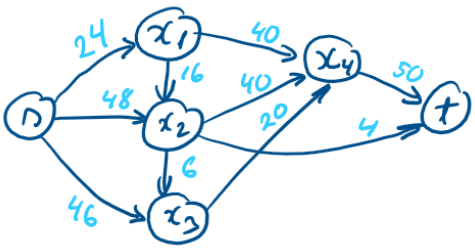
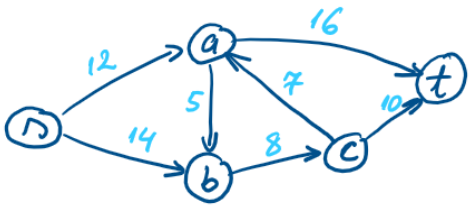
Suport seminar algoritmica grafurilor
V. Rețele de flux

Probleme:

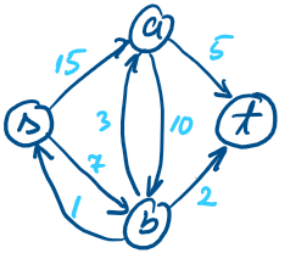
1. Folosind algoritmul Ford-Fulkerson să se determine fluxul maxim pentru graful din figura (1a).
2. Folosind algoritmul de pompare preflux să se determine fluxul maxim pentru graful din figura (1b).
3. Folosind algoritmul de pompare topologică să se determine fluxul maxim pentru graful din figura (1c).



(a)



(b)



(c)

Figura 1

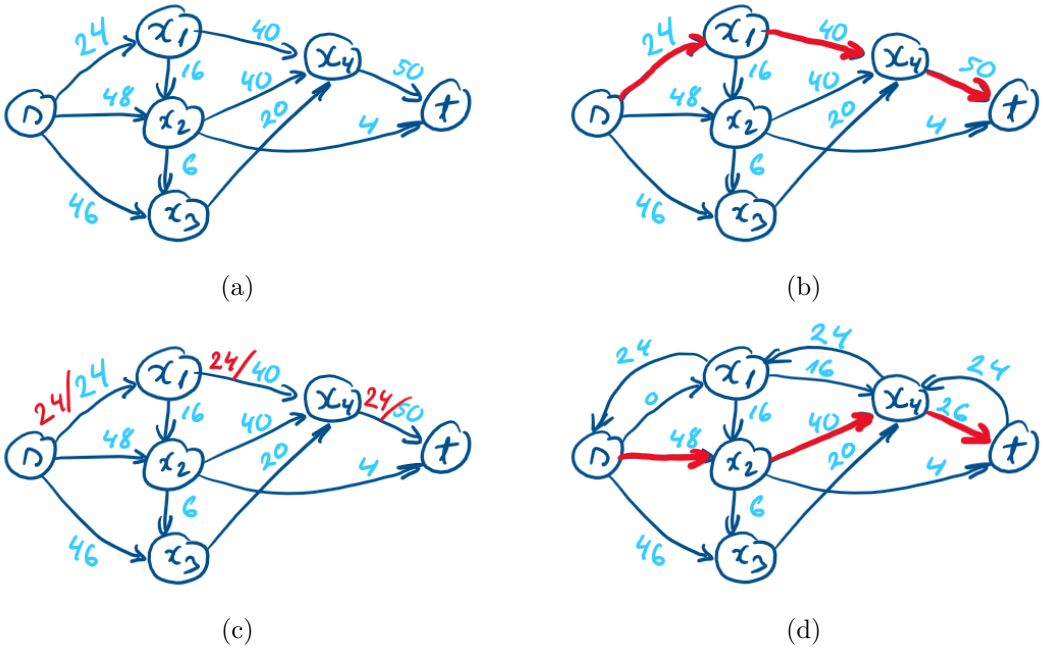
Rezolvări

Problema 1 Pentru a rezolva problema se aplică algoritmul Ford-Fulkerson. În fiecare iterație a algoritmului se găsește o cale reziduală p și se folosește p pentru a modifica fluxul f. Se înlocuiește f cu $f \uparrow f_p$ și se obține un nou flux cu valoarea $|f| + |f_p|$.

Algoritmul nu specifică cum se găsește calea reziduală. Dacă f^* reprezintă fluxul maxim din rețeaua de transport, în cel mai rău caz algoritmul execută liniile $1-9$ de cel mult $|f^*|$ ori deoarece fluxul crește în fiecare iterație cel puțin cu o unitate.

