**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МО ЭВМ**

# КУРСОВАЯ РАБОТА

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: Исследование рандомизированной дерамиды**

Студент гр. 9384 _____ Нистратов Д.Г.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

**ЗАДАНИЕ**

**НА КУРСОВУЮ РАБОТУ (КУРСОВОЙ ПРОЕКТ)**

Студент Нистратов Д.Г.

Группа 9384

Тема работы : Исследование рандомизированной дерамиды

Исходные данные:

GUI, разработанный на языке C++ с фреймворком “Qt”. Данные массива, генерируемые пользователем или случайные данные, генерируемые стандартной функцией C++ Rand().

Содержание пояснительной записки:

«Содержание» «Введение» «Анализ задачи» «Визуализация» «Сравнение данных» «Тестирование» «Заключение» «Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 15.11.2020

Дата сдачи реферата: 28.12.2020

Дата защиты реферата: 28.12.2020

Студент _____ Нистратов Д.Г.

Преподаватель _____ Ефремов М.А.

## АННОТАЦИЯ

В данном курсовом проекте осуществлено исследование методов вставки и удаления данных из рандомизированной дерамиды. Для отображения результатов вставки и удаления, реализовано отображение статистики в виде графиков. Также были реализованы случаи худшей вставки и удаления для сравнения результатов.

## SUMMARY

In this course project, a study of methods for inserting and deleting data from a randomized treap was carried out. To display the results of insertion and deletion, statistics are displayed in the form of graphs. Worse insertion and deletion cases were also implemented to compare the results.

**СОДЕРЖАНИЕ**

# ВВЕДЕНИЕ

Основной целью данной работы является исследование методов вставки и удаления данный в дерамиде, а также сравнения данных в худшем и лучшем случаях.

# 1. АНАЛИЗ ЗАДАЧИ

## 1.1. Основные теоретические сведения

Дерамида (Treap) – это дата структур, являющаяся комбинацией бинарного дерево (binary Tree) и бинарной кучи (heap). (tree + heap ⇒ Treap)

Более точнее, это структура, хранящая в себе пары двух элементов (X, Y). Где левый элемент всегда меньше корня, а правый – больше. Второй элемент является приоритетом вставки, при котором каждый левый и правый элемент приоритета больше или меньше корня.

Более точнее, это структура, хранящая в себе пары двух элементов (X, Y). Где левый элемент всегда меньше корня, а правый – больше. Второй элемент является приоритетом вставки, при котором каждый левый и правый элемент приоритета больше или меньше корня.

Преимущества такой организации данных:

Если бы приоритетов не было, то обычному бинарному дереву поиска соответствовало множество деревьев, некоторые из которых являются вырожденными (например в виде цепочки), а поэтому чрезвычайно медленные (операции добавления и удаления выполнялись бы за O(n)).

Дерамида описывается следующим образом:

Данные:

Value – значение

R – приоритет

Функции:

Split – разделяет дерево на два поддерева таким образом, что одно содержит все элементы меньше чем заданные, а второе содержит все элементы больше чем заданные.

Merge – обратная функция split, объединяет 2 поддерева и возвращает новое дерево. Функция находит дерево с наибольшим приоритетом и вызывает себя из другого дерева и наследника поддерева.

Операция реализованные в дерамиде:

Insert – вставка элемента

Erase – удаление элемента

Методы insert и erase аналогичны тем же методам в обычном бинарном дереве поиска, за исключением того что нам необходимо отслеживать приоритет каждого элемента дерева. Для этого используются функции split и merge.

Реализация insert – осуществляется спуск по дереву до первого элемента, в котором приоритет оказался меньше, далее функцией split мы разделяем дерево и вставляем новый элемент на место остановки заменяя наследника.

Реализация erase – осуществляется спуск по дереву до нахождения элемента. Найдя элемент вызывается функция merge от левого и правого наследника, а возвращаемое дерево, вставляется на место найденного элемента.

