Design and Analysis of Algorithms (COMP3001)

Tutorial 1 and 2
Mathematical Preliminaries,
and Algorithm Analysis

Question 1.

This question is designed to make you think about the growth rates of different functions for increasing input size.

a) Sketch the graph of $\log_2 n$, n, n^2, and 2^n for $n > 0$. Notice the significant differences in the growth of the functions when n increases.

b) Exercise 1.2-2 (Textbook: Cormen, *et al.*)
Suppose we are comparing implementations of insertion sort and merge sort on the same machine. For inputs of size n, insertion sort runs in $8n^2$, while merge sort runs in $64n \lg n$ steps. For which values of n does insertion sort beat merge sort?

HINT: You can solve this by graphing both functions and finding the intersection

c) Exercise 1.2-3 (Textbook: Cormen, *et al.*)
What is the smallest value of n such that an algorithm whose running time is $100\,n^2$ runs faster than an algorithm whose running time is 2^n on the same machine?

Question 2.

For the following Java method:

a) Describe its best case running time scenario, and calculate its best case asymptotic time complexity.

b) Describe its worst case running time scenario, and calculate its worst case asymptotic time complexity.

State any assumptions you make, if any, and why.

```
public void traverse(Node t)
{
        if (t != null)
        {
                traverse(t.leftChild);
                System.out.println(t.getData());
                traverse(t.rightChild);
        }
}
```

Question 3.

While induction proofs are important in themselves, both of these identities are very useful in complexity analysis, where summation is often used.

Use induction to prove the following.

a) $\sum_{i=1}^{n} i = n(n+1)/2$. **Note:** the proof for this equation has been discussed in the lecture

b) $\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6$.

c) $\sum_{i=1}^{n} 2^{i-1} = 2^n - 1$

Question 4.

This is now getting closer to the goal of actual complexity bounds analysis. Having calculated the actual complexity, you need to be able to derive the bounds.

a) Express each of the following functions in terms of O, Ω, and Θ. For each, prove that your answer is correct.

- $n^3/1000 - 100n^2 - 100n + 3$
- $\sum_{i=1}^{n} i$
- $\sum_{i=1}^{n} i^2$

b) Prove that $f(x) = 10 + 100 \log_2(x-3)$ is $O(\log x)$.
Hint: $\log_2(x-3) \leq \log_2(x)$, for $x > 3$.

c) Prove that $f(x) = 10 + 100 \log_2(x-3)$ is $\Omega(\log x)$.
Hint: $\log_2(x-3) \geq \log_2(x/2)$, for $x \geq 6$.

Question 5.

This question is somewhat more challenging. Feel free to work with friends or to look for guidance elsewhere, but make sure that you understand any answers obtained.

By Stirling's approximation, $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$

a) Use the Stirling's approximation to derive an expression for $\log_2 n!$.

b) Use the result in a) to prove that $\sum_{i=1}^{n} \log_2 i = O(n \log n)$.

Question 6.

This question puts everything together. Remember, in real life you'll be doing this for a whole program, and not just fragments.

Analyse the running time of each of the following program fragments assuming a RAM model of computation. For each analysis do the following.

(i) Calculate how many times each line of the program is executed.
(ii) Calculate the total number of steps to execute the program.
(iii) Express the result in (ii) in Big Oh.
(iv) Show how to compute the Big Oh in (iii).

State any assumptions or definitions you make.

a)
```
for (i = 0; i < n ; i++)
    System.out.println ("hello world");
```

b)
```
for i ← n downto 0 do
    for j ← 0 to i do
        A[i][j] ← A[i][j + 1] + 2
```

c)
```
i ← n
while i > 0 do
    set i ← i/2
```

Note: The "$i/2$" is an integer division, e.g., 7/2 = 3.

d)
```
for i ← 1 to n do
    j ← i
    while j > 0 do
        set j ← j/2
```

Note: The "$j/2$" is an integer division, e.g., 11/2 = 5.

Question 7.

This question may require some discussion with others in the unit.

Prove $\sum_{i=1}^{n} \Theta(i) = \Theta(n^2)$.

Question 8.