FORD_FULKERSON(G, s, t)
1: **for** $(u, v) \in E$ **do**
2: $\quad (u, v).f = 0$
3: **while** există o cale reziduală p de la s la t în graful rezidual G_f **do**
4: $\quad c_f(p) = min\{c_f(u, v)|(u, v) \in p\}$
5: $\quad$ **for** fiecare arc $(u, v) \in p$ **do**
6: $\quad\quad$ **if** $(u, v) \in E$ **then**
7: $\quad\quad\quad (u, v).f = (u, v).f + c_f(p)$
8: $\quad\quad$ **else**
9: $\quad\quad\quad (v, u).f = (v, u).f - c_f(p)$
10: **return** f

Figura 2 prezintă rezultatul fiecărei iterații pentru algoritm, în cazul căilor reziduale indicate. Figurile (2a)-(2g) prezintă iterațiile succesive ale buclei **while**. Coloana dreaptă prezintă grafurile reziduale G_f din linia 3 a algoritmului și calea reziduală (formată din arcele colorate în roșu). Coloana stângă prezintă noul flux din rețea. Pentru a indica capacitatea unui arc și fluxul transmis pe acel arc se folosește notația *flux/capacitate* (cu roșu se prezintă valoarea fluxului și cu albastru valoarea capacității). În prima iterație graful rezidual este identic cu rețeaua de flux (figurile (2a) și (2b)).



(a) (b)

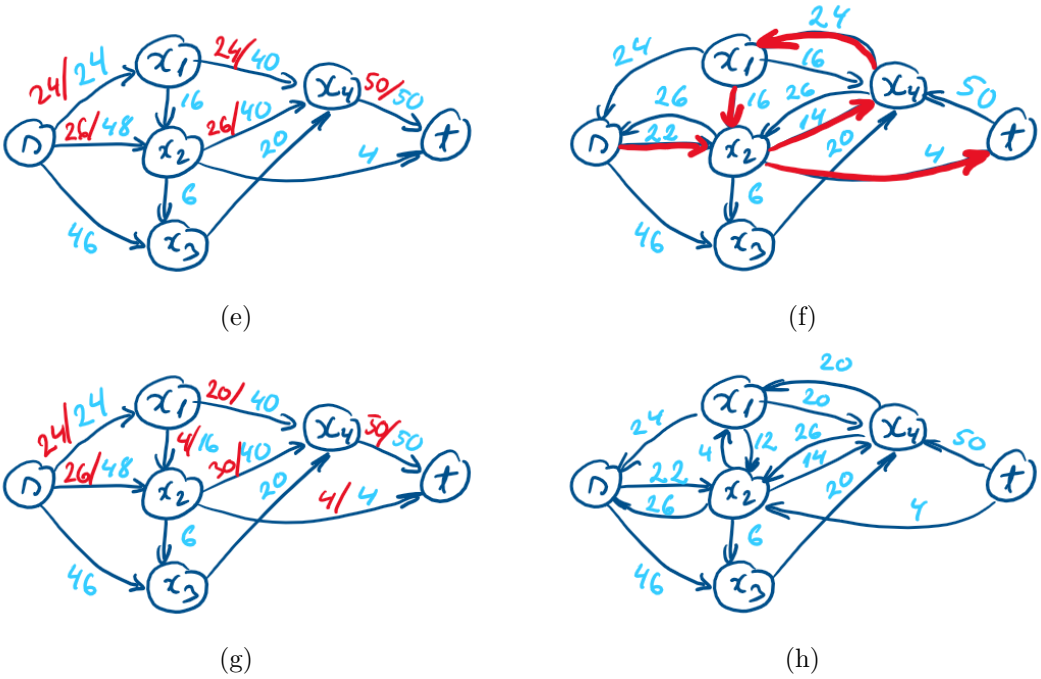
(c) (d)

Figura 2: Algoritmul Ford-Fulkerson aplicat grafului din figura (1a).

De exemplu, pentru graful rezidual din figura 2b) se găsește calea reziduală $s \to x_1 \to x_4 \to t$ cu capacitatea $c_f = min\{24, 40, 50\}$. Pentru această cale reziduala fluxul maxim în rețeaua de transport crește la valoarea $|f| = 24$ și va fi transmis pe calea reziduala $s \to x_1 \to x_4 \to t$ (figura (2c)). Graful rezidual (2d) prezintă rețeaua de transport pentru fluxul $|f| = 24$ și calea reziduală

identificată.

Algoritmul se oprește când în graful rezidual nu se mai găsește o cale rezidual, un drum de la vârful s la vârful t (figura (2h)). Fluxul maxim în graf are valoarea $|f| = 54$.



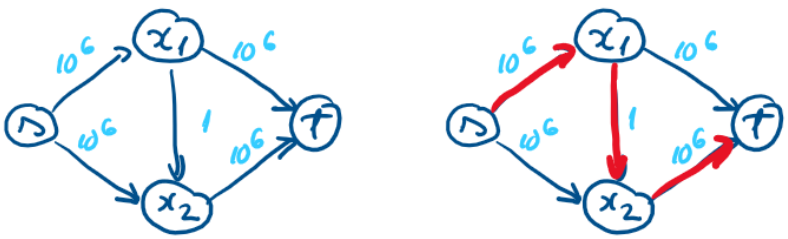
(e) (f)

(g) (h)

Figura 2: (continuare) Algoritmul Ford-Fulkerson aplicat grafului din figura (1a).

