|  |
| --- |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"МИРЭА – Российский технологический университет"**  **РТУ МИРЭА** |

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-14 «Цифровые технологии обработки данных»

**ОТЧЕТ   
о выполнении лабораторной работы №3**

**«Алгоритмы на графах»**

**по дисциплине   
«Алгоритмы и структуры данных»**

**Вариант № 79**

Выполнил: студент 2 курса

группы \_\_\_\_\_\_\_\_\_\_\_

шифр \_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_

*(фио студента)*

Проверил: *\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Москва 2020 г.

**Вариант № 79.**

**Алгоритм:** определить k-связанность заданного неориентированного графа и  
вывести полученное число k на экран. (Граф называется k-связным,  
если между любой парой вершин v и w существует не менее k  
разных путей, таких, что, за исключением вершин v и w, ни одна из  
вершин, входящих в один путь, не входит ни в какой другой из этих  
путей)

**Способ представления графа:** Матрица смежности

**Теория о Графах.**

Граф — это математический объект, который состоит из точек и линий, которые их соединяют. Точки называют вершинами графа, а линии — ребрами. Граф, ребра которого имеют направления, называется ориентированным, если же ребра графа не имеют направления, то такой граф называется неориентированным.



Рисунок 1. Пример ориентированного графа

Матрица смежности — это вид представления графа в виде матрицы, когда пересечение столбцов и строк задаёт дуги. Используя матрицу смежности, можно задать вес дуг и ориентацию. Каждая строка и столбец матрицы соответствуют вершинам, номер строки соответствует вершине, из которой выходит дуга, а номер столбца - в какую входит дуга. Пример матрицы смежности графа, изображенного на рисунке 1, представлен на рисунке 2.



Рисунок 2. Пример матрицы смежности

Задача поиска числа к-связности графа сводится к задаче поиска максимального потока в графе. Так-как наш граф является неориентированным и ненагруженным, то установим пропускную способность для каждого ребра равную 1/1 и найдем поток между каждой парой вершин. После этого находим минимальный из максимальных потоков. Это и будет наше число k. Для поиска максимального потока воспользуемся алгоритмом Форда-Фалкерсона.

**Пример работы алгоритма Форда-Фалкерсона.**

* Допустим наш алгоритм нашел следующий путь из вершинывнашей *остаточной сети*, которая на момент начала, совпадает с исходным графом.

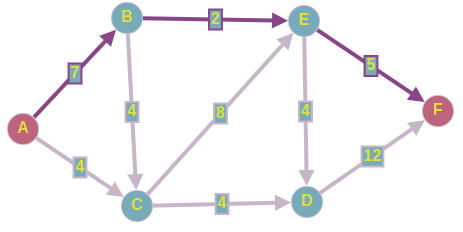


Рисунок 3. Начальная остаточная сеть

- Сколько потока можем провести по этому пути?  
- Больше 2 ед. мы никак не сможем пустить, пропускаем наш поток по этим рёбрам из *истока* в *сток.*

Получаем следующее:

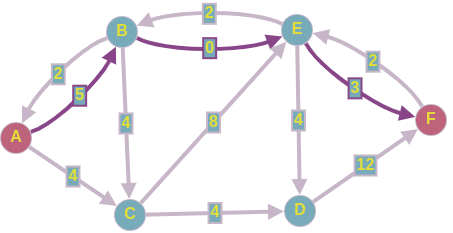


Рисунок 4. Остаточная сеть после 1-ой итерации

Рёбра с нулевым весом можно удалять. Таким образом, на первой итерации мы смогли увеличить максимальный поток на 2 ед.

Теперь дело за малым, остается всего лишь итерироваться до тех пор, пока существует путь из A в F.

