VIETNAM NATIONAL UNIVERSITY, HCMC

UNIVERSITY OF SCIENCE

FALCULTY OF INFORMATICS TECHNOLOGY

ADVANCED PROGRAM IN COMPUTER SCIENCE

TECHNICAL REPORT

Project: Data Structure Visualization

CS163 - Data Structures

Student:
Hoang Duy Vu
Student ID: 32125022

Lecturers:
Thanh Ho Tuan, M.Sc
Loc Truong Phuoc, M.Sc

Mục lục

§1 Introduction

When studying Information Technology, particularly competitive programming, I found myself fascinated by tree structures and graphs. This is because they are among the most widely applied structures today. In the Data Structures and Algorithms course, we were introduced to most of the common trees such as Binary Search Tree, AVL Tree, 2-3 Tree, 2-3-4 Tree, Heap Tree, etc. As I learned about these trees, my interest in tree structures grew, which motivated me to challenge myself with this project. I hope that by completing this project, I will gain a deeper understanding of the nature of tree structures and be able to apply them in practice.

This Data Structure Visualization project will help you understand how various tree operations work, such as adding nodes, deleting nodes, or searching for nodes, in the most intuitive and comprehensible way. In this project, I will explore the following tree types: **Heap Tree, AVL Tree, 2-3-4 Tree, Trie**, as well as **Hash Table** structures and **graphs**.

§2 Features

§2.1 Available data structures

- Heap Tree (Min Heap)
- AVL Tree
- 234 Tree
- Hash Table
- Trie
- Graph

§2.2 Features of data structures

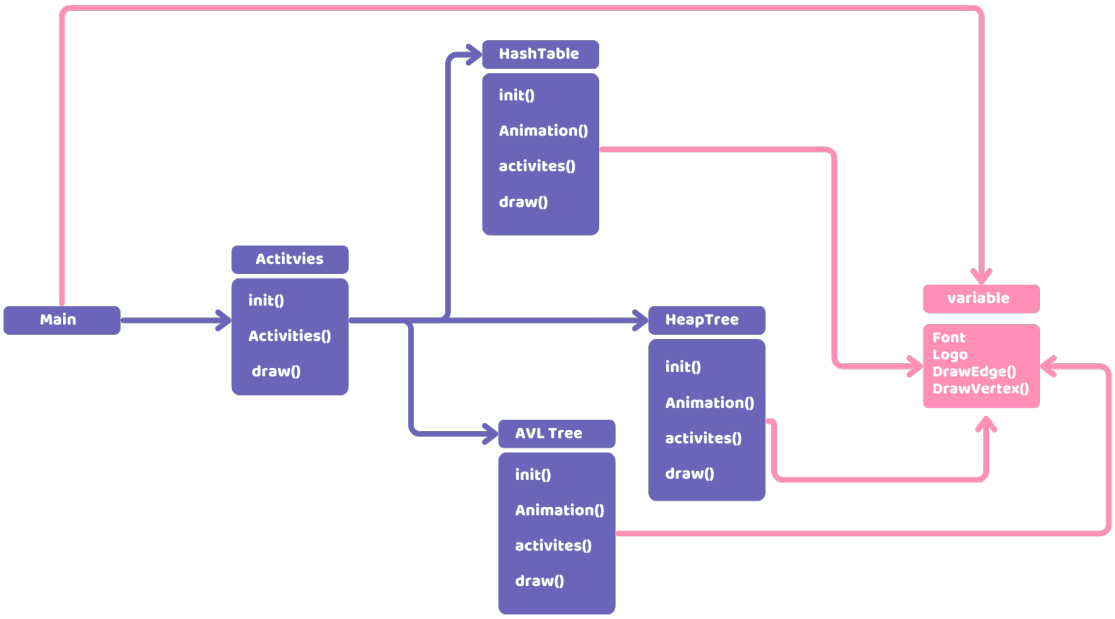
Feature ID	Feature Description	Grade	Graded by Student
	Hash table		
1	Init from file	1	1
2	Randomized data	1	1
3	Insert (run step by step)	2	2
4	Delete (run step by step)	2	2
5	Search (run step by step)	2	2
	AVL tree		
6	Init from file	1	1
7	Randomized data	1	1
8	Insert (run step by step)	2	2
9	Delete (run step by step)	2	2
10	Search (run step by step)	2	2
	234 tree		
11	Init from file	1	1
12	Randomized data	1	1
13	Insert (run step by step)	2	2
14	Delete (run step by step)	2	2
15	Search (run step by step)	2	2
	Min heap or Max heap		
16	Init from file	1	1
17	Randomized data	1	1
18	Insert (run at once)	2	2
19	Delete (run at once)	2	2
20	Get top	1	1
21	Size	1	1
	Trie		
22	Init from file	1	1
23	Randomized data	1	1
24	Insert (run at once, highlight new)	2	2
25	Delete (run at once)	2	2
26	Search (run step by step)	2	2
	Graph		
27	Init from file, from matrix	1	1
28	Randomized data	1	1
29	Connected components	1	1
30	Minimum spanning tree	1	1
	Color, size & style		
31	Hash table	1	1
32	Binary tree	1	1
33	AVL tree	1	1
34	234 tree	1	1
35	Heap	1	1

