Name (Print): Dang Nhi Ngoc Ngo

Problem Number(s)	Possible Points	Earned Points
1	25	20
2 (1)	15	15
2 (2)	15	15
3	20	~~15~~ 20
4 (1)	15	5
4 (2)	10	4
	TOTAL POINTS 100	

Exam Time: 50 minutes, 4 problems (8 pages, including this page)

- Print your name on this page and the last page, put your initials on the rest of the pages.
- This exam is close book and notes, and **Closed** neighbors, mobile devices, and internet. Only one-page of cheat sheet is allowed.
- If needed, use the back of each page or the last page.
- Show your work to get partial credits.
- Show your rational if asked. Just giving an answer can't give you full credits.
- You may use any algorithms (procedures) that we've learned in the class.
- Keep the answers as brief and clear as possible.

Name (Print): Dang Nhi Ngoc Ngo

1. (25 points) **Asymptotic Growth of Functions and Notations** (no partial credit)

Clues:

(1) $f_1(n) \in \Omega(3^{-n})$ (2) $f_2(n) \in \Omega(n \lg^2 n)$ (3) $f_3(n) \in O(3n+3)$

(4) $f_4(n) \in O(\lg\lg(8^n))$ (5) $f_5(n) \in \Theta((2^{\lg n})^3)$

Circle TRUE (the statement must be always TRUE based on the clues above) or circle FALSE otherwise.

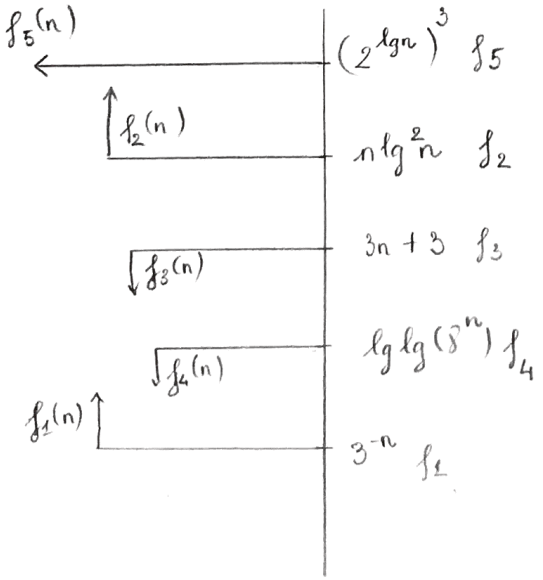
(a) $f_2(n) \in \Theta(f_5(n))$ (TRUE) FALSE

(c) $f_2(n) \in \Omega(f_1(n))$ TRUE (FALSE)

(d) $f_1(n) \in O(\lg\lg(8^n))$ $f_4(n)$ TRUE (FALSE)

(b) $f_3(n) \in O(f_2(n))$ (TRUE) FALSE

(e) $f_4(n) \in O(f_3(n))$ TRUE (FALSE)



Name (Print): Dang Nhi Ngo

2. (30 points) **Recurrence**

Derive a recurrence for the running time of each algorithm below and then solve the recurrence with one of the three methods we have learned. The solution should be a tight upper and lower bound solution. You must show the work of how to solve the recurrence. You do NOT need to write the algorithm.

(1) An algorithm solves a problem of size n by recursively processing ¾ of n elements, and dividing takes $\Theta(n)$ time.

$$T(n) = T\left(\frac{3n}{4}\right) + \Theta(n)$$

$a = 1$, $b = \frac{4}{3}$ $\Rightarrow \log_b a = \log_{\frac{4}{3}} 1 = 0$

Compare $n^{\log_b a} = n^0 = 1$ vs. $f(n) = \Theta(n) = n$

$\Rightarrow$ $f(n)$ is polynomially greater than $n^{\log_b a}$ ($n > 1$) !?

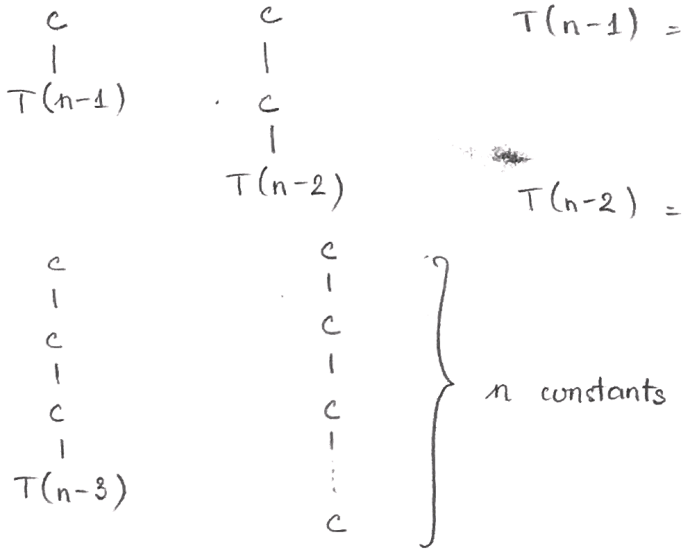
$\Rightarrow$ Case 3 in Master Method

$T(n) = \Theta(n)$

Name (Print): Dang Nhi Ngo

(2) An algorithm solves a problem of size n by recursively solving one sub-problems of size (n – 1), and then combining the solutions in constant time.

