ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

ОТЧЕТ

о выполнении лабораторной работы №8

«Графы»

по дисциплине

«Дискретный анализ»

Выполнил студент группы М8О-308Б-23:

Ибрагимов Далгат Магомедалиевич

Проверил:

Макаров Н.К.

Москва, 2025

Постановка задачи

Вариант 8: F.8 Поиск максимального паросочетания алгоритмом Куна

Задан неориентированный двудольный граф из n вершин и m ребер. Вершины пронумерованы от 1 до n. Требуется найти максимальное паросочетание в графе алгоритмом Куна. Для однозначности ответа списки смежности предварительно сортируются. Граф не содержит петель и кратных ребер.

Цель работы

Изучение представления двудольных графов и реализация поиска максимального паросочетания с помощью алгоритма Куна (поиск увеличивающих путей).

Задание

Формат ввода: В первой строке заданы $1 \leq n \leq 110000$ и $1 \leq m \leq 40000$. В следующих m строках записаны ребра: пара чисел — номера вершин, соединенных ребром.

Формат вывода: В первой строке вывести число ребер в найденном паросочетании. В следующих строках вывести ребра паросочетания по одному в строке. Каждое ребро — пара номеров вершин. В каждой паре числа упорядочены по возрастанию. Строки (ребра) отсортированы по минимальному номеру вершины на ребре.

Алгоритм решения

Решение состоит из следующих этапов:

1. **Чтение графа и построение списков смежности.** Граф неориентированный, поэтому каждое ребро добавляется в обе стороны.
2. **Сортировка списков смежности.** Необходима для детерминированности выбора увеличивающих путей и однозначности результата.
3. **Построение двудольного разбиения (раскраска в 2 цвета).** Так как вход задает только граф, а не доли, разбиение восстанавливается BFS-раскраской по компонентам связности.
4. **Поиск максимального паросочетания алгоритмом Куна.** Для каждой вершины левой доли (цвет 0) запускается DFS-поиск увеличивающего пути. Массив mt хранит текущую пару для вершин правой доли (цвет 1): $mt[v] = u$ означает, что вершина v сматчена с u.
5. **Формирование и сортировка ответа.** Из массива mt собираются ребра паросочетания, внутри ребра вершины упорядочиваются, затем весь список сортируется лексикографически (что эквивалентно требованию сортировки по минимальному номеру вершины).

Реализация программы

Программа реализована на языке C++ и включает ключевые компоненты:

- `BFS`-раскраска вершин (`side`) для восстановления долей двудольного графа.
- `bool kuhn(int v)` — DFS-поиск увеличивающего пути из вершины левой доли.
- `mt` — массив соответствий для вершин правой доли.

Для ускорения работы на больших n используется “таймер” посещений: вместо переинициализации массива `used` на каждой итерации применяется массив целых меток и счетчик запусков.

Листинг кода

```cpp
#include <bits/stdc++.h>
using namespace std;

int n, m;
vector<vector<int>> g;
vector<int> side;
vector<int> mt;
vector<int> used;
int timer = 0;

bool kuhn(int v) {
    if (used[v] == timer) return false;
    used[v] = timer;

    for (int to : g[v]) {
        if (mt[to] == -1 || kuhn(mt[to])) {
            mt[to] = v;
            return true;
        }
    }
    return false;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    cin >> n >> m;
    g.assign(n, {});

```

```
    for (int i = 0; i < m; ++i) {
        int u, v;
        cin >> u >> v;
        --u; --v;
        g[u].push_back(v);
        g[v].push_back(u);
    }

    for (int i = 0; i < n; ++i) {
        sort(g[i].begin(), g[i].end());
    }

    side.assign(n, -1);
    for (int i = 0; i < n; ++i) {
        if (side[i] != -1) continue;
        queue<int> q;
        side[i] = 0;
        q.push(i);
        while (!q.empty()) {
            int v = q.front(); q.pop();
            for (int to : g[v]) {
                if (side[to] == -1) {
                    side[to] = side[v] ^ 1;
                    q.push(to);
                }
            }
        }
    }

    mt.assign(n, -1);
    used.assign(n, 0);

    for (int v = 0; v < n; ++v) {
        if (side[v] == 0) {
            ++timer;
            kuhn(v);
        }
    }

    vector<pair<int,int>> ans;
    for (int v = 0; v < n; ++v) {
        if (mt[v] != -1) {
            int a = mt[v] + 1;
```

```
74              int b = v + 1;
75              if (a > b) swap(a, b);
76              ans.emplace_back(a, b);
77          }
78      }
79
80      sort(ans.begin(), ans.end());
81
82      cout << ans.size() << "\n";
83      for (auto &e : ans) {
84          cout << e.first << " " << e.second << "\n";
85      }
86
87      return 0;
88  }
```

Описание работы программы

Программа читает граф, строит списки смежности и сортирует их, чтобы обеспечить детерминированность обходов. Далее выполняется BFS-раскраска вершин в два цвета, восстанавливающая разбиение на доли (левая/правая). После этого запускается алгоритм Куна: для каждой вершины левой доли выполняется поиск увеличивающего пути в текущем паросочетании. Если путь найден, паросочетание увеличивается на одно ребро.

