Ministerul Educaţiei, Culturii și Cercetării
Universitatea Tehnică a Moldovei

Departamentul Ingineria Software și Automatică

RAPORT

Lucrarea de laborator nr. 5
la MATEMATICI SPECIALE

Tema: ALGORITMI DE DETERMINARE A FLUXULUI MAXIM

A efectuat: Cătălin Pleșu
st. gr. TI-206

A verificat: Lisnic Inga

Chișinău – 2021

Cuprins

1 Scopul și obiectivele lucrării

- Programarea algoritmului Ford-Fulkerson pentru determinarea fluxului maxim într-o reţea de transport.
- Studierea noţiunilor de bază leagate de reţelele de transport;

2 Sarcina lucrării

1. Realizaţi procedura introducerii unei reţele de transport cu posibilităţi de verificare a corectitudinii datelor introduse;
2. În conformitate cu algoritmul Ford-Fulkerson elaboraţi procedura determinării fluxului maxim pentru valori întregi ale capacităţilor arcelor;
3. Elaboraţi programul care va permite îndeplinirea următoarelor deziderate:
4. introducerea reţelei de transport în memorie;
5. determinarea fluxului maxim pentru reţeaua concretă;
6. extragerea datelor obținute (fluxul maxim şi fluxul fiecărui arc) la display şi printer.

3 Considerații teoretice

Rețele de transport

Un graf orientat $G = (X, U)$ se numeşte *reţea de transport* dacă satisface următoarele condiţii:

a există un vârf unic a din X în care nu intră nici un arc sau $d\_(a)=0$;
b există un vârf unic b din X din care nu iese nici un arc sau $d^{+}(a)=0$;
c G este conex şi există drumuri de la a la b în G;
d s-a definit o funcţie c: $U \rightarrow R$ astfel încât $c(u) \geq 0$ pentru orice arc u din U.

Vârful *a* se numește intrarea rețelei, vârful *b* se numește ieșirea rețelei, iar *c(u)* este capacitatea arcului u.

O funcţie f: $U \rightarrow R$ astfel încât $f(u) \geq 0$ pentru orice arc u se numeşte *flux* în reţeaua de transport G cu funcţia de capacitate c, care se notează $G = (X, U, c)$, dacă sunt îndeplinite următoarele două condiţii:

a Condiţia de conservare a fluxului: Pentru orice vârf x diferit de a şi b suma fluxurilor pe arcele care intră în x este egală cu suma fluxurilor pe arcele care ies din x.
b Condiţia de mărginire a fluxului: Există inegalitatea $f(u) \leq c(u)$ pentru orice arc $u \in U$.

Dacă $f(u) = c(u)$ arcul se numeşte *saturat*. Un *drum* se va numi *saturat* dacă va conţine cel puţin un arc saturat. Fluxul, toate drumurile căruia sunt saturate se va numi *flux complet*. Cel mai mare dintre fluxurile complete se numeşte *flux maxim*.

Pentru orice mulţime de vârfuri $A \in U$ vom defini o *tăietură* $w\_(A) = \{(x, y) \mid x \notin$ A, $y \in A$, $(x,y \in U\}$, adică mulţimea arcelor care intră în mulţimea A de vârfuri.

Prin $w^{+}(A)$ vom nota mulţimea arcelor care ies din mulţimea A de vârfuri.

Este justă afirmaţia: suma $f(u)$ pentru $u \in w^{+}(A)$ este egală cu suma $f(u)$ pentru arcele $u \in w\_(A)$. Această valoare comună se va nota f_b.

