Міністерство освіти і науки України

Черкаський державний технологічний університет

Кафедра програмного забезпечення автоматизованих систем

Звіт

про виконання лабораторної роботи №6

з дисципліни «Проектний практикум»

|  |  |
| --- | --- |
| Перевірив:  Асистент кафедри ПЗАС  Півень О.Б. | Виконав:  студент 3-го курсу  групи ПЗ-1644  Близнюк А. О. |

Черкаси 2017

**Тема роботи:** Метод рельєфів.

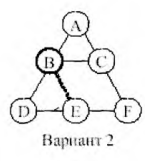
**Мета роботи**: Ознайомлення з методом рельєфів, що дозволяє вибрати шлях між двома вузлами мережі..

**Теоретичні відомості**

Метод рельефов относится к групповым распределенным методам динамического управления. Критерием выбора пути в этом случае является минимальная длина пути, выраженная числом транзитных участков. Возможно использование других критери ев, в частности времени задержки в установлении связи. Однако для определенности далее при описании метода рельефов в ка честве критерия будет выбрано число транзитных участков.   
 На сети связи, где динамическое управление осуществляется методом рельефов, должны выполняться следующие операции: формирование рельефа и его коррекция. Первая из этих опера ций выполняется в начальный момент (в момент пуска сети), а также при развитии сети, т. е. вводе в действие новых УК. Вто рая операция осуществляется периодически в процессе функцио нирования сети или в моменты возникновения повреждений или перегрузок на сети. Рассмотрим обе эти операции отдельно.

**Завдання:**

Реалізувати метод рельєфів.



#include <iostream>

#include <stdlib.h>

#include <time.h>

#include <iomanip>

#include <cmath>

#include <unistd.h>

using namespace std;

const int n = 7;

int GraphMap[n][n] = {{0,0,0,0,0,0,0},

//A B C D E F/\*

{0,0,1,1,0,0,0},//A 1

{0,1,0,1,1,1,0},//B

{0,1,1,0,0,0,1},//C 1

{0,0,1,0,0,1,0},//D 1

{0,0,1,0,1,0,1},//E

{0,0,0,1,0,1,0} //F

}; // 1 1 1

int adjacentPeaks[n][n];

int queue[n];

int countQueue = 0;

int markV[n] = {0};

struct Graph

{

int numberOfVertex;

int inputVertex[n];

int vertexCount;

int firstInputSignal;

int firstVertexSender;

};

Graph \*A = new Graph;

Graph \*B = new Graph;

Graph \*C = new Graph;

Graph \*D = new Graph;

Graph \*E = new Graph;

Graph \*F = new Graph;

Graph \* getVertex(int numberOfVertex){

switch(numberOfVertex){

case 1: return A;

case 2: return B;

case 3: return C;

case 4: return D;

case 5: return E;

case 6: return F;

}

}

void sendSignal(Graph \*vertex, int signal, int fromVertex){

cout<<"To "<<vertex->numberOfVertex<<" from "<<fromVertex<<endl;

if (vertex->inputVertex[fromVertex] == 0)

{

vertex->inputVertex[fromVertex] = signal;

vertex->vertexCount++;

}

if (vertex->firstInputSignal == 0 )

{

vertex->firstInputSignal = signal;

//cout<< vertex-> numberOfVertex<< " "<<signal<<endl;;

}

vertex->firstVertexSender = fromVertex;

}

bool checkInputVertex(Graph \*v){

int check[n] = {0,0,0,0,0,0,0};

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

{

if (v->inputVertex[i]!=0)

{

check[v->inputVertex[i]]++;

}

}

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

{

if (check[i]>1)

{

return true;

}

}

return false;

}

void createRelief(Graph \*v){

Graph \*temp;

if (markV[v->numberOfVertex]!=0)

return;

markV[v->numberOfVertex] = 1;

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

{

if (GraphMap[v->numberOfVertex][i]!=0 )

{

temp = getVertex(i);

if (markV[temp->numberOfVertex] == 0)

{

queue[countQueue]=temp->numberOfVertex;

countQueue++;

}

if (checkInputVertex(v))

{

sendSignal(temp,v->firstInputSignal + 1,v->numberOfVertex);

}

else{

if (v->firstVertexSender != i)

{

sendSignal(temp,v->firstInputSignal + 1,v->numberOfVertex);

}

}

}

}

while(countQueue!=0){

countQueue--;

temp = getVertex(queue[0]);

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

{

queue[i] = queue[i+1];

}

createRelief(temp);

}

}

void sendSignalToHead(Graph \*v, int inputSignal){

Graph \* temp;

v->firstInputSignal = inputSignal;

int lowestVertexWay = n;

int lowestVertexValue = n;

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

{

if (v->inputVertex[i]<lowestVertexValue && v->inputVertex[i]!=0 )

{

lowestVertexWay = i;

lowestVertexValue = v->inputVertex[i];

}

}

if (lowestVertexWay == n)

{

return;

}

cout<<"From "<< v->numberOfVertex<< " to "<<lowestVertexWay<<endl;

temp = getVertex(lowestVertexWay);

sendSignalToHead(temp,inputSignal);

}

int main(int argc, char const \*argv[])

{

A-> numberOfVertex = 1;

B-> numberOfVertex = 2;

C-> numberOfVertex = 3;

D-> numberOfVertex = 4;

E-> numberOfVertex = 5;

F-> numberOfVertex = 6;

createRelief(B);

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

{

B->inputVertex[i] = 0;

}

cout<<"Input point: ";

int p = 0;

cin>>p;

if(p == 1)

sendSignalToHead(A,10);

else if(p == 2)

sendSignalToHead(B,10);

else if(p == 3)

sendSignalToHead(C,10);

else if(p == 4)

sendSignalToHead(D,10);

else if(p == 5)

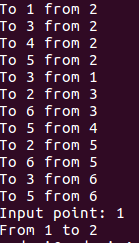
sendSignalToHead(E,10);

else if(p == 6)

sendSignalToHead(F,10);

return 0;

}



Висновки: Під час лабораторної роботи було освоєно навички реалізації методу рельєфів.