После завершения поиска из массива соответствий формируется список ребер, каждое ребро приводится к виду $(\min(u, v), \max(u, v))$, а затем весь список сортируется и выводится.

Дневник отладки

1. Проверена корректность формирования списков смежности для неориентированного графа (добавление в обе стороны).
2. Добавлена сортировка списков смежности для обеспечения однозначности результата.
3. Реализована BFS-раскраска для восстановления долей на произвольном двудольном графе с несколькими компонентами.
4. Оптимизирована отметка посещений в DFS (таймер), чтобы избежать $O(n)$ переинициализации на каждой итерации.

Оценка сложности

- Сортировка списков смежности: суммарно $O(m \log m)$ (точнее, сумма по вершинам).

- BFS-раскраска: $O(n + m)$.
- Алгоритм Куна: в худшем случае $O(|L| \cdot m)$, где $|L|$ — число вершин левой доли.
- Память: $O(n + m)$.

С учетом ограничений $m \leq 40000$ граф разреженный, поэтому на практике алгоритм работает эффективно.

Тест производительности

Сложность алгоритма (в худшем случае) оценивается как $O(|L| \cdot m)$ для DFS-версии алгоритма Куна, а также присутствуют затраты $O(n+m)$ на раскраску BFS и сортировку списков смежности. Ниже приведены экспериментальные замеры времени работы для $n = 110000$ при различных m. Для каждого m выполнено 9 прогонов на случайных двудольных графах, в график занесена медиана.

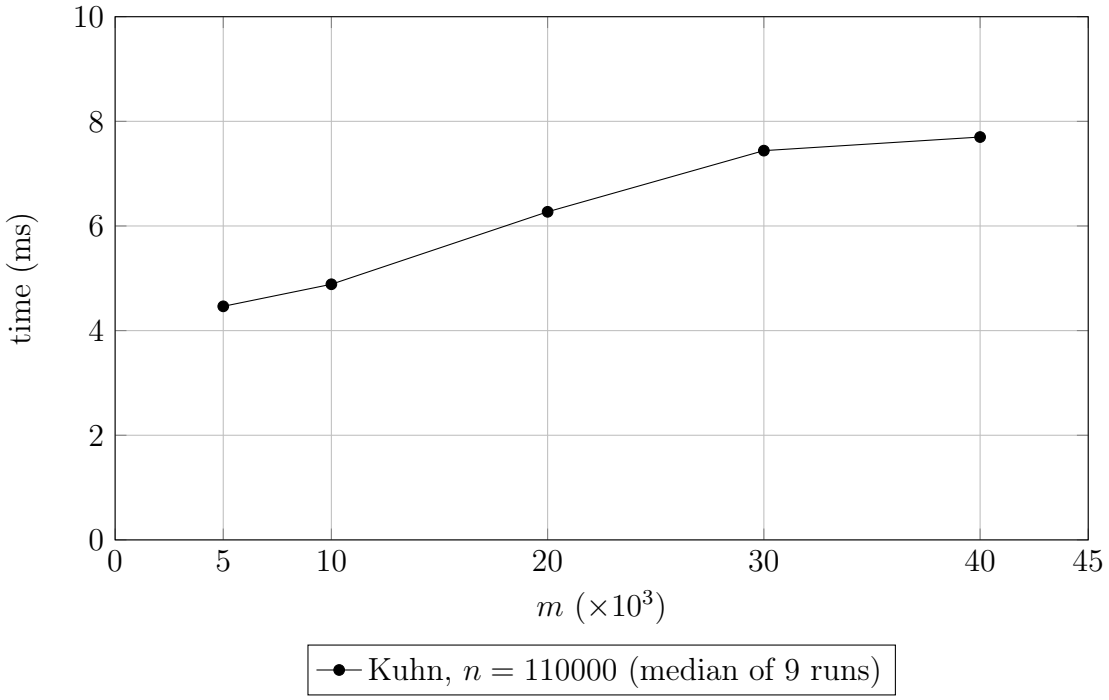

Рис. 1: Зависимость времени работы от числа ребер m при фиксированном $n = 110000$

Результаты тестирования

Программа успешно продемонстрировала корректные результаты для различных наборов входных данных. Это подтверждает правильность выбранного алгоритма и его эффективность в решении поставленной задачи.

Выводы

В ходе выполнения лабораторной работы была реализована программа для поиска максимального паросочетания в двудольном графе алгоритмом Куна. Были рассмотрены вопросы восстановления долей двудольного графа по исходному неориентированному описанию, а также обеспечена однозначность ответа путем сортировки списков смежности и выходных ребер. Реализация подтверждена тестами и соответствует формату вывода задачи.