9. PRELUCRAREA ARBORILOR ŞI A GRAFURILOR

9.1.Grafuri şi arbori

Un ***graf neorientat*** este o pereche ordonată G = (X,U), unde X este o mulțime finită şi nevidă de elemente numite ***noduri*** (sau ***vârfuri***), iar U este o mulțime de perechi (neordonate) de elemente distincte ale lui X, numite ***muchii***. O muchie având vârfurile i şi j (numite ***extremitățile*** sale) este notată prin [i, j] sau [j, i].

Exemplu 1. Graful neorientat G = (X = {1,2,3,4,5,6,7,8}, U={[1,2], [1,5], [2,3], [2,4], [2,6], [3,6], [4,5], [7,8]}) poate fi reprezentat grafic ca în figura 1.

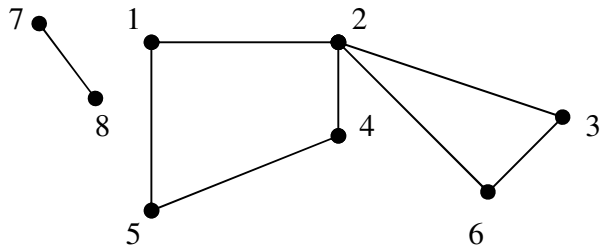


Figura 1

Un ***subgraf*** al unui graf G = (X,U) este un graf H = (Y, V), unde Y ⊂ X, iar V este formată din toate muchiile lui U care unesc vârfuri din Y. Astfel H = ({1,2,3,4}, {[1,2], [2,3], [2,4]}) este un subgraf al grafului din figura 1. Un ***graf parțial*** al unui graf G = (X,U) este un graf H = (X, V) cu V ⊂ U. Astfel H = ({1,2,3,4,5,6,7,8}, {[1,2], [2,3], [2,6] [3,6], [4,5], [7,8]}) este un subgraf al grafului din figura 1.

Numim ***lanț*** într-un graf o succesiune de muchii $[i_1,i_2]$, $[i_2,i_3]$,…, $[i_{n-1},i_n]$, notată prescurtat $[i_1, i_2, \ldots,i_n]$. ***Lungimea lanțului*** este definită ca fiind numărul muchiilor care intră în componența sa. Astfel în graful din figura 1, [1, 2, 3, 6] este un lanț. Un lanț $[i_1, i_2, \ldots,i_n]$ pentru care $i_1, i_2, \ldots,i_n$ sunt distincte se numeşte ***lanț elementar***.

Un ***ciclu elementar*** este un lanț $[i_1, i_2, \ldots,i_n, i_1]$ cu proprietatea că $[i_1, i_2, \ldots,i_n]$ este lanț elementar şi n ≥ 3. Un lanț $[i_1, i_2, \ldots,i_n, i_1]$ în care muchiile sunt diferite două câte două se

numeşte ***ciclu***. În graful din figura 1 [1, 2, 4, 5 ,1] este un ciclu elementar, iar [2, 1, 5 , 4, 2, 6, 3, 2] este ciclu (care nu este elementar).

Un vârf care este extremitatea unei singure muchii se numeşte vârf terminal. Două vârfuri unite printr-o muchie se numesc vârfuri adiacente.

Un graf neorientat se numeşte conex dacă oricare două vârfuri diferite sunt unite ale sale prin cel puţin un lanţ. Graful din figura 1 nu este conex (de exemplu, vârfurile 1 şi 7 nu sunt unite prin nici un lanţ). În cazul unui graf neconex se pune problema determinării componentelor sale conexe; o componentă conexă este un subgraf conex maximal, i.e. un subgraf în care nici un vârf al său nu este unit cu nici un vârf din afara subgrafului printr-o muchie. Se observă că împărţire unui subgraf în componenetele sale conexe determină o partiţie a mulţimii vârfurilor. De exemplu, pentru graful din figura 1 se obţine $X = \{1,2,3,4,5,6\} \cup \{7,8\}$.

