**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

дисциплина: Дискретная математика

Лабораторная работа №4.4

тема: «**Анализ алгоритмов построения покрывающего дерева минимальной стоимости**»

Выполнил: ст. группы ВТ-22  
Макаров Даниил Сергеевич

Проверил: Рязанов Ю.Д.

Белгород

2018г.

Цель занятия**:** изучить алгоритмы построения покрывающего дерева минимальной стоимости и выполнить их сравнительный анализ.

**Задания**

1. Разработать и реализовать алгоритм построения случайного

связного взвешенного графа с заданным числом вершин *n* и ребер *m*.

2. Реализовать алгоритмы построения оптимального покрывающего дерева:

2.1 Сортировка + алгоритм Краскала.

2.2 Алгоритм Краскала с выбором очередного ребра минимального веса.

2.3 Алгоритм Прима.

3. Разработать и написать программу, которая генерирует 100 случайных связных взвешенных графов с заданным числом вершин *n* и ребер *m*, для каждого графа строит покрывающее дерево минимальной

стоимости тремя алгоритмами и определяет время выполнения каждого алгоритма. Выполнить программу при

*n* = 10*,* 15 и 20. Результат для каждого *n* представить в виде таблицы.

n = 50

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кол-во ребер | n | | (n^2+3n)/6 | | n^2/3 | | (n^2-n)/2 | |
| min | max | min | max | min | max | min | max |
| Сортировка +алг. Краскала | 0 | 0 | 0 | 101 | 0 | 347 | 0 | 808 |
| Алг. Краскала с  выбором мин. ребра | 0 | 0 | 0 | 123 | 0 | 379 | 0 | 1032 |
| Алгоритм Прима | 0 | 0 | 0 | 0 | 0 | 31 | 0 | 16 |

n = 80

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кол-во ребер | n | | (n^2+3n)/6 | | n^2/3 | | (n^2-n)/2 | |
| min | max | min | max | min | max | min | max |
| Сортировка +алг. Краскала | 0 | 15 | 23 | 663 | 16 | 2514 | 47 | 5636 |
| Алг. Краскала с  выбором мин. ребра | 0 | 15 | 0 | 894 | 0 | 2453 | 53 | 5983 |
| Алгоритм Прима | 0 | 0 | 0 | 32 | 0 | 31 | 0 | 48 |

n = 100

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кол-во ребер | n | | (n^2+3n)/6 | | n^2/3 | | (n^2-n)/2 | |
| min | max | min | max | min | max | min | max |
| Сортировка +алг. Краскала | 0 | 46 | 20 | 2782 | 130 | 9864 | 227 | 22464 |
| Алг. Краскала с  выбором мин. ребра | 0 | 53 | 31 | 2289 | 85 | 10083 | 200 | 21556 |
| Алгоритм Прима | 0 | 22 | 0 | 47 | 0 | 47 | 0 | 47 |

Текст программы:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <limits.h>

int \*\*GetEdgeList(int \*\*Adj, int n, int \*m){

int edges = 0;

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

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

if (Adj[i][j]){

edges++;

}

}

}

int k = 0;

int \*\*result = malloc(edges \* sizeof(int \*));

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

result[i] = malloc(3 \* sizeof(int));

}

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

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

if (Adj[i][j]){

result[k][0] = Adj[i][j];

result[k][1] = i;

result[k][2] = j;

k++;

}

}

}

\*m = edges;

return result;

}

void swap(int \*a, int \*b){

int t = \*a;

\*a = \*b;

\*b = t;

}

void FreeEdgeList(int \*\*EL, int m){

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

free(EL[i]);

}

free(EL);

}

void SortEdges(int \*\*EdgeList, int m){

for (int i = 0; i<m-1; i++){

int min = i;

for (int j = i+1; j<m; j++){

if (EdgeList[j][0] < EdgeList[min][0]){

min = j;

}

}

for (int k = 0; k<3; k++){

swap(&EdgeList[i][k], &EdgeList[min][k]);

}

}

}

void PrintEdgeList(int \*\*EdgeList, int m){

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

printf("{w: %2i, %i<->%i}\n", EdgeList[i][0], EdgeList[i][1], EdgeList[i][2]);

}}

int \*\*MakeGraf(int n, int m){

int \*\*G = malloc(n \* sizeof(int \*));

//заполнение матрицы смежности

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

G[i] = malloc(n \* sizeof(int));

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

G[i][j] = 0;

}

}

int k = 0;

while (k<m){

int x = rand() % n;

int y = rand() % n;

if (x!=y && !G[x][y]){

G[x][y] = G[y][x] = (1 + rand()) % 15;

k++;

}

}

return G;

}

void FreeGraf(int \*\*G, int n){

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

free(G[i]);

}

free(G);

}

void PrintGraf(int \*\*G, int n){

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

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

printf("%2i ", G[i][j]);

}

printf("\n");

}}

int Kruskal(int\*\* EdgeList, int n, int m){

int count = n, i, k, x, y;

int \*B = malloc(n \* sizeof(int));

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

B[i] = i;

}

//Перебираем ребра

for(i = 0; i < m; i++){

x = EdgeList[i][1];

y = EdgeList[i][2];

if(B[x] != B[y]){

int t = B[y];

for(k = 0; k < n; k++){

if(B[k] == t){

B[k] = B[x];

}

}

count--; //количество букетов уменьшается

}

}

return count;

