Course 02249, DTU Compute
Carsten Witt

Computationally Hard Problems – Fall 2018
Assignment Project

Date: 09.10.2018, **Due date**: 06.11.2018, 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: [CrosswordPuzzleConstruction]
Input: Given are an alphabet Σ not containing the symbols "blank" or "#", a non-empty set of strings $S = \{s_1, \ldots, s_m\}$ over Σ, and an $n \times n$ matrix A for a non-negative integer n. The entries in A are either "blank" or "#".
Output: YES if the blank entries of A can be filled with letters from Σ in such a way that the result is a crossword puzzle with strings from S. That is, every maximal horizontal or vertical sequence in A consisting of letters from Σ is a string from S when read left-to-right respectively downwards. The same string may occur more than once.

We call the problem CPC for short.
Example: $\Sigma = \{a, b, c\}$, $S = \{aa, bb, cac, ab, cab, ca, a, ac, bba, aab, bab\}$,

$$A = \begin{array}{|c|c|c|}\hline \# & & \\ \hline & & \\ \hline & \# & \\ \hline \end{array}.$$

The answer to this input is YES as the following solution shows:

$$A = \begin{array}{|c|c|c|}\hline \# & a & b \\ \hline c & a & b \\ \hline a & \# & a \\ \hline \end{array}$$

Some test instances will be supplied on Campusnet in text files with the extension `CPC`. They assume that Σ is a subset of the ISO-8859-1 character set, excluding "_", "#" and ";", and have the structure shown on the following page. Note that the underscore "_" replaces the blanks.

```
<alphabet size>;<no. of strings>;<puzzle size>
<first letter of alphabet>;...;<last letter of alphabet>
<entry (1,1) of A>; ... ; <entry (1,<puzzle size>) of A>
.
.
.
<entry(puzzle size,1)>; ... ; <entry (puzzle size,puzzle size) of A>
<string 1>
.
.
.
<string no. of strings>
```

For the example above the file looks like this:

```
3;11;3
a;b;c
#;_;_
_;_;_
_;#;_
aa
bb
cac
ab
cab
ca
a
ac
bba
aab
bab
```

On Campusnet you find this in `test01.CPC` and another file `test02.CPC`, more might be added later.

What you have to do:

a) Read and understand the problem.

b) Determine whether the answer for `test02.CPC` is YES or NO.

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

d) Show that CPC is $\mathcal{NP}$-complete. As reference problem you have to select a problem form the list of $\mathcal{NP}$-complete problems given below.

e) Implement a "decoder" which reads a CPC-file from standard input and checks whether it correctly codes a CPC instance. The output of this decoder must be the single-line output `YES` or `NO`. See below for submission instructions.

f) Find a heuristic algorithm which finds a solution for the problem. It has to construct a crossword puzzle in case of a YES and should always solve the problem in at most exponential time (but possibly faster under lucky circumstances). Describe in words how the algorithm works.

g) Bound the worst-case running time of the algorithm from above.

h) Implement the algorithm you developed in Part f). It has to able to read CPC files from standard input and solve the corresponding problem. If the answer is NO or the input is invalid, the algorithm has to output `NO`, and if the answer is YES, the output must be a solution to the problem described by a filled-out puzzle written down in the same format as lines 3 to $n + 2$ of the input file. For instance, the solution to the example problem above would be output as follows:

```
#;a;b
c;a;b
a;#;a
```

The programs developed in part f) and g) including the source code have to be submitted **both** via the usual Assignments module on Campusnet and via CodeJudge at `https://dtu.codejudge.net/02249-e18`. Expect that CodeJudge will run your code on both invalid and valid inputs and on both YES- and NO-instances.
You are encouraged to use CodeJudge also during test and development of your software and not only for submission of the final solution. Only the very last solution you submit will be graded by the teachers.
Accepted programming languages are Java, C++, C#, C and Python. Other languages have to be agreed upon with the teaching assistants.

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

If you have problems solving the assignment, you may use a simplified version where there are no "#" symbols in A. **However, this will reduce your grade by one point.** Some test files for the simpler version will be made available with the name `simpleXX.CPC`.

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

Problem: [Satisfiability]
Input: A set of clauses $C = \{c_1, \ldots, c_k\}$ over n boolean variables $x_1, \ldots, x_n$.
Output: YES if there is a satisfying assignment, i. e., if there is an assignment

$$a\colon \{x_1, \ldots, x_n\} \to \{\texttt{true}, \texttt{false}\}$$

such that every clause c_j is satisfied, 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 if there is a satisfying assignment such that every clause has exactly one true literal, i. e., if there is an assignment

$$a\colon \{x_1, \ldots, x_n\} \to \{\texttt{true}, \texttt{false}\}$$

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

Problem: [ExactThreeCover]
Input: The set $X = \{1, 2, \ldots, n\}$ where $n = 3q$, $q \geq 1$, and a collection $\mathcal{C} = \{C_1, \ldots, C_m\}$ of subsets of X of size 3. That is, for $i \in \{1, \ldots, m\}$ we have $C_i \subseteq X$ and $|C_i| = 3$.
Output: YES if there is a set $\mathcal{A} \subseteq \mathcal{C}$ such that the elements of $\mathcal{A}$ are a partition of X, i.e., $\cup_{C_i \in \mathcal{A}} = X$ and $C_i \cap C_j = \emptyset$ for $C_i, C_j \in \mathcal{A}$ with $i \neq j$. NO otherwise.

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 $j \in \{1, 2, \ldots, n\}$ it holds $\sum_{x \in X_j} x = B$.

Problem: [MinimumCliqueCover]
Input: An undirected graph $G = (V, E)$ and a natural number k.
Output: YES if there is a 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: [MinimumRectangleTiling]
Input: An $n \times n$ array A of non-negative integers, 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: [MinimumDegreeSpanningTree]
Input: An undirected graph $G = (V, E)$ and an integer $k > 0$.
Output: YES if there is a spanning tree T for G in which every node has degree at most k; 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 adds k new edges to E_2.

End of Exercise 1