Fluxuri

Capitolul 9

- **Rețele de transport**
- **Fluxuri maxime**
- **Algoritmul Ford-Fulkerson**
- **Algoritmul Edmonds-Karp**
- **Tăietura minimă**
- **Rețele de transport particulare**
- **Rezumat**
- **Implementări sugerate**
- **Probleme propuse**
- **Soluțiile problemelor**

În cadrul acestui capitol vom prezenta noțiunea de rețea de transport, vom defini fluxul maxim într-o astfel de rețea şi vom arăta modul în care poate fi determinat un astfel de flux.

De asemenea, va fi introdusă noțiunea de tăietură minimă şi vor fi prezentate câteva rețele de transport particulare.

9.1. Rețele de transport

Prin ***rețea de transport*** înțelegem un graf orientat care conține două noduri speciale: o ***sursă*** şi o ***destinație***.

Un nod sursă este un nod al grafului în care nu intră nici un arc; aşadar, gradul interior al unei surse este întotdeauna zero.

Un nod destinație este un nod al grafului din care nu iese nici un arc; aşadar, gradul exterior al unei destinații este întotdeauna zero.

De obicei, nodul sursă este notat prin *s*, în timp ce pentru nodul destinație este folosită notația *t*.

O rețea de transport este asemuită frecvent cu mai multe conducte dispuse între un robinet sursă *s* şi un canal de scurgere *t*. Conductele reprezintă arcele rețelei de transport, iar punctele de intersecție ale conductelor reprezintă nodurile rețelei.

Fiecare conductă este caracterizată printr-o ***capacitate***. Pentru exemplul considerat, capacitatea reprezintă cantitatea maximă de apă care poate să treacă prin conductă la un moment dat.

În figura 9.1 este prezentată o rețea de transport cu şapte noduri. Sursa şi destinația au fost notate prin *s*, respectiv *t*, iar celelalte noduri ale rețelei au fost numerotate de la 1 la 5.

Se observă că nu există nici un arc care ajunge la nodul sursă şi nu există nici un arc care pleacă de la nodul destinație.

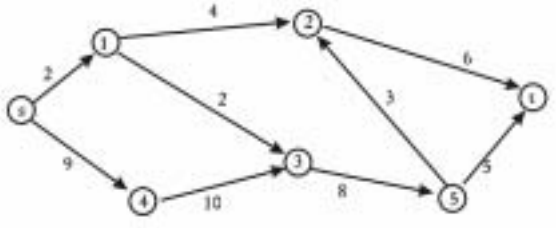


Figura 9.1: O rețea de transport

9.2. Fluxuri maxime

Păstrând exemplul sistemului de conducte, ***fluxul maxim*** este dat de cantitatea maximă de apă care poate fi pompată prin robinetul *s* astfel încât să nu se depăşească pentru nici una dintre conducte capacitatea maximă.

Din punct de vedere matematic, fluxul maxim este dat de o funcție *f* definită pe mulțimea arcelor grafului cu valori în mulțimea numerelor naturale (sau reale în anumite cazuri) care trebuie să satisfacă următoarele trei proprietăți:

- pentru fiecare arc, valoarea funcției *f* trebuie să fie mai mică sau egală cu capacitatea arcului respectiv;
- pentru fiecare nod (cu excepția sursei şi a destinației) suma valorilor funcției *f* pentru arcele care ajung la nodul respectiv trebuie să fie egală cu suma valorilor funcției *f* pentru arcele care pleacă de la nodul respectiv;
- suma valorilor funcției *f* pentru arcele care ajung la destinație trebuie să fie egală cu suma valorilor funcției *f* pentru arcele care pleacă de la sursă; această sumă este notată cu *F* şi trebuie să fie cât mai mare posibil; valoarea *F* reprezintă fluxul maxim al rețelei de transport.

De exemplu, pentru rețeaua de transport din figura 1, fluxul maxim este 10. Acesta poate fi obținut folosind funcția *f* definită astfel:

- $f(s, 1) = 2$
- $f(s, 4) = 8$
- $f(1, 2) = 2$
- $f(1, 3) = 0$
- $f(2, t) = 5$
- $f(3, 5) = 8$
- $f(4, 3) = 8$
- $f(5, 2) = 3$
- $f(5, t) = 5$

Se observă imediat că avem:

$$F = f(s, 1) + f(s, 4) = f(2, t) + f(5, t) = 2 + 8 = 5 + 5 = 10.$$

Prin studierea valorilor funcției *f* şi a capacităților arcelor, se observă imediat că şi prima condiție pe care trebuie să o satisfacă funcția *f* este îndeplinită (valorile alese pentru un arc sunt cel mult egale cu capacitatea arcului respectiv).

De asemenea, se observă că cea de-a doua condiție este îndeplinită pentru toate cele cinci noduri intermediare, deoarece avem:

- $f(s, 1) = f(1, 2) + f(1, 3) = 2$
- $f(1, 2) + f(5, 2) = f(2, t) = 5$
- $f(1, 3) + f(4, 3) = f(3, 5) = 8$
- $f(s, 4) = f(4, 3) = 8$
- $f(3, 5) = f(5, 2) + f(5, t) = 5$

9.3. Algoritmul Ford-Fulkerson

În cadrul acestei secțiuni vom descrie un prim algoritm care poate fi utilizat pentru a determina valoarea fluxului maxim într-o rețea de transport. Pentru început vom introduce noțiunile de arc special şi drum de creştere, iar apoi vom descrie algoritmul şi îi vom analiza complexitatea.