Un ***graf orientat*** este o pereche ordonată G = (X,U), unde X este o mulţime finită şi nevidă de elemente numite ***noduri*** (sau ***vârfuri***), iar U este o mulţime de perechi ordonate de elemente distincte ale lui X, numite ***arce***. Un arc având vârfurile i şi j (numite ***extremităţile*** sale) se notează prin (i, j). Deci, deosebirea faţă de grafurile neorientate constă în faptul că fiecare arc (i, j) are un sens de parcurgere şi anume de la ***extremitatea sa iniţială*** la ***extremitatea sa finală***.

Exemplu 2. Graful orientat G = (X = {1,2,3,4,5,6,7,8}, U={(1,2), (5,1), (2,3), (2,4), (6,2), (3,6), (4,5), (8,7)}) poate fi reprezentat grafic ca în figura 2.

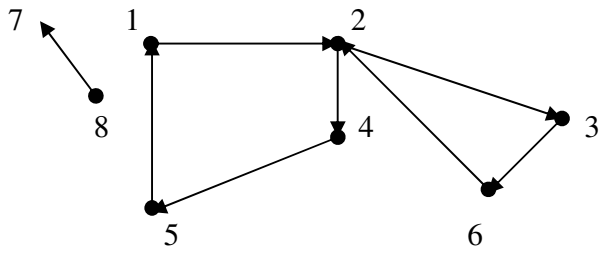


Figura 2

În cazul grafurilor orientate, noţiunile de lanţ şi ciclu îşi au corespondentul în noţiunile de ***drum*** şi ***circuit***. Astfel în graful din figura 2:

- (1,2,3,6,2,4) este un drum
- (1,2,3,6) este un drum elementar
- (1,2,3,6,2,4,5,1) este un circuit
- (1,2,4,5,1) este un circuit elementar

În cele ce urmează vom considera problema aflării drumurilor elementare dintre două vârfuri şi a circuitelor elementare dintre două vârfuri. Mulţimea vârfurilor grafului se poate defini cu ajutorul predicatului multime_varfuri([lista de vâfuri]), iar un arc de la i la j cu ajutorul predicatului arc1(i,j). În programul următor se presupune că graful este definit în fişierul "graf.dat". Acest fişier trebuie să conţină declaraţia predicatelor multime_varfuri şi arc1 sub forma:

```
predicates
        arc1(varf,varf)
        multime_varfuri(lista_varfuri)
```

şi să conţină în secţiunea clauses faptele care definesc mulţimea de vârfuri şi arcele grafului considerat.

```
        domains
            varf=symbol
            lista_varfuri=varf*
      include "graf.dat"
        predicates
            apartine(varf,lista_varfuri)
            nr_elem(lista_varfuri,integer)
            invers(lista_varfuri,lista_varfuri)
            inv(lista_varfuri,lista_varfuri,lista_varfuri)
            afis(lista_varfuri)
            nod_graf(varf)
            nod_initial(varf)
            extremitati(varf,varf)
            drum(varf,varf,lista_varfuri,lista_varfuri)
            drum_elem(varf,varf,lista_varfuri)
            circuit_elem(varf,lista_varfuri)
            fer
            meniu
            selectie(char)
```

```
    start
clauses
    apartine(X,[X|_]).
    apartine(X,[_|Y]):-apartine(X,Y).
    nr_elem([],0).
    nr_elem([_|X],N):-nr_elem(X,N1),N=N1+1.
    invers(X,Y):-inv(X,[],Y).
    inv([],X,X).
    inv([H|X],Y,Z):-inv(X,[H|Y],Z).
    afis([X]):-write(X),!.
    afis([H|X]):-write(H," -> "),afis(X).
    fer:-makewindow(1,113,36,"Drumuri si circuite elementare",0,0,25,80).
    nod_graf(X):-multime_varfuri(L),apartine(X,L).
    extremitati(X,Y):-write("Nodul initial = "), readln(X),
                write("Nodul final = "), readln(Y),
                write("\nDrumurile elementare de la ",X," la ",Y, ":\n").
    nod_initial(X):-write("Nodul initial = "),readln(X),
                    write("\nCircuitele elementare cu nodul initial (si final) ",X,":\n").
    drum(X,X,Y,Z):-invers(Y,Z).
    drum(X,Z,L,P):-arc1(X,Y),not(apartine(Y,L)),drum(Y,Z,[Y|L],P).
    drum_elem(X,Y,L):-drum(X,Y,[X],L).
    circuit_elem(X,[X|L]):-drum(X,X,[],L),nr_elem(L,N),N>1.
    meniu:-cursor(5,10),
      write("t --> pentru afisarea tuturor drumurilor elementare"),
      cursor(7,10),
      write("d --> pentru afisarea  drumurilor elementare dintre doua noduri"),
      cursor(9,10),
      write("z --> pentru afisarea tuturor circuitelor elementare"),
       cursor(11,10),
      write("c --> pentru afisarea circuitelor elementare cu nod initial fixat"),
      cursor(13,10),
       write("e --> pentru iesire  "),
```

```
    readchar(X),clearwindow,X<>'e',selectie(X),meniu.
meniu.
selectie('t'):- write("Drumuri elementare:\n"),
                nod_graf(X),nod_graf(Y),
                write("apasati orice tasta...\n\n"),readchar(_),
                write("Drumuri elementare de la ",X, " la ",Y,":\n"),
                drum_elem(X,Y,L),afis(L),
                write("\n"),fail.
selectie('t'):-clearwindow.
selectie('d'):-extremitati(X,Y),drum_elem(X,Y,L),afis(L),write("\n"),fail.
selectie('d'):-write("\n\n    apasati orice tasta..."),readchar(_),clearwindow.
selectie('c'):-nod_initial(X),circuit_elem(X,L),afis(L),write("\n"),fail.
selectie('c'):-write("\n\n    apasati orice tasta..."),readchar(_),clearwindow.
selectie('z'):- write("Circuite elementare:\n"),
          nod_graf(X),write("apasati orice tasta...\n\n"),readchar(_),
          write("Circuite elementare cu nodul initial (si final) ", X, ":\n"),
           circuit_elem(X,L),afis(L),
           write("\n") ,fail.
selectie('z'):-clearwindow.
start:-fer,meniu,removewindow.
```

