Gliwice, 10.06.2014

Laboratorium

Komputerowe wspomaganie podejmowania decyzji

Ćw. 4 i 5 – Procesy decyzyjne w postaci ekstensywnej o sumie zerowej i niezerowej

Walentek Adrian
Olberek Kamil
AiR, sem. 6, TI-2, sekcja 1

Laboratorium 4
Procesy decyzyjne w postaci ekstensywnej o sumie zerowej

1. Przykładowy problem wieloetapowy o liczbie etapów K=3.

Przykładowym problemem wieloetapowym może być rywalizacja dwóch firm. Każda z firm ma do wyboru albo strategie penetracji rynku (A) albo strategie rozwoju rynku (B). Firmy zmieniają swoje strategie, co pół roku. Dodatkowo każda z firm nie wie o podjętej decyzji przez przeciwnika. Firma F1 chce zminimalizować wartość wypłat, z kolei firma F2 chce maksymalizować.

Wyniki wyborów firm w ostatniej gałęzi drzewa są następujące:

4 49 25 45 14 1 33 27 26 9 44 47 49 31 17 26 23 17 18 4 20 24 12 35 7 47 41 41 24 23 33 21 2 2 46 24 19 50 27 45 18 5 21 33 49 4 35 24 32 7 26 14 49 47 40 17 10 18 34 22 45 20 42 37

Rozwiązując to zagadnienie zaczynamy od ostatniej zmiany strategii firm. W zależności od wybory strategii firm możemy otrzymać 16 możliwych wyników:

F1/F2	A	B
A	4	49
B	25	45

Wybory firm: F1=B, F2=B z wynikiem 45.

F1/F2	A	B
A	14	1
B	33	27

Wybory firm: F1=A, F2=A z wynikiem 14

F1/F2	A	B
A	26	9
B	44	47

Wybory firm: F1=A, F2=A z wynikiem 26

F1/F2	A	B
A	49	31
B	17	26

Wybory firm: F1=B, F2=B z wynikiem 26

F1/F2	A	B
A	23	17
B	18	4

Wybory firm: F1=A, F2=A z wynikiem 18

F1/F2	A	B
A	20	24
B	12	35

Wybory firm: F1=A, F2=B z wynikiem 24

F1/F2	A	B
A	7	47
B	41	41

Wybory firm: F1=B, F2=B z wynikiem 41

F1/F2	A	B
A	24	23
B	33	21

Wybory firm: F1=A, F2=A z wynikiem 24

F1/F2	A	B
A	2	2
B	46	24

Wybory firm: F1=A, F2=A z wynikiem 2

F1/F2	A	B
A	19	50
B	27	45

Wybory firm: F1=B, FB=A z wynikiem 45

F1/F2	A	B
A	18	5
B	21	33

Wybory firm: F1=A, F2=A z wynikiem 18

F1/F2	A	B
A	49	4
B	35	24

Wybory firm: F1=B, F2=A z wynikiem 35

F1/F2	A	B
A	32	7
B	26	14

Wybory firm: F1=B, F2=A z wynikiem 26

F1/F2	A	B
A	49	47
B	41	17

Wybory firm: F1=B, F2=A z wynikiem 40

F1/F2	A	B
A	10	18
B	34	22

Wybory firm: F1=A, F2=B z wynikiem 18.

F1/F2	A	B
A	45	20
B	42	37

Wybory firm: F1=B, F2=A z wynikiem 42.

Na podstawie wyników z ostatniej potyczki firm możemy stworzyć macierze dla kolejnej potyczki. Tutaj mamy 4 możliwe sytuacje.

F1/F2	A	B
A	45	14
B	26	26

Wybory firm: F1=B, F2=A z wynikiem 26.

F1/F2	A	B
A	18	24
B	41	24

Wybory firm: F1=A, F2=B z wynikiem 24

F1/F2	A	B
A	2	45
B	18	35

Wybory firm: F1=B, F2=B z wynikiem 35

F1/F2	A	B
A	26	40
B	18	42

Wybory firm: F1=A, F2=B z wynikiem 40

Na podstawie powyższych wyników tworzymy macierz dla potyczki pierwszej.

