Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ**

Лабораторная работа №8. Построение остовного дерева минимальной стоимости

тема

Преподаватель \_\_\_\_\_\_\_\_\_ Р.Ю. Царёв

подпись, дата инициалы, фамилия

Студент КИ18-17/1б 031830504 \_\_\_\_\_\_\_\_\_ Е.В. Железкин

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2020

Содержание

[Содержание 2](#_Toc33628970)

[1 Цель работы 3](#_Toc33628971)

[2 Задача работы 3](#_Toc33628972)

[3 Ход работы 3](#_Toc33628973)

[4 Вывод 8](#_Toc33628974)

# Цель работы

Изучение алгоритмов построения остовного дерева минимальной стоимости.

# Задача работы

Реализовать один из алгоритмов построения остовного дерева минимальной стоимости неориентированного графа: алгоритм Прима или алгоритм Краскала (по выбору студента).

Программа должна наглядно отображать работы алгоритма.

# Ход работы

Листинг 1 – содержание файла main.cpp проекта Lab\_8:

#include **<iostream>**#include **<string>**#include **"inputFunc.h"**#include **"dfs.h"  
  
using namespace** std;  
  
**int** main() {  
  
 **int** temp = **NUMBER\_ERROR**;  
 string str;  
 **while** (temp == **NUMBER\_ERROR**) {  
 cout << **"Введите количество вершин: "**;  
  
 cin >> str;  
 temp = Checker(str);  
 **if** (temp == **NUMBER\_ERROR**) cout << **"Некорректные данные\n"**;  
 }  
  
 vector<edge> data;  
 vector<**int**> way;  
 **bool** visited[temp];  
 **for** (**int** i = 0; i < temp; i++)  
 visited[i] = **false**;  
  
 **int** size = temp;  
 **int** v = 0;  
 **int** u = 0;  
 getline(cin, str);  
  
 **while** (**true**){  
 cout << **"Введите две вершины графа и длинну ребра для его создания(для выхода введите -1):\n"**;  
  
 temp = **NUMBER\_ERROR**;  
  
 **while** (temp == **NUMBER\_ERROR** || temp < 0 || temp > size) {  
 cout << **"Первая вершина: "**;  
 getline(cin, str);  
 temp = Checker(str);  
 **if** (temp == -1) **break**;  
 **if** (temp == **NUMBER\_ERROR** || temp < 0 || temp > size)  
 cout << **"Некорректные данные\n"**;  
 }  
 **if** (temp == -1) **break**;  
 v = temp - 1;  
 temp = **NUMBER\_ERROR**;  
  
 **while** (temp == **NUMBER\_ERROR** || temp < 0 || temp > size) {  
 cout << **"Вторая вершина: "**;  
 getline(cin, str);  
 temp = Checker(str);  
 **if** (temp == **NUMBER\_ERROR** || temp < 0 || temp > size)  
 cout << **"Некорректные данные\n"**;  
 }  
 u = temp - 1;  
  
 temp = **NUMBER\_ERROR**;  
  
 **while** (temp == **NUMBER\_ERROR** || temp < 0) {  
 cout << **"Вес: "**;  
 getline(cin, str);  
 temp = Checker(str);  
 **if** (temp == **NUMBER\_ERROR** || temp < 0)  
 cout << **"Некорректные данные\n"**;  
 }  
  
 data.push\_back(edge{temp, v, u, **false**});  
  
 }  
  
 visited[0] = **true**;  
 way.push\_back(0);  
 cout << **"Вес остова - "** << dfs(data, visited, &way, size) << **"Вершины, входящие в остов: "**;  
  
 **for** (**int** i : way)  
 cout << i + 1 << **" "**;  
  
 **return** 0;  
  
}

Листинг 2 – содержание файла inputFunc.cpp проекта Lab\_8:

#include **"inputFunc.h"  
  
  
int** Checker(string tempForCheck)  
{  
 **int** number = 0;  
 **int** flag = 0;  
  
 **for** (**unsigned int** i = 0; i < tempForCheck.length(); i++)  
 {  
 **if** (i == 0 && tempForCheck[0] == **'-'**)  
 {  
 flag++;  
 **continue**;  
 }  
  
 **if** (tempForCheck[i] == **'0'** || tempForCheck[i] == **'1'** || tempForCheck[i] == **'2'** || tempForCheck[i] == **'3'** ||  
 tempForCheck[i] == **'4'** || tempForCheck[i] == **'5'** || tempForCheck[i] == **'6'** ||  
 tempForCheck[i] == **'7'** || tempForCheck[i] == **'8'** || tempForCheck[i] == **'9'**) number = number \* **ORDER** +  
 (tempForCheck[i] - **ASCII\_CONST**);  
 **else  
 return NUMBER\_ERROR**;  
 }  
  
 **if** (flag > 0) number \*= **NEGATIVE**;  
  
 **return** number;  
}

Листинг 3 – содержание файла inputFunc.h проекта Lab\_8:

#ifndef **LAB\_3\_INPUTFUNC\_H**#define **LAB\_3\_INPUTFUNC\_H**#include **<iostream>**#define **NUMBER\_ERROR** -322  
  