§2.3 Algorithm Description

§2.3.1 Introduction

- The software is written in C++ and utilizes the Raylib library to support graphics.
- For each data structure, I use a class to store it. Each class contains almost everything needed to implement both the algorithm and the graphics part.
- Each class is stored in an HPP file, and a CPP file is used to handle the functions within that class.
- Next, I have a variable file to store functions and global variables so that all files can access and use them.
- Finally, there is an activities file that helps navigate operations and select the appropriate class based on user commands.
- Below is the class diagram to describe the files.

§2.3.2 Class Diagram



§2.4 Variable

Below are some images of certain functions and variables in the variable file.

```
struct Transforms{
    float progress;
    Vertex u,v;
    Transforms() {}
    Transforms(Vertex u,Vertex v): u(u), v(v) {}
};

struct TransformsEdge{
    float progress;
    Edge u,v;
    TransformsEdge() {}
    TransformsEdge(Edge u,Edge v): u(u), v(v) {}
};

void DrawVertex(Vector2 Postion,float radius,int val, int kind_color,unsigned char a);
void DrawVertexText(Vector2 Postion,float radius,char *text, int fontSize,unsigned char a);
void DrawEdge(Vector2 PostionX,Vector2 PostionY,int val, int kind_color,unsigned char a);
void DrawEdge2(Vector2 PostionX,Vector2 PostionY,int val, int kind_color,unsigned char a);
void DrawVertexRoot(Vector2 Postion,float radius,int val, int kind_color,unsigned char a);
void DrawVertexL(Vector2 Postion,float radius,int val, int kind_color,unsigned char a);
```

```
extern std::mt19937 rng;
extern const int screenWidth;
extern const int screenHeight;
extern const Color ConstColor1;
extern const Color ConstColor2;
extern const int Limitnode;
extern bool Loadfile;

extern Font customFont;
extern double deltaTime;

extern  int sel_n;
extern  int sel_v;
extern  int sel_k;
extern  int sel_i;
extern  int LimitNode;
extern char pathfile[40];
```

§2.5 HeapTree

Below are some images of certain functions and variables in the HeapTree class.

```
1096  public:
1097      void create(int n);
1098      void insert(int val);
1099      void ExtractMin(int k);
1100      void deletee(int i);
1101      void update(int i,int v);
1102      void loadfile();
1103      void init();
1104      void draw();
1105      void UpdatePostionNodePer();
1106      int UpdatePressOn();
1107      void Activity();
1108      void SelectPress(int pos);
1109      void SolveRemote();
1110      void Notification();
1111      Vector2 NewPos2D(Vector2 A,Vector2 B,float g);
1112      float NewPos1D(float x, float y, float g);
1113      void DrawAnimation(std::vector<Transforms> f,double g);
1114      void DrawAnimationEdge(std::vector<TransformsEdge> f,double g);
1115  };
```