}

int Kruskal2(int\*\* EdgeList, int n, int m){

int count = n, i, k, x, y;

int \*B = malloc(n \* sizeof(int));

int \*S = malloc(m \* sizeof(int));

for(i = 0; i<m; i++){

S[i] = 1;

}

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

B[i] = i;

}

for(i = 0; i < m; i++){

int min = 0;

for (int j = 0; j<m; j++){

if (S[j] && EdgeList[j][0] < EdgeList[min][0]){

min = j;

}

}

x = EdgeList[i][1];

y = EdgeList[i][2];

if(B[x] != B[y]){

//объединим букеты

int t = B[y];

for(k = 0; k < n; k++){

if(B[k] == t){

B[k] = B[x];

}

}

count--;

S[i] = 0;

}else{

S[i] = 0;

}

}

return count;

}

void Prim(int \*\*G, int n){

const int inf = 9999;

int \*B = malloc(n\*sizeof(int));

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

B[i] = 0;

}

int \*T = malloc(n\*sizeof(int));

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

T[i] = 0;

}

int v = 0;

B[v] = 1;

int \*D = malloc(n\*sizeof(int));

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

D[i] = inf;

}

int y = v;

int k = 0;

while (k<n){

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

if (G[i][y] && !B[i]){

if (G[i][y] < D[i]){

D[i] = G[i][y];

T[i] = y;

}

}

}

int min = 0;

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

if (!B[i]){

if (D[i]<D[min]){

min = i;

}

}

}

B[min] = 1;

k++;

y = min;

}

}

void Test1(int n, int m, long \*min\_, long \*max\_){

clock\_t start, stop, all;

all = 0;

long min = 99999999;

long max = 0;

for (int i = 0; i<100; i++){

int \*\*G = MakeGraf(n, m);

int edges;

int \*\*EL = GetEdgeList(G, n, &edges);

start=clock();

SortEdges(EL, edges);

Kruskal(EL, n, edges);

FreeEdgeList(EL, edges);

FreeGraf(G, n);

stop=clock();

all += (stop-start);

if (all > max){

max = all;

}

if (all < min){

min = all;

}

}

\*min\_ = min;

\*max\_ = max;

}

void Test2(int n, int m, long \*min\_, long \*max\_){

clock\_t start, stop, all;

all = 0;

long min = 99999999;

long max = 0;

for (int i = 0; i<100; i++){

int \*\*G = MakeGraf(n, m);

int edges;

int \*\*EL = GetEdgeList(G, n, &edges);

start=clock();

SortEdges(EL, edges);

Kruskal(EL, n, edges);

FreeEdgeList(EL, edges);

FreeGraf(G, n);

stop=clock();

all += (stop-start);

if (all > max){

max = all;

}

if (all < min){

min = all;

}

}

\*min\_ = min;

\*max\_ = max;

}

void Test3(int n, int m, long \*min\_, long \*max\_){

clock\_t start, stop, all;

all = 0;

long min = 99999999;

long max = 0;

for (int i = 0; i<100; i++){

int \*\*G = MakeGraf(n, m);

start=clock();

Prim(G, n);

FreeGraf(G, n);

stop=clock();

all += (stop-start);

if (all > max){

max = all;

}

if (all < min){

min = all;

}

}

\*min\_ = min;

\*max\_ = max;

}

int main() {

srand(time(NULL));

printf("Kruskal 1\n");

long min, max;

for (int n = 50; n<=110; n+=30){

int m = n;

Test1(n, m, &min, &max);

printf("n = %4i, m = %4i, time = %li min, %li max\n", n, m, min, max);

m = (n\*n+3\*n)/6;

Test1(n, m, &min, &max);

printf("n = %4i, m = %4i, time = %li min, %li max\n", n, m, min, max);

m = (n\*n)/3;

Test1(n, m, &min, &max);

printf("n = %4i, m = %4i, time = %li min, %li max\n", n, m, min, max);

m = (n\*n-n)/2;

Test1(n, m, &min, &max);

printf("n = %4i, m = %4i, time = %li min, %li max\n", n, m, min, max);

}

printf("Kruskal 2\n");

for (int n = 50; n<=110; n+=30){

int m = n;

Test2(n, m, &min, &max);

printf("n = %4i, m = %4i, time = %li min, %li max\n", n, m, min, max);

m = (n\*n+3\*n)/6;

Test2(n, m, &min, &max);

printf("n = %4i, m = %4i, time = %li min, %li max\n", n, m, min, max);

m = (n\*n)/3;

Test2(n, m, &min, &max);

printf("n = %4i, m = %4i, time = %li min, %li max\n", n, m, min, max);

m = (n\*n-n)/2;

Test2(n, m, &min, &max);

printf("n = %4i, m = %4i, time = %li min, %li max\n", n, m, min, max);

}

printf("Prim\n");

for (int n = 50; n<=110; n+=30){

int m = n;

Test3(n, m, &min, &max);

printf("n = %4i, m = %4i, time = %li min, %li max\n", n, m, min, max);

m = (n\*n+3\*n)/6;

Test3(n, m, &min, &max);

printf("n = %4i, m = %4i, time = %li min, %li max\n", n, m, min, max);

m = (n\*n)/3;

Test3(n, m, &min, &max);

printf("n = %4i, m = %4i, time = %li min, %li max\n", n, m, min, max);

m = (n\*n-n)/2;

Test3(n, m, &min, &max);

printf("n = %4i, m = %4i, time = %li min, %li max\n", n, m, min, max);

}

}