

## Week 2: Lab

COLLABORATION LEVEL 0 (NO RESTRICTIONS). OPEN NOTES.

1. Prove or disprove:  $f = O(g)$  implies that  $g = O(f)$ .

What we expect: If true, a justification that holds for any functions  $f, g$  and for all  $n > n_0$ .  
If false, a counterexample.

2. Prove or disprove:  $f = O(g)$  implies that  $g = \Omega(f)$ .

What we expect: If true, a justification that holds for any functions  $f, g$  and for all  $n > n_0$ .  
If false, a counterexample.

3. Find the order of growth of the following functions. For each function, give its  $\Theta()$ .

(a)  $n \lg \lg n + n \lg n + \sqrt{n} \lg^2 n$

(b)  $\sqrt{n} \lg n + n$

(c)  $n^2 + \sqrt{n} \lg^3 n$

(d)  $3^{\lg n} + n^2 + n \lg n$

(e)  $\sqrt{3}^{\lg n} + n^2 + n \lg n$

(f)  $2^n + 2^{2n}$

(g)  $2^{\lg n} + \lg n^2$

(h)  $(\lg n)^{\lg n} + n^3$

4. Arrange the following functions in ascending order of growth rate. For each pair of consecutive functions, give a brief justification on why they are in this order (For e.g., if you ordered  $A, B, C$ , you need to justify that  $A = O(B)$  and  $B = O(C)$ ).

$$2\sqrt{\log n}, 2^n, n^{4/3}, n(\log n)^3, n^{\log n}, 2^{2n}, 2^{n^2}$$

5. Express the worst case running time of Insertion Sort as a sum and find its asymptotic order of growth (using the formula discussed this week).
6. Consider the **element uniqueness problem**: check whether all the elements in a given array are distinct. The problem can be solved by the following straightforward algorithm:

```

UNIQUEELEMENTS( $A[0..n-1]$ )
1 // Checks whether all the elements in a given array are distinct
2 // Output: Returns “true” if all elements are distinct, and “false” otherwise
3 For  $i = 0$  to  $n-2$ 
4     For  $j = i+1$  to  $n-1$ 
5          $A[i] == B[j]$  return false
6 return true

```

What is the running time (best-case, worst-case) of the algorithm, as a function of the  $n$ ? (hint: express as a sum and use the formula discussed).

7. Given two  $n$ -by- $n$  matrices<sup>1</sup>  $A$  and  $B$ , by definition, their product  $C = AB$  is an  $n$ -by- $n$  matrix  $C$  where

$$C[i, j] = A[i, 0]B[0, j] + A[i, 1]B[1, j] + \dots + A[i, k]B[k, j] + \dots + A[i, n-1]B[n-1, j]$$

The straightforward algorithm to compute the product is given below:

```

MATRIXMULTIPLICATION( $A[0..n-1, 0..n-1], B[0..n-1, 0..n-1]$ )
1 // Multiplies two square matrices of order  $n$  by the definition-based algorithm
2 // Output: Matrix  $C = AB$ 
3 For  $i = 0$  to  $n-1$ 
4     For  $j = 0$  to  $n-1$ 
5         // computes  $C[i, j]$ 
6          $C[i, j] = 0$ 
7         For  $k = 0$  to  $n-1$ 
8             if  $C[i, j] + A[i, k] \times B[k, j]$ 

```

What is the running time (best-case, worst-case) of the algorithm, as a function of the order of the matrix,  $n$ ?

8. The following algorithm finds the number of binary digits in the binary representation of a positive decimal integer.

---

<sup>1</sup>If you haven't seen matrices before, you can assume a matrix is a 2-dimensional array.

```

BINARY( $n$ )
1  // Input: A positive decimal integer  $n$ 
2  // Output: The numbr of binary digits in  $n$ 's binary representation
3  count = 1
4  While  $n > 1$ 
5      count++
6      n = n/2
7  return count

```

What is the running time (best-case, worst-case) of the algorithm, as a function of  $n$ ?

9. Consider the following algorithm:

```

MYSTERY( $n$ )
1  // Input: A positive decimal integer  $n$ 
2  s = 0
3  For  $i = 1$  to  $n$ 
4      s = s +  $i \times i$ 
5  return s

```

- (a) What does this algorithm compute?  
 (b) What is the running time of the algorithm, as a function of  $n$ ?
10. Assume that we have an algorithm whose running time is  $f(n)$  microseconds. Determine the largest size of a problem that can be solved by the algorithm in: (a) 1 second; (b) 1 hour; (c) 1 month.
- (a)  $f(n) = \lg n$  (b)  $f(n) = n^{10}$  (c)  $f(n) = 2^n$ .

## Optional

11. You are presented with 9 marbles. All of the marbles look identical i.e. same shape, color, and dimensions(except for weight). However, 8 of the 9 marbles have exactly the same weight; the last marble is heavier. The only tool you have to measure weights is an old fashioned balance scale. You are only allowed to use the scale 2 times. How do you find the one marble that is not the same weight as the others?
- Generalize to  $n$  marbles: find it using the scale  $\log_3 n$  times. (note: interview question)
12. You are given a set of  $n$  points on a circle in the plane. Come up with an algorithm that determines if there exists a pair of points that are antipodal (two points are antipodal if they are diametrically opposite). Analyze its running time (note: interview question) .