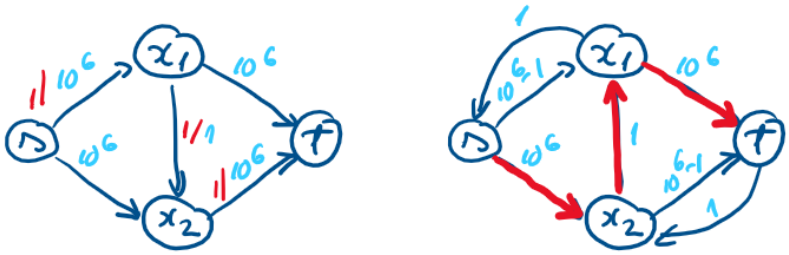
Pentru valori întregi ale capacităților și pentru un flux maxim $|f^*|$ mic, timpul de execuție pentru algoritmul Ford-Fulkerson este bun. Figura (3a) prezintă un caz extrem, ce se poate întâmpla în cel mai rău caz, pentru o rețea de flux mică pentru care $|f^*|$ este mare. Pentru această rețea fluxul maxim are valoarea $2 \cdot 10^6$ unde 10^6 unități de flux traversează drumul $s \to x_1 \to t$ și 10^6 unități de flux traversează drumul $s \to x_2 \to t$.

Dacă algoritmul Ford-Fulkerson găsește prima cale rezidual $s \to x_1 \to x_2 \to t$ (figura (3b)), fluxul maxim în rețea după prima iterație are valoarea 1. Graful rezidual rezultat este prezentat în figura (3d). Dacă în a doua iterație se găsește calea rezidual $s \to x_2 \to x_1 \to t$, ca în figura (3d), fluxul în rețea va avea valoarea 2. Dacă în iterațiile impare se alege calea rezidual $s \to x_1 \to x_2 \to t$ și în iterațiile pare se alege calea rezidual $s \to x_2 \to x_1 \to t$ se vor executa $2 \cdot 10^6$ iterații, fluxul crește în fiecare iterație cu o unitate.



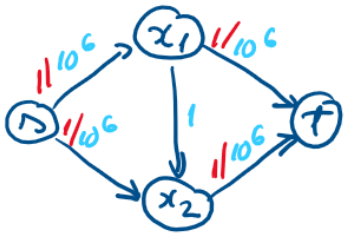
(a) (b)

Figura 3: Un exemplu pentru care algoritmul Ford-Fulkerson poate lua $\theta(E|f^*|)$ timp, f^* reprezintă fluxul maxim în graf. Pentru această reţea fluxul maxim este $|f^*| = 2 \cdot 10^6$.

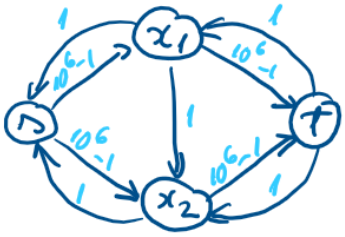


(c)

(d)



(e)



(f)

Figura 3: (continuare) Un exemplu pentru care algoritmul Ford-Fulkerson poate lua $\theta(E|f^*|)$ timp, f^* reprezintă fluxul maxim în graf. Pentru această reţea fluxul maxim este $|f^*| = 2 \cdot 10^6$.

Pentru a îmbunătăți algoritmul Ford-Fulkerson se vor cauta căile reziduale (linia 3) folosind un algoritm de parcurgere în lățime, algoritmul rezultat se numește Edmonds-Karp. Pentru rețeaua de transport din figura (3a) fluxul maxim se va determina în două iterații.

Problema 2 Algoritmii de pompare rezolva problema fluxului maxim din rețea într-un mod mai localizat/concentrat față de abordarea Ford-Fulkerson. Aceștia se "uită" la un moment dat doar doar la un vârf și la vecinii acestuia din graful rezidual, nu caută o îmbunătățire sub forma unui drum în tot graful rezidual.

Algoritmii de pompare nu respectă proprietatea de conservare a fluxului pe parcursul execuției. Fluxul se poate acumula într-un nod, pe parcursul execuției, sub numele de exces de flux $u.e = \sum_{v \in V}(v,u).f - \sum_{v \in V}(u,v).f$.

Intuitiv, funcționarea algoritmilor de pompare este similară cu curgerea lichidelor printr-un sistem de conducte de diverse capacități care leagă puncte aflate la diverse înălțimi. Inițial vârful sursă s al rețelei este cel mai înalt, celelalte vârfuri fiind la înălțimea 0. Destinația t rămâne permanent la înălțimea 0. Prefluxul este inițializat încărcând la capacitate maximă toate conductele care pornesc din s. În cursul funcționării, se poate întâmpla ca fluxul (lichidul) strâns într-un vârf u din conductele ce alimentează vârful să depășească posibilitatea de eliminare prin conductele de scurgere la care este conectat vârful (restricția de conservare a fluxului nu mai este îndeplinită). Excesul de flux este stocat într-un rezervor $u.e$ al vârfului, cu capacitatea nelimitată teoretic. Pentru a echilibra rețeaua și a elimina supraîncărcarea vârfurilor, sunt efectuate două operații de bază:

- Mărirea înălțimii unui vârf. Atunci când un vârf supraîncărcat u are o conductă de scurgere orientată spre un vecin v, care nu este încărcat la capacitate maximă, u este înălțat mai sus decât v, astfel încât fluxul să poată curge din u în v prin conducta (u, v).

