Mergesort and Quicksort

3-24-2002

Opening Discussion

- What did we talk about last class?
- Do you have any questions about assignment #4? Have you thought much about how you will be parsing the HTML to get out links?
- What do you know about mergesort and quicksort? How do they work? What are their advantages and limitations?

Recursive Sorts

- Last class we looked at some standard sorts that can be very easily implemented with nested loops and worked in $O(n^2)$ time. We also looked at a more sophisticated sort called Shellsort that could finish in under quadratic time. Today we are looking at two sorts that are most easily written with recursion and work in O(n log n) time on average. They are both examples of divide and conquer.

Mergesort

- The "simpler" of the two sorts we will discuss today is mergesort. This sort does basically no work in dividing the problem, but instead does most in the "conquer" stage at the end.
- It divides the array it needs to sort in two and calls itself with each half, obviously assuming they will return with them sorted.

Mergesort Continued

- When those calls return it runs through them picking the smallest from each and moving it to a new array. This produces a sorted version of the original array.
- The base case is when there is only one element left.
- The downfall here is that you have to have a second set of memory to copy things into. This makes the recursion a bit more difficult to write.

Analysis of Mergesort

- Mergesort goes through repeated divisions by 2 implying that it recurses down log n levels.
- At each level it has to do O(n) comparisons and copies implying that the total work done is O(n log n). This is both a worst and average case performance. The performance of mergesort is quite unaffected by the nature of the input.

Quicksort

- Another standard recursive sort is the one called quicksort. Quicksort fixes the problem of needing second array to copy things into. It also moves where the work is done to the divide stage with nothing really to do after the recursive calls return.
- Quicksort picks a pivot and moves all the items less than the pivot to one side. All items greater are on the other.

Quicksort Continued

- It then recurses for each side of the pivot until you get down to a base case.
- The key is in picking the pivot. Simple techniques include always using the first or picking at random. These produce O(n log n) average behavior, but O(n^2) worst case.
- By doing more comparisons you can pick a pivot close to the median.

Analysis of Quicksort

- On average a randomly selected pivot will be around the median. Because it isn't always, we can get more than $\log_2 n$ recursions, but average case it is only a small constant factor. Worst case you always pick something near the beginning or end which causes O(n) recursions and O(n^2) overall performance.

Minute Essay

- Assume you have a quicksort that chooses the first element for the pivot at each step. Do a trace of the sort of this array: {5,2,8,3,9,1}.
- Just a reminder, I also like into input or questions you might have.
- Remember that the design for assignment #4 is due today and Quiz #5 is on Wednesday.