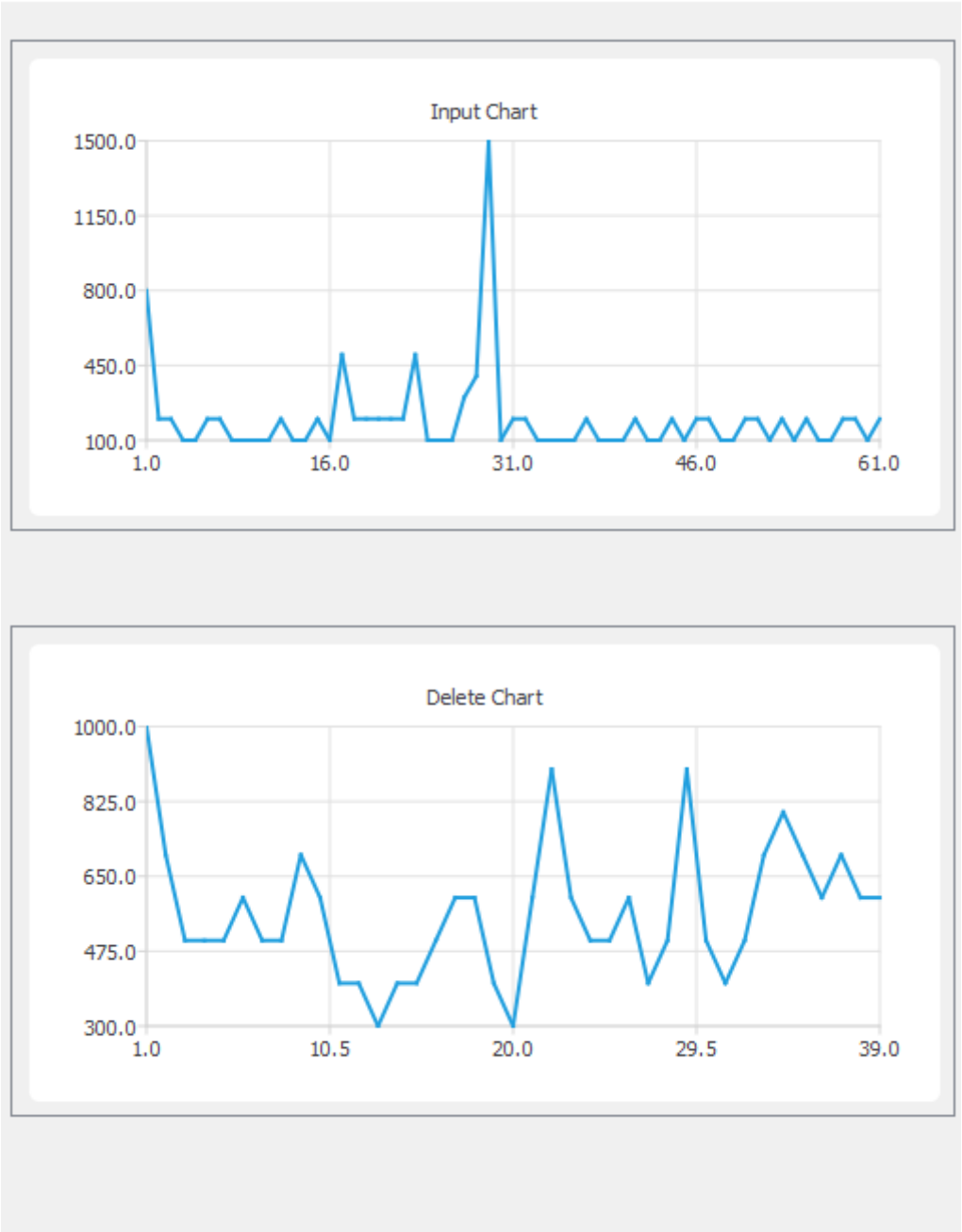
Также был осуществлен обход по дереву за O(n) для вывода данных в дереве.

## 2. ВИЗУАЛИЗАЦИЯ

Визуализация полученной статистики выполняется с помощью классов QGraphicsScene, QGraphicsView и QCharts фреймворка Qt. Реализовано отрисовка дерева, а также 2 графика показывающие время и кол-во элементов. Первый график показывает время, затраченное на вставку элемента. По оси x – количество элементов, по оси y – затраченное время. Второй график показывает время, затраченное на удаление элемента. По оси x – количество удалённый элементов, по оси y – затраченное время.

Пример графиков на изображении 1.

Изображение 1.