Algoritmul Ford-Fulkerson

Are loc următoarea **teoremă** (Ford-Fulkerson):

Pentru orice reţea de transport $G = (X, U, c)$ cu intrarea a şi ieşirea b valoarea maximă a fluxului la ieşire este egală cu capacitatea minimă a unei tăieturi, adică:

$$max\ f_b = min\ c(w\_(A)).$$

În baza acestei teoreme a fost elaborat următorul algoritm de determinare a fluxului maxim (Ford-Fulkerson) la ieşirea b a unei reţele de transport $G = (X, U, c)$, unde capacitatea c ia numai valori întregi:

1. Se defineşte fluxul iniţial având componente nule pe fiecare arc al reţelei, adică $f(u) = 0$ pentru orice arc $u \in U$;

2. Se determină lanţurile nesaturate de la a la b pe care fluxul poate fi mărit, prin următorul procedeu de etichetare:

a) Se marchează intrarea a cu [+];

b) Un vârf x fiind marcat, se va marca:

cu [+x] oricare vârf y nemarcat cu proprietatea că arcul $u = (x, y)$ este nesaturat, adică $f(u)<c(u)$;

cu [-x] - orice vârf y nemarcat cu proprietatea că arcul $u = (x, y)$ are un flux nenul, adică $f(u)>0$.

Dacă prin acest procedeu de marcare se etichetează ieşirea b, atunci fluxul f_b obţinut la pasul curent nu este maxim. Se va considera atunci un lanţ format din vârfurile etichetate (ale căror etichete au respectiv semnele + sau -) care uneşte pe a cu b şi care poate fi găsit uşor urmărind etichetele vârfurilor sale în sensul de la b către a.

Dacă acest lanţ este v, să notăm cu v^+ mulţimea arcelor (x, y), unde marcajul lui y are semnul “+”, deci care sunt orientate în sensul de la a către b şi cu $v\_$ mulţimea arcelor (x, y), unde marcajul lui y are semnul “-“, deci care sunt orientate în sensul de la b către a.

Determinăm cantitatea:

$e = min \{min(c(u) - f(u)), min f(u)\}. u \in v^+, u \in v\_$

Din modul de etichetare rezultă $e > 0$.

Vom mări cu e fluxul pe fiecare arc u din v^+ şi vom micşora cu e fluxul pe fiecare arc $u \in v\_$, obţinând la ieşire un flux egal cu f_b+e. Se repetă aplicarea pasului 2 cu fluxul nou obţinut.

Dacă prin acest procedeu de etichetare nu putem marca ieşirea b, fluxul f_b are o valoare maximă la ieşire, iar mulţimea arcelor care unesc vârfurile marcate cu vârfurile care nu au putut fi marcate constituie o tăietură de capacitate minimă (demonstraţi că se va ajunge în această situaţie după un număr finit de paşi).

4 Întrebări de control

1. Ce se numeşte reţea de transport?

   - Un graf orientat $G = (X, U)$ se numeşte *reţea de transport* dacă satisface următoarele condiţii:

     e existǎ un vârf unic a din X în care nu intrǎ nici un arc sau $d\_(a)=0$;
     f existǎ un vârf unic b din X din care nu iese nici un arc sau $d^{+}(a)=0$;
     g G este conex şi existǎ drumuri de la a la b în G;

     h s-a definit o funcţie c: $U \rightarrow R$ astfel încât $c(u) \geq 0$ pentru orice arc u din U.

2. Formulaţi noţiunile de flux şi capacitate.
   - Pentru orice arc din U se definește o funcție C:U→R, astfel încât C(u)>0 si C(u) se numește funcție de capacitate a arcului, sau capacitatea lui riziduală.
   - Rețeua de transport se noteaza G= Numim flux în rețeaua de transport G=<X,U,C> o funcție f:U→R, astfel încât f(u)≥0 pentru orice arc u din U și care satisface următoarele 2 condiții:
   - 1)Condiția de conservare a fluxului, adică pentru orice vârf al grafului (difert de sursă și destinație) suma fluxurilor care intră-n vârful dat este egală cu suma fluxurilor care pleacă din el;
   - 2)Condiția de mărginire a fluxului. Pentru orice arc u din U are loc inegalitatea: f(u)≤C(u).

3. Ce este un arc saturat? Dar un drum saturat?
   - Dacă f(u)=C(u), atunci arcul se numește saturat.
   - Drumul in rețeaua de transport se numește saturat, dacă conține cel puțin un arc saturat.
   -
4. Ce se numeşte flux complet? Ce este un flux maxim?
   - Fluxul,toate drumurile căruia sunt saturate se numește flux complet.

