--ФИО--

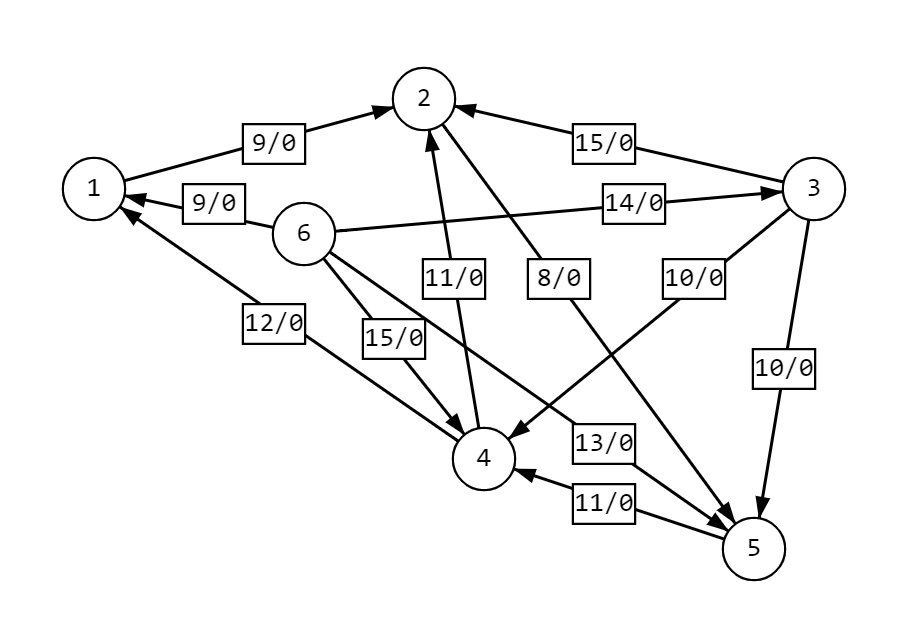
Лабораторная работа №4

|  |  |  |
| --- | --- | --- |
| Интервал весов | G1 | G2 |
| 8-15 | (6;12) | (6;12) |

В паре (i,j) i − число вершин, j − число дуг. G1 − для задачи о максимальном потоке, G2 − для задачи о кратчайшем пути. Интервалы весов указывают границы изменения пропускных способностей и длин дуг.

**Алгоритм Форда-Фалкерсона**

Граф:



Код программы:

import math

def get\_max\_vertex(k, V, S):

m = 0 # наименьшее допустимое значение

v = -1

for i, w in enumerate(V[k]):

if i in S:

continue

if w[2] == 1: # движение по стрелке

if m < w[0]:

m = w[0]

v = i

else: # движение против стрелки

if m < w[1]:

m = w[1]

v = i

return v

def get\_max\_flow(T):

w = [x[0] for x in T]

return min(\*w)

def updateV(V, T, f):

for t in T:

if t[1] == -1: # это исток

continue

sgn = V[t[2]][t[1]][2] # направление движения

# меняем веса в таблице для (i,j) и (j,i)

V[t[1]][t[2]][0] -= f \* sgn

V[t[1]][t[2]][1] += f \* sgn

V[t[2]][t[1]][0] -= f \* sgn

V[t[2]][t[1]][1] += f \* sgn

V = [[[0,0,1], [9,0,1], [0,0,1], [12,0,-1], [0,0,1], [9,0,-1]],

[[9,0,-1], [0,0,1], [15,0,-1], [11,0,-1], [8,0,1], [0,0,1]],

[[0,0,1], [15,0,1], [0,0,1], [10,0,1], [10,0,1], [14,0,-1]],

[[12,0,1], [11,0,1], [10,0,-1], [0,0,1], [11,0,-1], [15,0,-1]],

[[0,0,1], [8,0,-1], [10,0,-1], [11,0,1], [0,0,1], [13,0,-1]],

[[9,0,1], [0,0,1], [14,0,1], [15,0,1], [13,0,1], [0,0,1]]

