Practice Problems for the final [1]

1. Let X denote the decision version of the maximum independent set problem. Thus the input consists of an undirected graph $G = (V, E)$ (with no self-loops or parallel edges) and an integer B. The associated question is "Does G contain an independent set of size at least B?" (Recall that an independent set of G is a subset U of V such that no two vertices in U are connected by an edge in E.)

Let the decision problem Y be defined as follows. The input consists of a set S, a collection $A_1, \ldots, A_n$ of subsets of S, a positive integer weight w_i for each set A_i, and a positive integer B. The associated question is "Is there a subset I of $\{1, \ldots, n\}$ such that (1) for all distinct integers i and j in I, the sets A_i and A_j are disjoint, and (2) $\sum_{i \in I} w_i \geq B$?"

Prove that if there is a polynomial-time algorithm for solving decision problem Y, then there is a polynomial-time algorithm for solving decision problem X.

2. Let X denote the decision version of the knapsack problem. Thus the input to X consists of a collection of n items indexed from 1 to n, where item i has a positive integer value v_i and a positive integer weight w_i, and positive integers W and B. The associated question is "Is there a subset I of $\{1, \ldots, n\}$ such that $\sum_{i \in I} w_i \leq W$ and $\sum_{i \in I} v_i \geq B$?"

Let decision problem Y refer to the special case of decision problem X in which the two positive integers W and B are required to be equal.

Prove that if there is a polynomial-time algorithm for solving decision problem Y, then there is a polynomial-time algorithm for solving decision problem X.

3. Let S denote the set of all languages over the alphabet $\{a\}$. (Recall that a language over the alphabet $\{a\}$ is a subset of $a^* = \{\epsilon, a, aa, \ldots\}$.) *Use a diagonalization argument* to prove that S is uncountable. (Up to four points of partial credit will be given for a solution that is not based on diagonalization.)

4. Let decision problem X refer to the problem of determining whether a given TM halts when it is executed on the empty string (i.e., when the input string is the empty string). Let decision problem Y refer to the problem of determining whether a given TM ever writes the symbol "a" (at any step of the execution, on any cell of the tape) when it is executed on the empty string. Prove that if Y is decidable, then X is decidable. (Remark: Since X is known to be undecidable, this result implies that Y is undecidable.)

5. Recall the NP-hard load balancing problem that we discussed in the lectures: We are given n items with positive integer weights $w_1, \ldots, w_n$, a collections of k bins for some given positive integer k, and our objective is to distribute the n items across the k bins in such a way that the maximum load of any bin is minimized. Let $\mathcal{A}$ denote the first greedy algorithm that we discussed for this problem: The items are processed in the order of their indices, and an item is processed by adding it to a bin with minimum load.

[1] Questions provided by Prof. Greg Plaxton

(a) In class we proved that algorithm $\mathcal{A}$ finds an assignment of items to bins for which the maximum load of any bin is at most twice the optimal value. Prove that there exist instances for which the approximation ratio achieved by algorithm $\mathcal{A}$ is higher than 1.99.

(b) Suppose that the maximum weight of any item is 100, the number of bins k is at most 10, and the sum of the item weights is at least 5000. Prove that under these assumptions, algorithm $\mathcal{A}$ achieves an approximation ratio of at most $\frac{6}{5}$.

6. Recall that the input to the weighted set cover problem is a set S, a collections of subsets $A_1, \ldots, A_n$ of S such that $\cup_{1 \leq i \leq n} A_i = S$, and a positive integer weight w_i for each set A_i. A set cover corresponds to a subset I of $\{1, \ldots, n\}$ such that $\cup_{i \in I} A_i = S$. The objective is to determine a set cover of minimum total weight (i.e., minimizing $\sum_{i \in I} w_i$). In class we presented a greedy algorithm for computing a set cover with weight at most $O(\log |S|)$ time optimal. In analyzing this algorithm, we used a charging scheme to assign a specific cost to each element of S, where the sum of these costs is equal to the weight of the greedy set cover. Assume that $|A_1| = 4$ and that $w_1 = 120$. Let C_1 denote the total cost assigned to the four elements of A_1. Explain why C_1 cannot exceed 250.

7. The following questions are related to the lecture material on randomized algorithms. In each of these questions, if your final answer is correct, you will receive full credit. (You might want to show your work in order to have a better chance of receiving partial credit in case your final answer is incorrect.)

(a) Each time Joe reloads a certain web page, he is shown an ad chosen uniformly at random (and with replacement) from a set of n ads $A = \{a_1, \ldots, a_n\}$. Suppose that Joe repeatedly reloads the page until he sees ad a_1. Let the random variable X denote the number of page reloads. Give an exact expression (depending on n) for the expectation of X.

(b) Repeat part (a), but now suppose that Joe repeatedly reloads the page until each of the n ads has been shown at least twice. Let the random variable X denote the number of page reloads. Give a Θ-bound (in terms of the parameter n) for the expectation of X.

(c) Let $G = (V, E)$ be an undirected graph (with no self-loops or parallel edges). Assume that each each vertex of the graph is assigned a color chosen independently and uniformly at random from a set of three colors. Let the random variable X denote the number of edges e in E such that both endpoints of e are assigned the same color. Give an exact expression for the expecation of X.

(d) An urn contains n red balls and n blue balls for some positive integer n. Joe reaches into the urn and pulls out two (randomly chosen) balls. Give an exact expression for the probability that exactly one of the two balls is red.

(e) Assume that the randomized Quicksort algorithm discussed in the lecture is used to sort n distinct keys, where $n \geq 2$. Give an exact expression for the probability that the minimum key is compared to the maximum key at some point during the execution.