Curtin College

End of Study Period Final Assessment – Study Period 3, 2020

Unit: DSA1002 Data Structures and Algorithms

Duration: 3.5 Hours

THIS IS AN OPEN BOOK ASSESSMENT

This paper is an open book, open computer test. You may make use of internet sources but must answer questions in **your own words** and in **your own code**. In programming tasks, any List, Stack, Queue, Hash table, Graph or Tree data structures should be implemented using algorithms from the unit. Arrays in python should use numpy arrays. Answers are to be completed individually without outside assistance.

How to Submit

Answers to written questions must be submitted to Moodle in a document format (.docx or .pdf).

Do not submit a zip archive or similar.

ATTEMPT ALL QUESTIONS

Question 1 (Total 10 marks): General

a) Consider a situation where you must store a list of tasks that must be completed by a series of manufacturing robots. Each of the tasks contains a series of instructions that need to be completed by a robot. Once a robot has completed a task, the task can be discarded, and the robot can begin the next task. Some tasks are more important than others and must be completed sooner.

   a. **(3 marks)** What **ADT** would you use to store this data? Justify your choice.

   b. **(3 marks)** How does your choice handle more important tasks happening sooner than less important choices? Explain this with reference to **Big O** time complexity.

b) **(4 marks)**. A large data set of products with items listed in order of their **product code** needs to be searched by **name**. A large amount of memory is available, and the data does not change very often but searches happen many times a second. How would you decide to search the data?

Question 2 (Total 20 marks): Sorting and Recursion

a) **(3 marks)**. If you had to use a sorting algorithm in a low memory environment, what would be the most important factor to consider? Would a **mergesort** be an appropriate choice? Justify your answer.

b) **(5 marks)**. Do you agree with the statement: “The best **pivot** to select for a **quicksort** is the **median** of all the values.”? Justify your answer with reference to **Big O** time complexity.

c) **(3 marks)**. Divide and Conquer style sorting algorithms like **quicksort** and **mergesort** can be done recursively. What kind of problems could occur when recursively sorting very large datasets? Can this be mitigated, or should a different algorithm be used?

d) **(5 marks)**. Rewrite the following recursive algorithm as an iterative one.

Python

```
def findTheApple(fruits):
    recurseFindTheApple(fruits, len(fruits)-1)

def recurseFindTheApple(fruits, i):
    if fruits[i] == 'Apple':
        print('Found the apple!')
    else:
        print('Not an apple')
    if(i>0):
        recurseFindTheApple(fruits, i-1)
```

Java

```
public static void findTheApple(String[] fruits)
{
    recurseFindTheApple(fruits, fruits.length-1);
}

public static void recurseFindTheApple(String[] fruits, int i)
{
    if(fruits[i] == "Apple")
        System.out.println("Found the apple!");
    else
        System.out.println("Not an apple");
    if(i>0)
        recurseFindTheApple(fruits, i-1);
}
```

e) **(4 marks)** Explain how the above algorithm could be made more efficient if it did not need to print out the “Not an apple” message. Would this improve its time complexity?

Question 3 (Total 16 marks): Stacks, Queues and Lists

a) **(3 marks)**. Implementing a stack with an array allows O(1) push and O(1) pop operations. Using a Linked List instead also allows O(1) push and O(1) pop operations. Does this mean there's no difference between them in practical applications? Justify your answer.

b) **(3 marks)**. Is a double-ended list necessary to implement a queue? Can a single-ended list be used? What list operations are affected when attempting to use a single-ended list?

c) **(10 marks)**. Write an iterator that traverses a doubly linked list in reverse order (You do not need to implement the list, only it's iterator)

Question 4 (Total 20 marks): Trees

a) **(4 marks)**. Draw a diagram of the binary search tree that would result from inserting the following numbers in the order that they are shown:

49,87,86,36,66,20,15,65,31,78

b) **(4 marks)**. Redraw the tree from part (a) after deleting 36 and then 86. Explain the steps you used to delete these elements.

c) **(5 marks)**. Draw the **2-3-4 search tree** that results from inserting the following sequence of numbers into an initially empty tree (showing all of your working)

8,54,81,2,99,70,76,15,13

d) **(2 marks)**. Convert the tree from part 4c above into a **Red-Black tree**.

