CS180 Winter 2011

Homework 8

Due: 9^{th} March

When submitting your homework, please include your name at the top of each page. If you submit multiple pages, *please staple them together*. We also ask that you do something to indicate which name is your last name on the first page, such as underlining it.

Please submit your solutions with the problems solved in the order they are given.

1. Consider a set $A = \{a_1, \ldots, a_n\}$ and a collection $B_1, B_2, \ldots B_m$ of subsets of A (i.e., $B_i \subseteq A$ for each i)

   We say that a set $H \subseteq A$ is a *hitting set* for the collection $B_1, B_2, \ldots B_m$ if H contains at least one element from each B_i – that is, if $H \cap B_i$ is not empty for each i.

   We now define the Hitting Set Problem[1] as follows. We are given a set A and a collection B of subsets of A, and a number k. We are asked: is there a hitting set $H \subseteq A$ for B so that the size of H is at most k?

   Prove that Hitting Set is **NP**-Complete.

2. In the Bipartite Directed Hamiltonian Cycle problem, we are given a *bipartite* directed graph $G = (V, E)$ and asked whether there is a simple cycle which visits every node exactly once. Note that this problem is potentially *easier* than Directed Hamiltonian Cycle because it assumes a bipartite graph. Prove that Bipartite Directed Hamiltonian Cycle is in fact **NP**-Complete.

3. Multiple Interval Scheduling[2] - We've seen the interval scheduling problem in Chapters 1 and 4. Here, we consider a computationally much harder version of it that we will call Multiple Interval Scheduling. As before, you have a processor that is available to run jobs over some period of time.
   People submit jobs to run on the processor; the processor can only work on one job at any single point in time. Jobs in this model, however, are more complicated than we've seen in the past: each job requires a *set* of intervals of time during which it needs to use the processor. Thus, for example, a single job could require the processor from 10AM to 11AM, and again from 2PM to 3PM. If you accept this job, it ties up your processor during those two hours, but you could still accept jobs that need any other time periods (including the hours from 11AM to 2PM).
   Now you're given a set of n jobs, each specified by a set of time intervals, and you want to answer the following question: For a given number k, is it possible to accept least k of the jobs so that no two of the accepted jobs have any overlap in time?
   Show that Multiple Interval Scheduling is **NP**-Complete.

4. A natural way to express the input to a clustering problem is via a set of objects $p_1, p_2, \ldots, p_n$, with a numerical distance $d(p_i, p_j)$ defined on each pair. We require that $d(p_i, p_i) = 0$, that $d(p_i, p_j) > 0$ for distinct p_i and p_j, and that the distances are symmetric. Given n objects,

[1] This is problem 8.5 from your textbook

[2] This is problem 8.14 from your textbook

with distances between them, and a bound B, we define the Low-Diameter Clustering problem as follows: can the objects be partitioned into k sets, so that no two points in the same set are at a distance greater than B from each other?

Prove that the Low-Diameter Clustering[3] problem is **NP**-Complete.

[3]This is problem 8.20 from your textbook