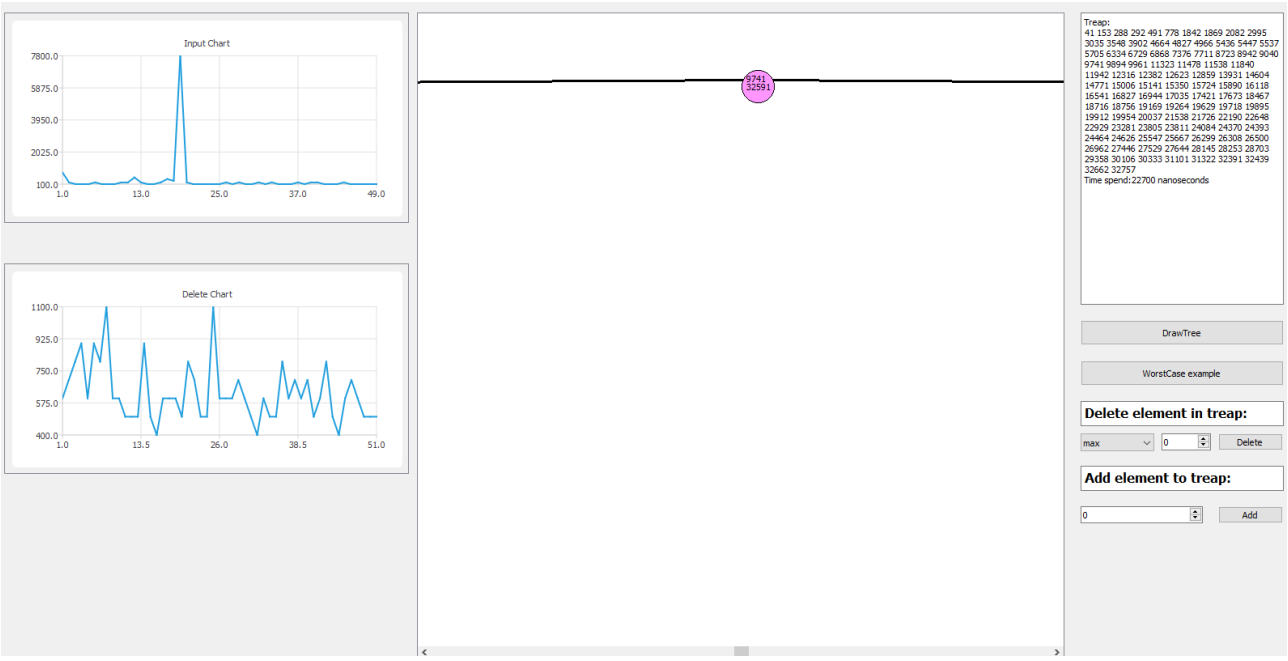


# 3. СРАВНЕНИЕ ДАННЫХ

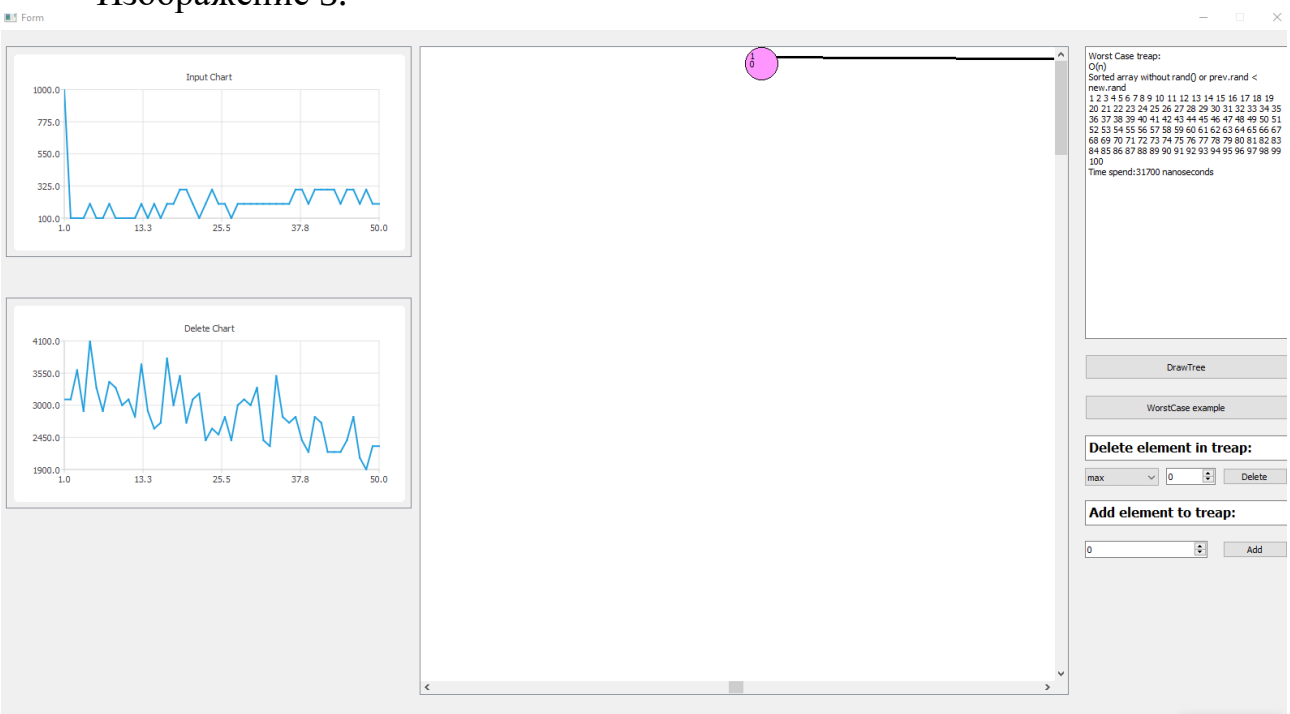
На изображении 2 представлен результат работы программы в среднем случае.

На изображении 3 представлен результат работы программы в худшем случае.

Изображение 2.



