Projektowanie Systemów Informatycznych

Rok akademicki 2015/2016	TEMAT: **Rozwiązanie układu równań metodą iteracji prostej (mnożenie macierzy przez wektor) A·x=x'**		
Kierunek studiów:	**Informatyka**	Wykonawca:	Dawid Gawiński Oskar Gruszczyński
Semestr:	**V**		
Data oddania:	**14. I. 2016**	Podpis:	

1. Wybór wariantu

Na podstawie numeru indeksu 8427 określono numer wariantu zadania AL.
AL został wyznaczony według wzoru

```
AL = 1 + (IND) mod 20
```

gdzie:

```
IND - liczba składająca się z dwóch ostatnich cyfr numeru indeksu.
```

Ostatecznie:

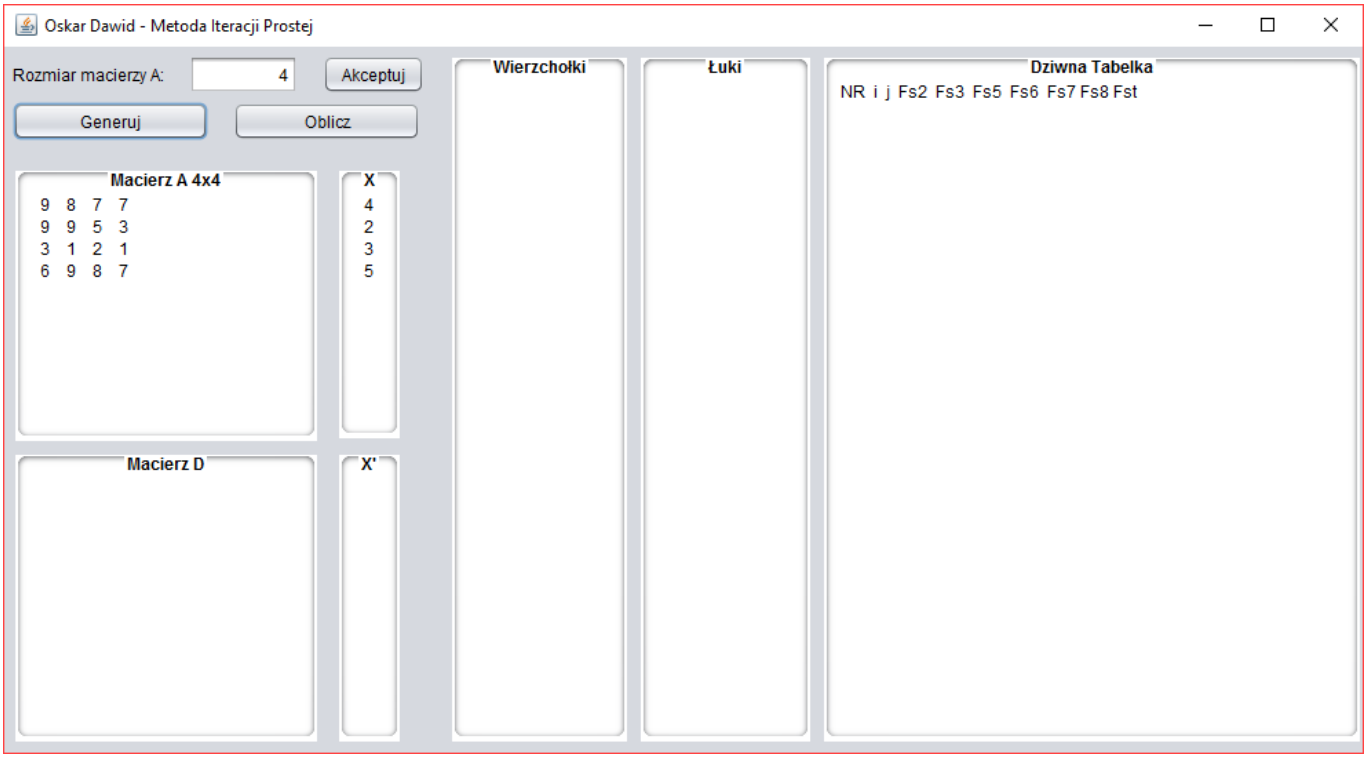
```
AL = 1 + (27) mod 20 = 8
```

ZADANIE:

