Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «вычислительная техника»

Пояснительная записка

К курсовому проектированию

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Реализация алгоритма поиска наибольшего паросочетания»

Выполнил:

Ст. гр. 23ВВВ4 Финаев М. А.

Приняла:

к.т.н. Юрова О. В.

Пенза 2024

**Содержание**

[Реферат 5](#_Toc184698454)

[Введение 6](#_Toc184698455)

[Постановка задачи 7](#_Toc184698456)

[Теоретическая часть задания 8](#_Toc184698457)

[Описание алгоритма программы 9](#_Toc184698458)

[Описание программы 12](#_Toc184698459)

[Тестирование 15](#_Toc184698460)

[Ручная проверка 17](#_Toc184698461)

[Заключение 18](#_Toc184698462)

[Список литературы 19](#_Toc184698463)

[Приложение А Листинг программы. 20](#_Toc184698464)

Реферат

**Отчет 16 стр. 9 рисунков.**

Алгоритм Хопкрофта-Карпа, теория графов, двудольный граф, паросочетание, матрица смежности.

Данная курсовая работа посвящена реализации алгоритма поиска максимального паросочетания в двудольном графе с использованием алгоритма Хопкрофта-Карпа. Задача максимального паросочетания является важной в теории графов и имеет множество практических приложений, например, в задачах распределения работы между работниками, поиска оптимальных путей и т.д. В курсовой работе описан алгоритм Хопкрофта-Карпа, а также приведена его реализация на языке программирования C. Основное внимание уделено разработке программы, ее тестированию и анализу производительности.

Введение

Алгоритм Хопкрофта — Карпа — алгоритм, принимающий на вход [двудольный граф](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D1%83%D0%B4%D0%BE%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84) и возвращающий максимальное [по мощности](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D1%89%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B0) [паросочетание](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%BE%D1%81%D0%BE%D1%87%D0%B5%D1%82%D0%B0%D0%BD%D0%B8%D0%B5), то есть наибольшее множество рёбер, таких что никакие два не имеют общую вершину.

Алгоритм был создан [Джоном Хопкрофтом](https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D0%BF%D0%BA%D1%80%D0%BE%D1%84%D1%82,_%D0%94%D0%B6%D0%BE%D0%BD) и [Ричардом Карпом](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BF,_%D0%A0%D0%B8%D1%87%D0%B0%D1%80%D0%B4_%D0%9C%D1%8D%D0%BD%D0%BD%D0%B8%D0%BD%D0%B3) в 1973 году. Как и ранее созданные алгоритмы (такие как [венгерский алгоритм](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BD%D0%B3%D0%B5%D1%80%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) и алгоритм из работы Эдмондса), алгоритм Хопкрофта — Карпа в цикле увеличивает паросочетание, находя увеличивающие пути ([цепи](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BF%D1%8C_%D0%B2_%D0%B3%D1%80%D0%B0%D1%84%D0%B5), рёбра которых попеременно принадлежат паросочетанию и не принадлежат ему, причём первая и последняя вершина не принадлежат паросочетанию; выполнив чередование паросочетания вдоль цепи, то есть убрав из паросочетания рёбра, бывшие в цепи, и добавив те, которых в нём не было, можно получить большее паросочетание). Вместо того чтобы находить один увеличивающий путь, алгоритм находит максимальное множество кратчайших увеличивающих путей. Та же идея используется в более сложных алгоритмах для поиска паросочетаний в недвудольных графах с тем же асимптотическим временем работы, как у алгоритма Хопкрофта — Карпа.

В качестве среды разработки была выбрана Microsoft Visual Studio 2022, язык программирования – С. Целью данной курсовой работы является разработка программы на языке С. Именно с его помощью реализуется алгоритм Хопкрофта – Карпа, осуществляющий нахождения наибольшего паросочетания в двудольном графе.

Постановка задачи

Требуется разработать программу, которая найдёт наибольшее паросочетание двудольного графа.

Задана матрица смежности двудольного графа, где одна часть вершин обозначена как U, а другая — как V. Требуется найти максимальное паросочетание, то есть максимальное количество рёбер, соединяющих вершины из двух частей графа так, чтобы каждое ребро соединяло различные вершины, и ни одна вершина не встречалась в нескольких рёбрах.

