Задание 4 исправленное) Предложите структуры данных для представления АТД из задания 3. Перечислите поля, их типы и комментарии к каждому полю. Укажите оценку эффективности (амортизированную или среднюю) для каждого метода с учетом использованных структур данных. Хранимая в структуре информация не должна дублироваться.

Структуры данных и их поля:

Таблица StolbName: vector— Массив названий столбцов

Stroki: vector<vector> — Массив массивов строк, где каждая строка представляет собой массив значений.

StolbKey: string — Название столбца, по которому осуществляется сортировка.

indexStolbKey: int — Индекс ключевого столбца.

SortedIndex: vector — Отсортированный массив индексов строк для итерации по ключевому столбцу.

Оценка эффективности методов:

CreateTable:O(1)

Stradd:O(n)

StrDel:O(n)

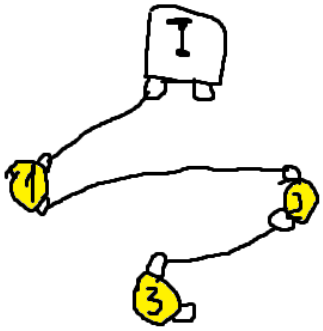
StrGet:O(1)

ValueUpd:O(1)

SetupSortKey:O(nlogn)

GetNextStr:O(1)

Задание 17)Укажите какой алгоритм нужно использовать для решения задачи и обоснуйте свой выбор. Из каких вершин и ребер граф будет состоять?
Джон решил украсить дом иллюминацией к Рождеству. Он соединил проводами несколько ламп и подключил их к электросети. Затем Джон повернул рубильник, но некоторые лампы не загорелись. Напишите программу, которая определит, используя информацию о выполненных соединениях, какие лампы будут гореть.
В первой строке ввода содержатся два целых числа, разделенных пробелом – количество ламп NN (1≤N≤1001≤N≤100) и количество проводов KK (1≤K≤10001≤K≤1000). Далее следует KK строк, в каждой строке сначала указывается номер элемента (0 для электросети, от 1 до NN для лампы), к которому был присоединен один из концов провода, потом через пробел – номер контакта 1 или 2 (все элементы имеют два контакта), затем указывается элемент и контакт, к которому был присоединен другой конец провода, в том же формате.
В выходной файл в первой строке вывести NN целых чисел, разделяя их пробелами – если ii-ая лампа будет гореть, то ii-ое число должно быть равно 1, иначе 0. Лампа будет гореть, только если ее контакты подсоединены к вершинам с различными потенциалами. Провод имеет нулевое сопротивление, поэтому разность потенциалов на вершинах, соединенных между собой проводами без нагрузки в виде других ламп будет равна 0.



Все лампы горят

```
>Запуск прог
3 3
0 1 1 1
1 2 2 1
2 2 3 1
```

Для решения задачи определения, какие лампы будут гореть, нужно использовать алгоритм поиска в ширину (BFS) для определения компонент связности в графе. Основная идея заключается в том, что если два контакта одной лампы принадлежат разным компонентам связности, то лампа будет гореть.

Граф состоит из вершин, представляющих контакты элементов (электросеть и контакты ламп),ребра графа представляют собой провода, соединяющие контакты. Что я и пытался продемонстрировать на рисунке.

Код:

```
#include <iostream>
#include <vector>
#include <queue>
#include <unordered_map>

using namespace std;

void bfs(int start, vector<vector<int>>& adj, vector<int>& component, int comp_id) {
    queue<int> q;
    q.push(start);
    component[start] = comp_id;

    while (!q.empty()) {
        int u = q.front();
        q.pop();

        for (int v : adj[u]) {
            if (component[v] == -1) {
                component[v] = comp_id;
                q.push(v);
            }
        }
    }
}

int main() {
    int N, K;
    cin >> N >> K;

    // 2*N + 2 контактов (2 контакта для каждой лампы + 2 контакта для электросети по условию)
```

```
vector<vector<int>> adj(2 * N + 2);
unordered_map<int, int> contact_map;

// Чтение входных данных и построение графа
for (int i = 0; i < K; ++i) {
    int e1, c1, e2, c2;
    cin >> e1 >> c1 >> e2 >> c2;

    // Преобразование контактов в уникальные индексы
    int u = (e1 == 0) ? c1 - 1 : 2 * (e1 - 1) + c1 - 1;
    int v = (e2 == 0) ? c2 - 1 : 2 * (e2 - 1) + c2 - 1;

    adj[u].push_back(v);
    adj[v].push_back(u);
}

// Инициализация компонент связности
vector<int> component(2 * N + 2, -1);
int comp_id = 0;

// Поиск компонент связности
for (int i = 0; i < 2 * N + 2; ++i) {
    if (component[i] == -1) {
        bfs(i, adj, component, comp_id++);
    }
}

// Проверка ламп
vector<int> result(N);
for (int i = 0; i < N; ++i) {
    int c1 = 2 * i;     // Контакт 1 лампы i
    int c2 = 2 * i + 1; // Контакт 2 лампы i
```

```
        if (component[c1] != component[c2]) {
            result[i] = 1;
        } else {
            result[i] = 0;
        }
    }

    // Вывод результата
    for (int i = 0; i < N; ++i) {
        cout << result[i] << " ";
    }
    cout << endl;

    return 0;
}
```

