210CT Coursework Mason Glover **6300263**

**2016/2017 Programming, Algorithms and Data Structures (210CT)**

**Coursework**

**MODULE LEADER: Dr. Diana Hintea**

**Student Name: Mason Glover**

**SID: 6300263**

**I can confirm that all work submitted is my own: Yes [ / ]**

210CT Coursework Mason Glover **6300263**

**210CT Coursework Submission**

**Mason Glover, Year 2**

Contents

[Question 1: 3](#_Toc468313523)

[Question 2: 4](#_Toc468313524)

[Question 3: 5](#_Toc468313525)

[Question 4: 6](#_Toc468313526)

[Question 5: 7](#_Toc468313527)

[Question 6: 8](#_Toc468313528)

[Question 7: 9](#_Toc468313529)

[Question 8: 10](#_Toc468313530)

[Question 9: 11](#_Toc468313531)

[Question 10: 12](#_Toc468313532)

[Question 11: 13](#_Toc468313533)

[Question 12: 14](#_Toc468313534)

[Question 13 and 14: 16](#_Toc468313535)

210CT Coursework Mason Glover **6300263**

# Question 1:

Write a function that randomly shuffles an array of integers and explain the rationale behind its implementation. Input: [5,3,8,6,1,9,2,7] (One potential example of) Output: [2,8,3,1,9,7,5,6]

=============================================================

#The Rationale behind this implemented algorithm is that it takes an

#existing list and based on the length of the list and a random integer,

#if the value pulled matches the current state of the list, it will

#write out that specific entry into a new list then repeat until all previous entries are written into the new list.

import random

def array(x):

#makes a integer of the list's length

length = len(x) - 1

#for loop for every entry from the passed list

for i in range(length, 0, -1):

#value variable can be anything from 0 to the length of the list being passed..

value = random.randint(0,i)

#If the value matches the integer being passed it will ignore it as its already sorted

if value == i:

pass

#Rearranges the value selected to the first available position.

x[value],x[i] = x[i], x[value]

#prints the shuffled list

return x

#The list to be used

myArray = [5,3,8,6,1,9,2,7]

#Does my algorithm have defined Inputs and Outputs?

#Yes, the Inputs are myArray and outputs are the returned x

#It does terminate

#The code is specified clearly and shows exactly what happens each line.

# The algorithm will reshuffle the list passed to the function every time with differing shuffles.

210CT Coursework Mason Glover **6300263**

# Question 2:

Count the number of trailing 0s (number of 0s at the end of the number) in a factorial number.

Input: 5 Output: 1, Input: 10 Output: 2

Hint: Count the prime factors of 5

=============================================================

def zeroes(x):

factorial = x

zeroes = 0

#the following code is performed as many times as the value of the factorial number in increments of one

for factorial in range(1,factorial+1):

#checks if the value entered is above or equal to 0

while factorial > 0:

#Check the number to see if it is divisible by 5

if factorial % 5 == 0:

#every time 5 can be divided to give no remainder, a zero is added. the number passed is ran through the conditionals again.

factorial = factorial / 5

zeroes = zeroes + 1

else:

#Used to break out of the if statement or the function would never print

break

return zeroes

210CT Coursework Mason Glover **6300263**

# Question 3:

Write the pseudocode for a function which returns the highest perfect square which is less or equal to its parameter (a positive integer). Implement this in the programming language of your choice.

=============================================================

PERFECTSQ(X)

sqrtX 🡨 X ^(1/2)

numSquared 🡨 sqrtX mult sqrtX

if (INTEGER = numSquared)

return sqrtX

else

return X((X^1/2)^2)^2

#import math class to use the floor and sqrt functions

import math

def perfectSquare(x):

#When X is passed, the SQRoot is calculated and floored to give an integer to compare with the integer squared

sqrtX = math.floor(x\*\*(1/2))

numSquared = sqrtX \* sqrtX

#Return previous variable if the number passed is a perfect square, else return the closest perfect square after square rooting x

if int(x) == numSquared:

return(str(sqrtX) + " is a perfect square of " + str(x))

else:

return("Not a perfect square, nearest is: " + str(round(x\*\*(1/2))\*\*2))

210CT Coursework Mason Glover **6300263**

# Question 4:

Look back at last week’s tasks. Describe the run-time bounds of these algorithms using the BigO notation.

