МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ
«МИСиС»

Институт информационных технологий и компьютерных наук

Кафедра автоматизированного проектирования и дизайна

Курсовая работа

по дисциплине
«Алгоритмы дискретной математики»

на тему
«Алгоритм проталкивания предпотока»

Выполнил(а) студент(ка): Гузев В.Н.
Группа: БИВТ-18-1
Проверил(а): доц., к.т.н. Калитин Д.В.
Оценка:

Москва
2020

Алгоритм проталкивания предпотока

Алгоритм проталкивания предпотока (англ. Push-Relabel algorithm) решает задачу поиска максимального потока в сети.

Сеть (англ. flow network) $G = (V, E)$ — это ориентированный граф, в котором каждое ребро $(u, v) \in E$ имеет пропускную способность $c(u, v) > 0$. Если $(u, v) \notin E$, предполагается, что $c(u, v) = 0$. В транспортной сети выделяются вершины S (исток) и T (сток).

Потоком (англ. flow) f в G является действительная функция $f: V \times V \to R$, удовлетворяющая условиям:

1. $f(u, v) = -f(v, u)$ (антисимметричность);
2. $f(u, v) \vee\leq c(u, v)$ (ограничение пропускной способности), если ребра нет, то $f(u, v) = 0$;
3. $\sum_{v} f(u, v) = 0$ для всех вершин u, кроме s и t (закон сохранения потока);

Величина потока f определяется как $f \vee \sum_{v \in V} f(s, v)$.

Интерпретировать транспортную сеть можно как трубопровод, по которому поступает количество воды, ограниченное пропускной способностью.

Предпотоком (англ. preflow) называется функция $f: V \times V \to R$, удовлетворяющая следующим свойствам:

1. $f(u, v) = -f(v, u)$ (антисимметричность);
2. $f(u, v) \vee\leq c(u, v)$ (ограничение пропускной способности);
3. $\forall u \in V \setminus \{s, t\} \quad \sum_{v} f(u, v) \geq 0$ (ослабленное условие сохранения потока);

Избыточный поток (англ. excess flow), входящий в вершину u — это величина $e(u) = \sum_{v \in V} f(v, u)$.

Алгоритм проталкивания предпотока является альтернативой алгоритма Форда-Фалкерсона. Преимуществом является лучшая эффективность работы.

Описание алгоритма

Сперва происходит инициализация — для всех рёбер, исходящих из s, предпоток устанавливается в максимальное значение пропускной способности: $f\big((s,u)\big) = c\big((s,u)\big)$. Для остальных рёбер это значение выставляется в ноль. Далее исток помечается как $h(s) = V \vee$, где V — количество вершин, остальные вершины помечаются нулём.

Далее весь алгоритм построен на двух операциях — push (проталкивание) и relabel (подъём / переразметка).

С помощью операции push производится «проталкивание» потока из вершины u в вершину v. Дозволено «проталкивать» поток только в том случае, если $h(u) = h(v) + 1$. Производится проталкивание потока $min\Big(x(u), c\big((u,v)\big) - f\big((u,v)\big)\Big)$.

Если вершина имеет избыток, но нету возможности от него избавиться, совершается «подъём» relabel, пока такая возможность не появиться.

Суммируя вышесказанное:

1. Инициализация предпотока и разметки;
2. Пока возможны операции push и relabel, они выполняются;
3. Предпоток становится максимальным потоком и возвращается как результат.

Листинг программы

```
from collections import deque
import math
n = 0
capacity = list(list())
flow = list(list())
height = list()
excess = list()
seen = list()
excess_vertices = deque()


def push(u, v):
    global flow
    global excess
    global excess_vertices
    d = min(excess[u], capacity[u][v] - flow[u][v])
    flow[u][v] += d
    flow[v][u] -= d
    excess[u] -= d
    excess[v] += d
    if d and excess[v] == d:
        excess_vertices.append(v)


def relabel(u):
    global height
    d = math.inf
    for i in range(n):
        if capacity[u][i] - flow[u][i] > 0:
            d = min(d, height[i])
    if d < math.inf:
        height[u] = d + 1


def discharge(u):
    global seen
    while excess[u] > 0:
        if seen[u] < n:
            v = seen[u]
            if capacity[u][v] - flow[u][v] > 0 and height[u] > height[v]:
                push(u, v)
```

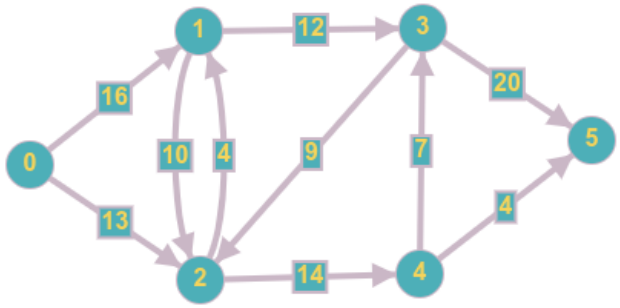
```
            else:
                seen[u] += 1
        else:
            relabel(u)
            seen[u] = 0


def max_flow(s, t):
    global height
    global flow
    global excess
    global seen
    height = [0 for i in range(n)]
    height[s] = n
    flow = [[0 for j in range(n)] for i in range(n)]
    excess = [0 for i in range(n)]
    excess[s] = math.inf
    for i in range(n):
        if i != s:
            push(s, i)
    seen = [0] * n
    while len(excess_vertices) != 0:
        u = excess_vertices.popleft()
        if u != s and u != t:
            discharge(u)
    max_flow = 0
    for i in range(n):
        max_flow += flow[i][t]
    return max_flow
```

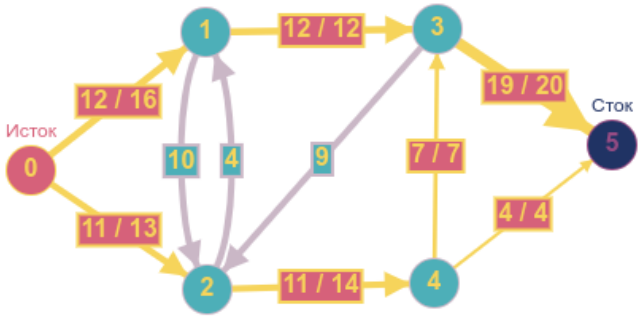
Результаты работы программы

Для демонстрации работы программы воспользуемся классическим примером из учебника CLRS.

Исходный граф:



Граф с вычисленным максимальным потоком:



В программе граф представлен матрицей весов. Результат вычислений совпал с предполагаемым значением, максимальный поток в исходном графе – 23.

```
if __name__ == "__main__":
    n = 6
    capacity = [[0, 16, 13, 0, 0, 0],
                [0, 0, 10, 12, 0, 0],
                [0, 4, 0, 0, 14, 0],
                [0, 0, 9, 0, 0, 20],
                [0, 0, 0, 7, 0, 4],
                [0, 0, 0, 0, 0, 0]]
```

```
print(max_flow(0, 5))
```

Вывод

В процессе работы был изучен и описан алгоритм проталкивания предпотока. Также была разработана реализация данного алгоритма с использованием языка Python.