: [SuperStringWithExpansion]
Input:

a) 2 disjoint alphabets called Σ and $\Gamma = \{\gamma_1, \dots, \gamma_m\}$,

b) a string $s \in \Sigma^*$,

c) k strings $t_1, \dots, t_k \in (\Sigma \cup \Gamma)^*$,

d) and subsets $R_1, \dots, R_m \subseteq \Sigma^*$.

Output: YES if there is a sequence of words $r_1 \in R_1, r_2 \in R_2, \dots, r_m \in R_m$ such that for all $1 \le i \le k$ the so-called *expansion* $e(t_i)$ is a substring of s; the expansion $e(\gamma_j)$ of the j-th letter $\gamma_j \in \Gamma$, $1 \le j \le m$, is defined by $e(\gamma_j) := r_j$, and the expansion $e(t)$ of a whole string $t \in (\Sigma \cup \Gamma)^*$ is obtained by replacing all letters from Γ appearing within t by their expansions. Otherwise output NO.

Formally, we say that $\boldsymbol{v} = v_1 v_2 \cdots v_{\ell_v}$ is a *substring* of $\boldsymbol{w} = w_1 w_2 \cdots w_{\ell_w}$ if there is a j, $1 \le j \le \ell_w - \ell_v + 1$, such that for all $k = 1, 2, \dots, \ell_v$ we have $v_k = w_{j+k-1}$.

Also formally, *replacing* the i-th letter v_i of the string $\boldsymbol{v} = v_1 v_2 \cdots v_{\ell_v}$ by the string $\boldsymbol{w} = w_1 w_2 \cdots w_{\ell_w}$ results in the string $v_1 v_2 \cdots v_{i-1} w_1 w_2 \cdots w_{\ell_w} v_{i+1} v_{i+2} \cdots v_{\ell_v}$.

Some problem instances on the alphabets $\Sigma = \{\mathtt{a}, \mathtt{b}, \dots, \mathtt{z}\}$, $\Gamma = \{\mathtt{A}, \mathtt{B}, \dots, \mathtt{Z}\}$ are given on Campusnet as text files in the following SWE 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 $\{\mathtt{0}, \mathtt{1}, \dots, \mathtt{9}\}$, lower-case letters ($\Sigma$), upper-case letters ($\Gamma$), the colon ("`:`") and the comma symbol.

The first line of the file contains the number k. The second line contains the string s and the following k lines the strings $t_1, \dots, t_k$. Finally, the last lines (at most 26) start with a letter $\gamma_j \in \Gamma$ followed by a colon and the contents of the set R_j belonging to the letter, where the elements of the set are separated by commas.

An example: The file `test01.SWE` reads as follows:

```
4
abdde
ABD
DDE
AAB
ABd
A:a,b,c,d,e,f,dd
B:a,b,c,d,e,f,dd
C:a,b,c,d,e,f,dd
D:a,b,c,d,e,f,dd
E:aa,bd,c,d,e
```

What you have to do:

a) Read and understand the problem.

b) Determine whether the answer for `test01.SWE` is YES or NO.

c) Show that SuperStringWithExpansion is in $\mathcal{NP}$.

d) Show that SuperStringWithExpansion is $\mathcal{NP}$-complete. As reference problem you have to 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.

e) Design and implement a "decoder" which reads SWE files *from standard input* and checks whether they correctly code a SWE instance (with the alphabets Σ and Γ consisting of lower-case and upper-case letters as above). In particular, reject instances where required sets R_j are not correctly specified by printing `NO` to *standard output*. Do not submit this decoder alone but use it as a building block for the following tasks.

f) Design a heuristic algorithm which always gives the correct answer for an input, i. e., which always stops and determines whether the instance given on standard input is a YES or NO isntance. The algorithm is allowed have exponential worst-case running time. Describe in words how the algorithm works.

   If the instance is a NO-instance (or the input is malformed), your algorithm has to print `NO`. In case of a YES, your algorithm has to construct a solution $r_1, \ldots, r_m$ and must output the solution. The format of the output should then only be a list of lines in the format $\gamma_i \colon r_i$, where γ_i is an upper-case letter and r_i the chosen element of R_i, for example

   ```
   A:a
   B:b
   C:c
   D:d
   E:e
   ```

   (which, of course, is not a solution to `test01.SWE`).

g) Analyze the worst-case running time of the algorithm.

h) Implement the algorithm you developed in Part f). It has to be able to read inputs in SWE format from *standard input* (not as a command line argument) and, as we demand in f), must always solve the corresponding problem correctly by outputting a solution to *standard output* (or `NO` if no solution is possible). Test and submit your program on `CodeJudge` via `https://dtu.codejudge.net/02249-e19`.

A compiled or interpretable version of the program (.EXE/.JAR/.PY format) including the source code has to be submitted on Campusnet as well. Accepted programming languages are Java, C, C++, Python. Other languages have to be agreed upon with the teachers. `CodeJudge` will test your program on some SWE instances and it is mandatory to pass these tests to obtain the top score (6/6) for this assignment. Note that there is a voluntary competition on `CodeJudge` as well, which is not relevant for this score.

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

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

Problem: [PartitionInto3-Sets]
Input: A sequence $X = (x_1, x_2, \ldots, x_{3n})$ of $3n$ natural numbers, and a natural number B, such that $(B/4) < x_i < (B/2)$ for all $i \in \{1, 2, \ldots, 3n\}$ and $\sum_{i=1}^{3n} x_i = nB$.
Output: YES if X can be partitioned into n disjoint sets $X_1, X_2, \ldots, X_n$ such that for all $j \in \{1, 2, \ldots, n\}$ one has $\sum_{x \in X_j} x = B$.

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 if 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.

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: [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 numbers, positive integers 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: [Minimum Graph Transformation]
Input: Undirected graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ and an integer $k > 0$.
Output: YES if there is a transformation of order k that makes G_1 isomorphic to G_2, and NO otherwise. A transformation of order k removes k existing edges from E_1 and then adds k new edges to E_1.

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

End of Exercise 1