Module 1: Algorithm Structures

1 With the pseudo-code below, determine the final value of `sum` when `n`=23.

Algorithm 1

```
procedure
  sum = 0
  for (i=2, i<n, i=i+3) do
    if i mod 2 == 0
      sum = sum + i
```

2 Use the pseudo-code below to answer the following question.

(a) Without working through the pseudo-code, how many times with `val` be updated in Line 7?

(b) What is the final value of `val`.

Algorithm 2

```
1: procedure
2:   dFour = [1, 2, 3, 4]
3:   dSix = [1, 2, 3, 4, 5, 6]
4:   val = 0
5:   for (x in dFour) do
6:     for (y in dSix) do
7:       val = val + x·y
```

3 Using the pseudo-code below, what is the ending value for `count` when `n = 100`? When `n = 10,000`? When `n` = 10^{42}?

Algorithm 3

```
procedure
  count=0
  while (n>1) do
    count++
    n = n/10
```

4 This question assumes familiarity with modular arithmetic discussed in Unit 2. In the pseudo-code below, Line 7 performs string addition (concatenation), for example:
`"a" + "b" = "ab"`. Even if the addends look like numbers, it will still do string addition:
`"2" + "3" = "23"`. Answer the questions below.

a. What is the output of the algorithm when $x = 99$ and $b = 2$?

b. What is the output of the algorithm when $x = 1642$ and $b = 8$?

Algorithm 4

```
1: procedure EXPAND(x, b)
2:    Input:  integers x and b
3:    Output:  outVal, x written in base b
4:
5:    outVal = an empty string
6:    while (x>0) do
7:       outVal = string(x mod b) + outVal
8:       x = x div b
9:    return(outVal)
```

Modules 2 & 3: Analyzing Algorithms and Big-$\mathcal{O}$ Estimates

Clarification of notation

The following statements all mean the same thing:

$f(x)$ is $\mathcal{O}(g(x))$ OR

$f(x)$ is of $\mathcal{O}(g(x))$ OR

$f(x) = \mathcal{O}(g(x))$ OR

$f(x) \in \mathcal{O}(g(x))$

$\mathcal{O}(g(x))$ is a collection of functions (i.e. a set) so what we should say is $f(x) \in \mathcal{O}(g(x))$, but $f(x) = \mathcal{O}(g(x))$ is commonly used. This can be confusing since $\mathcal{O}(n) = \mathcal{O}(n^2)$ but $\mathcal{O}(n^2) \neq \mathcal{O}(n)$!!

See this interesting thread on StackExchange Big O Notation is element of or is equal. Note the Wiki cites Donald Knuth.

Big-$\mathcal{O}$ classification of common functions

Order of Asymptotic Behaviour. Remember that these are sets. So while it might be said that $2n+3$ is $\mathcal{O}(n)$; actually its an element in that set: $2n+3 \in \mathcal{O}(n) \subset \mathcal{O}(n^2) \subset \ldots$.

Here's a list of some functions listed from slowest to fastest function growth (shortest to longest runtime for an algorithm).

- 1 - Constant
- $\log(n)$ - Logarithmic

  Ignore log bases: $\mathcal{O}(\log_{10}(n)) = \mathcal{O}(\log_2(n)) = \mathcal{O}(\log(n))$

  Same growth as $\mathcal{O}(\log(n^k)) = \mathcal{O}(k\log(n)) = \mathcal{O}(\log(n))$
- n - Linear
- $n\log(n)$ - Linearithmic or Loglinear
- n^2 - Quadratic
- $n^2\log(n)$
- n^3 - Polynomial (including $n^4, n^5, n^6, \ldots$)
- 2^n - Exponential (including $3^n, 4^n, 5^n \ldots$)
- $n!$ - Factorial

5 Order each function according to their growth from slowest to fastest (This is the same as execution speed from fastest to slowest for an algorithm).

(a) $n \cdot \log n$

(b) $\log_{10} n$

(c) 10^n

(d) n^{10}

(e) $10n$

(f) 100

(g) n^2

6 The provided expressions are the processing time of an algorithm for problems of size n. For each expression, find the *lowest possible* Big-$\mathcal{O}$ complexity stated in simplest form.

