Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут ім. Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації та управління

ЗВІТ

про виконання лабораторного практикуму №7(1)

Виконав:

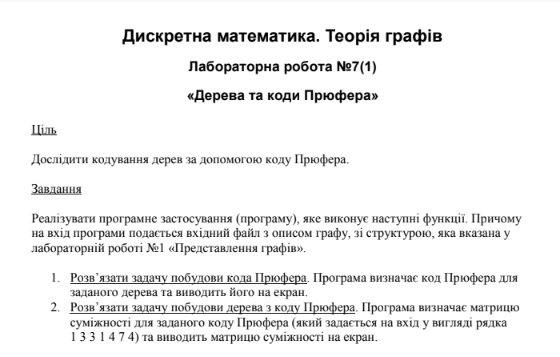
студент 1- го курсу ФІОТ

групи *ІП-91*

*Кінчур Вадим Вікторович*

Київ 2020

1. Умова лабораторної роботи



1. Програмний код (С++):

#include <iostream>

#include <fstream>

#include <list>

#include <stack>

#include <string>

#include <vector>

#include <iomanip>

#include <algorithm>

#include <regex>

using namespace std;

void get\_input(const char\*, list<int>\*&, int&);

int\* get\_parents(list<int>\*, int);

int\* get\_prufer\_code(list<int>\*, int);

int get\_pointer(int\*, int, int);

int\* get\_degrees(list<int>\*, int);

void print\_array(int\*, int);

string read\_code(const char\*);

vector<int> convert\_to\_int(string);

int\* fill\_array(int, int);

int\*\* fill\_matrix(int, int);

int\* degree\_from\_code(vector<int>, int);

int\*\* decode\_tree(vector<int>);

void print\_matrix(int\*\*, int);

void show\_menu();

void work\_with\_answer(int);

void process\_menu();

void delete\_matrix(int\*\*, int);

void print\_matrix(int\*\*, int);

int main() {

process\_menu();

}

void get\_input(const char\* file\_name, list<int>\*& adj\_l, int& v\_num) {

ifstream file\_handler(file\_name);

int e\_num, start, end;

file\_handler >> v\_num >> e\_num;

adj\_l = new list<int>[v\_num];

for (auto i = 0; i < e\_num; ++i) {

file\_handler >> start >> end;

adj\_l[start - 1].push\_back(end - 1);

adj\_l[end - 1].push\_back(start - 1);

}

file\_handler.close();

}

int\* get\_parents(list<int>\* adj\_l, const int v\_num) {

int\* parent = new int[v\_num];

stack<int> stack;

int v;

parent[v\_num - 1] = -1;

stack.push(v\_num - 1);

while (!stack.empty()) {

v = stack.top();

stack.pop();

for (auto to : adj\_l[v]) {

if (to != parent[v]) {

parent[to] = v;

stack.push(to);

}

}

}

return parent;

}

int\* get\_degrees(list<int>\* adj\_l, int v\_num) {

auto\* degree = new int[v\_num];

for (auto i = 0; i < v\_num; ++i) {

degree[i] = adj\_l[i].size();

}

return degree;

}

int get\_pointer(int\* degree, const int v\_num, const int start\_v) {

for (auto i = start\_v; i < v\_num; ++i)

if (degree[i] == 1)

return i;

}

int\* get\_prufer\_code(list<int>\* adj\_l , const int v\_num)

{

auto\* prufer\_code = new int[v\_num - 2];

auto\* degree = get\_degrees(adj\_l, v\_num);

auto\* parent = get\_parents(adj\_l, v\_num);

auto ptr = get\_pointer(degree, v\_num, 0);

auto leaf = ptr;

for (auto i = 0; i < v\_num - 2; ++i) {

int next = parent[leaf];

prufer\_code[i] = next + 1;

--degree[next];

if (degree[next] == 1 && next < ptr) {

leaf = next;

}

else {

ptr = get\_pointer(degree, v\_num, ++ptr);

leaf = ptr;

}

}

delete[] degree; delete[] parent;

return prufer\_code;

}

void print\_array(int\* arr, const int size) {

for (auto i = 0; i < size; ++i)

cout << arr[i] << " ";

cout << '\n';

}

vector<int> convert\_to\_int(string str) {

vector<int> arr\_int;

string temp;

for (auto i = 0; i < str.length(); ++i) {

if (str[i] == ' ') {

temp = regex\_replace(temp, regex("^ +"), "");

arr\_int.push\_back(stoi(temp) - 1);

temp = "";

}

else temp += str[i];

}

arr\_int.push\_back(stoi(temp) - 1);

return arr\_int;

}

int\* fill\_array(const int size, const int filler) {

auto\* arr = new int[size];

for (auto i = 0; i < size; ++i)

arr[i] = filler;

return arr;

