CS 430
Week 8: Lecture

Objectives:

1. To introduce the Merge Sort algorithm.
2. To introduce the Quick Sort algorithm.

Reading Assignment:

Neapolitan and Naimipour: Chapters 7.4-7.5.

Contents:

1. Outline different Merge Sort Algorithms. \( \left( \frac{3}{4} \text{ hour} \right) \)
2. Analysis of Extra Space Usage. \( \left( \frac{1}{4} \text{ hour} \right) \)
3. Describe Quick Sort Algorithm. \( \left( \frac{1}{2} \text{ hour} \right) \)
4. Outline Pseudocode for Quick Sort Algorithm. \( \left( \frac{1}{2} \text{ hour} \right) \)
5. Illustrate Examples of Quick Sort. \( \left( \frac{1}{2} \text{ hour} \right) \)
6. Ways to Improve Quick Sort. \( \left( \frac{1}{2} \text{ hour} \right) \)
CS 430: Week 8

1. Outline different Merge Sort Algorithms. (3/4 hour)
   - Dynamic Programming
   - Linked
   - Complex

2. Analysis of Extra Space Usage. (1/4 hour)
   - How it relates to the time complexity of the algorithm.

3. Describe Quick Sort Algorithm. (1/2 hour)
   - Advantages of quick sort over merge sort.
   - Time complexity

4. Outline Pseudocode for Quick Sort Algorithm. (1/2 hour)
   ```
   void quicksort(index low, index high)
   {
       index pivotpoint;
       if (high > low)
       {
           partition(low, high, pivotpoint);
           quicksort(low, pivotpoint - 1);
           quicksort(pivotpoint + 1, high);
       }
   }
   ```

5. Illustrate Examples of Quick Sort. (1/2 hour)

6. Ways to Improve Quick Sort. (1/2 hour)
   - Reduce extra space usage of quick sort.
   - Stack smaller subarrays while the other is sorted.
   - Partition function can be improved.
   - Explicit stacking instead of stacking by recursion.
   - Determine threshold value at which the algorithm is iterative instead of recursive.
   - If the algorithm is already mostly sorted, do not select the first item right away – select the median.
Handouts, etc. for Lecture: None.