

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет Информатика и вычислительная техника

Кафедра Кибербезопасность информационных систем

**ПРАКТИЧЕСКАЯ РАБОТА № 6**

на тему «Деревья: двоичное дерево поиска, обходы дерева, дерево отрезков, декартово дерево»

Выполнила обучающаяся гр. ВКБ32

Дедиченко Дарья Валентиновна

Проверил:

Доцент, Савельев Василий Александрович

Ростов-на-Дону

2021

Е. Декартово дерево

#include <iostream>

#include <fstream>

#include <time.h>

#include <stdlib.h>

using namespace std;

ifstream in("input.txt");

ofstream out("output.txt");

class vertex{

    public:

    int x, y, label;

    struct vertex \*parent, \*left, \*right;

    vertex() {}

    void set(int xi, int yi, int labeli){

        x = xi;

        y = -yi;

        label = labeli;

        parent = NULL;

        left = NULL;

        right = NULL;

    }

    vertex& operator=(const vertex& a){

        x = a.x;

        y = a.y;

        label = a.label;

        parent = parent;

        left = left;

        right = right;

        return \*this;

    }

    friend void swap (vertex& a, vertex& b);

};

void swap (vertex& a, vertex& b){

    vertex shift;

    shift = a;

    a = b;

    b = shift;

}

typedef vertex \*pvertex;

class decTree{

    private:

    int treeSize;

    vertex \*vertexs;

    pvertex \*ordered;

    pvertex root;

    int vertexCount;

    void \_sortIns(pvertex \*pvertexs , int left, int right){

        pvertex shift;

        for (int i = left + 1; i < right; i++){

            int j = i - 1;

            shift = pvertexs[i];

            for (int k = i-1; ((k >= left) && (pvertexs[j]->x > shift->x)) ; k--, j--){

                pvertexs[k + 1] = pvertexs[k];

            }

            pvertexs[j + 1] = shift;

        }

    }

    void \_sortQuick(pvertex \*pvertexs, int left, int right){

        srand(time(NULL));

        int l = left;

        int r = right-1;

        pvertex p;

        p = pvertexs[l + rand() % (right - left)];

        do{

            while (pvertexs[l]->x < p->x) {

                l++;

            }

            while (pvertexs[r]->x > p->x) {

                r--;

            }

            if (l <= r){

                swap(pvertexs[l++], pvertexs[r--]);

            }

        } while (l <= r);

        if (r - left > 16){

            \_sortQuick(pvertexs, left, r + 1);

        } else if (r > left){

            \_sortIns(pvertexs, left, r + 1);

        }

        if (right - l > 16){

            \_sortQuick(pvertexs, l, right);

        } else if (l < right - 1){

            \_sortIns(pvertexs, l , right);

        }

        ////

    }

    void \_makeTree(){

        pvertex last = NULL;

        root = ordered[0];

        last = root;

        for (int i = 1; i < treeSize; i++){

            while (last->parent != NULL && ordered[i]->y > last->y){

                last = last->parent;

            }

            if (ordered[i]->y <= last->y){

                ordered[i]->left = last->right;

                ordered[i]->parent = last;

                if (last->right){

                    last->right->parent = ordered[i];

                }

                last->right = ordered[i];

            } else {

                ordered[i]->left = last;

                last->parent = ordered[i];

                root = ordered[i];

            }

            last = ordered[i];

        }

    }

    public:

    void vertexAdd(int x, int y){

        vertexs[vertexCount].set(x, y, vertexCount + 1);

        ordered[vertexCount] = vertexs + vertexCount;

        vertexCount++;

        \_sortQuick(ordered, 0, treeSize);

        \_makeTree();

    }

    decTree(int size):treeSize(size){

        vertexCount = 0;

        vertexs = new vertex[treeSize];

        ordered = new pvertex[treeSize];

    }

    decTree(){

        int a , b;

        vertexCount = 0;

        in >> treeSize;

        vertexs = new vertex[treeSize];

        ordered = new pvertex[treeSize];

        for (int i = 0; i < treeSize; i++ ){

            in >> a >> b;

            vertexs[vertexCount].set(a, b, vertexCount + 1);

            ordered[vertexCount] = vertexs + vertexCount;

            vertexCount++;

        }

        \_sortQuick(ordered, 0, treeSize);

        \_makeTree();

    }

    void resultsDisplay(){

        int parent, left, right;

        out << "YES" << endl;

        for (int i = 0; i < treeSize; i++){

            if (vertexs[i].parent) {

                parent = vertexs[i].parent->label;

            } else {

                parent = 0;

            }

            if (vertexs[i].left){

                left = vertexs[i].left->label;

            }  else {

                left = 0;

            }

            if (vertexs[i].right) {

                right = vertexs[i].right->label;

            } else {

                right = 0;

            }

            out << parent << " " << left << " " << right << endl;

        }

    }

};

int main(void){

    decTree \*mytree = new decTree();

    mytree->resultsDisplay();

    return 0;

}