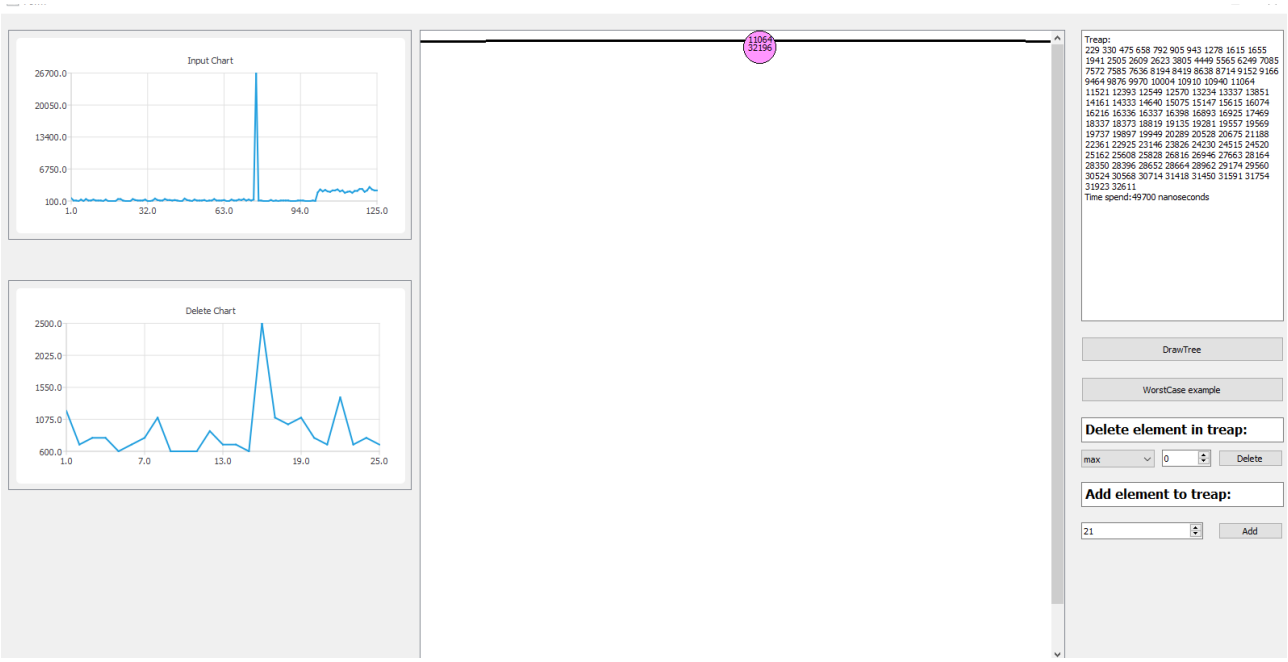

Изображение 3.



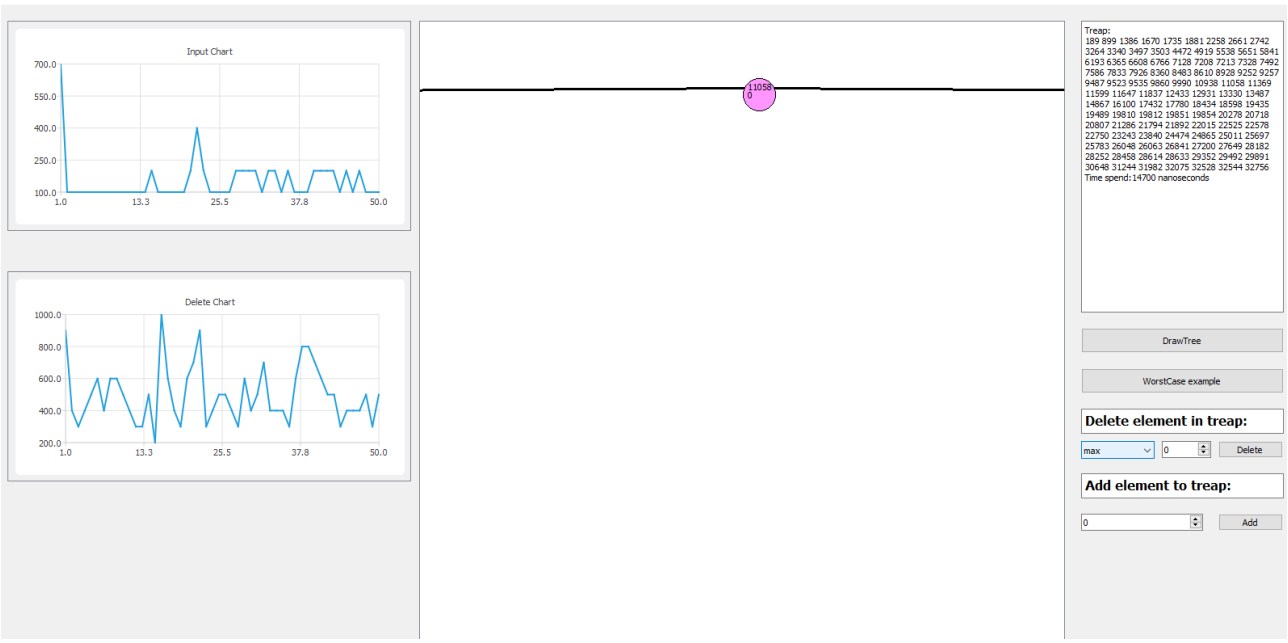
На изображении 2 видно что методы вставки и удаления имеют логарифмическую сложность(O(log(n))). Однако если сравнивать с изображением 3, где вставка и удаление имеют логарифмическую сложность(O(N)),т.к  массив чисел отсортирован и приоритеты каждого элемента <= приоритета следующего элемента, то мы можем заметить существенные различия во времени выполнения удаления и вставки элементов.

