

Отчёт по расчётной работе

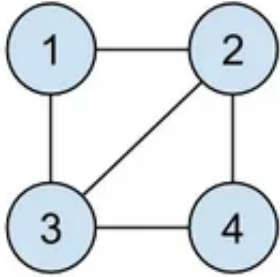
Цель работы - изучить основы теории графов, способы представления графов, базовые алгоритмы для работы с разными видами графов.

Задание: реализовать код на языке 'C++', который, используя в качестве способа выражения графа матрицу инцидентности, будет находить минимальный простой разрез графа.

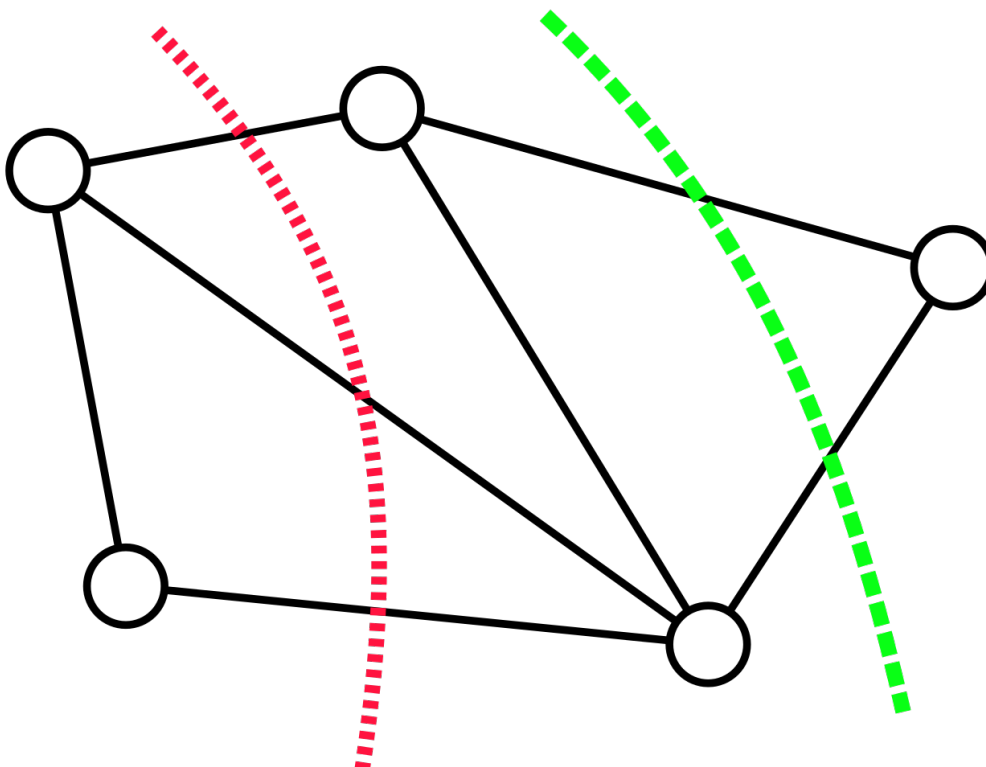
Ключевые понятия:

Граф - математическая абстракция реальной системы любой природы, объекты которой обладают парными связями.

Матрица инцидентности - одна из форм представления графа, в которой указываются связи между инцидентными элементами графа (ребро(дуга) и вершина). Столбцы матрицы соответствуют ребрам, строки — вершинам. Ненулевое значение в ячейке матрицы указывает связь между вершиной и ребром (их инцидентность).

Граф	Матрица инцидентности
	<div><div>Ребро 1-2 ↓</div><div>Ребро 1-3 ↓</div><div>Ребро 2-3 ↓</div><div>Ребро 2-4 ↓</div><div>Ребро 3-4 ↓</div></div> <div>Вершина 1 →</div> <div>Вершина 2 →</div> <div>Вершина 3 →</div> <div>Вершина 4 →</div> <div>$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$</div>

Разрез графа — это разбиение вершин графа на два непустых непересекающихся подмножества.



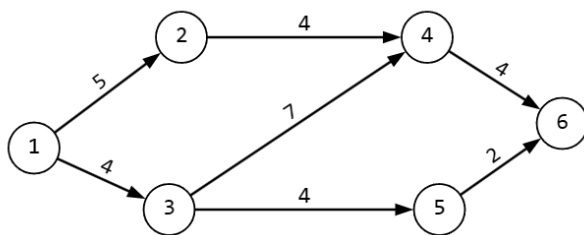
Минимальный разрез графа — разрез, при котором размер или вес разреза не превышает размер

любого другого разреза. Для невзвешенного графика минимальным разрезом будет просто разрез с наименьшим количеством ребер. Для взвешенного графика сумма весов всех ребер на разрезе определяет, является ли это минимальным разрезом.