For Question 1, I found that it was quite simple to calculate the Big O notation via runtime of the program as there was only one for loop that would run for as many numbers in the array compared to a nested if statement which may only run some of the time in worst cases.

Question 2 included multiple nested conditionals such as an if statement within a while loop, looping n times within a for loop which is looping n times by itself.

=============================================================

Question 1 Code

def array(x):

#makes a integer of the list's length

length = len(x) – 1 **(1)**

#for loop for every entry from the passed list

for i in range(length, 0, -1): **(n Times)**

#value variable can be anything from 0 to the length of the list being passed..

value = random.randint(0,i) **(1 Times)**

#If the value matches the integer being passed it will ignore it as its already sorted

if value == i: **(n Times)**

pass **(n?)**

#Rearranges the value selected to the first available position.

x[value],x[i] = x[i], x[value] **(1 Times)**

#prints the shuffled list

return x **(1)**

**2N + 3 🡪 SIMPLIFED INTO 🡪 O(N)**

Question 2 Code

def zeroes(x):

factorial = x **(1)**

zeroes = 0 **(1)**

for factorial in range(1,factorial+1): **(N times)**

while factorial > 0: **(N \* N times)**

if factorial % 5 == 0: **(N \* N times)**

factorial = factorial / 5 **(N \* N times)**

zeroes = zeroes + 1 **(N \* N times)**

else:

break **(1)**

return zeroes **(1)**

**1N + 4N^2 + 3 🡺 SIMPLIFIED INTO 🡺 4N^2 SIMPLIFIED INTO 🡺 O(N^2)**

210CT Coursework Mason Glover **6300263**

# Question 5:

Write the pseudocode corresponding to functions for addition, subtraction and multiplication of two matrices, and then compute A = B\*C-2\*(B+C), where B and C are two matrices of order N. What is the run-time?

==================================================================

Matrice Addition

MADD(B,C):

result 🡨 ((0,0),(0,0)) **(1)**

for I to B[1…I] **(N Times)**

for J to B[0] **(N \* N Times)**

result 🡨 B[I][J] + C[I][J] **(N \* N Times)**

return result **(1)**

**O(N^2)**

**N+2N^2+2**

Matrice Subtraction

MSUB(B,C):

result 🡨 ((0,0),(0,0)) **(1)**

for I to B[1...I] **(N Times)**

for Jto B[0] **(N \* N Times)**

result 🡨 B[I][J] – C[I][J] **(N \* N Times)**

return result **(1)**

**O(N^2)**

**N+2N^2+2**

Matrice Multiplication

MMULTI(B,C):

result 🡨 ((0,0),(0,0)) **(1)**

for I to B[1…I] **(N Times)**

for J in C[0**] (N \* N Times)**

for K to C[1…K] **(N \* N Times)**

result 🡨 B[I][K] \* C[K][J] **(N \* N Times)**

return result **(1)**

**O(N^3)**

**N+3N^3+2**

**3N+5N^7+6 🡪 SIMPLIFIED INTO 🡪 O(N^2)**

MATRICE1 (B\*C) = O(N^3)

MATRICE2 (B+C) = O(N^2)

A = O(N^3) – 2(O(N^2)

210CT Coursework Mason Glover **6300263**

# Question 6:

Write the Pseudocode and code for a function that reverses the words in a sentence. Input: “This is awesome” Output: “awesome is This. Give the Big O notation.

==================================================================

Pseudocode

REVERSEORDER(SENTENCE):

word 🡨 “”

sentenceReversed 🡨 “”

for entry in sentence

if (I = “”)

sentenceReversed 🡨 word + “ “ + sentenceReversed

word 🡨 “”

else

word 🡨 word + entry

sentenceReversed 🡨 word + “ “ + sentenceReversed

return sentenceReversed

Code

def reverseOrder(sentence): **(1)**

#Two variables, word is the elements catched before spaces, sentenceReversed stores them in reverse order"

word = "" **(1)**

sentenceReversed = "" **(N)**

#for every element in the passed string.

for i in sentence: **(N)**

#if the element is a space; **(N)**

if i == ' ' or i == "," or i == ".": **(N)**

#this variable takes on any values passed into the word variable and resets its value.