F1/F2	A	B
A	26	24
B	35	40

Wybory firm: F1=A, F2=A z wynikiem 26.

Wynikiem całej potyczki jest wynik 26.

2. Program zaimplementowany w Matlabie.

Program zwraca nam wszystkie macierze w raz z wynikami, dla każdej z nich. Zwraca także końcowy wynik gry oraz decyzje graczy. Dodatkowo wizualizuje obliczone decyzje.

Fragment zwróconego wyniku przez program:

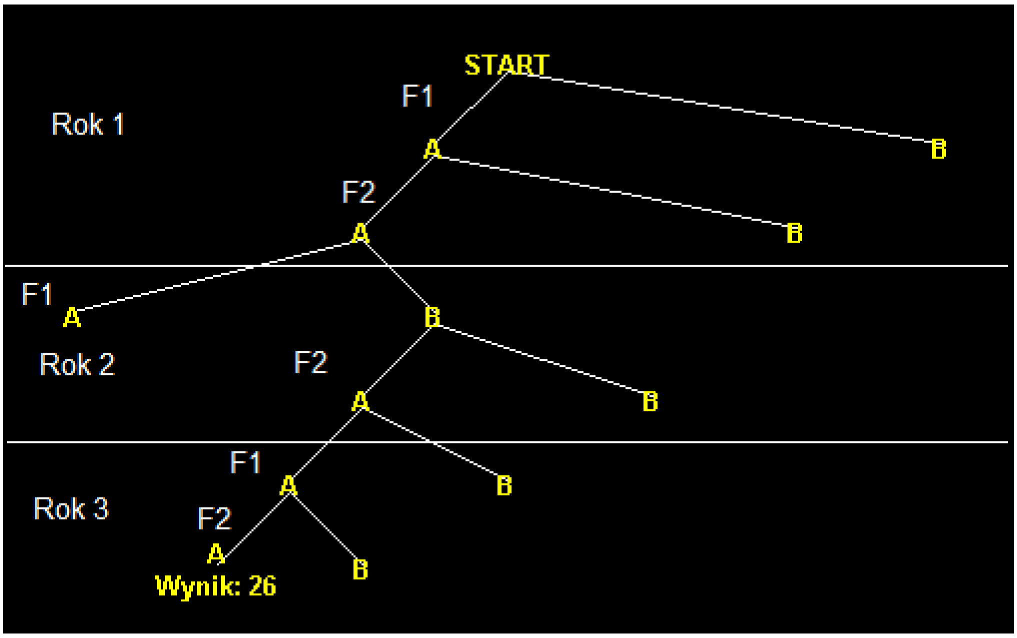
Kolejne decyzje graczy to:

1 2 1

1 1 1

Wynik gry to:26

Poniższy graf przedstawia ścieżkę podjętych decyzji przez firmy wyrysowany przez program:



