



**POLITECHNIKA
RZESZOWSKA**
im. IGNACEGO ŁUKASIEWICZA

Zadanie projektowe 1

Opracowanie: Sabina Lizis

I rok Inżynierii i Analizy Danych, grupa 1 projektowa



**WYDZIAŁ
MATEMATYKI
I FIZYKI STOSOWANEJ**
POLITECHNIKI RZESZOWSKIEJ

Spis treści

1. Wstęp	3
1.1 Treść zadania	3
1.2. Wstęp teoretyczny	3
2. Pseudokod	5
3. Schemat blokowy	9
4. Działanie programu	10
Graf 1:	11
Graf 2:	12
Graf 3:	13
5. Wnioski	13

Spis rysunków:

Rysunek 1 Odpowiednio: Graf nieskierowany, Graf skierowany	3
Rysunek 2 Listy sąsiedztwa-przykład	4
Rysunek 3 przykład dla grafu z Rysunek 1	5
Rysunek 4 Schemat blokowy dla nr 6	10
Rysunek 5 Konsola dla grafu 1	11
Rysunek 6 Konsola dla grafu 2	12
Rysunek 7 Konsola dla grafu 3	13

1. Wstęp

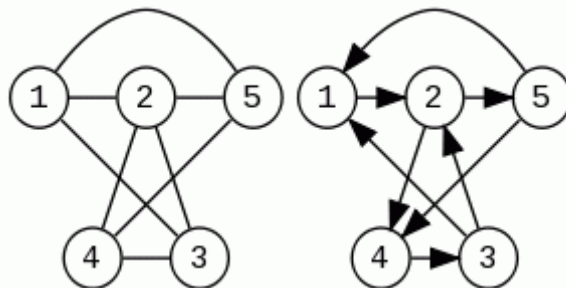
1.1 Treść zadania

Zadanie polegało na implementacji programu, który dla zadanego grafu skierowanego reprezentowanego przy pomocy listy krawędzi wyznaczy i wpisze następujące informacje:

- 1) wszystkich sąsiadów dla każdego wierzchołka grafu (sąsiad danego w_i to ten wierzchołek, do którego prowadzi krawędź z w_i)
- 2) wszystkie wierzchołki, które są sąsiadami danego wierzchołka
- 3) stopnie wychodzące wszystkich wierzchołków
- 4) stopnie wchodzące wszystkich wierzchołków
- 5) wszystkie wierzchołki izolowane
- 6) wszystkie pętle
- 7) wszystkie krawędzie dwukierunkowe

1.2. Wstęp teoretyczny

W informatyce **grafem** nazywamy strukturę $G=(V, E)$ składającą się z węzłów (wierzchołków, oznaczanych przez V) wzajemnie połączonych za pomocą krawędzi (oznaczonych przez E). Grafy dzielimy na grafy skierowane i nieskierowane:



Rysunek 1 Odpowiednio: Graf nieskierowany, Graf skierowany¹

Jeśli krawędź łączy dwa wierzchołki to jest z nimi **incydentna**.

Pętla własna to krawędź łączące wierzchołek z samym sobą.

Stopień wierzchołka w grafie nieskierowanym to liczba incydentnych z nim krawędzi.