Predicatul drum_elem(X,Y,L) are următoarea semnificație: L este lista vârfurilor din drumul elementar care leagă X de Y; X şi Y sunt argumente de intrare iar L argument de ieşire. Predicatul circuit_elem(X,L) are două argumente: X este argument de intrare şi reprezintă nodul inițial (deci şi final), şi L este lista nodurilor prin care trece circuitul. Predicatele invers, inv, nr_elem, şi apartine au fost explicate în capitolul în care s-a prezentat prelucrarea listelor. Un exemplu de fişier "graf.dat" este următorul:

```
 predicates
arc1(varf,varf)
multime_varfuri(lista_varfuri)
clauses
```

```
multime_varfuri([a,b,c,d,e,f,g,h,i]).
arc1(a,b).
arc1(b,c).  arc1(b,e).
arc1(c,d).  arc1(c,f).
arc1(d,e).
arc1(e,a).  arc1(e,c).
arc1(f,b).  arc1(f,a).
arc1(g,h).
arc1(h,i).
arc1(i,g).
```

Programul anterior poate fi folosit şi pentru aflarea lanţurilor elementare şi a ciclurilor elementare într-un graf neorientat, dacă pentru fiecare fapt de forma arc1(i,j), din fişierul "graf.dat", se adaugă faptul arc1(j,i).

Arborele este un graf neorientat conex şi fără cicluri. Un exemplu de arbore apare în figura 3.

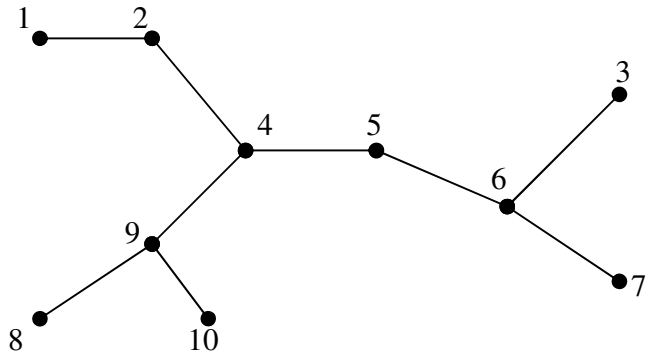


Figura 3

În multe aplicaţii, arborele este însoţit de punerea în evidenţă a unui vârf numit ***rădăcină***. Acest lucru are următoarele consecinţe:

- putem ***aşeza arborele pe nivele*** astfel
  - se plasează rădăcina pe nivelul 1;
  - se plasează pe fiecare nivel $i \geq 2$ vârfurile pentru care lungimea lanţurilor care le leagă de rădăcină este $i-1$;
  - se trasează muchiile grafului.

