Announcements

- Reminder: no class Nov 27 due to Thanksgiving Holiday
- Program 5 late turn in date: 11/27 at 11:59 PM
    - -5% to program grade
- I will have everything graded by this Sunday
- Program 6 will be assigned by Monday December 2

Sorting

Insertion Sort

Insertion sort is a sorting algorithm that treats the input as two parts, a sorted part and an unsorted part, and repeatedly inserts the next value from the unsorted part into the correct location in the sorted part.

Let’s look at an example...

Sorting (cont.)

Shell Sort

Shell sort is a sorting algorithm that treats the input as a collection of interleaved lists, and sorts each list individually with a variant of the **insertion sort** algorithm.

Shell sort uses **gap values** to determine the number of interleaved lists.

A **gap value** is a positive integer representing the distance between elements in an interleaved list.

For each interleaved list, if an element is at index i, the next element is at index i + gap value.

Sorting (cont.)

1. Shell sort begins by choosing a gap value K and sorting K interleaved lists in place.
2. Shell sort finished by performing a standard insertion sort on the entire array.
3. Because the interleaved parts have already been sorted, smaller elements will be close to the array’s beginning and larger elements towards the end.
4. Insertion sort can then quickly sort the nearly-sorted array.
5. Any positive integer gap value can be chosen. In the case that the array size is not evenly divisible by the gap value, some interleaved lists will have fewer items than others

Sorting (cont.)

Let’s look at an example of shell sort...

Sorting (cont.)

Quicksort

Quicksort is a sorting algorithm that repeatedly partitions the input into low and high parts (each part unsorted), and then recursively sorts each of those parts.

To partition the input, **quicksort** chooses a **pivot value** to divide data into low and high parts.

The **pivot** can be any value with within the array being sorted, commonly the value of the middle of the array element.

Sorting (cont.)

- To partition the input, the quicksort algorithm divides the array into two parts, referred to as the **low partition** and the **high partition**.
- All values in the **low partition** are less than or equal to the pivot value.
- All values in the **high partition** are greater than or equal to the pivot value.
- The values in each partition are not necessarily sorted.

Let’s look at an example...

Sorting (cont.)

- Once partitioned, each partition needs to be sorted. **Quicksort** is typically implemented as a recursive algorithm using calls to quicksort to sort the low and high partitions.
- This recursive process continues until a partition has one or zero elements, and thus already sorted.

Let’s look at an example...

Sorting (cont.)

Merge Sort

Merge Sort is a sorting algorithm that divides a list into two halves, recursively sorts each half, and then merges the sorted halves to produce a sorted list.

The recursive partitioning continues until a list of 1 element is reached.

Merge sort merges the two sorted partitions into a single list by repeatedly selecting the smallest element from either the left or right partition and adding that element to a temporary merged list.

Once fully merged, the elements in the temporary merged list are copied back to the original list.