#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct node {

string name;

node \*next;

node \*down;

int weight;

bool visit;

node()

{

visit = false;

next = NULL;

down = NULL;

weight = 0;

}

};

class AdjList {

node \*current, \*last;

node \*chead, \*clast, \*ccurrent;

node \*p3, \*p4;

int node\_count, c1 = 0, cnodes = 0;

float density, density\_cluster = 0;

string str1;

public:

node \*head;

AdjList() {

head = NULL;

last = NULL;

current = NULL;

call();

}

void call() {

cout << "Enter density :";

cin >> density;

}

void insertn(string m, string n) {

bool check = 1,check1=1;

node \*ptr = new node;

ptr->name = n;

if (head == NULL) {

node \*p = new node;

p->name = m;

head = p;

last = p;

p->next = ptr;

}

else {

current = head;

while (current != NULL) {

if (current->name == m) {

check = 0;

ccurrent = current;

}

if (current->name == n) {

check1 = 0;

}

current = current->down;

}

if (check1 != 0) {

node \*p = new node;

p->name = n;

last->down = p;

last = p;

}

if (check!=0) {

node \*p = new node;

p->name = m;

last->down = p;

p->next = ptr;

last = p;

}

else {

while (ccurrent->next!=NULL) {

ccurrent = ccurrent->next;

}

ccurrent->next= ptr;

}

}

}

void wset() {

current = head;

node \*ptr;

while (current != NULL) {

ptr = current;

while (ptr->next != NULL&&ptr->next->name!=current->name) {

//cout << "Assinging weight for row " << current->name << " and node :" << ptr->name <<endl;

if(current->name!=ptr->next->name)

ptr->next->weight = assing\_weight(current->name, ptr->next->name);

ptr = ptr->next;

}

current = current->down;

}

}

int assing\_weight(string A, string B) {

int count = 0;

node \*p = search(A), \*p1 = search(B), \*p2 = p1;

for (p = search(A); p->next != NULL; p = p->next) {

//cout << p->next->name << ": " << p->name << "\n";

for (p2 = p1; p2->next != NULL; p2 = p2->next) {

//cout << p2->next->name << ": " << p2->name << "\n";

if (p->next->name == p2->next->name) {

count++;

//cout << count;

}

}

}

//cout << "its out\n";

return count;

}

float density\_value() {

if (chead == NULL||chead->next==NULL) {

return 1;

}

else {

float d, nc = 0, edg = 0;

node \*pta = chead;

while (pta != NULL) {

nc++;

pta = pta->next;

}

edg = nc - 1;

d = (edg \* 2) / (nc\*(nc - 1));

cnodes = nc;

return d;

}

}

bool visited\_all() {

current = head;

while (current != NULL) {

if (!current->visit)

return 0;

current = current->down;

}

return 1;

}

void cluster\_start(node \*clust) {

clust->visit = true;

clust = search(clust->name);

clust->visit = true;

density\_cluster = density\_value();

cout <<"Density = "<< density\_cluster << " for " << clust->name << "\n";

if (chead == NULL) {

make\_cluster(clust);

cout << clust->name << "<-started with ";

if (clust->next != NULL) {

greater\_node(clust);

cout << clust->name << " : greater node \n";

cluster\_start(clust);

}

else {

cout << "Alone boy-> " << clust->name << " ";

print\_cluster();

if (!visited\_all())

cluster\_start(search(current->name));

}

}

else if (density\_cluster > density || cnodes<2) {

cout << "Round 2 : \n";

node \*ptr = clust;

make\_cluster(clust);

if (clust->next != NULL) {

cout << clust->name;

greater\_node(clust);

cout << " has '"<<clust->name << "' for its greater node \n";

if(clust->name!=ptr->name)

cluster\_start(clust);

else {

print\_cluster();

if (!visited\_all()) {

cluster\_start(current);

}

}

}

else {

print\_cluster();

if (!visited\_all())

cluster\_start(current);

}

}

else if (!visited\_all() || density\_cluster < density) {

if (clust->visit == false) {

make\_cluster(clust);

}

print\_cluster();

visited\_all();

cluster\_start(current);

}

else {

print\_cluster();

cout << "is this over?";

}

}

void greater\_node(node \*&ptr) {

node\* p=ptr;

current = ptr;

while (current->next != NULL) {

if (ptr->weight < current->weight&&!current->visit) {

ptr = current;

}

current = current->next;

}

if (p->name == ptr->name&&ptr->next!=NULL) {

ptr = ptr->next;

}

}

void make\_cluster(node \* &inside) {

node \*ptr = new node;

ptr->name = inside->name;

inside->visit = true;

current = search(inside->name);

current->visit = true;

if (chead == NULL) {

chead = ptr;

clast = ptr;

}

else {

clast->next = ptr;

clast = ptr;

}

}

node \* search(string A) {

current = head;

while (current->name != A) {

current = current->down;

}

return current;

}

void print() {

current = head;

node \*p;

while (current != NULL) {

p = current;

while (p != NULL) {

cout << p->name << " " << p->weight << " ";

p = p->next;

}

cout << endl<<endl;

current = current->down;

}

}

void writenode() {

current = head;

node \*p;

ofstream myfile("output.txt");

if (myfile.is\_open())

{

while (current != NULL) {

p = current;

while (p != NULL) {

myfile << "->" << p->name << p->weight;

p = p->next;

}

myfile << " \n";

current = current->down;

}

myfile.close();

}

else cout << "Unable to open file";

}

void print\_cluster() {

current = chead;

c1++;

ofstream myfile("output1.txt", ios::app);

if (myfile.is\_open())

{

myfile << "\n";

while (current != NULL) {

myfile << "->" << current->name;

current = current->next;

}

myfile << "\n Cluster :" << c1;

myfile.close();

}

chead = NULL;

density\_cluster = 0;

}

};

int main()

{

AdjList a;

string line, x, y;

//ifstream myfile("nodes.txt");

ifstream myfile("nodes1.txt");

if (myfile.is\_open())

{

while (!myfile.eof())

{

myfile >> x;

myfile >> y;

if(x != y)

a.insertn(x, y);

}

myfile.close();

}

else cout << "Unable to open file";

cout << endl << "now what?\n";

a.wset();

cout << endl << "lets start printing?\n";

a.print();

a.writenode();

cout << "Go Cluster it : \n";

a.cluster\_start(a.head);

system("pause");

}

