Instituto Tecnológico de Costa Rica

Engineering School

Computer Engineering

CE-3104 - Lenguajes, Compiladores e Interpretes

II Semester 2017

Project

Sorting Algorithms

Jóse Andres Rivera Tencio, 2015045476

Kenneth Jeanpol Alvarado Mendez, 2015095715

Group 01

Professor: Marco Rivera

November 15, 2017

Contents

1 Abstract **3**

2 Objectives **3**

2.1 General Objective . . . 3

2.2 Specific Objectives . . . 3

3 User Guide **3**

4 Description of the implemented functions. **4**

4.1 BubbleSort . . . 4

4.2 SelectionSort . . . 4

4.3 Insertion Sort . . . 4

4.4 Merge Sort . . . 5

4.5 Quick Sort . . . 5

5 Description of the data structures developed. **5**

5.1 Struct . . . 5

5.2 Node . . . 5

5.3 Lists . . . 5

5.4 Doubly linked list . . . 5

6 Known Issues **6**

6.1 Selection Sort Memory . . . 6

7 Problems Found **6**

7.1 Run-time error with BubbleSort Algorithm . . . 6

7.1.1 Problem . . . 6

7.1.2 Description . . . 6

7.1.3 Solution Tries . . . 6

7.1.4 Solution . . . 6

7.1.5 Advice . . . 6

7.1.6 Conclusion . . . 6

7.1.7 Links . . . 6

8 Timesheet **7**

9 Work Log **7**

10 Conclusions **8**

1 Abstract

Abstract

One of the fundamental issues in computer science is ordering a list of items. Although there is a huge number of sorting algorithms, sorting problem has attracted a great deal of research; because efficient sorting is important to optimize the use of other algorithms. This paper aims to demonstrate the different run-times between some of the most popular sorting algorithms.
Keywords: Sort, Node, List, Swaps, Time Complexity.

2 Objectives

2.1 General Objective

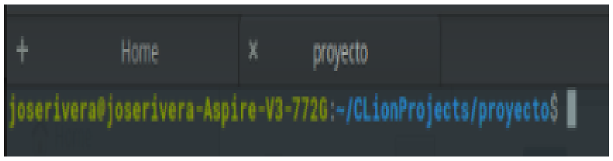
Develop an application that reaffirms the knowledge of the paradigm of imperative programming.

2.2 Specific Objectives

- Develop an application in the programming language C.
- Apply the concepts of mandatory programming.
- Create and manipulate lists as data structures.

3 User Guide

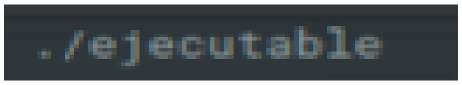
- First, you must open the terminal on Linux platform



```
joserivera@joserivera-Aspire-V3-772G:~/CLionProjects/proyecto$
```

Figure 1

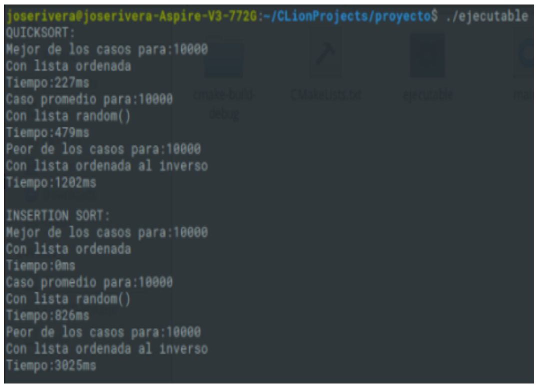
- Then, enter to the directory where the executable is saved. You may access to the directory, typing the cd command on terminal follow by the path that you want to access.



```
./ejecutable
```

Figure 2

- Once you are in the director where the file is saved, type './ejecutable' to run the executable



```
joserivera@joserivera-Aspire-V3-772G:~/CLionProjects/proyecto$ ./ejecutable
QUICKSORT:
Mejor de los casos para:10000
Con lista ordenada
Tiempo:227ms
Caso promedio para:10000
Con lista random()
Tiempo:479ms
Peor de los casos para:10000
Con lista ordenada al inverso
Tiempo:1202ms

INSERTION SORT:
Mejor de los casos para:10000
Con lista ordenada
Tiempo:0ms
Caso promedio para:10000
Con lista random()
Tiempo:826ms
Peor de los casos para:10000
Con lista ordenada al inverso
Tiempo:3025ms
```

Figure 3

Immediately will be shown the run-time from the 5 sorting algorithms proposed in this project

4 Description of the implemented functions.

4.1 BubbleSort

is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. Although the algorithm is simple, it is too slow and impractical for most problems

4.2 SelectionSort

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.