Istnieje kilka algorytmów do przechowania grafu w pamięci.²

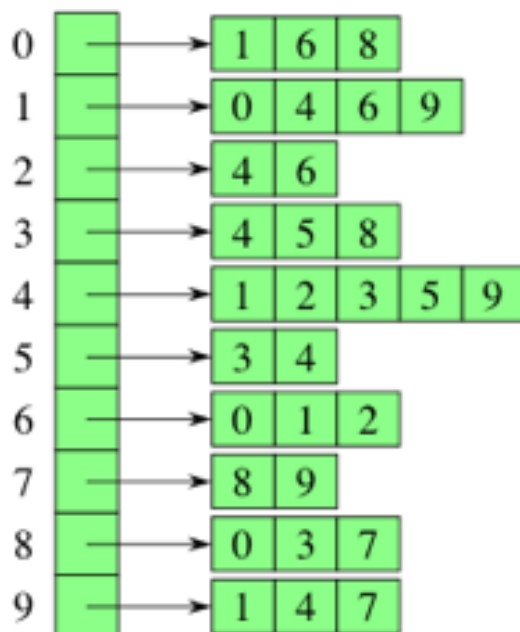
Listy sąsiedztwa

Reprezentowanie grafu przy użyciu **listy sąsiedztwa** jest połączeniem macierzy sąsiedztwa i listy krawędzi. Dla każdego wierzchołka i i przechowywać będziemy tablicę wierzchołków sąsiadujących z nim. Zazwyczaj mamy tablicę $|V||V|$ vertical bar, V , vertical bar list

¹ <http://www.algorytm.org/klasyczne/grafy-i-ich-reprezentacje.html>

² <http://www.algorytm.org/klasyczne/grafy-i-ich-reprezentacje.html>

sąsiedztwa, jedną listę na każdy wierzchołek. I tutaj pokazujemy reprezentację za pomocą list sąsiedztwa dla naszego przykładu grafu sieci społecznej:³



Rysunek 2 Listy sąsiedztwa-przykład⁴

Lista incydencji.

Należy utworzyć listę dla każdego wierzchołka v , w której przechowujemy zbiór wierzchołków połączonych krawędzią z v .

Lista krawędzi jest to lista, na której przechowujemy wszystkie krawędzie występujące w grafie.

Przykład dla grafu skierowanego z Rysunek 1 Odpowiednio: Graf nieskierowany, Graf skierowany

- 1 - 2
- 2 - 4
- 2 - 5
- 3 - 1
- 3 - 2
- 4 - 3
- 5 - 1
- 5 - 4

Zapisując przy pomocy tej reprezentacji graf, w którym występują krawędzie skierowane i nieskierowane należy w przypadku krawędzi nieskierowanej z u do v zapisać krawędź dwukrotnie: $u - v$ oraz $v - u$.

³ <https://pl.khanacademy.org/computing/computer-science/algorithms/graph-representation/a/representing-graphs>

⁴ <https://pl.khanacademy.org/computing/computer-science/algorithms/graph-representation/a/representing-graphs>

Złożoność pamięciowa: $O(E)$

Macierz incydencji to tablica o rozmiarach $V \times E$. Składa się ona z E kolumn i V wierszy, jeśli krawędź wychodzi z danego wierzchołka to piszemy w odpowiedniej kolumnie (-1), jeśli do niego wchodzi piszemy (+1), jeśli wierzchołek nie należy do krawędzi piszemy 0, jeśli jest to pętla własna piszemy 2.

Oto przykład dla grafu z Rysunek 1 Odpowiednio: Graf nieskierowany, Graf skierowany

	1	2	3	4	5
1 - 2	-1	1	0	0	0
1 - 3	1	0	-1	0	0
1 - 5	1	0	0	0	-1
2 - 4	0	-1	0	1	0
2 - 5	0	-1	0	0	1
2 - 3	0	1	-1	0	0
3 - 4	0	0	1	-1	0
5 - 1	1	0	0	0	-1
5 - 4	0	0	0	1	-1

Rysunek 3 przykład dla grafu z Rysunek 1

Macierz grafu została opracowana w Instytucie Informatyki PP przez dr M. Sternę. Jest to trochę bardziej skomplikowana reprezentacja grafu niż poprzednie.

Należy utworzyć tablicę o rozmiarach $(V+2)^2$. Wiersze i kolumny numerujemy od 0 do $V+1$.

