**Trees**

The Exam will only cover Binary Trees and Binary Search Trees.

**Multiple choice/Short answer question topics**

* Understand the purpose for Tree structures
* Understand the Tree vocabulary terms: root, inner-nodes, leaf-nodes
* Name the order of complexity for the following operations in a tree: find, insert, delete
* Understand how the Tree functions above (find, insert, delete) use recursion
* Are pre-order, in-order, post-order traversal depth first or breadth first?

**Coding**

* Be able to write a pre-order, in-order, and post-order traversal in Python
* Be able to write the find function, recursively in Python

**Recursion**

**Multiple choice/Short answer questions**

* Understand when to use recursion (when the problem is recursive in nature).
* Understand what “recursive in nature means. I.e. when the solution to the problem is the same logic no matter how large the problem gets.

Examples: factorials, whatever the number is, the logic is the same multiple the number x number-1 x number-2 … 1

Fibonacci series

Trees. The functions to find, insert, and delete into a tree is the same logic no matter how big the tree is.

**Coding**

* Write a function to solve a factorial for any given number.

**Python Dictionary/JSON**

**Multiple Choice/Short answer questions**

* Understand what the difference between a key and a value is
* Understand how dictionaries can contain any type including other dictionaries.
* Understand how to be able to extract the contents of a dictionary to analyze the data (with keys and loops).

**Coding**

* Be able to write the code to exact a list out of a dictionary and analyze it (something simple like find the mean, max, min), then save the dictionary to a JSON file.

**Queues/Stacks**

**Multiple Choice/Short Answer**

* Understand what problems are best suited to each data structure. I.e. queues for data that needs to be processed in the order it was received (FIFO). Stacks process data, the most recent first (LIFO).

**Coding**

* Be able to use a Python list as a queue and a stack. You will not be required to use the actual Queue and Stack objects, a list is fine.

Hint: Understand how to use the list pop() function for a queue and stack (pop() for stack, pop(0) for queue).

**Sorting Algorithms**

**Multiple Choice/Short Answer**

* Understand how the following sorts work: Bubble sort, Insertion sort, Selection sort, Merge sort, Quick sort.
* You will not be required to code them all, but you should be able to look at them and tell the difference i.e. If I give the code for insertion sort, you should be able to recognize it is the insertion sort.

**Coding**

* Be able to code the bubble sort.

**Algorithm Analysis, Big O Notation**

**Multiple Choice/Short Answer**

* Understand how big O notation is used to measure and compare algorithms.
* Be able to give the big O notation for the following algorithms: finding the middle element of a list, printing the items in a list, quick sort, finding the average of a 2-dimensional list.
* Be able to put the following run times in order of speed:
* *N*,
* log *N*,
* *N* log *N*,
* *N*1.5,
* *N*2,
* *N*3,
* 2*N*

**Coding**

* Be able to write the code for an algorithm for the following big O notations: N, log N, N^2

**Questions we will review in our Monday Zoom session.**

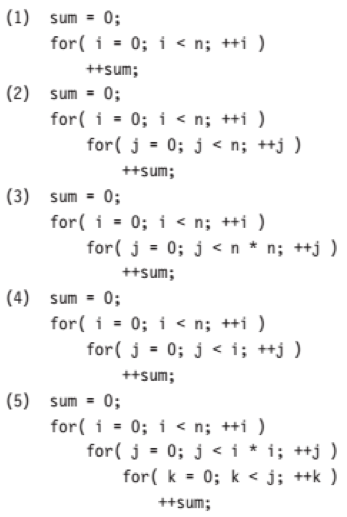
Q- what is the order of complexity for the selection sort

Q- What is faster, the bubblesort or the selection sort

Q- What is the order of complexity for the binary search

Q- What is the order of complexity when searching a list of random unordered numbers?

1. In a recent court case, a judge cited a city for contempt and ordered a fine of $2 for the first day. Each subsequent day, until the city followed the judge’s order, the fine was squared (i.e., the fine progressed as follows: $2, $4, $16, $256, $65,536, . . .). a. What would be the fine on day N?
2. Please give the complexity for each code sample…



1. Is the Linux/Unix Find command Depth first or Breadth first. Does it print the contents pre-order, in-order, or post-order…

abrim24:~/workspace/tree $ pwd

/home/ubuntu/workspace/tree

abrim24:~/workspace/tree $ ls

alex/ bill/ mark/

abrim24:~/workspace/tree $ find .

.