5. Definiţi noţiunea de tăietură.

   - Dacǎ prin acest procedeu de etichetare nu putem marca ieşirea b, fluxul f_b are o valoare maximǎ la ieşire, iar mulţimea arcelor care unesc vârfurile marcate cu vârfurile care nu au putut fi marcate constituie o tǎieturǎ de capacitate minimǎ (demonstraţi cǎ se va ajunge în aceastǎ situaţie dupǎ un numǎr finit de paşi).

6. Formulaţi teorema Ford-Fulkerson.
   - Pentru orice rețea de transport G=<X,U,C> cu intrarea a și destinația b valoarea maxima a fluxului la ieșire fb este egală cu capacitatea minimă a unei tăieturi, adică: max fb =minC(W-(A)).

7. Descrieţi algoritmul de determinare a fluxului maxim.

   1. Se defineşte fluxul iniţial având componente nule pe fiecare arc al reţelei, adicǎ $f(u) = 0$ pentru orice arc $u \in U$;

   2. Se determinǎ lanţurile nesaturate de la a la b pe care fluxul poate fi mǎrit, prin urmǎtorul procedeu de etichetare:

   a) Se marcheazǎ intrarea a cu [+];

   b) Un vârf x fiind marcat, se va marca:

   cu [+x] oricare vârf y nemarcat cu proprietatea cǎ arcul $u = (x, y)$ este nesaturat, adicǎ $f(u)<c(u)$;

   cu [-x] - orice vârf y nemarcat cu proprietatea cǎ arcul $u = (x, y)$ are un flux nenul, adicǎ $f(u)>0$.

Dacă prin acest procedeu de marcare se etichetează ieşirea *b*, atunci fluxul f_b obţinut la pasul curent nu este maxim. Se va considera atunci un lanţ format din vârfurile etichetate (ale căror etichete au respectiv semnele + sau -) care uneşte pe *a* cu *b* şi care poate fi găsit uşor urmărind etichetele vârfurilor sale în sensul de la *b* către *a*.

Dacă acest lanţ este *v*, să notăm cu v^+ mulţimea arcelor *(x, y)*, unde marcajul lui *y* are semnul “+”, deci care sunt orientate în sensul de la *a* către *b* şi cu $v\_$ mulţimea arcelor *(x, y)*, unde marcajul lui *y* are semnul “-“, deci care sunt orientate în sensul de la *b* către *a*.

Determinăm cantitatea:

$e = min \{min(c(u) - f(u)), min\ f(u)\}.\ u \in v^+, u \in v\_$

Din modul de etichetare rezultă $e > 0$.

Vom mări cu *e* fluxul pe fiecare arc *u* din v^+ şi vom micşora cu *e* fluxul pe fiecare arc $u \in v\_$, obţinând la ieşire un flux egal cu f_b+e. Se repetă aplicarea pasului 2 cu fluxul nou obţinut.

Dacă prin acest procedeu de etichetare nu putem marca ieşirea *b*, fluxul f_b are o valoare maximă la ieşire, iar mulţimea arcelor care unesc vârfurile marcate cu vârfurile care nu au putut fi marcate constituie o tăietură de capacitate minimă (demonstraţi că se va ajunge în această situaţie după un număr finit de paşi).

8. Demonstraţi că algoritmul se va opri după un număr finit de paşi.
   - Datorită faptului că unele arce se vor satura în cele din urmă va fi imposibil de etichetat ieșirea astfel v-om obține taietura minimă.

