Algorithms Lab 3

1 Review

Topics covered this week:

- heaps and heapsort
- started quicksort

Review the lecture notes and work through the exercises handed out in class.

2 Homework

The homework problems are due on Friday 2/17.

Collaboration policy: You are allowed and encourged to discuss the problems with a group of peers, assuming this helps you learn. Keep the group small (groups of at most 3 people) and write your solutions individually.

Grading: The assignment will be evaluated based not only on the final asnwer, but also on clarity, neatness and attention to details.

Writing: Please write each problem on a separate sheet of paper, and write your name on each sheet. The problems will be graded by different TAs.

1. What is the minimum and maximum number of elements in a heap of height h? Use this to prove a tight bound for the height of a heap of n elements.

2. Come up with an algorithm that finds the kth smallest element in a a set of n distinct integers in $O(n + k \lg n)$ time.

3. (C-4.9) Suppose we are given a sequence S of n elements, each of which is colored red or blue. Assuming S is represented as an array, give an in-place method for ordering S so that all blue elements are listed before all the red elements.

4. (CLRS 8-2) Suppose we have an array of n data records to sort and that the key of each record has the value 0 or 1. An algorithm for sorting such a set of records might possess some subset of the following three desirable characteristics.:

   (1) The algorithm runs in $O(n)$ time.

   (2) The algorithm is stable.

   (3) The algorithm sorts in place, using no more than a constant amount of storage space in addition to the original array.

(a) Give an algorithm that satisfies (1) and (2) above.

(b) Give an algorithm that satisfies (1) and (3) above.

(c) Give an algorithm that satisfies (2) and (3) above.

(d) How would you extend your algorithm from (b) to handle the case when the values are $0, 1$ or 2; that is, you want to sort in place in $O(n)$ time.

5. (CLRS 7-3) Professors Dewey, Cheatham, and Howe have proposed the following "elegant" sorting algorithm:

   STOOGE-SORT(A, i, j)
   if $A[i] > A[j]$
       then exchange $A[i] \leftrightarrow A[j]$
   if $i + 1 \geq j$
       then return
   $k \leftarrow \lfloor (j - i + 1)/3 \rfloor$
   STOOGE-SORT$(A, i, j - k)$
   STOOGE-SORT$(A, i + k, j)$
   STOOGE-SORT$(A, i, j - k)$

   a. Argue that STOOGE-SORT$(A, 1, \mathit{length}[A])$ correctly sorts the input array $A[1..n]$, where $n = \mathit{length}[A]$.

   Hint: Argue that it sorts correctly any array of 1 or 2 elements. Then assume that it sorts sorrectly any array of $2n/3$ elements, and argue that this implies that it sorts correctly any array of n elements (What is true after the first recursive call? After the second?)

   b. Give a recurrence for the worst-case running time of STOOGE-SORT and a tight asymptotic (Θ-notation) bound on the worst-case running time.

   c. Compare the worst-case running time of STOOGE-SORT with that of insertion sort, merge sort, heapsort, and quicksort. Do the professors deserve tenure?

6. (CLRS 6.5-9) Assume you have k sorted lists containing a total of n elements, and you want to merge them together in a single (sorted) list containing all n elements. For simplicity you may assume that the k lists contain the same number of elements.

   (a) Approach 1: merge list 1 with list 2, then merge the result with list 3, then merge the result with list 4, and so on. What is the worst-case running time ?

   (b) Approach 2: split the k lists into two halves, merge each one recursively, then use the standard 2-way merge procedure (from mergesort) to combine the two halves. What is the worst-case running time ?

   (c) Give another approach (to merge the k lists) that uses a heap, and runs in $O(n \lg k)$-time.