CSM 61B Spring 2017

Counting Sorts & Tries

Week 14: April 25, 2017

1 Berkeley Bytes Buffet

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.

1.1 Now, for taxes, you need to submit a list of the ages of your customers in sorted order. Define a procedure, `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) {
        int[] ageCounts = new int[150];
        for (int age : ages) {
            ageCounts[age - 1] += 1;
        }
        return ageCounts;
    }
    public static int[] ageSort(int[] ages) {









    }
}
```

1.2 Time passes and your restaurant is doing well. Unfortunately, our robot overlords have advanced medicine to the point where humans have become immortal.

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

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

2 Getting to Know You

2.1 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
```

3 More Asymptotics Potpourri

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

4 New World Order

4.1 Given a list of words (possibly repeated), devise a scheme to efficiently return a list of all the words that start with a given prefix.

4.2 Given a dictionary of words, describe a procedure for checking if a new word can be created out of the concatenation of two words in the dictionary. For example, if our dictionary contains the words, `"news"`, `"paper"`, `"new"`, and `"ape"`, we should be able to discover the new word, `"newspaper"`.

5 Sorting Mechanics *Extra for Experts*

5.1 Below, the **leftmost column** is an unsorted list of strings. The **rightmost column** gives the same strings in sorted order. Each of the remaining columns gives the contents of the list during some intermediate step of one of the algorithms listed below. Match each column with its corresponding algorithm.

· Merge sort · Quicksort · Heap sort · LSD radix sort · MSD radix sort

For quicksort, choose the topmost element as the pivot. Use the recursive (or top-down) implementation for merge sort.

	A	B	C	D	E	F	G
1	4873	1876	1874	1626	9573	2212	1626
2	1874	1874	1626	1874	7121	8917	1874
3	8917	2212	1876	1876	9132	7121	1876
4	1626	1626	1897	4873	6973	1626	1897
5	4982	3492	2212	4982	4982	9132	2212
6	9132	1897	3492	8917	8917	6152	3492
7	9573	4873	4873	9132	6152	4873	4873
8	1876	9573	4982	9573	1876	9573	4982
9	6973	6973	6973	1897	1626	6973	6152
10	1897	9132	6152	3492	1897	1874	6973
11	9587	9587	7121	6973	1874	1876	7121
12	3492	4982	8917	9587	3492	9877	8917
13	9877	9877	9132	2212	4873	4982	9132
14	2212	8917	9573	6152	2212	9587	9573
15	6152	6152	9587	7121	9587	3492	9587
16	7121	7121	9877	9877	9877	1897	9877