Najpierw zajmujemy się częścią macierzy ograniczoną przez indeksy od 1 do V . Zał. że w wierszach mamy i -te wierzchołki a w kolumnach j -te wierzchołki. Liczby, które mogą znaleźć się na skrzyżowaniu wierzchołka i oraz j można podzielić na 3 grupy:

- od 1 do V - istnieje krawędź skierowana: $i \leftarrow j$
- od $V+1$ do $2 \cdot V$ - istnieje krawędź skierowana: $i \rightarrow j$
- od $(-V)$ do (-1) - brak incydentnych krawędzi⁵

2. Pseudokod

```
int liczba_wierzchołkow_grafu
```

```
{
```

```
    int max_wierzcholek=-1;
```

```
    dla i=0; i<graf.size(); wykonuj(++)
```

```
        { max_wierzcholek=max(max_wierzcholek, graf[i][0]);
```

⁵ <http://www.algorytm.org/klasyczne/grafy-i-ich-reprezentacje.html>

```

        max_wierzcholek=max(max_wierzcholek,graf[i][1]);
    }
Zwróć max_wierzcholek;
void wypisz_graf{ Dla i=0;i<graf.size() wykonuj i++)}
void wszyscy_sasiedzi
{
    Dla i=0;i<=liczba_wierzchołkow; wykonuj i++)
    { dla j=0;j<=liczba_wierzchołkow; wykonuj j++)    {
        sasiedzi[j]=false; }
    dla j=0; j<graf.size(); wykonuj j++)
    {
        Jeśli (graf[j][0]==i { To sasiedzi[j]=true;
        { dla j=0;j<=liczba_wierzchołkow;j wykonuj j++)
        {
            Jeśli (sasiedzi[j]==true)
            {
                To wypisz: j
            { void wszystkie_wierzchołki_sasiedzi_wszystkich_wierzchołkow
        { dla i=0;i<=liczba_wierzchołkow; wykonuj i++ {
        Dla j=0;j<=liczba_wierzchołkow; wykonuj j++)
            { sasiedzi[j]=false;
        }
        dla j=0;j<graf.size(); wykonuj j++)    {
        jeśli (graf[j][1]==i)
            {To sasiedzi[graf[j][0]]=true;}
        }
        wtedy bool flag=true;
        dla j=0;j<=liczba_wierzchołkow; wykonuj j++
        {
            Jeśli (sasiedzi[j]==false)

```

```
    { To flag=false;
      break;  } }
```

Jeśli (flag==true)

```
    { To wypisz i  }  }}
```

void stopnie_wychodzace

```
{
    dla i=0;i<=liczba_wierzchołkow; wykonuj i++)    {
        dla j=0;j<graf.size(); wykonuj j++
        {
            Jeśli (graf[j][0]==i)
                { To wykonuj stopien++;  }}
    Wypisz stopień wychodzący wierzchołka  }}
```

void stopnie_wchodzace

```
{ Wypisz: stopnie wchodzące wierzchołków
Dla i=0;i<=liczba_wierzchołkow; wykonuj i++)    {
    Dla j=0;j<graf.size(); wykonuj j++)
        { Jeśli (graf[j][1]==i)    { To wykonuj stopien++;
          }
        }
    Wypisz stopień wchodzący wierzchołka    }
}
```

