CS 61B Fall 2023

Sorting

Discussion 12: November 6, 2023

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 Crystal Has Been Waiting For This

Claire and Ada, two alumni 61B TAs, are trying to sort the TAs by height so they can snap a photo. Can you help them out?

```java
public class TA {
    private String name;
    private int height;

    public TA(String name, int height) {
        this.name = name;
        this.height = height;
    }
}
```

(a) Implement a `TAComparator` below such that it compares two TAs' height. Recall that a `Comparator`'s `compare` method returns a negative number when `o1` is "less than" `o2`, positive number when `o1` is "greater than" `o2`, and 0 when they are the same.

(b) Jedi suggests that we use Quicksort with our comparator. Given the following list of TAs, who would make the worst pivot? What about the best pivot?

```java
TA anish = new TA("Anish", 6);
TA sherry = new TA("Sherry", 9001);
TA adit = new TA("Adit", 1);
TA kenneth = new TA("Kenneth", 5);
TA sree = new TA("Sree", 7);
TA noah = new TA("Noah", 25);
TA dhruti = new TA("Dhruti", 9);
TA hailey = new TA("Hailey", 4);
TA eric = new TA("Eric", 8);
TA austin = new TA("Austin", 8);
```

(c) Austin points out that even though he got in line after Eric, he ended up in front of him in the sorted list produced by Quicksort, which he doesn't like, because that makes it seem like he's shorter than Eric! How might we ensure that Austin ends up behind Eric?

(d) Our TAs have just been sorted by height, but suddenly Angelina and Aram come running in late! Which sort will do the most minimal work to get them in their correct spots, and what is the additional runtime it will take (ie. not including the runtime for sorting all the other TAs first)?

3 Zero One Two-Step

(a) Given an array that only contains 0's, 1's and 2's, write an algorithm to sort it in linear time without creating a new array. You may want to use the provided helper method, `swap`.

```
public static void specialSort(int[] arr) {
    int front = 0;
    int back = arr.length - 1;
    int curr = 0;

    while (________________________) {
        if (arr[curr] < 1) {

            _______________________;

            _______________________;

            _______________________;
        } else if (arr[curr] > 1) {

            _______________________;

            _______________________;
        } else {

            _______________________;
        }
    }

private static void swap(int[] arr, int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

(b) We just wrote a linear time sort, how cool! Why can't we always use this sort, even though it has better runtime than Mergesort or Quicksort?

(c) The sort we wrote above is also "in place". What does it mean to sort "in place", and why would we want this?