Устройства ввода – клавиатура и мышь.

Теоретическая часть задания

Граф G (рисунок 1) является двудольным графом с множеством вершин U и V соответственно.

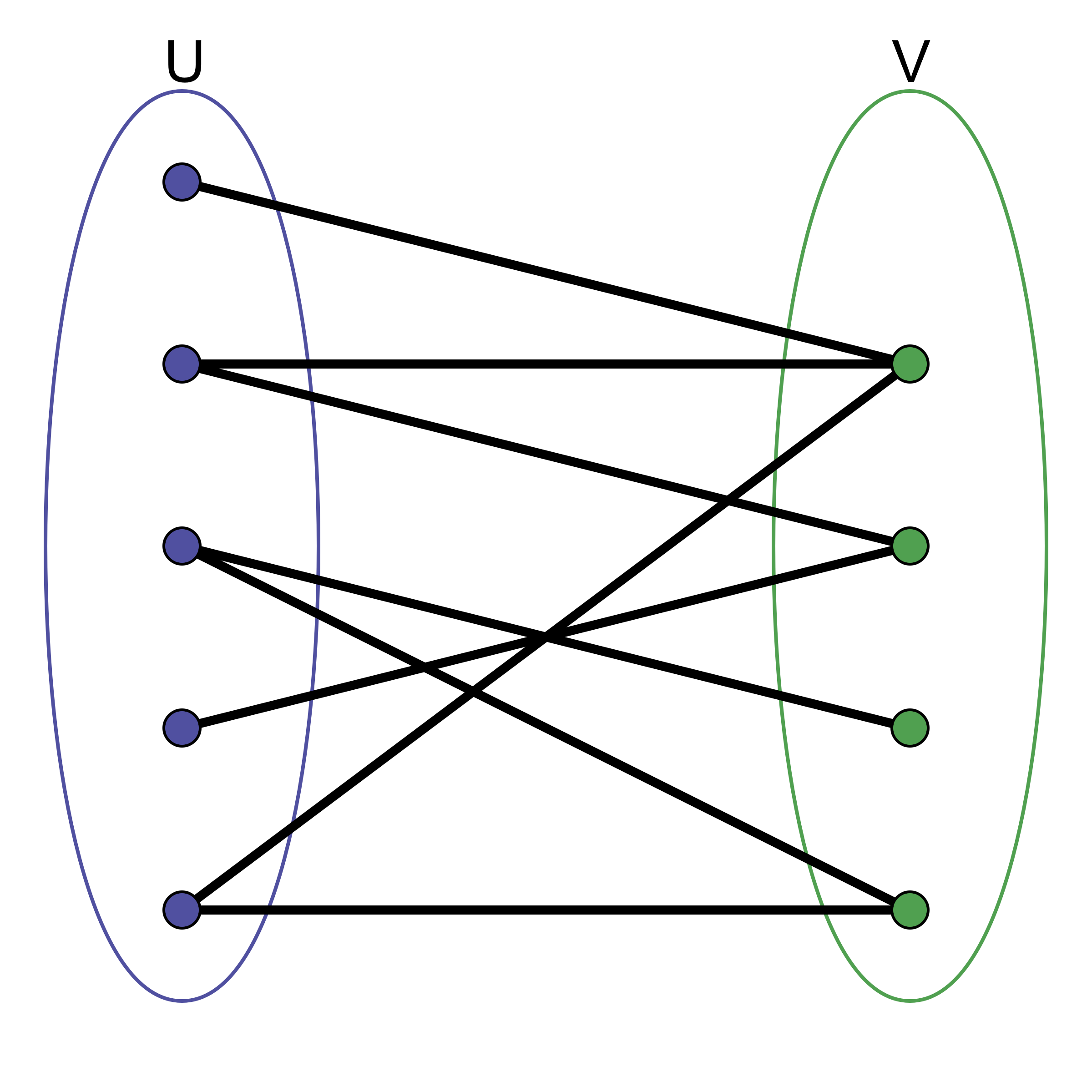


Рисунок 1 – Пример двудольного графа

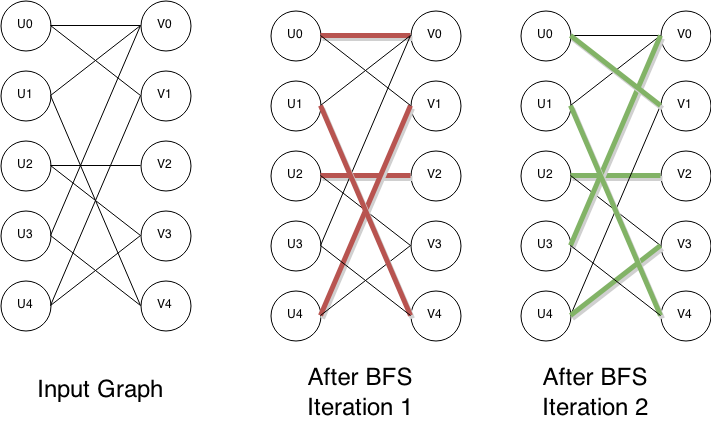
Алгоритм Хопкрофта-Карпа используется для нахождения максимального паросочетания в двудольных графах. Он работает за время O(√V \* E), где V — количество вершин в графе, а E — количество рёбер. Алгоритм использует два основных этапа: поиск в ширину (BFS) и поиск в глубину (DFS). BFS используется для нахождения максимального расстояния в графе, а DFS — для построения паросочетаний. Алгоритм оптимизирует процесс поиска паросочетаний, минимизируя количество лишних обходов графа.

Рисунок 2 – Пример работы алгоритма Хопкрофта-Карпа

Описание алгоритма программы

Для программной реализации понадобиться объявить такие методы, как bfs, dfs, hopcroftKarp.

Основной цикл программы:

* Пользователь вводит с клавиатуры количество вершин левой и правой частей графа.
* Пользователь заполняет матрицу смежности
* Выполняется алгоритм Хопкрофта-Карпа для нахождения наибольшего паросочетания
* Наибольшее количество паросочетаний данного графа выводится на экран
* Пользователь выбирает либо завершить программу, либо повторить ее с первого пункта

Константа MAX\_NODES задает максимальное количество вершин, с её помощью задается граф int graph, представленный матрицей смежности.