- Pomparea fluxului unui vârf. Un vârf u supraîncărcat, aflat la o înălțime mai mare decât un vecin v și conectat cu v printr-o conductă subîncărcată, pompează flux din rezervorul $u.e$ spre v prin conducta (u, v).

Treptat, înălțimile vârfurilor cresc monoton și pot depăși înălțimea sursei s. În acest moment, fluxul în exces din rețea se pompează sursei. Numărul de operații de mărire a înălțimii vârfurilor și de pompare a fluxului este limitat iar atunci când nu mai este posibilă o operație de acest fel, fluxul din rețea este maxim.

Algoritmul de pompare preflux și rutinele de inițializare, pompare și înălțare sunt prezentate mai jos.

INITIALIZARE_PREFLUX(G,s,t)

\\ inițializare $f(u,v)$ și $h(u,v)$, $\forall u,v \in V$
for fiecare $v \in V$ **do**
    $v.h = 0$
    $v.e = 0$
for fiecare $(u,v) \in E$ **do**
    (u,v).f=0
$s.h = |V|$
for fiecare $v \in s.Adj$ **do**
    $(s,v).f = c(s,v)$
    $v.e = c(s,v)$
    $s.e = s.e - c(s,v)$

POMPARE(u,v)

\\ condiție de aplicare: $u \notin \{s,t\} \wedge u.e > 0 \wedge c_f(u,v) > 0 \wedge u.h = v.h + 1$
\\ acțiune: pompează cantitatea de flux $\Delta_f(u,v) = min(u.e, c_f(u,v))$
$\Delta_f(u,v) = min(u.e, c_f(u,v))$
if $(u,v) \in E$ **then**
    $(u,v).f = (u,v).f + \Delta_f(u,v)$
else
    $(v,u).f = (v,u).f - \Delta_f(u,v)$
$u.e = u.e - \Delta_f(u,v)$
$v.e = v.e + \Delta_f(u,v)$

ÎNĂLȚARE(u)

\\ condiție de aplicare: $u \notin \{s,t\} \wedge u.e > 0 \wedge [u.h \leq v.h | \forall v \in V, (u,v) \in E_f]$
\\ acțiune: mărește înălțimea $u.h$
$u.h = 1 + min\{v.h | (u,v) \in E_f\}$

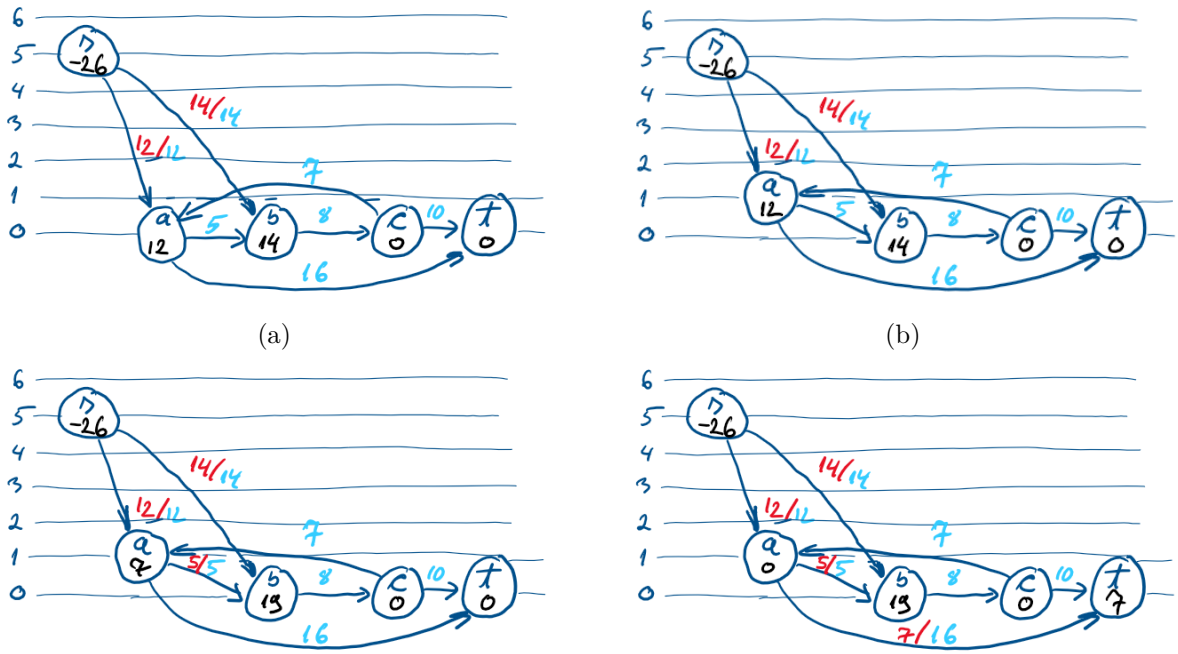
POMPARE_PREFLUX(G,s,t)