- arborele devine graf orientat, stabilindu-se pentru fiecare muchie un sens de parcurgere şi anume de la nivelul superior (cu număr de ordine mai mic) la nivelul imediat inferior (cu număr de ordine mai mic cu o unitate).

De exemplu pentru arborele din figura 3 alegându-se drept rădăcină vârful 4 obținem arborele aşezat pe nivele din figura 4

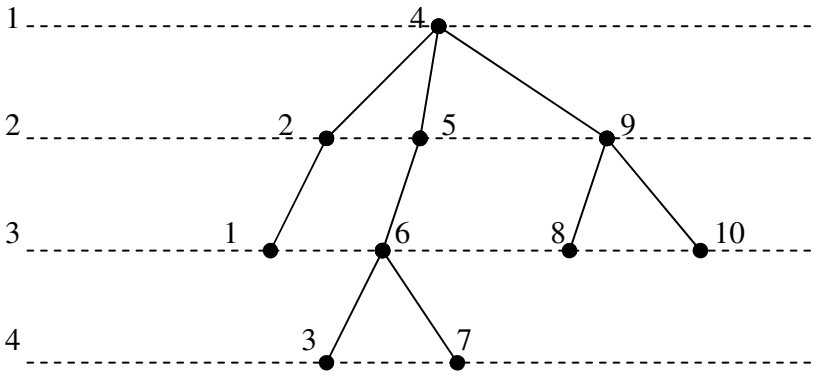


Figura 4

Graful orientat obținut este următorul:

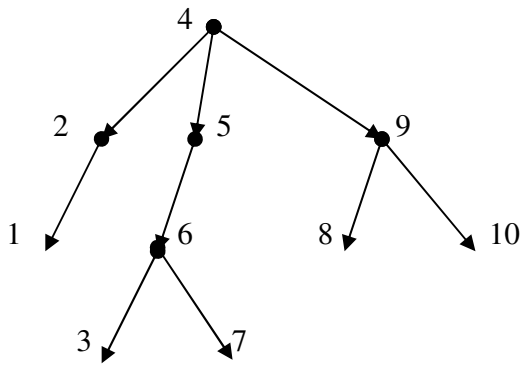


Figura 5

Cele de mai sus justifică următoarele denumiri pentru arborii aşezați pe nivele

- nivelul cu număr de ordine maxim se numeşte ***înălțimea arborelui***;
- vârfurile de un nivel i ≥ 2 legate de acelaşi vârf k de pe nivelul i-1 se numesc ***succesorii*** (sau ***descendenții direcți***, sau ***fiii***) vârfului k, iar k poartă numele de ***ascendent direct*** (sau ***tată***) al acestor vârfuri;
- numărul de descendenți direcți ai unui vârf se numeşte ***gradul vârfului***;
- vârfurile cu grad zero se numesc ***vârfuri frunză*** (sau ***terminale***); celelalte vârfuri se numesc ***vârfuri interne*** (sau ***neterminale***);

- dacă există un drum de la vârful i de pe nivelul m la un vârf j de pe nivelul n ≥ m+1, atunci i se numeşte ***ascendent*** al lui j, iar j ***descendent*** al lui i;

Un arbore se numeşte ***arbore n-ar*** dacă valoarea maximă a gradului vârfurilor sale este n. Dacă toate vârfurile sale au acelaşi grad, n, arborele se numeşte ***arbore n-ar strict***. În cazul n = 2 arborele se numeşte ***binar***, respectiv ***binar strict***.

