Quiz #7: Analysis of Algorithms/Complexity Classes ICS-33 Winter 2017

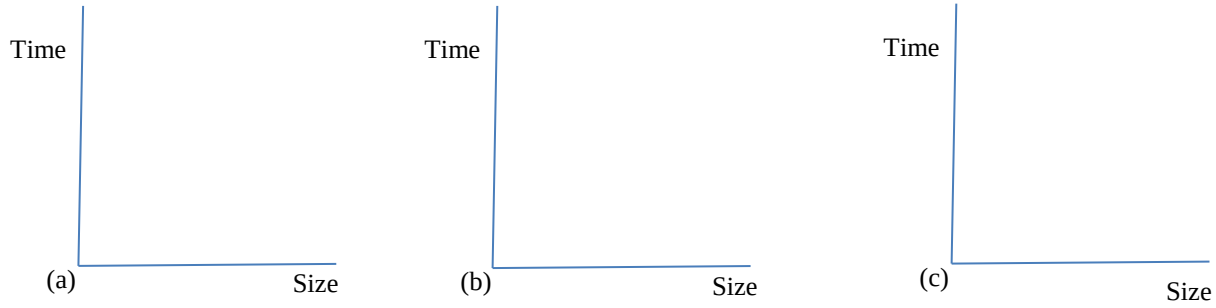
Name (Last, First) _____

Lab # _____

When working on this quiz, recall the rules stated on the Academic Integrity statement that you signed. There is no helper project file for this assignment. Submit your completed written exam as the start of lecture on Friday. If you print a copy of this document it should be **one page, 2-sided** (no staples, folding, etc.) or you will lose a bit of credit. I will post my solutions to EEE reachable via the **Solutions** link on Friday after class.

---

1. (3 pts) Sketch approximate Size vs. Time curves for the two algorithmic complexity classes required in each of the pictures below: for one, write **Impossible** instead: (a) an O(N) algorithm that is **always** faster than an $O(N^2)$ algorithm.. (b) an O(N) algorithm that is **never** faster than an $O(N^2)$ algorithm. (c) an O(N) algorithm that is **sometimes** faster than an $O(N^2)$ algorithm.




2. (2 pts) (a) What "better-known"/simpler complexity class is equivalent to **$O(N \log N^2)$**; briefly explain why. (b) Explain why it makes little sense for an algorithm to be in the complexity class **O(1/n)**?

(a)

(b)

3. (6 pts) Assume that functions **f1** and **f2** compute the same result by processing the same argument. Empirically we find that **$T_{f1}(N) = 110\ N$** and **$T_{f2}(N) = 10\ N \log_2 N$** where the times are in seconds. (a) Solve algebraically what size **N** would these two functions take the same amount of time? Show how you **calculated** your answer. (b) for what size arguments is it better to use `f1`? `f2`? (c) Briefly describe how we can write a simple function **f** that runs as fast as the fastest of **f1** and **f2** for all size inputs. (d1) What exact integer value **N** (±1) solves **$23\sqrt{N} = 10\ (\log_2 N)^2 + 1000$**? Use a calculator, spreadsheet, or a program to guess and refine your answer (try plotting values to see where the curves meet). (d2) Based on your calculation, which complexity class $O(\sqrt{N})$ or **$O((\log_2 N)^2)$** is lower?

(a)

(b) `f1` is faster for all **N …**
`f2` is faster for all **N …**

(c)

(d1)

(d2)

4. (6 pts) The following functions each determine if any two values in **alist** sum to **asum**. As is shown in the notes, (a) write the complexity class of each statement on its **right**, where **N** is **len(alist)**. (b) Write the **full calculation** that computes the complexity class for the entire function. (c) Simplify what you wrote in (b).

```
def sums_to_1 (alist,asum):
    for f in alist:
        for s in alist:
            if f+s == asum:
                return (f,s)
    return None
```

(b)

(c)

```
def sumsto_2 (alist,asum):
    aset = set(alist)
    for v in alist:
        if asum-v in aset
            return(v,asum-v)
    return None
```

(b)

(c)

5. (4 pts) Assume that function **f** is in the complexity class **O(N $(\log_2 N)^2$)**, and that for **N = 1,000** the program runs in **9 seconds**.

(1) Write a formula, **T(N)** that computes the approximate time that it takes to run **f** for any input of size **N**. Show your work/calculations by hand, approximating logarithms, then finish/simplify all the arithmetic.

(2) Compute how long it will take to run when **N = 1,000,000 (**which is also written $\mathbf{10^6}$). Show your work/calculations by hand, approximating logarithms, finish/simplify all the arithmetic.

6. (3 pts) Fill in the last line of the three empty rows, which shows the size of a problem can be solved in the **same amount of time** for each complexity class on a new machine that **runs four times as fast as the old one**. Solve algebraically when you can, use Excel or a calculator when you must: I used a calculator only for **O(N $\log_2$ N)** and solved it to 3 significant digits. Solving a problem in the same amount of time on the new machine is equivalent to solving a problem that takes four times the amount of time on the old machine. See **O(N)** for an example.

| N = Problem Size | Complexity Class | Time to Solve on Old Machine (secs) | N Solvable in the same Time on a New Machine 4x as Fast |
|---|---|---|---|
| $10^6$ | O($\log_2$ N) | 1 | |
| $10^6$ | O(N) | 20 | $4 \times 10^6$ |
| $10^6$ | O(N $\log_2$ N) | 20 | |
| $10^5$ | O($N^2$) | 1000 | |