INITIALIZARE_PREFLUX(G,s,t)
while TRUE **do**
    if $\exists u \notin \{s,t\} \wedge u.e > 0 \wedge c_f(u,v) > 0 \wedge u.h = v.h + 1$ **then**
        POMPARE(u,v)
        continue
    if $\exists u \notin \{s,t\} \wedge u.e > 0 \wedge [u.h \leq v.h | \forall v \in V, (u,v) \in E_f]$ **then**
        INALTARE(u)

8: continue
9: break

Figura (4a) prezintă rețeaua de flux din figura (1b) după procedura de inițializare. Vârful sursă are atributul înălțime $s.h = 5$ și pompează fluxul maxim posibil vârfurilor vecine. Restul vârfurilor se află la înălțimea 0. Excesul de flux este notat sub eticheta vârfului (numerele colorate cu negru), excesul pentru vârfurile adiacente cu s este $a.e = 12$ și $b.e$=14.

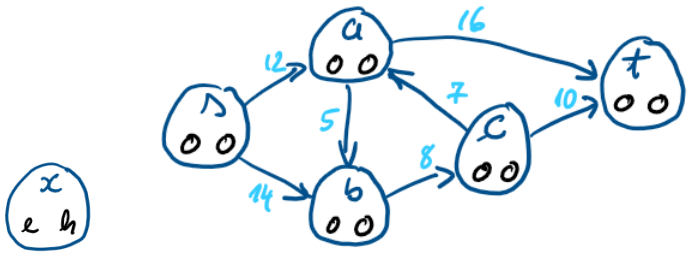
După procedura de inițializare nu se poate descărca excesul de flux din vârfurile $u \in V \setminus \{s, t\}$ și se aplică procedura de înălțare pentru vârful a, figura (4b). După acest pas $a.h = 1$. În figura (4c) se descarcă excesul de flux din vârful a în vârful b, după acest pas $b.e = 19$ și $a.e = 7$. În pasul următor (figura (4d)), restul de exces de flux din vârful a este descărcat în vârful destinație t, $a.e = 0$ și $t.e = 7$.



(a) (b) (c) (d)

Figura 4: Un exemplu de pompare preflux.

Pentru simplitatea prezentării se va adopta notația din figura (5a) unde sub eticheta unui vârf se va trece valoarea atributului exces și înălțime. (5b) prezintă rețeaua de flux cu noua notație.



(a) (b)

Figura 5: Pompare preflux. Sub eticheta vârfului se trece valoarea atributelor exces şi înălţime.

