**CS430 Lecture 06 Activities**

Opening Questions

1. Mergesort is ThetaO(n lg n) runtime in best case, worst case and average case. How much memory is needed for Mergesort on input size n?

2. Mergesort does all the work of sorting items in the Merge function, after recursively splitting the collection down to the base case. Briefly explain the difference with Quicksort.

Quicksort

A recursive divide and conquer sorting algorithm.

* base case: a list of length one element is sorted
* Divide “Partition” array into 2 sub-arrays with small #’s in first, large #’s in second and known index dividing them
* Conquer - recursively sort each sub-array
* Combine - Nothing to do

1. Write recursive pseudocode for Quicksort (similar to Mergesort).

Partition idea (you should be able to do this in place): pick the last element in the current array as the “pivot”, the number used to decide large or small. Then make a single pass of the array to move the “small” numbers before the “large” numbers and keep the “large” numbers after the “small” numbers. Then put the “pivot” between the two subarrays and return the location of the pivot.



2. Write iterative pseudocode for Partition. How much memory is needed?

3. Demonstrate Partition on this array.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 8 | 7 | 1 | 3 | 5 | 6 | 4 |

4. What do you think the best possible outcome would be for a call to Partition, and why? What about worst possible outcome?

5. Write (and solve) recurrence relations for Quicksort in the best case partition and worst case partition.

6. What if there is a pretty bad, but not awful, partition at every call. Try always a 9 to 1 split from partition. Write and solve recurrence relation.

With all the other sorts we could describe a particular input order that would yield worst case run time.

7. How can we avoid a particular input order yielding worst case run time for quicksort?

Visual Sorting Software By A. Alegoz, previous CS430 student (1.8Mb zipped, Win only)

<http://www.cs.iit.edu/~cs430/IITSort.zip>