Assignment 1

CS-GY 6033 INET Spring 2025

Due date: Feb 14th 11:55pm on 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) 10 points 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). Remember that the default base for *log* is base 2.

$$\log(50),\ (2^{10}+1)(2^{10}+n),\ \frac{n^2+\log n}{\sqrt{n}+\log n},\ 4^{2n+1},\ \sqrt{\log n+1},\ 4^{n/2+1},\ (1^n+2^n)(3^n+4^n),$$

$$(\log 2^n)\ ,\ 2^{\log n},\ (\log(n/2))^2$$

(b) 8 points Determine if each of the following statements are true or false. If the statement is false, provide a counter example. If the statement is true, justify the statement using the formal definitions from class.

1. If $f(n)$ is $O(n\log n)$, does this imply that $f(n)$ is also $\Omega(n^2)$?
2. If $f(n)$ is $O(n^2)$, does this imply that $f(n)$ is also $O(2^n)$?

(c) 16 points 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, justifying your statement is true for all $n \geq k$. (provide the value of k).

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

Question 2: Sorting Algorithms:

.

(a) 8 points Consider the problem of sorting a list $L = (a, b, c, d)$ of three numbers. Determine the worst-case number of comparisons (the exact number!) made between elements of L for each of the following sorting algorithms: Insert Sort, and Mergesort. Ensure that your answer is an exact number, not something with an n in it.

(b) 8 points Consider the following input : 3568792140. Below are four sequences of numbers, where each sequence represents an intermediary stage of of a sorting algorithm. The intermediary stage could be *any* intermediate state of the algorithm execution. For each sequence, you must determine which sorting algorithm is being used. You must select from: Selection sort, Insertion sort, Bubble Sort, and MergeSort.

You must justify in your answer *why* you excluded three of the options. Note that each option is used exactly once.

3568792140
0568792143
3567892140
3567821409

(c) 6 points Consider the following proposed sorting algorithm called MySort(A, s, f), which takes as input array A indexed between s and f. Note that the algorithm makes a call to an updated version of the Merge algorithm, called NewMerge(A, s, q, f), which merges the sorted lists from $A[s, q]$ and $A[q + 1, f]$. The original Merge code is copied below, and the new updates are shown in purple.

```
MySort(A, s, f)
    if s < f
        q = round-down((s+f)/2)
        MySort(A, s, q)
        MySort(A, q+1,f)
        NewMerge(A,s,q,f)
```

```
NewMerge(A, s, q, f)
  if A[q] < A[q + 1]
    return
  n = f − s + 1
  Initialise L[1 . . . , n] and R[1, . . . , n] to INFINITY
  Copy elements from A between s and q into array L[]
  Copy elements from A between q + 1 and f into array R[]
  i = 1, j = 1
  for k = s to f do:
    if L[i] < R[j]
      A[k] = L[i]
      i++
    else
      A[k] = R[j]
      j++
```

1. Does the above procedure correctly sort all input arrays A of at least one element?
2. What is the best-case number of comparisons carried out by this version of MergeSort, where input array A has n elements?

Question 3: Recurrences

(a) 6 points Consider the the pseudo-code below for the recursive algorithm FunnyRun(A, s, f), which takes as input an array A, indexed between s and f.

```
FunnyRun(A,s,f)
  if 4 < f − s
    q1 = ⌊(2s + f)/3⌋
    q2 = ⌊(s + 2f)/3⌋
    FunnyRun(A, q1, f)
    FunnyRun(A, s, q2)
    FunnyRun(A, q1, f)
  else
    InsertionSort(A, s, f)
```

Your job:
Write and justify the runtime recurrence for the above algorithm.
Use Master Method to find a tight upper bound for the runtime of the algorithm.
Does the procedure correctly sort the input $4, 3, 5, 1, 6, 2$? You must justify your answer.

(b) 8 points Below is the pseudo-code for an algorithm that takes as input an array of positive integers, A, indexed between s and f. The algorithm prints out certain elements of the array.

```
SillyPrint(A,s,f)
    if 1 ≤ f − s
        q1 = ⌊(s + f)/2⌋
        print A[q1]
        SillyPrint(A,q1+1,f)
        print A[q1]
    else
        print A[s]
```

- Execute the algorithm on the input $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]$.
- Find and justify the runtime recurrence for the algorithm: $T(n)$.
- Use the above recurrence to show that $T(n)$ is $O(\log n)$ using substitution method.

(c) 12 points Use the recursion tree to find a tight asymptotic bound for each of :

- $T(n) = 2T(n/4) + n$
- $T(n) = 2T(n/4) + 1$
- $T(n) = 16T(n/4) + n$

(d) 8 points Apply the master theorem to to each of the following, or state that it does not apply:

- $T(n) = 6T(n/2) + n^2 \log n$
- $T(n) = 16T(n/4) + \log n$
- $T(n) = 6T(n/4) + n^2$