МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

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

Лабораторная работа №4.3

по дисциплине: Дискретная математика
тема: «Графы: Связность»

Выполнил: ст. группы ВТ-211
Руденко Ксения Ильинична

Проверили:
Рязанов Юрий Дмитриевич
Бондаренко Татьяна Владимировна

Белгород, 2022 г.

Лабораторная работа №4.3 «Графы: Связность»

Цель работы: изучить алгоритм Краскала построения покрывающего леса, научиться использовать его при решении различных задач.

Содержание отчета:

- Тема лабораторной работы;
- Цель лабораторной работы;
- Задания к лабораторной работе;
- Выводы.

Задания к лабораторной работе: вариант №2.

1. Реализовать алгоритм Краскала построения покрывающего леса.

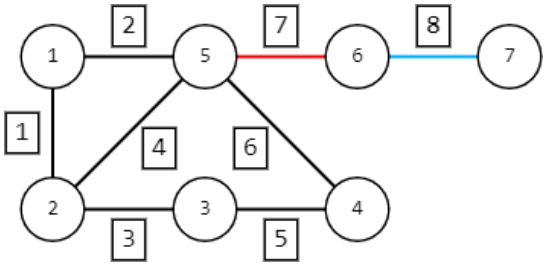
```
int kruskal_algorithm(Graph &g) {
    vector<int> b(g.rows);
    for (size_t i = 0; i < g.rows; i++)
    b[i] = i;

    int count = g.rows;
    for (size_t i = 0; i < g.rows; i++)
        for (size_t j = 0; j < g.columns; j++)
            if (g.values[i][j] && b[i] != b[j]) {
                for (size_t k = 0; k < b.size(); k++)
                    if (b[k] == b[j])
                        b[k] = b[i];

                count--;
            }

    return count;
}
```

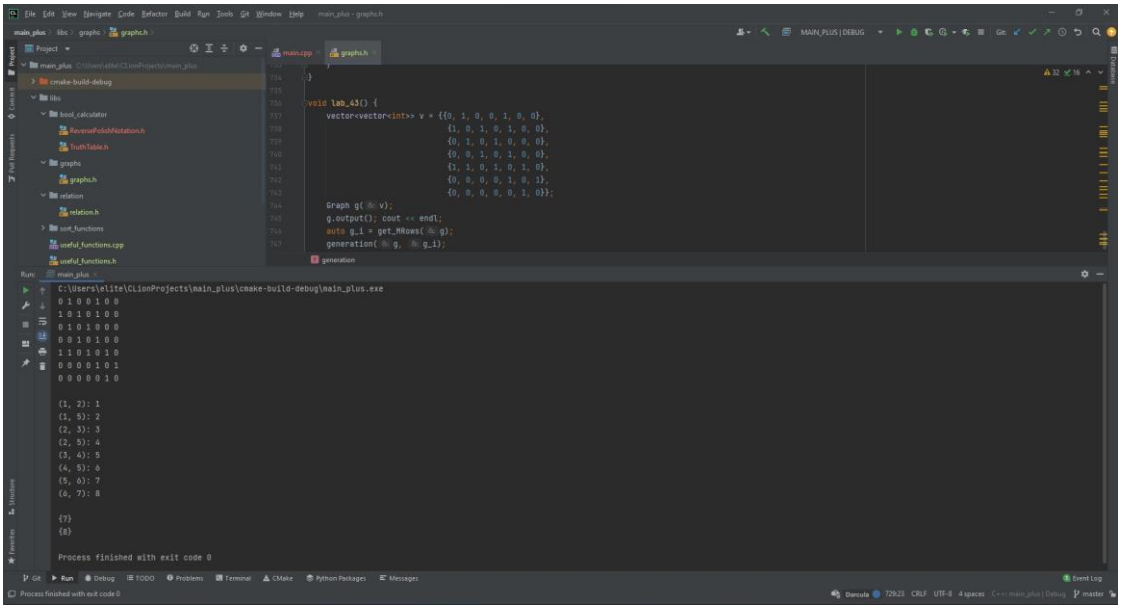
2. Используя алгоритм Краскала, разработать и реализовать алгоритм решения задачи: «Найти минимальное множество ребер, удаление которых из связного графа делает его несвязным».

```
Graph get_MRows(Graph &g) {
    Graph e(2, (g.rows * g.rows - g.rows) / 2);
    int count = 0;
    for (size_t i = 0; i < g.rows; i++)
        for (size_t j = i; j < g.columns; j++)
            if (g.values[i][j]) {
```

```
                e.values[0][count] = i;
                e.values[1][count] = j;
                cout << ' (' << i + 1 << ", " << j + 1 << "): " << count + 1 << endl;
                count++;
            }

    cout << endl;
    e.columns = count;
    for (size_t i = 0; i < 1; ++i)
        e.values[i].resize(count);

    return e;
}

void generation_(Graph &g, Graph &e, vector<int> tree, size_t i, int b, bool &isFind) {
    for (size_t x = b; x <= (e.columns - tree.size() + i); x++) {
        tree[i] = x + 1;
        g.values[e.values[0][x]][e.values[1][x]] = 0;
        g.values[e.values[1][x]][e.values[0][x]] = 0;
        if (i == (tree.size() - 1)) {
            if (kruskal_algorithm(g) == 2) {
                cout << tree;
                isFind = true;
            }
        } else
        generation_(g, e, tree, i + 1, x + 1, isFind);

        g.values[e.values[0][x]][e.values[1][x]] = 1;
        g.values[e.values[1][x]][e.values[0][x]] = 1;
    }
}

void generation(Graph &g, Graph &e) {
    bool find = false;
    for (size_t i = 1; !find and i < g.rows; ++i) {
        vector<int> tree(i);
        generation_(g, e, tree, 0, 0, find);
    }
}

void lab_43() {
    vector<vector<int>> v = {{0, 1, 0, 0, 1, 0, 0},
                             {1, 0, 1, 0, 1, 0, 0},
                             {0, 1, 0, 1, 0, 0, 0},
                             {0, 0, 1, 0, 1, 0, 0},
                             {1, 1, 0, 1, 0, 1, 0},
                             {0, 0, 0, 0, 1, 0, 1},
                             {0, 0, 0, 0, 0, 1, 0}};
    Graph g(v);
    g.output(); cout << endl;
    auto g_i = get_MRows(g);
    generation(g, g_i);
}
```

3. Подобрать тестовые данные. Результат представить в виде диаграммы графа.





Вывод: изучили алгоритм Краскала построения покрывающего леса, научились использовать его при решении различных задач.