void wierzchołki_izolowane{

wypisz: wierzchołki izolowane:

```
dla int i=0;i<=liczba_wierzchołkow; wykonuj i++)    {
dla int j=0;j<graf.size(); wykonuj j++)
    {
        Jeśli (graf[j][1]==i)
            { wykonuj stopien++  }
        }Jeśli (stopien==0)
```

```
    {Wypisz i} } }
```

```
void petle
```

```
{
```

```
    dla i=0;i<liczba_wierzchołkow; wykonuj i++
```

```
    { wtedy petle[i]=false }
```

```
    dla i=0;i<graf.size();wykonuj i++)
```

```
    {
```

```
        Jeśli (graf[i][0]==graf[i][1])
```

```
        {To petle[graf[i][0]]=true; } }
```

```
Wypisz: wszystkie wierzchołki które posiadają pętle
```

```
dla i=0;i<liczba_wierzchołkow; wykonuj i++
```

```
    { Jeśli (petle[i]==true)
```

```
    {Wypisz i} } }
```

```
void krawedzi_dwustronne
```

```
{    dla(int i=0;i<krawedzie.size(); wypisz i++)
```

```
    {
```

```
    To krawedzie[i]=false;
```

```
    } Dla i=0;i<graf.size()-1; wykonuj i++)
```

```
    {dla j=i+1; j<graf.size(); wykonuj j++)
```

```
    {
```

```
    Jeśli (graf[i][0]==graf[j][1] i graf[i][1]==graf[j][0])
```

```
        { to krawedzie[i]=true; } }
```

```
Wypisz krawędzie dwustronne
```

```
dla i=0;i<krawedzie.size();wykonuj i++)
```

```
    {
```

```
    Jeśli krawedzie[i]==true)
```

```
    {
```

```
        To wypisz graf[i][0]<<" "<<graf[i][1]<<" i "<<graf[i][1]<<" "<<graf[i][0]<<"\n"; } }
```

```
int main()
```

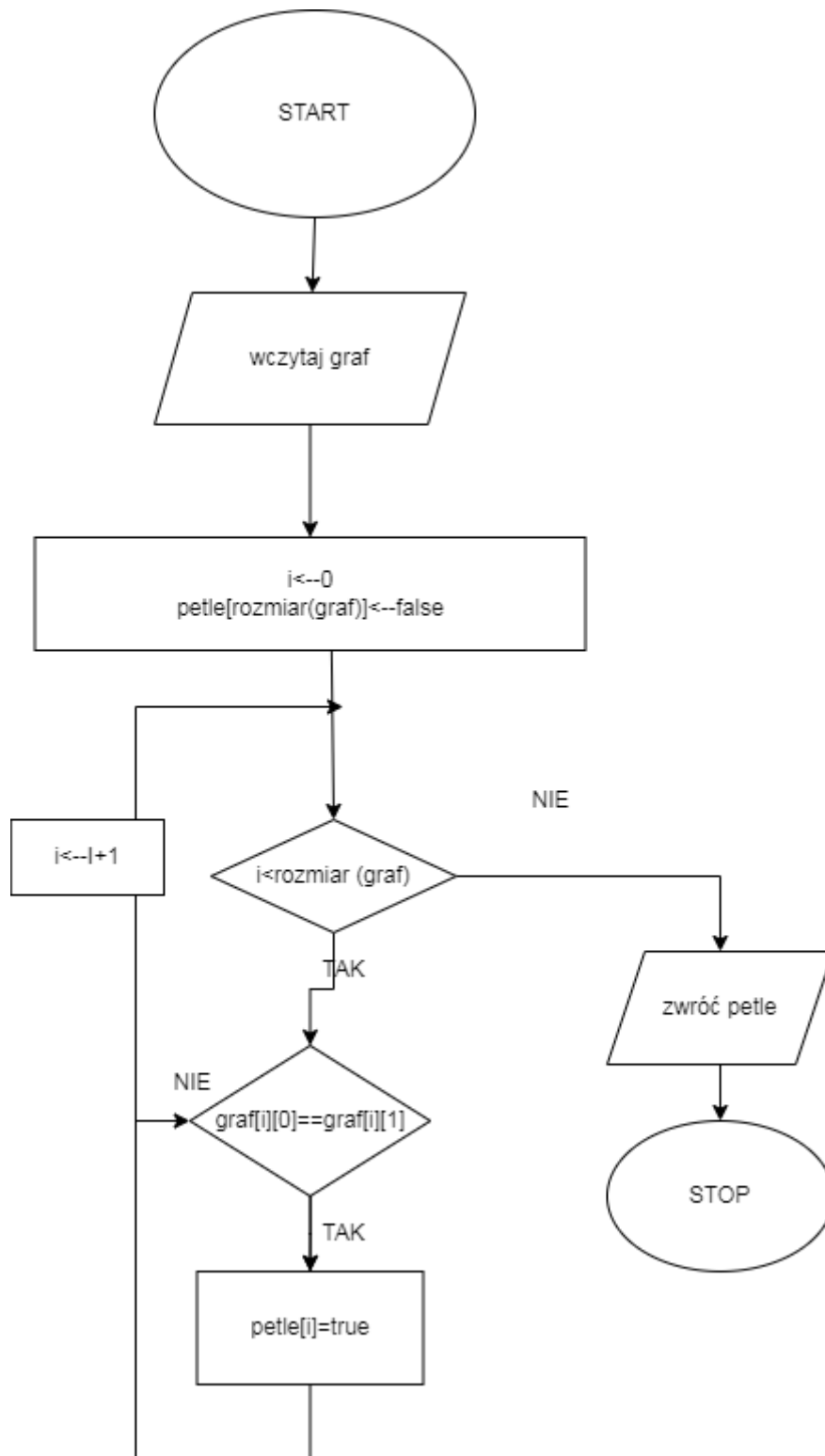
```
{
```



```
graf=wczytaj_graf("graf.txt");  
wszyscy_sasiedzi(graf);  
wypisz_graf(graf);  
wszystkie_wierzcholki_sasiedzi_wszystkich_wiercholkow(graf);  
stopnie_wychodzace(graf);  
stopnie_wchodzace(graf);  
wierzcholki_izolowane(graf);  
petle(graf);  
krawedzi_dwustronne(graf);  
zapisz_graf_do_pliku(graf,"Sabina.txt");  
return 0;  
}
```