Int parU[MAX\_NODES], parV[MAX\_NODES] задают массивы пар, а массив int dist[MAX\_NODES] задаёт массив расстояний

Переменные int n, m хранят количество вершин в двух частях двудольного графа.

Ниже псевдокодом представлены методы bfs(), dfs(int u), hopcroftKarp() и main()

**bfs():**

1. Инициализировать очередь queue с максимальным количеством элементов
2. Установить front = 0 и rear = 0
3. Для каждого i от 0 до n-1:
4. Если pairU[i] равно -1:
5. dist[i] = 0
6. Добавить i в очередь queue
7. Иначе:
8. dist[i] = Бесконечность
9. dist[-1] = Бесконечность
10. Пока front < rear:
11. u = queue[front] // Извлечь элемент из очереди
12. front = front + 1
13. Если dist[u] < dist[-1]:
14. Для каждого v от 0 до m-1:
15. Если граф[u][v] существует и dist[pairV[v]] = Бесконечность:
16. dist[pairV[v]] = dist[u] + 1
17. Добавить pairV[v] в очередь queue
18. Вернуть (dist[-1] != Бесконечность)

**dfs(u):**

1. Если u не равно -1:
2. Для каждого v от 0 до m-1:
3. Если граф[u][v] существует и dist[pairV[v]] = dist[u] + 1:
4. Если dfs(pairV[v]) возвращает истину:
5. pairV[v] = u
6. pairU[u] = v
7. Вернуть истину
8. dist[u] = Бесконечность
9. Вернуть ложь
10. Вернуть истину

**hopcroftKarp():**

1. Для каждого i от 0 до n-1:
2. pairU[i] = -1
3. Для каждого i от 0 до m-1:
4. pairV[i] = -1
5. matching = 0
6. Пока bfs() возвращает истину:
7. Для каждого u от 0 до n-1:
8. Если pairU[u] = -1 и dfs(u) возвращает истину:
9. matching = matching + 1
10. Вернуть matching