Алгоритм выполнения задания

1. Находим 2 строки, суммы модулей чисел в которых максимальны.
2. Проверяем есть ли в этих строках на одинаковых местах равные элементы.
 - (a) Если такие элементы есть, удалить столбцы с номерами этих элементов.
 - (b) Если нет - вместо второй строки взять следующую по сумме модулей элементов и вернуться ко 2 пункту алгоритма.
3. Прибавить вторую строку к первой.
4. Удалить вторую строку.
5. Пока количество строк больше 2 повторять 1-5 пункты алгоритма.
6. Найти сумму модулей чисел первой строки в получившейся матрице.
7. Вывести эту сумму на экран.

Пример выполнения алгоритма

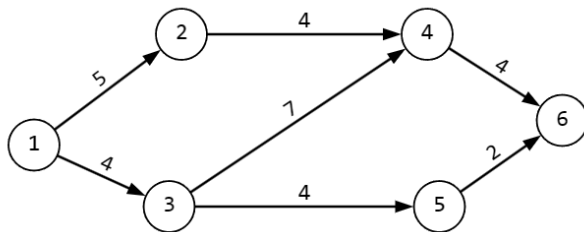


Матрица инцидентности

```

5, 4, 0, 0, 0, 0, 0
-5, 0, 4, 0, 0, 0, 0
0, -4, 0, 7, 4, 0, 0
0, 0, -4, -7, 0, 4, 0
0, 0, 0, 0, -4, 0, 2
0, 0, 0, 0, 0, -4, -2

```

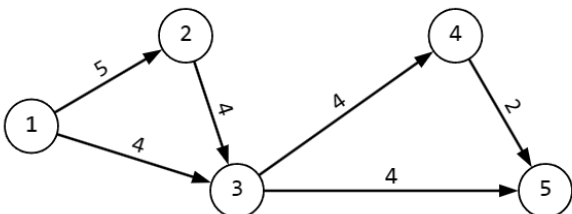


Матрица инцидентности

```

5, 4, 0, 0, 0, 0, 0
-5, 0, 4, 0, 0, 0, 0
0, -4, 0, 7, 4, 0, 0
0, 0, -4, -7, 0, 4, 0
0, 0, 0, 0, -4, 0, 2
0, 0, 0, 0, 0, -4, -2

```

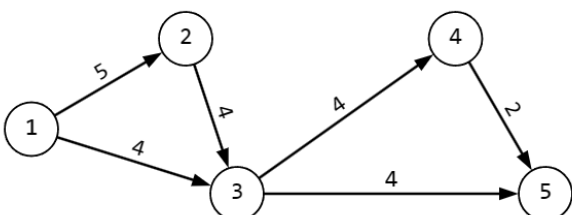


Матрица инцидентности

```

5, 4, 0, 0, 0, 0, 0
-5, 0, 4, 0, 0, 0, 0
0, -4, -4, 4, 4, 0, 0
0, 0, 0, -4, 0, 2, 2
0, 0, 0, 0, -4, -4, -2

```

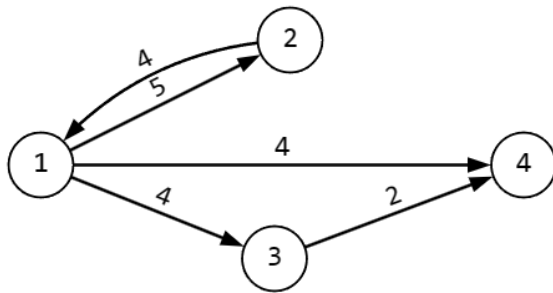


Матрица инцидентности

```

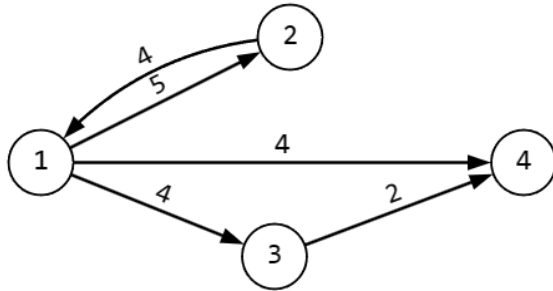
5, 4, 0, 0, 0, 0, 0
-5, 0, 4, 0, 0, 0, 0
0, -4, -4, 4, 4, 0, 0
0, 0, 0, -4, 0, 2, 2
0, 0, 0, 0, -4, -4, -2

