Algorithms in the Real World (15-853), Fall 07
Assignment #5

This problem set covers both Linear/Integer programming and Computational Biology. You can look up material on the web and books, but you cannot look up solutions to the given problems. You can work in groups, but must write up the answers individually. Note that there are five problems on two pages. **Due Dec 6.**

Problem 1: Convex Programming (12pt)
Which of the following problems are NP-hard?

1. Linear Programming (LP)
2. Integer Linear Programming (ILP) — the constraints are linear constraints, but the variables are constrained to be integers.
3. Mixed Integer Programming (MIP) — only some of the variables are constrained to be integral, whereas the rest are allowed to take on fractional values.
4. Quadratic Programming (QP) — the constraints are quadratic constraints, but the variables are no longer constrained to be integers.

If you claim a problem is NP-hard, you must give a proof. These proofs should either involve reductions from **3SAT**, or from *problems you have already shown to be NP-hard.* (You might have seen the hardness of some of these problems in class, but you cannot use those since the reductions were from Knapsack, TSP, etc.)

Problem 2: Polytope Containment (15pt)
Let $S = \{x \in R^n : Ax \leq b\}$ and $T = \{x \in R^n : Cx \leq d\}$. Assuming that S and T are nonempty, describe an algorithm for checking whether $S \subset T$. (Your algorithm should run in time polynomial in the number of dimensions n, and the number of constraints m.)

Problem 3: Approximation Algorithms using Linear Programming (15pt)
By rounding a linear solution to an integer solution it is sometimes possible to get good approximations to various NP-hard problems (e.g. an answer that is within a constant factor of optimal). This is an approach that has been used extensively in the past 10 years to get good approximation bounds on various hard problems. Here is an example.
The *Set Cover* problem consists of a set of elements U and a family of subsets $\{S_1, S_2, \ldots, S_m\}$ with each $S_i \subseteq U$ and $\cup_i S_i = U$. Each set S_i has a cost C_i. We consider special instances of Set Cover, where each element is contained in at most f sets. This problem is still NP-hard, so we don't know how to solve the integer linear program formulation below in polynomial time.

$$\min \textstyle\sum_i C_i x_i \tag{1}$$

$$\text{subject to } \textstyle\sum_{i:j\in S_i} x_i \geq 1 \qquad \text{for all elements } j \in U \tag{2}$$

$$x_i \in \{0,1\}. \tag{3}$$

So instead, you solve the *LP relaxation* obtained by replacing the integrality constraints (3) by non-negativity constraints.

$$\min \textstyle\sum_i C_i x_i \tag{4}$$
$$\text{subject to } \textstyle\sum_{i:j\in S_i} x_i \geq 1 \qquad \text{for all elements } j \in U \tag{5}$$
$$x_i \geq 0. \tag{6}$$

Let Z_{IP} be the optimal value of the integer program, Z_{LP} be the optimal value of the LP relaxation (with the values of the variables in the LP being x_i^*).

1. Show that $Z_{LP} \leq Z_{IP}$.
2. Given a positive value τ, consider the following *rounding* operation: set $x_i' = 1$ if $x_i^* \geq \tau$, and $x_i' = 0$ otherwise. What is the largest value of τ (in terms of f) which will always ensure that x' is a feasible solution to the integer program.
3. Show that the cost of the solution $\sum_i C_i x_i' \leq (1/\tau) \times Z_{IP}$, and hence we have a set cover solution whose cost is at most $1/\tau$ times the cost of the optimal solution.

Problem 4: Interleaving Subsequences (10pt)
Given two strings S_1 and S_2 and a text T, you want to find whether there is an occurrence of S_1 and S_2 *interwoven* in T, possibly with spaces. E.g., the strings $S_1 =$ `banana` and $S_2 =$ `tantan` occur interwoven in $T =$`tabaranantangna` as follows:

```
  **  **     **
tabaranantangna
++      ++++
```

(Note that each letter has either a `*`, a `+`, or neither, but not both.) Give an efficient algorithm for this problem (i.e. one that is polynomial in the size of the inputs).

Problem 5: "Truncated Edit Costs" (10pt)
Consider the following edit-distance cost model: (a) each insertion or deletion costs one unit; (b) however, if there are more than k consecutive insertions, or k consecutive deletions, they cost only k units. Give an algorithm that finds the minimum edit distance under this cost model in time $O(nm)$. Note that the time should not depend on k. (Do not worry about space efficiency for this problem).