CS 330: Algorithms Spring 2016

Lab 12- Solution

Instructor: John Byers

Question 1. Suppose you are managing a system which has a set of n processes and a set of m resources. At any given point in time, each process specifies a set of resources that it requests to use and each resource can only be used by a single process at a time. If a process is allocated all of the resources it requests it becomes active, and otherwise it is blocked. Given a set of processes and resources, the set of requested resources for each process and a number k the Resource Reservation problem is to determine whether it is possible to allocate the resources so that at least k processes will be active. Recall that we have done parts (a) and (b) in lab last week.

(a) Give a polynomial time algorithm that solves the restriction of this problem to $k = 2$.

(b) Show that the Resource Reservation problem is in NP.

(c) Reduce the Independent Set problem to the Resource Reservation problem.

(d) Conclude from parts (b) and (c) that Resource Reservation is NP-Complete.

Solution:

(c) We can restate the general resource reservation problem as follows: we have a set of m resources and n processes, each of which requests a subset of the resources. The problem is to decide if there is a set of k processes whose requested resource sets are disjoint. Suppose we are given an instance of the independent set problem, specified by a graph G and a number k. We can create an equivalent resource reservation problem by letting the resources be the edges and the processes nodes of the graph. The process corresponding to node u requests the resources corresponding to the edges incident on u.

If there are k processes whose requested resources are disjoint, then the k nodes corresponding to these processes form an independent set. This is true because any edge between nodes would be a resource that they both request. If there is an independent set of size k, then the k processes corresponding to these nodes form a set of k processes that request disjoint sets of resources.

(d) Since we have shown that the resource reservation problem is in NP and we can reduce the independent set problem to it, we have that the resource reservation problem is NP-complete.

Question 2. Suppose you are planning a huge celebration for the end of the semester. You sat down and made a list of all your friends that you want to invite. However, you really want this party to stay drama free and you know that there are some people who have issues with others and will definitely be bringing down the overall mood of the celebrations. In order to avoid this issue, you decided to use your algorithms knowledge to help you out. You created nodes to represent all your friends and drew an edge between two people if they have any drama. The goal, then, is to find a set of k people in this graph G who are all connected with each other because that means that every single person in that group has drama with every other person in that group and you know not to invite those k people. You start trying to figure out an algorithm to find those k people only to realize that it is actually quite difficult. Oh no! You realize that this problem might just be NP-complete. Show that, indeed, the *clique problem* is NP-complete by (a) showing that it is in NP and (b) reducing independent set to clique.

Solution:

(a) We can see that it is in NP because, given an instance of $< G, k >$ and a k-clique, we are able to verify whether or not we actually have a k-clique in polynomial time. We simply need to ensure that those k nodes are all connected to each other. This can be done in about $O(n^2)$.

(b) Suppose we have an instance independent set, specified by a graph G and a number k. Given G, we will first construct the complement G'. We can observe that the nodes that are in the independent set of size k in G will form a k-clique in G'.

Question 3. Recall the Interval Scheduling Problem that we have seen before. Here, we will consider a computationally harder version of the problem that we'll call Multiple Interval Scheduling. As before, we 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 a bit more complicated: each job requries 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 then, again, from 2pm to 3pm. If this particular job is accepted, then it ties up the processor during those two hours, but a job that needs any other time period (e.g 11am to 2pm) can still be accepted.

We are given a set of n jobs, each specified by a set of time intervals and we 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? Show that the Multiple Interval Scheduling is NP-complete.

Solution:
The problem is in NP since, given a set of k intervals, we can check simply that none overlap. Suppose that we have an instance of independent set, specified by a graph G and a number k. Let us assume that there are n vertices and m edges where each node corresponds to a job and each edge corresponds to a time interval. Two nodes are the endpoints of an edge if two jobs require that time interval to run on the processor. We can observe that no two jobs can both be accepted if they are the endpoints of some edge. Thus, if there is an independent set of size k in G, then we are able to solve this scheduling problem.