CS 61B Spring 2021

Comparison Sorts

Topical Review Session 6: April 18, 2021

1 Sorting - Step by step

Show the steps taken by each sort on the following unordered list of integers (where duplicates are indicated with letters):

`2, 1, 8, 4A, 6, 7, 9, 4B`

(a) Selection Sort

(b) Insertion Sort

(c) Merge Sort

(d) Heap Sort *Note: if both children are equal, sink to the left.*

2 Sorting Runtime

Fill out the best-case and worst case runtimes for these sorts as well as whether they are stable or not in the table below.

	Best Case	Worst Case	Stable
Insertion Sort			
Selection Sort			
Merge Sort			
Heap Sort			
Quick Sort			

3 Identifying Sorts

Below you will find intermediate steps in performing various sorting algorithms on the same input list. The steps do not necessarily represent consecutive steps in the algorithm (that is, many steps are missing), but they are in the correct sequence. For each of them, select the algorithm it illustrates from among the following choices: insertion sort, selection sort, mergesort, quicksort (first element of sequence as pivot), and heapsort.

Input list: 1429, 3291, 7683, 1337, 192, 594, 4242, 9001, 4392, 129, 1000

(a) 1429, 3291, 7683, 192, 1337, 594, 4242, 9001, 4392, 129, 1000

1429, 3291, 192, 1337, 7683, 594, 4242, 9001, 129, 1000, 4392

192, 1337, 1429, 3291, 7683, 129, 594, 1000, 4242, 4392, 9001

(b) 1337, 192, 594, 129, 1000, 1429, 3291, 7683, 4242, 9001, 4392

192, 594, 129, 1000, 1337, 1429, 3291, 7683, 4242, 9001, 4392

129, 192, 594, 1000, 1337, 1429, 3291, 4242, 9001, 4392, 7683

(c) 1337, 1429, 3291, 7683, 192, 594, 4242, 9001, 4392, 129, 1000

192, 1337, 1429, 3291, 7683, 594, 4242, 9001, 4392, 129, 1000

192, 594, 1337, 1429, 3291, 7683, 4242, 9001, 4392, 129, 1000

(d) 1429, 3291, 7683, 9001, 1000, 594, 4242, 1337, 4392, 129, 192

7683, 4392, 4242, 3291, 1000, 594, 192, 1337, 1429, 129, 9001

129, 4392, 4242, 3291, 1000, 594, 192, 1337, 1429, 7683, 9001

In all these cases, the final step of the algorithm will be this:

129, 192, 594, 1000, 1337, 1429, 3291, 4242, 4392, 7683, 9001

4 Sorting Hat

The Sorting Hat has asked the 4 Houses to provide one new sorting algorithm each, which are modified versions of the sorts you have learned in 61B so far. For each of the algorithms, give the best and worst case runtimes using $\Theta(\cdot)$ notation with respect to the number of the elements in our input list, N. (Note that Slytherin chose not to participate as they were not selected as winners for the previous Sorting Hat competition).

(a) **LionSort:** Gryffindor has modified selection sort to additionally swap the maximum element of our unsorted list to the back at each iteration. Assume finding this maximum element takes $\Theta(N)$ time.

(b) **EagleSort:** Ravenclaw has decided to insert elements into a BST and then perform an in-order traversal on the tree to find their sorted order. In addition to performance, what sort is this most like?

(c) **BadgerSort:** Hufflepuff decide to modify merge sort by changing how sorted runs are merged. Instead of merging two sorted runs by iteratively choosing the minimum, Hufflepuff insertion sort the concatenation of the runs at each level.

5 Choose A Sort

For each of the following scenarios, choose the best sort to use. Explain your reasoning.

(a) The list you have to sort was created by taking a sorted list and swapping N pairs of adjacent elements.

(b) You have to sort a list on a machine where swapping two elements is much more costly than comparing two elements and you want to do the sort in place.

(c) Your list is so large that not all of the data will fit into your computer at once. As is, at any given time most of the list must be stored in some external device (an HDD), where accessing it is extremely slow.

(d) Given a list of emails ordered by send time, sort the list such the emails are ordered by the sender's name first while secondarily maintaining the time-ordering.

(e) You have a randomly shuffled list, where each number is unbounded in size, and want to sort the elements.