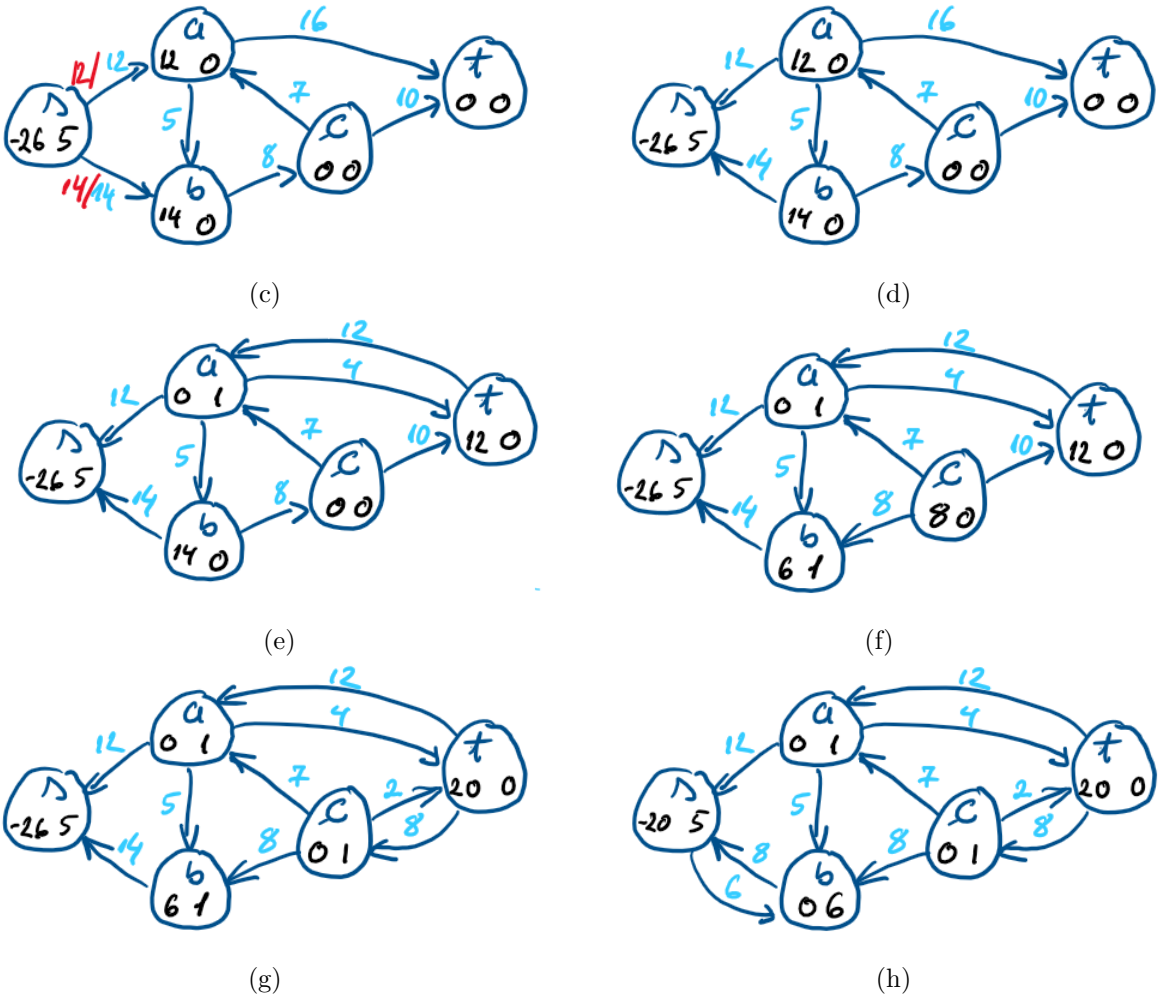


Figura 5: (continuare) Pompare preflux.

Figurile (5c)-(5h) prezintă pașii algoritmului de pompare preflux pentru a determina fluxul maxim în rețea. În figura (5c) se inițializează prefluxul. Figurile (5d)-(5h) prezintă graful rezidual și valoarea atributelor e și h după operații de înălțare și pompare. Astfel după inițializare se înalță vârful a și se pompează excesul de flux în vârful t, figura (5e). După acest pas se înalță vârful b și se pompează 8 unități de flux în vârful c, figura (5f). În următorii doi pași se înalță vârful c și excesul său de flux se pompează în vârful t, figura (5g). Singurul vârf din rețea diferit de sursă și destinație care mai are exces de flux este vârful b, în următorii pași asupra acestuia se aplică procedura de înălțare până când se poate pompa excesul de flux înapoi în vârful sursă nemaiexistând alt drum în afară de (b, s) pe care să se poată pompa excesul de flux. Înălțimea vârfului b, după acești pași, este $b.h = 6$ și excesul de flux $b.e = 0$, figura (5h). Algoritmul se oprește, nu mai există vârfuri asupra cărora să se aplice procedurile de pompare sau înălțare. Fluxul maxim în graf este $|f| = 20$.

Problema 3 Algoritmul de pompare topologică este o variantă a pompării prefluxului. Spre deosebire de algoritmul de pompare preflux, pomparea topologică impune o disciplină strictă de secvențiere a operațiilor elementare de pompare a prefluxului prin rețeaua de transport G și de înălțare a vârfurilor rețelei. Pașii algoritmului sunt descriși în curs. Mai jos este prezentat algoritmul de pompare topologică și procedura de descărcare a excesului de flux.

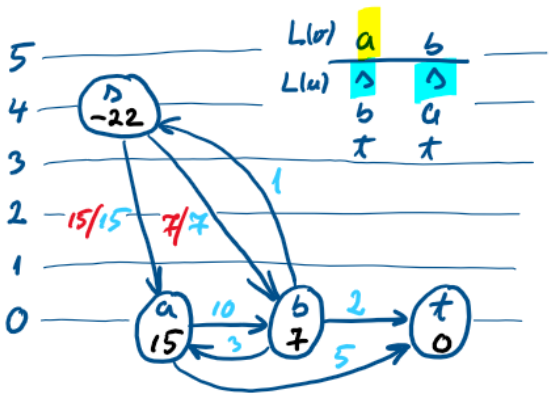
Algoritmul este o variantă a celui de pompare preflux. În algoritmul de pompare preflux, opera-

țiile de înălțare și pompare pot fi executate în orice ordine, inclusiv în ordinea impusă de regulile pompării topologice.