Задание 10 исправленное) Определить АТД Полином, обеспечивающий метод calc для вычисления значения полинома в точке xx (используйте схему Горнера или барицентрическую форму интерполяционного многочлена Лагранжа). Реализовать полином через представление на значениях в точках. В конструкторе задается набор значений y0,...,yn−1y0,...,yn-1, x0x0 и ΔxΔx (xi=x0+i·Δxxi=x0+i·Δx). Определить операцию +.

```
#include <iostream>
#include <vector>
using namespace std;
class Polynomial {
public:
    Polynomial(const std::vector<double>& y_values, double x0, double delta_x)
        : y_values(y_values), x0(x0), delta_x(delta_x), n(y_values.size()) {}

    // Метод для отображения полинома
    void print() const {
        cout << "Polynomial(y_values=[";
        for (size_t i = 0; i < n; ++i) {
            cout << y_values[i];
            if (i < n - 1) {
                cout << ", ";
            }
        }
        cout << "], x0=" << x0 << ", delta_x=" << delta_x << ")" <<endl;
    }

    // Функция для вычисления значения полинома с использованием схемы Горнера
    double calc(const vector<double>& y_values, double x) const {//Метод из лекции
        double r = y_values.back();
        for (int i = y_values.size() - 2; i >= 0; --i) {
            r = r * x + y_values[i];
        }
        return r;
```

```
    }

private:
    vector<double> y_values;
    double x0;
    double delta_x;
    size_t n;
};

int main() {
    // Пример использования:
    vector<double> y_values = {1, 2, 3}; // Полином вида: 3x^2 +2x +1
    double x0 = 0.0;
    double delta_x = 1.0;

    Polynomial p(y_values, x0, delta_x);

   cout << "Полином: ";
    p.print();

    // Вычисление значения полинома в точке x
    double x = 2.0;
    cout << "Значение полинома в точке " << x << ": " << p.calc(y_values, x) << endl;

    return 0;
}
```

Задание 12) Определить АТД Матрица, обеспечивающий метод [i,ji,j] для доступа к элементам матрицы. В конструкторе задаются размеры матрицы.
Определить операцию *. Выполнить возведение матрицы N×N$N\times N$ из целых чисел 0 и 1 в степень KK по модулю 2: Aк(mod2)$A^K(\mathrm{mod}2)$. Для возведения матрицы в степень использовать метод уменьшения размера задачи в 2 раза.

```
#include <iostream>
#include <vector>
using namespace std;
class Matrix {
public:
    Matrix(size_t rows, size_t cols)
        : rows_(rows), cols_(cols), data_(rows, vector<int>(cols)) {}

    // Оператор для доступа к элементам матрицы
    vector<int>& operator[](size_t i) {
        return data_[i];
    }

    const vector<int>& operator[](size_t i) const {
        return data_[i];
    }

    // Метод для возведения матрицы в степень по модулю 2
    Matrix pow(size_t k) const {
        if (rows_ != cols_) {
            throw std::invalid_argument("Матрица должна быть квадратной");
        }

        Matrix result = identity(rows_);
        Matrix base = *this;

        while (k > 0) {
            if (k % 2 == 1) {
```

```
            result = result * base;
        }
        base = base * base;
        k /= 2;
    }

    return result;
}

// Оператор умножения матриц по модулю 2
Matrix operator*(const Matrix& other) const {
    if (cols_ != other.rows_) {
        throw invalid_argument("Матрицы должны быть одинаковой размерности.");
    }

    Matrix result(rows_, other.cols_);

    for (size_t i = 0; i < rows_; ++i) {
        for (size_t j = 0; j < other.cols_; ++j) {
            result[i][j] = 0;
            for (size_t k = 0; k < cols_; ++k) {
                result[i][j] ^= (data_[i][k] * other[k][j]) % 2;
            }
        }
    }

    return result;
}

// Метод для печати матрицы
void print() const {
    for (size_t i = 0; i < rows_; ++i) {
```

```
            for (size_t j = 0; j < cols_; ++j) {
                cout << data_[i][j] << " ";
            }
            cout << endl;
        }
    }

private:
    size_t rows_, cols_;
    vector<vector<int>> data_;

    // Метод для создания единичной матрицы
    static Matrix identity(size_t size) {
        Matrix id(size, size);
        for (size_t i = 0; i < size; ++i) {
            id[i][i] = 1;
        }
        return id;
    }
};

int main() {
    size_t n = 3; // Размер матрицы
    Matrix mat(n, n);

    // Заполнение матрицы значениями 0 и 1
    mat[0][0] = 1; mat[0][1] = 1; mat[0][2] = 0;
    mat[1][0] = 0; mat[1][1] = 1; mat[1][2] = 1;
    mat[2][0] = 1; mat[2][1] = 0; mat[2][2] = 1;

    cout << "Исходная матрица:" << endl;
    mat.print();
```

```
    size_t k = 3; // Степень
    Matrix result = mat.pow(k);

   cout << "Матрица в степени " << k << " по модулю 2:" << endl;
    result.print();

    return 0;
}
```