9.3.1. Arce speciale

În vederea aplicării algoritmului pe care îl vom prezenta ulterior, pentru fiecare arc (i, j) de capacitate k, vom introduce un arc (j, i) a cărui capacitate va fi 0. Un astfel de arc va fi numit ***arc special***.

Dacă unui arc îi este asociată capacitatea c atunci, inițial, costul arcului va fi c, iar costul arcului special corespunzător va fi 0. După cum vom vedea, suma capacităților celor două arce va fi întotdeauna c.

9.3.2. Drumuri de creştere

Un ***drum de creştere*** într-o rețea de transport este dat de un drum de la sursă la destinație, care conține numai arce de cost strict pozitiv (printre acestea se pot număra şi arce speciale).

De exemplu, pentru graful din figura 9.1, un posibil drum de creştere este $s - 4 - 3 - 5 - t$. Un alt drum de creştere ar putea fi $s - 1 - 3 - 5 - 2 - t$.

Valoarea unui drum de creştere va fi dată de cel mai mic cost al unui arc care face parte din drumul respectiv.

9.3.3. Prezentarea algoritmului

Practic, algoritmul *Ford-Fulkerson* constă în identificarea succesivă a unor drumuri de creştere până în momentul în care nu mai există nici un astfel de drum.

După identificarea unui drum de creştere se determină valoarea acestuia, iar aceasta se scade din costurile fiecărui arc (i, j) de pe drumul respectiv şi se adună la costurile arcelor corespunzătoare de forma (j, i). De asemenea, valoarea respectivă se adună la fluxul maxim determinat până în momentul respectiv.

De exemplu, pentru drumul de creştere s – 4 – 3 – 5 – t, avem valoarea 5. Vom scădea cu 5 costurile arcelor (s, 4), (4, 3), (3, 5) şi (5, t) şi vom creşte cu 5 costurile arcelor (4, s), (3, 4), (5, 3) şi (t, 5).

Datorită faptului că un drum de creştere conţine arce care au costuri pozitive, valoarea sa va fi întotdeauna un număr pozitiv. Ca urmare, pentru fiecare drum de creştere determinat, valoarea fluxului va creşte cu cel puţin o unitate.

Datorită faptului că avem capacităţi finite, fluxul maxim este un număr finit. Din aceste motive suntem siguri că, mai devreme sau mai târziu, algoritmul se va încheia.

Determinarea unui drum de creştere se poate realiza prin orice metodă dar, din motive de eficienţă, trebuie utilizată una al cărei ordin de complexitate este $O(M + N)$. Vom prezenta în continuare versiunea în pseudocod a algoritmului.

```
Subalgoritm Ford_Fulkerson(G)
                                                  { G – reţeaua de transport }
  creează matricea a                          { aij reprezintă capacitatea unui arc }
                                                  { de la nodul i la nodul j }
  flux_maxim ← 0
  cât timp există drumuri de creştere execută
    determină un drum de creştere D
    min ← ∝
    pentru fiecare muchie (i,j) din D execută
      dacă aij < min atunci
        min ← aij
      sfârşit dacă
    flux_maxim ← flux_maxim + min
    sfârşit pentru
    pentru fiecare muchie (i,j) din D execută
      aij ← aij - min
      aji ← aji + min
    sfârşit pentru
  sfârşit cât timp
sfârşit subalgoritm
```

Practic se va încerca la fiecare pas determinarea unui drum de creştere şi algoritmul se va opri în momentul în care nu mai poate fi găsit nici un astfel de drum.

De obicei, este necesară şi determinarea valorilor funcției *f* (cantitatea de apă care se află pe o anumită conductă – sau fluxul pe un anumit arc). Pentru aceasta este suficient să se afişeze costurile arcelor speciale. În figura 9.2 sunt prezentate costurile tuturor arcelor după încheierea execuției algoritmului *Ford-Fulkerson*. Se observă că nu mai există nici un drum de creştere şi că valoarea fluxului este 10.

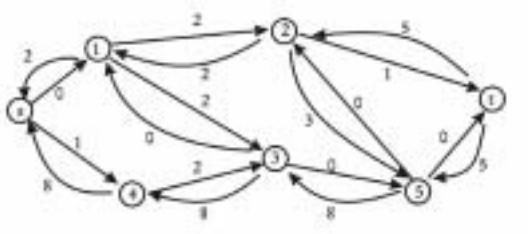


Figura 9.2: Costurile arcelor după executarea algoritmului *Ford-Fulkerson*

9.3.4. Corectitudinea algoritmului

Deşi există o demonstrație matematică a corectitudinii algoritmului prezentat, aceasta nu va fi descrisă aici, deoarece ea nu este absolut necesară pentru a înțelege mecanismul de funcționare al acestuia.

Să presupunem că, pentru graful din figura 9.1 determinăm următoarele trei drumuri de creştere: la primul pas este găsit drumul *s* – 1 – 3 – 5 – *t*, la al doilea pas drumul *s* – 4 – 3 – 5 – *t*, iar la al treilea pas *s* – 4 – 3 – 5 – 2 – *t*. În acest moment arcele grafului au costurile prezentate în figura 9.3. Se observă că nici unul dintre aceste drumuri nu conține arce speciale, iar fluxul obținut este doar 8.