Listing programu

```
function skrypt(A,k)
i=1;
liczba_decyzji = 0;
%obliczanie potrzebnych liczby decyzji
for i=k:-1:1
    liczba_decyzji = liczba_decyzji + (4^(i-1));
end
wektor_aktualny=A;
n=1;
%petla glowna rozwiazujaca problem
for j=k:-1:1
    disp(['Poziom ' num2str(j)]);
    dlugosc=length(wektor_aktualny);
    m=1;
    for i=1:4:dlugosc
        %macierz wynikow
        B=[wektor_aktualny(i), wektor_aktualny(i+1); wektor_aktualny(i+2),
wektor_aktualny(i+3)]
        %obliczanie punktu siodlowego
        [wektor_aktualny(m), D1, D2]=siodlowy(B,'minmax');
        %zapis danych potrzebnych do znalezenia decyzji graczy
        D(1,n) = D1;
        D(2,n) = D2;
        D(3,n) = wektor_aktualny(m);
        m=m+1;
        n=n+1;
    end
    wektor_aktualny=wektor_aktualny(1:m-1);
    %rysowanie drzewa dla aktualnego poziomu gry
    t = ntree(2,(k-j+1)*2);
    plot(t);
end

%obliczanie wykonywanych decyzji po kolei
Dk = [D(:,liczba_decyzji)];
lb = 1;
Dtemp2 = [];
for i=2:k
    lb = lb + 4^(i-1);
    D_temp = D(:, (liczba_decyzji - lb +1):(liczba_decyzji - lb + 4^(i-1)));
    for(j=1:size(Dk,2))
        if(Dk(1,j) == 1)
            D_temp = D_temp(:, 1:(size(D_temp,2)/2));
        else
            D_temp = D_temp(:, (size(D_temp,2)/2)+1:size(D_temp,2));
        end
        if(Dk(2,j) == 1)
            D_temp = D_temp(:, 1:(size(D_temp,2)/2));
        else
            D_temp = D_temp(:, (size(D_temp,2)/2)+1:size(D_temp,2));
        end
    end
    Dk = [Dk D_temp];
end
disp('Kolejne decyzje graczy to:');
disp(Dk(1:2,:));

%rysowanie drzewa dla rozwiazania gry
Drzewo = ntree(2,0);
x = 0;
for i=1:k
    Drzewo = nodesplt(Drzewo, x);
    x = x*2 + Dk(1,i);
    Drzewo = nodesplt(Drzewo, x);
    x = x*2 + Dk(2,i);
```

```
end
plot(Drzewo);

wynik=wektor_aktualny(1);
disp(['Wynik gry to:' num2str(wynik)]);
end
```

Funkcja obliczająca punkt siodłowy

```
function [punkt_siodlowy, decyzjaD1, decyzjaD2] = siodlowy(A, rola)
[Nx,Ny]=size(A);
%'maxmin'; %pierwszy maksymalizuje, drugi minimalizuje
%'minmax'; %pierwszy minimalizuje, drugi maksymalizuje
%--------------------------------------------------
if rola=='maxmin'
    A=A';
    [Nx,Ny]=size(A);
end
%gracz pierwszy
for i=1:Nx
    Tab_max(i)=max(A(i,:));
end
[S_D1, decyzjaD1]=min(Tab_max);
%gracz drugi
for j=1:Ny
    Tab_min(j)=min(A(:,j));
end
[S_D2, decyzjaD2]=max(Tab_min);
if S_D1==S_D2
    punkt_siodlowy=S_D1;
    disp(['Wybory graczy D1=', num2str(decyzjaD1), '; D2=' num2str(decyzjaD2)]);
    disp(['Punkt siodlowy istnieje. Poziom bezpieczeństwa graczy to: ' num2str(S_D2)]);
end
if S_D1~=S_D2
    disp('Punkt siodlowy nie istnieje.');
end
```

Laboratorium 5
Procesy decyzyjne w postaci ekstensywnej o sumie niezerowej

1. Rozwiązanie przykładowego problemu wieloetapowego o liczbie etapów K=3.

Powyższy problem dla dwóch graczy opisany jest poniższymi macierzami strat:
Gracz G1:
```
3 14 11 10 2 11 6 16 24 24 4 15 2 3 9 4 4 12 16 7 16 3 0 3 10 16 11 20
5 15 2 18 20 18 2 23 1 14 11 24 25 15 6 20 23 14 5 20 7 7 2 23 1 20 22
8 4 12 19 16 9 5 15 2
```
Gracz G2:
```
10 13 19 12 18 23 25 2 6 8 2 20 2 2 18 15 21 23 20 0 9 9 6 6 7 19 21 21
3 15 2 20 11 6 2 18 5 16 3 3 21 19 12 23 5 12 19 15 12 19 7 19 17 23 7
7 24 2 8 20 4 8 21 10
```
Obaj gracze dążą do zminimalizowania swoich strat.
Dla 3 etapu gry każdy gracz musi podjąć 16 decyzji:

Gracz G1

G1/G2	A	B
A	3	14
B	11	10

Gracz G2

G1/G2	A	B
A	10	13
B	19	12

Strategia (1,1) wynik (3,10) - Nash

Gracz G1

G1/G2	A	B
A	2	11
B	6	16

Gracz G2

G1/G2	A	B
A	18	23
B	25	2

Strategia (1,1) wynik (2,18) - Nash

Gracz G1

G1/G2	A	B
A	24	24
B	4	15

Gracz G2

G1/G2	A	B
A	6	8
B	2	20

Strategia (2,1) wynik (4,2) - Nash

Gracz G1

G1/G2	A	B
A	2	3
B	9	4

Gracz G2

G1/G2	A	B
A	2	2
B	18	15

Strategia (1,1) wynik (2,2) - Nash

Gracz G1

G1/G2	A	B
A	4	12
B	16	7

Gracz G2

G1/G2	A	B
A	21	23
B	20	0

Strategia (2,2) wynik (7,0) - Nash

Gracz G1

G1/G2	A	B
A	16	3
B	0	3

Gracz G2

G1/G2	A	B
A	9	9
B	6	6

Strategia (2,1) wynik (0,6) - Nash

Gracz G1

G1/G2	A	B
A	10	16
B	11	20

Gracz G2

G1/G2	A	B
A	7	19
B	21	21

Strategia (1,1) wynik (10,7) - Nash

Gracz G1

G1/G2	A	B
A	5	15
B	2	18

Gracz G2

G1/G2	A	B
A	3	15
B	2	20

Strategia (2,1) wynik (2,2) - Nash

Gracz G1

G1/G2	A	B
A	20	18
B	2	23

Gracz G2

G1/G2	A	B
A	11	6
B	2	18

Strategia (2,1) wynik (2,2) - Nash

Gracz G1

G1/G2	A	B
A	1	14
B	11	24

Gracz G2

G1/G2	A	B
A	5	16
B	3	3

Strategia (1,1) wynik (1,5) - Nash

Gracz G1

G1/G2	A	B
A	25	15
B	6	20

Gracz G2

G1/G2	A	B
A	21	19
B	12	23

Strategia (2,1) wynik (6,12) - Nash

Gracz G1

G1/G2	A	B
A	23	14
B	5	20

Gracz G2

G1/G2	A	B
A	5	12
B	19	15

Strategia (2,1) wynik (5,19) - Minimax

Gracz G1

G1/G2	A	B
A	7	7
B	2	23

Gracz G2

G1/G2	A	B
A	12	19
B	7	19

Strategia (2,1) wynik (2,7) - Nash

Gracz G1		
G1/G2	A	B
A	1	20
B	22	8

Gracz G2		
G1/G2	A	B
A	17	23
B	7	7

Strategia (1,1) wynik (1,17) - Nash

Gracz G1		
G1/G2	A	B
A	4	12
B	19	16

Gracz G2		
G1/G2	A	B
A	24	2
B	8	20

Strategia (1,2) wynik (12,2) - Nash

Gracz G1		
G1/G2	A	B
A	9	5
B	15	2

Gracz G2		
G1/G2	A	B
A	4	8
B	21	10

Strategia (2,2) wynik (2,10) - Nash

Na 2 etapie gry każdy z graczy podejmuje 4 decyzje:

Gracz G1		
G1/G2	A	B
A	3	2
B	4	2

Gracz G2		
G1/G2	A	B
A	10	18
B	2	2

Strategia (1,1) wynik (3,10) - Nash

Gracz G1		
G1/G2	A	B
A	7	0
B	10	2

Gracz G2		
G1/G2	A	B
A	0	6
B	7	2

Strategia (1,1) wynik (7,0) - Nash

Gracz G1		
G1/G2	A	B
A	2	1
B	6	5

Gracz G2		
G1/G2	A	B
A	2	5
B	12	19

Strategia (1,1) wynik (2,2) - Nash

Gracz G1		
G1/G2	A	B
A	2	1
B	12	2

Gracz G2		
G1/G2	A	B
A	7	17
B	2	10

Strategia (1,1) wynik (2,7) - Nash

W pierwszym etapie gracze podejmują 1 decyzje:

Gracz G1		
G1/G2	A	B
A	3	7
B	2	2

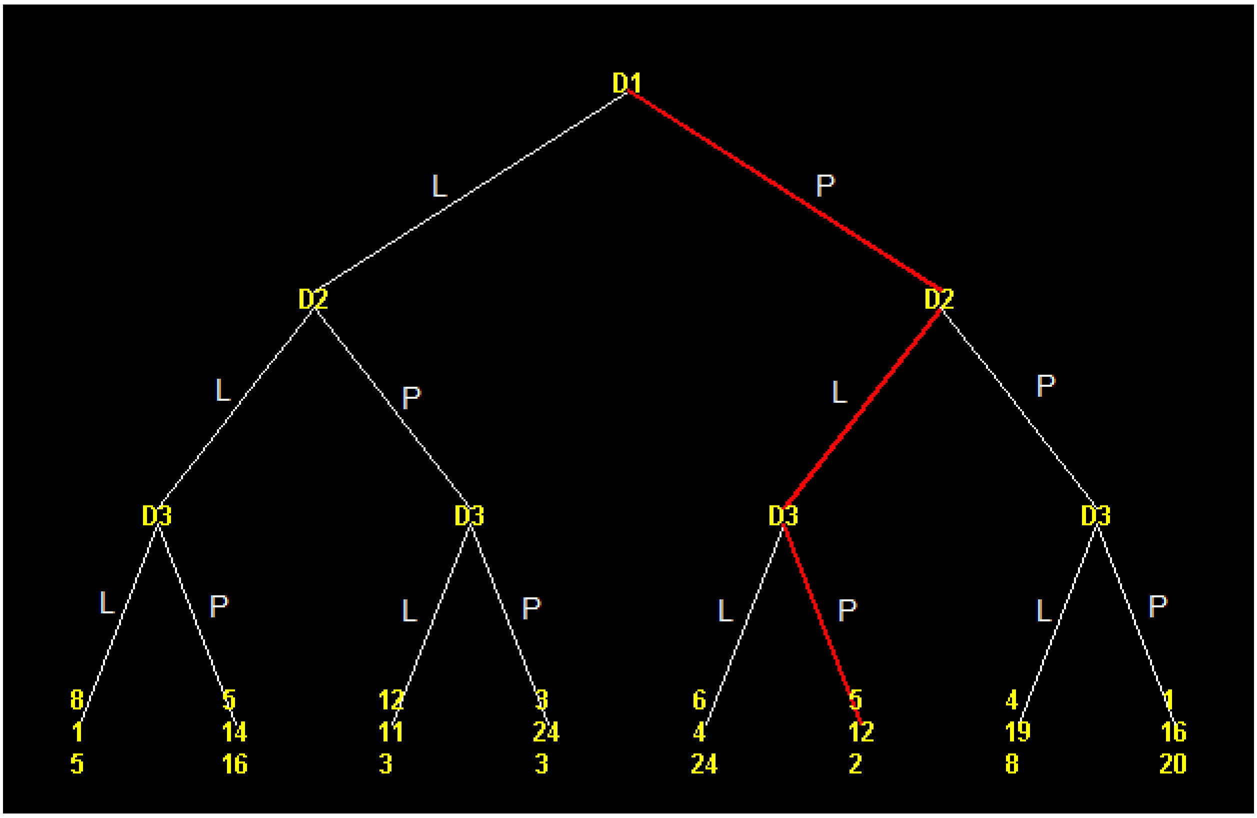
Gracz G2		
G1/G2	A	B
A	10	0
B	2	7

Strategia (2,1) wynik (2,2) - Nash

Rozwiązaniem gry jest wynik (2,2).
Decyzje, które do tego doprowadziły:
-etap 1 (B,A)
-etap 2 (A,A)
-etap3 (B,A)

2. Rozwiązanie przykładowego problemu dla trzech graczy.

Mamy następujący problem dany drzewem decyzyjnym.