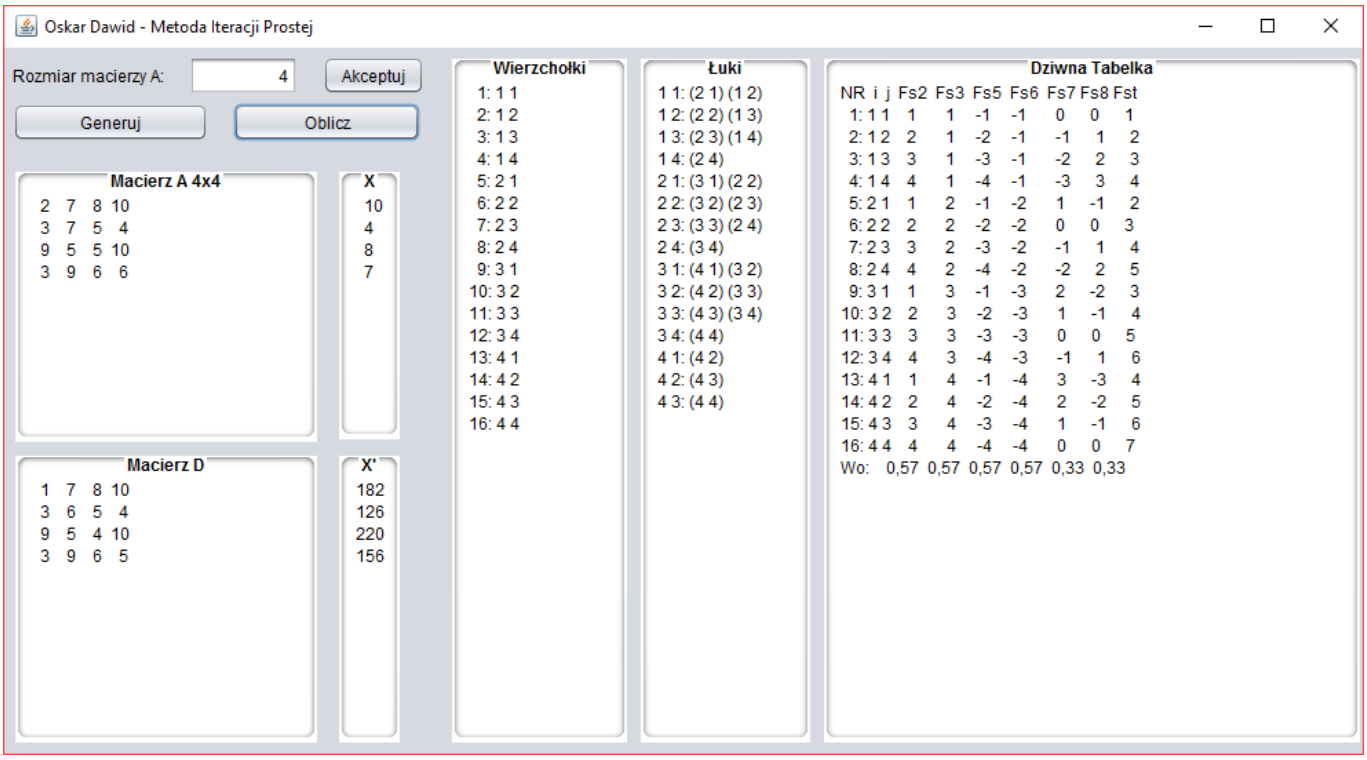
Rozwiązanie układu równań metodą iteracji prostej (mnożenie macierzy przez wektor) A·x=x'

2. Program realizujący zadanie

Korzystając z środowiska NeatBeans 8.0 został opracowany program realizujący powyższe zadanie:



Rys.1. Wygenerowane losowo dane.



Rys.2. Okno aplikacji wraz z danymi wyjściowymi.

Poniżej przedstawiono 2 fragmenty kodu źródłowego, których zadaniem jest:

KOD 1
- generowanie losowo danych do obliczenia

KOD 2
- wyznaczenie macierzy D potrzebnej do obliczeń
- obliczenia kolejnych wartości zmiennej x_n
- wyznaczenie wierzchołków oraz listy łuków

OBJAŚNIENIA

```
i – wiersze
j – kolumny
rn- zmienna pomocnicza
N – rozmiar macierzy kwadratowej
l – przechowuje informacje o bieżącym wierzchołku
tabb[]-tablica, w której przechowujemy wartości wyrazów wolnych
taba[]-tablica, w której przechowujemy wartości w macierzy A
determ-zmienna odpowiedzialna za wyznacznik macierzy A
tabd[]-tablica,na której wykonujemy działania
tabx[]-tablica, która przechowuje obliczone wartości X`
```

KOD 1

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        Random rn = new Random();
        jTextArea2.setText("");
        jTextArea1.setText("");
        jt_w.setText("");
        jt_l.setText("");
        for (int x=0;x<size;x++)
        {
            tabb[x]=rn.nextInt(10) + 1;
            jTextArea1.append("  "+tabb[x]+" ");
            for (int y=0;y<size;y++)
            {
                taba[x][y]=rn.nextInt(10) + 1;
                if (taba[x][y]<10)
                {
                    jTextArea2.append("  "+taba[x][y]+ "  ");
                }
```