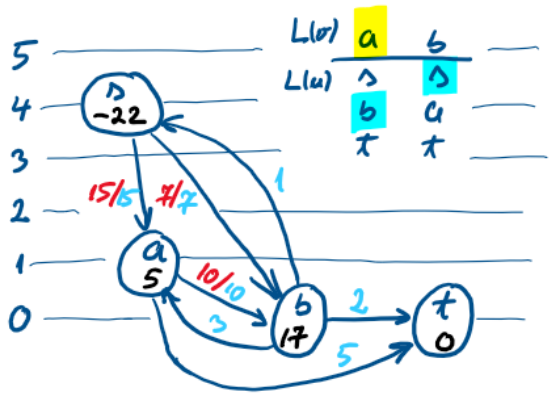
```
POMPARE_TOPOLOGICA(G,s,t)
1: INITIALIZARE_PREFLUX(G,s,t)
2: L = V \ {s,t}
3: for fiecare u ∈ V \ {s,t} do
4:     u.curent = u.N.head
5: u = L.head
6: while u ≠ NIL do
7:     înălțime_veche = u.h
8:     DESCARCARE(u)
9:     if u.h > înălțime_veche then
10:        mută u în capul listei L
11:    u.next
```

```
DESCARCARE(u)
1: while u.e > 0 do
2:     v = u.curent
3:     if v == NIL then
4:         INALTARE(u)
5:         u.curent = u.N.head
6:     else if c_f(u,v) > 0 ∧ u.h == v.h + 1 then
7:         POMPARE(u,v)
8:     else
9:         u.curent = u.urmatorul_vecin
```

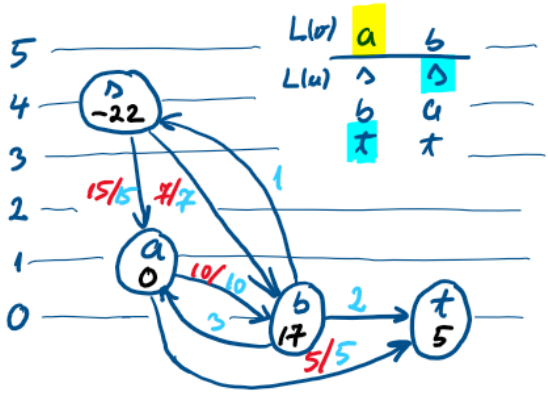
Pentru a ilustra funcționalitatea algoritmului de pompare topologică, numărul dintr-un vârf (colorat cu negru) reprezintă excesul de flux al acestuia, în dreapta rețelei de flux este prezentată lista $L(V)$ și listele $L(u)$ pentru $u \in V \setminus \{s,t\}$.



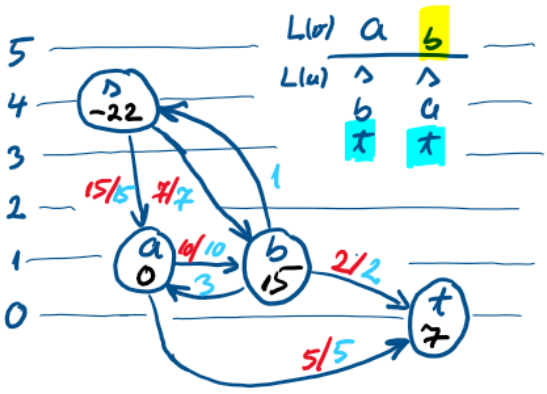
Pentru graful din cerință (figura (1c)), starea rețelei după inițializare este prezentată în figura de mai sus. $L(a) = (s\ b\ t)$, $L(b) = (s\ a\ t)$, $L(V) = (a\ b)$, arcele (s,a) și (s,b) au flux maxim și excesul vârfurilor a și b este $a.e = 15$, $b.e = 17$. În lista $L(V)$ cu galben este marcat vârful curent, în lista $L(u)$ cu albastru este marcat vârful curent.



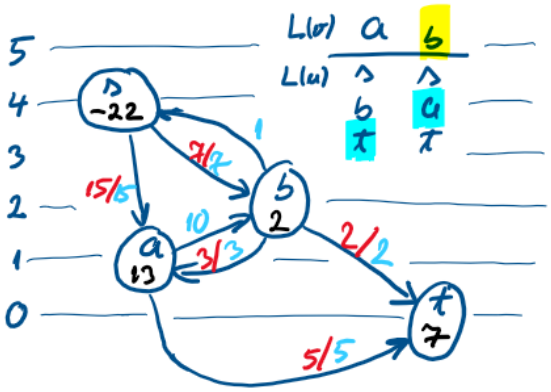
Este apelat algoritmul $DESCARCARE(a)$. Se parcurge lista $L(a)$ fără a se întâlni un arc admisibil (prin care să se poată pompa excesul de flux). În momentul când se parcurge toată lista $L(a)$ are loc operația $INALTARE(a)$. Reîncepe parcurgerea listei $L(a)$. Prin arcul (a, b) se transmit 10 unități de flux, $(a, b).f = 10$ (fluxul transmis printr-un arc este indicat în figură prin culoarea roșie, numerele colorate cu albastru reprezintă capacitatea arcelor). Supraîncărcarea vârfului a scade iar supraîncărcarea vârfului b crește



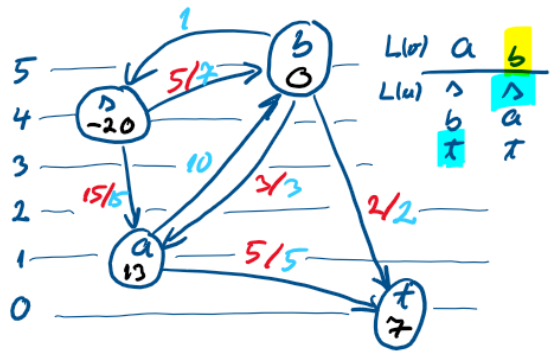
Continuă parcurgerea liste $L(a)$. Prin arcul (a, t) se transmit 5 unități de flux, $(a, b).f = 10$. Deoarece $a.e = 0$, operația $DESCARCARE(a)$ se termină. În ciclul central al algoritmului (linia 9) se constată că înălțimea actuală a lui a este mai mare decât înălțimea lui a când s-a întrat în algoritmul $DESCARCARE(a)$ și atunci vârful a este deplasat în vârful listei $L(V)$.