**int main():**

1. Установить локаль "RUS"
2. cont = 1
3. Пока cont = 1:
4. Вывести "Введите количество вершин в левой части графа:"
5. Ввести n
6. Вывести "Введите количество вершин в правой части графа:"
7. Ввести m
8. Вывести "Введите матрицу смежности (где 1 — ребро, 0 — отсутствие ребра):"
9. Для i от 0 до n-1:
10. Для j от 0 до m-1:
11. Ввести graph[i][j]
12. maxMatching = hopcroftKarp()
13. Вывести "Размер наибольшего паросочетания:", maxMatching
14. Вывести "Ещё раз? (1 - да, 0 - нет)"
15. Ввести cont

Описание программы

Для написания данной программы использован язык программирования Си. Язык программирования Си – универсальный язык программирования, который завоевал особую популярность у программистов, благодаря сочетанию возможностей языков программирования высокого и низкого уровней.

Проект был создан в виде консольного приложения в терминале Windows (консольное приложение C++).

Данная программа является многомодульной поскольку состоит из нескольких методов: bfs, dfs, hopcroftkarp, int main.

Работа программы начинается с ввода количества вершин левой, а затем и правой частей двудольного графа. После этого пользователь заполняет матрицу смежности. Выполняется алгоритм поиска наибольшего паросочетания Хопкрофта-Карапа и результат выводится на экран

Вид кода, который осуществляет данную реализацию программы:

int bfs() {

int queue[MAX\_NODES], front = 0, rear = 0;

for (int i = 0; i < n; i++) {

if (pairU[i] == -1) {

dist[i] = 0;

queue[rear++] = i;

}

else {

dist[i] = INT\_MAX;

}

}

dist[-1] = INT\_MAX;

while (front < rear) {

int u = queue[front++];

if (dist[u] < dist[-1]) {

for (int v = 0; v < m; v++) {

if (graph[u][v] && dist[pairV[v]] == INT\_MAX) {

dist[pairV[v]] = dist[u] + 1;

queue[rear++] = pairV[v];

}

}

}

}

return dist[-1] != INT\_MAX;

}

int dfs(int u) {

if (u != -1) {

for (int v = 0; v < m; v++) {

if (graph[u][v] && dist[pairV[v]] == dist[u] + 1) {

if (dfs(pairV[v])) {

pairV[v] = u;

pairU[u] = v;

return 1;

}

}

}

dist[u] = INT\_MAX;

return 0;

}

return 1;

}

int hopcroftKarp() {

memset(pairU, -1, sizeof(pairU));

memset(pairV, -1, sizeof(pairV));

int matching = 0;

while (bfs()) {

for (int u = 0; u < n; u++) {

if (pairU[u] == -1 && dfs(u)) {

matching++;

}

}

}

return matching;

}

int main() {

setlocale(LC\_ALL, "RUS");

int cont = 1;

do

{

printf("Введите количество вершин в левой части графа: ");

scanf("%d", &n);

printf("Введите количество вершин в правой части графа: ");

scanf("%d", &m);

printf("Введите матрицу смежности (где 1 — ребро, 0 — отсутствие ребра):\n");

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

scanf("%d", &graph[i][j]);

}

}

int maxMatching = hopcroftKarp();

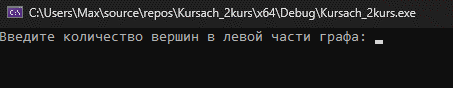
printf("Размер наибольшего паросочетания: %d\n\nЕщё раз?(1 - да, 0 - нет)", maxMatching);

scanf("%d", &cont);

} while (cont == 1);

return 0;

}

Далее показан начальный вид консоли при запуске пограммы:

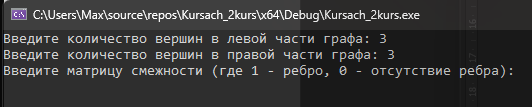
Рисунок 3 – Начальное вид консоли

Рисунок 4 – Ввод количества вершин правой и левой

частей двудольного графа

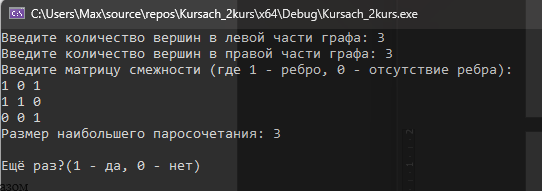


Рисунок 5 – Заполнение матрицы смежности и вывод

результата работы программы

Тестирование

Среда разработки Microsoft Visual Studio 2022 представляет все средства, необходимые при разработке и отладке многомодульной программы.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с вводом данных, поиском в глубину и ширину, а также работой алгоритма Хопкрофта-Карпа.