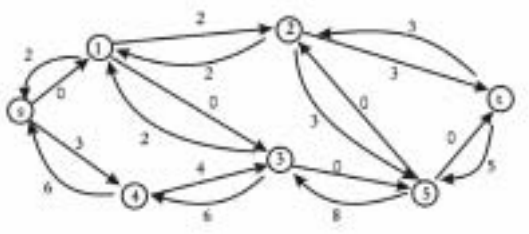


Figura 9.3: Costurile arcelor după determinarea primelor trei drumuri de creştere

De asemenea, observăm faptul că nu mai există nici un drum de creştere care nu conține arce speciale. Totuşi, după cum am arătat, fluxul maxim este 10.

În acest moment vom arăta semnificația drumurilor de creştere care conțin şi arce speciale. Pentru graful din figura 9.3. există un singur drum de creştere şi anume *s* – 4 – 3 – 1 – 2 – *t*; acest drum conține arcul special (3, 1). Revenind la similitudinea cu rețeaua de conducte, putem spune că "scoatem apa de pe conducta respectivă".

Studiind fenomenele care au loc la trecerea apei prin conducte remarcăm următoarele:

- de pe conducta (1, 3) a fost scoasă toată apa, deci prin ea nu mai trece apă;
- în nodul 1 am redirecționat fluxul de valoare 2, care ajungea prin conducta (*s*,1); astfel, el nu mai iese prin conducta (1, 3), ci prin conducta (1, 2);
- fluxul care ajunge în nodul 3 va avea în continuare valoarea 8, dar acum va ajunge în întregime pe conducta (4, 3), deoarece pe conducta (1, 3) nu mai există apă.

Aceasta este doar o demonstrație intuitivă, neriguroasă, având scopul de a arăta faptul că folosirea arcelor speciale nu duce la obținerea unor rezultate incorecte.

9.3.5. Analiza complexității

La fiecare pas se determină un drum de creştere; aşa cum am arătat anterior, ordinul de complexitate al unei astfel de operații trebuie să fie $O(M + N)$. În continuare se execută câteva operații pe baza arcelor de pe drumul de creştere. Deoarece drumul de creştere poate conține cel mult N noduri, ordinul de complexitate al acestor operații este $O(N)$. Ca urmare, ordinul de complexitate al operațiilor efectuate la un anumit pas este $O(M + N)$.

Din nefericire, algoritmul ne asigură doar faptul că la fiecare pas valoarea fluxului va creşte cu cel puțin o unitate. Aşadar, dacă fluxul maxim este F, există posibilitatea de a efectua F paşi.

Ca urmare, ordinul de complexitate al algoritmului *Ford-Fulkerson* este $O(F \cdot (M + N))$. Se observă că, în cazul în care valoarea fluxului este foarte mare, acest ordin de complexitate este inacceptabil.

Din nefericire, chiar există rețele de transport în care fluxul maxim creşte cu o unitate la fiecare pas, iar valoarea sa este foarte mare. În aceste cazuri, algoritmul este practic inutilizabil. Exemplul clasic care ilustrează această situație este prezentat în figura 9.4.

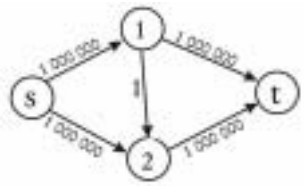


Figura 9.4: Caz în care nu poate fi utilizat algoritmul *Ford-Fulkerson*

Evident, dacă algoritmul găseşte drumurile de creştere s-1-t şi s-2-t, se vor executa doar doi paşi.

Totuşi, algoritmul nu impune nici o restricție asupra drumurilor identificate, deci poate fi ales la primul pas drumul $s - 1 - 2 - t$. La al doilea pas se poate alege drumul $s - 2 - 1 - t$, la al treilea din nou $s - 1 - 2 - t$ şi aşa mai departe. Execuția algoritmului se va încheia după două milioane de iterații în loc de două.

9.4. Algoritmul Edmonds-Karp

Pentru a evita situațiile care pot apărea în cazul rețelelor de transport de tipul celor din figura 9.4, trebuie găsită o modalitate eficientă de determinare a drumurilor de creştere în rețeaua de transport.

Practic, algoritmul *Edmonds-Karp* reprezintă doar o implementare eficientă a algoritmului *Ford-Fulkerson*.

Ideea care stă la baza algoritmului este de a identifica la fiecare pas un drum de creştere care conține un număr minim de arce. După cum vom arăta în continuare, o astfel de alegere ne asigură că se vor efectua cel mult $O(M \cdot N)$ iterații.

Fiecare drum de creştere conține, aşa cum rezultă din definiție, cel puțin un ***arc critic*** (arcul care dă valoarea drumului). Să presupunem că arcul (i, j) apare pentru prima

dată ca arc critic într-un drum de creştere în momentul în care acesta se află la distanța d de sursă (este al d-lea arc de pe drumul de creştere). Se poate arăta faptul că, alegând de fiecare dată un drum de creştere cu număr minim de muchii, în momentul în care arcul (i, j) va apărea pentru a doua oară ca arc critic, el se va afla la o distanță de cel puțin $d + 2$ de sursă. Din nou nu vom prezenta demonstrația matematică a acestui fapt, ea fiind irelevantă în acest context (ne interesează doar faptul că afirmația anterioară este adevărată). Ca urmare, fiecare arc (i, j) va putea fi arc critic de cel mult $[N / 2]$ ori pe parcursul executării algoritmului, aşadar vom avea cel mult $O(N)$ drumuri de creştere caracterizate prin arcul critic (i, j). Deoarece avem $2 \cdot M$ arce (incluzându-le pe cele speciale), în total vor exista cel mult $O(M \cdot N)$ drumuri de creştere, deci ordinul de complexitate al algoritmului *Edmonds-Karp* va fi $O((M + N) \cdot M \cdot N)$, deoarece parcurgerea în lățime necesară identificării unui drum de creştere are ordinul de complexitate $O(M + N)$.

