Министерство цифрового развития, связи и массовых коммуникаций РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский государственный университет телекоммуникаций и
информатики»
(СибГУТИ)

Кафедра ВС СибГУТИ

Отчёт
По курсовой работе
«Максимальный поток»

Выполнили студенты группы МГ-211:

Бурдуковский И.А.

Стояк Ю.К.

Проверил профессор кафедры ПМиК:

Рубан А.А.

Новосибирск 2023

Оглавление

Задание

Реализовать алгоритм Форда - Фалкерсона для решения задачи о максимальном потоке.

Описание

Дан граф G(V, E) с пропускной способностью $c(u, v)$и потоком $f(u, v) = 0$ для рёбер из u в v. Необходимо найти максимальный поток из источника s в сток t. На каждом шаге алгоритма действуют те же условия, что и для всех потоков:

- $0 \leq f(u, v) \leq c(u, v)$. Поток из u в v не превосходит пропускной способности.
- $f(u, v) = -f(u, v)$.
- $\sum_V f(u, v) = 0 \leftrightarrow f_{in}(u) = f_{out}(u)$ для всех узлов u, кроме s и t. Поток не изменяется при прохождении через узел.

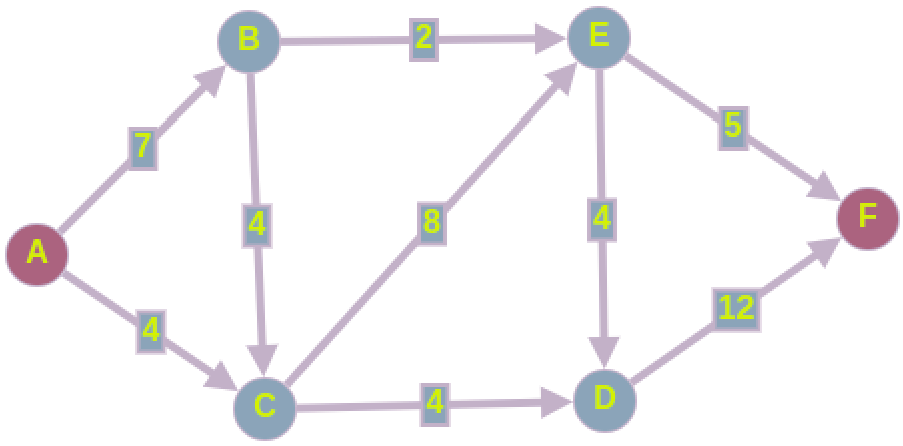
Остаточная сеть $G_f(V, E_f)$ — сеть с пропускной способностью$c_f(u, v) = c(u, v) - f(u, v) + f(v, u)$ и без потока.

Вход Граф G с пропускной способностью C, источник s и сток t
Выход Максимальный поток f из s в t

1. $f(u, v) \coloneqq 0$ для всех рёбер (u, v)
2. Пока есть путь p из s в t вG_f, такой что $c_f(u, v) > 0$ для всех рёбер $(u, v) \in p$:
    1. Найти $c_f(p) = \min\{c_f(u, v) | (u, v) \in p\}$
    2. Для каждого ребра $(u, v) \in p$
        1. $f(u, v) \coloneqq f(u, v) + c_f(p)$
        2. $f(v, u) \coloneqq f(u, v) - c_f(p)$
        3. $c_f(u, v) \coloneqq c(u, v) - f(u, v)$
        4. $c_f(v, u) \coloneqq f(u, v)$

Путь может быть найден, например, поиском в ширину (алгоритм Эдмондса — Карпа) или поиском в глубину/

Результат работы программы




```
Максимальный поток: 10
Найденные пути:
[0, 1, 4, 5]
[0, 2, 3, 5]
[0, 1, 2, 4, 5]
[0, 1, 2, 4, 3, 5]
```

Листинг

```
import pandas as pd


class Graph:
    def __init__(self, graph, source, sink):
        self.graph = graph
        self.vertex_count = len(graph)
        self.start = source
        self.end = sink
        self.pathes = []


def find_path_BFS(g, parent):
    visited = [False] * g.vertex_count
    queue = [g.start]
    visited[g.start] = True

    while queue:
        u = queue.pop(0)
        # print(g.graph[u])
        for ind, val in enumerate(g.graph[u]):
            if visited[ind] == False and val > 0:
                queue.append(ind)
                visited[ind] = True
                parent[ind] = u

    if visited[g.end]:
        return True
    else:
        return False


def ford_fulkerson(g):
    parent = [-1] * g.vertex_count
    max_flow = 0

    while find_path_BFS(g, parent):

        path_history = [g.end]

        path_flow = float("Inf")
        s = g.end
        while s != source:
            path_flow = min(path_flow, g.graph[parent[s]][s])
            s = parent[s]
            path_history = [s] + path_history
            # print(s)

        g.pathes.append(path_history)

        max_flow += path_flow

        v = g.end
        while (v != source):
            u = parent[v]
            g.graph[u][v] -= path_flow
            g.graph[v][u] += path_flow
            v = parent[v]

    return max_flow


if __name__ == "__main__":
    graph = []

    with open('ford_falkerson_flow3.in', 'r') as file:
        for line in file.readlines():
            rows = list(map(int, line.split()))
            graph.append(rows)

    print("Граф:")
    print(pd.DataFrame(graph), "\n")

    source = 0
    sink = 5
```

```
g = Graph(graph, source, sink)

print("Максимальный поток: %d " % ford_fulkerson(g))
print("Найденные пути: ")
for path in g.pathes:
    print(path)

print("Граф:")
print(pd.DataFrame(graph), "\n")
```