Due: Thursday, April 2, 2020 4:59PM on MarkUs

Note: You will receive 20% of the points for any (sub)problem for which you write "I do not know how to answer this question." You will receive 10% if you leave a question blank. If instead you submit irrelevant or erroneous answers you will receive 0 points. You will receive partial credit for the work that is clearly "on the right track."
You may choose to spend your time looking for solutions on the internet and may likely succeed in doing so but you probably won't understand the concepts that way and will then not do well on the quizzes, midterm and final. So at the very least try to do the assignment initially without searching the internet. If you obtain a solution directly from the internet, you must cite the link and explain the solution in your own words to avoid plagarizing.

Note: As we have to change the grading scheme, it is especially important that you work individually (or in your established teams for this assignment) and not take solutions from others. This is always important but if we are going to increase the weight of assignments, we clearly have to have confidence that your work is your work.

There was an error in the way we were weighting question 3, so we are going to reweight all the questions to better reflect the time needed and importance of the questions.

1. (10 points)

   We want to consider the following decision problem called DENSESUBGRAPH:
   Given a graph $G = (V, E)$ and two integers a, b, does there exist $V' \subseteq V$ such that $|V'| = a$ and $|(V' \times V') \cap E| \geq b$. That is, the problem is to determine if there is a subset of vertices of size a such that the subgraph induced by V' has at least b edges.

   Prove that DENSESUBGRAPH is NP complete.

2. (10 pts)

   Suppose $P = NP$. Show how to solve the graph coloring problem in polynomial time. That is, given a graph $G = (V, E)$, find a valid coloring $\chi(G) : V \to \{1, 2, \ldots, k\}$ for some k satisfying the property that $(u, v) \in E$ implies $\chi(u) \neq \chi(v)$ so as to minimize the fewest number k of "colors".

3. (25 points) The standard form for an integer program is

$$\begin{array}{lrl} \text{Maximize} & c^T x & \\ \text{Subject to} & Ax & \leq b \\ & x & \in \mathbb{N}^n \end{array}$$

Using the matrix notation, you can define the dual of an IP in standard form in the same way as for an LP in standard form.

- (5 points) Show that the statement of weak duality holds for an IP.
- Consider the following primal problem expressed as an IP and also as an LP .

  maiximize y
  subject to $y - \frac{3}{2}x \leq 0$
  $y + \frac{3}{2}x \leq 3$
  $y, x \in \mathbb{N}$ for the IP and $y, x \geq 0$ for the LP

  (a) (5 points) Draw the feasible region and determine the optimum integral solution and the optimum fractional solution.

  (b) (5 points) Provide the dual of this primal and verify that the optimum fractional value for the dual matches the optimum fractional value for the primal.

  (c) (5 points) Determine an integral value for the dual that is greater than the integral optimum for the primal thereby showing that strong duality does not generally hold for IP.

4. (20 points) Consider the weighted set cover problem as defined in the KT text chapter on approximation algorithms. Namely, the input is a collection of $\mathcal{C} = \{S_1, S_2, \ldots, S_n\}$ where $S_i \subset U$, $\sum_i S_i = U$ and w_i is the weight of set S_i. The objective is to select a subcollection $\mathcal{C}'$ covering the elements in U so as to minimize $\sum_{S_i \in \mathcal{C}'} w_i$.

   Now suppose that we restrict attention to those inputs (which we will call *frequency f instances*) where each universe element occurs in at most f sets.

   (a) (10 points)
   Show that the vertex cover problem can be viewed as a a set cover problem where every input is a frequency 2 instance.

   (b) (10 points) Show how to represent the set cover problem as an integer programming problem. Then show how to use linear programming and rounding to derive a factor f approximation algorithm for every frequency f instance of the set cover problem.

5. (20 points) Consider the knapsack problem with inputs $\{(s_1, v_1), \ldots, (s_n, v_n)\}; W\}$. We know that there is an algorithm (using dynamic programming and rounding) that uses time $O(n^3/\epsilon)$ time and computes a solution within a factor $(1+\epsilon)$ of the optimal solution for the (general) knapsack problem. Now consider the simple knapsack problem where $v_i = s_i$ for all i. We want a faster approximation algorithm for this problem.

(a) (10 points) Describe a $O(n \log n)$ greedy time algorithm *Greedy* for the simple knapsack problem that always produce a solution within a factor of 2 of the optimal solution; that is, $Greedy(\mathcal{I}) \geq (1/2)OPT(\mathcal{I})$ for every input $\mathcal{I} = (s_1, s_2, \ldots, s_n; W)$. Here we assume $s_i \leq W$ for all i.

Give a brief argument as to why your solution obtains the stated approximation guarantee.

(b) (10 points) Describe an $O(n \log n)$ time algorithm ALG such that $ALG(\mathcal{I}) \geq (2/3)OPT(\mathcal{I})$ for every input $\mathcal{I}$.
Hint: Partition the inputs into three sets, those with weight $w_i \leq W/3$, those with weight $W/3 < w_i \leq (2/3)W$ and those with weight $w_i > (2/3)W$. Your algorithm should be "greedy-like" in the sense that in each iteration it will consider a couple of items and then decide what items to place in the knapsack.

Give a brief argument as to why your solution obtains the stated approximation guarantee.

6. (20 points) Comsider the following weighted set packing problem. The input is collection $\mathcal{C} = \{S_1, S_2, \ldots, S_m\}$ of sets where $|S_i| \leq k$ for some fixed integer k and w_i is the weight of S_i. The objective is to select a subcollection $\mathcal{C}'$ of disjoint sets so as to maximize $\sum_{S_i \in \mathcal{C}'} w_i$.

Design a greedy algorithm *Greedy* that achieves a k approximation ratio. That is, for every input $\mathcal{C}$, $Greedy(\mathcal{C}) \geq \frac{1}{k}OPT(\mathcal{C})$. Use a charging argument to show that your algorithm obtains the stated approximation.