# 4. ТЕСТИРОВАНИЕ

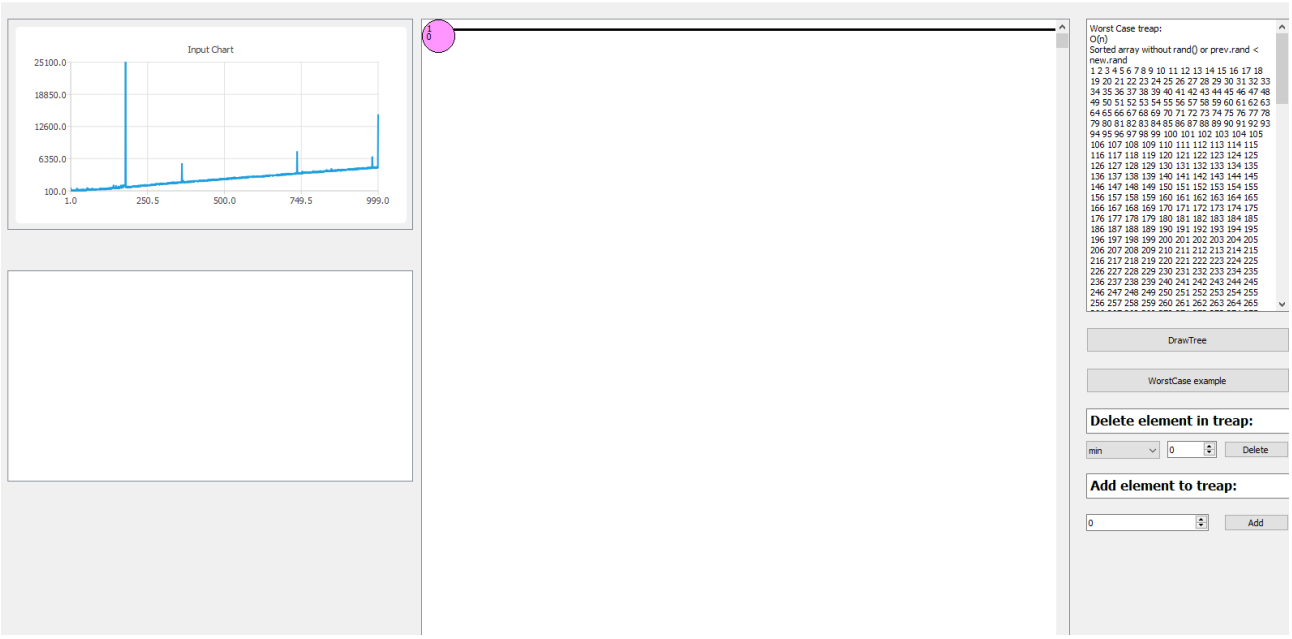
Приоритеты включены, кол-во элементов 100. Добавлено 50 новых элементов, удалено 25.




Приоретеиы выключены, кол-во элементов 100. Удалено 50.



Худший случай, 999 элементов. Удалено 0



# ЗАКЛЮЧЕНИЕ

При выполнении работы были исследованы методы вставки и удаления из рандомизированной дерамиды, а также была реализовано отображение данных в виде графика и построенного дерева. В ходе исследования была показана сложность вставки и удаления элементов в дерамиде.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Научная статья // cs.umd.edu URL: https://www.cs.umd.edu/class/fall2019/cmsc420-0201/Lects/lect07-treap-skip.pdf (дата обращения 29.11.2020)

# ПРИЛОЖЕНИЕ А
# ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: input.h

```
#ifndef INPUT_H
#define INPUT_H

#include <QWidget>
#include <QValidator>
#include <QRegExpValidator>
#include "startmenu.h"

namespace Ui {
class input;
}

class input : public QWidget
{
    Q_OBJECT

public:
    explicit input(QWidget *parent = nullptr);
    ~input();
    std::string getline();

    void setarrsize(int n);
    void setPrior(bool n);

public slots:
    void on_ok_clicked();
    void on_closew_clicked();

private:
    Ui::input *ui;
    StartMenu* sWindow;

    int arraysize;
    bool prior;
};

#endif // INPUT_H
```

Название файла: mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QGraphicsScene>
#include <QGraphicsTextItem>
#include "treap.h"
#include <QTextStream>
#include <QElapsedTimer>
#include <QIntValidator>
#include <QFileDialog>
#include <string>
#include <sstream>
#include "startmenu.h"
#include "input.h"