]

N = len(V) # число вершин в графе

init = 5 # вершина истока (нумерация с нуля)

end = 1 # вершина стока

Tinit = (math.inf, -1, init) # первая метка маршруто (a, from, vertex)

f = [] # максимальные потоки найденных маршрутов

j = init

while j != -1:

k = init # стартовая вершина (нумерация с нуля)

T = [Tinit] # метки маршрута

S = {init} # множество просмотренных вершин

while k != end: # пока не дошли до стока

j = get\_max\_vertex(k, V, S) # выбираем вершину с наибольшей пропускной способностью

if j == -1: # если следующих вершин нет

if k == init: # и мы на истоке, то

break # завершаем поиск маршрутов

else: # иначе, переходим к предыдущей вершине

k = T.pop()[2]

continue

c = V[k][j][0] if V[k][j][2] == 1 else V[k][j][1] # определяем текущий поток

T.append((c, j, k)) # добавляем метку маршрута

S.add(j) # запоминаем вершину как просмотренную

if j == end: # если дошди до стока

f.append(get\_max\_flow(T)) # находим максимальную пропускную способность маршрута

updateV(V, T, f[-1]) # обновляем веса дуг

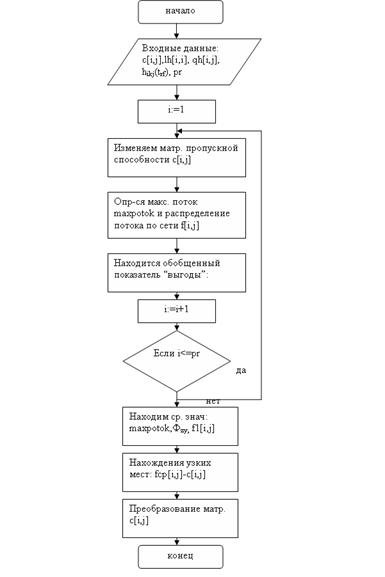
break

k = j

F = sum(f)

print(f"Максимальный поток равен: {F}")

Блок-схема:



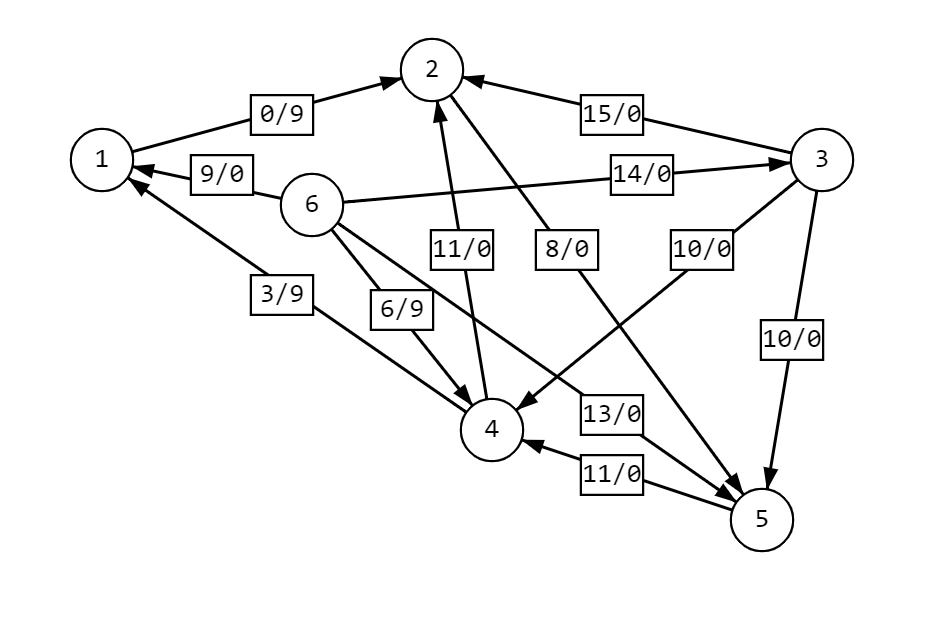
Описание:

Алгоритм Форда-Фалкерсона предназначен для решения задачи о максимальном потоке в сети.