```
                    else
                    {
                        jTextArea2.append(taba[x][y]+ "  ");
                    }
                }
                jTextArea2.append("\n");
                jTextArea1.append("\n");
            }
    }
```

KOD 2

```
    if (determ(taba,size)<1)
            {
                for (int i=0;i<size;i++)
                {
                    for (int j=0;j<size;j++)
                    {
                        if (i==j)
                        {
                            tabd[i][j]=taba[i][j]-1;
                        }
                        else
                        {
                            tabd[i][j]=taba[i][j];
                        }
                        if (tabd[i][j]<10)
                        {
                            jTextArea3.append("  "+tabd[i][j]+ "  ");
                        }
                        else
                        {
                            jTextArea3.append(tabd[i][j]+ "  ");
                        }
                    }
                    jTextArea3.append("\n");
                }
                for(int i=0;i<size;i++)
                {
                    tabx[i]=tabb[i];
                }
                int l=0;
                for (int i=0;i<size;i++)
                {
                    for (int j=0;j<size;j++)
                    {
                        tabb[i]=tabb[i]+(tabd[i][j]*tabx[j]);
                         l=l+1;
                         if(l<10)
                         {
                             jt_w.append("  "+l +": "+(i+1)+" "+(j+1)+"\n");
                         }
                         else
                         {
                            jt_w.append(l +": "+(i+1)+" "+(j+1)+"\n");
                         }
                     }
          for(int j=0;j<size;j++)
                {
                    jTextArea7.append(+tabb[j]+"\n");
                }
            }
            else
            {
        jTextArea3.append("Wyznacznik  =  "+  determ(taba,size)  +  "\ndalsze
    działania nie mozliwe");
            }
```

Do dalszych rozważań wybrano rozmiar macierzy kwadratowej N=4.

3. Graf zależności informacyjnych

Opracowany program na podstawie danych wejściowych (N) generuje tabele wierzchołków i łuków grafu algorytmu.

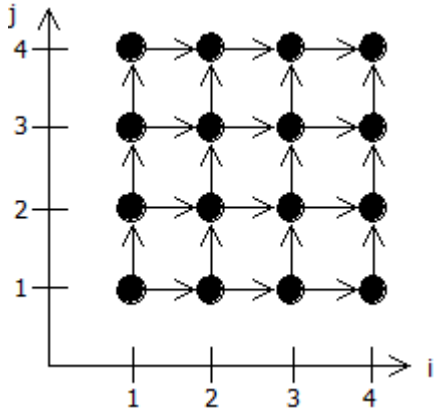
Tab.1. Lista wierzchołków

Lp	i	j	A (i , j)	B (i)	X (i)
1	1	1	(1 , 1)	1	1
2	1	2	(1 , 2)	1	1
3	1	3	(1 , 3)	1	1
4	1	4	(1 , 4)	1	1
5	2	1	(2 , 1)	2	2
6	2	2	(2 , 2)	2	2
7	2	3	(2 , 3)	2	2
8	2	4	(2 , 4)	2	2
9	3	1	(3 , 1)	3	3
10	3	2	(3 , 2)	3	3
11	3	3	(3 , 3)	3	3
12	3	4	(3 , 4)	3	3
13	4	1	(4 , 1)	4	4
14	4	2	(4 , 2)	4	4
15	4	3	(4 , 3)	4	4
16	4	4	(4 , 4)	4	4

Tab.2. Lista łuków

Lp	Początek A(i , j)	Koniec A(i , j)
1	(1 , 1)	(1 , 2)
2	(1 , 1)	(2 , 1)
3	(1 , 2)	(2 , 2)
4	(1 , 2)	(1 , 3)
5	(1 , 3)	(2 , 3)
6	(1 , 3)	(1 , 4)
7	(1 , 4)	(2 , 4)
8	(2 , 1)	(3 , 1)
9	(2 , 1)	(2 , 2)
10	(2 , 2)	(3 , 2)
11	(2 , 2)	(2 , 3)
12	(2 , 3)	(3 , 3)
13	(2 , 3)	(2 , 4)
14	(2 , 4)	(3 , 4)
15	(3 , 1)	(4 , 1)
16	(3 , 1)	(3 , 2)
17	(3 , 2)	(4 , 2)
18	(3 , 2)	(3 , 3)
19	(3 , 3)	(4 , 3)
20	(3 , 3)	(3 , 4)
21	(3 , 4)	(4 , 4)
22	(4 , 1)	(4 , 2)
23	(4 , 2)	(4 , 3)
24	(4 , 3)	(4 , 4)

Na podstawie wyżej wygenerowanych tabel, został utworzony graf :



4. Macierz zależności informacyjnych

Na podstawie grafu wyznaczono macierz zależności informacyjnych $D = [d_1 d_2]$,

$$D = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ ,gdzie } d_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ a } d_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} .$$

5. Odwzorowanie przestrzenno – czasowe

