# CSCE 221 Cover Page Homework Assignment #2

Due March 25 at 23:59 pm to eCampus

First Name: Pratik Last Name: Patel UIN: 527004337

User Name: p.pratik99 E-mail address: p.pratik99@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more Aggie Honor System Office <http://aggiehonor.tamu.edu/>

|  |  |  |  |
| --- | --- | --- | --- |
| Type of sources |  |  |  |
| People |  |  |  |
| Web pages (provide URL) |  |  |  |
| Printed material |  |  |  |
| Other Sources |  |  |  |

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.

“On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.”

Your Name: Pratik Patel Date: 03/23/2019

## Homework 2

**due March 25 at 11:59 pm to eCampus.**

1. (15 points) Describe in pseudo code how to implement the stack ADT using two queues.
   1. Write a C++ function that implements your solution. You can use the C++ STL queue container.

A screenshot of a cell phone

Description automatically generated

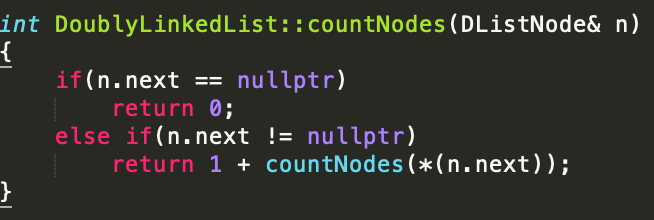
* 1. What is the running time of the push and pop functions in this case?

push() –

pop() –

Because it has to put the first n-1 elements in the second queue that takes n operations. Therefore, the time complexity is

1. (15 points) Write a recursive function in C++ that counts the number of nodes in a singly linked list.



* 1. Test your function using different singly linked lists. Include the code and screenshots with testing cases.

A screenshot of a cell phone

Description automatically generated

A picture containing indoor, object, wall

Description automatically generated

* 1. Write a recurrence relation that represents your algorithm.

* 1. Solve the relation using the iterating or a recursive tree method to obtain the running time of the algorithm in Big-O notation.

1. (15 points) Write a C++ recursive function that finds the maximum value in an array of integers without using any loops.
   1. Test your function using different input arrays. Include the code and screenshots with testing cases.

A screenshot of a cell phone

Description automatically generated

A clock mounted to the side

Description automatically generated

* 1. Write a recurrence relation that represents your algorithm. Solve the relation and obtain the running time of the algorithm in Big-O notation.

1. (15 points) What is the best, worst and average running time of quick sort algorithm? Provide ar- rangement of the input and the selection of the pivot point at every case. Provide a recursive relation and solution for each case.

* **Best:** Big-Oh for quick sort for the best case is . The best case occurs when the pivot point is in the middle of the sequence. That makes both sides equal in terms of size. Therefore, each recursive call will divide the sequence in half.
* **Average:** Big-Oh for average case is also . The average case occurs when the pivot point is between first and last element(exclusive) and not the middle point. This situation divides the sequence differently every time. Hence, we take average of each test case.

The recursive relation is same as best case.

* **Worst:** The worst case appears when the pivot is the first or the last element in the sequence. If this happens then the next call contains n-1 elements, making it a function of running time of .

1. (10 points) What is the best, worst and average running time of merge sort algorithm? Use two methods for solving a recurrence relation for the average case to justify your answer.

Best, Average, Worst all have running time for merge sort.

* Iterative Method:

* Master Theorem:

For this case Master theorem can be written as below

T(n) = 2T(n/2) + Θ(n)

We have Therefore, we have case 2 of master theorem. Hence,

6. (10 points) R-10.17 p. 493

For the following statements about red-black trees, provide a justification for each true statement and a counterexample for each false one.

* 1. A subtree of a red-black tree is itself a red-black tree.

**False** — a subtree with a red root is not a red-black tree.

* 1. The sibling of an external node is either external or it is red.

**True** - The depth property says that all external nodes must have the same black depth, that is, the same number of black ancestors.

* 1. There is a unique (2,4) tree associated with a given red-black tree.

**True –** Because every node in a red-black tree is represented uniquely in a (2, 4) tree.

* 1. There is a unique red-black tree associated with a given (2,4) tree.

**False** – There can be two different red-black trees associated with one (2, 4) tree.

7. (10 points) R-10.19 p. 493

Consider a tree *T* storing 100,000 entries. What is the worst-case height of *T* in the following cases?

1. *T* is an AVL tree.
2. *T* is a (2,4) tree.
3. *T* is a red-black tree.
4. *T* is a binary search tree.

8. (10 points) R-9.16 p. 418

Draw an example skip list that results from performing the following series of operations on the skip list shown in Figure 9.12: erase(38), insert(48,x), insert(24,y), erase(55). Record your coin flips, as well.

A document with black text

Description automatically generated

A close up of text on a white surface

Description automatically generated

A close up of text on a white background

Description automatically generated