9.5. Tăietura minimă

În cadrul acestei secțiuni vom introduce noțiunea de tăietură minimă a unei rețele de transport, vom prezenta teorema flux maxim – tăietură minimă şi vom arăta modul în care poate fi determinată o astfel de tăietură.

9.5.1. Noțiuni teoretice

Să considerăm o rețea de transport G. Vom partiționa mulțimea nodurilor în două mulțimi disjuncte astfel încât sursa şi destinația să nu facă parte din aceeaşi mulțime. Fiecare nod al grafului (rețelei) trebuie să facă parte din una dintre cele două mulțimi.

Mulțimea arcelor care pornesc dintr-un nod care face parte din mulțimea corespunzătoare sursei şi ajung într-un nod care face parte din mulțimea corespunzătoare destinație formează o ***tăietură*** a rețelei de transport.

Revenind la similitudinea cu sistemul de conducte, o tăietură reprezintă o mulțime de conducte a căror astupare face ca apa să nu poată ajunge de la sursă la destinație.

Evident, ***tăietura minimă*** va fi cea pentru care suma capacităților arcelor care o formează este cea mai mică.

9.5.2. Flux maxim – tăietură minimă

În cele ce urmează vom prezenta una dintre cele mai interesante teoreme ale teoriei grafurilor şi anume cea care prezintă relația dintre valoarea fluxului maxim şi cea a tăieturii minime.

Vom considera o rețea de transport G al cărei flux maxim este F. În această situație următoarele trei afirmații sunt echivalente:

1. F este fluxul maxim al rețelei de transport G.

2. Rețeaua de transport G nu conține nici un alt drum de creştere, cu excepția celor identificate pentru determinarea fluxului F.
3. Există o tăietură a cărei capacitate este F.

Pentru a demonstra echivalența, vom arăta că prima afirmație o implică pe a doua, a doua pe a treia, iar a treia pe prima. Este evident că, în această situație, a doua afirmație o implică pe prima (prin "intermediul" celei de-a treia), a treia o implică pe a doua (prin intermediul primeia) şi prima o implică pe a treia (prin intermediul celei de-a doua).

Prima implicație este uşor de demonstrat prin metoda reducerii la absurd. Dacă ar exista încă un drum de creştere, acesta ar avea o valoare pozitivă x, deci fluxul maxim ar creşte cu x, ceea ce contravine afirmației inițiale potrivit căreia F este fluxul maxim în rețeaua respectivă.

Pentru a demonstra cea de-a doua implicație, vom considera mulțimea S ca fiind formată din toate vârfurile la care se mai poate ajunge pornind de la sursă. Evident, destinația nu va face parte din această mulțime deoarece, în caz contrar, ar mai exista un drum de creştere, ceea ce contravine ipotezei. Ca urmare, am obținut o tăietură validă a grafului. Remarcăm faptul că pentru fiecare arc (i, j) al tăieturii, fluxul pe acel arc este egal cu capacitatea sa deoarece, în caz contrar, s-ar fi putut ajunge de la sursă la nodul j, deci şi acesta ar fi făcut parte din S. Pentru ca apa să poată ajunge la destinație, fluxul total al conductelor care ies din mulțimea S trebuie să fie egal cu fluxul maxim al rețelei, deci capacitatea tăieturii minime este egală cu fluxul maxim. (Am recurs din nou la exemplul utilizat în cadrul acestui capitol pentru a evita folosirea unor formule matematice relativ complicate şi practic inutile.)

Pentru a demonstra cea de-a treia implicație trebuie să observăm un fapt evident, şi anume că fluxul maxim va fi întotdeauna mai mic decât capacitatea unei tăieturi (în caz contrar acesta nu ar mai fi tăietură sau, pentru exemplul nostru, nu ar mai putea împiedica toată apa să ajungă la destinație). Folosind această observație rezultă imediat că F este fluxul maxim, deoarece această valoare este egală cu capacitatea tăieturii minime.

9.5.3. Determinarea tăieturii minime

Teorema prezentată anterior ilustrează şi modul în care poate fi determinată tăietura minimă a unei rețele de transport. Se aplică un algoritm de determinare a fluxului maxim şi apoi, în momentul în care nu mai există nici un drum de creştere, se identifică nodurile la care se poate ajunge pornind de la sursă. Arcele care pleacă de la un astfel de nod, dar nu ajung tot la un astfel de nod, vor forma tăietura minimă.

Aşa cum am demonstrat prin teorema precedentă, capacitatea totală a tăieturii minime este egală cu valoarea fluxului maxim prin rețeaua respectivă.

9.6. Rețele de transport particulare

În această secțiune vom prezenta câteva rețele de transport particulare (unele dintre ele nu respectă definiția de la începutul acestui capitol, dar cunoaşterea lor este utilă în rezolvarea unor probleme).