3. Schemat blokowy

Poniżej prezentuję przykładowy schemat blokowy dla punktu nr 6 (czyli prezentacja pętli).



Rysunek 4 Schemat blokowy dla nr 6

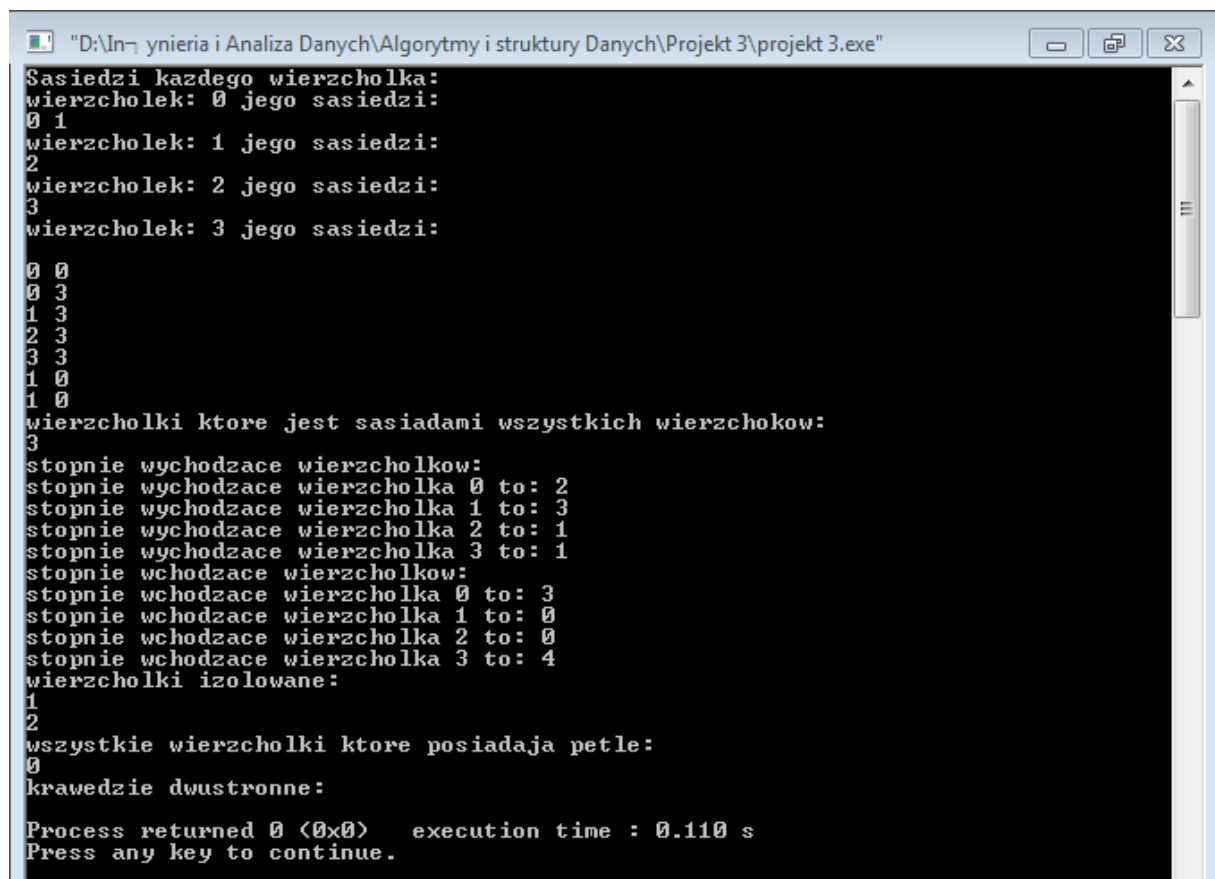
4. Działanie programu

Poniżej prezentuję działanie programu, które obejmuje odczyt z pliku (graf.txt) oraz zapis do pliku (Sabina.txt). W programie zaimplementowane zostały testy sprawdzające poprawność działania programu.

Przy pomocy listy krawędzi program wyznacza i wpisze następujące informacje, które widzimy na konsoli:

- 1) wszystkich sąsiadów dla każdego wierzchołka grafu (sąsiad danego w_i to ten wierzchołek, do którego prowadzi krawędź z w_i)
- 2) wszystkie wierzchołki, które są sąsiadami danego wierzchołka
- 3) stopnie wychodzące wszystkich wierzchołków
- 4) stopnie wchodzące wszystkich wierzchołków
- 5) wszystkie wierzchołki izolowane
- 6) wszystkie pętle
- 7) wszystkie krawędzie dwukierunkowe

Graf 1:



```
"D:\Inżynieria i Analiza Danych\Algorytmy i struktury Danych\Projekt 3\projekt 3.exe"
Sasiedzi kazdego wierzchołka:
wierzcholek: 0 jego sasiedzi:
0 1
wierzcholek: 1 jego sasiedzi:
2
wierzcholek: 2 jego sasiedzi:
3
wierzcholek: 3 jego sasiedzi:
0 0
0 3
1 3
2 3
3 3
1 0
1 0
wierzchołki ktore jest sasiadami wszystkich wierzchołkow:
3
stopnie wychodzace wierzchołkow:
stopnie wychodzace wierzchołka 0 to: 2
stopnie wychodzace wierzchołka 1 to: 3
stopnie wychodzace wierzchołka 2 to: 1
stopnie wychodzace wierzchołka 3 to: 1
stopnie wchodzace wierzchołkow:
stopnie wchodzace wierzchołka 0 to: 3
stopnie wchodzace wierzchołka 1 to: 0
stopnie wchodzace wierzchołka 2 to: 0
stopnie wchodzace wierzchołka 3 to: 4
wierzchołki izolowane:
1
2
wszystkie wierzchołki ktore posiadaja petle:
0
krawedzie dwustronne:

Process returned 0 (0x0)   execution time : 0.110 s
Press any key to continue.
```

Rysunek 5 Konsola dla grafu 1

Graf 2:

