

# 15-750 — Graduate Algorithms — Spring 2008

Miller and Sinop and Wu

Assignment 2 Due date: Friday, February 22

## Some Reminders:

- Read the Policies section on the course web site before you start working on this assignment. Collaboration is permitted for this assignment.
- You should refrain from using outside sources when solving these problems. For each problem, state whether you have seen it before. If you have questions, contact the course staff.
- We prefer that you type up your solutions (preferably using LaTeX). You may neatly hand-write your solutions, but if we have trouble reading them you will be required to type up future solutions.

## 1 Simple Paths and Convex Hull

[20 points]

Suppose that  $P = \{p_1, \dots, p_n\}$  is a set of points in the plane. We say the the sequences of distinct points  $Path = (p_1, \dots, p_k)$  is a simple path if the line segments  $l_i = [p_i p_{i+1}]$  are disjoint except for  $l_i \cap l_{i+1} = p_{i+1}$ . We may also allow  $p_1 = p_k$  and in this case  $l_{k-1} \cap l_1 = p_k$ .

In the following questions we shall investigate the relation between finding a simple path of a set of points and finding their convex hull.

1. Design an algorithm for finding a simple path using the points  $P$ . Make your algorithm as time efficient as possible.
2. In class we showed that computing the convex hull of  $n$  points in a comparison based model requires  $\Omega(n \log n)$  time. Show that given a simple path for these points one can find the convex hull in  $O(n)$  time.

HINT:

The idea is to run a variant of incremental convex hull where we add the points in the order they appear on the path. Suppose we are give a simple path  $Path = (p_1, \dots, p_n)$  on  $n$  distinct points and for simplicity no three are collinear. We start by constructing the triangle from the first three points and storing it as a doubly linked list of edges and recording which vertex is connected to the remain points on the path.

Let  $I = \{i \mid p_i \in CH(p_1, \dots, p_i)\}$  We will for each  $i \in I$  incrementally compute the convex hull of  $(p_1, \dots, p_i)$ . Make sure your algorithm handles the case when the point  $p_{i+1}$  is interior to  $CH(p_1, \dots, p_i)$ .

Use amortized analysis to show that your algorithm runs in  $O(n)$  time.

3. Show that in general any comparison based algorithm that finds a simple path of the points in  $P$  requires  $\Omega(n \log n)$  comparisons.

## 2 Smallest Enclosing Rectangle

[10 points] Suppose you are given a convex polygon of  $n$  points in two dimensional space, give an algorithm that returns the smallest (in terms of area) rectangle containing all the  $n$  points. Note that a rectangle's edge does not need to be vertical or horizontal. Show the correctness of your algorithm and analyze its time complexity. For full credit your algorithm should run in linear time.

HINT: You may use the following fact without proof. The boundary of the minimum enclosing rectangle must include an edge from the convex hull.

## 3 Making a Palindrome

[20 points]

Suppose we have two sets of code words  $V$  and  $W$  with  $n$  words each where the longest word has length  $l$ . Assume we are working over the binary alphabet.

### 3.1 Common Message

A common message of codes  $V$  and  $W$  is a message that can be formed from code words in  $V$  and from code words in  $W$ .

- Give (and prove) an upper bound on the length of the shortest common message of  $V$  and  $W$ . Here, length refers to number of bits, not number of words. (Try to make your bound as good as possible. You don't need to show a lower bound.)
- Give an  $O(l^2 n^2)$  time algorithm to determine if we can form a message from  $V$  that is also a message from  $W$ . You are allowed to use the same word more than once.

### 3.2 Palindrome

Give an  $O(l^2 n^2)$  time algorithm to determine if we can form a palindrome using words in  $W$ . You are allowed to use the same word more than once.

Recall: A palindrome is a string  $m$  which is equal to its reversal, e.g. "010010" and "00100".

Hint: We recommend you do the common message part first, if you haven't already.