Algorithms
Computer Science 140 & Mathematics 168
Spring 2012
Homework 5a
Due Thursday, February 16

- The first exam will be on Tuesday, February 21. Part 1 of the exam will be in-class and will comprise short questions that survey the topics and ideas that we've covered so far: Divide-and-conquer, dynamic programming, greed, and amortization. You may prepare an 8.5×11 sheet with notes on both sides to bring to the in-class part of the exam. You will then take Part 2 of the exam home with you. This part of the exam will comprise two homework-style problems. You can take that part of the exam in a 3-hour block of your choosing and those problems will be due at the beginning of class on Thursday, February 23. For the take-home part of the exam, you may use your 8.5×11 sheet, your own lecture notes, and your homework solutions and our sample solutions, but no other resources.

- The next homework, 5b, will have only one problem on it in order to give you time to study for the midterm exam.

1. **[30 Points] Amazing Delete-Less Dictionaries!** Recall that a dictionary is an abstract data type (ADT) that supports operations INSERT, FIND, and DELETE. Usually, self-balancing search trees are used to implement dictionaries because they support all of these operations in $O(\log n)$ worst-case time. However, imagine that we don't care about the cost of any *one* operation, we only care that a sequence of n operations will cost no more than a total of $O(n \log n)$. That is, we only care that our data structure have *amortized* $O(\log n)$ running time. In this case, there are a number of clever data structures that we can use in lieu of self-balancing trees. This problem explores one such data structure. **For simplicity though** we will assume that only INSERT and FIND are being supported.

   Our data structure works like this: Our data will be distributed over multiple arrays. Each array has length which is a power of 2, all the arrays have different lengths, all the arrays are completely full, and all the arrays are kept sorted. For example, if there are 9 elements in the dictionary, then since the binary representation of 9 is $2^3 + 2^0$, 8 of the elements will be in a sorted array of length 8 and 1 of the elements will be in an array of length 1. If a new element is added now, we'll have 10 elements. Thus, we'll keep 8 of the elements in their own array but place the new element with the element previously in the array of length 1 into a new sorted array of length 2. There is no particular relationship between elements in different arrays. We keep a linked list of the active arrays from smallest array to largest array.

   (a) Carefully describe how the FIND operation works for this data structure and analyze its worst-case running time. No amortization here; this is plain old worst-

case analysis! (It won't be asymptotically quite as good as $\log n$ but it will be asymptotically much better than n.)

(b) Carefully describe how to insert a new element in this array so that the total cost of n insertions (beginning from an empty dictionary) is $O(n \log n)$. Describe the algorithm and then give the amortized analysis using either the aggregation or accounting methods.

(c) [**8 Point Optional Bonus**] Do the amortized analysis for the n insertions again, this time using the potential method.