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

public slots:
    void on_change_clicked();
    void on_inputarray_clicked();

private:
    Ui::MainWindow *ui;
    StartMenu* sWindow;
    input* inWindow;
};
#endif // MAINWINDOW_H
```

Название файла: startmenu.h

```
#ifndef STARTMENU_H
#define STARTMENU_H
```

```
#include <QWidget>
#include <QGraphicsScene>
#include <QGraphicsTextItem>
#include <QtCharts>
#include "treap.h"
#include <QLineSeries>
#include <QPainter>
#include <QTextStream>
#include <QElapsedTimer>
#include <QIntValidator>
#include <QFileDialog>
#include <string>
#include <sstream>

namespace Ui {
class StartMenu;
}

class StartMenu : public QWidget
{
    Q_OBJECT

public:
    explicit StartMenu(QWidget *parent = nullptr);
    ~StartMenu();

    void start();
    void setArrSize(int n);
    void setPrior(bool n);

    void random();
    void ingen(std::string str);
    template<typename T>
    void generate(T* array, size_t s, std::string info, bool prior);

public slots:
    void on_worst_clicked();
    void on_draw_clicked();
    void on_add_clicked();
    void on_deleteb_clicked();

private:
    Ui::StartMenu *ui;
    QGraphicsScene* scene, *chart1_scene, *chart2_scene;
    QLineSeries *series1, *series2;
```

```
    QChart *chart1, *chart2;
    QChartView *chartView1, *chartView2;

    Treap<int>* treap;

    std::string info;
    int* array, *curr_array;
    int arraysize, curr_arraysize, del_arraysize = 0,curr_del = 0,curr_del_last = 0,
time_array[10000], del_array[10000];
    bool prior, newprior;

    void setArray(std::string str);
    void setChart();
    void setDelChart();
};

#endif // STARTMENU_H
```

Название файла: treap.h

```
#ifndef TREAP_H
#define TREAP_H

#include <iostream>
#include <fstream>
#include <QPainter>
#include <QGraphicsScene>
#include <QGraphicsTextItem>
#include <QPen>
#include <math.h>

template <typename T>
class Treap
{
public:
    Treap() {};
    Treap(T value, int R) : value(value), R(R), Left(nullptr), Right(nullptr) {};

    void erase(T val)
    {
        _erase(root, val);
    }

    void insert(T val, int r)
    {
        _insert(root, new Treap<T>(val, r));
```

```
    }

    void print(std::ostream& streamOut)
    {
        _print(root, streamOut);
    }

    void write(std::ofstream &file)
    {
        _write(root, file);
    }

    void drawTree(QGraphicsScene *scene, int size)
    {
        sz = 2*std::log2(size);
        draw(scene, root, 0);
    }

    void info_expand(std::string &str)
    {
        _info_expand(root, str);
    }
private:

    T value;
    int R;

    Treap *Left, *Right;
    Treap* root = nullptr;
    int sz;
    void split(Treap* t, Treap *&left, Treap *&right, T& val)
    {
        if (t == nullptr)
        {
            left = nullptr;
            right = nullptr;
        }
        else if (t->value > val)
        {
            split(t->Left, left, t->Left, val);
            right = t;
        }
        else
        {
            split(t->Right, t->Right, right, val);
```

```
        left = t;
    }
}

void merge(Treap*& t, Treap* left, Treap* right)
{
    if (!left){
        t = right;
        return;
    }
    if (!right){
        t = left;
        return;
    }
    if (left->R >= right->R){
        merge(left->Right, left->Right, right);
        t = left;
    }
    else{
        merge(right->Left, left, right->Left);
        t = right;
    }
}

    void _insert(Treap*& t, Treap* v)
{
    if (!t)
        t = v;
    else if (v->R > t->R)
    {
        split(t, v->Left, v->Right, v->value);
        t = v;
    }
    else
    {
        _insert (v->value < t->value ? t->Left : t->Right, v);
    }
}

void _erase (Treap*& t, T key)
{
    if (!t) return;
    if (t->value == key){
        merge(t, t->Left, t->Right);
    }
```

```
        else if (t->value > key)
            _erase(t->Left, key);
        else
            _erase(t->Right, key);
    }

    void _print(Treap *&t, std::ostream& streamOut)
    {
        if (t == nullptr){
            streamOut << std::endl;
            return;
    }
        _print(t->Left, streamOut);
        streamOut << t->value <<' ';
        _print(t->Right, streamOut);
    }

    void draw(QGraphicsScene *scene, Treap<T> *tr, int index)
    {
        int x = 800;
        if (tr){
            if(index == 0){
                drawL(scene, tr->Left, index+1, x/2);
                drawR(scene, tr->Right, index+1, x/2);
                scene-
>addEllipse(x/2,50*(index+1),40,40,QColor(0,0,0),QColor(255,150,255));
                scene->addText(QString().setNum(tr->value))-
>setPos(x/2+2,50*(index+1));
                scene->addText(QString().setNum(tr->R))-
>setPos(x/2+2,50*(index+1)+10);
            }
        }
    }