Изначально мы выбираем вершину, которая будет являться истоком и выбираем ребро, смежное с вершиной, такое, что у него будет максимальный поток. Продолжаем данное действие, пока не дойдем до стока( в этом случае записываем максимальный поток на данном маршруте, равный минимальному потоку ребер, через которые мы прошли, а также изменяем потоки в графе), или же пока не дойдем до вершины, из которой максимальный поток будет равен 0 (в этом случае идем в предыдущую вершину и выбираем другое смежное ребро, если же это невозможно, то есть «тупиковая» вершина – исток, то завершаем поиск маршрутов и суммируем максимальные потоки на всех маршрутах).

Ход решения:

V6-исток, v2-сток

1. V6-текущая вершина

s=(v1;v3;v4;v5)

fmax=f(9;14;15;13)=15

v4-текущая вершина

s=(v1;v2)

fmax=f(12, 11)=12

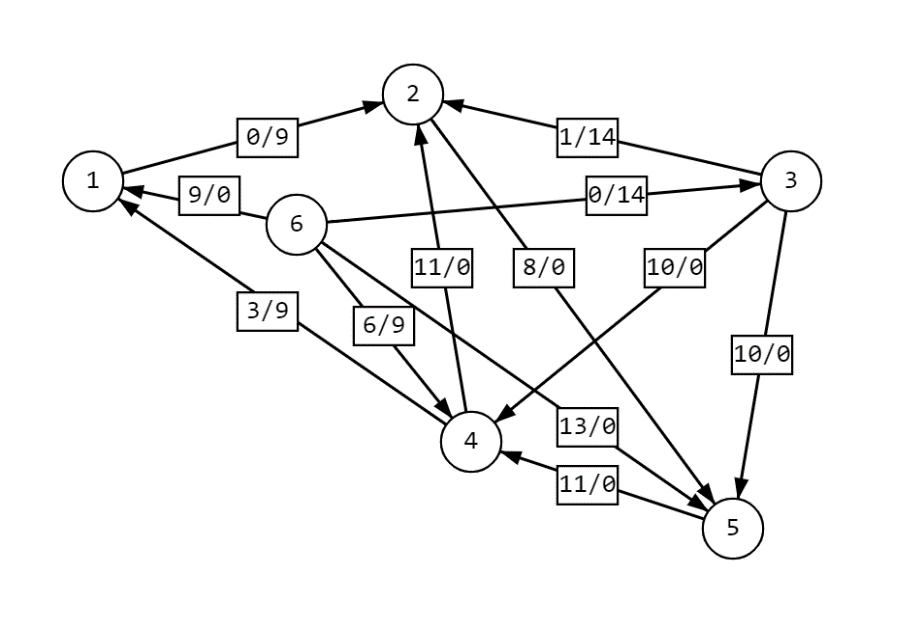
v1-текущая вершина

s=(v2)

fmax=f(9)=9

v2-текущая вершина (сток)

f1=min(∞;15;12;9)=9

1. v6-текущая вершина

s=(v1;v3;v4;v5)

