Homework 2

Henry Z. Lo

Due: Tuesday, June 24 in class

This homework assignment will review data structures and algorithms that we've studied. For these problems, write pseudo-code; your algorithms should contain enough detail to implement. Try to solve these problems on your own, but if you look up help, be sure to acknowledge your source and its contributions.

1. Hash tables:

   (a) Given an array of integers, find three numbers which sum to 0. Write an $O(n^2)$ algorithm for this problem. Example input / output:

   ```
   threesum([4,-2,6,-3,-5,1]) == [4,-5,1]
   ```

   (b) Given an array of integers, reverse the array in place. Do not use another array or variable to store any values. Example:

   ```
   reverse([4,1,5,3]) == [3,5,1,4]
   ```

2. Dynamic programming:

   (a) Write an $O(n)$ algorithm for `strstr(x,y)`, where n is the length of x. This algorithm determines if a string y is contained in a string x. Assume x contains unique characters. Example input / output:

   ```
   strstr("reader","read") == true
   strstr("reader","and") == false
   ```

   (b) Consider the algorithm `substr(x,y)`, which returns the longest common substring between x and y. For some example strings x and y, draw a $m \times n$ table to represent the comparison of x and y. m and n are the lengths of x and y.

   (c) Write an $O(mn)$ algorithm for `substr`. Example input / output:

   ```
   substr("explain","plant") == "pla"
   substr("everything","herring") == "ing"
   substr("cheese","mat") == ""
   ```

3. Memoization:

   (a) Every integer can be written as a multiple of prime numbers; furthermore, this set of prime numbers is unique. Some examples:

$$\texttt{factors(24)} = 2 * 2 * 2 * 2 * 3$$
$$\texttt{factors(15)} = 3 * 5$$
$$\texttt{factors(100)} = 2 * 2 * 5 * 5$$

   Given a function `prime(x)`, which returns the x^{th} prime number, and an input n, write an algorithm for calculating the prime factors of n.

   (b) Rewrite the previous algorithm using memoization.

4. Divide and conquer:

   (a) Given a sorted integer array of length n and number i, find the number of times i occurs in n. Your algorithm should be $O(\log n)$. Example input / output:

$$\texttt{count([1,2,2,2,4,6], 2)} = 3$$

   (b) Recall the maximum subarray problem – given an integer array, find the subarray with the largest sum. Example input / output:

$$\texttt{maxsubarray([6,-2,-5,7,4,-3,6,-9,5])} = \texttt{[7,4,-3,6]}$$

   If you were to use divide and conquer, what are the three cases for where the start and end of the subarray can be?

   (c) Write a **divide and conquer** algorithm for this problem. You can just return the largest sum. You should get an $O(n \log n)$ algorithm.