§2.6 User Guide

§2.6.1 Main Menu

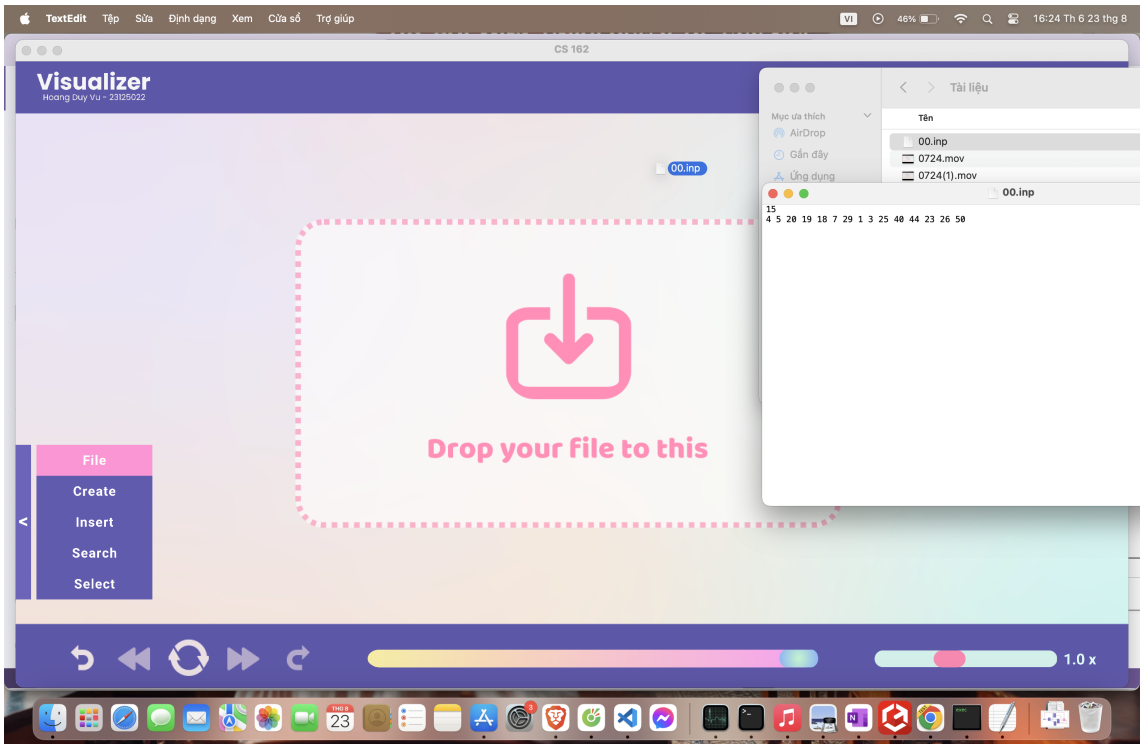
With a user-friendly interface in mind, the main screen is designed to be simple and easy to navigate. To select the data structure you want, just click on its icon with the left mouse button.

As shown here, I will select the AVL tree.