9.6.1. Fluxuri de cost minim

În unele situații, există posibilitatea asocierii unor costuri pentru arcele grafului (pe lângă capacitate). În această situație va trebui să determinăm un flux maxim de cost minim. Cu alte cuvinte, costul total al muchiilor care fac parte din flux (valoarea funcției f pentru aceste muchii nu este 0) trebuie să fie minim.

Pentru a determina un astfel de flux de cost minim este suficient să folosim o variantă a algoritmului *Ford-Fulkerson* în cadrul căreia să determinăm la fiecare pas drumuri de creştere de cost minim.

9.6.2. Surse şi/sau destinații multiple

Există situații în care rețelele de transport au mai multe surse şi/sau destinații. Pentru a determina fluxul maxim într-o astfel de rețea vom introduce o ***supersursă*** şi/sau o ***superdestinație***.

O supersursă este un nod din care pleacă arce de capacitate infinită spre toate sursele. Similar, superdestinația este un nod la care sosesc arce de capacitate infinită de la toate destinațiile.

În acest moment avem o rețea de transport clasică ce conține o singură sursă şi o singură destinație. Fluxul maxim nu va fi influențat de capacitățile infinite (în practică se vor utiliza, de fapt, valori foarte mari), deoarece el va fi limitat de capacitățile arcelor din rețeaua propriu-zisă.

9.6.3. Capacități în noduri

În unele situații există posibilitatea de a se impune restricții referitoare la capacitatea unui nod (într-un nod poate intra o cantitate de apă cel mult egală cu capacitatea sa). În acest caz, fiecare nod x va fi înlocuit prin două noduri x_1 şi x_2, între care va exista un singur arc de capacitate egală cu capacitatea nodului x. Toate arcele care ajungeau la nodul x vor ajunge acum la nodul x_1, iar toate arcele care plecau de la nodul x vor pleca acum de la nodul x_2. Astfel, am redus problema la determinarea fluxului maxim într-o rețea de transport cu capacități doar pe arce, dar care conține un număr dublu de noduri.

9.7. Rezumat

Acest capitol a fost dedicat prezentării noțiunii de flux maxim. Pentru început am definit rețelele de transport, iar apoi fluxurile. În continuare am arătat modul în care pot

fi determinate fluxurile maxime în rețelele de transport folosind algoritmul *Ford-Fulkerson*. De asemenea, am arătat faptul că acest algoritm devine ineficient în anumite situații.

În continuare am arătat modul în care pot fi evitate problemele apărute în cazul algoritmului *Ford-Fulkerson* şi am descris algoritmul *Edmonds-Karp*. Am efectuat o analiză a complexității celor doi algoritmi care pot fi utilizați pentru determinarea fluxului maxim.

De asemenea, în acest capitol am introdus noțiunea de tăietură minimă, am prezentat teorema „flux maxim – tăietură minimă" şi am descris modalitatea prin care poate fi determinată o tăietură minimă folosind algoritmul *Ford-Fulkerson*.

În final am prezentat câteva rețele de transport particulare, şi anume cele în cadrul cărora arcelor le sunt asociate costuri şi se doreşte determinarea unui flux maxim de cost minim, cele care conțin mai multe surse şi/sau mai multe destinații, precum şi cele în care sunt asociate capacități şi pentru noduri.

9.8. Implementări sugerate

Pentru a vă familiariza cu modul în care trebuie implementate rezolvările problemelor ale căror soluții necesită cunoştințe referitoare la rețelele de transport, vă sugerăm să încercați să realizați implementări pentru:

1. determinarea fluxului maxim într-o rețea de transport, folosind algoritmul *Ford-Fulkerson*;
2. determinarea fluxului maxim într-o rețea de transport, folosind algoritmul *Edmonds-Karp*;
3. determinarea unei tăieturi minime într-o rețea de transport;
4. determinarea fluxului maxim de cost minim folosind algoritmul *Bellman-Ford* pentru identificarea drumurilor de creştere;
5. determinarea fluxului maxim într-o rețea de transport cu mai multe surse şi mai multe destinații;
6. determinarea tăieturii minime într-o rețea de transport cu mai multe surse şi mai multe destinații;
7. determinarea fluxului maxim într-o rețea de transport în care au fost asociate capacități şi nodurilor;
8. determinarea simultană a fluxului maxim şi a tăieturii minime.

9.9. Probleme propuse

În continuare vom prezenta enunțurile câtorva probleme pe care vi le propunem spre rezolvare. Rezolvarea acestor probleme necesită, pe lângă informațiile prezentate în cadrul acestui capitol, cunoştințe referitoare la determinarea unor drumuri în grafuri.

9.9.1. Intranet

Descrierea problemei
O companie are la dispoziție o rețea locală formată din N calculatoare între care există un număr total de M legături cu rate de transfer diferite. Legăturile sunt de tip simplex, adică informațiile pot circula într-un singur sens. Se doreşte transmiterea unui volum mare de date de la calculatorul identificat prin 1, la calculatorul identificat prin N. Va trebui să determinați care este dimensiunea maximă a datelor care poate fi transmisă inițial pentru ca în rețea să nu apară congestii. Durata unui transfer este foarte mică, motiv pentru care vom considera că intervalul de timp necesar comunicării între oricare două calculatoare este 0.

Date de intrare
Prima linie a fişierului de intrare **INTRANET.IN** conține numărul N al calculatoarelor din rețea şi numărul M al legăturilor directe dintre acestea. Aceste numere vor fi separate printr-un spațiu. Fiecare dintre următoarele M linii va conține câte trei numere întregi x, y şi c cu semnificația: există o legătură directă de la calculatorul identificat prin x la calculatorul identificat prin y, iar rata de transfer a acestei legături este de c Mbps.