* Допустим, на *2-ой итерации* мы нашли такой путь в нашей *остаточной сети*:

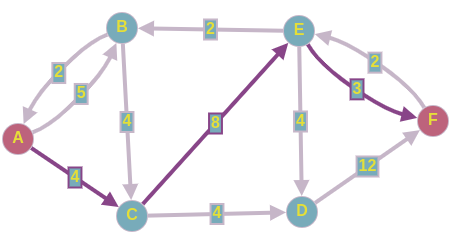


Рисунок 5. Остаточная сеть после 1-ой итерации

Пропускаем 3 ед**.** потока по этому пути, и перестраиваем *остаточную сеть.*   
Получаем следующее:

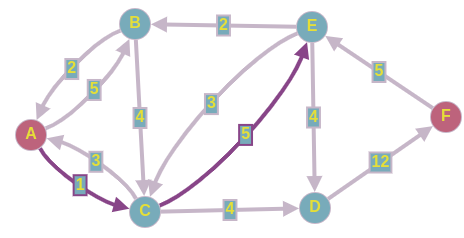


Рисунок 6. Остаточная сеть после 2-ой итерации

На 3-ей итерации нашли такой путь в нашей модифицированной *остаточной сети:*

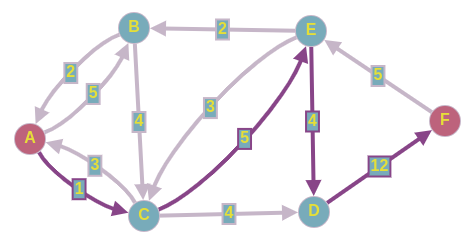


Рисунок 7. Остаточная сеть после 2-ой итерации.

Пускаем 1 ед.потока по этому пути и перестраиваем нашу *остаточную сеть.*  
Получим следующую картину:

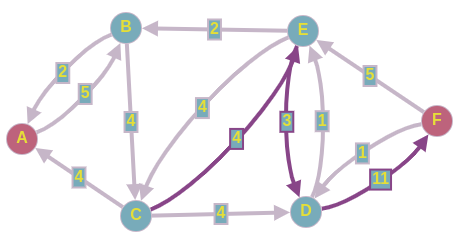


Рисунок 8. Остаточная сеть после 3-ей итерации

На 4-ой итерации находим следующий путь в *остаточной сети:*

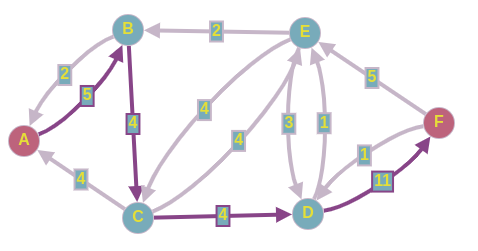


Рисунок 9. Остаточная сеть после 3-ей итерации

Пускаем 4 ед.потока по этому пути и перестраиваем нашу *остаточную сеть.*  
Получим следующую картину:

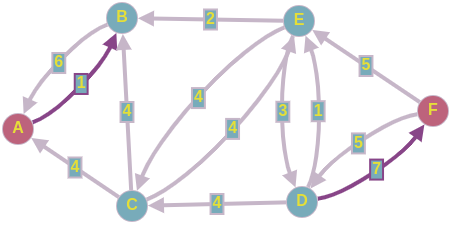


Рисунок 10. Итоговая остаточная сеть

На этом этапе наш алгоритм прекратит выполнение из-за того, что пути из *истока* в *сток* не существует. И ответом к поставленной задаче будет служить сумма потоков всех найденных *увеличивающихся путей.****Ответ:*** 10 ед.

**Листинг программы.**

def find\_graph\_k(*matrix*: list[list[int]]):

    graph = []

    dim = len(*matrix*)

    def bfs(*s*, *t*, *parent*):

        visited = [False] \* dim

        queue = []

        queue.append(*s*)

        visited[*s*] = True

        while queue:

            u = queue.pop(0)

            for ind, val in enumerate(graph[u]):

                if visited[ind] == False and val > 0:

                    queue.append(ind)

                    visited[ind] = True

*parent*[ind] = u

                    if ind == *t*:

