

CSI 33 LECTURE NOTES (Ojakian)

Topic 11: Sorting Algorithms

OUTLINE

(References: ch. 6, 13, 15)

1. Sorting Algorithms

1. The Goal: To Sort

Given an unordered list, produce an ordered list. Issues:

- (a) Time complexity (worst and typical case)
- (b) Space complexity
- (c) Simplicity of algorithm
- (d) Recall Bubble Sort from earlier in semester.

2. Selection Sort

(Ch. 6)

- (a) Worst case: n^2 . Average case: n^2
- (b) Find the smallest element and move it to the front (or into a different sublist).
 - i. If moved to a sublist, remove it from the original list
 - ii. If moved to the front, keep track of the boundary between the sorted sublist and the unsorted original list
- (c) Repeat till the sorted sublist contains all the elements.

3. Heap Sort

(Ch. 13)

- (a) From 1964? Worst case: $n \log n$. Average case: $n \log n$.
- (b) Phase 1: Build a max heap as an array.
- (c) Phase 2: Repeatedly remove the max element (at the the root) and put it in a sorted sublist (similar issues to selection sort).

4. Merge Sort

(Ch. 6)

- (a) From 1945? Worst case: $n \log n$. Average case: $n \log n$.
- (b) Divide list in half
- (c) Recursively call MergeSort on each half
- (d) Merge the two sorted halves.

5. Quick Sort

(Ch. 15.2)

- (a) From 1961? Worst case: n^2 . Average case: $n \log n$
- (b) Like MergeSort, but ...
- (c) Break up into less and more than the *pivot*
- (d) Note: Still recursively do each half, but now do “in-place” movements, avoiding the merge step.