fmax=f(9;14;6;13)=14

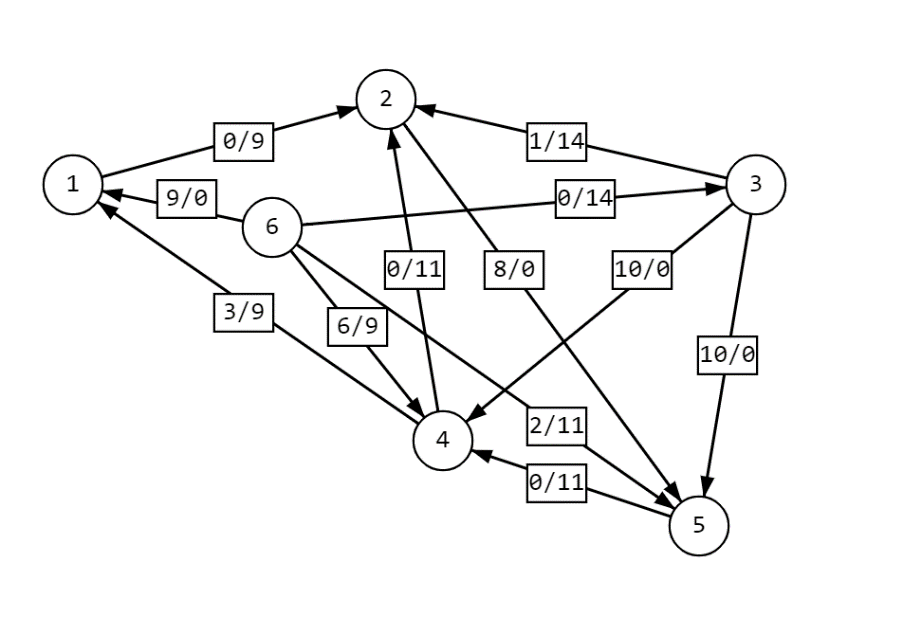
v3-текущая вершина

s=(v2;v4;v5)

fmax=f(15;10;10)=15

v2-текущая вершина (сток)

f2=min(∞;14;15)=14

1. v6-текущая вершина

s=(v1;v4;v5)

fmax=f(9;6;13)=13

v5-текущая вершина

s=(v4)

fmax=f(11)=11

v4-текущая вершина

s=(v1;v2)

fmax=f(3;11)=11

v2-текущая вершина (сток)

f3=min(∞;13;11;11)=11

4. V6-текущая вершина

s=(v1;v4;v5)

fmax=f(9;6;2)=9

v1-текущая вершина

s=(v4)

fmax=f(9)=9

v4-текущая вершина

s=(v5)

fmax=f(11)=11

v5-текущая вершина

s=()

откат назад в v4

v4-текущая вершина

s=()

откат назад в v1

v1-текущая вершина

s=()

откат назад в v6

v6-текущая вершина

s=(v4;v5)

fmax=f(6;2)=6

v4-текущая вершина

s=(v1;v5)

fmax=f(3;11)=11

v5-текущая вершина

s=()

откат назад в v4

v4-текущая вершина

s=(v1)

fmax=f(3)=3

v1-текущая вершина

s=()

откат назад в v4

v4-текущая вершина

s=()

откат назад в v6

v6-текущая вершина

s=(v5)

fmax=f(3)=3

v5-текущая вершина

s=()

откат назад в v6

v6-текущая вершина

s=()

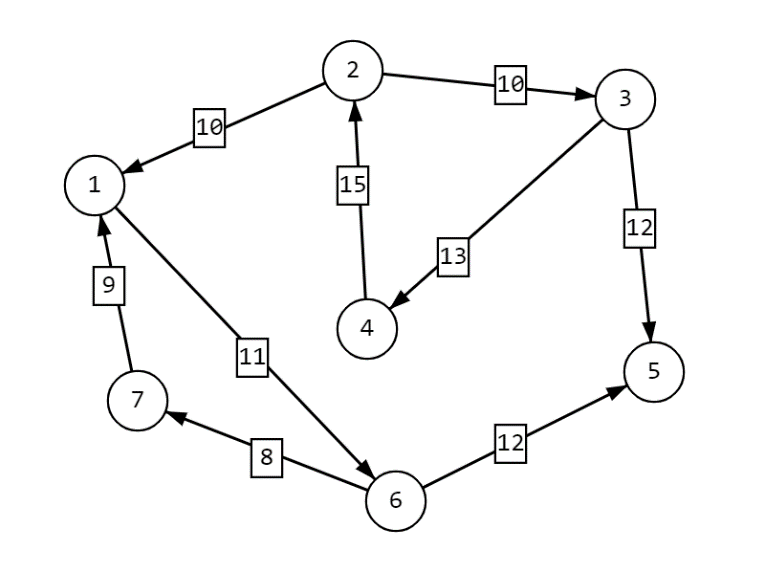
откат назад невозможен

*fmax=f1+ f2+f3=9+14+11=34*

Таким образом, максимальный поток равен 34.

