Algorithmic Bioinformatics I: Exercises
Assignment 9

**Deadline:** Tuesday, 07.07.2009, 10 ct

**Exercise 1 (Simple Reductions):**
To further train the principles of reduction, show that the following problems are NP complete. Therefore, show that the given solution can be verified in polynomial time and that a known NP complete problem can be reduced to the new problem. I.e. describe a polynomial transformation of an instance of the known NP complete problem to an instance of the new problem. In this example, solutions are sets of sets. The instance size is the number of given sets.

(a) **X3C (Exact Cover by 3-Sets):** Given a set $X$ with $|X| = 3q$ and a collection $C$ of 3-element subsets of $X$. Is there a $C' \subseteq C$ which exactly covers $X$? I.e. is there a subset $C'$ of $C$ such that every element of $X$ is contained in exactly one set of $C'$? (*Hint:* 3DM is NP complete.)

(b) **Set Packing:** Given a collection $C$ of finite sets and a positive integer $K \leq |C|$. Does $C$ contain at least $K$ mutually disjoint sets?

**Exercise 2 (Bridge search):**
An edge in a connected, undirected graph is called *bridge* if the removal of this edge splits the graph in two unconnected parts. Develop an efficient algorithm which determines all bridges in a given graph. Analyze the runtime of this algorithm. The instance size is the number of nodes in the graph.

*Hint:* A simple solution is based on spanning trees. A more efficient solution is based on depth-first-search. Here, attention should be turned to the steps/timepoints nodes are reached by the DFS.

*Attention:* Algorithms that “enumerate all paths that fulfill $Y$” or “test all $X$ for $Y$” often have exponential runtime complexity. Check your algorithm for such or similar cases and if necessary show that there are reasons why your algorithm has only polynomial (or lower) complexity.
Exercise 3 (Original PARTITION):
Implement the pseudo polynomial algorithm for the PARTITION problem as given in the lecture.
I.e. create a table with entries $t(i, j) : i \in (1, \ldots, n), j \in (1, \ldots, \frac{B}{2})$ using dynamic programming.
The program should print TRUE if a partition is possible and FALSE else. Commandline parameter
is the path to a file. This file should contain the weights as positive integers, one per line.
Test your program with the given files. We expect that your program can not partition the weights
given in bigweights.txt. Determine the reason why this is the case. A statement like "Because Java
throws an error" is not sufficient.

Hint: Use Double.parseDouble(String s), to parse expressions like 10E5.

Exercise 4 (Improved(?) PARTITION):
Improve your implementation of the pseudo polynomial algorithm, such that it can partition big-
weights.txt or similar instances. Probably you don’t need to store the full $t(i, j)$ matrix.
Test your program with the given files. We expect that your program can not partition the weights
given in manyweights.txt. Determine the reason why this is the case. A statement like "Because Java
throws an error" is not sufficient.