5 Codul programului

```
from collections import defaultdict
import networkx as nx
import matplotlib.pyplot as plt
import json

# import subprocess


class GRAF:
def __init__(self):
self.graf = defaultdict(list)
self.c = defaultdict(int)
self.f = defaultdict(list)
self.sursa = 0
self.destinatia = 0
self.fMax = 0
self.L = []
self.E = []
self.A = []
self.X = []
self.XA = []
self.WA = []

def citirea(self):
print("citirea listei de adiacenta")
self.graf = defaultdict(list)
i = 1
while True:
print("pentru a termina tastati ( q )")
print("aveti muchia", i, "cu prima extremitate")
extr1 = input()
if extr1 == "q":
break
print("si a doua extremitate")
extr2 = input()
if extr2 == "q":
break
self.graf[int(extr1)].append(int(extr2))
print("capacitatea arcului", extr1, "->", extr2, "este:")
self.c[(int(extr1), int(extr2))] = int(input())
i += 1
self.curatare()

def curatare(self):
self.c = defaultdict(int)
```

```
self.f = defaultdict(list)
self.sursa = 0
self.destinatia = 0
self.fMax = 0
self.L = []
self.E = []
self.A = []
self.X = []
self.XA = []
self.WA = []
for v in [*self.graf]:
for c in self.graf[v]:
if c not in [*self.graf]:
self.graf[c] = []
self.graf[v].sort()
self.graf[v] = list(dict.fromkeys(self.graf[v]))
self.graf = dict(sorted(self.graf.items()))

def afiseazaLista(self):
print("lista de adiacenta")
for k in [*self.graf]:
print(k, "-", end=" ")
for v in self.graf[k]:
print(str(v)+"(" + str(self.c[(k, v)]) + "), ", end="")
print("0")

def salveaza(self):
print("denumirea fisierului")
f = input()
json.dump(self.graf, open(f+".json", 'w'))

temp = defaultdict(list)
for k in [*self.c]:
temp[k[0]].append([k[1], self.c[k]])
json.dump(temp, open(f+".c.json", 'w'))

def impota(self, f=""):
print("denumirea fisierului")
if f == "":
f = input()
self.graf = json.load(open(f+".json"))
# self.graf = {int(k): [int(i) for i in v] for k, v in self.graf.items()}
temp = defaultdict(list)
for key in [*self.graf]:
for v in self.graf[key]:
temp[int(key)].append(v)
self.graf = temp
```

```
self.curatare()

temp = json.load(open(f+".c.json"))
for k in [*temp]:
for l in temp[k]:
self.c[(int(k), l[0])] = l[1]

def deseneazaGraful(self):
g = nx.DiGraph()
for i in [*self.graf]:
text = ""
for j in self.graf[i]:
text = str(self.c[(i, j)])
for e in self.f[(i, j)]:
text = text+"("+str(e)+")"
g.add_edge(i, j, weight=text)
pos = nx.circular_layout(g)
edge_labels = {(u, v): d['weight'] for u, v, d in g.edges(data=True)}
nx.draw(g, pos, with_labels=True, node_size=1700, font_size=40)
nx.draw_networkx_edge_labels(
g, pos, edge_labels=edge_labels, font_size=17)
plt.savefig('output.png')
plt.show()
# subprocess.run(["google-chrome", "output.png"])

def removeVertex(self):
print(self.graf)
print("varful pe care doriti sa il stergeti :")
v = int(input())
for i in [*self.graf]:
self.graf[i] = [item for item in self.graf[i] if item != v]
self.graf.pop(v, None)

def removeEdge(self):
print(self.graf)
print("varful din care iese muchia :")
e = int(input())