$T(n) = T(n-1) + c$ (c: positive constant)



$T(n-1) = T(n-2) + c$

$T(n-2) = T(n-3) + c$

$T(n) = \underbrace{c + c + \dots + c}_{n \text{ constants}}$

$= c \cdot n = \Theta(n)$

Name (Print): Dang Nhi Ngo

3. **(20 points) Substitution method**
Use the substitution method to prove that recurrence T(n) = 3T(n/3) + cn (where c is a positive constant) is in the tight bound of *nlgn*. Please prove for both upper bound and lower bound.

$$T(n) = 3T\left(\frac{n}{3}\right) + cn$$

- Upper bound: $T(n) \leq 3T\left(\frac{n}{3}\right) + cn$ (c: positive constant)

  Guess: $T(n) = O(n.\lg n)$

  $\leq d(n.\lg n)$ (d: positive constant)

  $T\left(\frac{n}{3}\right) \leq d.\frac{n}{3}\lg\frac{n}{3}$

  $T\left(\frac{n}{3}\right) \leq \frac{1}{3}d.n\lg\left(\frac{n}{3}\right)$

  Substitution: $T(n) \leq 3.\frac{1}{3}d.n.\lg\left(\frac{n}{3}\right) + c.n$

  $= d.n(\lg n - \lg 3) + cn$

  $= dn\lg n - d\lg 3n + cn$

  $\leq dn\lg n$ (if $(-d\lg 3n + cn) \leq 0$)

  $\Rightarrow d\lg 3.n \geq cn$

  $\Rightarrow d \geq c/\lg 3$

  Therefore, $T(n) = O(n\lg n)$ (1)

- Lower bound: $T(n) \geq 3T\left(\frac{n}{3}\right) + cn$ (c: positive constant)

  Guess: $T(n) = \Omega(n\lg n)$

  $\geq d.n.\lg n$ (d: positive constant)

  $T\left(\frac{n}{3}\right) \geq d.\frac{n}{3}\lg\left(\frac{n}{3}\right)$

  $T\left(\frac{n}{3}\right) \geq \frac{1}{3}.d.n.\lg\left(\frac{n}{3}\right)$

  Substitution: $T(n) \geq 3.\frac{1}{3}.dn.\lg\left(\frac{n}{3}\right) + cn$

  $= d.n(\lg n - \lg 3) + cn$

  $= d.n\lg n - d\lg 3.n + cn$

  $\geq dn\lg n$ (if $(-d\lg 3n + cn) \geq 0$)

  $\Rightarrow d\lg 3.n \leq cn$

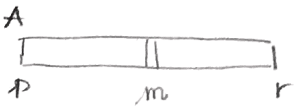
  $\Rightarrow d \leq c/\lg 3$

Name (Print): Dang Nhi Ngo

4. **(25 points) Design and analysis of an Algorithm**
You are given an array *A*, which stores *n* non-negative integers. Design a **divide-and-conquer** algorithm that accepts *A* and *n* as inputs only and returns the **index of the maximum value** in the array *A*. You may use a helper function if needed.

(1) (15 points) **Pseudocode**: *(please use textbook conventions)*

```
Find_index (A, n)
    return Find_helper (A, p, r)
```




```
Find_helper (A, p, r)                                   Cost
1. if (p == r)                                          c1
2.     return (A[p] == p)                               c2
3.  m = ⌊(p + r)/2⌋                                     c3
4.  if (A[m] < m)                                       c4
5.      return Find_helper (A, p, m-1)                  T(n/2)
6.  else return Find_helper (A, m+1, r)                 T(n/2)
```

Can't solve the given problem!

Name (Print): _____

(2) (10 points) **Analysis:** Derive a recurrence for the running time of your algorithm (**You don't need to solve the recurrence**). Justify your answer by listing the cost for executing each line of code and the number of executions for each line. You may mark the cost with your algorithm on the previous page or specify the cost using the line numbers on this page.

4

Can 4 simply add together!

$T(n) = c1 + c2 + c3 + c4 + T(\frac{n}{2})$

$= T(\frac{n}{2}) + c$

$= T(\frac{n}{2}) + \theta(1)$ ..

Case 2 in Master Method ($a = 1, b = 2 \Rightarrow \log_b a = \log_2 1 = 0$)

$T(n) = \theta(\lg n)$

Name (Print): Dang Nhi Ngo

(You may use this page to write answers if needed. Please mark the problem number clearly)