

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»

**Лабораторная работа №6**

Выполнил:

Студент группы ВКБ31

Карпов Д.И.

Проверил:

доц., к.т.н. Савельев В.А.

Ростов-на-Дону,

2021г.

**Лабораторная работа №6**

Вам даны пары чисел (ai,bi)(ai,bi), Вам необходимо построить декартово дерево, такое что ii-ая вершина имеет ключи (ai,bi)(ai,bi), вершины с ключом aiai образуют бинарное дерево поиска, а вершины с ключом bibi образуют кучу.

Входные данные

В первой строке записано число NN — количество пар. Далее следует NN (1≤N≤500001≤N≤50000) пар (ai,bi)(ai,bi). Для всех пар |ai|,|bi|≤30000|ai|,|bi|≤30000. ai≠ajai≠aj и bi≠bjbi≠bj для всех i≠ji≠j.

Выходные данные

Если декартово дерево с таким набором ключей построить возможно, выведите в первой строке YES, в противном случае выведите NO. В случае ответа YES, выведите NN строк, каждая из которых должна описывать вершину. Описание вершины состоит из трёх чисел: номер предка, номер левого сына и номер правого сына. Если у вершины отсутствует предок или какой-либо из сыновей, то выводите на его месте число 0.

Если подходящих деревьев несколько, выведите любое.

Примеры

|  |  |
| --- | --- |
| входные данные  7  5 4  2 2  3 9  0 5  1 3  6 6  4 11 | выходные данные  YES  2 3 6  0 5 1  1 0 7  5 0 0  2 4 0  1 0 0  3 0 0 |

using namespace std;

int father = 0;

struct treap {

int x, y, num; //координаты и порядковый номер точки в дереве

treap \*left, \* right; //левое, правое, предковое деревья

treap(int x, int y, treap\* l, treap\* r, int n) { //конструктор

this->x = x;

this->y = y;

this->left = l;

this->right = r;

this->num = n;

}

struct treap\* merge(treap\* l, treap\* r) { //слияние двух деревьев

if (l == nullptr) return r;

if (r == nullptr) return l;

if (l->y < r->y)

return new treap(l->x, l->y, l->left, merge(l->right, r), l->num);

else

return new treap(r->x, r->y, merge(l, r->left), r->right, r->num);

}

void split(int x, treap \*&l, treap \*&r) {

treap\* newtree = nullptr;

if (this->x <= x) {

if (this->right == NULL) r = nullptr;

else {

newtree = new treap(right->x, right->y, nullptr, nullptr, right->num);

right->split(x, newtree, r);

}

l = new treap(this->x, this->y, left, newtree, this->num);

}

else

{

if (this->left == nullptr) l = nullptr;

else {

newtree = new treap(left->x, left->y, nullptr, nullptr, left->num);

left->split(x, l, newtree);

}

r = new treap(this->x, this->y, newtree, right, this->num);

}

}

struct treap\* add(int x, int y, int n) {

treap\* l = new treap(-1, -1, nullptr, nullptr, 0), \* r = new treap(-1, -1, nullptr, nullptr, 0);

split(x, l, r);

treap\* m = new treap(x, y, nullptr, nullptr, n);

if (l == nullptr || l->x == -1) l = nullptr;

if (r == nullptr || r->x == -1) r = nullptr;

return merge(merge(l, m), r); //end - конечное дерево

}

treap\* find(int x) {

if (this == NULL)

return NULL;

else if (x < this->x) {

father = this->num;

return this->left->find(x);

}

else if (x > this->x) {

father = this->num;

return this->right->find(x);

}

else {

return this;

}

}

};

int main() {

int n, key, pr;

cin >> n >> key >> pr;

vector<int> val;

val.push\_back(key);

treap\* tree = new treap(key, pr, nullptr, nullptr, 1);

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

cin >> key >> pr;

tree = tree->add(key, pr, i + 1);

val.push\_back(key);

}

cout << "YES" << endl;

treap\* aim = NULL;

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

aim = tree->find(val[i]);

cout << father << " " << ((aim->left != NULL) ? aim->left->num : 0) << " " << ((aim->right != NULL) ? aim->right->num : 0) << endl;

father = 0;

}

return 0;}