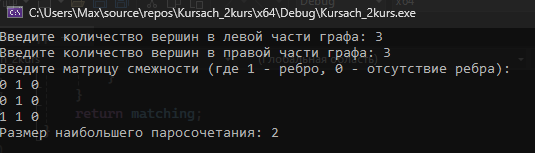
Ниже продемонстрирован результат тестирования программы при вводе пользователем количества вершин левой и правой частей двудольного графа, матрицы смежности вводимую с клавиатуры вручную.

Рисунок 6 – тестирование при вводе количества

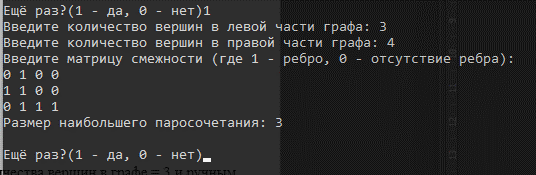
вершин левой и правой части графа по 3

Рисунок 7 – тестирование при вводе количества

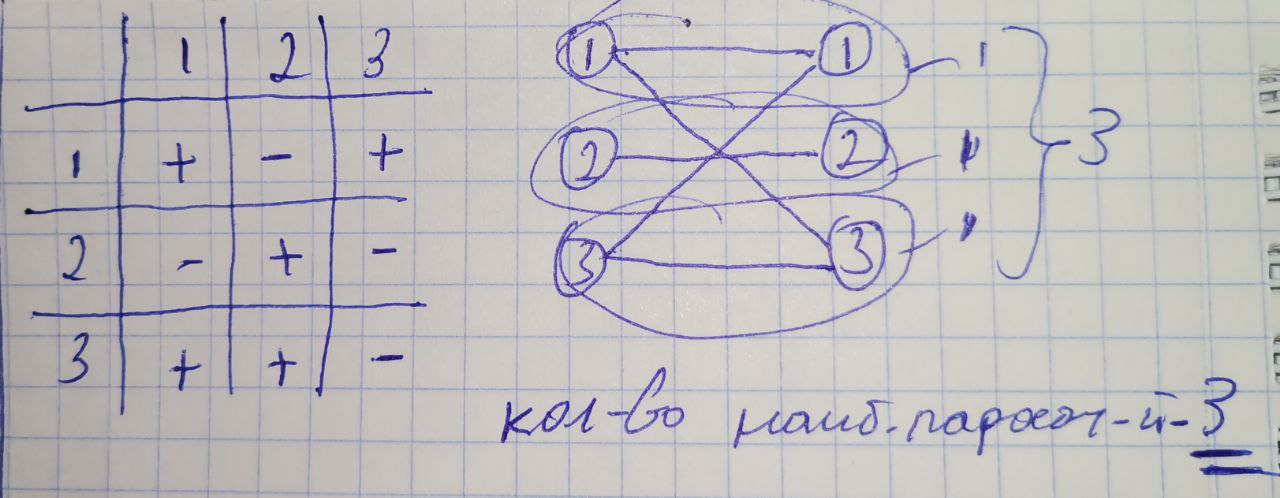
вершин левой части 3, правой - 4

Таблица 1 – описание поведения программы при тестировании

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Полученный результат |
| Запуск программы | Ожидание сообщения о вводе количества вершин левой части графа | Верно |
| Ввод количества вершин левой части графа | Ожидания сообщения о вводе количества вершин правой части графа | Верно |
| Вывод сообщения о вводе количества вершин правой части графа | Ожидание заполнения матицы смежности | Верно |
| Заполнение матрицы смежности | Вывод количества наибольших паросочетаний | Верно |
| Запрос на повторение программы | Повтор программы или её завершения в зависимости от выбора пользователя | Верно |

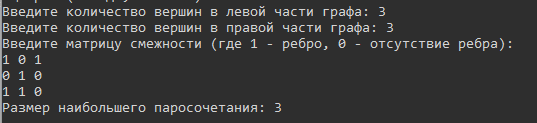
В результате тестирования было выявлено, что программа успешно проверяет данные на соответствие необходимым требованиям.

