CSCI 533 Analysis of Algorithms
Exam #2

This is a take-home test. You may consult your textbook, but no other sources. You may not talk or work with other students on these problems. If you have questions, please come see me or e-mail me.

Due date: TBA

1. **Disjoint Set.** Suppose you implement the disjoint set data structure as given in the textbook, altered so that it uses union-by-rank, but not path compression. Give a sequence of \( m \) union and find operations on \( n \) elements that take \( \Omega(m \log n) \) time. You can give the general idea where you argue the time complexity, and then give an example.

2. **Amortized Analysis.** Consider the following “trick” sometimes used in hashing when a hash table becomes too full. Assume the hash table has \( n \) slots. When the table becomes full, a new hash table is created that is twice the size of the previous table. The values in the original table are rehashed and placed into the new table (half filling it). Of course, as more insertions are performed, the doubling of the size of the table and rehashing may be required many times. Assume a typical insertion takes \( \Theta(1) \) time. When rehashing is required, the time is \( n+1 \). That is, it takes \( n \) time to rehash the full table of size \( n \), plus one for the insertion of the new item that caused the overflow. Show, using any of the amortization techniques we considered, that the insertion operation can be performed in \( \Theta(1) \) amortized time.

3. **Greedy.** Consider the problem of optimally deciding where to put line breaks when formatting text with the goal of minimizing the number of lines used in the layout. Let \( M \) be the maximum number of characters that can be placed on a line. The input is the value of \( M \) and a sequence of \( n \) words \( w_1, w_2, \ldots, w_n \) of lengths \( t_1, t_2, \ldots, t_n \), measured in characters where \( t_i \leq M \) for all \( i \). Words are not to be split between lines, and are to be output in the order given.
   
   (a) Give a greedy algorithm that optimally solves this problem. Describe the algorithm as clearly and succinctly as possible. What is the complexity of the algorithm?
   
   (b) Prove that your greedy algorithm will minimize the number of lines in the layout.

4. **Dynamic Programming.** Consider the following game. A dealer produces a sequence \( s_1, s_2, \ldots, s_n \) of cards face up, where each card \( s_i \) has a value \( v_i \). Then two players take turns picking a card from the sequence, but can only pick the first or last card of the remaining sequence. The goal is to collect cards of the highest total value. For example, you can think of the cards as bills of different denominations. Assume \( n \) is even.
   
   (a) Show a sequence of cards such that it is not optimal for the first player to start by picking up the available card of larger value. That is, the natural greedy strategy is suboptimal.
   
   (b) Give an \( O(n^2) \) algorithm to compute an optimal strategy for the first player. Given the initial sequence, your algorithm should precompute in \( O(n^2) \) time some information, and then the first player should be able to make each move optimally in \( O(1) \) time by looking up the precomputed information.