Analizując graf algorytmu możliwe jest wyznaczenie takich parametrów jak :

- liczba wymiarów przestrzeni struktury urządzenia
$$m = 1,$$
- liczba wymiarów przestrzeni grafu
$$n = 2,$$

parametry te decydują o rozmiarze macierzy F (m+1,n). Na podstawie macierzy D oraz wzorów :

$$F_S \cdot d_i = -1 \vee 0 \vee 1,$$
$$F_T \cdot d_i = 1,$$

Gdy w naszym przypadku ‘i’ równe jest 2, otrzymujemy zatem układy równań :

$$F_S \cdot d_1 = -1 \vee 0 \vee 1 \qquad F_T \cdot d_1 = 1$$
$$F_S \cdot d_2 = -1 \vee 0 \vee 1 \qquad F_T \cdot d_2 = 1$$

Wynikiem układów równań są macierze F czyli liniowe odwzorowanie przestrzenno – czasowe

$$F = \begin{bmatrix} F_S \\ F_T \end{bmatrix}.$$

Dla rozważanego przeze nas przypadku istnieje dziewięć możliwych macierzy F, jednak muszą one spełniać warunek

$$\det F \neq 0$$

Po sprawdzeniu zapisanej wyżej nierówności odrzucono trzy spośród dziewięciu możliwych rozwiązań układów równań. W ten sposób określono liczbę struktur realizujących dany algorytm, w rozważanym przypadku liczba ta to ‘6’.

$$F_2 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, \quad F_3 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \quad F_5 = \begin{bmatrix} 0 & -1 \\ 1 & 1 \end{bmatrix}$$
$$F_6 = \begin{bmatrix} -1 & 0 \\ 1 & 1 \end{bmatrix}, \quad F_7 = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \quad F_8 = \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix}$$

6. Liczba elementów przetwarzających oraz taktów

Na podstawie wzoru :

$$F_s \cdot K_i \ ,$$

wyznaczono liczbę elementów przetwarzających dla każdej z struktur, a na podstawie wzoru:

$$F_t \cdot K_i$$

wyznaczono licząc nie powtarzające się wyniki ilość taktów, czyli w jakim czasie zostanie zrealizowana poszczególna struktura

Tab.3. Liczba elementów przetwarzających

Struktura	Liczba elementów przetwarzających	Liczba taktów
F2	4	7
F3	4	7
F5	4	7
F6	4	7
F7	7	7
F8	7	7

7. Połączenia pomiędzy elementami przetwarzającymi

Na podstawie wzoru :

$$F_s \cdot d_i \ ,$$

określamy liczbę, długość i kierunek kanału, czyli połączeń pomiędzy elementami przetwarzającymi.

Tab.4. Połączenia pomiędzy elementami przetwarzającymi

Struktura	Liczba kanałów łączących	Długość i kierunek kanału (0-brak)
F2	1	1 , 0
F3	1	0 , 1
F5	1	-1 , 0
F6	1	0 , -1
F7	2	-1 , 1
F8	2	1 , -1

10. Współczynnik obciążenia

Jednym z ważniejszych parametrów opisujących układ przetwarzający jest współczynnik obciążenia. Wyznacza się go w oparciu o wzór :

$$\Delta = \frac{LiczbaK}{T \cdot LiczbaEP}$$

LiczbaK – liczba wierzchołków grafu,
LiczbaEP – liczba elementów przetwarzających układu,
T – czas realizacji algorytmu

Tab.5. Obciążenie

Struktura	Δ
F2	0,57
F3	0,57
F5	0,57
F6	0,57
F7	0,32
F8	0,32

11. Analiza wyników i wybór układu przetwarzającego

Wcześniejszy graf i wyliczone wartości współczynników obciążenia układów przetwarzających pozwalają wybrać strukturę najbardziej optymalną. Liczba taktów wykonania algorytmu jest stała dla wszystkich architektur, więc szybkość nie będzie cechą decydującą o wyborze. Suma połączeń pomiędzy elementami układów jest różna, więc jest to jedna cecha decydująca o wybraniu optymalnej struktury oraz liczba magistrali również będzie podstawowym kryterium wyboru.

Najmniejszą liczbą elementów przetwarzających cechuje się struktura F2,F3,F5 i F6. Wszystkie elementy przetwarzające wykonują jednakowe zadania. Wydaje się, iż optymalną architekturą będzie realizacja algorytmu za pomocą jednej ze wyżej wymienionych struktur, ponieważ te układy posiadają najlepszy współczynnik obciążenia procesorów, a także wymagają najmniejszej ilości procesorów oraz mniejszej ilości prowadzonych kanałów.