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

Name (Last, First) _____

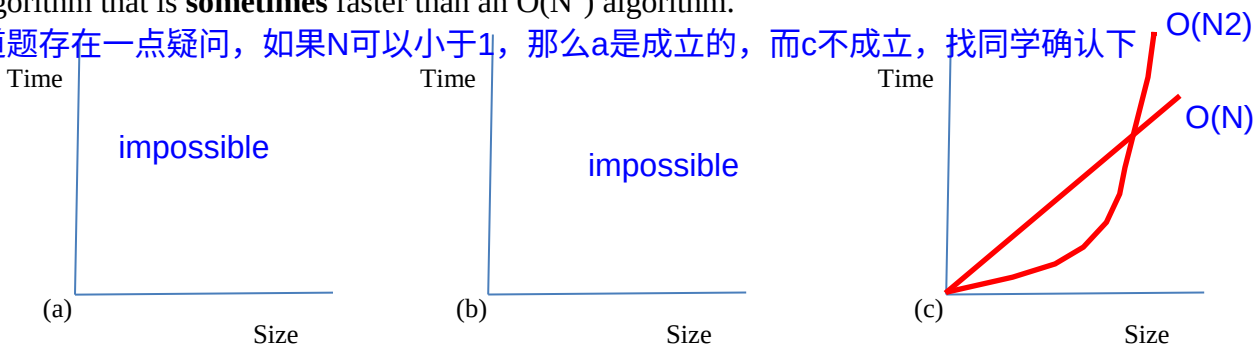
Lab # _____

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 some 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 **never** faster than an $O(N^2)$ algorithm. (b) an O(N) algorithm that is **always** faster than an $O(N^2)$ algorithm. (c) an O(N) algorithm that is **sometimes** faster than an $O(N^2)$ algorithm.

这道题存在一点疑问，如果N可以小于1，那么a是成立的，而c不成立，找同学确认下



2. (3 pts) (a) When determining the complexity class of code from the formula showing its actual number of instructions executed in the worst case, what is kept? What is discarded? (b) Two functions in the identical complexity class will take different amounts of running time when run on the same computer. Explain why.

(a) Kept : the biggest term

Discarded: all the lower (more slowly growing) terms and the constant in front of the most frequently executed statement (the fastest growing term).

(b) cpu shcedule ,cache and other factors may affect the running time

3. (6 pts) Assume that functions **f1** and **f2** compute the same result by processing the same argument. Empirically we find that $\mathbf{T_{f1}(N) = 10\,N\,\log_2 N}$ and $\mathbf{T_{f2}(N) = 100N}$ where the times are in seconds. (a) Solve algebraically for what size **N** these two functions would take the same amount of time, showing 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. (d) What integer value **N** solves $\mathbf{78\sqrt{N}\,\log_2 N + 10{,}000 = 2N}$? Use a calculator, spreadsheet, or a program to compute (guess and refine) your answer (plotting values to see where the curves meet). Your answer should be correct for all digits up to the ones-place: e.g., a number like 23,728.

(a) N=1024

(b) `f1` is faster for … N<1024

`f2` is faster for … N>1024

(c) Write a function **f** … 这道题不会

(d) **N** ~ 565341

4. (6 pts) The following two functions each determine the distance between the two closest values in list `l`, with `len(l)` = `N`. (a) Write the complexity class of each statement in the box on its right. (b) Write the **full calculation** that computes the complexity class for the entire function. (c) Simplify what you wrote in (b).

```
def closest(l:[int])->int:
  a = set()                                  O(1)
  for i in range(len(l)):                    O(N)
    for j in range(len(l)):                  O(N)
      if i != j:                             O(1)
        a.add(abs(l[i]-l[j]))                O(1)
  return min(a)                              O(1)
```

(b) 1+N*(N+1+1) + 1

(c) O(N^2)

```
def closest(l:[int])->int:
  a = sorted(l)                              O(NlogN)
  min = None                                 O(1)
  for i in range(len(a)-1):                  O(N)
    if min==None or a[i+1]-a[i]<min:         O(1)
      min = a[i+1]-a[i]                      O(1)
  return min;                                O(1)
```

(b) NlogN + 1+N*(1+1)+1

(c) O(NlogN)

5. (4 pts) Assume that function **f** is in the complexity class **O**($\sqrt{N}\ Log_2 N$), and that for **N** = **1,000,000** the program runs in **.06 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, finish/simplify all the arithmetic.

T(N) = $\sqrt{N}\ Log_2 N \times 3 \times 10^{-5}$

(2) Compute how long it will take to run when **N** = **4,000,000**. Show your work/calculations by hand, approximating logarithms, finish/simplify all the arithmetic.

T(N)=1.32 seconds

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 nine as fast as the old one**. Solve by hand 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/faster machine is equivalent to solving a problem that takes nine 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 9x as Fast
10^6	O(Log$_2$ N)	1	179
10^6	O(N)	1	9×10^6
10^6	O(N Log$_2$ N)	1	1.61 x 10^9
10^6	O(N^2)	1	9*10^12