(a) $100n^2 + 0.0002n^3 + 10,000$

(b) $20n + 2n \log_{10} n + 30$

(c) $\log_{10} 5 + \log_2 7$

(d) $5n + \sqrt{n}$

(e) $0.01(2^n) + n^5 + \ln n$

(f) $100 \cdot 4^n + 200 \cdot 3^n$

7 $f(x) = 6x \log_4 x$ is which of the following? **Mark all that apply**.

○ $\mathcal{O}(6)$ ○ $\mathcal{O}(\log x)$ ○ $\mathcal{O}(6 \log x)$ ○ $\mathcal{O}(x \log x)$ ○ $\mathcal{O}(x^2)$ ○ $\mathcal{O}(2^x)$

8 Let $h(x) = 6x^4 + 2x^3 + x + 100$. Which of the following is true? **Mark all that apply**.

○ $h(x) \in \mathcal{O}(x^4)$ ○ $h(x) \in \Theta(x^4)$ ○ $h(x) \in \Omega(x^4)$

9 Let $h(x) = 6 \log_2 x$. Which of the following is true? **Mark all that apply**.

○ $h(x) \in \mathcal{O}(4x^2)$ ○ $h(x) \in \Theta(4x^2)$ ○ $h(x) \in \Omega(4x^2)$

10 Let $f(x) = x^2 \log_3 x$. Which of the following is true? **Mark all that apply**.

○ $f(x) \in \mathcal{O}(2x \log_{10} x)$ ○ $f(x) \in \Theta(2x \log_{10} x)$ ○ $f(x) \in \Omega(2x \log_{10} x)$

11 Let $g(x) = 2x \log_{10} x$. Which of the following is true? **Mark all that apply**.

○ $g(x) \in \mathcal{O}(x^2 \log_3 x)$ ○ $g(x) \in \Theta(x^2 \log_3 x)$ ○ $g(x) \in \Omega(x^2 \log_3 x)$

12 Using the provided pseudo-code, find the worst case performance in Big-$\mathcal{O}$ notation.

Algorithm 5 Some more messy pseudocode

```
procedure SomeProcedure2
    j=0
    for i=0; i<n; i++ do
        j=i+j
```

A. $\mathcal{O}(\log n)$

B. $\mathcal{O}(n \log n)$

C. $\mathcal{O}(n^2)$

D. $\mathcal{O}(n)$

13 Using the provided pseudo-code, find the worst case performance in Big-$\mathcal{O}$ notation.

Algorithm 6 Some messy pseudo-code

```
procedure SomeProcedure
    for i=1 and i<=n do
        j=1
        while j<n do
            j=j+2
```

A. $\mathcal{O}(\log n)$

B. $\mathcal{O}(n \log n)$

C. $\mathcal{O}(n^2)$

D. $\mathcal{O}(n)$

14 Using the provided pseudo-code, find the worst case performance in Big-$\mathcal{O}$ notation. Assume we know that **someMethod(n)** is $\mathcal{O}(\log n)$.

Algorithm 7 Some pseudocode with a method in it

```
procedure SomeProcedure3
    j=0
    for i=0; i<n; i++ do
        j=someMethod(n)
```

A. $\mathcal{O}(\log n)$

B. $\mathcal{O}(n \log n)$

C. $\mathcal{O}(n^2)$

D. $\mathcal{O}(n)$

15 Using the provided pseudo-code, find the worst case performance in Big-$\mathcal{O}$ notation.

Algorithm 8 Some more messy pseudocode

```
procedure SomeProcedure4
    while n> 1 do
        n=n/2
```

A. $\mathcal{O}(\log n)$

B. $\mathcal{O}(n \log n)$

C. $\mathcal{O}(n^2)$

D. $\mathcal{O}(n)$

16 Using the provided pseudo-code, find the worst case performance in Big-$\mathcal{O}$ notation.

Algorithm 9

```
procedure
    a = 1
    c = 0
    while a < n do
        for i = 0; i < n; i++ do
            c = c + 1
        a = a * 3
```

A. $\mathcal{O}(\log n)$

B. $\mathcal{O}(n \log n)$

C. $\mathcal{O}(n^2)$

D. $\mathcal{O}(n)$