print("varful in care intra muchia :")
i = int(input())
self.graf[e] = [item for item in self.graf[e] if item != i]

def addEdge(self):
print(self.graf)
print("puteti adauga orice muchie (chiar cu varfuri noi)")
print("varful din care iese muchia :")
e = int(input())
print("varful in care intra muchia :")
```

```
i = int(input())
self.graf[e] = i
print("capacitatea:")
c = int(input())
self.c[(e, i)] = c

def edit(self):
print("puteti :\nsterge un varf - v\nsterge o muchie - m\nadauga o muchie - a")
o = input()
if o == "v":
self.removeVertex()
elif o == "m":
self.removeEdge()
elif o == "a":
self.addEdge()
self.curatare()

def determinare_a_b(self):
intrari = defaultdict(bool)
for k in [*self.graf]:
intrari[k] = False
for k in [*self.graf]:
if not len(self.graf[k]):
self.destinatia = k
for v in self.graf[k]:
intrari[v] = True
for v in [*intrari]:
if not intrari[v]:
self.sursa = v

def Ford_Fulkerson_pain(self):

F_max = 0
s = self.sursa
t = self.destinatia
m = defaultdict(list)
rg = defaultdict(list)
c = self.c
g = self.graf

f = self.f
for k in [*c]:
f[k].append(0)

# cod ciotkii) =>
z = 1
A = []
while True:
```

```
l = [s]
blacklist = set()
while l[len(l)-1] != t:
remove = True
for v in g[l[len(l)-1]]:
# print(l)
if v not in l and v not in blacklist:
if sum(f[(l[len(l)-1], v)]) != c[(l[len(l)-1], v)]:
# print("inainte")
m[v].append(("+", l[len(l)-1]))
rg[v].append(l[len(l)-1])
l.append(v)
remove = False
break
for v in rg[l[len(l)-1]]:
if v not in l and v not in blacklist:
if sum(f[v, (l[len(l)-1])]):
# print("inapoi")
m[v].append(("-", l[len(l)-1]))
l.append(v)
remove = False
break
if remove:
blacklist.add(l[len(l)-1])
if m[l[len(l)-1]]:
m[l[len(l)-1]].pop()
l.pop()
if s in blacklist:
A = blacklist
break
# print(blacklist)
if A:
break
# print("l", z, l)
self.L.append(l)
E = []
for i in range(1, len(l)):
e = list(m[l[i]]).pop()
if e[0] == "+":
E.append(c[(e[1], l[i])]-sum(f[(e[1], l[i])]))
else:
E.append(sum(f[(l[i], e[1])]))
# print("E", z, " _ min", E, end=" = ")
self.E.append(E)
E = min(E)
# print(E)
F_max += E
```

```
# print("F_max", F_max)
for i in l:
if i == s:
p = s
else:
if list(m[i]).pop()[0] == "+":
f[(p, i)].append(E)
else:
f[(i, p)].append(-E)
p = i
z += 1

# print("A", A)
X = list(g)
# print(X)
XA = [i for i in X if i not in A]
# print(XA)
print(A)
print(XA)
for o in A:
for d in XA:
if (o, d) in [*c]:
self.WA.append((o, d))
self.A = A
self.X = X
self.XA = XA
self.fMax = F_max

def afiseazadata(self):
s3 = '‾'*65
for i in range(0, len(self.L)):
s1 = "l"+str(i+1) + " = "+str(self.L[i])
s2 = "E" + str(i+1) + " = min " + \
str(self.E[i]) + " = "+str(min(self.E[i]))
print(s3)
print('{:30s} {:>1} {:30s}'.format(s1, "|", s2))
print(s3)

for a in [*self.f]:
print("fluxul arcului", a[0], "->", a[1], "=", sum(self.f[a]))
print(s3)

print("Secțiunea minimală se obține pentru A =",
self.XA, "(mulțimea vârfurilor nemarcate )")
print("W-A", self.WA, "- tăietura de capacitate minimă,")
capacitatea = 0
print("c", end="")
```

