CS7800: Advanced Algorithms. Fall 2016
Homework 2

Instructor: Jonathan Ullman, TA: Mehraneh Liaee

Due Friday, October 7 at 11:59pm
(Email to m.liaee2050+CS7800@gmail.com)

- You must type your solutions using LaTeX. Please submit both the source and PDF files using the naming conventions `lastname_hw2.tex` and `lastname_hw2.pdf`.
- Strive for clarity and conciseness in your solutions, emphasizing the main ideas over low-level details. I recommend looking at the introduction in Jeff Erickson's textbook for advice on writing up solutions to algorithms problems.
- Do not share written solutions, and remember to cite all collaborators and sources of ideas. Sharing written solutions, and getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.

Problem 1 (Too Many Horses, 15 points)**.** You are cleaning out your little brother's room and insist that he has too many My Little Pony™ dolls. Specifically, you believe that out of his n dolls, at least $\lceil n/2 \rceil$ of them are the same. However, not being a My Little Pony™ expert, you cannot actually distinguish one doll from the other. All you can do is show your little brother two of the dolls and ask him to tell you if they are the same.

You could ask him about all $\binom{n}{2} = \Theta(n^2)$ pairs, but if you do so, he will get bored and stop cooperating. Design an $O(n \log(n))$ time divide-and-conquer algorithm to determine whether or not at least $\lceil n/2 \rceil$ out of the n dolls are the same. You may use a subroutine AREEQUAL(i, j) to check if doll i and doll j are the same in $O(1)$ time. Clearly describe your algorithm, prove its correctness, and analyze its running time.

Problem 2 (Cartesian Sums, 20 points)**.** Given two sets of integers A and B, we define their *sum* $A + B$ to be

$$\{a + b \mid a \in A, b \in B\}.$$

For this problem you may assume that it takes $O(1)$ time to add or compare any two numbers $a \in A, b \in B$.

(a) Design a simple $O(n^2 \log(n))$ time algorithm that takes two sets A, B of size n and computes $A + B$.[1] Clearly describe the algorithm, prove that it is correct, and prove that it achieves the desired running time.

(b) Suppose we further require that $A, B \subseteq \{1, 2, \ldots, m\}$ for some parameter m. Use the Fast Fourier Transform to design an $O(m \log(m))$ time algorithm that computes $A + B$. This algorithm is faster than the previous algorithm whenever $m = o(n^2)$.

Problem 3 (More On Minimum Spanning Trees, 25 points)**.** In this problem we will see how the structure underlying MST algorithms (i.e. the cut and cycle properties) can be used to say a lot more about minimum spanning trees. In all the following questions, let $G = (V, E)$ be a graph with edge-weights $\{w_e\}_{e \in E}$ and assume that all edge-weights are distinct.

(a) Prove that, because all edge weights are distinct, G has a *unique* minimum spanning tree.

(b) Let T be the minimum spanning tree of G and let T' be the *second minimum spanning tree*. That is, T' is the minimum-weight spanning tree that is not identical to T. Prove that T and T' differ on exactly one edge.

(c) Using part **(b)**, design an efficient algorithm that finds the second minimum spanning tree. Clearly describe the algorithm, prove that it is correct, and analyze its running time. For full credit your algorithm should run in time at most $O(nm)$, but slightly slower algorithms will receive most of the points.

[1] Hint: $|A + B|$ is *not* always n^2!

Problem 4 (Maximum Weight Independent Set on Grids, 20 points)**.** Given a graph $G = (V, E)$ with node weights $\{w_v\}_{v \in V}$, the *maximum-weight independent set problem* asks for a subset $I \subseteq V$ such that 1) for every $u, v \in I$, $(u, v) \notin E$, and 2) $\sum_{v \in I} w_v$ is maximized. As we will learn later, this problem is NP-complete and is unlikely to have a polynomial time algorithm or even an algorithm that "substantially" beats the trivial $O(n2^n)$ time algorithm that tries all subsets of V. In this problem we will see that maximum-weight independent set can be solved more efficiently on a special type of graph called a *grid graph*.

A grid graph $GRID_{w,h}$ of width w and height h is a graph with $n = wh$ vertices laid out in a $w \times h$ grid. Each vertex is identified by a pair of integers $(x, y) \in \{1, \ldots, w\} \times \{1, \ldots, h\}$ and two vertices (x_1, y_1) and (x_2, y_2) are connected by an edge if and only if $|x_1 - x_2| + |y_1 - y_2| = 1$.

(a) Show that in any grid graph with $n = wh$ vertices, there exists a set of $\sqrt{n}$ vertices such that removing those vertices leaves two grid subgraphs that are not connected to one another and have at most $n/2$ vertices each. We will call such a set a *separator*.

(b) Show that the recurrence $T(n) = 2^{\sqrt{n}} \times (T(n/2) + 1)$, where $T(n) = 1$, satisfies $T(n) = 2^{O(\sqrt{n})}$. If you can, find the specific constant c such that $T(n) = \Theta(2^{c\sqrt{n}})$.

(c) Prove that maximum-weight independent set can be solved on any grid graph $GRID_{w,h}$ in time $2^{O(\sqrt{n})}$—much faster than the trivial algorithm.[2]

Problem 5 (Makin' Change, 20 points)**.** In a previous life, you worked as a cashier on Caprica, spending the better part of your day giving change to your customers. Because paper is a very rare and valuable resource on Caprica, cashiers were required by law to use the fewest bills possible whenever they gave change. Thanks to the numerological predilections of the gods of Caprica, the currency of Caprica, the Centon, was available in the following denominations: \$1, \$4, \$7, \$13, \$28, \$52, \$91, \$365.

(a) The greedy change algorithm repeatedly takes the largest bill that does not exceed the target amount. For example, to make \$122 using the greedy algorithm, we first take a \$91 bill, then a \$28 bill, and finally three \$1 bills. Give an example where this greedy algorithm uses more bills than the minimum possible.

(b) Design a dynamic programming algorithm that computes, given an integer k, the minimum number of bills needed to make k Centons in time polynomial in k. Clearly state your algorithm, prove its correctness, and analyze its running time.

[2] Hint: Use divide-and-conquer. How many ways can the max-weight independent set intersect the separator?