Assignment 1

CS-GY 6033 CF01 Spring 2022

Due date: 11:55pm on **Feb. 26th, 2021 on Gradescope**.

Instructions:

Below you will find the questions which make up your homework. They are to be written out (or typed!) and handed in online via Gradescope before the deadline.

Question 1: Asymptotic Notation

.

(a) Rank the following functions in order (non-decresasing) of their asymptotic growth. Next to each function, write its big-Theta value, (ie. write the correct $\Theta(g(n))$ next to each function but you are not required to *prove* the big-Theta value).

$$\sqrt{\log n + (\log n)^2},\ n\log(n^{0.2n}),\ 10!n!,\ n^{0.2}(\log n)^2,\ n^{2n},\ (2^n{+}n)(\sqrt{n}{+}2^n),\ ,n(\log n)^3\ \frac{n^3\log n}{(\log n)^2+n},\ \sqrt{(\log n+1)}$$

(b) Consider the two sorting algorithms below, which each take as input array $A[]$ indexed from s to f.

```
SwapSort1(A, s, f)
    swapped = true
    while (swapped)
        swapped = false
        for i = s to f-2
            if A[i] > A[i+2]
                Swap A[i] and A[i+2]
                swapped = true
```

```
SwapSort2(A, s, f)
    swapped = true
    while (swapped)
        swapped = false
        for i = s to f-2
            if A[i] > A[i+2]
                Swap A[i] and A[i+2]
                swapped = true
    swapped = true
    while (swapped)
        swapped = false
        for i = s to f-1
            if A[i] > A[i+1]
                Swap A[i] and A[i+1]
                swapped = true
```

- Execute SwapSort1 on array $A = [7, 6, 5, 4, 3, 2, 1]$ indexed from $s = 1$ to $f = 7$.
- Which of the above two algorithms is *correct?*. Justify your answer.
- What is the worst-case number of comparisons are made by SwapSort2 when the input array A has length n and is sorted in reverse order?
- Does your result from above represent the **worst-case** number of comparisons made by SwapSort2? Justify your answer.
- Justify that the worst-case runtime of SwapSort2 is of the form $T(n) = an^2 + bn + c$ for constants a, b, c.

(c) A child is sorting her favorite toys in order of preference, from least favorite to most favorite. She has never learned about sorting algorithms. So she comes up with her own idea. She lays out all the n toys in a row from left to right on her bedroom floor. She starts by standing in front of the two left-most toys and carries out the following: if the two toys directly in front of her are in the **wrong** relative order, she swaps them, and then steps to the **left** (as long as there are toys on that side). Otherwise, she steps to the **right**. She repeats this step this until she reaches the right-most toy and doesn't carry out any swaps. Justify why her technique correctly sorts the toys. What algorithm from class is she executing?

**A figure is included at the end of the assigment*

(d) For each of the following $f(n)$, show that $f(n)$ is $\Theta(g(n))$ for the correct function $g(n)$. Prove your result using the definitions from class, including an explicit value for k justifying your statement is true for all $n \geq k$.

- $f(n) = n^{1.5}\log(2n) + n^2\log(n^2) + \sqrt{n}$
- $f(n) = 2^n \cdot n^2 + 100 \cdot 3^n + n^4$

Question 2: Recurrences

(a) Below is the pseudo-code for two algorithms: Practice1(A,s,f) and Practice2(A,s,f), which take as input a **sorted** array A, indexed from s to f. The algorithms make a call to *Bsearch(A,s,f,k)* which we saw in class. Determine the worst-case runtime recurrence for each algorithm: $T_1(n)$ and $T_2(n)$. Show that $T_1(n)$ is $O(\log n)$ and $T_2(n)$ is $O(n \log n)$.

```
Practice1(A,s,f)
    if s < f
        q1 = ⌊(s + f)/2⌋
        if BSearch(A,s,q1,1) = true
            return true
        else
            return Practice1(A, q1+1, f)
    else
        return false
```

```
Practice2(A,s,f)
    if s < f
        if BSearch(A,s+1,f,1) = true
            return true
        else
            return Practice2(A, s, f-1)
    else
        return false
```

(b) Apply the master theorem to to each of the following, or state that it does not apply:

- $T(n) = T(n/3) + n\log n$
- $T(n) = 16T(n/4) + n^{1.5}\log n$
- $T(n) = 4T(n/16) + \sqrt{n}$.

(c) The recursive algorithm below takes as input an array A of distinct integers, indexed between s and f, and an integer k. The algorithm returns the **index** of the integer k in the array A, or -1 if the integer k is not contained within A. Complete the missing portion of the algorithm in such a way that you make **three** recursive calls to subarrays of approximately one third the size of A.

- Write and justify a recurrence for the runtime $T(n)$ of the above algorithm.
- Use the recursion tree to show that the algorithm runs in time $O(n)$.

```
FindK(A,s,f,k)
    if s < f
        if f = s + 1
            if k = A[s] return s
            if k = A[f] return f
        else
            q1 = ⌊(2s + f)/3⌋
            q2 = ⌊(q1 + 1 + f)/2⌋
             ... to be continued.
    else
        ... to be continued.
```

Question 3: Hashing

.