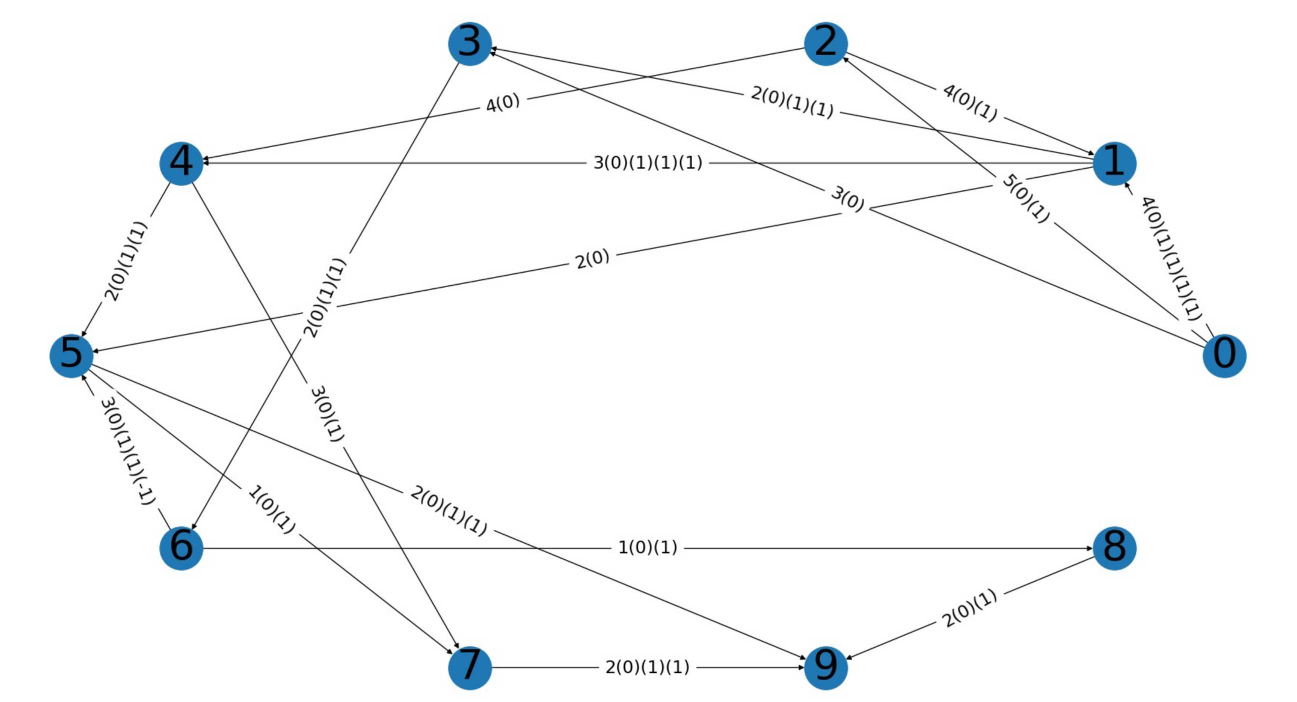
```
i = 0
for a in self.WA:
i += 1
capacitatea += self.c[a]
print(self.c[a], end="")
if i != len(self.WA):
print("+", end="")
print("=", capacitatea, " –capacitatea tăieturii")
print("Conform teoremei lui Ford-Fulkerson")
print("F_max =c", end="")
flux = 0
i = 0
for a in self.WA:
i += 1
flux += self.c[a]
print(self.c[a], end="")
if i != len(self.WA):
print("+", end="")
print("=", self.fMax)

def START(self):
print("program la msp")
while True:
print("( q ) - pentru a iesi")
print(
"( c ) - pentru a citi din memorie lista de adiacenta (in caz ca a fost salvata precedent )")
print("( s ) - pentru a scrie in memorie lista de adiacenta")
print("")
print("( 1 ) - pentru a citi de la tastatura lista de adiacenta")
print("( 2 ) - pentru a afisa lista")
print("( 3 ) - pentru a afisa forma grafica")
print("( 4 ) - FORD-FULKERSON")
print("( 8 ) - EDITARE")
print("( 9 ) - pentru a genera un graf intamplator")
o = input()
if o == "q":
break
elif o == "c":
self.impota()
self.determinare_a_b()
print("sursa", self.sursa, "destinatia", self.destinatia)
# self.deseneazaGraful()
elif o == "1":
self.citirea()
self.determinare_a_b()
elif o == "s":
self.salveaza()
```

```
elif o == "2":
self.afiseazaLista()
elif o == "3":
self.deseneazaGraful()
elif o == "4":
self.Ford_Fulkerson_pain()
self.afiseazadata()
elif o == "8":
self.edit()
elif o == "h":
print("graf:", self.graf)
print("capacitati:", self.c)
print("flux", self.f)


graf = GRAF()

graf.START()
```

