CSE 373: Section 7

Sorting

November 8th

Quicksort

Use quicksort to sort the following array of integers. Use the first index as the pivot and show each partition and swap. You may use a cutoff of 3.

$$[6, 3, 2, 5, 1, 7, 4, 0]$$

Merge sort

Use merge sort to sort the following array of integers. You may start with them as individual elements. When dividing the array, the smaller half should be the second half.

$$[1, 6, 3, 2, 5, 0, 4]$$

Heap sort

Use a max-heap sort to sort the following array of integers. Use Floyd's method to create the heap at the beginning and show the array after every element is removed.
Remember, heap sort with a max-heap swaps the element to the back of the array

$$[3, 5, 0, 1, 2, 4, 7, 6]$$

Short answer

1. We are expecting the majority of the data that we are sorting to be "almost in order. What would be a good sorting algorithm to use?

2. Which of the sorting algorithms act as linear-time verifiers?

3. Consider an array that has a sorted portion of length N at the beginning, and a non-sorted portion of length $f(N)$ at the end. How large can $f(N)$ be if we want the whole array to be sorted in $O(N)$ time?
   Hint: what sort algorithm works best with sorted sub-portions?