Use the master method to give tight asymptotic bounds for the following recurrences.

a) $T(n) = 4T(n/2) + n$

b) $T(n) = 4T(n/2) + n^2$

c) $T(n) = 4T(n/2) + n^3$

d) $T(n) = 3T(\sqrt[2]{n}) + log_2\, n$

Hint: The function is not in a form suitable for the master method. However, it can be converted to the required form for master method by using another variable $k = \log_2 n$. For details, read page 86 of the textbook (third edition).

Question 9.

a) Guess the solution of recurrence function $T(n) = T(\lceil n/2 \rceil) + 1$ using a recursion tree, and prove by induction that your guess is correct.

Hint. $T(n) = O(\lg n)$.

b) Guess the solution of the following recurrence relation using a recursion tree, and prove your result by induction:

$$P(n) = \begin{cases} O(1) & ,if\ n = 1 \\ 2P(n/2) + O(n) & otherwise \end{cases}$$

Hint. $T(n) = O(n \lg n)$.

<u>Note:</u> You can verify your result using the master method

c) Guess the solution of recurrence function $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$ using a recursion tree, and prove by induction that your guess is correct.

Hint. $T(n) = O(n)$.

Question 10.

The solution of $T(n) = 2T(\lfloor n/2 \rfloor) + n$ is $O(n \lg n)$. Show that the solution of this recurrence is also $\Omega(n \lg n)$.

Hint. You can prove that the solution is $\theta(n \lg n)$.

Question 11.

The recurrence $T(n) = 7T(n/2) + n^2$ describes the running time of an algorithm A. A competing algorithm A' has a running time of $T'(n) = aT'(n/4) + n^2$. What is the largest value for a such that A' is asymptotically faster than A?

Question 12.

(Exercise 4.5-4). Can the master method be applied to the recurrence $T(n) = 4T(n/2) + n^2 \lg n$? Why or why not? Give an asymptotic upper bound for this recurrence.

Question 13.

Exercise 2.3-4 (Textbook).

Insertion sort can be expressed as a recursive procedure as follows. In order to sort A[1 .. n], we recursively sort A[1 .. n-1] and then insert A[n] into the sorted array A[1 .. n-1].

(i) Write a recurrence for the running time of this recursive version of insertion sort.

(ii) Solve the recurrence in (i) to find the running time of the procedure.

Question 14.

Analyse the time complexity of the following Euclid's algorithm to compute the greatest common divisor (GCD).

GCD (x, y)
if $y = 0$ **then return** x
return GCD (y, x **mod** y)

To analyse the function, answer the following questions.

(i) What is the problem size?
Hint. The size is the number of bits n to represent each integer. Why?

(ii) Compute the time complexity of the recursive function.
Hint. (x **mod** y) $\leq$ x / 2. So, what is the size of the problem after n? Is it $n/2$ or $n - 1$?

What is the time complexity of computing x **mod** y? You can assume $O(1)$.

You can then find the recurrence function of the time complexity of GCD, i.e., $T(n) = T(??) + O(1)$, and solve the recurrence function.

Simple Mathematic Functions

1) Floor function: $\lfloor x \rfloor$; example: $\lfloor 3.1 \rfloor = 3$, $\lfloor 3.7 \rfloor = 3$

2) Ceiling function: $\lceil x \rceil$; example: $\lceil 3.1 \rceil = 4$, $\lceil 3.7 \rceil = 4$

3) $x - 1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x + 1$

4) $a^0 = 1$; $a^{-1} = \frac{1}{a}$

5) $(a^m)^n = a^{mn}$

6) $a^m a^n = a^{m+n}$

7) $b^{\log_b a} = a$

8) $\log_c(ab) = \log_c a + \log_c b$

9) $\log_b a^n = n \log_b a$

10) $\log_b a = \frac{\log_c a}{\log_c b}$

11) $\log_b(1/a) = -\log_b a$

12) $\log_b a = \frac{1}{\log_a b}$

13) $a^{\log_b c} = c^{\log_b a}$

14) $n! = 1 * 2 * 3 * 4 * \ldots * (n-1) * n$

15) Geometric series:
$x^0 + x^1 + x^2 + \ldots + x^{h-1} + x^h = (x^{h+1} - 1) / (x - 1)$

<u>For x=2:</u>
$2^0 + 2^1 + 2^2 + \ldots + 2^{h-1} + 2^h = 2^{h+1} - 1$