CS 430 Introduction to Algorithms – Fall 2017
Recitation Class Note 4
Xiaolang Wang

1. Comparisons in Sorting

   To sort n numbers (assuming n is exponential of 2), how many comparisons do we need using different sorting algorithms?

   - Since we learned the lower bound on comparison-based sort this week, we are expecting the numbers are at least $\Theta(n \lg n)$
   - Insertion Sort
     - ❖ Best case: If we have a sorted array, from the second number on, each of them only compares with the last number in the sorted zone, and that's it. So (?)
     - ❖ Worst case: The i^{th}, $(2 \le i \le n)$ number needs to compare with each $(of\ i-1)$ number in the sorted zone, and there will be $\sum_{i=2}^{n}(i-1) = \sum_{j=1}^{n-1} j =$? comparisons. (And what might this array look like?)
   - Selection Sort
     - ❖ Best case/worst case: each iteration, we first set the first number in the remaining part (say the i^{th} in the whole array) to be $tempMin$, then compare it with $(n-i)$ remaining numbers to find the minimum. Thus there will be $\sum_{i=1}^{n-1}(n-i) = \sum_{i=1}^{n-1} i =$? comparisons.
     - ❖ We can see the number of comparisons is (dependent on/independent from) the permutation.
   - Merge Sort
     - ❖ When do we need to compare in a Merge Sort? (Merging)
     - ❖ To merge two $\left(\frac{n}{2}\right) -$ length array to one need at least $\left(\frac{n}{2}\right)$ comparisons and at most $(n-1)$ comparisons. Thus, given the i^{th} $(1 \le i \le \lg(n) - 1)$ layer and to merge into the $(i-1)^{th}$ layer, might sum up to at least $(\frac{n}{2})$ and at most $(n - 2^{i-1})$ comparisons.
     - ❖ Best case: $(\lg(n) - 1)\times\left(\frac{n}{2}\right) = \frac{1}{2}(n \lg n - n)$
     - ❖ Worst case: $\sum_{i=1}^{\lg(n)-1}\left(n - 2^{i-1}\right) = n \lg n\ - \frac{3}{2}n - 1$
   - Quicksort
     - ❖ When do we need to compare in Quick Sort? (Partitioning)
     - ❖ If we partition the array evenly, we will have more pivots in lower layers and a little fewer comparisons, and of course we have fewer layers.
     - ❖ Best case: we will have $\lg n$ layers, and each layer we have $n - |pivots\ on\ this\ layer|$ comparisons. In the i^{th} $(0 \le i \le \lg(n) - 2)$ layer, there are 2^i subarrays (and of course 2^i pivots), thus there are: $\sum_{i=0}^{\lg n - 2}(n - 2^i) = n \lg n - \frac{5}{2}n - 1$
     - ❖ Worst case: we will have $n-1$ layers and the i^{th} $(0 \le i \le n-2)$ layer needs $n - i - 1$ comparisons, total will be: $\sum_{i=2}^{n}(i-1) = \sum_{j=1}^{n-1} j =$? comparisons.
   - What about other sorting algorithms?

2. Binary Heap Operations
   - Full binary tree and complete binary tree.
   - Binary heaps are complete binary tree.

- How to find the parent in the heap? $(k-1)/2$, please check range
- How to insert? (Insert 21 to {18,12,8,9,11,4,3,5})
    - ❖ Insert to the last possible spot and move the number up along one path.
    - ❖ Might from leaf to root, thus at most $\lg n$ steps.
- How to get the max? (same example)
    - ❖ Swap the root(max) with the last element, then delete the max from the Heap
    - ❖ Compare the number at root with children while moving down through a path
    - ❖ Might from root to leaf, thus at most $\lg n$ steps.