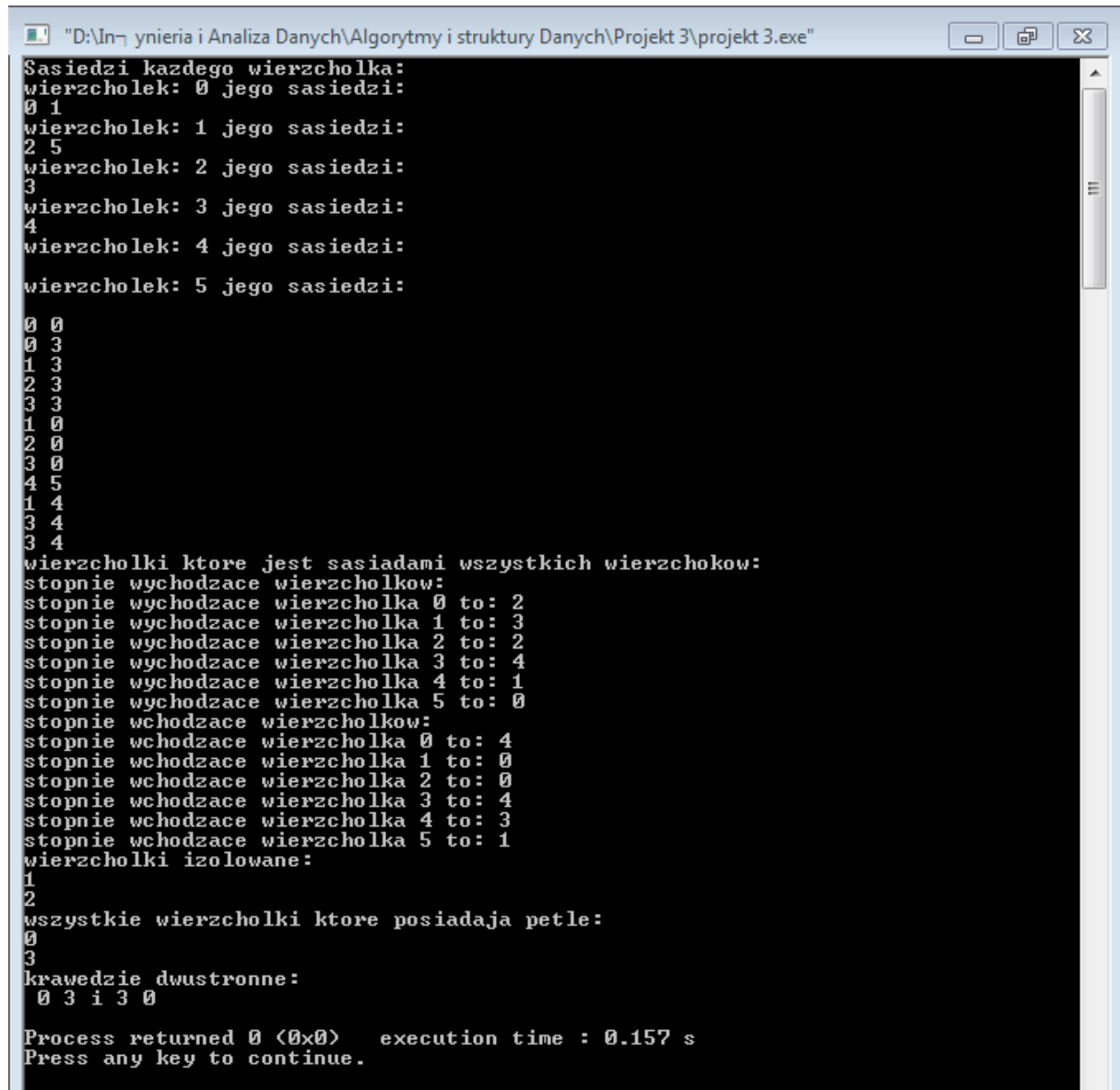
```
"D:\Inżynieria i Analiza Danych\Algorytmy i struktury Danych\Projekt 3\projekt 3.exe"
Sasiadzi kazdego wierzchołka:
wierzchołek: 0 jego sasiadzi:
0 1
wierzchołek: 1 jego sasiadzi:
2 5
wierzchołek: 2 jego sasiadzi:
3
wierzchołek: 3 jego sasiadzi:
4
wierzchołek: 4 jego sasiadzi:
wierzchołek: 5 jego sasiadzi:

0 0
0 3
1 3
2 3
3 3
1 0
2 0
3 0
4 5
1 4
3 4
0 3
0 0
4 5
4 0
wierzchołki ktore jest sasiadami wszystkich wierzchołkow:
stopnie wychodzace wierzchołkow:
stopnie wychodzace wierzchołka 0 to: 4
stopnie wychodzace wierzchołka 1 to: 3
stopnie wychodzace wierzchołka 2 to: 2
stopnie wychodzace wierzchołka 3 to: 3
stopnie wychodzace wierzchołka 4 to: 3
stopnie wychodzace wierzchołka 5 to: 0
stopnie wchodzace wierzchołkow:
stopnie wchodzace wierzchołka 0 to: 6
stopnie wchodzace wierzchołka 1 to: 0
stopnie wchodzace wierzchołka 2 to: 0
stopnie wchodzace wierzchołka 3 to: 5
stopnie wchodzace wierzchołka 4 to: 2
stopnie wchodzace wierzchołka 5 to: 2
wierzchołki izolowane:
1
2
wszystkie wierzchołki ktore posiadaja petle:
0
3
krawedzie dwustronne:
0 0 i 0 0
0 3 i 3 0
3 0 i 0 3

Process returned 0 (0x0)   execution time : 0.126 s
Press any key to continue.
```

