WASCAL
West African Science Service Centre on Climate Change and Adapted Land Use

UNIVERSITE DE OUAGADOUGOU
BURKINA FASO
POPULI SAPIENTIA POPULO

SPONSORED BY THE
Federal Ministry of Education and Research

Doctoral School in Informatics for Climate Change

Algorithm and Programming 2020 – 2021

Report on Program

By

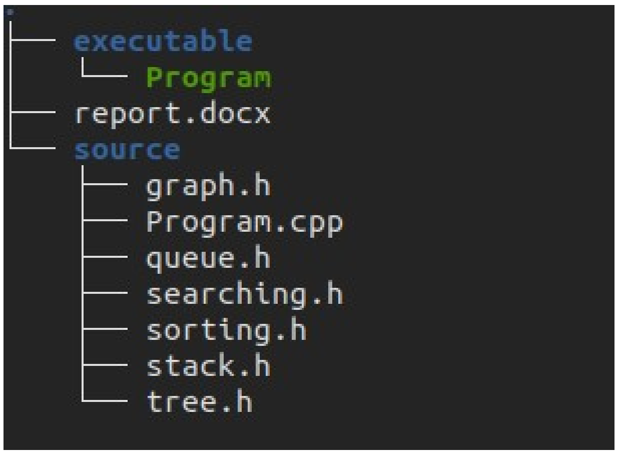
BOUBAKAR Zourkalaini

I. Program Presentation

This program is the product of programming project on Algorithm and Programming course (ICC 2021). The aim of the project is to validate the acquired skills in programming by developing a program under Ubuntu in C++.

The program is a Command Line Interface.

The submitted work folder is structured as follow:



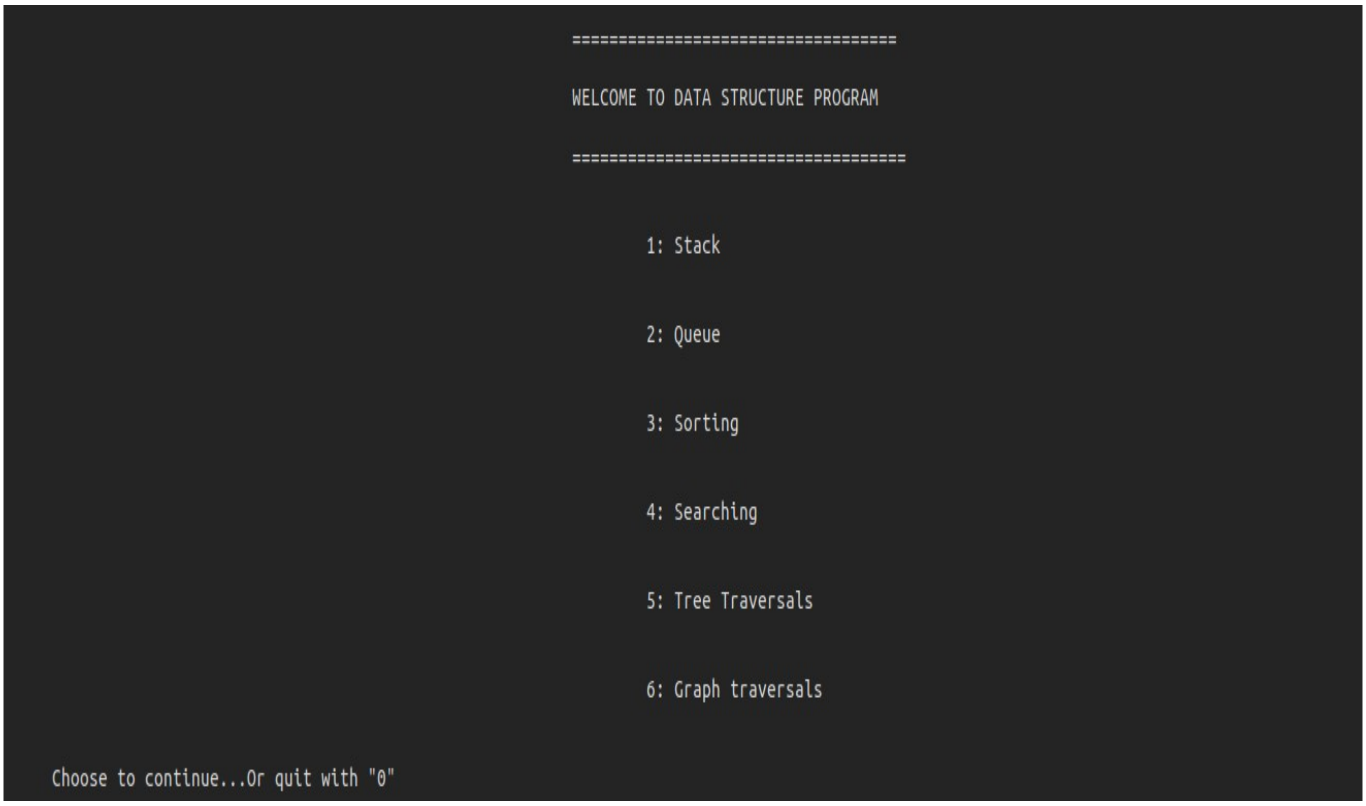
```
.
├── executable
│   └── Program
├── report.docx
└── source
    ├── graph.h
    ├── Program.cpp
    ├── queue.h
    ├── searching.h
    ├── sorting.h
    ├── stack.h
    └── tree.h
```

