Санкт-Петербургский Политехнический Университет

Высшая школа прикладной математики и вычислительной физики, ФизМех

01.03.02 Прикладная математика и информатика

# Лабораторная работа №2

# Дисциплина “Дискретная математика”

# Тема “Графы”

# Вариант “Алгоритм поиска наибольшего паросочетания”

### Поставленная задача

Реализовать алгоритм поиска наибольшего паросочетания в двудольном графе.

Оценить временную сложность, обосновать.

Определить лучший и худший тип графа для этого алгоритма.

### Используемый язык программирования

Python 3.10.6

### Описание алгоритма

#### Алгоритм

Алгоритм основан на идее поиска в глубину, для отметки вершин используется массив

x: array [1..p1] of bool

Паросочетание представлено массивом

M: array [1..p2] of 0..p1,

где M[u] = 0, если вершина u свободна в паросочетании, и M[u] = v, если ребро (v, u) входит в паросочетание.

Вход: двудольный граф G(V1, V2, E).

Выход: наибольшее паросочетание M: array [1..p2] of 0..p1.

for u from 1 to p2 do

M[u] : = 0 //паросочетание пусто

end for

//запускаем поиск в глубину из всех вершин первой доли

for v from 1 to p1 do

for w from 1 to p1 do

x[w] : = false //перед поиском всё не отмечено

end for

Aug(v) //поиск в глубину от вершины v

end for

#### Структура данных

Для представления графа в программе выбрана хэш-таблица: так как граф двудольный, вершины внутри его долей не связаны, следовательно, можно построить таблицу с ключами из вершин первой доли, каждому ключу соответствует список вершин второй доли.

Проверка наличия ребра (v, u) занимает O(1)

Проход по всем ребрам занимает O(p1 + p2)

#### Оценка сложности

Инициализация:

1. M: занимает O(p2)
2. x: занимает O(p1^2) (перезаполняем на каждом шаге цикла по p1)
3. Список связности: занимает O(p1\*p2)

Итого: O(p1^2)

Основная часть программы

1. Функция augmental: пусть количество ребер в графе равно d, в таком случае, временная сложность этой функции O(p1\*d) (DFS)
2. Подытожив, сложность основной части есть O(d\*p1^2)

Итоговая временная сложность: O(d\*p1^2)

Ясно, что с ростом числа ребер алгоритм работает хуже, следовательно, на полном графе алгоритм работает хуже всего и, наоборот, лучше всего на пустом графе.

### Пример работы алгоритма

Дан двудольный граф G = (V1, V2, E).

V1 = {A1, A2, A3}, V2 = {B1, B2, B3},

E = {(A1, B1), (A1, B2), (A1, B3), (A2, B1), (A2, B2), (A3, B1)}.

Паросочетания в процессе работы функции Aug перестраиваются следующим образом:

{(A1, B1)} → {(A1, B1), (A1, B2)} → {(A2, B1), (A1, B2)} → {(A2, B1), (A1, B2), (A1, B3)} →

→ {(A2, B1), (A2, B2), (A1, B3)} → {(A2, B2), (A1, B3), (A3, B1)}.

### 

### Вывод

1. Цель: Реализация алгоритма поиска наибольшего паросочетания в двудольном графе с использованием увеличивающих путей.
2. Метод: Алгоритм использует обход в глубину (DFS) для поиска увеличивающих путей, что позволяет увеличивать паросочетание, пока оно не станет максимальным.
3. Сложность: Временная сложность составляет O(d\*p1^2).