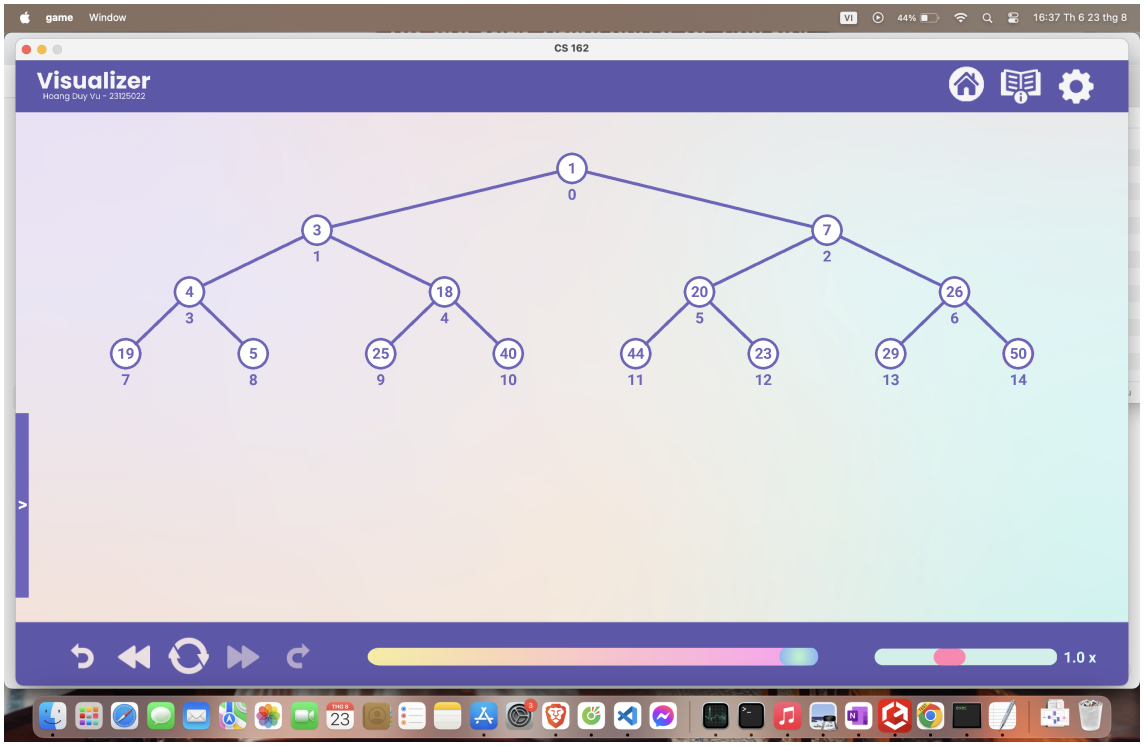
§2.6.2 Heap Tree

Since all data structures have the same button functions, I will provide usage instructions through the Heap Tree.

Input File

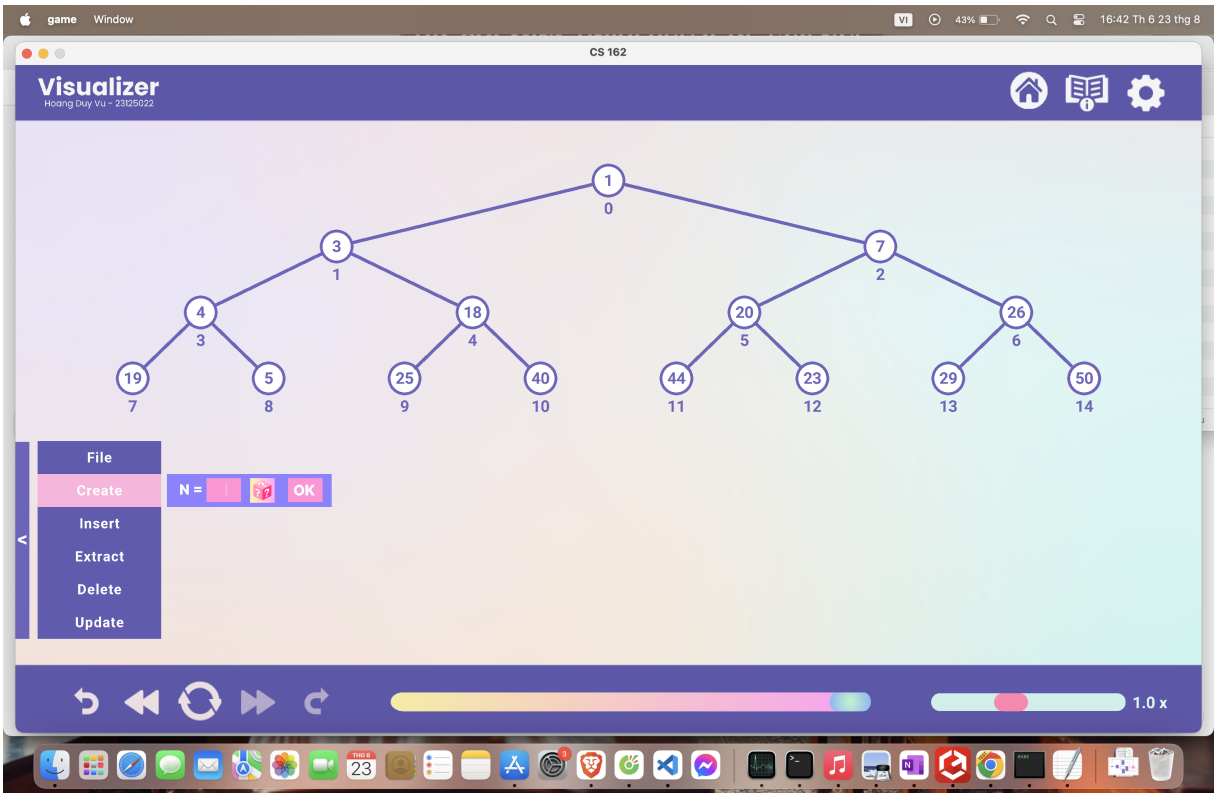


I will prepare a text file in advance with a format consisting of the number n n and n n vertices, then drag it onto the screen, and the application will automatically build a Heap Tree for me.