```



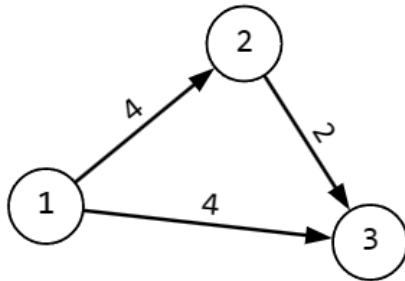
Матрица инцидентности

5	-4	4	4	0
-5	4	0	0	0
0	0	-4	0	2
0	0	0	-4	-2



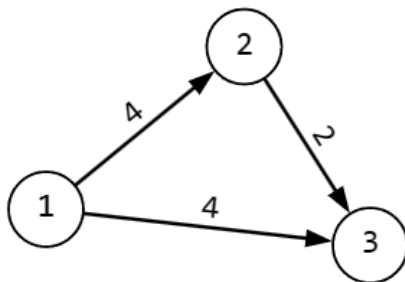
Матрица инцидентности

5	-4	4	4	0
-5	4	0	0	0
0	0	-4	0	2
0	0	0	-4	-2



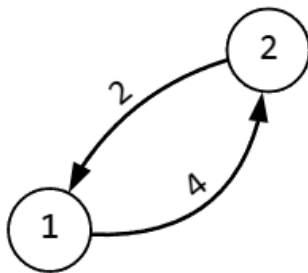
Матрица инцидентности

4	4	0
-4	0	2
0	-4	-2



Матрица инцидентности

4	4	0
-4	0	2
0	-4	-2



Матрица инцидентности

4	-2
-4	2

Код программы

```
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;

// Функция, которая возвращает сумму модулей элементов вектора
int sum_abs(vector<int> v) {
    int sum = 0;
```

```

    for (int i = 0; i < v.size(); i++) {
        sum += abs(v[i]);
    }
    return sum;
}

// Функция, которая возвращает индекс вектора с максимальной суммой модулей элементов из списка векторов
int max_sum_abs_index(vector<vector<int>> v) {
    int max_index = 0;
    int max_sum = sum_abs(v[0]);
    for (int i = 1; i < v.size(); i++) {
        int curr_sum = sum_abs(v[i]);
        if (curr_sum > max_sum) {
            max_sum = curr_sum;
            max_index = i;
        }
    }
    return max_index;
}

// Функция, которая удаляет столбец с заданным индексом из матрицы
void delete_column(vector<vector<int>>& v, int index) {
    for (int i = 0; i < v.size(); i++) {
        v[i].erase(v[i].begin() + index);
    }
}

// Функция, которая прибавляет второй вектор к первому поэлементно
void add_vectors(vector<int>& v1, vector<int>& v2) {
    for (int i = 0; i < v1.size(); i++) {
        v1[i] += v2[i];
    }
}

// Функция, которая выполняет алгоритм из задания
int algorithm(vector<vector<int>> v) {
    // Пока количество строк больше 2
    while (v.size() > 2) {
        // Находим 2 строки, суммы модулей чисел в которых максимальны
        int i1 = max_sum_abs_index(v);
        vector<int> v1 = v[i1];
        v.erase(v.begin() + i1);
        int i2 = max_sum_abs_index(v);
        vector<int> v2 = v[i2];
        v.erase(v.begin() + i2);

        // Проверяем есть ли в этих строках на одинаковых местах равные числа
        bool equal_found = false;
        do {
            equal_found = false;
            for (int i = 0; i < v1.size(); i++) {
                if (abs(v1[i]) == abs(v2[i])) {
                    // Если такие числа есть, удалить столбец с номером этого места
                    delete_column(v, i);
                    v1.erase(v1.begin() + i);
                    equal_found = true;
                    break;
                }
            }
        }
        // Если нет - вместо второй строки взять следующую по сумме модулей элементов
    }
}

```

```

        if (!equal_found) {
            i2 = max_sum_abs_index(v);
            v2 = v[i2];
            v.erase(v.begin() + i2);
        }
    } while (!equal_found);

    // Прибавить вторую строку к первой
    add_vectors(v1, v2);

    // Удалить вторую строку
    v2.clear();
}

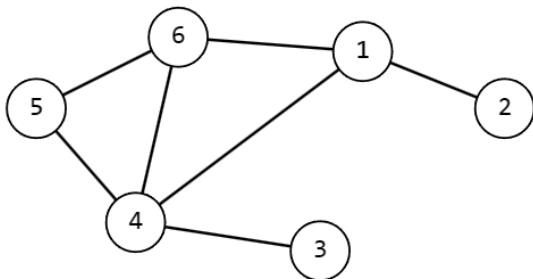
// Найти сумму модулей чисел первой строки в получившейся матрице
int result = sum_abs(v[0]);

// Вернуть эту сумму
return result;
}

int main() {
    setlocale(LC_ALL, "Russian");
    int n, m;
    cout << "Введите количество вершин в графе: ";
    cin >> n;
    cout << "Введите количество рёбер в графе: ";
    cin >> m;
    vector<vector<int>> v(n, vector<int>(m));
    cout << "Введите матрицу инцидентности: " << endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> v[i][j];
        }
    }
    int result = algorithm(v);
    cout << "Размер минимального разреза данного графа равен: " << result;
    return 0;
}