./mark

./mark/book

./mark/book/ch3

./mark/book/ch2

./mark/book/ch1

./mark/course

./mark/course/cop5350

./mark/junk

./bill

./bill/suzuki

./bill/kawasaki

./bill/yamaha

./bill/work

./bill/course

./bill/course/cop3212

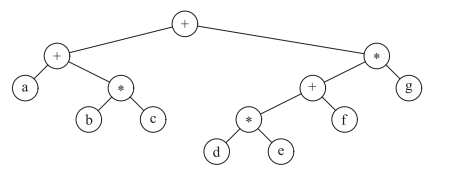
./bill/course/cop3212/fall2

./bill/course/cop3212/fall1

./alex

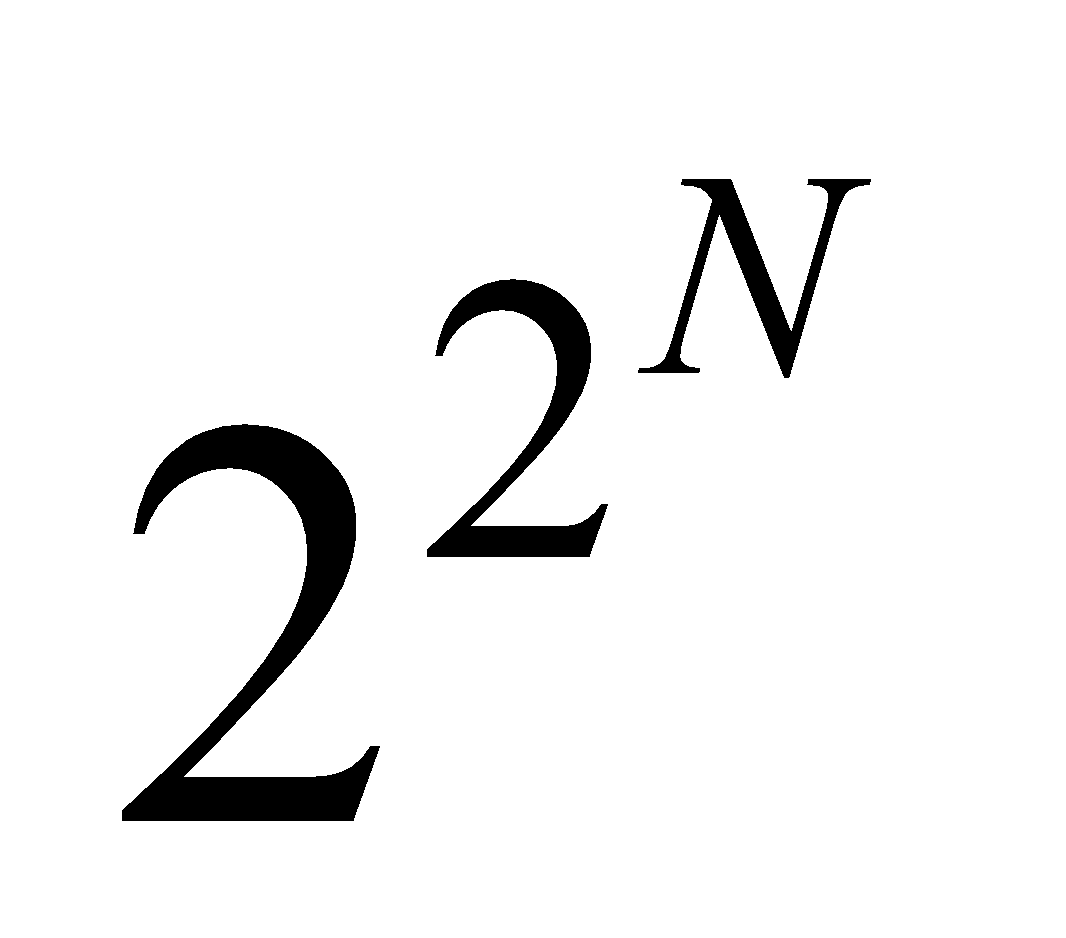
./alex/junk

abrim24:~/workspace/tree $



print the tree above, traversing pre-order, in-order, post-order

Answers

1. **(a)** 

**2.**

**(I)**The running time is *O*(*N*).

**(II)**The running time is *O*(*N*2).

**(III)**The running time is *O*(*N*3).

**(IV)**The running time is *O*(*N*2).

**(V)***j* can be as large as *i*2, which could be as large as *N*2. *k* can be as large as *j*, which is *N*2. The running time is thus proportional to *N*⋅*N*2⋅*N*2, which is *O*(*N*5).

3. Depth First, Pre-order

4.

++a\*bc\*+\*defg

a+b\*c+d\*e+f\*g

abc\*+de\*f+g\*+