sentenceReversed = word + ' ' + sentenceReversed **(N)**

word = '' “**(N)**

else: **(N)**

#the element is added to the word variable.

word += I **(N)**

#the word variable is added onto whatever is already present within sentenceReversed in a past state.

sentenceReversed = word + ' ' + sentenceReversed **(1)**

return sentenceReversed **(1)**

**8N+4 🡪 SIMPLIFIED INTO 🡪 O(N)**

210CT Coursework Mason Glover **6300263**

# Question 7:

Write a recursive function (pseudocode and code) to check if a number n is prime (hint: check whether n is divisible by any number below n)

==================================================================

Pseudocode:

PRIMECHECK(NUMBER, DIVIDER)

If (NUMBER = 2)

isPrimeNum 🡨 False

else If(NUMBER <=1)

isPrimeNum 🡨 False

else If(NUMBER mod DIVIDER = 0)

isPrimeNum 🡨 False

else If(DIVIDER \* DIVIDER > NUMBER)

isPrimeNum 🡨 True

else

isPrimeNum 🡨 primeCheck(NUMBER, DIVIDER + 1)

return isPrimeNum

Code:

def primeCheck(number, divider=2):

#if the passed number is 2, then its already a prime number.

if number == 2:

isPrimeNum = True

#if the number is either 1 or 0, then it is not a prime number.

elif number < 1 or number == 1:

isPrimeNum = False

#if the current number can divide N to give no remainders before then it is not a prime number.

elif number % divider == 0:

isPrimeNum = False

#if the dividing number is equal to N, it is a prime number(BASE CASE)

elif divider \* divider > number:

isPrimeNum = True

#The recursion function recalls the function but adds 1 to the dividing number.

else:

isPrimeNum = primeCheck(number, divider+1)

return isPrimeNum

210CT Coursework Mason Glover **6300263**

# Question 8:

Write a recursive function (pseudocode and code) that removes all vowels from a given string s. Input: "beautiful" Output: "btfl".

==================================================================

Pseudocode:

REMOVEVOWELS(SENTENCE)

If (SENTENCE[i…n] = 0)

return(SENTENCE)

else (if SENTENCE[0] = “aeiou” or SENTENCE[0] = “AEIOU”)

return(REMOVEVOWELS(SENTENCE[1:n]

return SENTENCE[0] + REMOVEVOWELS(SENTENCE[1…n])

Code:

def removeVowels(sentence):

#nothing was entered or base case was reached.

if len(sentence)== 0 :

#Nothing will be returned, this base case will be hit after the function so it has to return nothing.

return(sentence)

#check if the first value taken in the string is a vowel lowercase or uppercase.

elif sentence[0] in "aeiou" or sentence[0] in "AEIOU":

return removeVowels(sentence[1:])

#returns the value and any other values that are excluded from the called functions recursion until there is nothing left in the initial string.

return sentence[0] + removeVowels(sentence[1:])

210CT Coursework Mason Glover **6300263**

# Question 9:

Adapt the binary search algorithm so that instead of outputting whether a specific value was found, it outputs whether a value within an interval (specified by you) was found. Write the pseudocode and code and give the time complexity of the algorithm using the Big O notation. Example input: L = [2,3,5,7,9,13] low= 10 high = 14 Output: True

def rangeSearch(low, high, array):

#bottom variable refers to the first position of the array

bottom = 0

#top variable is the last position of the array.

top = len(array)-1

#variable is set to false as nothing has been found yet

rangeFind = False

#this condition is true whilst bottom is not equal to the top position

#and an element in the range specific has not been found

while bottom <= top and rangeFind == False:

#this is the pointer for the middle of the array

middle = (bottom+top)//2

#if the mid point lands on an element(s) in which the value

#is higher than the low variable but smaller than the high variable

if array[middle]< high and array[middle] > low:

found = True

return str(found) + " " + str(array[middle])

#if the pointer is only small than the high variable then the bottom

#position is moved up.

elif array[middle] < high:

bottom = middle + 1

elif array[middle] > low:

#similar scenario for the first elif except it will move

#the last element pointer back

top = middle - 1

#return whether a value was found

return("No value exists between the specified ranges.")

mylist = [1,2,3,5,7,8,9,10,14,15]

210CT Coursework Mason Glover **6300263**

# Question 10:

Given a sequence of n integer numbers, extract the sub-sequence of maximum length which is in ascending order

#there is a bug where the last part of any passed list is ignored when the second to last lower value is detected

def maxSubSeq(x):

#this list holds the current longest sub sequence in ascending order.

tempList = [ ]

#this variable controls where the function starts checking for ascending order from.

position = 0

#this variable gets updated with the value for the highest length for sub sequence everytime a new longest length is discovered.

maxLenSubSec = 0

#for every element in the list

for i in range(1,len(x)):

#if the current elements value is lower than the previous element, break off the list and add it into the tempList variable.

