CS 61BL | Sorting

Summer 2021 | Quiz 12: Friday August 6, 2021

# 1 Best Sort

(a) Which sort is best for nearly sorted arrays?

( ) Selection sort ( ) Insertion sort ( ) Quicksort

What would be the runtime of your selected algorithm on the nearly sorted array? Let $n$ be the number of elements in the array.

[ ] $\Theta(\log(n))$
[ ] $\Theta(n)$
[ ] $\Theta(n\log(n))$
[ ] $\Theta(n^2)$

(b) Each student in our course has a unique ID of length 10 which are allocated by alphabetizing the students by their names and counting off.

If we wanted to sort our student objects, what would be the fastest sort you could use to achieve this?

( ) Insertion sort ( ) Quicksort ( ) Radix sort

What would be the runtime of your selection above?

[ ] $\Theta(\log(n))$
[ ] $\Theta(n)$
[ ] $\Theta(n\log(n))$
[ ] $\Theta(n^2)$

# HeapSort Steps

1) Bottom Up Heapification n

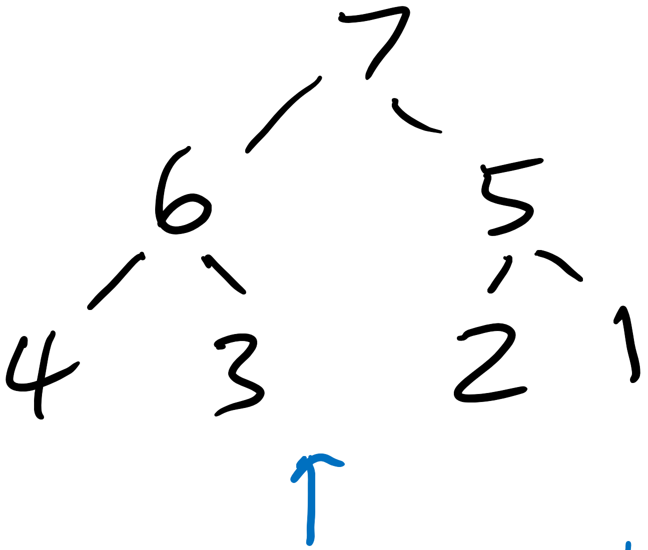
↳ Bubble down, reverse order

2) Keep removing max, put at end

↳ Reheapify!

Best: n

Worst: nlogn



Order of consideration for 1 by numeric value

[7, 6, 5, 4, 3, 2, 1] ←

Swaps: 卌 I

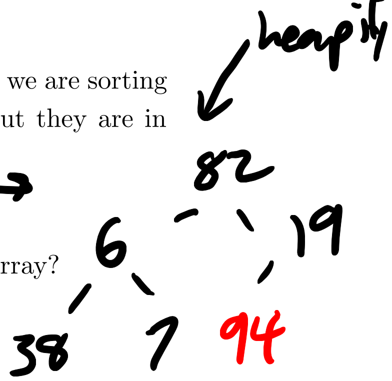
# 2 Mechanical

For all parts, input each element in the array in their correct order with a **single space** between each element.

For example, if the array contained {`1, 2, 3`}, you would input `1 2 3`. For all parts below, we are sorting the same array. Also note that Quicksort and MSD Radix sort aren't covered below, but they are in scope for the final.

(a) Suppose we are sorting the array {`7, 38, 6, 94, 82, 19`}.

After **6** swaps of **in-place heapsort using a max heap**, what is the state of our array?



(b) Suppose we are sorting the array {`7, 38, 6, 94, 82, 19`}.

After **3** swaps of **selection sort**, what is the state of our array?

6 | 38 7 94 82 19

6 7 | 38 94 82 19

6 7 19 | 94 82 38

(c) Suppose we are sorting the array {`7, 38, 6, 94, 82, 19`}.

After **3** swaps of **insertion sort**, what is the state of our array?

(d) Suppose we are sorting the array {`7, 38, 6, 94, 82, 19`} with `mergeSort`.

Before the final merge pass, what is the state of the two arrays? Select the *two* arrays below. Assume if we divide an odd length array in half, the right half is longer.

[ ] (6 7 38)
[ ] (6 7)
[ ] (38 82 94)
[ ] (7 38)
[ ] (19 82 94)
[ ] (82 94)
[ ] (7 38 6)
[ ] (94 82 19)

(e) Suppose we are sorting the array {`7, 38, 6, 94, 82, 19`} with `LSD sort`.

After the *first* pass, what is the state of the array?

**5. Subsequences (240 Points)**

In this problem, we will analyze the behavior of different sorting algorithms on the array [652, 135, 101, 383, 495, 651]. For each part below, we will give a partial state of the array, e.g. [1, 2, 3, ...] where the ... represents the rest of the array, and you will select **all** sorting algorithms that could have the given array state **at any point** during the sorting algorithm's execution (i.e., even in the middle of an operation). If none of the above apply, select **None**.

**Hint:** You do not need to manually carry out all four sorting algorithms in their entirety to find the answer! You can do this, of course, but manually carrying out every sort (especially heapsort) will be tedious and much slower than thinking more carefully.

135 652 101

For this part, the possible sorting algorithms are:

- Insertion sort
- Selection Sort
- Quicksort using leftmost item as pivot, no shuffling, and with ~~Hoare partitioning~~
- Heapsort using a max heap and bottom up heapification

Here is a diagram of the array that is being sorted for your convenience:

652
135 101
383 495 651

| | | | | | |
|---|---|---|---|---|---|
| 652 | 135 | 101 | 383 | 495 | 651 |
| 0 | 1 | 2 | 3 | 4 | 5 |

new
1 2 3 4 | 0
1 2 4 6 | 3
3 4 6

**a) (50 Points)** [652, 495, 651, ...]

☐ Insertion sort ☐ Selection sort ☐ ~~Quicksort~~ ☒ Heapsort

**b) (50 Points)** [101, 135, 652, ...]

☒ Insertion sort ☒ Selection sort ☐ ~~Quicksort~~ ☐ Heapsort

**c) (50 Points)** [135, 101, 652, ...]

☒ Insertion sort ☐ Selection sort ☐ ~~Quicksort~~ ☐ Heapsort

[... , 652]

**2. Quicksort and Heap Convergence**

**a) (50 Points)** The array [383, 135, 101, 495, 651, 652] occurs during execution for both Quicksort and Heapsort when applied to the array from part 1. Assume Quicksort and Heapsort are as described in part 1. After how many completed partition operations does this array occur for Quicksort?

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

**b) (50 Points)** The array [383, 135, 101, 495, 651, 652] occurs during execution for both Quicksort and Heapsort when applied to the array from part 1. Assume Quicksort and Heapsort are as described in part 1. After how many completed deletion operations does this array occur for Heapsort?

○ 0 ○ 1 ○ 2 ⊗ 3 ○ 4 ○ 5 ○ 6 ○ 7