Due: Wed, October 5, 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 first 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.
Advice: Do NOT spend an excessive amount of time on any question and especially not on a bonus question. If you wish to spend "free time" thinking about (say) bonus questions that is fine but you should not sacrifice time needed for other courses.

1. (15 points)

   Consider the following greedy algorithm for graph colouring. Without loss of generality we will let the input $G = (V, E)$ be a connected graph with $V = \{v_1, v_2, \ldots, v_n\}$ ($n \geq 1$) and let the colours be $C = \{1, 2, \ldots\}$. We also let $Nbhd(v)$ denote the neighbourhood (i.e. adjacent vertices) of node v.

   GREEDY COLOURING ALGORITHM (using breadth first search)

   Let $\chi(v_1) = 1; L(0) := \{v_1\}; i := 0$ % $\chi()$ is the colouring function
   % $L(i)$ will denote the nodes in the current level of the breadth first search
   Let $A := \{v_1\}$ % A will be the nodes already coloured
   Let $U := V - \{v_1\}$ % U will be the nodes not yet coloured
   While $U \neq \emptyset$
   $L(i+1) := \emptyset$
   For $j := 1..n : v_j \in L(i)$
   For $k = 1..n : v_k \in Nbhd(v_j) \cap U$
   % colour the node being added to the next level
   $\chi(v_k) := min_{c \in C} : c \notin \cup_{v_h \in Nbhd(v_k) \cap A} \chi(v_h)$
   $U := U - \{v_k\}; A := A \cup \{v_k\}$
   $L(i+1) := L(i+1) \cup \{v_k\}$
   END For
   END For
   $i := i + 1$
   END While

   (a) (10 points) Give a short but convincing argument showing that the above greedy algorithm will colour every 2-colourable graph using 2 colours; that is, the greedy algorithm is optimal for 2-colourable graphs.

   (b) (5 points) Give an example of a 3-colourable graph for which the above greedy algorithm will use more than 3 colours.

2. Consider the following scheduling problem. An input is a set of jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$ where each job J_i is described by a triple of non-negative real numbers r_i, d_i, p_i where r_i (respectively d_i, p_i) is the release time (resp. the deadline, the processing time) of job J_i. A job that is scheduled cannot start sooner than its release time r_i and must end before its deadline d_i and since it requires p_i time to process, the time $\sigma(i)$ that the job is scheduled to begin must satisfy $r_i \leq \sigma(i) < d_i - p_i$. The goal is to schedule as many jobs as possible without scheduled jobs intersecting.
Consider the following "earliest completion time" greedy algorithm:

$D := 0$ % D is the current finishing time of the last scheduled job
$S := \emptyset$ S is the set of jobs accepted thus far
$Y := \mathcal{J}$ % Y is the set of jobs not yet considered.
While $Y \neq \emptyset$
  Let $j = argmin_{k:J_k \in Y}[\max(r_k, D) + p_k]$ (breaking ties in any defined way)
  $Y := Y - \{J_j\}$ %Only consider a job once
  If $\max(r_j, D) + p_j < d_j$
    then $D := \max(r_j, D) + p_j$;
      $S := S \cup \{J_j\}$ J_j is added to the solution
  End If
END While

(a) (10 points) Give a charging argument showing that the above greedy algorithm is a 2-approximation algorithm; that is, the number of jobs scheduled in an optimal solution is no more than twice the number of jobs scheduled by the greedy algorithm.

(b) (5 points) Give an example of a set of jobs for which the above greedy algorithm will not produce an optimal schedule.

3. (15 points) We wish redefine the *cost of a path* in various ways and then see if Dijkstra's shortest path algorithm will still optimally solve the least cost paths problem. For each of the following definitions of the cost of a path, state and justify whether or not Dijkstra's algorithm optimally solves the least cost problem. We assume a non negative cost $c(e)$ for each edge e in the graph. If Dijkstra's algorithm is not optimal then show a counter example. If Dijkstra's algorithm is still optimal then say what is the key observation in the proof that still holds.

(a) (5 points) The cost of a path π is $\max_{e \in \pi} c(e)$

(b) (5 points) The cost of a path π is $\min_{e \in \pi} c(e)$

(c) (5 points) The cost of a path π is the average cost of an edge in π; that is, $cost(\pi) = \frac{\sum_{e \in \pi} c(e)}{|e : e \in \pi|}$

4. (15 points) Consider the following k-median problem for the real line. The input is a set X of n points $x_1, \ldots, x_n$ on the real line and without loss of generality let us say $x_1 < x_2 < \ldots < x_n$. We are also given an integer parameter k, $1 \leq k \leq n$. The goal is to select select k points $Y \subseteq X$ so as to minimize $\sum_{i=1}^{n} \min_{y \in Y} |x_i - y|$. Provide a dynamic programming (DP) algorithm for this problem.

   - (5 points) Provide a semantic array definition for computing the cost of an optimal solution.
     Hint: Think of the solution as being a clustering of the points into k groups where each group is a consecutive sequence of points.
   - (5 points) Provide a recursive (computational) definition for computing values of this array and briefly justify why your computational definition is equivalent to the semantic definition.
   - (5 points) What is the asymptotic complexity of your algorithm in terms of the number of arithmetic operations and comparisons as a function of n and k?