CMSC451 Homework 1
Spring 2017 due February 9, 2017 at 3pm EST

For all algorithms, provide time complexity analysis as well as a formal proof of correctness. Homework solutions should be clearly written and electronically submitted via ELMS (`http://elms.umd.edu`) by the due date listed above.

1. Suppose f, g are two functions mapping positive real numbers to positive real numbers and $f = O(g)$. Which of the following statements must be true?

   (a) $\log_2 f = O(\log_2 g)$

   (b) $\sqrt{f} = O(f)$

   (c) $f^k + 100f^{k-1} = O(g^k)$, for $k \geq 1$

2. Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer.

   (a) Describe a greedy algorithm to make change consisting of quarters, dimes, nickels and pennies. (Don't forget to prove correctness!)

   (b) Suppose that the available coins are in the denominations that are powers of c, i.e., the denominations are $c^0, c^1, \ldots, c^k$ for some integer $c > 1$ and $k \geq 1$. Show that the greedy algorithm always yields an optimal solution.

   (c) Give a set of coin denominations and an n for which the greedy algorithm does not yield an optimal solution. Your set should include a penny so that there is a solution for every value of n.

3. Suppose you have n children and $\frac{n}{2}$ hospital rooms, each room having two beds. Each child has a preference list on the other $n-1$ children as potential roommates and you want to find a stable assignment of children to children so that no two children have incentive to ditch their roommates for each other. Does a stable assignment always exist? Prove or find a counterexample. Assume n is even.

4. Suppose you have n input wires and n output wires, each directed from a source to a terminus. Each input wire meets each output wire in exactly one distinct point, at a special piece of hardware called a *junction box*. Points on the wire are naturally ordered in the direction from source to terminus; for two distinct points x and y on the same wire, we say that x is *upstream* from y if x is closer to the source than y, and otherwise we say x is *downstream* from y. The order in which one input wire meets the output wires in not necessarily the same as the order in which another input wire meets the output wires. (And similarly for the orders in which output wires meet input wires.)

   Each input wire is carrying a distinct data stream, and this data stream must be *switched* onto one of the output wires. If the stream of Input i is switched onto Output j at junction box B, then this stream passes through all the junction boxes upstream of B on Input i and all the junction boxes downstream from B on Output j.

It does not matter which input data stream gets switched onto which output wire, but each input data stream must be switched onto a *different* output wire. Furthermore, no two data streams can pass through the same junction box following the switching operation. See Figure 1.

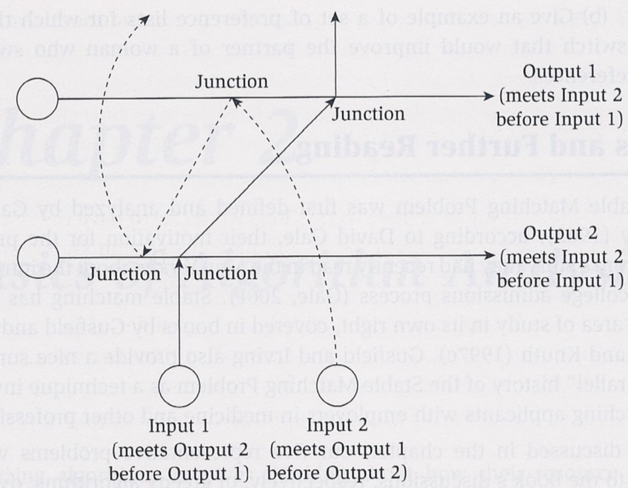

Figure 1.8 An example with two input wires and two output wires. Input 1 has its junction with Output 2 upstream from its junction with Output 1; Input 2 has its junction with Output 1 upstream from its junction with Output 2. A valid solution is to switch the data stream of Input 1 onto Output 2, and the data stream of Input 2 onto Output 1. On the other hand, if the stream of Input 1 were switched onto Output 1, and the stream of Input 2 were switched onto Output 2, then both streams would pass through the junction box at the meeting of Input 1 and Output 2—and this is not allowed.

Figure 1: From *Algorithm Design*, by Kleinberg and Tardos.

Show that for any specified pattern in which the input wires and output wires meet each other (each pair meeting exactly once), a valid switching of the data streams can always be found – one in which each input data stream is switched onto a different output, and no two of the resulting streams pass through the same junction box. Additionally, give an algorithm to find such a valid switching and prove its correctness.

5. Suppose you have n unit length jobs, where each job j has release time 0, deadline d_j and profit p_j. Design an $O(n \lg n)$ algorithm to schedule jobs onto a single machine, so as to maximize the cumulative profit over scheduled jobs. Explain any data structures needed to implement your algorithm: initialization, the operations needed by the algorithm, and each operation's running time.