(a) Consider a table indexed from $0 \ldots 12$ and where hashing is carried out using quadratic probing, using the function $h(k, i) = k + 4i + 2i^2 \mod 13$. Show the result of the insertion of the following keys: $38, 16, 50, 7, 18, 19, 12, 37, 29, 22$.

(b) Using the quadratic probe sequence from part (a), write the pseudo-code for an algorithm called HashInsert(T, k) that takes as input a hash table $T[0, \ldots, 12]$ and a key k to be inserted into the table. Your algorithm must insert the item k into the table, using the probe sequence $h(k, i)$ and return **true** if the insertion is successful, and **false** otherwise.

**You do not have to check for duplicates (you can assume when inserting k that it is not already in the table). You can assume that the table was initially empty and only inserts were carried out, no deletions*

(c) Consider arrays A and B, each of which is indexed from 1 to n. The array entries are bank customer PIN numbers, where a PIN number is a sequence of exactly 3 digits (ex. 460 or 013 are each valid PINs). Array A consists of the PIN numbers of customers from bank A, and array B consists of the PIN numbers of customers from bank B. Your job is to design an algorithm that outputs all **safe** passwords from bank A. The **safe** passwords from bank A are those which are used by a customer from bank A but **not also used** by a customer in bank B. Write the pseudo-code for your algorithm, and justify why it runs in time $O(n)$ in the worst-case.

Question 4: Selection

.

(a) Carry out the *Select* algorithm on the following set, using $k = 19$ (return the element of rank 19). Show your steps! Ensure that you run the partition algorithm in such a way that you maintain the elements in their original relative order.

$$56, 78, 34, 19, 67, 32, 13, 12, 90, 92, 50, 51, 30, 1, 99, 58, 43, 42, 24, 65, 21, 25, 68, 69, 101$$

(b) Carry out the *Randomized Select* algorithm from class on the set of elements from part (a), using $k = 19$. Show your steps! When selecting a random pivot, suppose you always chose the *last element* in the array. Ensure that you run the partition algorithm in such a way that you maintain the elements in their original relative order

(c) Suppose you are given a set of 100m sprint times from the 2020 olympics. The input list has size n and is not sorted, and you may assume that each time is distinct (there are no exact ties). Your team-mate completed in the event, and her time was exactly 10.57 seconds. She returns from the olympics upset, claiming that at least half the competitors had a faster time. Design an algorithm that takes as input the array A consisting of the n sprint times, and outputs the closest $n/4$ sprint times that are *faster* than your friend's time. Justify why your algorithm runs in $O(n)$ in the worst case.

**For simplicity, you may assume $n/4$ is an integer*

Question 5: Heaps

.

(a) Suppose that we would like to verify if the elements in array A satisfy the heap property, assuming the usual heap implementation shown in class. Write the pseudo-code for a recursive algorithm called VerifyHeap(A), which returns **true** if A is indeed a valid max-heap, and **false** otherwise. You may use the attribute $A.heapsize$. Write the recurrence for the runtime of your algorithm, and justify the worst-case runtime of $O(n)$ and the best-case of $O(1)$.

(b) A max-heap is contained in array A, using the implementation as shown in class. However, the array contains an error, in that the entry at index k is empty. Instead of rebuilding the entire heap from scratch, design an algorithm that repairs the heap. The result of your algorithm must contain a valid heap in array A with a correct $A.heapsize$ attribute. Write the pseudo-code for your algorithm, which must be called RepairHeap(A,k). Justify the worst-case runtime of $O(\log n)$ and the best-case of $O(1)$.

(c) At the 2020 Tokyo Olympics, there are n athletes competing in the 100m sprint. The organizers have planned for exactly 5 heats, with each heat containing $n/5$ athletes. The finish times of *each* heat will be stored in a min-heap: $H1, H2, H3, H4, H5$. For example, $H1$ is a min-heap containing the finish times of the athletes from the first heat. Once the heats are complete, the organizers must determine the top 40 athletes who will continue to the quarter-finals. Design an algorithm that takes as input the five heaps, and outputs the best 40 finish times, over *all* athletes, regardless of heat. Call your algorithm Top40(H1, H2, H3, H4, H5) and provide the pseudo-code. Justify the runtime of $O(\log n)$.

Question 6: Lower Bounds and Linear time Sorting

. **(a)** A set of n natural numbers are uniformly distributed in the range $1 \leq x < \sqrt{n}$. Determine the runtime (in big-Theta notation) of Counting Sort and Radix Sort. Find the expected runtime of Bucket sort using 10 buckets. Which algorithm has the best asymptotic runtime?

(b) Given a set of three numbers $\{a, b, c\}$ draw the comparison-based decision tree that represents the execution of bubble-sort. Justify the length of the longest-path in your tree, using the results of problem 5 from practice set 1.

Figure for problem 1c: Below shows one step of the girls sorting process. Examining bears 5 and 1, she decides they need to be swapped. Then she moves left and will examine bears 4 and 1 next.