# Department of Computing

# School of Electrical Engineering and Computer Science

**CS - 250: Data Structure and Algorithms**

**Class: BSCS 10AB**

**Lab 07 : Asymptotic Complexity of Algorithms  
&   
Sorting Algorithms**

**Date: 16th November, 2021**

**Time: 10:00 am – 12:50 pm   
&  
 02:00 pm – 4:50 pm**

# Instructor: Dr. Yasir Faheem

# Lab Engineer: Aftab Farooq

# Mashal Ashfaque

# 337203

# CS 10-A

# Lab 07 : Asymptotic Complexity Analysis & Sorting Algortihms

**Introduction**

This lab is based on the analysis of different algorithms and sorting 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**

**Part A : Asymptotic Complexity Analysis**

**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?

The big O complexity of a printing function for a list will be O(N). Because the inner loop to print/iterate through all elements of list will run n times.

**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**?

The big O time complexity for this function will be O(N2)

1. Suppose the elements are printed using a recursive function given below.   
   Void RecursivePrint( node \*temp){

If (temp!=NULL){

RecursivePrint(temp->next);

cout<<temp->data;

}// end of if.

}

What is the Big-O time complexity of this function? What is the Big-O space complexity? Hint: stack, function calls!

The big O time complexity for this function will be O(N)

**Task 3:**

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? Note: Sir allowed in class to skip code in question # 3.

The big O time complexity for this function will be O (1).

**Task 4:**

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

The big O time complexity for best case will be O (N).

The big O time complexity for worst case will be O (N).

**Task 5:**

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

The big O time complexity for best case will be O (1).

The big O time complexity for worst case will be O (1).

**Task 6:**

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). Traverse 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?

The big O time complexity for this function will be O(N)

Note:Sir allowed in class to skip code.

**Task 7:**

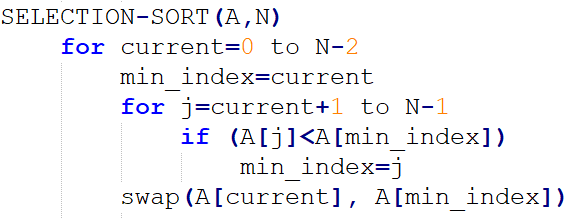
In the below given table,

|  |  |  |
| --- | --- | --- |
| **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(n) |
| Insertion at the front of an array list of size n | O(N) | O(n) |
| Insertion at the tail end of an array list of size n | O(1) | 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 | N2 | N2 |
| Finding an element via Binary Search algorithm in a sorted array-list of size n. | Log(n) | 1 |
| Finding an element via Binary Search algorithm in an **unsorted** array-list of size n. Think about it! | impossible | 1 |

**Part B : Sorting Algorithms**

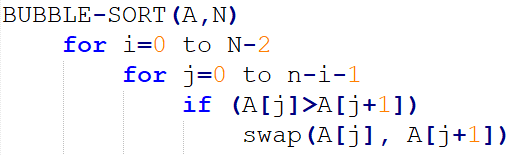
**Selection Sort:**

Selection sort is a popular sorting algorithm, which is quite simple to implement. The pseudo code is as follows:



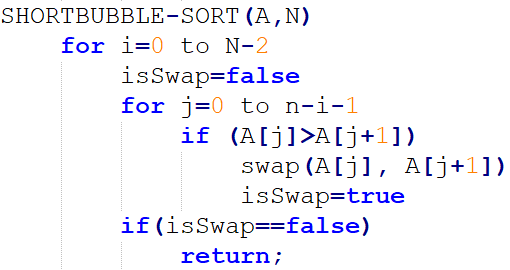
**Bubble Sort:**

Insertion sort is a popular sorting algorithm, which is quite simple to implement. The pseudo code is as follows:



**Short Bubble Sort:**

It is a variant of Bubbles sort the best-case complexity of which is Ω (n). Note that the best-case and worst-case complexities of Bubble sort are of order n2.