Ручная проверка

Рисунок 8 – Ручное нахождение наибольшего

паросочтания двудольного графа

Таким образом, можно сделать вывод, что программа работает верно.

Рисунок 9 – Сравнение ручной проверки с работой пограммы

Заключение

Таким образом, в процессе выполнения данной курсовой работы была разработана программа, реализующая алгоритм Хопкрофта-Карпа для нахождения наибольшего паросочетния двудольного графа, в Microsoft Visual Studio 2022.

При выполнении были получены навыки разработки программ, закреплены методы обхода в глубину и ширину, освоен принцип работы алгоритма Хопкрофта-Карпа. Улучшены навыки владения языком программирования Си.

В будущем программу можно улучшить, добавив возможность работы с более крупными графами, а также поддерживать динамическое изменение графа (добавление или удаление рёбер) в процессе работы программы. Также можно рассмотреть использование других алгоритмов для поиска паросочетаний в графах, таких как алгоритм Прима или алгоритм Эдмондса.

Список литературы

1. Хопкрофт Дж., Карп Р. "Алгоритмы построения паросочетаний в графах". 1973.
2. Кормен Т.Х., Лейзерсон Ч.Е., Ривест Р.Л., Штайн К. "Алгоритмы: построение и анализ". 2009.
3. Дейкстра Э. "Основы теории графов". 1982.

Приложение А Листинг программы.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <limits.h>

#include <locale.h>

#define MAX\_NODES 100 // Максимальное количество вершин

int graph[MAX\_NODES][MAX\_NODES]; // Граф, представленный матрицей смежности

int pairU[MAX\_NODES], pairV[MAX\_NODES]; // Массивы пар

int dist[MAX\_NODES]; // Массив расстояний

int n, m; // Количество вершин в двух частях двудольного графа

// Поиск в ширину (BFS)

int bfs() {

int queue[MAX\_NODES], front = 0, rear = 0;

for (int i = 0; i < n; i++) {

if (pairU[i] == -1) {

dist[i] = 0;

queue[rear++] = i;

}

else {

dist[i] = INT\_MAX;

}

}

dist[-1] = INT\_MAX;

while (front < rear) {

int u = queue[front++];

if (dist[u] < dist[-1]) {

for (int v = 0; v < m; v++) {

if (graph[u][v] && dist[pairV[v]] == INT\_MAX) {

dist[pairV[v]] = dist[u] + 1;

queue[rear++] = pairV[v];

}

}

}

}

return dist[-1] != INT\_MAX;

}

// Поиск в глубину (DFS)

int dfs(int u) {

if (u != -1) {

for (int v = 0; v < m; v++) {

if (graph[u][v] && dist[pairV[v]] == dist[u] + 1) {

if (dfs(pairV[v])) {

pairV[v] = u;

pairU[u] = v;

return 1;

}

}

}

dist[u] = INT\_MAX;

return 0;

}

return 1;

}

// Алгоритм Хопкрофта-Карпа

int hopcroftKarp() {

memset(pairU, -1, sizeof(pairU));

memset(pairV, -1, sizeof(pairV));

int matching = 0;

while (bfs()) {

for (int u = 0; u < n; u++) {

if (pairU[u] == -1 && dfs(u)) {

matching++;

}

}

}

return matching;

}

int main() {

setlocale(LC\_ALL, "RUS");

int cont = 1;

do

{

// Ввод данных

printf("Введите количество вершин в левой части графа: ");

scanf("%d", &n);

printf("Введите количество вершин в правой части графа: ");

scanf("%d", &m);

printf("Введите матрицу смежности (где 1 — ребро, 0 — отсутствие ребра):\n");

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

scanf("%d", &graph[i][j]);

}

}

// Вычисление максимального паросочетания

int maxMatching = hopcroftKarp();

// Вывод результата

printf("Размер наибольшего паросочетания: %d\n\nЕщё раз?(1 - да, 0 - нет)", maxMatching);

// Повторить?

scanf("%d", &cont);

} while (cont == 1);

return 0;

}