- The subarray which is already sorted.
- Remaining subarray which is unsorted.

In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.

4.3 Insertion Sort

Here, a sub-list is maintained which is always sorted. For example, the lower part of an array is maintained to be sorted. An element which is to be inserted in this sorted sub-list, has to find its appropriate place and then it has to be inserted there.

4.4 Merge Sort

Divide the unsorted list into n sublists, each containing 1 element (a list of 1 element is considered sorted). Repeatedly merge sublists to produce new sorted sublists until there is only 1 sublist remaining. This will be the sorted list.

4.5 Quick Sort

Quicksort first divides a large array into two smaller sub-arrays: the low elements and the high elements. Quicksort can then recursively sort the sub-arrays. Basically, it picks an element, called a pivot, from the array.Then reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position. Recursively applies the above steps to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values.

5 Description of the data structures developed.

5.1 Struct

A struct in the C programming language is a composite data type declaration that defines a physically grouped list of variables to be placed under one name in a block of memory, allowing the different variables to be accessed via a single pointer, or the struct declared name which returns the same address.

5.2 Node

A "node" is a basic unit used in computer science. Nodes contain data and also may link to other nodes. Links between nodes are often implemented by pointers.

5.3 Lists

The linked list is an TDA that allows us to store data in an organized way, just like the vectors but, unlike these, this structure is dynamic, so we do not have to know "a priori" the elements that can contain. Lisp is based on List management

5.4 Doubly linked list

In computer science, a doubly linked list is a linked data structure that consists of a set of sequentially linked records called nodes. Each node contains two fields, called links, that are references to the previous and to the next node in the sequence of nodes.

6 Known Issues

6.1 Selection Sort Memory

With the Selection Sort Algorithm, can not sort huge amounts of elements because it uses all the RAM.

7 Problems Found

7.1 Run-time error with BubbleSort Algorithm

7.1.1 Problem

BubbleSort Algorithm did not compiled on Linux platform, just on windows.

7.1.2 Description

Predefined C compiler on Clion at Linux did not compile the BubbleSort algorithm implemented.

7.1.3 Solution Tries

- Use another IDE in order to test out the BubbleSort
- Implement another BubbleSort code.

7.1.4 Solution

The problem was that BubbleSort did not support more than 100 000 elements, so when compiling the program failed.

7.1.5 Advice

Research about the limits of the algorithms implemented, in order to prevent any bug on the program.

7.1.6 Conclusion

Besides BubbleSort is easy to implement, it is not practical because it does not support a huge amount of elements.

7.1.7 Links

[Geeksforgeeks]Geeksforgeeks. (2017, 10 15). *http://www.geeksforgeeks.org/c-program-bubble-sort-linked-list/* consultado el 14/11/2017

8 Timesheet

Table 1: Invested time on each task

Activity	Performance	Date	Time
José Rivera	Implements Merge Sort	11/10/2017	3
José Rivera	Implements Insertion Sort	12/10/2017	3
José Rivera	Implements Quick Sort	13/10/2017	3
José Rivera	Implements Basic list methods	14/11/2017	3
José Rivera	Documentation	14/11/2017	4
Kenneth Alvarado	Implements Insertion Selection Sort	11/11/2017	2
Kenneth Alvarado	Implements Bubble Sort	12/11/2017	3
Kenneth Alvarado	Development of the Paper	13/11/2017	4
Kenneth Alvarado	Graph the run-time obtained	14/11/2017	3
Kenneth Alvarado	Documentation	14/11/2017	4

9 Work Log

Table 2: Invested time on each task

Performance	Date	Time
Research on work in general	11/10/2017	1
Research about sorting algorithms.	11/10/2017	2
Research on how to get time in C	11/10/2017	1
Creation of data graphics.	11/11/2017	2
Documentation	14/11/2017	4

10 Conclusions

- Do not try to do a whole problem at once. Apply the divide and conquer to solve complex problems.
- Pointers are a very powerful tool for data structures, but require a great responsibility in their management. Always check that the pointers used are released later and fulfill their function.

References

[1] [Brian W. Kerngham] *The C Programming Language* consultado el 14/11/2017