

## Homework 3

October 2, 2007

Due Date: October 9

1. (25 points) In this problem we consider a collection of elements that correspond to cell phones that are in the range covered by a given tower. For each cell phone object the following instance variables are stored:

- number (10 digit number)
- latitude
- longitude
- name
- total time in use (while in range of the tower)

We consider a variety of applications that all depend upon this data. All should allow new elements to be added or removed from the collection as cell phones go in and out of range of the tower.

For each of these applications indicate which untagged algorithmically positioned collection ADT would be the best fit and which instance variable(s) should be used to organize the data. For example, if you were going to use a comparator, which instance variable(s) should be used to define the comparator. You should give a brief explanation as to why the ADT you chose is the best fit for the problem.

- (a) (5 points) Given a phone number return the name of the person with that phone number, and their location.
  - (b) (5 points) For a given location with latitude  $x$  and longitude  $y$  and range  $d$ , return the list of all people whose cell phones have a latitude between  $x - d$  and  $x + d$  (inclusive), and a longitude between  $y - d$  and  $y + d$  (inclusive)
  - (c) (5 points) Iterate through the list of the names in alphabetical order of all users with a cell phone in the collection.
  - (d) (5 points) Return a sorted list of all phone numbers in the collection that begin with a specified prefix. For example, the application might want to receive all cell phones with area code 314.
  - (e) (5 points) Return the cell phone number that has had the most usage time.
2. (30 points) Prove a lower bound for the following two problems.
- (a) (15 points) You are given an array `votes` of size 4 with each element holding either a 1 or a 2 (which indicates the candidate for which a vote was cast). The goal is to determine if there is a candidate who received 3 or more votes. The output should either be the empty set if no candidate received at least 3 votes, or otherwise a set holding the indices in `votes` for a candidate with 3 or more votes.

Consider a model of computation in which the algorithm can only access `votes` by asking if `v[i] == v[j]` for any  $i, j$ . You are to give the best lower bound you can on the number of comparisons.

- (b) (15 points) You are given  $n$  coins identical in appearance; either all are genuine or exactly one is fake. It is unknown whether the fake coin is heavier or lighter than the genuine ones. Your model of computation to solve this problem is as follows. You can only learn about the coins through a provided `weigh(set1, set2)` method that takes as input two sets of coins `set1` and `set2` and returns one of the three possibilities: the two sets of coins have the same weight, `set1` is heavier, or that `set2` is heavier.

The problem is to determine if any coin is fake, and if so which coin is fake and whether it is heavier or lighter than the genuine ones.

3. (5 points) Consider the following list of 3-letter initials

DZB  
JZA  
JCM  
JAB  
CWJ  
CZM

Illustrate the execution of radix sort on the above list of initials by showing the list after each of the three phases of counting sort has completed.

4. (10 pts) Suppose you are given the task to sort 10000 numbers between 0 and  $10^{12} - 1$  (i.e. the keys are 12 digit numbers). You have decided to use radix sort but need to decide how many digits to group for each radix sort digit. Which is best among having 1 digit per radix sort digit, 3 digits per radix sort digit, 6 digits per radix sort digit, or directly using counting sort (i.e. 12 digits per radix sort digit)? You are provided with a counting sort procedure with exact time complexity of  $13n + 9k + 4$ . Show your work.
5. (30 pts) Let  $L_1, \dots, L_r$  be  $r$  unsorted lists, whose elements hold integers in the range  $[0, k - 1]$ . Let  $n$  be the total number of elements among all of the lists. That is, if  $n_i$  is the number of elements in  $L_i$ , then  $n = n_1 + n_2 + \dots + n_r$ . Describe an algorithm with **total worst-case** asymptotic time complexity of  $O(r + k + n)$  for getting all of the  $r$  lists into sorted order. So your final output will be  $r$  sorted lists (containing  $L_1, L_2, \dots, L_r$  in sorted order). Be sure to analyze the time complexity of your algorithm.

If you cannot think of an algorithm with the specified time complexity, then for partial credit describe the most efficient algorithm you can and correctly analyze its time complexity.

### Extra Credit Problems:

- 6\* (5 points) Consider the problem of sorting an unsorted array  $A$  of  $n$  elements which take on only two different values (e.g.  $A = [37, 60, 37, 37, 60, 60, 60, 60]$ ). Note that the two values are not known to the algorithm.
- (3 points) Briefly but clearly describe an  $O(n)$  *comparison-based* algorithm for solving this problem.
  - (2 points) Why doesn't the comparison sorting lower bound of  $\Omega(n \log n)$  apply to this problem?