Rysunek 6 Konsola dla grafu 2

Graf 3:



```
"D:\Inżynieria i Analiza Danych\Algorytmy i struktury Danych\Projekt 3\projekt 3.exe"
Sasiadzi kazdego wierzchołka:
wierzcholek: 0 jego sasiadzi:
0 1
wierzcholek: 1 jego sasiadzi:
2 5
wierzcholek: 2 jego sasiadzi:
3
wierzcholek: 3 jego sasiadzi:
4
wierzcholek: 4 jego sasiadzi:
wierzcholek: 5 jego sasiadzi:

0 0
0 3
1 3
2 3
3 3
1 0
2 0
3 0
4 5
1 4
3 4
3 4
wierzchołki ktore jest sasiadami wszystkich wierzchołkow:
stopnie wychodzace wierzchołkow:
stopnie wychodzace wierzchołka 0 to: 2
stopnie wychodzace wierzchołka 1 to: 3
stopnie wychodzace wierzchołka 2 to: 2
stopnie wychodzace wierzchołka 3 to: 4
stopnie wychodzace wierzchołka 4 to: 1
stopnie wychodzace wierzchołka 5 to: 0
stopnie wchodzace wierzchołkow:
stopnie wchodzace wierzchołka 0 to: 4
stopnie wchodzace wierzchołka 1 to: 0
stopnie wchodzace wierzchołka 2 to: 0
stopnie wchodzace wierzchołka 3 to: 4
stopnie wchodzace wierzchołka 4 to: 3
stopnie wchodzace wierzchołka 5 to: 1
wierzchołki izolowane:
1
2
wszystkie wierzchołki ktore posiadaja petle:
0
3
krawedzie dwustronne:
0 3 i 3 0
Process returned 0 (0x0)   execution time : 0.157 s
Press any key to continue.
```

Rysunek 7 Konsola dla grafu 3

5. Wnioski

Jak widać przygotowany przeze mnie program działa jak najbardziej poprawnie. Zostało to sprawdzone przy pomocy testów umieszczonych w kodzie programu. Czas pracy działania programu dla przykładowych 3 grafów umieszczonych w sprawozdaniu wynosił odpowiednio 0,110s, 0,126s oraz 0,157s co wskazuje na szybką pracę programu. Podsumowując, implementacja takiego kodu programu nie była rzeczą najłatwiejszą, ale finalnie udało mi się uzyskać program, który działa poprawnie, na co wskazują testy zaimplementowane w kodzie programu.