    void drawL(QGraphicsScene *scene, Treap<T> *tr, int index, int offset)
    {
        if (tr){
            int x = offset - 8*pow(2,sz-index);
            scene->addLine(x + 12.5,50*(index+1)+12.5,offset+12.5,50*index
+12.5,QPen(Qt::black,3));
            drawL(scene, tr->Left, index+1, x);
            drawR(scene, tr->Right, index+1, x);
            scene-
>addEllipse(x,50*(index+1),40,40,QColor(0,0,0),QColor(255,150,255));
            scene->addText(QString().setNum(tr->value))->setPos(x+2,50*(index+1));
```

```
            scene->addText(QString().setNum(tr->R))->setPos(x+2,50*(index+1)+10);
        }
    }

    void drawR(QGraphicsScene *scene, Treap<T> *tr, int index, int offset)
    {
        if (tr){
            int x = offset + 8*pow(2,sz-index);
            scene->addLine(x + 12.5,50*(index+1)+12.5,offset+12.5,50*index
+12.5,QPen(Qt::black,3));
            drawL(scene, tr->Left, index+1, x);
            drawR(scene, tr->Right, index+1, x);
            scene-
>addEllipse(x,50*(index+1),40,40,QColor(0,0,0),QColor(255,150,255));
            scene->addText(QString().setNum(tr->value))->setPos(x+2,50*(index+1));
            scene->addText(QString().setNum(tr->R))->setPos(x+2,50*(index+1)+10);
        }
    }
    void _info_expand(Treap *t, std::string &str)
    {
        if (t == nullptr){
            return;
    }
        _info_expand(t->Left, str);
        str.append(std::to_string(t->value));
        str.append(" ");
        _info_expand(t->Right, str);
    }
};

#endif // TREAP_H
```

Название файла: input.cpp

```
#include "input.h"
#include "ui_input.h"

input::input(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::input)
{
    ui->setupUi(this);
    QRegExp rx("[0-9 ]*");
    QValidator* validator = new QRegExpValidator(rx, this);
    sWindow = new StartMenu();
    ui->text->setValidator(validator);
```

```
}

input::~input()
{
    delete ui;
}

std::string input::getline()
{
    return qPrintable(ui->text->text());
}

void input::setarrsize(int n)
{
    arraysize = n;
}

void input::setPrior(bool n)
{
    prior = n;
}

void input::on_ok_clicked()
{
    std::string temp = getline();
    sWindow->setArrSize(arraysize);
    sWindow->setPrior(prior);
    sWindow->ingen(temp);
    sWindow->show();
    this->close();
}

void input::on_closew_clicked()
{
    this->close();
}
```

Название файла: main.cpp

```
#include "mainwindow.h"

#include <QApplication>
#include <iostream>
#include <time.h>
```

```
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

Название файла: mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <iostream>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_change_clicked()
{
    sWindow= new StartMenu();
    sWindow->show();
    sWindow->setArrSize(ui->arrsize->value());
    sWindow->setPrior(ui->prior->checkState());
    sWindow->random();
}

void MainWindow::on_inputarray_clicked()
{
    inWindow = new input();
    inWindow->setarrsize(ui->arrsize->value());
    inWindow->setPrior(ui->prior->checkState());
    inWindow->show();
}
```

Название файла: startmenu.cpp

```
#include "startmenu.h"
#include "ui_startmenu.h"
```

```
StartMenu::StartMenu(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::StartMenu)
{
    ui->setupUi(this);
    scene = new QGraphicsScene(0, 0, 800, 800, this);
    chart1_scene = new QGraphicsScene(0, 0, 500, 260, this);
    chart2_scene = new QGraphicsScene(0, 0, 500, 260, this);
    ui->drawview->setScene(scene);
    scene->setSceneRect(scene->itemsBoundingRect());
    chart1 = new QChart();
    chart2 = new QChart();
}

StartMenu::~StartMenu()
{
    delete ui;
}

void StartMenu::start()
{
    ui->textBrowser->clear();
    scene->clear();
    ui->textBrowser->setText("Treap visualization\n");
}

void StartMenu::setArrSize(int n)
{
    arraysize = n;
}

void StartMenu::setPrior(bool n)
{
    prior = n;
}

void StartMenu::random()
{
    start();
    info = "Randomly generated array:\nO(nlogn)\nInserting the keys one at a time\n";
    array = new int[arraysize];
    for (int i = 0; i < arraysize; i++ ) {
        array[i] = rand();
    }
}
```

