Assignment 4. **Lower bounds on sorting and $O(n)$ sorting.**

Given February 11, due Monday, February 23.

1. (from CLR, page 175) Show that it requires at least $2n-2$ comparisons to merge two sorted lists of length n using only binary comparisons. This a problem of combinatorics. You need to know the number of different ways the two lists could be merged, that this number of ways is $2^{(2n-2)}$ or something like that. The number 2^n is the number of subsets of a set of size n. If A and B are two subsets of the set $\{1, 2, \ldots, n\}$. The number of such pairs is 2^{2n}. Each pair corresponds to a different merging of the two lists. The merge starts with some numbers from list 1, then some from list 2, then more from list 1, then more from list 2 and so on. The subset A represents the indices from list 1 that start a streak of list 1 elements in the merged list. The B numbers represent the starts of streaks from the B list.

2. (a widely used question) Show that radix sort can sort n numbers in the range $[1, n^p]$ using $O(n)$ storage and $O(pn)$ work.

3. **a.** Suppose

$$a_{k+1} = \sqrt{a_k} \ , \tag{1}$$

that $a_0 = n$, and that $r(n)$ is the first k so that $a_k \leq 2$. Show that $r(n) = O(\log(\log(n)))$. *Hint:* Take the logarithm of both sides of (1).

b. Let $T(n)$ be the work required for a certain algorithm on input of size n. Suppose that

$$T(n) \leq Cn + \sqrt{n}T(\sqrt{n}) \ , \tag{2}$$

and that $T(2) = 1$. Show that $T(n) = O(n\log(\log(n)))$. *Hint:* Iterate the relation (2) until n gets to 2. Use part a to bound the number of iterates needed.

c. Suppose there were a way to choose buckets n buckets in bucket sort so that the number of elements in each bucket would never exceed $\sqrt{n}$. Show that a recursive sorting algorithm based on this would have work $O(n\log(\log(n)))$.

d. Suppose we select $n/\log(n)$ elements from an unsorted list of length n, sort the selected elements, and use them as boundaries for buckets into which the rest of the elements of the list are inserted. Show that the number in each bucket is less than $\sqrt{n}$ with probability greater than $1 - exp(-Cn^{1/3})$. If it were possible to select the bucket in constant time (as it is in bucket sort), this would give a probabilistic algorithm whose expected time to sort n numbers is $O(n\log(\log(n)))$. In bucket sort, the bucket is selected using the key, assumed to be a floating point number, and taking the integer part. *Note:* This last step is very hard. I will post a hint next week.