#define **ORDER** 10  
  
#define **NEGATIVE** -1  
  
#define **ASCII\_CONST** 48  
  
**using namespace** std;  
  
**int** Checker(string tempForCheck);  
  
#endif *//LAB\_3\_INPUTFUNC\_H*

Листинг 4 – содержание файла dfs.cpp проекта Lab\_8:

#include **<iostream>**#include **"dfs.h"  
  
using namespace** std;  
  
**int** dfs(vector<edge> data, **bool**\* visited, vector<**int**>\* way, **int** size) {  
  
 **int** min = **MAX**;  
 **int** res = 0;  
 **int** temp = 0;  
 **int** sum = 0;  
  
 **for** (**int** j = 0; j < size - 1; j++){  
  
 min = **MAX**;  
 res = 0;  
 temp = -1;  
  
 **for** (**int** i = 0; i < data.size(); i++){  
 **if** (visited[data[i].u] == 0 && visited[data[i].v] != 0 && min > data[i].length && data[i].flag == 0) {  
 min = data[i].length;  
 temp = data[i].u;  
 res = i;  
 }  
  
 **if** (visited[data[i].v] == 0 && visited[data[i].u] != 0 && min > data[i].length && data[i].flag == 0) {  
 min = data[i].length;  
 temp = data[i].v;  
 res = i;  
 }  
 }  
  
 data[res].flag = **true**;  
 visited[temp] = **true**;  
 way->push\_back(temp);  
 sum += data[res].length;  
  
 }  
  
 **return** sum;  
  
}

Листинг 5 – содержание файла dfs.h проекта Lab\_8:

#ifndef **LAB\_6\_DFS\_H**#define **LAB\_6\_DFS\_H**#define **MAX** 999999999  
  
#include **<vector>  
  
struct** edge{  
 **int** length;  
 **int** u;  
 **int** v;  
 **bool** flag;  
};  
**int** dfs(std::vector<edge>, **bool**\*, std::vector<**int**>\*, **int**);  
  
#endif *//LAB\_6\_DFS\_H*

1. Примеры работы:

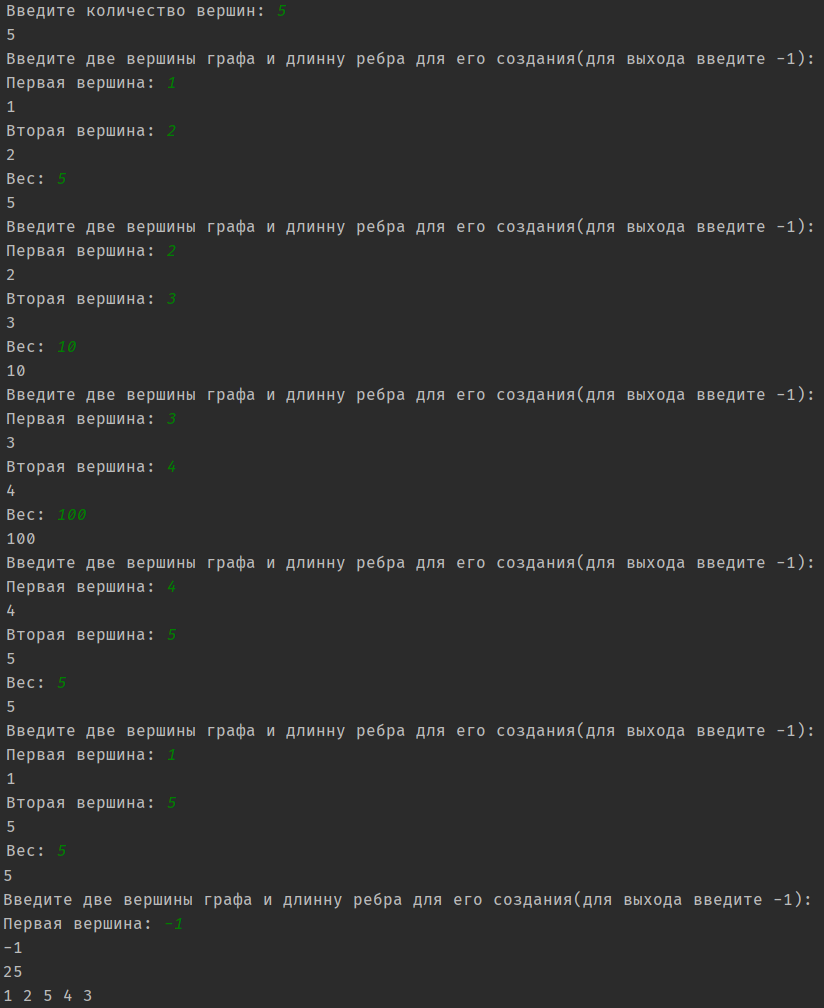


Рисунок 1 – результаты работы алгоритма Дейкстры

# Вывод

В ходе данной лабораторной работы был реализован алгоритм построения остовного дерева минимальной стоимости(Прима).