**CS 3510 C: Design & Analysis of Algorithms** 08/31/2023

# Practice Exam 1

Abrahim Ladha

- This is the CS 3510 practice exam for Exam 1. This does **not** approximate the difficulty or length of the actual exam; it serves as just a big question bank for you to practice!
- Topics include: Big-O, Master Theorem, Divide and Conquer, Arithmetic, Cryptography.
- Note that this assignment does not need to be submitted, but we highly recommend going through it to prepare for the exam!

# Big-O

**1.)** Explain why or why not the following claims are true (no need for any formal proofs):

1. If $a \in \mathbb{Z}_+$, then
   $a^{n+1} = \mathcal{O}(a^n)$

2. If $a \in \mathbb{Z}_+$, then
   $a^{2n} = \mathcal{O}(a^n)$

3. If $f(n) = \Omega(p(n))$ and $f(n) = \mathcal{O}(q(n))$, then
   for all $n \in \mathbb{R}$, $p(n) \leq q(n)$

4. If $a, b \in \mathbb{Z}_+$ and $a > b$, then
   $n^a - n^b = O(n^{a-b})$

5. If $a, b \in \mathbb{Z}_+$ and $a > b$, then
   $n^b - n^a = O(n^a)$

6. If $a \in \mathbb{Z}_+$, then
   $\mathcal{O}(n^a) - \mathcal{O}(n^a) = 0$

**2.)** For each of the following, describe the relationship between $f(n)$ and $g(n)$ using Big-O notation.

1. $f(n) = n^3 + 2n, \quad g(n) = 12n^2 + 24\sqrt{n}$
2. $f(n) = (\log n)^2, \quad g(n) = \log n + \sqrt{n}$
3. $f(n) = 2\log_5(3^n), \quad g(n) = n + 4n^{0.2}$
4. $f(n) = 8^{\log_7 n}, \quad g(n) = n \log n$

**3.)** Show that, $c$ is a positive real number, then $g(n) = 1 + c + c^2 + \cdots + c^n$ is

$$g(n) = \begin{cases} \Theta(1) & \text{if } c < 1. \\ \Theta(n) & \text{if } c = 1. \\ \Theta(c^n) & \text{if } c > 1. \end{cases}$$

***Hint**: Use geometric series.*

# Master Theorem

**4.)** Give the big-$\mathcal{O}$ runtime of the following recurrence relations.

1. $T(n) = 2T(n/4) + \mathcal{O}(n)$
2. $T(n) = 2T(n/4) + \mathcal{O}(1)$
3. $T(n) = 2T(n/4) + \mathcal{O}(\sqrt{n})$
4. $T(n) = 3T(n/3) + \mathcal{O}(1)$
5. $T(n) = 4T(n/3) + \mathcal{O}(n^2)$
6. $T(n) = 5T(2n/3) + \mathcal{O}(n^2)$

# Divide & Conquer

**5.)** Given an ordered array, $A$, of size $n$ with unique integers, and two boundary numbers, $l$ and $u$, design a Divide & Conquer algorithm to determine the number of integers within $A$ that lie between $l$ and $u$, both inclusive.

**6.)** Let $x$ and $y$ be two $n$-digit binary numbers, where $n$ is a power of 3. Let $x_L$, $x_M$ and $x_R$ consist of the first third, middle third, and final third of the digits of $x$, so that $x = 2^{\frac{2n}{3}} x_L + 2^{\frac{n}{3}} x_M + x_R$, and define $y_L$, $y_M$, and $y_R$ analogously.

(a.) Express $xy$ in terms of $x_L, x_M, x_R, y_L, y_M, y_R$. Simplify your answer.

(b.) Give a recursive algorithm that calculates the above expression with a recurrence relation of $T(n) = 6T(n/3) + O(n)$ and calculate its runtime.

**7.)** You've become an explorer and you've come across some mountains and valleys you need to cross. Unluckily for you, you can't climb mountains yet, so you need to find a valley. You're given a list representing a 2d height map like the one shown below.

An index, $i$, in the list, $A$, is considered a valley if $A[i] \leq A[i-1] \wedge A[i] \leq A[i+1]$ is true. Give a quick solution for finding valleys. Now assume, $A[1] > A[2]$ and $A[n-1] < A[n]$. Give a complete solution to find a valley under these conditions.

