МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ УПРАВЛІННЯ

ЗВІТ

З ЛАБОРАТОРНОЇ РОБОТИ №3

ЗА ТЕМОЮ “БАЗОВІ СТРУКТУРИ ДАНИХ: ЧЕРВОНОЧОРНІ ДЕРЕВА”

Виконав студент

групи КН-221д

Габорець

Перевірив

Солонська С.В.

Харків 2021

ЛАБОРАТОРНА РОБОТА №3

Тема: БАЗОВІ СТРУКТУРИ ДАНИХ: ЧЕРВОНОЧОРНІ ДЕРЕВА.

Мета роботи: познайомитися з червоно-чорними деревами та отримати навички програмування алгоритмів, що їх обробляють.

Завдання

Розробити програму, яка читає з клавіатури числа N, M (1 < N, M < 256); послідовність N ключів (цілих, дійсних чисел або рядків (до 255 символів) в залежності від варіанту завдання); послідовність M ключів. Програма зберігає першу послідовність до червоно-чорного дерева.

Кожного разу, коли до дерева додається новий елемент, потрібно вивести статистику (згідно варіанту завдання).

1.Мінімальний елемент та його колір;

2.Максимальний елемент та його колір.

Після побудови дерева для кожного елементу x другої послідовності потрібно вивести результати наступних операцій над деревом (згідно варіанту завдання).

1 Чи є елемент у дереві та його колір.

2 Successor(x) та його колір.

3 Predecessor(x) та його колір.

Використовувати готові реалізації структур даних (наприклад, STL) заборонено, але можна використати реалізацію рядків (наприклад, std::string у C++).

Варіант завдання:

(2.1.)

2.Максимальний елемент та його колір.

1 Чи є елемент у дереві та його колір.



Код:

#include<iostream>

//Варіант(4): 2. Максимальний елемент та його колір 1. Чи є елемент у дереві та його колір

enum COLOR { RED = 0, BLACK = 1};

struct node

{

node(int const key) : key(key) {}

COLOR color = RED;

int key;//data

node\* left = nullptr, \* right = nullptr, \* parent = nullptr;

};

// дерево

node\* root = nullptr;

// поиск деда

node\* grandparent(node\* temp)

{

node\* ret = nullptr;

if (temp && temp->parent)

{

ret = temp->parent->parent;

}

return ret;

}

// поиск дяди

node\* uncle(node\* temp)

{

node\* ret = nullptr;

if (node const\* ded = grandparent(temp))

{

ret = (ded->left == temp->parent ? ded->right : ded->left);

}

return ret;

}

void rotate\_left(node\* rhs)

{

node\* pivot = rhs->right;

pivot->parent = rhs->parent;

if (rhs->parent != nullptr)

{

if (rhs->parent->left == rhs)

{

rhs->parent->left = pivot;

}

else

{

rhs->parent->right = pivot;

}

}

else

{

root = pivot;

}

rhs->right = pivot->left;

if (pivot->left != nullptr)

{

pivot->left->parent = rhs;

}

rhs->parent = pivot;

pivot->left = rhs;

}

// правый поворот

void rotate\_right(node\* rhs)

{

node\* pivot = rhs->left;

pivot->parent = rhs->parent;

if (rhs->parent != nullptr)

{

if (rhs->parent->left == rhs)

{

rhs->parent->left = pivot;

}

else

{

rhs->parent->right = pivot;

}

}

else

{

root = pivot;

}

rhs->left = pivot->right;

if (pivot->right != nullptr)

{

pivot->right->parent = rhs;

}

rhs->parent = pivot;

pivot->right = rhs;

}

//Functions

void push\_case1(node\* temp);

void push\_case2(node\* temp);

void push\_case3(node\* temp);

void push\_case4(node\* temp);

void push\_case5(node\* n);

void push(node\* &root, int const key)//Вставка

{

node\*\* temp = &root;

node\* parent = nullptr;

while (\*temp)

{

parent = \*temp;

if (key < (\*temp)->key)

{

temp = &(\*temp)->left;

}

else if (key > (\*temp)->key)