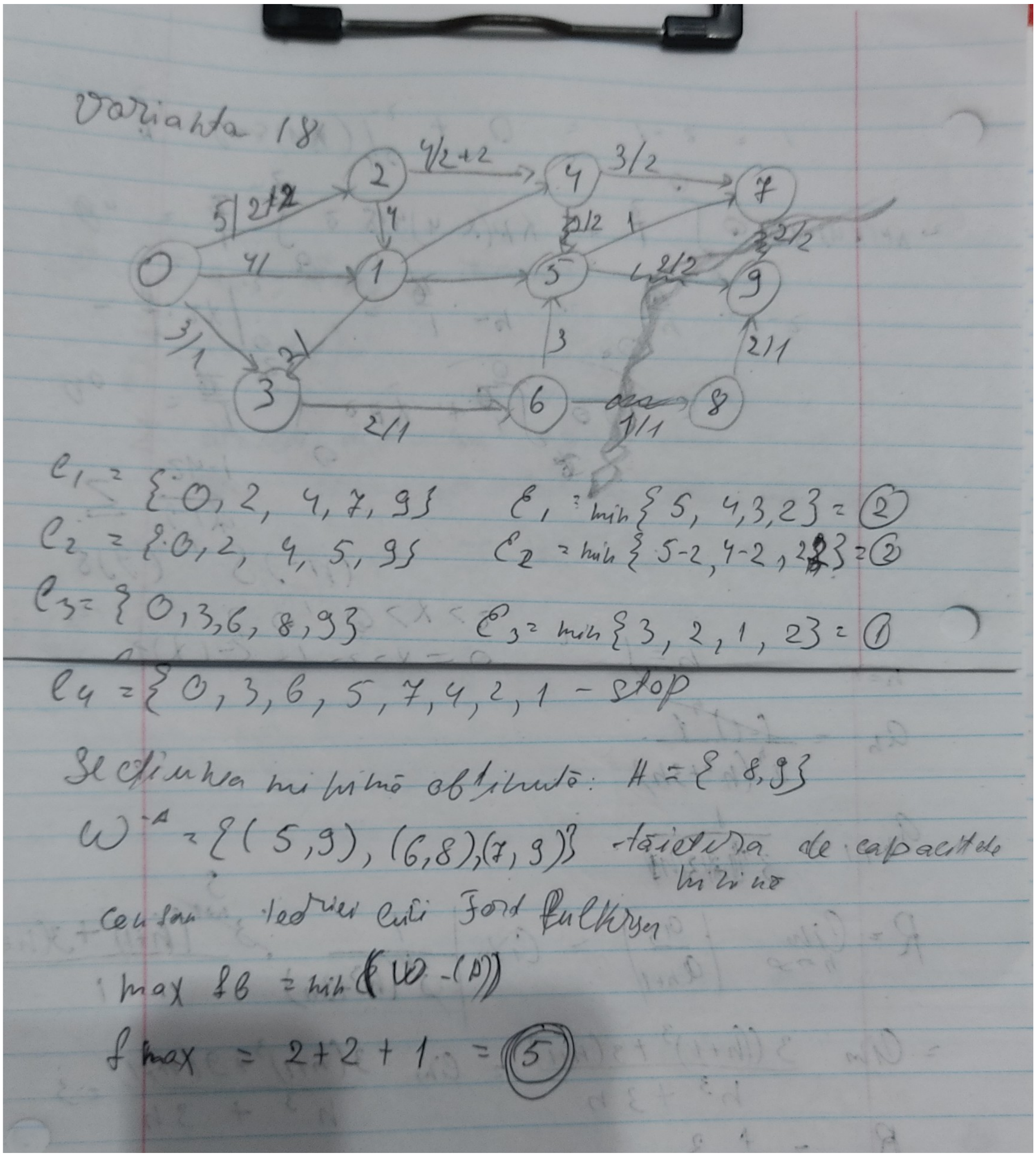
6 Testarea programului

```
( q ) - pentru a iesi
( c ) - pentru a citi din memorie lista de adiacenta (in caz ca a fost
salvata precedent )
( s ) - pentru a scrie in memorie lista de adiacenta

( 1 ) - pentru a citi de la tastatura lista de adiacenta
( 2 ) - pentru a afisa lista
( 3 ) - pentru a afisa forma grafica
( 4 ) - FORD-FULKERSON
( 8 ) - EDITARE
( 9 ) - pentru a genera un graf intamplator
4
{0, 1, 2, 3, 4, 5, 6, 7}
[8, 9]
------------------------------------------------------------------
l1 = [0, 1, 3, 6, 5, 7, 9]      | E1 =  min [4, 2, 2, 3, 1, 2] = 1
------------------------------------------------------------------
l2 = [0, 1, 3, 6, 5, 9]         | E2 =  min [3, 1, 1, 2, 2] = 1
------------------------------------------------------------------
l3 = [0, 1, 4, 5, 6, 8, 9]      | E3 =  min [2, 3, 2, 2, 1, 2] = 1
------------------------------------------------------------------
l4 = [0, 1, 4, 5, 9]            | E4 =  min [1, 2, 1, 1] = 1
------------------------------------------------------------------
l5 = [0, 2, 1, 4, 7, 9]         | E5 =  min [5, 4, 1, 3, 1] = 1
------------------------------------------------------------------
```

```
------------------------------------------------------------------
fluxul arcului 0 -> 1 = 4
fluxul arcului 0 -> 2 = 1
fluxul arcului 0 -> 3 = 0
fluxul arcului 1 -> 3 = 2
fluxul arcului 1 -> 5 = 0
fluxul arcului 1 -> 4 = 3
fluxul arcului 2 -> 1 = 1
fluxul arcului 2 -> 4 = 0
fluxul arcului 3 -> 6 = 2
fluxul arcului 4 -> 5 = 2
fluxul arcului 4 -> 7 = 1
fluxul arcului 5 -> 9 = 2
fluxul arcului 5 -> 7 = 1
fluxul arcului 6 -> 5 = 1
fluxul arcului 6 -> 8 = 1
fluxul arcului 7 -> 9 = 2
fluxul arcului 8 -> 9 = 1
------------------------------------------------------------------

Secţiunea minimală se obţine pentru A = [8, 9] (mulţimea vârfurilor nem
arcate )
W-A [(5, 9), (6, 8), (7, 9)] - tăietura de capacitate minimă,
c2+1+2= 5  -capacitatea tăieturii
Conform teoremei lui Ford-Fulkerson
F_max =c2+1+2= 5
```