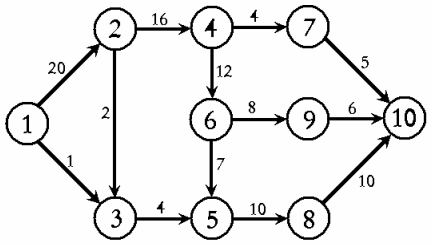
Date de ieşire
Fişierul de ieşire **INTRANET.OUT** va conține o singură linie pe care se va afla volumul maxim de date (exprimat în Mb) care poate fi transmis fără a apărea congestii.

Restricții şi precizări
- $1 \le N \le 100$;
- $1 \le M \le 1000$;
- rata de transfer a unei legături este un număr întreg cuprins între 1 şi 1000000;
- există cel mult o legătură directă de la un calculator x la un calculator y;
- dacă există o legătură directă de la un calculator x la un calculator y, atunci nu poate exista şi o legătură directă de la calculatorul y la calculatorul x;
- datele vor putea ajunge întotdeauna de la calculatorul identificat prin 1, la calculatorul identificat prin N.

Exemplu

INTRANET.IN	INTRANET.OUT
10 13 1 2 20 1 3 1 2 3 2 2 4 16 3 5 4	19

```
4 6 12
4 7 4
5 8 10
6 5 7
6 9 8
7 10 5
8 10 10
9 10 6
```



Timp de execuție: 1 secundă/test

9.9.2. Bomboane

Descrierea problemei

La o fabrică de bomboane, sunt puse pe un stand *n* cutii de bomboane. Fiecare din cele *n* cutii conține exact *m* bomboane dintr-un singur sortiment. Cele *n* cutii conțin bomboane din sortimente diferite (aşadar există un număr total de *n* sortimente). Un angajat mai distrat începe să se amuze şi amestecă bomboanele din cutii. Spre a nu fi observată modificarea, el are grijă ca în fiecare cutie să rămână câte *m* bomboane.

Problemele încep să apară atunci când angajatului i se cere să aducă câte o bomboană din fiecare cutie, deci din fiecare sortiment. Fiind urmărit de către superiori, el nu va avea voie să extragă decât o bomboană din fiecare cutie.

Angajatul acordă fiecărei extrageri un grad de risc. Astfel, la extragerea unei bomboane din sortimentul cu numărul *i* din cutia care conținea inițial sortimentul cu numărul *j*, gradul de risc va fi valoarea absolută a diferenței dintre *i* şi *j*. Gradul de risc al tuturor celor *n* extrageri va fi egal cu suma riscurilor fiecăreia.

Să se determine ce sortiment de bomboană va trebui să extragă muncitorul din fiecare cutie pentru ca gradul de risc total să fie minim.

Date de intrare

Prima linie a fişierului **`BOMBOANE.IN`** conține numerele *n* şi *m*, separate printr-un singur spațiu. Pe următoarele *n* linii se află câte *m* numere, despărțite printr-un spațiu. Cele *m* numere reprezintă sortimentele bomboanelor care se regăsesc în fiecare cutie după amestecare, în ordine, începând cu cutia având numărul de ordine 1 până la cutia având numărul de ordine *n*.

Date de ieşire

Fişierul de ieşire **`BOMBOANE.OUT`** va conține două linii. Pe prima dintre ele se va afla un număr natural reprezentând gradul total minim de risc. Cea de-a doua linie va conține *n* numere naturale, despărțite prin câte un spațiu, reprezentând numărul de

ordine al cutiei din care va fi extrasă bomboana din fiecare sortiment, în ordine, începând cu sortimentul 1 până la sortimentul n.

Restricții și precizări

- $1 \leq n \leq 100$;
- $1 \leq m \leq 1000$;
- sortimentele şi cutiile sunt identificate prin numere întregi cuprinse între 1 şi N;
- în cazul în care există mai multe posibilități de extragere cu risc total minim, se va determina doar una dintre ele.

Exemplu

BOMBOANE.IN
```
7 3
1 2 7
6 6 4
4 7 3
4 2 3
2 1 3
7 5 1
5 5 6
```

BOMBOANE.OUT
```
10
1 5 3 4 7 2 6
```

Timp de execuție: 5 secunde/test

9.9.3. Cezar

Pe o tablă sunt desenate N cercuri. Cezar trebuie să deseneze un număr total de M săgeți, astfel încât fiecare săgeată pornească de la un cerc şi să ajungă la alt cerc. Pentru fiecare cerc se cunoaşte numărul săgeților care trebuie să plece din cercul respectiv şi numărul săgeților care trebuie să ajungă la cercul respectiv.

Va trebui să verificați dacă Cezar poate desena săgețile şi, dacă acest lucru este posibil, să descrieți modul în care acestea vor fi desenate.

Date de intrare

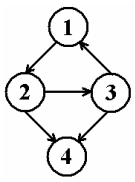
Prima linie a fişierului de intrare **CEZAR.IN** conține numărul N al cercurilor şi numărul M al săgeților. Cea de-a doua linie conține N numere întregi, separate prin câte un spațiu, care reprezintă numărul săgeților care trebuie să pornească de la fiecare cerc. Cea de-a treia linie conține N numere întregi, separate prin câte un spațiu, care reprezintă numărul săgeților care trebuie să ajungă la fiecare cerc. Primul număr de pe fiecare dintre aceste linii corespunde primului cerc (identificat prin 1), al doilea număr corespunde celui de-al doilea cerc (identificat prin 2) etc. Suma numerelor de pe fiecare dintre aceste două linii este întotdeauna M.