e) **(5 marks)**. Given the partial definition of the BinaryTree and associated TreeNode classes below, write a method to find the second largest node (parent of the right-most node)

```
class BinaryTree():
    class TreeNode():
        def __init__(self,value):
            self.value = value
            self.left = None
            self.right = None

    def __init__(self):
        self.root = None

    def secondLargest(self):
        #you need to implement this
```

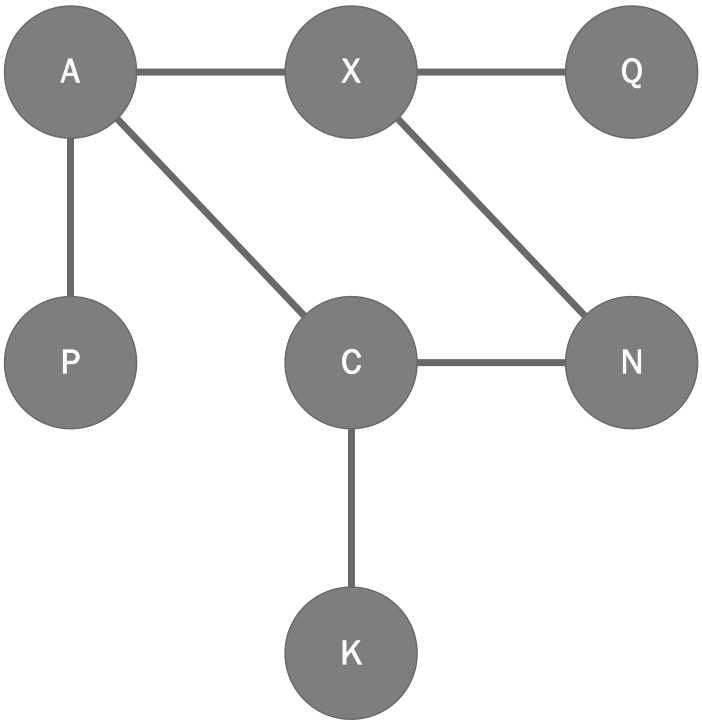
```
public class BinaryTree {

    private class TreeNode {
        public int value;
        public TreeNode left;
        public TreeNode right;
    }

    private TreeNode root;

    public int secondLargest(){
        //you need to implement this
    }
}
```

Question 5 (Total 12 marks): Graphs



a) **(3 marks)**. Provide the adjacency matrix representation for the above graph.

b) **(3 marks)**. Provide the adjacency list for each vertex of the above graph.

c) **(3 marks)**. Perform a **depth-first** search of the graph above.

d) **(3 marks)**. Perform a **breadth-first** search of the graph above.

Question 6 (Total 12 marks): Heaps

Given the following list of numbers:

99,80,14,31,79,41,37,31,52,63

a) **(5 marks)**. Draw the heap (as a tree diagram) that would be built if the above numbers were inserted into a max heap in the order they are listed. Show your working.

b) **(2 marks)**. Convert the heap tree in your answer from question 6a into an array representation of the tree.

c) **(2 marks)**. Show what the array form of the heap would look like after inserting the value of 8 into the heap (hint: use your tree to help you trace the ‘trickle-up’).

e) **(3 marks)**. Give an example of a real-world application for a heap tree. Why is it a useful data structure?

Question 7 (Total 10 marks): Hash Tables

a) **(4 marks)**. Given the following hash function that maps an integer key to an index given a table of length 10:

```
public int hash(int key) {
    int hashIdx;
    hashIdx = key % 10;
    return hasIdx;
}
```

```
def hash(key):
    hashIdx = key % 10
    return hashIdx
```

Use the above hash function to insert the following key-value pairs into the hash table below.

Handle collisions using the **Linear Probing** method.

Key: Value

1: “Apple”

356: “Pear”

23: “Cherry”

43: “Orange”

964: “Banana”

Index	Key	Value
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

b) **(4 marks)**. What do you notice about the locations of the inserted data due to collisions? How could a different collision strategy reduce the chances of collisions occurring?

c) **(2 marks)**. Other than different collision strategies, what else could you do to lower the probability of collisions occurring?