Problem Set 4

Due: Beginning of class, Thursday, February 11, 1999.

Suggested Reading: Chapters 14, 15, and 18.

Problem 4-1. Counting segments

Consider n chords on the unit circle, each defined by its endpoints. I.e. each chord is defined by a pair (α, β) with $0 \leq \alpha < \beta \leq 2\pi$. Describe an $O(n \lg n)$-time algorithm for determining the number of pairs of chords that intersect inside the circle. (For example, if the n chords are all diameters that meet at the center, then the correct answer is $\binom{n}{2}$.) Assume that no two chords share an endpoint.

Problem 4-2. In honor of Prof. D. T. Lee's seminar talk

Exercise 15.3-7, on page 295.

Problem 4-3. Amortized analysis

Binary search of a sorted array takes logarithmic search time, but the time to insert a new element is linear in the size of the array. We can improve the time for insertion by keeping several sorted arrays.

Specifically, suppose that we wish to support SEARCH and INSERT on a set of n elements. Let $k = \lceil \lg(n+1) \rceil$, and let the binary representation of n be $\langle n_{k-1}, n_{k-2}, \ldots, n_0 \rangle$. We have k sorted arrays $A_0, A_1, \ldots, A_{k-1}$, where for $i = 0, 1, \ldots, k-1$, the length of array A_i is 2^i. Each array is either full or empty, depending on whether $n_i = 1$ or $n_i = 0$, respectively. The total number of elements held in all k arrays is therefore $\sum_{i=0}^{k-1} n_i \, 2^i = n$. Although each individual array is sorted, there is no particular relationship between elements in different arrays.

(a) Describe how to perform the SEARCH operation for this data structure. Analyze its worst-case running time.

(b) Describe how to insert a new element into this data structure. Analyze its worst-case and amortized running times.

(c) Discuss how to implement DELETE.