{

temp = &(\*temp)->right;

}

else

{

break;

}

}

if (nullptr == \*temp)

{

\*temp = new node(key);

(\*temp)->parent = parent;

push\_case1(\*temp);

}

}

void push\_case1(node\* temp)

{

if (temp->parent == nullptr)

{

temp->color = BLACK;

}

else

{

push\_case2(temp);

}

}

void push\_case2(node\* temp)

{

if (temp->parent->color == BLACK)

{

return; /\* С деревом ещё всё ок \*/

}

else

{

push\_case3(temp);

}

}

void push\_case3(node\* temp)

{

node\* u = uncle(temp), \* g = nullptr;

if ((u != nullptr) && (u->color == RED))

{

// && (n->parent->color == RED) Второе условие проверяется в insert\_case2, то есть родитель уже является красным.

temp->parent->color = BLACK;

u->color = BLACK;

g = grandparent(temp);

g->color = RED;

push\_case1(g);

}

else

{

push\_case4(temp);

}

}

void push\_case4(node\* temp)

{

node\* g = grandparent(temp);

if ((temp == temp->parent->right) && (temp->parent == g->left))

{

rotate\_left(temp->parent);

temp = temp->left;

}

else if ((temp == temp->parent->left) && (temp->parent == g->right))

{

rotate\_right(temp->parent);

temp = temp->right;

}

push\_case5(temp);

}

void push\_case5(node\* n)

{

node\* g = grandparent(n);

n->parent->color = BLACK;

g->color = RED;

if ((n == n->parent->left) && (n->parent == g->left))

{

rotate\_right(g);

}

else

{ /\* (n == n->parent->right) && (n->parent == g->right) \*/

rotate\_left(g);

}

}

void print(node\* root, int indent = 0)

{

if (root)

{

print(root->right, indent + 1);

for (int i = 0; i < indent; ++i)

{

std::cout << ".";

}

std::cout << root->key << std::endl;

print(root->left, indent + 1);

}

}

void find(int const data)

{

node\* ret = root;

while (ret)

{

if (data < ret->key)

{

ret = ret->left;

}

else if (data > ret->key)

{

ret = ret->right;

}

else if (data == ret->key)

{

std::cout<<"There IS such node "<<data<<" in tree\n";

std::cout <<"COLOR (0-RED,1-BLACK) : "<< ret->color << "\n";

break;

}

}

//there is not such NODE in tree

}

void find\_max()

{

int max\_key = root->key;

node\* ret = root;

node\* color\_temp = ret;

while (ret)

{

if (max\_key > ret->key)

{

ret = ret->right;

}

else if (max\_key <= ret->key)

{

max\_key = ret->key;

color\_temp = ret;

ret = ret->right;

}

}

std::cout << "Tree MAX : " << max\_key << "\n";

std::cout << "COLOR (0-RED,1-BLACK) : " << color\_temp->color << "\n";

}

int main()

{

unsigned int N = 0, M = 0;

while (N <= 1 || N >= 256 || M <= 1 || M >= 256)

{

std::cout << "Input 1 < N < 256:";

std::cin >> N;

std::cout << "Input 1 < M < 256:";

std::cin >> M;

}

std::cout << "Input key(N):\n";

int key;

for (unsigned int i = 0; i < N; i++)// N keys

{

std::cin >> key;//Вводишь ключ

push(root, key);

find\_max();

//Statistic 1. Максимальний елемент та його колір

}

std::cout << "\n\n";

std::cout << "Input key(M):\n";

for (unsigned int i = 0; i < M; i++)// N keys

{

std::cin >> key;//Вводишь ключ

push(root, key);

//Statistic 2. 2.1 Чи є елемент у дереві та його колір

}

/\* print(root);\*/

std::cout << "Loocking for nodes with (5,12,1) keys...\n\n";

find(5);//Если в консоли ничего нету,то такого узла в дереве нет!

find(12);

find(1);;

return 0;

}

Висновок: виконавши лабораторну роботу №3, я познайомився з червоно-чорними деревами та отримав навички програмування алгоритмів, що їх обробляють.