}

int\*\* fill\_matrix(const int size, const int filler) {

int\*\* matrix = new int\* [size];

for (auto i = 0; i < size; ++i)

matrix[i] = new int[size];

for (auto i = 0; i < size; ++i)

for (auto j = 0; j < size; ++j)

matrix[i][j] = filler;

return matrix;

}

string read\_code(const char\* file\_name) {

ifstream file\_handler(file\_name);

string code;

getline(file\_handler, code);

file\_handler.close();

return code;

}

int\* degree\_from\_code(vector<int> prufer\_code, int v\_num) {

auto\* degree = fill\_array(v\_num, 1);

for (auto i = 0; i < v\_num - 2; ++i)

++degree[prufer\_code[i]];

return degree;

}

int\*\* decode\_tree(vector<int> prufer\_code) {

auto const v\_num = prufer\_code.size() + 2;

auto\* degree = degree\_from\_code(prufer\_code, v\_num);

int\*\* adj\_m = fill\_matrix(v\_num, 0);

auto ptr = get\_pointer(degree, v\_num, 0);

auto leaf = ptr;

for (auto i = 0; i < v\_num - 2; ++i) {

int v = prufer\_code[i];

adj\_m[leaf][v] = 1;

adj\_m[v][leaf] = 1;

--degree[leaf];

if (--degree[v] == 1 && v < ptr)

leaf = v;

else {

ptr = get\_pointer(degree, v\_num, ++ptr);

leaf = ptr;

}

}

for (auto i = 0; i < v\_num - 1; ++i) {

if (degree[i] == 1)

adj\_m[i][v\_num - 1] = 1;

}

delete[] degree;

return adj\_m;

}

void print\_matrix(int\*\* matrix, int size) {

for (auto i = 0; i < size; ++i) {

for (auto j = 0; j < size; ++j)

cout << setw(3) << matrix[i][j];

cout << '\n';

}

}

void show\_menu() {

cout << "Choose one of the following : \n"

<< "1 - find prufer code from tree\n"

<< "2 - decode tree from prufer code\n";

}

void work\_with\_answer(int answer) {

if (answer == 1) {

list<int>\* adj\_l = nullptr;

int v\_num;

get\_input("input.txt", adj\_l, v\_num);

auto\* prufer\_code = get\_prufer\_code(adj\_l, v\_num);

cout << "Prufer code : \n";

print\_array(prufer\_code, v\_num - 2);

delete[] prufer\_code;

}

else {

auto code = read\_code("input.txt");

auto int\_code = convert\_to\_int(code);

auto const v\_num = int\_code.size() + 2;

auto\*\* adj\_m = decode\_tree(int\_code);

cout << "Decoded adjancency matrix : \n";

print\_matrix(adj\_m, v\_num);

delete\_matrix(adj\_m, v\_num);

}

}

void process\_menu() {

int answer = -1;

show\_menu();

while (answer < 1 || answer > 2) {

cout << "Type your choice : ";

cin >> answer;

cout << '\n';

}

work\_with\_answer(answer);

}

void delete\_matrix(int\*\* matrix, int row\_size) {

for (auto i = 0; i < row\_size; ++i)

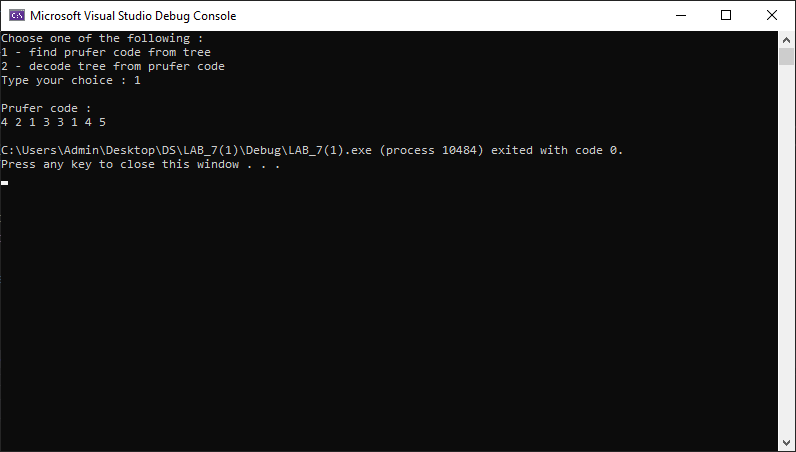
delete[] matrix[i];

delete[] matrix;

}

1. Результат виконання

**Вхідні дані:**

10 9

1 2

1 3

1 4

3 8

3 9

4 5

4 6

2 7

5 10

**Вхідні дані:**

4 2 1 3 3 1 4 5