**Алгоритм Дейкстры**

Граф:



Код программы:

import math

def get\_link\_v(v,D):

for i, weight in enumerate(D[v]):

if weight>0:

yield i

def arg\_min(T, S):

amin = -1

m = math.inf # максимальное значение

for i, t in enumerate(T):

if t < m and i not in S:

m = t

amin = i

return amin

D = ((0, 0, 0, 0, 0, 11, 0),

(10, 0, 10, 0, 0, 0, 0),

(0, 0, 0, 13, 12, 0, 0),

(0, 15, 0, 0, 0, 0, 0),

(0, 0, 0, 0, 0, 0, 0),

(0, 0, 0, 0, 12, 0, 8),

(9, 0, 0, 0, 0, 0, 0))

N = len(D) # число вершин в графе

T = [math.inf]\*N # последняя строка таблицы

v = 3 # стартовая вершина (нумерация с нуля)

S = {v} # просмотренные вершины

T[v] = 0 # нулевой вес для стартовой вершины

while v != -1: # цикл, пока не просмотрим все вершины

for j in get\_link\_v(v, D): # перебираем все связанные вершины с вершиной v

if j not in S: # если вершина еще не просмотрена

w = T[v] + D[v][j]

if w < T[j]:

T[j] = w

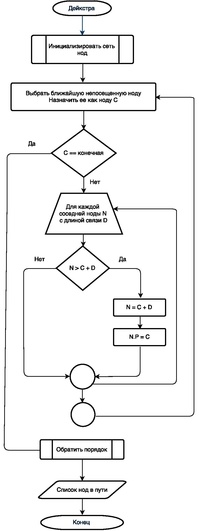
v = arg\_min(T, S) # выбираем следующий узел с наименьшим весом

if v >= 0: # выбрана очередная вершина

S.add(v) # добавляем новую вершину в рассмотрение

print(T)

Блок-схема:



Описание:

Алгоритм Дейкстры предназначен для решения задачи о поиске кратчайшего пути в графе от вершины-истока до всех остальных вершин в графе.

Алгоритм строит множество вершин, для которых мы знаем кратчайшее расстояние, на каждой итерации в множество добавляется та из оставшихся вершин, расстояние до которой от истока меньше, чем для других. Когда множество будет содержать все вершины орграфа, будут найдены кратчайшие пути от истока до всех вершин графа.

Ход решения:

M(i)=(0;+) для i=1,2,…,n

M(4)=(4;0)

Q={4}

λ2=λ0+(0;2)=15

M(2)=(0;15)

Q={0;2}

λ1= λ2+(2;1)=25

λ3= λ2+(2;3)=25

M(1)=(2;25)

Q={0;2;1}

λ6= λ1+(1;6)=36

M(6)=(1;36)

M(3)=(2;25)

Q={0;2;3}

λ5= λ3+(3;5)=37

M(5)=(3;37)

M(6)=(1;36)

Q={0;2;1;6}

λ5= λ6+(6;5)=48

λ7= λ6+(6;7)=44

M(7)=(6;44)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 |  |  |  |  |  |  |  |
| 1 |  | 15 |  |  |  |  |  |
| 2 | 25 |  | 25 |  |  |  |  |
| 3 |  |  | 25 |  |  | 36 |  |
| 4 |  |  |  |  | 37 | 36 |  |
| 5 |  |  |  |  | 37 |  | 44 |

Вывод:

В ходе данной работы были изучены алгоритмы Дейкстры и Форд-Фалкерсона.