```
void StartMenu::ingen(std::string str)
{
   start();
   info = "Generate treap from input:\nO(nlogn)\nArray:\n";
   std::stringstream ss;
   int tempcnt = 0;
   array = new int[arraysize];
   ss << str;
   for (int i = 0; !ss.eof() && (i < arraysize); i++){
      tempcnt++;
      ss >> array[i];
   }
   setArrSize(tempcnt);
}

void StartMenu::on_worst_clicked()
{
   start();
   info = "Worst Case treap:\nO(n)\nSorted array without rand() or prev.rand <
new.rand\n";
   newprior = false;
   int wc[arraysize];
   for (int i = 0; i < arraysize; i++)
      wc[i] = i+1;
   generate<int>(wc, arraysize, info, false);
}

void StartMenu::on_draw_clicked()
{
   start();
   info = "Treap:\n";
   newprior = prior;
   generate<int>(array, arraysize, info, prior);
}

void StartMenu::on_add_clicked()
{
   int temp = ui->spinBox->value();
   QElapsedTimer timer;
   std::string temparray;
   if (newprior){
      timer.start();
      treap->insert(temp, rand());
   }
```

```
    else
    {
        timer.start();
        treap->insert(temp, rand()%1);
    }
    int tmr = timer.nsecsElapsed();
    time_array[curr_arraysize++] = tmr;
    treap->info_expand(temparray);
    setArray(temparray);
    setChart();
    scene->clear();
    treap->drawTree(scene, curr_arraysize);
}

void StartMenu::on_deleteb_clicked()
{
    QElapsedTimer timer;
    std::string temparray;
    bool exist = false;
    if (curr_arraysize == 0) return;
    switch (ui->comboBox->currentIndex()) {
    case 0:
        timer.start();
        treap->erase(curr_array[0]);
        del_array[del_arraysize++] = timer.nsecsElapsed();
        curr_arraysize--;
        break;
    case 1:
        for (int i = 0; i < curr_arraysize; i++){
            if (ui->delbox->value() == curr_array[i])
                exist = true;
        }
        if (!exist) return;
        timer.start();
        treap->erase(ui->delbox->value());
        del_array[del_arraysize++] = timer.nsecsElapsed();
        curr_arraysize--;
        break;
    case 2:
        timer.start();
        treap->erase(curr_array[curr_arraysize-1]);
        del_array[del_arraysize++] = timer.nsecsElapsed();
        curr_arraysize--;
        break;
    }
```

```
    treap->info_expand(temparray);
    setArray(temparray);
    setChart();
    setDelChart();
    scene->clear();
    treap->drawTree(scene, curr_arraysize);
}

void StartMenu::setArray(std::string str)
{
    std::stringstream ss;
    curr_array = new int[curr_arraysize];
    ss << str;
    for (int i = 0; i < curr_arraysize; i++)
        ss >> curr_array[i];
}

void StartMenu::setChart()
{
    chart1->legend()->hide();
    chart1->removeAllSeries();
    series1 = new QLineSeries();
    for (int i = 0; i < curr_arraysize; i++)
    {
        *series1 << QPoint(i+1, time_array[i]);
    }
    chart1->addSeries(series1);
    chart1->createDefaultAxes();
    chart1->setTitle("Input Chart");
    chartView1 = new QChartView(chart1);
    chartView1->setRenderHint(QPainter::Antialiasing);
    chartView1->resize(500, 260);
    chart1_scene->addWidget(chartView1);
    ui->chart_1->setScene(chart1_scene);
}

void StartMenu::setDelChart()
{
    chart2->legend()->hide();
    chart2->removeAllSeries();
    series2 = new QLineSeries();
    for (int i = 0; i < del_arraysize; i++)
    {
        *series2 << QPoint(i+1, del_array[i]);
    }
```

```
    chart2->addSeries(series2);
    chart2->createDefaultAxes();
    chart2->setTitle("Delete Chart");
    chartView2 = new QChartView(chart2);
    chartView2->setRenderHint(QPainter::Antialiasing);
    chartView2->resize(500, 260);
    chart2_scene->addWidget(chartView2);
    ui->chart_2->setScene(chart2_scene);
}

template<typename T>
void StartMenu::generate(T *array, size_t s, std::string info, bool prior)
{
    QElapsedTimer timer;
    std::string temparray;
    int temp = 0, tmr = 0;
    treap = new Treap<int>();
    for (size_t t = 0; t < s; t++)
        {
            if (prior){
                timer.start();
                treap->insert(array[t], rand());
                temp = timer.nsecsElapsed();
            }
            else
            {
                timer.start();
                treap->insert(array[t], rand()%1);
                temp = timer.nsecsElapsed();
            }
            time_array[t] = temp;
            tmr += temp;
        }
    curr_arraysize = arraysize;
    treap->info_expand(info);
    treap->info_expand(temparray);
    setArray(temparray);
    info.append("\nTime spend:");
    info.append(std::to_string(tmr));
    info.append(" nanoseconds");
    ui->textBrowser->setText(QString::fromStdString(info));
    treap->drawTree(scene, arraysize);
    curr_del_last = arraysize;
    setChart();
    chart2->removeAllSeries();
```

```
    del_arraysize = 0;
    curr_del = 0;
}
```