Jeśli gracz D1 zagra w lewo to gracze D1 i D2 zagrają poniższą grę dwumacierzową:

Gracz D1

D2/D3	L	P
L	1	14
P	11	24

Gracz D2

D2/D3	L	P
L	5	16
P	3	3

Równowaga Nasha w tej grze to strategia (1,1) z wynikiem (1,5)

Jeśli jednak gracz D1 zagra w prawo to gracze D1 i D2 rozegrają poniższą grę dwumacierzową:

Gracz D1

D2/D3	L	P
L	4	12
P	19	16

Gracz D2

D2/D3	L	P
L	24	2
P	8	20

Równowaga Nasha w tej grze to strategia (1,2) z wynikiem (12,2)

Gracz D1 wybierze drogę P, gdyż daje mu to stratę równą 5. Tak więc strategia (D1,D2,D3)=(P,L,P) da nam równowagę Nasha o wyniku (5,12,2). Na powyższym drzewie na czerwono została zaznaczona strategia dająca równowagę Nasha.

Podsumowanie:

Dzięki użyciu drzew decyzyjnych można lepiej opisać strukturę gry, uwzględniając kolejność podejmowania decyzji oraz informacje jakie każdy z graczy posiada na danym etapie. Obejmując wzrokiem wszystkie możliwe opcje gry, jesteśmy w stanie wyznaczyć strategie odpowiednią dla wszystkich graczy.

W przypadku problemu decyzyjnego dla 3 graczy nie ma jednoznacznego rozwiązania. W zależności od wyboru jednej ze strategii Nasha lub zagrania w strategie bezpieczne każdy z graczy uzyskuje lepszy lub gorszy wynik. Nie można wybrać z nich najlepszego rozwiązania, ponieważ zawsze, któryś z graczy ucierpi, a inni zyskają.

3. Program zaimplementowany w Matlabie.

Program zwraca nam wszystkie macierze w raz z wynikami, dla każdej z podgry. Zwraca także końcowy wynik gry oraz decyzje graczy. Dodatkowo wizualizuje obliczone decyzje.

Fragment zwróconego wyniku przez program:

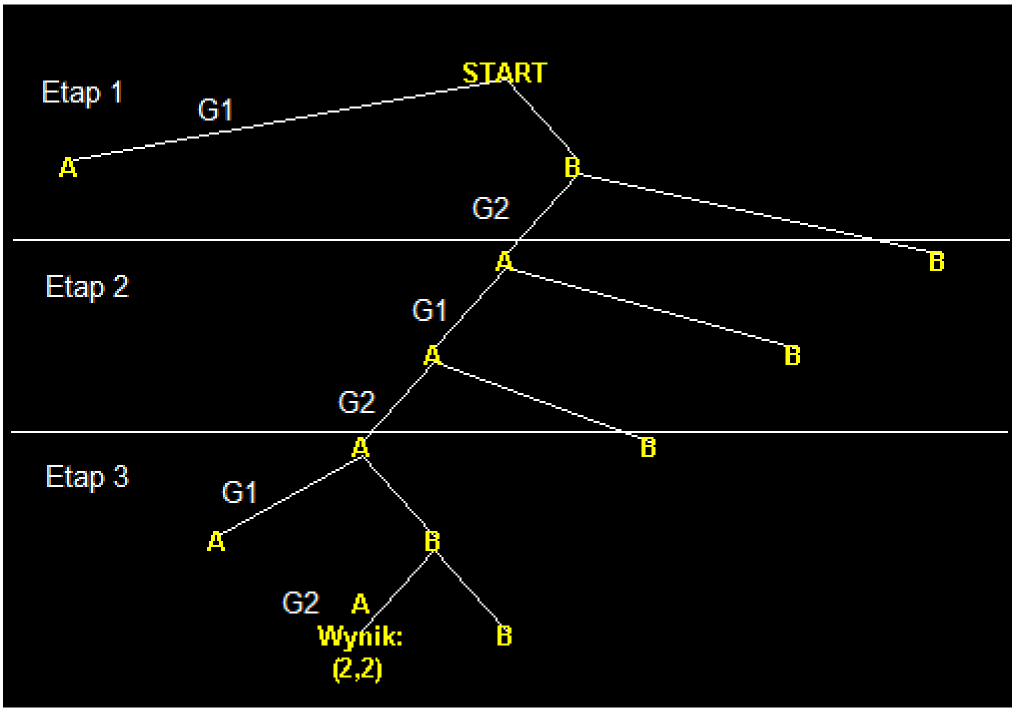
Kolejne decyzje graczy to:

