CSE 421 Intro to Algorithms Autumn 2017

Homework 8

Due Friday December 8, 4:00 pm

Problem 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 (so H "hits" all the sets B_i).

We now define the *Hitting Set Problem* as follows. We are given a set $A = \{a_1, \ldots, a_n\}$, a collection $B_1, B_2, \ldots, B_m$ of subsets of A, and a number k. We are asked: Is there a hitting set $H \subseteq A$ for $B_1, B_2, \ldots, B_m$ so that the size of H is at most k?

Prove that Hitting Set is NP-complete.

Problem 2:

Consider the problem of figuring out the cost of monitoring the nodes in a computer network represented by a graph $G = (V, E)$. A monitor that is placed at a node in the network can a monitor that node and all of the immediate neighbors of the node. For each node v in the network there is an integer cost c_v of placing a monitor at that node. The *Network Monitoring Problem (NMP)* is the problem of determining, given the graph G, the costs c_v for each node $v \in V$, and an integer K, whether or not there is a way to place monitors of total cost at most K so that every node in the network is monitored.

Prove that the Network Monitoring Problem is NP-complete.

Problem 3:

We've seen the Interval Scheduling Problem before, both with and without weights. Here we consider a computationally much harder version of it we'll call *Multiple Interval Scheduling*. As before, you have a processor that is available to run jobs over some period of time which we can think of as time 0 to T.

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 have seen in the past: Though they don't have weights, 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 times 6 to 8 and again from times 10 to 11. If you accept this job, it ties up the processor during those three time units, but you could still accept jobs that need any other time periods (including between times 8 and 10).

Now you're given a set of n jobs, each specified by a set of time intervals between 0 and T, and you want to answer the following question: For a given number k, is it possible to accept at least k of the jobs so that no two of the accepted jobs have any overlap in time?

Prove that Multiple Interval Scheduling is NP-complete.

Problem 4 (Extra Credit):

Given an undirected graph $G = (V, E)$, a *feedback set* is a set $X \subseteq V$ with the property that $G - X$ has no cycles. The *Undirected Feedback Set Problem* asks: Given G and k, does G contain a feedback set of size at most k? Prove that Undirected Feedback Set is NP-complete.