“Program” file is the executable program and “Program.cpp” is the source program. The features of the program are encapsulated in “.h” files, which are used by the main program (Program.cpp).

In source folder, all files are documented and contain the necessary information to understand and if needed to modify the source code.

II. Program Feature

When executed, the user interface looks like the image below.



```
                              ==================================

                              WELCOME TO DATA STRUCTURE PROGRAM

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


                                     1: Stack


                                     2: Queue


                                     3: Sorting


                                     4: Searching


                                     5: Tree Traversals


                                     6: Graph traversals


Choose to continue...Or quit with "0"
```

1. Stack

It contains the implementation of Stack data structure. When choose, the user can create a new stack by providing the size. After creating a new stack, the available operations are:

- MakeEmpty: to make the stack empty by clearing all its items.

-IsEmpty: test whether the stack is empty. If the stack is empty, it will display “Yes” otherwise “No”.

-IsFull: test whether the stack is full. If the stack is full, it will display “Yes” otherwise “No”.

-Push: push an element on top of the stack.

-Pop: pop out the element on top of the stack.

-DisplayStack: print all the element of the stack.

For the operation on the stack, the data are provide by the user. The implementation simulates a record of student name and their age. So when pushing, the user is asked to give the **name** and the **age** of a student.

2. Queue

It contains the implementation of Queue data structure. When choose, the user can create a new queue by providing the size. After creating a new stack, the available operations are:

- MakeEmpty: make the queue empty by clearing all its items.

-IsEmpty: test whether the queue is empty. If the queue is empty, it will display “Yes” otherwise “No”.

-IsFull: test whether the queue is full. If the queue is full, it will display “Yes” otherwise “No”.

-Enqueue: insert an element at the rear of the queue.

-Dequeue: delete the element at the front of the queue.

-DisplayStack: print all the element of the stack.

For the operation on the queue, the data are provide by the user. The implementation simulates a record of student name and their age. So when inserting in the queue, the user is asked to give the **name** and the **age** of a student.

3. Sorting

It contains Selection sort and Merge sort algorithm. Both required input data (integers) from the user. Each algorithm, after receiving input data, displays the output, which is the sorted input data.