9.2. Arbori de decizie

O metodă de optimizare a deciziilor în condiţii de risc este metoda arborilor de decizie. Într-un arbore de decizie vârfurile (nodurile) şi arcele au semnificaţii speciale. Nodurile unui arbore de decizie se clasifică în trei categorii:

- noduri decizionale – reprezintă un moment decizional (un moment în tipm corespunzător luării unei decizii)
- noduri aleatoare – reprezintă o consecinţă posibilă a luării unei decizii
- noduri consecinţă (economică) – reprezintă evaluarea unei consecinţe posibile a unei decizii

Rădăcina unui arbore de decizie este întotdeauna un nod decizional. Descendenţii direcţi ai oricărui nod decizional sunt întotdeauna noduri aleatoare. Descendenţii direcţi ai oricărui nod aleator sunt întotdeauna fie noduri decizionale fie noduri consecinţă economică. Nodurile consecinţă economică sunt nodurile terminale ale arborelui de decizie.

Arcele care unesc noduri decizionale cu noduri aleatoare reprezintă decizii. Arcele care unesc nodurile aleatoare cu noduri decizionale sau noduri consecinţă economică reprezintă probabilităţile asociate diverselor stări sau situaţii.

Fiecărui nod consecinţă economică i se asociază o valoare numerică ce reprezintă utilitatea sau importanţa consecinţei (asociate momentului respectiv).

Considerăm următoarele mulţimi

- D – mulţimea nodurilor decizionale,
- A – mulţimea nodurilor aleatoare,
- C – mulţimea nodurilor consecinţă economică.

Speranţa matematică este o funcţie $S : D \cup A \cup C \to R$, definită prin

$$S(N) = \begin{cases} \text{valoarea lui N, dacă } N \in C \\ \sum_{X \in Succ(N)} p(X)S(X), \text{ dacă } N \in A \\ \max_{X \in Succ(N)} S(X), \text{ dacă } N \in D \end{cases}$$

unde pentru fiecare arc (N, X), cu $N \in A$ şi $X \in D$, p(X) reprezintă probabilitatea ataşată arcului (N, X), iar pentru un nod $X \in D \cup A$, Succ(X) reprezintă mulțimea succesorilor (descendenților direcți) ai lui X.

Pentru fiecare nod decizional speranța matematică indică nivelul de importanță al deciziei optime din nodul N. Metoda arborilor de decizie presupune reprezentarea situației decizionale printr-un arbore, şi determinarea apoi a nivelului de importanță al variantelor decizionale prin calculul speranței matematice.

9.3. Sisteme expert care utilizează arbori de decizie

Un arbore de decizie poate fi utilizat drept bază de cunoştințe pentru un sistem expert de determinare a deciziei optime. Pentru această trebuie să reprezentăm arborele sub forma unei mulțimi de enunțuri simbolice.

În PROLOG, un arbore se poate reprezenta printr-o mulțime de fapte. O posibilitate ar fi să reprezentăm arborele printr-un singur fapt. Dacă arborele este complex, această reprezentare este greu de urmărit.

O altă posibilitatea este să reprezentăm arborele printr-o mulțime de fapte, astfel încât fiecare faptă să corespundă unui nod al arborelui şi descendenților săi direcți. Vom asocia descendentului direct al unui nod decizional decizia care a condus la nodul respectiv (tipul lui se defineşte prin

nod_decizie_succ=dn(decizie,nod_aleator))

Descendentului direct al unui nod aleator i se va asocia cazul (situația) care a condus la nodul respectiv şi probabilitatea acestei situații (tipul lui se defineşte prin

nod_aleator_succ= cpn(caz,probabilitate,nod_decizie_sau_consecinta)).

În programul următor semnificația predicatului speranta(X,Y,Z) este următoarea: X este nodul pentru care se calculează speranța matematică, Y este valoarea speranței matematice, iar Z este

decizia optimă, în cazul în care X este nod decizional (dacă X nu este nod decizional Z = n); X este argument de intrare, Y şi Z sunt argumente de ieşire. Fişierul "arbore.dat" trebuie să conțină declarațiile următoarelor predicate:

```
predicates
        nod_decizie_si_succesori(nod_decizie,lista_succesori_nod_decizie)
        nod_aleator_si_succesori(nod_aleator,lista_succesori_nod_aleator)
        nod_consecinta_economica(symbol,nod_con_ec)
```

