Due: Wed, November 2, beginning of lecture

NOTE: Each problem set only counts 5% of your mark, but it is important to do your own work (but see below). Similar questions will appear on the next term test. You may consult with others concerning the general approach for solving problems on assignments, but you must write up all solutions entirely on your own. Anything else is *plagiarism*, and is subject to the University's Code of Behavior. You will receive 1/5 points for any (non bonus) question/subquestion for which you say "I do not know how to answer this question". You will receive .5/5 points if you just leave the question blank.

1. (20 points)

   Consider the following variant of the knapsack problem. As in the standard knapsack problem, the input is $\{(v_1, w_1), \ldots, (v_n, w_n); W\}$ where v_i (respectively, w_i) is the value (resp. size) of the ith item and W is the total size limit of the knapsack. All input parameters are positive integers. Now (in this variant) a feasible solution is a non-negative integer vector $(m_1, \ldots, m_n)$ such that $m_i \leq 3$ for all i and $\sum_i m_i w_i \leq W$. We want to maximize $\sum_i m_i^2 v_i$ for a feasible solution. That is, items that are used more often increase the value of the solution. We wish to derive a polynomial time DP algorithm for computing the value of an optimal solution to this knapsack problem when the value parameters satisfy $v_i \leq n^2$ for all i. That is,

   - provide a semantic array for the problem and indicate how the desired answer is obtained from this array.

   - provide an equivalent recursively defined computational array (including any base cases).

   - briefly indicate why your computationally defined array is equivalent to the semantic array given.

   - What is the time complexity of your algorithm assuming that $(\forall i) v_i \leq n^2$?

2. (20 points)

   Consider the following triangulation problem. We are given a set of $n \geq 3$ points $x_1, x_2, \ldots, x_n$ in the plane which are the vertices (in clockwise order) of a convex polygon P in the plane. We wish to triangulate the polygon P so as to minimize the sum of the lengths of the $n-3$ interior edges used to form the triangulation. Provide a polynomial time dynamic programming algorithm with complexity polynomial in n for computing an optimal triangulation. Provide appropriate semantic and computational (i.e. recursively defined) arrays and justify why these are equivalent. What is the complexity of your algorithm assuming that one can calculate the length of a line in constant time?
   Hint: You may find it useful to consider a dynamic program in the style used for the matrix chain problem.

3. (10 points)

   For each of the following statements about integral capacity flow networks, indicate if the statement is true or false. If true justify why and if false give a counter-example.

   - There is always a *unique* minimum capacity cut.
   - It is always possible to reduce a positive maximum flow by one unit (of flow) by decreasing the capacity of some edge by one unit.

4. (20 points) Suppose we are given a flow network $\mathcal{F}$ with integer edge capacities and we are also given an integral max flow f in $\mathcal{F}$. Suppose we want to increase the max flow by one unit by increasing the capacity of certain edges. Show how to efficiently compute a smallest set of edges E' such that increasing the edge capacity for each edge $e \in E'$ will increase the max flow by 1 unit.
   Hint: Your method should take time proportional to Dijkstra's shortest path algorithm, say $O(m \log n)$ where m is the number of edges in the network and n is the number of nodes.

5. (20 points)

   Consider the m-machine makespan problem for the restricted machines model. In this model, a job J_i is specified by a pair (p_i, S_i) where p_i is the processing time of the job and $S_i \subseteq \{1, \ldots, m\}$ is the set of allowable machine for this job. That is, job J_i can only be scheduled on a machine whose index is in S_i. Recall that the goal of the makespan problem is to minimize the latest (over all machines) finishing time. Suppose that all, jobs have unit processing time (i.e. $p_i = 1$ for all i). Show how to reduce the makespan problem for this special case to a flow problem.
   Hint 1: Use a max flow algorithm to determine whether or not a makespan with a particular value can be achieved. Hint 2: This is somewhat similar to the max bipartite matching problem.

6. (20 points)

   Show how to polynomial time reduce the following optimization problems to the associated decision problem.

   (a) Reduce the max clique optimization problem (.e. given a graph G, find a maximum size clique in G) to the decision problem CLIQUE (i.e. given (G, k) does G have a clique of size k).

   (b) Reduce the min colouring optimization problem (i.e. given a graph G, find a minimum colouring of the graph) to the decision problem COLOUR (.e. given (G, k) does G have a colouring using k colours). Here you are to compute the actual colouring (i.e. the function mapping each node to a colour) and not just the value of the minimum colouring).