Introduction to Algorithms	**Problem Set 8**
CS 4820, Spring 2018	**Due 11:59pm Thursday, March 29**

Hand in your solution electronically using CMS. Collaboration is encouraged while solving the problems, but:

1. **list the names of those with whom you collaborated;**
2. **you must write up the solutions in your own words.**

Remember that when a problem asks you to design an algorithm, you must also prove the algorithm's correctness and analyze its running time.

(1) *(8 points)* At Cornell University, doctoral students in Computer Science can earn their Ph.D. either in Ithaca or at the new Cornell Tech campus in New York City. However all students initiate their study in Ithaca and complete some course requirements before potentially moving to New York City. This inspires the following course scheduling problem.

A department has p professors and $2p$ courses. Each professor teaches two courses, one per semester. Assume that professors are numbered $0, \ldots, p-1$ and courses are numbered $0, \ldots, 2p-1$ such that professor i teaches courses $2i$ and $2i+1$, for each $i = 0, \ldots, p-1$. There are s students (numbered $0, \ldots, s-1$) each of whom is trying to finish their remaining course requirements in the fall semester so that they can move to a different campus in the spring. The remaining course requirements for student j are fulfilled by taking at least $n(j)$ courses from a specified list $L(j)$.

The Impatient Grad Students Problem (IGSP) is the following decision problem: given the numbers $p, s, n(0), \ldots, n(s-1)$ (in binary) and the lists $L(0), \ldots, L(s-1)$, decide whether the courses can partitioned into the fall and spring semesters such that

1. each professor teaches one fall course and one spring course
2. for each list $L(j)$, at least $n(j)$ of the courses in $L(j)$ are scheduled in the fall semester.

Prove that IGSP is NP-complete.

(2) *(10 = 2+8 points)* In the Paired Interval Scheduling problem one is given sets $S_1, \ldots, S_n \subset \mathbb{R}$, each of which is a union of two disjoint closed intervals with distinct integer endpoints. In other words, $S_i = [a_i, b_i] \cup [c_i, d_i]$ where $a_i < b_i < c_i < d_i$ are all integers. One is also given a positive integer $k \leq n$. The problem is to decide whether there exist k pairwise disjoint sets among the collection $\{S_1, \ldots, S_n\}$.

(a) *(2 points)* What is wrong with the following incorrect proof that Paired Interval Scheduling is NP-complete?

> Faced with an instance of the Paired Interval Scheduling problem, we can form a *conflict graph* whose vertex set is $\{1, \ldots, n\}$, with an edge joining i to j if S_i and S_j intersect. The problem asks whether this graph contains an independent set of size k. The Independent Set problem is NP-Complete, so the Paired Interval Scheduling problem is also NP-Complete.

(b) *(8 points)* Prove that Paired Interval Scheduling is NP-complete.

(3) *(12 = 2+10 points)* If G is a flow network and f is a flow in G, we say that f *saturates* an edge e if the flow value on that edge is equal to its capacity, i.e. $f(e) = c_e$. The FLOW SATURATION problem is the following decision problem: given a flow network G and a positive integer k, determine if there exists a flow f in G such that f saturates at least k edges of G. This problem asks you to prove that FLOW SATURATION is NP-complete.

(a) *(2 points)* What is wrong with the following incorrect solution?

> The FLOW SATURATION problem is in NP because it has a polynomial-time verifier that takes the pair (G, k) along with a flow f, and checks that f satisfies conservation and capacity constraints (in linear time). While checking capacity constraints it also keeps a counter of how many edges are saturated, and it reports "yes" if the conservation and capacity constraints are satisfied, and the number of saturated edges is at least k.
>
> To prove that FLOW SATURATION is NP-complete we reduce from HAMILTONIAN PATH. Given a directed graph G_0 that is an instance of HAMILTONIAN PATH, our reduction creates a new graph G consisting of a copy of G_0 together with four extra vertices $\{s, s', t', t\}$. G has two new edges (s, s') and (t', t), and it also has edges from s' to every vertex of G_0 and from every vertex of G_0 to t'. Finally we set all edge capacities to 1, and we treat this as an instance of FLOW SATURATION with source s, sink t, and parameter $k = n + 3$. The reduction takes linear time: if G_0 has n vertices and m edges, then G has $n+4$ vertices and $m+2n+2$ edges, and it takes constant time to insert each vertex or edge into the adjacency list representation of G.
>
> To prove the correctness of the reduction, we show the following two statements.
>
> **If G_0 has a Hamiltonian path, then G has a flow that saturates $n + 3$ edges.** Indeed, if the Hamiltonian path P_0 in G_0 is from s_0 to t_0, then G contains a path P of length $n+3$ from s to t, that begins with s, s', s_0, then traverses the entire path P_0 to reach t_0, then ends with t_0, t', t. Sending one unit of flow on this path P saturates all of its $n + 3$ edges.
>
> **If G has a flow that saturates $n + 3$ edges, then G_0 has a Hamiltonian path.** Indeed, since G has just a single unit-capacity edge leaving s, the max-flow value in G is equal to 1 and so a maximum flow consists of a single path from s to t. If this path saturates $n + 3$ edges, then two of its edges leave s and s', respectively, two of its edges come into t' and t, respectively, and the remaining $n - 1$ edges belong to G_0. Those $n - 1$ edges must form a Hamiltonian path in G_0.

(b) *(10 points)* Prove that the FLOW SATURATION problem is indeed NP-complete.

REMARK: Part (a) contains a valid proof that FLOW SATURATION **belongs to NP. Therefore, in doing part (b), you do not need to include that step in your solution.**