Advanced Sorting

11-27-2002

Opening Discussion

- What did we talk about last class?
- Do you have any questions about the assignment?

Motivation

- You saw last time that sorting many thousands of elements can be a bit time consuming. In fact, that is the real problem with the $O(n^2)$ algorithms. When n gets really large things grind to a halt. We'd like to have something better and faster.
- Can you think of any way we might do this?

Revisiting Divide and Conquer

- Earlier we looked at algorithms that use the divide and conquer technique of solving problems. Some of you used this in your equation parsing assignment.
- The basic idea is that we break the problem into smaller parts assuming it is easier to solve that way, then build a solution to the larger problem from the solutions to the smaller pieces.
- Today we apply that to sorting.

Merge Sort

- This sort is based on the observation that if we have two sorted arrays, we can build one larger sorted array quickly, in O(n) time. So we break the problem apart until we get to single elements, then merge the sorted sub-arrays back together.
- The problem with this sort is that it can't be done well in place. We have to have a second array to copy into. It always works in O(n log n) time thought.

Quick Sort

- While merge sort does all its work coming back up the recursion, quick sort does it going down.
- At each step we pick a pivot and put all the elements smaller than the pivot on one side and all the larger ones on the other side. This can be done in O(n) time and when we get to the bottom we have a sorted array. It worked in place and has average O(n log n) performance.

Generic Sorting

- The one problem that we haven't dealt with yet in sorting is that we have to write different sort methods every time we want to sort different types or sort on different values. To get around this we need polymophism. C has very poor support for this.
- What we have to do is cast our array to a void* and pass a function pointer to be used for comparisons.

Minute Essay

- Assume we have a quick sort that always uses the first element as the pivot. Trace the behavior of a sort of {5, 1, 6, 3, 8, 4} using this.
- Have a great Thanksgiving! Remember that assignment #9 is due when you get back.