CSM 61B Fall 2017

Sorting Algorithms

Mentoring 10: October 30, 2017

1 Asymptotics Potpourri

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.

Algorithm	Best-case	Worst-case	Stable
Selection Sort			
Insertion Sort			
Merge Sort			
Heapsort			
Quicksort			

2 QuickCo

Malicious Mallory has been hired by a competitor to break into QuickCo, the world leader in sorting algorithms, and tamper with its Quicksort algorithms by making them as slow as possible. Mallory succeeded in unlocking the mainframe, but now she needs your help in slowing QuickCo's algorithms down to a halt!

For this question, assume that the Quicksort algorithm used has three steps.

1. Iterate through the array, to count how many elements are smaller than the pivot, larger than the pivot, and equal to the pivot.
2. Create 3 arrays for smaller, larger and equal elements of the correct size and then, in a second run through the array, place the appropriate elements in these arrays.
3. Finally, recurse on the smaller and larger arrays.

```
int[] data = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
```

2.1 Mallory decides to change the way QuickCo chooses a pivot for Quicksort.

Given the `int[] data` of size `n`, what choice of pivot would cause the worst-case runtime for Quicksort? Expand out the first few steps of the quicksort algorithms in such a case.

2.2 Mallory finds an algorithm which always selects the middle element but she is unable to gain write access to it. If the array has an even number of elements, the algorithm picks the element to the left. However, Mallory discovers a way to intercept the incoming data and rearrange it before the algorithm runs.

Given the `int[] data` of size `n`, rearrange the numbers such that the algorithm will run in its worst-case time. Expand out the first few steps of the quicksort algorithms in such a case.

2.3 Does the worst-case runtime of Quicksort depend on the array order, pivot choice, or both? Why?

2.4 If the worst-case runtime of Quicksort is $O(N^2)$ while the worst-case runtime of Mergesort is $O(N \log N)$, why do we ever use Quicksort?

3 Vertigo

3.1 We have a list of N elements that should be sorted, but to our surprise we recently discovered that there are at most k pairs out of order, or k **inversions**, in the list. The list `{ 0, 1, 2, 6, 4, 5, 3 }`, for example, contains 5 inversions: $(6, 4), (6, 5), (6, 3), (4, 3), (5, 3)$.

For each value of k below, state the most efficient sorting algorithm and give a tight asymptotic runtime bound.

(a) $k \in O(\log N)$

(b) $k \in O(N)$

(c) $k \in O(N^2)$

4 Getting to Know You

4.1 Run MSD and LSD radix sort on the following DNA sequence such that the output is sorted in alphabetical order ($A < C < G < T$).

```
ACAG
CTAG
ACAA
TGAG
CCTC
GAGT
```

```
ACAG
CTAG
ACAA
TGAG
CCTC
GAGT
```

4.2 Performing Radix Sort seems to be a fast sorting algorithm. Why don't we always use it?

4.3 Why does Least-Significant Digit radix sort work?

5 More Asymptotics Potpourri *Extra Practice*

Algorithm	Best-case	Worst-case	Stable
Counting Sort			
LSD Radix Sort			
MSD Radix Sort			

6 Berkeley Bytes Buffet *Extra Practice*

You are the proud owner of Berkeley Bytes Buffet and business is good! You have a policy where children under 8 eat free and seniors eat 50 percent off. Since you're a savvy business owner, you keep the ages of all your customers.

6.1 For taxes, you must to submit a list of the ages of your customers in sorted order. Define `ageSort`, which takes an **`int[] array`** of all customers' ages and returns a sorted array. Assume customers are less than 150 years old.

```
public class BerkeleyBytes {
    public static int[] histogram(int[] ages) {
        private static int maxAge = 149;
        int[] ageCounts = new int[maxAge + 1];
        for (int age : ages) {
            ageCounts[age] += 1;
        }
        return ageCounts;
    }
    public static int[] ageSort(int[] ages) {









    }
}
```

6.2 Time passes and your restaurant is doing well. Unfortunately, our robot overlords advanced medicine to the point where humans are now immortal.

(a) How could we extend the algorithm to accept a list of any ages?

(b) When would we be able to use this type of sort?