

# 15-750 — Graduate Algorithms — Spring 2006

Miller and Derryberry

Assignment 4 Due date: Wednesday, April 5, 2006.

## Some Reminders:

- Read the Policies section on the course website before you start working on this assignment.
- You may work in **groups of size up to 3** for this problem set if you wish. However, **you should write up your solutions separately**. That is, collaboration should be limited to talking about the problems, so that your writeup is written entirely by you and not copied from your partner. In addition, state whether you worked alone and **list all collaborators**.
- Please refrain from consulting external materials when solving these problems. This does not apply to looking up standard inequalities, definitions, and such things (e.g.,  $(1 + 1/x)^x \leq e$  or Stirling's Formula).
- Please **submit both an electronic version, as well as a hard copy of your solutions** at the beginning of class on the due date.
- In all problems, it is implicit that you should show that your answer is correct, even when this is not explicitly stated.
- If you have questions, contact the course staff.

## 1 Union-Find

- (5 points) Suppose we do not perform path compressions while using the union-find algorithm, and when we perform a union we randomly pick one of the two elements to link to the other. What is the worst-case expected running time (in  $\Omega$ -notation) of a sequence of  $n - 1$  unions and  $n$  finds, where  $n$  is the number of elements, using this variant of union-find? Give an example that proves your answer.
- (10 points) Now, suppose we perform path compressions while using union-find, but we arbitrarily link one element as the child of the other when we perform a union. Prove that a sequence of  $m$  union and find operations on a set of  $n$  elements costs  $O((m + n) \log n)$ . (Hint: Recall our analysis of splay trees.)

## 2 Hamming Distance Using FFT

Define the *hamming distance* between two strings  $a$  and  $b$  of equal length to be the number of positions at which  $a$  and  $b$  differ. The goal of this problem is to develop an algorithm that takes as input a string  $s$  of length  $n$  and a pattern  $p$  of length  $m$ , both of which use characters from the set

$\{1, \dots, k\}$ , and outputs the hamming distance between  $p$  and every contiguous substring of length  $m$  in  $s$ .

- (a) (5 points) Suppose we restrict the alphabet to  $\{1, 2\}$ . How can we compute the hamming distances of  $p$  to the substrings of  $s$  in time  $O(n \log m)$ ?
- (b) (10 points) Now suppose we expand the alphabet to  $\{1, 2, 3\}$ . How can we compute the hamming distances of  $p$  to the substrings of  $s$  in time  $O(n \log m)$ ? (Hint: Recall the details of the FFT string processing lectures.)
- (c) (5 points) Now, give a fast algorithm for computing the hamming distances of  $p$  to the substrings of  $s$  if we expand the alphabet to  $\{1, \dots, k\}$  where  $k$  is relatively small compared to  $n$  and  $m$  but not necessarily constant. What is the running time of your algorithm in terms of  $n$ ,  $m$ , and  $k$ ?

### 3 My Dominators

Let  $G$  be a directed graph and  $s$  a distinguished vertex such that every vertex is reachable from  $s$ . We say that vertex  $a$  **dominates** vertex  $b$  in  $(G, s)$  if every path from  $s$  to  $b$  uses  $a$ .

- (a) (5 points) Give an  $O(V + E)$  time algorithm to determine if  $a$  dominates  $b$  in  $(G, s)$ .  
Hint: This part is easy.
- (b) (15 points) Give an  $O(V + E)$  time algorithm to determine all the dominators of  $b$  in  $(G, s)$ .