Heap Tree after dropping the file in.

Create



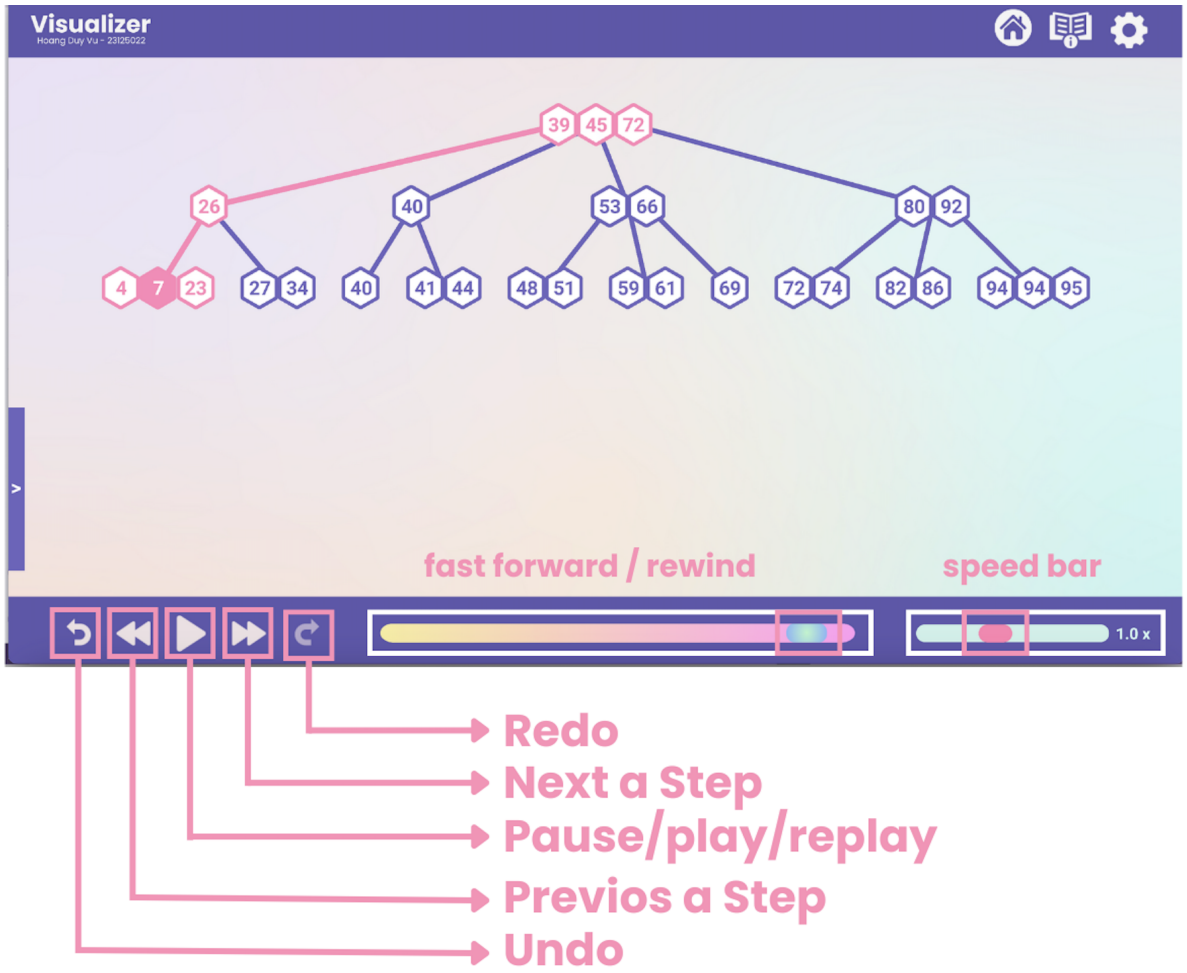
I have two options:

- Random - I will click on the dice icon to randomly generate vertices.
- Keyboard Input - I will enter values from the keyboard to get the desired number of vertices. However, in the HeapTree, I am limiting it to a maximum of 30 vertices, so it will automatically adjust to 30 if the input value exceeds that number. Finally, press OK to start.

Other Functions

These are similar to the above sections, so you can use the app to experience it in the best way possible.

§2.7 Bonus features: Tool bar



§3 Github

- Link of repository: CS163
- Number of commits: 21 commits