Practice Problem Set 4

.

Problem 1

Justify the runtime of each of the following sorting algorithm using the described input type. (Note: we used "numbers" to mean natural numbers).

1. Counting sort run on n numbers, where each number is less than or equal to n^2.
2. Counting sort run on n numbers, where each number is less than or equal to $\log n$
3. Radix sort on n numbers expressed in binary, where each number is less than 2^n
4. Radix sort on n numbers expressed in binary, where each number is less than n^2
5. Radix sort on n numbers expressed in base 10, where each number is less than n.
6. Radix sort on n numbers expressed in base 10, where each number is less than 100.
7. Counting sort on 100 numbers, where each number is less than n.
8. Insertion sort on 100 numbers, where each number is less than n.
9. Counting sort on n numbers, where each number is less than 100.
10. Insertion sort on n numbers, where each number is less than 100.
11. Radix sort on n real numbers where each number is of the form xxx.xx (i.e: a decimal number with at most 2 decimals).

Problem 2

- Describe the difference between the **expected runtime** and **worst-case runtime**
- Suppose you are given a set of n real numbers, uniformly distributed in the range $-10 \leq x \leq 10$.

  -Describe how to use Bucket sort such that the **expected** runtime is $\Theta(n)$.

  -What is the worst-case run-time of your algorithm?

  -Can you improve the worst-case runtime by using a different algorithm?
- Can you determine the expected runtime of bucket sort in n numbers if the input is *not* necessarily uniformly distributed?

Problem 3

A professor would like to sort a list of n final grades. Each grade has been exported from Brightspace and is a real number between 0 and 100 inclusive. Brightspace stores a grade with at most 2 decimal places. Is it appropriate to use Bucket Sort to sort these grades? Which sorting algorithms have a worst-case runtime of $\Theta(n)$ on this input?

Problem 4:.

Repeat the above question assuming that the grades stored could have any number of decimal places.

Problem 5

Suppose you are given a set of n fractions as input, where each fraction represents a positive real number in the range $0 \ldots 1$. Furthermore, the denominator of each fraction is at most 1000. Which algorithm(s) can be used to sort this input in worst-case time $O(n)$?

Problem 6

- Explain why counting sort (shown in class) inserts the elements from A into the final output array starting with those at the *back* of array A.
- Describe how to update the pseudo-code of merge-sort, insertion-sort, and bubble-sort so that they are also *stable* sorts.

Problem 7:

Build a decision tree to find the minimum of 3 numbers. Repeat for 4 numbers. Each node in the decision tree represents a comparison of exactly 2 numbers. For the general version of n numbers, what is the shortest possible path in the tree? Use the result to explain that the runtime to determine the minimum element in a set of n numbers is $\Omega(n)$.

Problem 8:

Suppose you implement bucket sort on n real numbers that are distributed uniformly in the range $0 \leq x \leq 1$. If we use $n/2$ equal-sized buckets instead of n, how many elements do you expect in each bucket? Is there a change in the expected runtime of Bucket-sort? Repeat for $\sqrt{n}$ buckets and n^2 buckets.

Problem 9:

Write the pseudo-code for Counting sort. You may assume that your input is in array $A[1, \ldots n]$, and that each element is an integer that is at least 1. You do not have any other information about the input. Your final sorted array must be in array A.

Problem 10:

- Draw the decision tree that represents the comparisons made by insertion sort on $\{a, b, c\}$. What is the shortest and longest path in the tree?
- Consider the decision tree that models the comparisons made by insertion sort on n numbers. What is the shortest path in the tree and the longest path in the tree? Justify your answer using facts about the insertion sort algorithm.

- Consider the decision tree that models the comparisons made by selection sort on n numbers. How is this tree different from the insertion sort tree?
- Is it possible that a new comparison-based sorting algorithm exists that has a decision tree with a path that uses less than $n-1$ comparisons?
- Is it possible that a comparison-based algorithm for finding the median element has a decision tree with a path of less than $n-1$ comparisons?

Problem 11:

Given an unsorted list of n natural numbers in the range 0 to n^2, determine how to find the median element in each of the following ways:

1. Using Radix Sort
2. Using Counting Sort
3. Using the Select algorithm