Date de ieşire
Prima linie a fişierului de ieşire **CEZAR.OUT** va conține mesajul DA în cazul în care există cel puțin o posibilitate de desenare a săgeților şi mesajul NU în caz contrar. În cazul în care săgețile pot fi desenate, fişierul va mai conține M linii pe care se vor afla câte două numere întregi x şi y cu semnificația: va fi desenată o săgeată care porneşte de la cercul identificat prin x şi ajunge la cercul identificat prin y.

Restricții şi precizări
- $1 \leq N \leq 100$;
- $1 \leq M \leq 1000$;
- cercurile sunt identificate prin numere cuprinse între 1 şi N;
- poate fi desenată cel mult o săgeată care poneşte de la un cerc x şi ajunge la un cerc y;
- dacă s-a desenat o săgeată care poneşte de la un cerc x şi ajunge la un cerc y, atunci poate sau nu fi desenată şi o săgeată care poneşte de la cercul y şi ajunge la cercul x;
- dacă există mai multe soluții atunci poate fi aleasă oricare dintre ele.

Exemplu

CEZAR.IN	**CEZAR.OUT**
4 5	DA
1 2 2 0	1 2
1 1 1 2	2 3
	3 1
	2 4
	3 4



Timp de execuție: 1 secundă/test

9.10. Soluțiile problemelor

Vom prezenta acum soluțiile problemelor propuse în cadrul secțiunii precedente. Pentru fiecare dintre acestea va fi descrisă metoda de rezolvare şi va fi analizată complexitatea algoritmului prezentat.

9.10.1. Intranet

Rețeaua de calculatoare poate fi privită ca fiind o rețea de transport în care nodurile reprezintă calculatoarele, iar arcele reprezintă legăturile dintre acestea. Capacitatea unui arc va fi dată de rata de transfer a legăturii corespunzătoare.

Sursa rețelei de transport va fi dată de nodul corespunzător calculatorului identificat prin 1, iar destinația sa va fi dată de nodul corespunzător calculatorului identificat prin *N*.

În aceste condiții, problema se reduce la determinarea unui flux în această rețea de transport. Valoarea fluxului va reprezenta dimensiunea maximă a datelor care pot fi transmise fără a apărea congestii. Această valoare va fi scrisă în fișierul de ieșire.

Analiza complexității

Citirea datelor de intrare implică citirea celor *M* muchii ale rețelei de transport şi a capacităților acestora; aşadar ordinul de complexitate al acestui subalgoritm este **O(M)**. În paralel cu citirea se realizează crearea structurii de date în care este memorată rețeaua, ordinul de complexitate al acestei operații fiind tot **O(M)**.

După crearea rețelei de transport vom aplica algoritmul de determinare a fluxului maxim, al cărui ordin de complexitate este $\mathbf{O((M + N) \cdot M \cdot N)}$.

Datele de ieşire constau într-o singură valoare, deci operația de scriere a acestora are ordinul de complexitate **O(1)**.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(M) + O(M) + O((M + N) \cdot M \cdot N) + O(1) = \mathbf{O((M + N) \cdot M \cdot N)}$.

9.10.2. Bomboane

În vederea rezolvării acestei probleme vom crea o rețea de transport astfel: introducem artificial o sursă şi o destinație; pe lângă aceste două noduri, rețeaua va mai conține alte $2 \cdot n$ noduri (dublul numărului de sortimente distincte). Vom uni pe rând sursa de fiecare dintre primele *n* noduri (reprezentând sortimentele distincte de bomboane, numerotate de la 1 la *n*), considerând că arcele introduse au capacitatea 1 şi costul 0. Celelalte *n* noduri (care reprezintă cele *n* cutii, şi care sunt numerotate de la $n + 1$ la $2 \cdot n$) vor fi unite cu destinația prin arce de capacitate 1 şi cost 0.

Va exista un arc de la un nod *i* ($1 \le i \le n$) la un nod *j* ($n + 1 \le j \le 2 \cdot n$) dacă şi numai dacă bomboana din sortimentul *i* se află în cutia $j - n$ după amestecare. Acest arc va avea capacitatea 1 şi costul egal cu valoarea absolută dintre $j - n$ şi *i* (valoarea *n* este scăzută datorită faptului că am numerotat cutiile începând cu $n + 1$).

După construirea rețelei este suficient să determinăm un flux maxim de cost minim pentru a obține soluția problemei.

Fluxul va avea întotdeauna valoarea *n*, iar o bomboană va fi extrasă dintr-o anumită cutie doar dacă există flux pe arcul care leagă nodul corespunzător bomboanei (sortimentului) de nodul corespunzător cutiei.

După determinarea fluxului se afişează costul acestuia, precum şi alegerile efectuate.

Analiza complexității

Ordinul de complexitate al operație de citire a datelor de intrare este **O(m · n)** deoarece sunt citite *m* linii, fiecare conținând *n* numere.