#from the start position to the offending element

if x[i]<x[i-1]:

tempList.append(x[position:i])

#when this if statement is true, update the position variable to start from the next element

position = i

#iterate through each sub sequence within a list in the tempList list variable, same rule applies of if the longest sub sequence is detected except

# it applies to the length of the list, when it finishes iterating, it will return the longest sub sequence

for i in range(len(tempList)):

if len(tempList[i]) > len(tempList[i-1]):

maxLenSubSec = i

return(tempList[maxLenSubSec])

#return(tempList)

numbers = [1,2,3,4,1,5,1,6,7]

. 210CT Coursework Mason Glover **6300263**

# Question 11:

Based on the Python code or the C++ code provided in class as a starting point, implement the double linked list node delete function.

class Node(object):

def \_\_init\_\_(self, value):

self.value=value

self.next=None

self.prev=None

class List(object):

def \_\_init\_\_(self):

self.head=None

self.tail=None

def insert(self,n,x):

#Not actually perfect: how do we prepend to an existing list?

if n!=None:

x.next=n.next

n.next=x

x.prev=n

if x.next!=None:

x.next.prev=x

if self.head==None:

self.head=self.tail=x

x.prev=x.next=None

elif self.tail==n:

self.tail=x

def display(self):

values=[]

n=self.head

while n!=None:

values.append(str(n.value))

n=n.next

print ("List: ",",".join(values))

def remove(self, n):

#if the previous node does not exist, assign the node position to the next available node.

if n.prev!= None:

n.prev.next = n.next

else:

#make the head of the heap the next available node

self.head = n.next

# #if the next node does not exist then assign that nodes previous position to the current nodes previous position.

if n.next!= None:

n.next.prev = n.prev

else:

#assign the end of the queue to the nodes previous node.

self.tail = n.prev

if \_\_name\_\_ == '\_\_main\_\_':

l=List()

l.insert(None, Node(4))

l.insert(l.head,Node(6))

l.insert(l.head,Node(8))

l.insert(l.head,Node(12))

l.remove(l.head)

l.display()

. 210CT Coursework Mason Glover **6300263**

# Question 12:

Implement TREE\_SORT algorithm in a language of your choice, but make sure that the INORDER function is implemented iteratively.

class BinTreeNode(object):

def \_\_init\_\_(self, value):

self.value=value

self.left=None

self.right=None

def tree\_insert( tree, item):

if tree==None:

tree=BinTreeNode(item)

else:

if(item < tree.value):

if(tree.left==None):

tree.left=BinTreeNode(item)

else:

tree\_insert(tree.left,item)

else:

if(tree.right==None):

tree.right=BinTreeNode(item)

else:

tree\_insert(tree.right,item)

return tree

def postorder(tree):

if(tree.left!=None):

postorder(tree.left)

if(tree.right!=None):

postorder(tree.right)

print(tree.value)

def in\_order(tree):

#this creates an empty stack that will append tree nodes

s = []

#this variable will turn true after each tree node has been examined

done = False

while(done != True):

#append the first node found to the stack then assign the lower value as the root

if tree != None:

s.append(tree)

tree = tree.left

#Go back to the top node once the left side is done, check the right side

elif len(s) >0 :

#pop the most recent value from the stack which is the top node, print its value then look on the right side nodes.

tree = s.pop()

print (tree.value)

tree = tree.right

#repeat the same principles from left side to right, once the right side is empty, set the done variable to true to stop running the function.

else:

done = True

if \_\_name\_\_ == '\_\_main\_\_':

t=tree\_insert(None,6);

tree\_insert(t,10)

tree\_insert(t,5)

tree\_insert(t,2)

tree\_insert(t,3)

tree\_insert(t,4)

tree\_insert(t,11)

in\_order(t)