2 1 2

1 1 1

Wynik gry to (2,1)=(2,2)

Poniższy graf przedstawia ścieżkę podjętych decyzji przez graczy wyrysowany przez program:



Listing programu:

```
function punkt1(A,B,k)
    i=1;
    liczba_decyzji = 0;
    %obliczanie potrzebnych liczby decyzji
    for i=k:-1:1
        liczba_decyzji = liczba_decyzji + (4^(i-1));
    end
    wektor_aktualnyA=A;
    wektor_aktualnyB=B;
    n=1;
    %głowna czesc programu, obliczanie kolejnych rozwiazan dla poziomow
    for j=k:-1:1
        disp(['Poziom ' num2str(j)]);
        m=1;
        dlugosc=length(wektor_aktualnyA);
         for i=1:4:dlugosc
            temp_A=[wektor_aktualnyA(i), wektor_aktualnyA(i+1); wektor_aktualnyA(i+2),
wektor_aktualnyA(i+3)]
            temp_B=[wektor_aktualnyB(i), wektor_aktualnyB(i+1); wektor_aktualnyB(i+2),
wektor_aktualnyB(i+3)]
            [i_wA, i_wB, wektor_aktualnyA(m), wektor_aktualnyB(m),
metoda]=nash(temp_A,temp_B);
            disp(['Rozwiazanie ' metoda ': (' num2str(i_wA) ',' num2str(i_wB) ')=('
num2str(wektor_aktualnyA(m)) ',' num2str(wektor_aktualnyB(m))  ')']);
            D(1,n) = i_wA;
            D(2,n) = i_wB;
            D(3,n) = wektor_aktualnyA(m);
            D(4,n) = wektor_aktualnyB(m);
            m=m+1;
            n=n+1;
         end
         wektor_aktualnyA=wektor_aktualnyA(1:m-1);
         wektor_aktualnyB=wektor_aktualnyB(1:m-1);
         %rysowanie drzewa dla aktualnego poziomu gry
         t = ntree(2,(k-j+1)*2);
         plot(t);
    end
    %obliczanie wykonywanych decyzji po kolei
    Dk = [D(:,liczba_decyzji)];
    lb = 1;
    Dtemp2 = [];
    for i=2:k
        lb = lb + 4^(i-1);
        D_temp = D(:, (liczba_decyzji - lb +1):(liczba_decyzji - lb + 4^(i-1)));
        for(j=1:size(Dk,2))
            if(Dk(1,j) == 1)
                D_temp = D_temp(:, 1:(size(D_temp,2)/2));
            else
                D_temp = D_temp(:, (size(D_temp,2)/2)+1:size(D_temp,2));
            end
            if(Dk(2,j) == 1)
                D_temp = D_temp(:, 1:(size(D_temp,2)/2));
            else
                D_temp = D_temp(:, (size(D_temp,2)/2)+1:size(D_temp,2));
            end
        end
        Dk = [Dk D_temp];
    end
    disp('Kolejne decyzje graczy to:');
    disp(Dk(1:2,:));

    %rysowanie drzewa dla rozwiazania gry
    Drzewo = ntree(2,0);
    x = 0;
    for i=1:k
```

```
        Drzewo = nodesplt(Drzewo, x);
        x = x*2 + Dk(1,i);
        Drzewo = nodesplt(Drzewo, x);
        x = x*2 + Dk(2,i);
    end
    plot(Drzewo);
    %wyswietlanie wyniku
    disp(['Wynik gry to (' num2str(D(1,n-1)) ',' num2str(D(2,n-1)) ')=('
num2str(wektor_aktualnyA(1)) ',' num2str(wektor_aktualnyB(1))  ')']);
end
```