iar în secțiunea clauses să conțină fapte care să descrie arborele.

```
domains
nod_decizie=nd(symbol)
nod_aleator=na(symbol)
nod_con_ec=nc(real)
nod=nd(symbol);na(symbol);nc(real)
nod_decizie_sau_consecinta=nd(symbol);nc(real)
decizie=symbol
caz=symbol
probabilitate=real
nod_decizie_succ=dn(decizie,nod_aleator)
nod_aleator_succ=cpn(caz,probabilitate,nod_decizie_sau_consecinta)
lista_succesori_nod_decizie=nod_decizie_succ*
lista_succesori_nod_aleator=nod_aleator_succ*
include "arbore.dat"
predicates
max(lista_succesori_nod_decizie,real,decizie)
medie(lista_succesori_nod_aleator,real)
speranta(nod,real,decizie)
meniu
selectie(char)
start
```

```
fer1
fer2
fer3
ferestre
sterge_ferestre
clauses
speranta(nc(X),X,n):-nod_consecinta_economica(_,nc(X)).
speranta(nd(X),M,D):-nod_decizie_si_succesori(nd(X),L),max(L,M,D).
speranta(na(X),Y,n):-nod_aleator_si_succesori(na(X),L),medie(L,Y).
max([dn(_,na(H1)),dn(D2,na(H2))],M2,D2):-
speranta(na(H1),M1,_),speranta(na(H2),M2,_),M1<M2,!.
max([dn(D1,na(H1)),dn(_,na(H2))],M1,D1):-
speranta(na(H1),M1,_),speranta(na(H2),M2,_),M1>=M2,!.
max([dn(_,na(H))|Y],M2,D2):- max(Y,M1,D2),speranta(na(H),M2,_),M1<M2,!.
max([dn(_,na(H))|Y],M1,D1):- max(Y,M1,D1),speranta(na(H),M2,_),M1>=M2,!.
medie([],0).
medie([cpn(_,P,nd(H))|X],Y):-speranta(nd(H),Y1,_),medie(X,Y2),Y=P*Y1/100.0+Y2.
medie([cpn(_,P,nc(H))|X],Y):-speranta(nc(H),Y1,_),medie(X,Y2),Y=P*Y1/100.0+Y2.
fer1:-makewindow(1,113,37,"Speranta",0,0,25,80).
fer2:-makewindow(2,113,36,"Nod...",2,2,9,76).
fer3:-makewindow(3,113,36,"Speranta=",13,2,10,76).
ferestre:-fer1,fer2,fer3,shiftwindow(1).
sterge_ferestre:-shiftwindow(3),removewindow,shiftwindow(2),removewindow,
                 shiftwindow(1),removewindow.
meniu:-cursor(7,20),write("d --> pentru nod decizional"),
       cursor(9,20),write("a --> pentru nod aleator"),
       cursor(11,20),write("c --> pentru nod consecinta economica"),
       cursor(13,20),write("e --> pentru iesire  "),
       readchar(X),clearwindow,X<>'e',selectie(X),meniu.
meniu.
selectie('d'):-shiftwindow(2),clearwindow,write("\n\n\n                    Nod                    =
"),readln(X),speranta(nd(X),Y,Z),
```

```
            shiftwindow(3),
            write("\n\n Speranta pentru nodul decizie: ",X," este ",Y,"\n"),
            write("\n Decizia ce trebuie  luata este ", Z, "\n"),
            write("\n            apasati orice tasta..."),
           readchar(_),clearwindow,shiftwindow(1).
selectie('a'):-shiftwindow(2),clearwindow,write("\n\n\n                Nod                =
"),readln(X),speranta(na(X),Y,_),
            shiftwindow(3),
            write("\n\n\n Speranta pentru nodul aleator: ",X," este ",Y,"\n"),
            write("\n\n            apasati orice tasta..."),
           readchar(_),clearwindow,shiftwindow(1).
selectie('c'):-shiftwindow(2),clearwindow, write("\n\n\n Nod = "),
           readln(C),nod_consecinta_economica(C,nc(X)),speranta(nc(X),Y,_),
            shiftwindow(3),
            write("\n\n\n Speranta pentru nodul consecinta: ",C," este ",Y,"\n"),
            write("\n\n            apasati orice tasta..."),
            readchar(_),clearwindow,shiftwindow(1).
start:-ferestre,meniu,sterge_ferestre.
```

Interogarea acestui program:

```
Goal: Start
```

Reproducem în continuarea un exemplu de problemă de decizie din [17] (pg. 248), şi apoi îi asociem arborele de decizie corespunzător, şi fişierul "arbore.dat", care descrie acest arbore. Lanțul decizional optim poate fi aflat prin interogarea programului anterior.