**Task 1:**

Implement Selection sort, Bubble sort, Short Bubble sort algorithms in C++.  
  
Declare an array of large size e.g. 1000 or more, randomly store values in all the indexes in the range e.g. 1 to 3000. Then call the sort functions one by one. Do multiple runs of the program and note both clock time and the no of iterations.

**Selection Sort**

#include <iostream>

#include <chrono>

using namespace std;

using namespace std::chrono;

void selectionSort(int arr[], int n);

void bubbleSort(int arr[], int n);

void printarr(int[]);

int main()

{

int arr[1000];

//initializing with random data

for (int i = 0; i < 1000; i++)

{

arr[i] = rand() % 3000;

}

//sorting the array

auto start = high\_resolution\_clock::now(); //for calculating time

selectionSort(arr, 1000);

auto stop = high\_resolution\_clock::now();

auto duration = duration\_cast<microseconds>(stop - start);

cout<<"\nTime for execution using selection sort: " << duration.count() << endl;

for (int i = 0; i < 1000; i++)

{

arr[i] = rand() % 3000;

}

// bubbleSort(arr, 1000);

}

void printarr(int arr[])

{

for (int i = 0; i < 1000; i++)

{

cout << arr[i] << " ";

}

}

void swap(int\* x, int\* y)

{

int temp = \*x;

\*x = \*y;

\*y = temp;

}

void selectionSort(int arr[], int n)

{

int i, j, min\_idx;

// One by one move boundary of unsorted subarray

for (i = 0; i < n - 1; i++)

{

// Find the minimum element in unsorted array

min\_idx = i;

for (j = i + 1; j < n; j++)

if (arr[j] < arr[min\_idx])

{

min\_idx = j;

}

// Swap the found minimum element with the first element

swap(&arr[min\_idx], &arr[i]);

}

}





Bubble Sort:

#include <iostream>

#include <chrono>

using namespace std;

using namespace std::chrono;

void bubbleSort(int arr[], int n);

void printarr(int[]);

int main()

{

int arr[1000];

//initializing with random data

for (int i = 0; i < 1000; i++)

{

arr[i] = rand() % 3000;

}

//sorting the array

auto start = high\_resolution\_clock::now(); //for calculating time

bubbleSort(arr, 1000);

auto stop = high\_resolution\_clock::now();

auto duration = duration\_cast<microseconds>(stop - start);

cout<<"\nTime for execution using bubble sort: " << duration.count() << endl;

}

void printarr(int arr[])

{

for (int i = 0; i < 1000; i++)

{

cout << arr[i] << " ";

}

}

void swap(int\* x, int\* y)

{

int temp = \*x;

\*x = \*y;

\*y = temp;

}

void bubbleSort(int arr[], int n)

{

int i, j;

for (i = 0; i < n - 1; i++)

// Last i elements are already in place

for (j = 0; j < n - i - 1; j++)

if (arr[j] > arr[j + 1])

swap(&arr[j], &arr[j + 1]);

}





**Deliverables:**

Compile a single word document by filling in the solution part and submit this Word file on LMS. The name of word document should follow this format. i.e. **YourFullName(reg)\_Lab#.** This lab grading policy is as follows: The lab is graded between 0 to 10 marks. The submitted solution can get a maximum of 5 marks. At the end of each lab or in the next lab, there will be a viva related to the tasks. The viva has a weightage of 5 marks. Insert the solution/answer in this document. You must show the implementation of the tasks in the designing tool, along with your complete Word document to get your work graded. You must also submit this Word document on the LMS. In case of any problems discuss it by emailing it to [aftab.farooq@seecs.edu.pk](mailto:aftab.farooq@seecs.edu.pk).

**Note:** Students are required to upload the lab on LMS before deadline.

Use proper indentation and comments. Lack of comments and indentation will result in deduction of marks.