CS 61B
Fall 2026

Comparison Sorts

Discussion 12: November 16, 2026

1 All Sorts of Sorts

Show the steps taken by each sort on the following unordered list:

`0, 4, 2, 7, 6, 1, 3, 5`

(a) Insertion sort

(b) Selection sort

(c) Merge sort

(d) Use heapsort to sort the following array (hint: draw out the heap).
Draw out the array at each step: `0, 6, 2, 7, 4`

2 Quicksort

(a) Sort the following unordered list using Quicksort. Assume that we always choose first element as the pivot and that we use the 3-way merge partitioning process described in lecture. Show the steps taken at each partitioning step.

```
18, 7, 22, 34, 99, 18, 11, 4
```

(b) Suppose we use 3-scan partitioning with the **middle item** (rounded down — i.e. the item at index $\lfloor \frac{N-1}{2} \rfloor$) as the pivot at every level of recursion. Describe an input that gives $\Theta(N^2)$ runtime, assuming we don't shuffle the array first.

(c) What are two techniques that can be used to reduce the probability of Quicksort taking its worst-case runtime?

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. When we split an odd length array in half in mergesort, assume the larger half is on the right.

Input list: `1429, 3291, 7683, 1337, 0192, 594, 4242, 9001, 4392, 0129, 1000`

(a)
```
1429, 3291, 7683, 1337, 0192, 0594, 4242, 9001, 4392, 0129, 1000
1429, 3291, 0192, 1337, 7683, 0594, 4242, 9001, 0129, 1000, 4392
0192, 1337, 1429, 3291, 7683, 0129, 0594, 1000, 4242, 4392, 9001
```

(b)
```
1337, 0192, 0594, 0129, 1000, 1429, 3291, 7683, 4242, 9001, 4392
0192, 0594, 0129, 1000, 1337, 1429, 3291, 7683, 4242, 9001, 4392
0129, 0192, 0594, 1000, 1337, 1429, 3291, 4242, 4392, 7683, 9001
```

(c)
```
1337, 1429, 3291, 7683, 0192, 0594, 4242, 9001, 4392, 0129, 1000
0192, 1337, 1429, 3291, 7683, 0594, 4242, 9001, 4392, 0129, 1000
0192, 0594, 1337, 1429, 3291, 7683, 4242, 9001, 4392, 0129, 1000
```

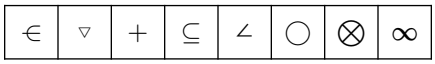
(d)
```
1429, 3291, 7683, 9001, 1000, 0594, 4242, 1337, 4392, 0129, 0192
7683, 4392, 4242, 3291, 1000, 0594, 0192, 1337, 1429, 0129, 9001
0129, 4392, 4242, 3291, 1000, 0594, 0192, 1337, 1429, 7683, 9001
```

In all these cases, the final step of the algorithm will be this:
`0129, 0192, 0594, 1000, 1337, 1429, 3291, 4242, 4392, 7683, 9001`

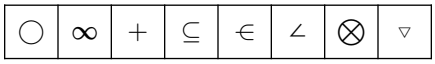
4 Reverse Engineering

Consider the following unsorted array, and the array after an unknown number of iterations of selection sort. Assume no two elements are equal.

Unsorted:



Sorted:



(a) For each relation below, write $<$, $>$, or ? for insufficient information regarding the relation between the two objects.

