CSM 61B Spring 2019

More Sorting!

Mentoring 11: April 15, 2019

1 Stability

Stability is a property of some sorting algorithms. Stability essentially means that if we have two elements that are equal, then their relative ordering in the sorted list is the same as the ordering in the unsorted list. For instance, let's say that we had an array of integers.

```
{ 1, 2, 1, 3, 1, 2, 4 }
```

Since we have multiple 1 and 2s, let's label these.

```
{ 1A, 2A, 1B, 3, 1C, 2B, 4 }
```

A stable sort would result in the final list being

```
{ 1A, 1B, 1C, 2A, 2B, 3, 4 }
```

Why is this desirable? Say that we have an Excel spreadsheet where we are recording the names of people who log in to CSM Scheduler. The first column contains the timestamps, and the second column contains their username. The timestamps are already ordered in increasing order. If we wanted to sort the username, so that we could group the list to see when each username logs in, we would want that the timestamps maintain their relative order. This is precisely what a stable sort ensures.

1.1 Why does Java's built-in `Array.sort` method use quicksort for **`int`**, **`long`**, **`char`**, or other primitive arrays, but merge sort for all `Object` arrays?

2 Pivot Choice

2.1 For each pivot selection strategy below, what is the best, average and worst case runtime?

(a) Always choose the first value in the list.

(b) Always find and choose the median value in the list. Assume finding the median takes $O(N)$ time where N is the length of the list.

(c) Always choose a random pivot.

3 Even More Sorting

In the table below, the runtimes of the sorts gone over in last week's section are written for you. Fill out the best-case and worst-case runtimes for Quicksort as well as whether all of the sorts we've seen so far are stable or not.

Algorithm	Best-case	Worst-case	Stable
Selection Sort	$\Theta(N^2)$	$\Theta(N^2)$	
Insertion Sort	$\Theta(N)$	$\Theta(N^2)$	
Merge Sort	$\Theta(N \log N)$	$\Theta(N \log N)$	
Heapsort	$\Theta(N)$	$\Theta(N \log N)$	
Quicksort			

3.1 Run the quicksort algorithm. Assume we pick the middle element as the pivot; if there is no exact middle, pick the element to the right of the middle.

```
{ 1, 3, 8, 2, 6, 4, 5, 9 }
```

4 Sorting Out My Head!

4.1 Web developers use many different sorts for the different types of lists that they might want to sort. For each of these, provide the best sorting algorithm amongst the following: Mergesort, Quicksort (with Hoare Partitioning), Insertion Sort, LSD Sort. Also, state the worst-case runtime.

(a) A list of N packets received by a server over time. Each packet has the timestamp at which the sender sent it. However, some packets may be dropped or arrive out-of-order due to the faulty network. Sort this list by that timestamp (sent time).

(b) A list of N websites. Each website has the number of total visitors. Sort this list by visitor count.

(c) After sorting by visitor count, we now want to sort by webpage file size. If websites have the same file size, they should be ordered by visitor count.

(d) A list of 20 names. Sort in alphabetical order.

5 QuickSort vs. Merge Sort

5.1 (a) What are the advantages and disadvantages of quicksort?

(b) What are the advantages and disadvantages of merge sort?

6 Cheapest Flights Within K Stops *Extra Practice*

6.1 This question was adapted from LeetCode: cheapest-flights-within-k-stops

There are n cities connected by m flights. Each flight starts from city u and arrives at v with a price w. Thus a city is represented as [u, v, w].

Given all the cities and flights, together with starting city `src` and the destination `dst`, your task is to find the cheapest price from `src` to `dst` with up to K stops. If there is no such route, output -1.