

## MET CS 566 - Analysis of Algorithms

### Midterm Assignment - 20 Points

Please note the honor code policy regarding this Assignment. You may not discuss particular questions or discuss or transmit answers from the assignment with other people, except for the MET CS 566 teaching team. [Please submit to the Gradescope!](#)

### Tasks

1. **Ordering by asymptotic growth rates (4 points):**

Rank the following functions by order of growth. This means to find an arrangement  $g_1, g_2, \dots, g_9$  of the functions that satisfies  $g_1 = \Omega(g_2), g_2 = \Omega(g_3), \dots, g_7 = \Omega(g_8)$ .

1.  $g_1(n) = \lg(\lg(n))$
2.  $g_2(n) = 2^{\lg(n)}$
3.  $g_3(n) = 2^{\sqrt{2 \times \lg(n)}}$
4.  $g_4(n) = n^{5.2} + \lg(n)$
5.  $g_5(n) = n^{6.2} + n!$
6.  $g_6(n) = n^{1024} + n^{512} + n \times \lg(n)$
7.  $g_7(n) = e^n + e^{\lg(n)}$
8.  $g_8(n) = \sqrt{\lg(n)}$

Tip: Work in step-wise format. Find first the fastest growing function first, and then the next fast one.

---

2. **Asymptotic Notations (6 points)** For each pair of the following functions  $f(n)$  and  $g(n)$ , check if

1.  $f(n) = \Theta(g(n))$  ?
2.  $f(n) = O(g(n))$  ?
3.  $f(n) = \Omega(g(n))$  ?
4.  $f(n) = o(g(n))$  little o?
5.  $f(n) = \omega(g(n))$  little omega?

Functions  $f(n)$  and  $g(n)$  are:

1.  $f(n) = (128)^{\binom{n}{4}}$  vs.  $g(n) = (512)^{\binom{n}{8}}$
2.  $f(n) = n^{64} \times 2^n$  vs.  $g(n) = \lg(n) \times 2^n + 4^n + n^{32}$
3.  $f(n) = n^{1024}$  vs.  $g(n) = 2^{\lg(n) \times \lg(n)}$

You do not need to provide formal proofs. Describe your justifications in text form.

## MET CS 566

### Midterm Assignment

#### 3. Big O notation (2 points):

What is the running time of this code in Big O notation regarding  $n$ ?

---

**Algorithm 1** What is the running time of this code in Big O notation regarding  $n$ ?

---

```
1:  $y = 0$ 
2:  $j = 1$ 
3: while ( $j * j \leq n$ ) do
4:    $y = y + 1$ 
5:    $j = j + 1$ 
6: end while
```

▷ Note: line 4 is some constraint cost

#### 4. Big O notation (2 points):

What is the running time of this code in Big O notation regarding  $n$ ?

---

**Algorithm 2** What is the running time of this code in Big O notation regarding  $n$ ?

---

```
1:  $a = 0$ 
2:  $k = n^2$ 
3: while ( $k > 1$ ) do
4:   for  $j \leftarrow 0$  to  $n^2$  do
5:      $a = a + j$ 
6:   end for
7:    $k = k/2$ 
8: end while
```

#### 5. Recursive Function (2 points):

What is the running time of the following python code in Big O notation regarding  $n$ ?

```
def isThis(n):
    print(n)
    if (n == 1):
        return True
    if (n == 0):
        return False
    if ((n % 3) == 0):
        return (isThis(n / 3))
    else:
        return False
```

Listing 1: What is the running time of this code in Big O notation regarding  $n$ ?

6. **HEAP-EXTRACT-MAX worst case (1 points)**

For an arbitrary input array, A that in visualized Heap form does not satisfy the max-heap property.

Describe when the worst case of HEAP-EXTRACT-MAX happens.

Provide an example.

---

7. **Hash Function (1 points)**

Name two properties that a good hash function should have?

Describe your answer (If possible with an example).

---

8. **Double Hashing (2 points)**

When you use double hashing in a hash table, is the number of potential collisions depending on the sequence of key insertions into the table?

In other words, if you change the sequence of insertions would the number of collisions change?

Support your answer using an example.