CS 61B Spring 2018

Sorting

Exam Prep 13: April 17, 2018

1 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) the left and right halves do not interact with each other until the very end

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

merge sort

(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

quicksort

(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

insertion sort

(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

heap sort

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

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

insertion sort: element always moves to the left

selection sort: sorted list grows from left to right

merge sort: local sorted list combines together

quick sort: partion to less than, equal to, and greater than

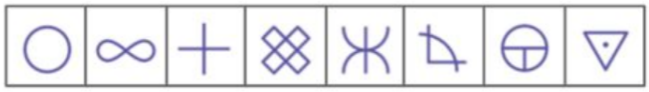
heap sort: sorted list grows from right to left

2 Reverse Engineering

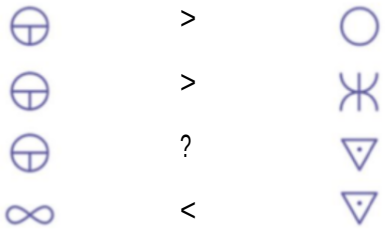
Consider the following unsorted array, and the array after an unknown number of iterations of selection sort as discussed in class (where we sort by identifying the minimum item and moving it to the front by swapping). Assume no two elements are equal

Unsorted:

After ? Iterations of Selection Sort:



For each relation below, **write** $<$, $>$, or ? for insufficient information regarding the relation between the two objects



3 Conceptual Sorts

Answer the following questions regarding various sorting algorithms that weve discussed in class. If the question is T/F and the statement is true, provide an explanation. If the statement is false, provide a counterexample.

(a) (T/F) Quicksort has a worst case runtime of Θ(NlogN), whereN is the number of elements in the list that were sorting.

F: the worst case run time is O(N^2), where the chosen pivot is always the smallest or greatest element.

(b) We have a system running insertion sort and we find that its completing faster than expected. What could we conclude about the input to the sorting algorithm?

almost all elements are sorted, except a small number of items.

(c) Give a 5 integer array such that it elicits the worst case running time for insertion sort.

5, 4, 3, 2, 1

(d) (T/F) Heapsort is stable.

F: seen the above example, where the order of equivalent items cannot be preserved

(e) Give some reasons as to why someone would use mergesort over quicksort

quicksort is unstable. so if we are going to sort among object items, and we want stability, we had better use merge sort

(f) You will be given an answer bank, each item of which may be used multiple times. You may not need to use every answer, and each statement may have more than one answer.

A. QuickSort (nonrandom, inplace using Hoare partitioning, and choose the leftmost item as the pivot)

B. MergeSort

C. Selection Sort

D. Insertion Sort

E. HeapSort

N. (None of the above)

List all letters that apply. List them in alphabetical order, or if the answer is none of them, use N indicating none of the above. All answers refer to the entire sorting process, not a single step of the sorting process. For each of the problems below, assume that N indicates the number of elements being sorted.

A B C E

E is incorrect, because it's an overall average result

——— Bounded by Ω(NlogN)lower bound.

B E

——— Has a worst case runtime that is asymptotically better than Quicksorts worstcase runtime.

C

——— In the worst case, performs Θ(N) pairwise swaps of elements.

A B C D

——— Never compares the same two elements twice.

N

——— Runs in best case Θ(logN)time for certain inputs

C is incorrect, because when dealing with duplicates, selection sort won't putting them together, but just deal with one of them