Exemplu. O editură studiază posibilitatea publicării unui volum special ocazionat de aniversarea unui anumit eveniment cultural-istoric. Dacă decizia este pozitivă se pune apoi problema determinării tirajului având în vedere următoarele costuri de producție:

- pentru drepturi de autor = 170 000 u.m.;
- costuri fixe = 130 000 u.m.;
- cost variabil unitar = 30 u.m.;

Editura ar putea testa succesul volumului în chestiune punându-l în vânzare pe o piață regională (sau într-un număr de librării alese la întâmplare). În acest scop va produce 5 000 de exemplare astfel că investiția va fi:

$I_1 = 170\,000 + 130\,000 + 5000 \times 30 = 450\,000$ u.m.

Dacă va decide să pună în vânzare volumul pe toată piața fără să-l testeze, va tipării 50 000 exemplare astfel că în acest caz costul investiției va fi:

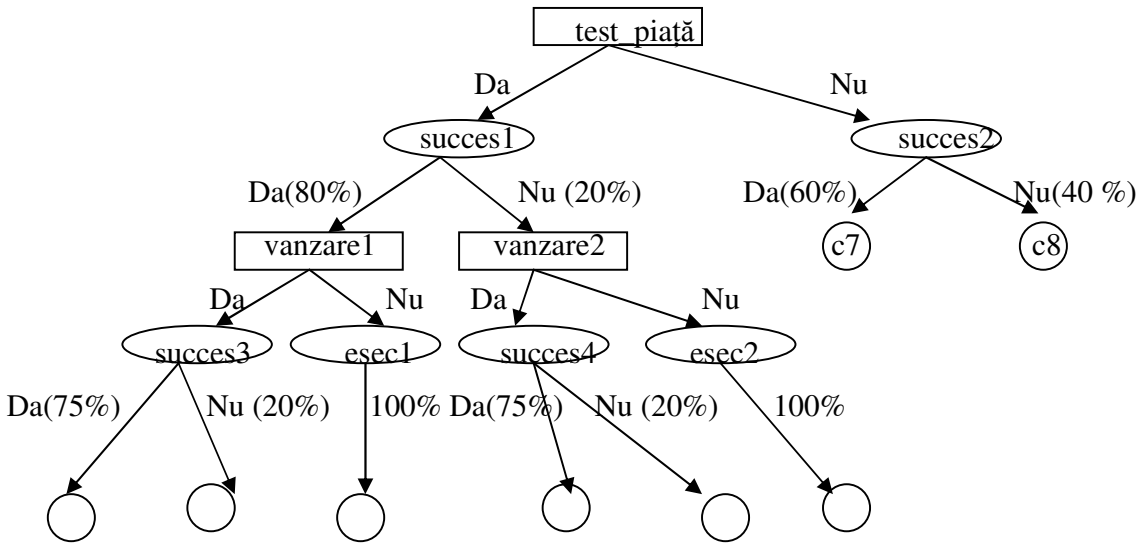
$I_2 = 170\,000 + 130\,000 + 50\,000 \times 30 = 1\,800\,000$ u.m.

În situația în care testul regional este pozitiv editura va tipării încă 45 000 de exemplare pe care le va vinde pe toată piața, costul investiției fiind:

$I_3 = 130\,000 + 45\,000 \times 30 = 1\,480\,000$

Se apreciază că probabilitatea ca testul să fie pozitiv este 80% şi de nereuşită 20%. De asemenea, se apreciază că în cazul unui test pozitiv vânzarea pe întreaga piață are succes cu o probabilitate de 75% şi insucces cu 25%, iar în cazul unui test negativ că probabilitatea de succes este 35% şi de insucces 65%. Să remarcăm că am considerat de fiecare dată doar două situații (stări ale naturii sau beneficiilor) şi anume succesul care presupune vânzarea volumului sau insuccesul când acesta nu se vinde (în sensul că numărul exemplarelor vândute este mic în raport cu tirajul şi îl socotim practic zero).