0
1
2
3
4
5
6
7
8
9
4(0)
2(0)(1)(1)
4(0)(1)
3(0)(1)(1)(1)
3(0)
5(0)(1)
4(0)(1)(1)(1)(1)
2(0)(1)(1)
2(0)(1)(1)
2(0)
3(0)(1)
3(0)(1)(1)(-1)
1(0)(1)
2(0)(1)(1)
1(0)(1)
2(0)(1)
2(0)(1)(1)



Varianta 18
$e_1 = \{0, 2, 4, 7, 9\}$ $\varepsilon_1 = \min\{5, 4, 3, 2\} = 2$
$e_2 = \{0, 2, 4, 5, 9\}$ $\varepsilon_2 = \min\{5-2, 4-2, 2, 2\} = 2$
$e_3 = \{0, 3, 6, 8, 9\}$ $\varepsilon_3 = \min\{3, 2, 1, 2\} = 1$
$e_4 = \{0, 3, 6, 5, 7, 4, 2, 1$ - stop
Secțiunea minimă obținută: $A = \{8, 9\}$
$\omega^{-A} = \{(5,9), (6,8), (7,9)\}$ tăietura de capacitate minimă
Conform teoriei lui Ford Fulkerson
$\max f_B = \min(\omega^-(A))$
$f_{max} = 2 + 2 + 1 = 5$

7 Concluzii

- Datorită efectuării acestui laborator am studiat mai aprofundat algoritmul Ford-Fulkerson.
- Am utilizat acest algoritm pentru a afla fluxul maxim și taietura minima.
- Implement
- Bazândumă pe teorema ford am creat un program care calculează fluxul maxim.
- Debugherul este un instrument foarte util, fără utilizarea căruia ar fi imposibil să realizez acest program.
- Am aplicat temele studiate anterior, mai exact parcurgerea în adâncime pentru generarea drumurilor.
- În mare parte am modificat programul de la laboratorul precedent.