Funkcja obliczająca rozwiązania macierzy podgier

```
function [index_wA, index_wB, wA, wB, metoda]=nash(A,B)
    [Nx,Ny] = size(A);
    row_nasha=zeros(4,Ny);
    %rownowaga nasha
    k=1;
    for j=1:Ny
            [minimumA,indexA]=min(A(:,j));
            [minimumB,indexB]=min(B(indexA,:));
            if(j==indexB)
                row_nasha(1,k)=indexA;
                row_nasha(2,k)=indexB;
                row_nasha(3,k)=minimumA;
                row_nasha(4,k)=minimumB;
                k=k+1;
            end
    end
    liczba_wynikow=k-1;
    %sprawdzanie ilosci wynikow dopuszczalnych
    liczba_dopuszczalnych=0;
    if liczba_wynikow>1
        Suma = sum(row_nasha(3:4,:));
        minimum=min(Suma);
        for j=1:Ny
            if Suma(j)==minimum
                liczba_dopuszczalnych=liczba_dopuszczalnych+1;
                pozycja_dopuszczalnego=j;
            end
        end
    end
    if liczba_dopuszczalnych == 1
        metoda='Nash';
        index_wA=row_nasha(1,pozycja_dopuszczalnego);
        wA=row_nasha(3,pozycja_dopuszczalnego);
        index_wB=row_nasha(2,pozycja_dopuszczalnego);
        wB=row_nasha(4,pozycja_dopuszczalnego);
    end
    if liczba_wynikow == 1
        metoda='Nash';
        index_wA=row_nasha(1,1);
        wA=row_nasha(3,1);
        index_wB=row_nasha(2,1);
        wB=row_nasha(4,1);
    end
    %rownowaga minimaksowa
    if liczba_wynikow==0 || liczba_dopuszczalnych>=2
        metoda='Minmax';
        k=1;
        for i=1:Nx
            [maksimum,j]=max(A(i,:));
            [minimum,index] = min(A(:,j));
            if maksimum >= minimum
                minmaks1(1,k)=index;
                minmaks1(2,k)=j;
```

```
                minmaks1(3,k)=A(index,j);
                k=k+1;
            end
        end
        if k==1
            disp('Brak minimaksowych rozwiazan w macierzy A');
        else
            k=1;
            for j=1:Ny
                [maksimum,i]=max(B(:,j));
                [minimum,index] = min(B(i,:));
                if maksimum >= minimum
                    minmaks2(1,k)=index;
                    minmaks2(2,k)=j;
                    minmaks2(3,k)=B(index,j);
                    k=k+1;
                end
            end
        end
        if k==1
                disp('Brak minimaksowych rozwiazan w macierzy B');
        else
                for i=1:k-1;
                    if minmaks1(1,i)==minmaks2(1,i) && minmaks1(2,i)==minmaks2(2,i)
                        disp(['Rozwiazanie minimaksowe: (' num2str(minmaks2(1,i)) ','
num2str(minmaks2(2,i)) ')=(' num2str(minmaks1(3,i)) ',' num2str(minmaks2(3,i))  ')']);
                        metoda='Minmax';
                        index_wA=minmaks2(1,i);
                        wA=minmaks1(3,i);
                        index_wB=minmaks2(2,i);
                        wB=minmaks2(3,i);
                    end
                end
        end
    end
end
```