```

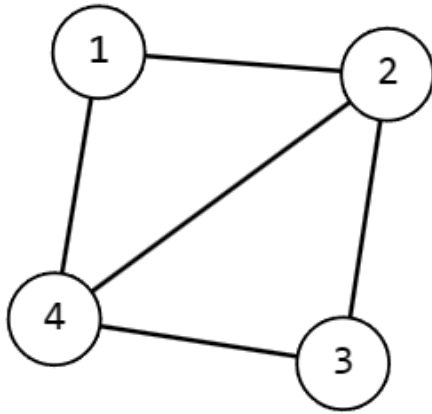
Тестирование



```

Консоль отладки Microsoft V  ×  +  ▾
Введите количество вершин в графе: 6
Введите количество рёбер в графе: 7
Введите матрицу инцидентности:
1 1 0 0 0 1 0
1 0 0 0 0 0 0
0 0 0 0 0 0 1
0 0 0 1 1 1 1
0 0 1 0 1 0 0
0 1 1 1 0 0 0
Размер минимального разреза данного графа равен: 1

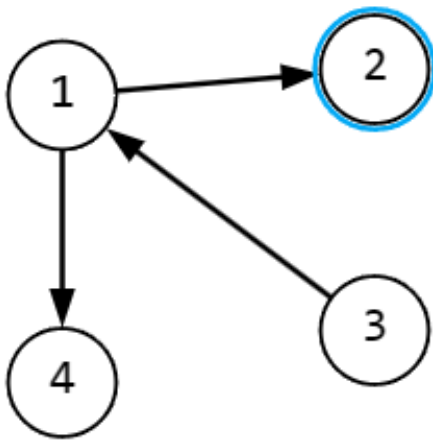
```



```

Консоль отладки Microsoft V
Введите количество вершин в графе: 4
Введите количество рёбер в графе: 5
Введите матрицу инцидентности:
1 1 0 0 0
1 0 0 1 1
0 0 1 1 0
0 1 1 0 1
Размер минимального разреза данного графа равен: 2

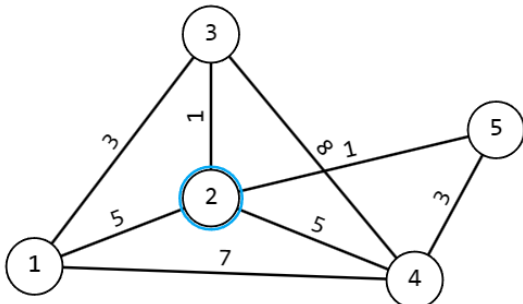
```



```

Консоль отладки Microsoft V
Введите количество вершин в графе: 4
Введите количество рёбер в графе: 3
Введите матрицу инцидентности:
1 1 -1
-1 0 0
0 0 1
0 -1 0
Размер минимального разреза данного графа равен: 1

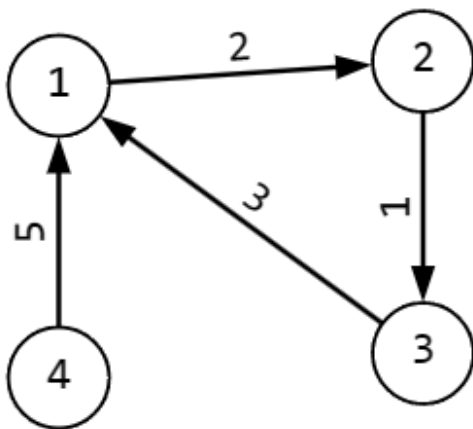
```



```

Консоль отладки Microsoft V
Введите количество вершин в графе: 5
Введите количество рёбер в графе: 8
Введите матрицу инцидентности:
3 5 7 0 0 0 0 0
0 5 0 0 1 1 0 5
3 0 0 8 1 0 0 0
0 0 7 8 0 0 3 5
0 0 0 0 0 1 3 0
Размер минимального разреза данного графа равен: 4

```



```

Консоль отладки Microsoft V
Введите количество вершин в графе: 4
Введите количество рёбер в графе: 4
Введите матрицу инцидентности:
-5 2 -3 0
0 -2 0 1
0 0 3 -1
5 0 0 0
Размер минимального разреза данного графа равен: 3

```

Выводы

В результате выполнения данной работы были получены следующие практические навыки:

- изучены основы теории графов;

- изучены способы представления графов;
- изучены базовые алгоритмы для работы с графами.