# ЦЕЛЬ РАБОТЫ

Изучение метода построения кратчайших остовых деревьев графа на примере алгоритма Прима-Краскала.

# ПОСТАНОВКА ЗАДАЧИ

1. Составить схему алгоритма программы, определяющей кратчайшее остовое дерево графа с помощью алгоритма Прима-Краскала.
2. Создать программу, реализующую алгоритм Прима-Краскала.
3. Вывести список ребер, входящих в кратчайшее остовое дерево.
4. Исходный граф задается в виде матрицы смежности (табл.1).

Таблица 1. Матрица смежности, для неориентированного графа, заданного вариантом задания №4.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 |  | 9 | 4 |  |  |  |  |  |  |
| 2 | 9 |  | 7 | 10 |  |  |  |  |  |
| 3 | 4 | 7 |  | 6 |  | 4 |  | 6 |  |
| 4 |  | 10 | 6 |  | 3 |  | 1 |  |  |
| 5 |  |  |  | 3 |  | 9 | 1 |  |  |
| 6 |  |  | 4 |  | 9 |  | 8 | 7 |  |
| 7 |  |  |  | 1 | 1 | 8 |  |  | 6 |
| 8 |  |  | 6 |  |  | 7 |  |  | 4 |
| 9 |  |  |  |  |  |  | 6 | 4 |  |

# 

# Рисунок 1 – Неориентированный граф, построенный в соответствии с матрицей смежности (табл.1)

# СТРУКТУРНЫЕ СХЕМЫ АЛГОРИТМОВ



Рисунок 2 – Структурная схема алгоритма поиска кратчайшего остового дерева графа

# ТЕКСТ ПРОГРАММЫ

#include <iostream>

#include <vector>

using namespace std;

typedef vector<size\_t> arr;

typedef vector<vector<size\_t>> matrix;

const size\_t INF = numeric\_limits<size\_t>::max(); // бесконечность

const size\_t ROOT\_AMOUNT = 9; // количествовершин

void fillMatr(matrix &rootMatr);

size\_t solve(matrix &g, matrix &result, size\_t startRoot = 0);

void printMatrix(matrix &a);

int main() {

matrix ROOT\_MATRIX(ROOT\_AMOUNT, arr(ROOT\_AMOUNT,0));

matrix PATHs(2, arr(ROOT\_AMOUNT , INF));

fillMatr(ROOT\_MATRIX);

size\_t k = solve(ROOT\_MATRIX, PATHs,0);

printMatrix(PATHs);

cout << endl << "Length of path: " << k;

return 0;

}

void fillMatr(matrix &rootMatr) {

rootMatr[0][1] = 9;

rootMatr[0][2] = 4;

rootMatr[1][2] = 7;

rootMatr[1][3] = 10;

rootMatr[2][5] = 4;

rootMatr[2][7] = 6;

rootMatr[3][2] = 6;

rootMatr[3][4] = 3;

rootMatr[4][6] = 1;

rootMatr[5][4] = 9;

rootMatr[5][7] = 7;

rootMatr[6][3] = 1;

rootMatr[6][5] = 8;

rootMatr[6][8] = 6;

rootMatr[7][8] = 4;

rootMatr[1][0] = 9;

rootMatr[2][0] = 4;

rootMatr[2][1] = 7;

rootMatr[3][1] = 10;

rootMatr[5][2] = 4;

rootMatr[7][2] = 6;

rootMatr[2][3] = 6;

rootMatr[4][3] = 3;

rootMatr[6][4] = 1;

rootMatr[4][5] = 9;

rootMatr[7][5] = 7;

rootMatr[3][6] = 1;

rootMatr[5][6] = 8;

rootMatr[8][6] = 6;

rootMatr[8][7] = 4;

}

size\_t solve(matrix &g, matrix &result, size\_t startRoot) {

size\_t path = 0;

bool d[ROOT\_AMOUNT] = { false }; // массив для пометок

d[startRoot] = true;

for (size\_t i = 0; i < ROOT\_AMOUNT; ++i) {

bool flag = false;

size\_t minPath = INF;

size\_t startIndex = 0;

size\_t finishIndex = 0;

for (size\_t j = 0; j < ROOT\_AMOUNT; ++j) {

if (d[j]) {

for (size\_t k = 0; k < ROOT\_AMOUNT; ++k) {

if (!d[k] && g[j][k] && (g[j][k] < minPath)) {

minPath = g[j][k];

startIndex = j;

finishIndex = k;

flag = true;

}

}

}

}

if (flag) {

path += minPath;

}

result[0][i] = startIndex + 1;

result[1][i] = finishIndex + 1;

d[finishIndex] = true;

}

return path;

}

void printMatrix(matrix &a) {

cout << "Start roots: ";

for (size\_t i = 0; i < a[0].size()-1; ++i) {

cout << a[0][i]<< " ";

}

cout << endl;

cout << "Finish roots: ";

for (size\_t i = 0; i < a[0].size()-1; ++i) {

cout << a[1][i] << " ";

}

# РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

Теоретически найденное кратчайшее остовое дерево, заданного неориентированного графа, представлено на рисунке 3:

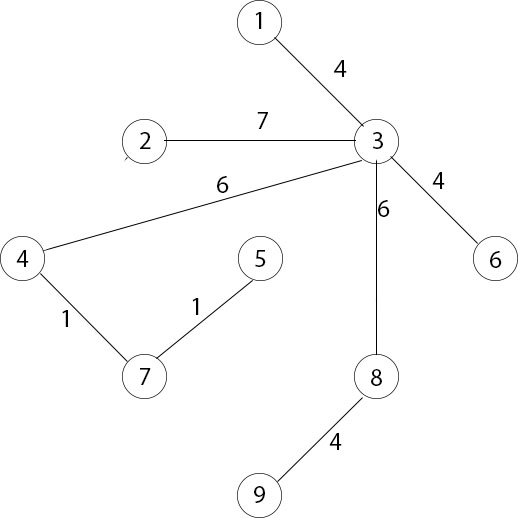


Рисунок 3 – Кратчайшее остовое дерево

Результат можно представить в виде таблицы:

Таблица 2. Номера вершин и веса дуг остового дерева.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № нач. вершины | 1 | 3 | 2 | 3 | 4 | 7 | 3 | 8 |
| № кон. вершины | 3 | 6 | 3 | 4 | 7 | 5 | 8 | 9 |
| Вес дуги | 4 | 4 | 7 | 6 | 1 | 1 | 6 | 4 |

Очевидно, что длина пути соответствует сумме весов всех дуг, и в данном случае равна 33.

Выполнив тестирование программы, получен результат, отображенный на рисунке 4.

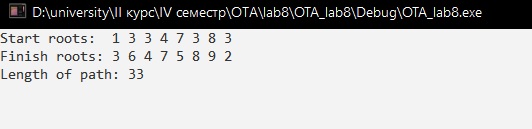


Рисунок 4 – Результат выполнения программы, поиска дуг кратчайшего остового дерева неориентированного графа

# ВЫВОДЫ

В ходе лабораторной работы были исследован алгоритм Прима-Краскала, для поиска кратчайшего остового дерева в неориентированных графах. Сравнив теоретически построенный остов, а так же его характеристики (табл.2) и результат, полученный при выполнении программы, можно сделать вывод, что они совпадают, и это значит программа работает правильно.