Homework 4: Pivots, Partitions, and Sorting

Handed out Wednesday, November 2. Due at the start of class Thursday, November 8.

Problem 1. Consider the following recurrence with $0 \leq a < 1$ and $0 \leq b < 1$, and $c > 0$.

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ T(n) = T(an) + T(bn) + cn & \text{otherwise.} \end{cases}$$

(a) Show that $T(n)$ is bounded above by a linear function when $a + b < 1$.

(b) Show that $T(n)$ is in $O(n \log n)$ when $a + b = 1$.

Problem 2. For each of the following sorting algorithms, derive the asymptotic Θ running time *given that the input array is given in reverse sorted order* (that is, they keys appear in decreasing order in the initial array). You may assume all the elements are distinct. In each case briefly explain your answer, but a formal proof is not required.

(a) MergeSort.

(b) HeapSort.

(c) QuickSort. (Rather than picking the pivot at random, assume that the pivot is deterministically chosen to be the first element in the subarray, that is, $A[p]$.)

Problem 3. Give a $O(n \log_2 k)$–time algorithm to merge k sorted lists into one sorted list, where n is the number of elements in all the input lists. You may explain your algorithm at a high level. Briefly explain how your algorithm works and derive its running time. (Hint: Use a heap for the k-way merging.)

Problem 4. You are given an array of n keys, each with one of the values *red*, *white*, and *blue*. Give an $O(n)$ algorithm for "sorting" the keys so that all the *reds* come before all the *whites*, and all the *whites* come before all the *blues*. The only operations permitted are examination of a key to find out what color it is (ie, `if (keys[i] == Red)`), and swap of two keys specified by their indices (`Swap(i, j)`). That is to say, you *may not create any additional arrays!* Give psuedocode for your algorithm and derive its running time.