Se reia parcurgerea lui $L(V)$ cu succesorul vârfului a și se apelează $DESCARCARE(b)$. Începe parcurgerea listei $L(b)$ fără a descoperi un arc prin care să se poată pompa excesul de flux din b. Când se ajunge la capătul listei $L(b)$ are loc $INALTARE(b)$ și $b.h = 1$. Se reîncepe parcurgerea listei $L(b)$ și se găsește arcul (b, t) ca fiind admisibil (se poate pompa excesul de flux). Are loc pomparea fluxului $(b, t).f = 2$.

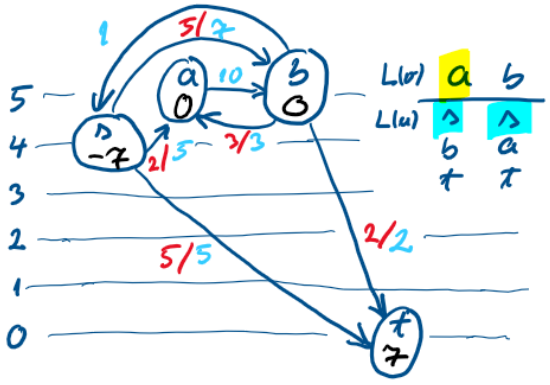


Este continuată parcurgerea listei $L(b)$, pentru că se parcurge toată lista fără să se mai găsească un arc admisibil are loc operația $INALTARE(b)$. Lista $L(b)$ este parcursă de la vârf și arcul (b, a) este găsit admisibil: $c_f(b, a) = 13 \wedge b.h = a.h + 1$. Pe arc se pompează 13 unități de flux. Ca efect, fluxul $(a, b), f$ este **anulat** și există flux $(b, a).f = 3$.



Pentru că $b.e > 0$ se continuă parcurgerea listei $L(b)$. Toate arcele $(b, x), x \in L(b)$ nu sunt admisibile (nu se mai poate pompa excesul de flux din b în alt vârf). Astfel se ajunge la finalul listei $L(b)$ și are loc $INALTARE(b)$. Pasul de parcurgere al listei $L(b)$ și $INALTARE(b)$ se repetă până când $b.h = 5$, până când înălțimea vârfului b depășește înălțimea sursei s.

Atunci când $b.h = 5$ se reia parcurgerea listei $L(b)$ de la vârf. Arcul (b, s) este admisibil: $c_f(b, s) = 8 \wedge b.h = s.h + 1$. Pe arc se pompează 2 unități de flux. Ca efect $(s, b).f = 5$. Deoarece $b.e = 0$, operația $DESCARCARE(b)$ se termină iar b este mutat în vârful listei $L(V)$.



Parcurgerea listei $L(V)$ se reia cu vârful a. Se execută $DESCARCARE(a)$ în ciclul central al algoritmului și, implicit, reîncepe parcurgerea listei $L(a)$ din punctul în care a fost lăsată la ultima revenire din $DESCARCARE(a)$. Toate arcele $(a, x), x \in L(a)$ nu sunt admisibile (nu se mai poate pompa excesul de flux din a în alt vârf), vârful a este înălțat până când $a.h = 5$.

Se reia parcurgerea listei $L(a)$ de la vârf. Arcul (a, s) este admisibil: $c_f(a, s) = 15 \wedge a.h = s.h+1$. Pe arc se pompează 13 unități de flux. Ca efect, fluxul de pe arcul (s, a) devine $(s, a).f = 2$. Deoarece $a.e = 0$, operația $DESCARCARE(a)$ se termină iar a este mutat în vârful listei $L(V)$.

Parcurgerea listei $L(V)$ continuă cu vârful b. Operația $DESCARCARE(b)$ se termină imediat și nu are nici un efect, pentru că $b.e = 0$. În final, parcurgerea listei $L(V)$ se termină. Se observă că toate vârfurile din $v \setminus \{s, t\}$ au excesul de flux $a.e = b.e = 0$. Fluxul maxim determinat în rețea este $|f| = 7$.

Soluția obținută nu este singura posibilă. Dacă ordinea vârfurilor în lista $L(V)$ sau în listele $L(u)$ diferă atunci soluția obținută poate fi alta, fluxul maxim va avea tot valoarea 7.

Algoritmul poate lucra direct pe rețeaua de flux G, nu este necesară menținerea explicită a rețelei reziduale.