CS325: Analysis of Algorithms, Fall 2017

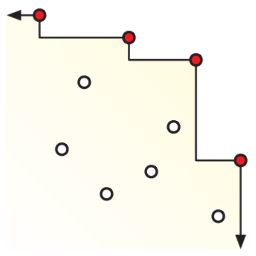
Practice Assignment 2*

Due: Tue, 10/17/17

Homework Policy:

1. Students should work on practice assignments individually. Each student submits to TEACH one set of *typeset* solutions, and hands in a printed hard copy in class or slides it under my door before the midnight of the due day. The hard copy will be graded.
2. Practice assignments will be graded on effort alone and will not be returned. Solutions will be posted.
3. The goal of the assignments is for you to learn solving algorithmic problems. So, I recommend spending sufficient time thinking about problems individually before discussing them with your friends.
4. You are allowed to discuss the problems with others, and you are allowed to use other resources, but you *must* cite them. Also, you *must* write everything in your own words, copying verbatim is plagiarism.
5. More items might be added to this list. ☺

Problem 1. Suppose you are given a set P of n points in the plane. A point $p \in P$ is maximal in P if no other point in P is both above and to the right of p. Intuitively, the maximal points define a "staircase" with all the other points of P below it.



A set of ten points, four of which are maximal.

Describe and analyze an algorithm to compute the number of maximal points in P in $O(n \log n)$ time.

Problem 2. Call a sequence $X[1 \cdots n]$ of numbers bitonic if there is an index i with $1 < i < n$, such that the prefix $X[1 \cdots i]$ is increasing and the suffix $X[i \cdots n]$ is decreasing. Describe an $O(\log n)$ time algorithm to search a bitonic sequence of length n for a number k.

*Some problems are from Jeff Erickson's lecture notes. Looking into similar problems from his lecture notes on recursion and dynamic programming is recommended.

Problem 3. Call a sequence $X[1 \cdots n]$ of numbers oscillating if $X[i] < X[i+1]$ for all even i, and $X[i] > X[i+1]$ for all odd i. Describe an efficient algorithm to compute the length of the longest oscillating subsequence of an arbitrary array A of integers.

Do *not* submit solutions for the following problems, they are just for practice.

Practice Problem A. A shuffle of two strings X and Y is formed by interspersing the characters into a new string, keeping the characters of X and Y in the same order. For example, the string BANANAANANAS is a shuffle of the strings BANANA and ANANAS in several different ways.

$$^{\mathtt{BANANA}}{}_{\mathtt{ANANAS}} \qquad ^{\mathtt{BAN}}{}_{\mathtt{ANA}}{}^{\mathtt{ANA}}{}_{\mathtt{NAS}} \qquad ^{\mathtt{B}}{}_{\mathtt{AN}}{}^{\mathtt{AN}}{}_{\mathtt{A}}{}^{\mathtt{A}}{}_{\mathtt{NA}}{}^{\mathtt{NA}}{}_{\mathtt{S}}$$

Similarly, the strings PRODGYRNAMAMMIINCG and DYPRONGARMAMMICING are both shuffles of DYNAMIC and PROGRAMMING:

$$_{\mathtt{PRO}}{}^{\mathtt{D}}{}_{\mathtt{G}}{}^{\mathtt{Y}}{}_{\mathtt{R}}{}^{\mathtt{NAM}}{}_{\mathtt{AMMI}}{}^{\mathtt{I}}{}_{\mathtt{N}}{}^{\mathtt{C}}{}_{\mathtt{G}} \qquad ^{\mathtt{DY}}{}_{\mathtt{PRO}}{}^{\mathtt{N}}{}_{\mathtt{G}}{}^{\mathtt{A}}{}_{\mathtt{R}}{}^{\mathtt{M}}{}_{\mathtt{AMM}}{}^{\mathtt{IC}}{}_{\mathtt{ING}}$$

Given three strings $A[1..m]$, $B[1..n]$, and $C[1..m+n]$, describe an algorithm to determine whether C is a shuffle of A and B. Prove your algorithm is correct and analyze its running time.

Practice Problem B.

(a) Suppose we are given a set L of n line segments in the plane, where each segment has one endpoint on the line $y = 0$ and one endpoint on the line $y = 1$, and all $2n$ endpoints are distinct. Describe and analyze an algorithm to compute the largest subset of L in which no pair of segments intersects.

(b) Suppose we are given a set L of n line segments in the plane, where each segment has one endpoint on the line $y = 0$ and one endpoint on the line $y = 1$, and all $2n$ endpoints are distinct. Describe and analyze an algorithm to compute the largest subset of L in which every pair of segments intersects.