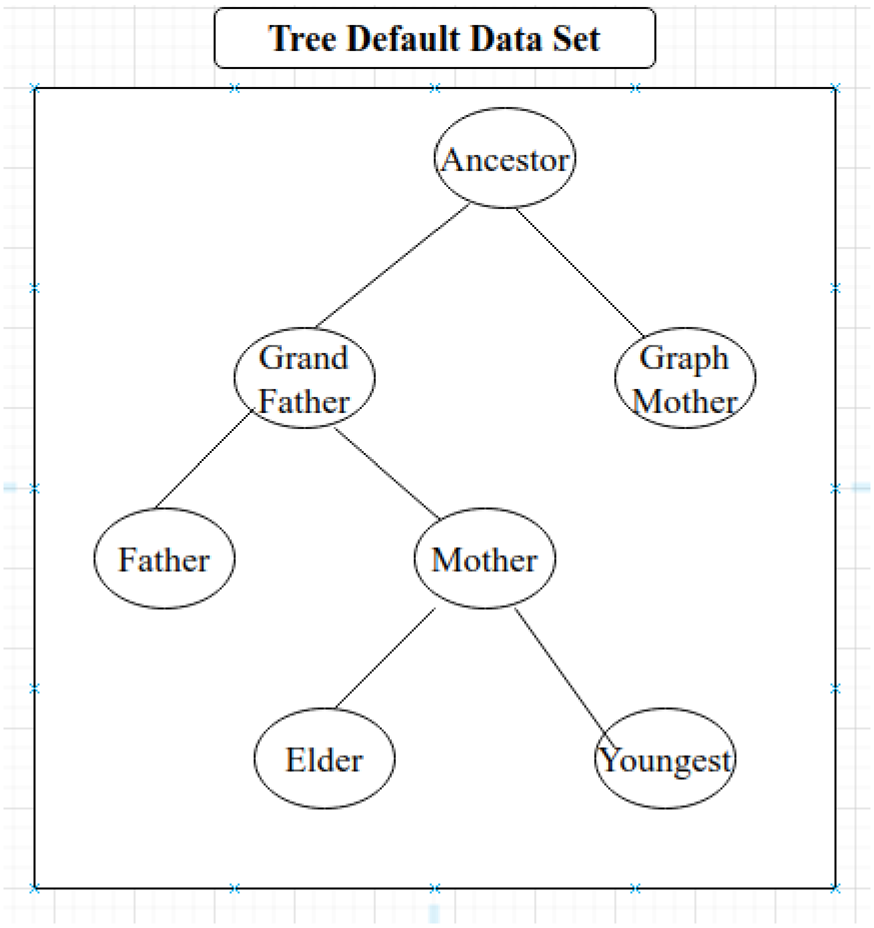
4. Searching

It contains Sequential search and Binary search algorithm. Both use a default data set (integers). The user is ask to give the element (integer) that he is looking for. The result is displayed after giving the element.

The default data set can be modified by editing the source code.

5. Tree traversals

It contains the implementation of tree traversals algorithm. It uses a default tree as follow:



the available operations are:

- Preoder: print tree nodes in preorder (root-left-right).

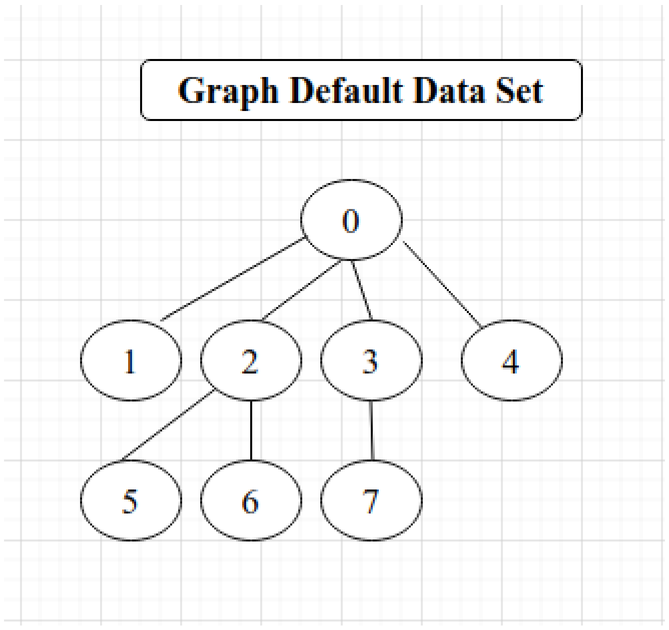
-Inorder: print tree nodes in inorder (left-root-right).

-Postorder: print nodes in postorder (left-right-root)

The default data set can be modified by editing the source code

6. Graph traversals

It contains the implementation of graph traversals algorithm. It uses a default tree as follow:



the available operations are:

- Depth First traversal: display nodes in depth first traversal.

- Breadth First traversal: display node in breadth first traversal.

The default data set can be modified by editing the source code