                        return True

        return False

    def ford\_fulkerson(*source*, *sink*):

        parent = [-1] \* dim

        max\_flow = 0

        while bfs(*source*, *sink*, parent):

            path\_flow = float("Inf")

            s = *sink*

            while(s != *source*):

                path\_flow = min(path\_flow, graph[parent[s]][s])

                s = parent[s]

            max\_flow += path\_flow

            v = *sink*

            while(v != *source*):

                u = parent[v]

                graph[u][v] -= path\_flow

                graph[v][u] += path\_flow

                v = parent[v]

        return max\_flow

    k = None

    for i in range(len(*matrix*)):

        for j in range(len(*matrix*)):

            if i == j:

                continue

            graph = [row[:] for row in *matrix*]

            local\_k = ford\_fulkerson(i, j)

            if k is None or local\_k < k:

                k = local\_k

    return k

with open("graph\_task79.txt", "r") as f:

    matrix = list(map(list, [map(int, line.split()) for line in f.readlines()]))

print(f"K-связность графа равна: {find\_graph\_k(matrix)}")

**Скриншот работы программы:**



Рисунок 12. Пример работы программы

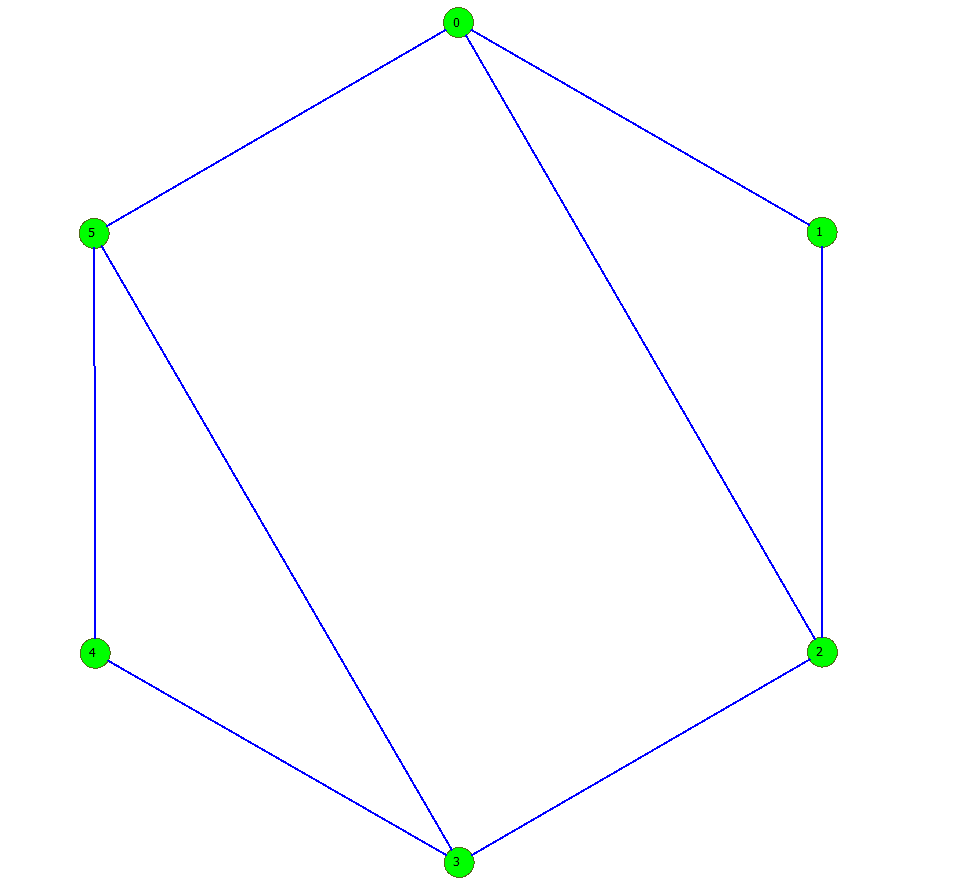


Рисунок 11. Пример ненагруженного неориентированного графа

**Выводы.**

В результате выполнения данной работы был реализован алгоритм Форда-Фалкерсона для поиска максимального потока в ориентированном нагруженном графе. На основе этого алгоритма был реализован поиск числа k-связности графа.

**Литература:**

1. Алгоритмы: построение и анализ. Кормен Т.Х., Лейзерсон Ч.И.
2. <https://en.wikipedia.org/wiki/Ford%E2%80%93Fulkerson_algorithm>
3. <https://en.wikipedia.org/wiki/K-edge-connected_graph>