Pentru crearea rețelei de transport va trebui, mai întâi, să construim cele $2 \cdot n + 2$ noduri ale acesteia, operație al cărei ordin de complexitate este $O(n)$. Pentru a lega primele *n* noduri de sursă avem nevoie de un timp de ordinul $O(n)$; acelaşi ordin are şi timpul necesar legării celorlalte *n* noduri de destinație. De fiecare dintre ultimele *n* noduri vor fi legate *m* dintre primele *n*; ordinul de complexitate al operației pentru un nod este $O(m)$, deci pentru toate nodurile obținem un timp de ordinul $O(m \cdot n)$. Aşadar, ordinul de complexitate al operației de construire a rețelei de transport este $O(n) + O(n) + O(n) + O(m \cdot n) = \mathbf{O(m \cdot n)}$.

Ordinul de complexitate al algoritmului de determinare al fluxului maxim de cost minim depinde de modul în care se determină drumurile de cost minim.

Pentru a scrie datele de ieşire va trebui doar să verificăm care dintre arce conțin fluxuri nenule. Deoarece avem $m \cdot n$ astfel de arce, ordinul de complexitate al operației de creare a fişierului de ieşire va fi **O(m · n)**.

Putem trage concluzia că ordinul de complexitate al algoritmului de rezolvare a acestei probleme depinde doar de modul în care se determină fluxul, deoarece această operație va avea un ordin de complexitate superior față de cel al celorlalte operații (toate celelalte operații au ordinul de complexitate cel mult $O(m \cdot n)$).

9.10.3. Cezar

În cazul în care considerăm că cercurile sunt vârfurile unui graf orientat, problema se reduce la alegerea unei configurații a arcelor, astfel încât fiecare nod să aibă un grad exterior dat (numărul săgeților care pleacă din cercul corespunzător) şi un grad interior dat (numărul săgeților care ajung la cercul corespunzător).

Pentru aceasta vom crea o rețea de transport care va conține $2 \cdot N + 2$ noduri. Două dintre acestea vor fi sursa şi destinația. *N* dintre noduri vor fi legate de sursă prin arce ale căror capacități vor fi egale cu gradele interioare ale nodurilor. Celelalte *N* noduri vor fi legate de destinație prin arce ale căror capacități vor fi egale cu gradele exterioare ale nodurilor. De la fiecare dintre primele *N* noduri vor pleca arce de capacitate 1 spre fiecare dintre celelalte *N* noduri.

În această rețea vom determina fluxul maxim. Dacă acesta va fi *M*, atunci săgețile pot fi desenate (graful poate fi construit). Arcele rețelei de transport pe care există flux vor indica modul în care vor fi desenate săgețile (configurația arcelor).

Analiza complexității

Datele de intrare constau din $2 \cdot N + 2$ numere, deci ordinul de complexitate al operației de citire a acestora va fi **O(N)**.

Pentru crearea rețelei de transport va trebui, mai întâi să construim cele $2 \cdot N + 2$ noduri ale acesteia, operație al cărui ordin de complexitate este $O(N)$. Pentru a lega

primele N noduri de sursă avem nevoie de un timp de ordinul $O(N)$; acelaşi ordin are şi timpul necesar legării celorlalte N noduri de destinaţie. Fiecare dintre primele N noduri va fi legat de fiecare dintre ultimele N (cu o singură excepţie – nu va exista un arc de la nodul i la nodul $N + i$, deoarece o săgeată trebuie să pornească de la un cerc şi să ajungă la un alt cerc); ordinul de complexitate al operaţiei pentru un nod este $O(N)$, deci pentru toate nodurile obţinem un timp de ordinul $O(N^2)$. Aşadar, ordinul de complexitate al operaţiei de construire a reţelei de transport este $O(N) + O(N) + O(N) + O(N^2) = \mathbf{O(N^2)}$.

Vom observa că numărul muchiilor reţelei de transport va fi $N + N + N \cdot (N - 1) = N \cdot (N + 1)$. În această reţea va trebui să determinăm un flux maxim. Datorită faptului că ştim că acesta va fi cel mult M, putem folosi algoritmul *Ford-Fulkerson* al cărui ordin de complexitate va fi $\mathbf{O}(M \cdot (N + N \cdot (N + 1))) = \mathbf{O(M \cdot N^2)}$, deoarece fluxul maxim va fi M, iar numărul de muchii va fi $N \cdot (N + 1)$. Dacă am alege algoritmul *Edmonds-Karp*, am avea, teoretic, ordinul de complexitate $O((N \cdot (N + 1) + \mathrm{N}) \cdot N \cdot (N + 1) \cdot N) = O(N^5)$. Totuşi, datorită parcurgerii în lăţime, fiecare drum are, de fapt, lungimea 3 în foarte multe situaţii (marea majoritate), deci ordinul real de complexitate în cazul mediu va fi, de fapt, $\mathbf{O(N^3)}$.

După determinarea fluxului va trebui doar să verificăm arcele care conţin flux nenul şi să scriem datele în fişierul de ieşire. Ordinul de complexitate al acestei operaţii este $\mathbf{O(N^2)}$.

În concluzie, algoritmul de rezolvare a acestei probleme pentru cazul mediu are ordinul de complexitate $O(N) + O(N^2) + O(M \cdot N^2) + O(N^2) = \mathbf{O(M \cdot N^2)}$ dacă fluxul este determinat cu ajutorul algoritmului *Ford-Fulkerson* şi $O(N) + O(N^2) + O(N^3) + O(N^2) = \mathbf{O(N^3)}$ dacă se utilizează algoritmul *Edmonds-Karp*.