Să mai presupunem că în cazul netestării pieței se apreciază că succesul de vânzare ar fi 60% şi insuccesul 40%. Prețul de vânzare al volumului se determină a fi 80 u.m. Arborele de decizie este reprezentat în figura 6.



c1 c2 c3 c4 c5 c6

Figura 6

Am reprezentat nodurile decizionale prin dreptunghiuri, nodurile aleatoare prin elipse şi nodurile consecință economică prin cercuri. Calculăm beneficiile asociate fiecărei consecințe c1, c2, …, c8 (notate b1,b2, …, b8, respectiv). Pentru a nu complica exemplul nu se iau în considerare decât costurile de producție şi încasările. În baza informațiilor de mai sus se obține:

$b1 = 45\,000 \times 80 - (45\,000 \times 30 + 130\,000) + 5\,000 \times 80 - (170\,000 + 130\,00000 + 5\,000 \times 30) = 2\,070\,000$ u.m.

$b2 = 0 - (45\,000 \times 30 + 130\,000) + 5\,000 \times 80 - (170\,000 + 130\,00000 + 5\,000 \times 30) = -1\,530\,000$ u.m.

$b3 = 5\,000 \times 80 - (170\,000 + 130\,00000 + 5\,000 \times 30) = -50\,000$ u.m.

$b4 = 45\,000 \times 80 - (45\,000 \times 30 + 130\,000) + 0 - (170\,000 + 130\,00000 + 5\,000 \times 30) = 1\,670\,000$ u.m.

$b5 = 0 - (45\,000 \times 30 + 130\,000) + 0 + (170\,000 + 130\,00000 + 5\,000 \times 30) = -1\,930\,000$ u.m.

$b6 = 0 - 0 + 0 - (170\,000 + 130\,00000 + 5\,000 \times 30) = -450\,000$ u.m.

$b7 = 5\,000 \times 80 - (50\,000 \times 30 + 130\,00000) + 0 - 170\,000 = 2\,200\,000$ u.m.

$b8 = 0 - (50\,000 \times 30 + 130\,000) + 0 + 170\,000 = -1\,800\,000$

Fişierul (“arbore.dat”) care conține descrierea arborelui de decizie de mai sus este următorul:

```
predicates
nod_decizie_si_succesori(nod_decizie,lista_succesori_nod_decizie)
nod_aleator_si_succesori(nod_aleator,lista_succesori_nod_aleator)
nod_consecinta_economica(symbol,nod_con_ec)
clauses
nod_decizie_si_succesori(nd(test_piata),
        [dn(da,na(succes1)),dn(nu,na(succes2))]).
```

```
nod_decizie_si_succesori(nd(vanzare1),
        [dn(da,na(succes3)),dn(nu,na(esec1))]).
nod_decizie_si_succesori(nd(vanzare2),
        [dn(da,na(succes4)),dn(nu,na(esec2))]).
nod_aleator_si_succesori(na(succes1),
        [cpn(da,80,nd(vanzare1)),cpn(nu,20,nd(vanzare2))]).
nod_aleator_si_succesori(na(succes2),
        [cpn(da,60,nc(2200000)),cpn(nu,40,nc(-1800000))]).
nod_aleator_si_succesori(na(succes3),
        [cpn(da,75,nc(2070000)),cpn(nu,25,nc(-1530000))]).
nod_aleator_si_succesori(na(esec1),
        [cpn(o,100,nc(-1530000))]).

nod_aleator_si_succesori(na(succes4),
        [cpn(da,35,nc(1670000)),cpn(nu,65,nc(-1930000))]).
nod_aleator_si_succesori(na(esec2),
        [cpn(o,100,nc(-450000))]).
nod_consecinta_economica(c1,nc(2070000)).
nod_consecinta_economica(c2,nc(-1530000)).
nod_consecinta_economica(c3,nc(-50000)).
nod_consecinta_economica(c4,nc(1670000)).
nod_consecinta_economica(c5,nc(-1930000)).
nod_consecinta_economica(c6,nc(-450000)).
nod_consecinta_economica(c7,nc(2200000)).
nod_consecinta_economica(c8,nc(-1800000)).
```