

COMP3506/COMP7505—Algorithms and Data Structures

School of Information Technology and Electrical Engineering

Week 3 Tutorial Sample Solutions

Question 1.

The logarithm of a number to a given base is the power or exponent to which the base must be raised in order to produce the number. (<http://en.wikipedia.org/wiki/Logarithm>)

$$\text{if } x = b^y, \text{ then } y = \log_b(x)$$

Determine the logarithms of the following numbers to the base 10.

- (a) $\log_{10}(1000) = 3$
- (b) $\log_{10}(100) = 2$
- (c) $\log_{10}(10) = 1$
- (d) $\log_{10}(1) = 0$

Determine the logarithms of the following numbers to the base 2

- (e) $\log_2(1) = 0$
- (f) $\log_2(2) = 1$
- (g) $\log_2(4) = 2$
- (h) $\log_2(8) = 3$
- (i) $\log_2(16) = 4$

Note that taking the logarithm of an exponential sequence of numbers yields a linear sequence of numbers.

Question 2.

Analyse the running time of the following function for arbitrary values of the input parameter n .

Algorithm: PrintIntegers(n)

Input: An integer giving the number of integers to print

Output: Printed integers, no return value

```
i = 0
while i < n do
    i += 1
    println i
return
```

Running time is $n \log n$. **println** displays an integer on each loop iteration. The time taken to display an integer is proportional to the number of characters in the string representation of the integer, which in a base-10 representation is going to be proportional to $\log_{10} i$, and i maxes out at n .

Question 3.

Analyse the running time of the Algorithm BinarySum (below) for arbitrary values of the input parameter n .

Algorithm: BinarySum(A, i, n)

Input: An array A and integers i and n .

Output: The sum of the n integers in A starting at index i .

if $n = 1$ **then**

return $A[i]$

return BinarySum($A, i, \lceil n/2 \rceil$) + BinarySum($A, i + \lceil n/2 \rceil, \lfloor n/2 \rfloor$)

Binary Sum will be called until called on each of the n individual members of A :
 $k = 1 + 2 + 4 + 8 + \dots + n$ times

One may recognise this as a geometric series that simplifies as:

$$1 + 2 + 4 + 8 + \dots + n = 2n - 1$$

Therefore BinarySum is called, $2n - 1$ times.

Each execution of Binary Sum runs in constant time.

Therefore the running time of Binary Sum is still $O(n)$

Why then might you want to divide the work in this manner?

Question 4.

Order the following functions by asymptotic growth rate:

$4 n \log n + 2 n$	2^{10}	$2^{\log n}$
$3 n + 100 \log n$	$4 n$	$2n$ (should be 2^n)
$n^2 + 10 n$	n^3	$n \log n$

Remembering,

$$\mathbf{b^c = a^{(c \log_a b)}}$$

$$\mathbf{n^1 = 2^{(1 \log_2 n)}}$$

Drop constants and lower order terms

- | | | | |
|-----|-------------------|--------------|-----------------|
| 1. | 2^{10} | $= 1024 = 0$ | $= O(1)$ |
| 2. | $2^{\log n}$ | $= n$ | $= O(n)$ |
| 3. | $2 n$ | $= n$ | $= O(n)$ |
| 4. | $4 n$ | $= n$ | $= O(n)$ |
| 5. | $3n + 100 \log n$ | $= n$ | $= O(n)$ |
| 6. | $n \log n$ | $= n \log n$ | $= O(n \log n)$ |
| 7. | $4 n \log n + 2n$ | $= n \log n$ | $= O(n \log n)$ |
| 8. | $n^2 + 10 n$ | $= n^2$ | $= O(n^2)$ |
| 9. | n^3 | $= n^3$ | $= O(n^3)$ |
| 10. | 2^n | $= 2^n$ | $= O(2^n)$ |

Question 5.

There is an algorithm, find2D, to find an element x in an $n \times n$ array A .

The algorithm find2D iterates over the rows of A , and calls the algorithm arrayFind (below) on each row, until x is found or it has searched all rows of A .

Algorithm: arrayFind(x, A):

Input: An element x and an n -element array, A .

Output: The index i such that $x = A[i]$ or -1 if no element of A is equal to x .

```
 $i \leftarrow 0$ 
while  $i < n$  do
    if  $x = A[i]$  then
        return  $i$ 
    else
         $i \leftarrow i + 1$ 
return -1
```

- a) What is the worst case running time of find2D in terms of n ?

The worst case running time of arrayFind is $O(n)$.

As find2D runs arrayFind on all n rows, the worst case running time of find2D, in terms of n , is $O(n^2)$.

- b) What is the worst case running time of find2D in terms of N , where N is the total size of A ?

Total size of array $A = N = n^2$

Therefore the worst case running time of find2D, in terms of N , is $O(N)$.

- c) Would it be correct to say that find2D is a linear-time algorithm? Why or why not?

find2D is a linear algorithm in terms of N . If the user code calling considered the area of the matrix to be more significant than the length of a side then the algorithm could be considered linear.

However, if the matrix were generated from input of size n , then the algorithm should be viewed as quadratic in nature.