# 

# Department of Computing

**CS 250: Data Structures and Algorithms**

**Lab 4a: Asymptotic Complexity of Algorithms**

**CLO1: Understand the fundamentals of data structures and algorithms**

**Date: July 20, 2023**

# Instructor: Dr. Syed Imran Ali

# Lab 4a: Asymptotic Complexity Analysis

**Introduction**

This lab is based on the analysis of different algorithms.

**Objectives**

Objective of this lab is to make students analyze different algorithms and their asymptotic complexities.

**Tools/Software Requirement**

Visual Studio 2012 or gcc or g++

**Tasks**

**Task 1:**

You have already implemented a function that prints all elements of a list of size n, where n>=0. What is the Big-O complexity of that operation?

**Answer:**

O (n)

**Task 2:**

In assignment 1, you were asked to implement a function that prints all elements of a singly linked list in the **reverse order.** Your task is to answer the following questions:

1. Suppose the elements of a singly linked list are printed using an iterative approach with the help of two nested loops. What is the Big-O time complexity of printing n values in the reverse order? What is the Big-O **space complexity**?

**Answer:**

**Time complexity:** O (n^2)

**Space complexity:** O (1)

1. Suppose the elements are printed in reverse order using a recursive function given below. What is the Big-O time complexity of this function? What is the Big-O space complexity? Hint: stack, function calls!

Void RecursivePrint( node \*temp){

If (temp!=NULL){

RecursivePrint(temp->next);

cout<<temp->data;

}// end of if.

}

**Answer:**

**Time complexity:** O (n)

**Space complexity:** O (n)

**Task 3:**

Suppose you have a **singly linked list** of size **n**. Implement a function takes a position number **pos** as input from the user, and returns the value stored at that position. What is the Big-O time complexity of this function? What is its best-case time complexity?

**Answer:**

**Worst-case time complexity:** O (n)

**Best-case time complexity:** O (1)

**Task 4:**

Suppose you have an **array-based list** of size **n**. Implement a function takes a position number **pos** as input from the user, and returns the value stored at that position. What is the Big-O time complexity of this function? What is its best-case time complexity?

**Answer:**

**Worst-case time complexity:** O (1)

**Best-case time complexity:** O (1)

**Task 5:**

What is best-case and worst-case time complexity to destroy a linked list of size n?

**Answer:**

**Worst-case time complexity:** O (n)

**Best-case time complexity:** O (1)

**Task 6:**

What is best-case and worst-case time complexity to destroy an array-based list of size n?

**Answer:**

**Worst-case time complexity:** O (n)

**Best-case time complexity:** O (1)

**Task 7:**

Your task is to reverse the order of all n elements of a singly linked list using stack. Declare a stack of pointers to class node (It should store the address to an object of class node). Travers the linked list and push the address of every node onto a stack. Pop the elements and update the links by reversing order of nodes in a list. Update the start and last pointers. What is the best case and Big-O time complexities to reverse a linked list using this approach? What is its Big-O space complexity?

**Answer:**

**Worst-case time complexity:** O (n)

**Big-O space complexity:** O (n)

**Best-case time complexity:** O (n)

**Task 8:**

In the below given table,

|  |  |  |
| --- | --- | --- |
| **Operation** | **Big-O Complexity** | **Best-case Complexity** |
| Insert an element at the front of a singly linked list of size n | O (1) | O (1) |
| Insert an element at the tail end of a singly linked list of size n. **plast** points to last node. | O (1) | O (1) |
| Delete the last node of a singly linked list of size n. **plast** points to its last node. | O (n) | O (1) |
| Insertion at the front of an array list of size n | O (n) | O (1) |
| Insertion at the tail end of an array list of size n | O (n) | O (1) |
| Enqueue in a queue of length n. | O (1) | O (1) |
| Dequeue in a queue of length n. | O (1) | O (1) |
| Converting an expression of length n from infix to postfix form using stack | O (n) | O (n) |
| Finding an element via Binary Search algorithm in a sorted array-list of size n. | O (log n) | O (1) |
| Finding an element via Binary Search algorithm in an **unsorted** array-list of size n. Think about it! |  |  |

Finding an element via Binary Search algorithm in an **unsorted** array-list of size n. Think about it!

**Answer:**

Binary search algorithm is designed to work on sorted arrays. So, the array must be sorted for Binary search algorithm to work on it. Alternatively, we can use Linear search algorithm for unsorted arrays, which has a Big-O complexity of O (n) and best-case time complexity of O (1).

**Deliverable**

You are required to upload the lab tasks on LMS before the deadline.