. 210CT Coursework Mason Glover **6300263**

# Question 13 and 14:

Write the pseudocode for an unweighted graph data structure. You either use an adjacency matrix or an adjacency list approach. Also, write a function to add a new node and a function to add an edge. Following that, implement the graph you have designed in the programming language of your choice. You may use your own linked list from previous labs, the STL LL, or use a simple fixed size array (10 elements would be fine)

Implement BFS and DFS traversals for the above graph. Save the nodes traversed in sequence to a text file. class Vertex:

PSEUDOCODE

CLASS VERTEX:

LABEL 🡨 0

CLASS GRAPH

AdjList 🡨 dictionary ( )

ADDNODE(SELF,NODE):

If node.label in adjList:

Continue

Else:

adjList[node.label] = [ ]

ADDEDGE(SELF,FROMNODE,TONODE):

If fromNode in adjList:

If toNode in adjList:

AdjList[fromNode].append (toNode)

AdjList[toNode].append (fromNode)

RETURNGRAPH(SELF)

Return adjList

#when an object is instaniated, it is given a label which is the nodes name.

def \_\_init\_\_(self,label):

self.label = label

class Graph:

#adjList is the method I show my data graph structure, with the node as a dictionary linking to a list of neighboring nodes.

adjList = { }

def addNode(self,node):

#if the node being entered already exists, ignore it. Else, add the node as a Vertex object with a value of an empty list.

if node.label in self.adjList:

pass

else:

. 210CT Coursework Mason Glover **6300263**

self.adjList[node.label] = []

def addEdge(self,fromNode,toNode):

#First check if the two nodes being used to add an edge exists.

if fromNode in self.adjList:

if toNode in self.adjList:

#when both nodes have been found, append each of the node's dictionary key to include each other within their values.

self.adjList[fromNode].append(toNode)

self.adjList[toNode].append(fromNode)

def returnGraph(self):

#return the adjecency list data graph for the object being passed.

return(self.adjList)

def dfs(graph,startNode):

#start with a empty stack and visited list, the visited list will keep a record of nodes that have been visited already.

stack = [ ]

visited = [ ]

#append the provided node to the stack then start a while loop conditional for if the stack has a node inside.

stack.append(startNode)

#while the stack is not empty

while stack:

#pop the first unvisited node linked to the current before continuing.

edge = stack.pop()

#if the node currently being visited was not visited before, add it into the existing list in visited

#then update the stack list before adding the rest of the edges before moving onto the next node.

if edge not in visited:

visited.extend(edge)

#Traverse to the first unvisited node linked to the previous popped node, stop once all connected nodes are in the visited list.

stack = stack + graph.adjList[edge]

#create a text file in the same folder as the script, write the start node passed then how the nodes were visited.

text\_file = open("outputtedWork.txt", "w")

text\_file.write("DFS Result of " + str(startNode + str(visited)))

text\_file.close()

return visited

def bfs(graph,startNode):

#Similar process to DFS except when it finds its first available node

stack = [ ]

visited = [ ]

stack.append(startNode)

while stack:

edge = stack.pop(0)

if edge not in visited:

. 210CT Coursework Mason Glover **6300263**

visited.extend(edge)

stack = stack + graph.adjList[edge]

text\_file = open("outputtedWork.txt", "w")

text\_file.write("BFS Result of " + str(startNode) + str(visited))

text\_file.close()

return visited

#graph will be an object of Graph class which I use to conduct the necessary tasks

graph = Graph()

#Simple fixed array of 10 elements to be used as nodes

nodes = ['0','1','2','3','4','5','6','7','8','9']

#Adds each element of the node function into a graph format by calling the addNode function with the argument being the node's position as a Vertex object.

for i in range(0,len(nodes)):

graph.addNode(Vertex(nodes[i]))

#Example code to add edges inbetween existing nodes

graph.addEdge('0','2')

graph.addEdge('2','3')

graph.addEdge('3','4')

graph.addEdge('3','5')

graph.addEdge('5','6')

#Brief pseudo code provided from lecture slides to help me understand

#CLASS VERTEX

#LABEL <--- 0

# EDGES < ---- [ ]

#example <--- new VERTEX ( example = Vertex(24,3)

#example.label <--- 24 (24 IS A NODE)

#example.edges.add(3) <------ 3 is the node it is going to