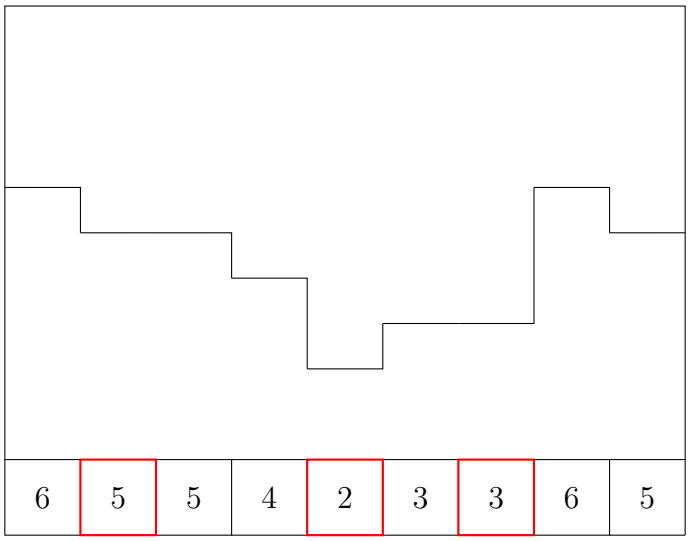


Figure 1: This figure displays an example of what a list of numbers would be represented as; the possible valleys are highlighted in red.

**8.) Guess the Path**

There exists an $n \times n$ grid in which you can only move down and to the right. Your goal is to escape the grid by starting at the top left corner $(1, 1)$ and moving to the bottom right corner $(n, n)$. However, there is only one correct path that will allow you to escape, and only the TAs know the answer. Your goal is to determine this path by querying the TAs.

| | | | |
|---|---|---|---|
| (1, 1) | (1, 2) | (1, 3) | (1, 4) |
| (2, 1) | (2, 2) | (2, 3) | (2, 4) |
| (3, 1) | (3, 2) | (3, 3) | (3, 4) |
| (4, 1) | (4, 2) | (4, 3) | (4, 4) |

Figure 1: Correct path

Treat a query to the TAs as a blackbox algorithm that takes in a path, and returns the coordinates of tiles in your proposed path that are in the correct path. Assume that this takes $\mathcal{O}(1)$ time for simplicity. For example, the correct path in a $4 \times 4$ grid is shown in Figure 1. If you query the paths shown in Figures 2 and 3 (marked in bold and italic), the green tiles show coordinates returned by the TAs.

| | | | |
|---|---|---|---|
| ***(1, 1)*** | (1, 2) | (1, 3) | (1, 4) |
| ***(2, 1)*** | (2, 2) | (2, 3) | (2, 4) |
| ***(3, 1)*** | (3, 2) | (3, 3) | (3, 4) |
| ***(4, 1)*** | ***(4, 2)*** | ***(4, 3)*** | ***(4, 4)*** |

(a) Figure 2

| | | | |
|---|---|---|---|
| ***(1, 1)*** | ***(1, 2)*** | ***(1, 3)*** | (1, 4) |
| (2, 1) | (2, 2) | ***(2, 3)*** | (2, 4) |
| (3, 1) | (3, 2) | ***(3, 3)*** | ***(3, 4)*** |
| (4, 1) | (4, 2) | (4, 3) | ***(4, 4)*** |

(b) Figure 3

(a.) Propose an algorithm to find the correct path that takes $\mathcal{O}(n^2)$ time.

(b.) Now, design a Divide-and-Conquer algorithm to find the correct path that is faster than your approach in part (a).

(c.) What changes about your solutions to part a and b if the grid is an $m \times n$ rectangle? Give new runtimes of both algorithms.

# Modular Arithmetic & RSA

**9.)** Solve the following problems:

1. Find $2^{20} + 3^{30} + 4^{40} + 5^{50} + 6^{60} \bmod 7$.
2. Solve the congruence $x^{103} \equiv 4 \bmod 11$.

**10.)** Suppose Tom wants to send Jerry a message using the RSA scheme.

1. Who should set up the RSA key?

2. Suppose he (the person generating the key) chooses as the two primes $p = 23$ and $q = 29$, and chooses as the encryption exponent $e = 3$. What number must he select for $d$, the decryption exponent? Show your work.

3. If the message being sent is $m = 15$, what is the encrypted message? Please show your work.

4. Let us say that when you generate your RSA key you pick $p$ and $q$ as 1024-bit primes, but by some luck when your algorithm randomly chooses the encryption exponent it gets a small value, $e = 3$. At first glance, it might seem helpful since it would require less computation for the sender to encrypt their messages. But there's actually a problem here, and you should modify your algorithm to ensure that $e$ is suitably large. Can you think of why this is?

   *Hint: Think about what happens when the message is really short, say a few bits. This problem can also be fixed by padding the message with extra bits, which is often how this is resolved in general.*