P O L I T E C H N I K A R Z E S Z O W S K A
im. Ignacego Łukasiewicza
WYDZIAŁ MATEMATYKI I FIZYKI STOSOWANEJ

STACH KACPER

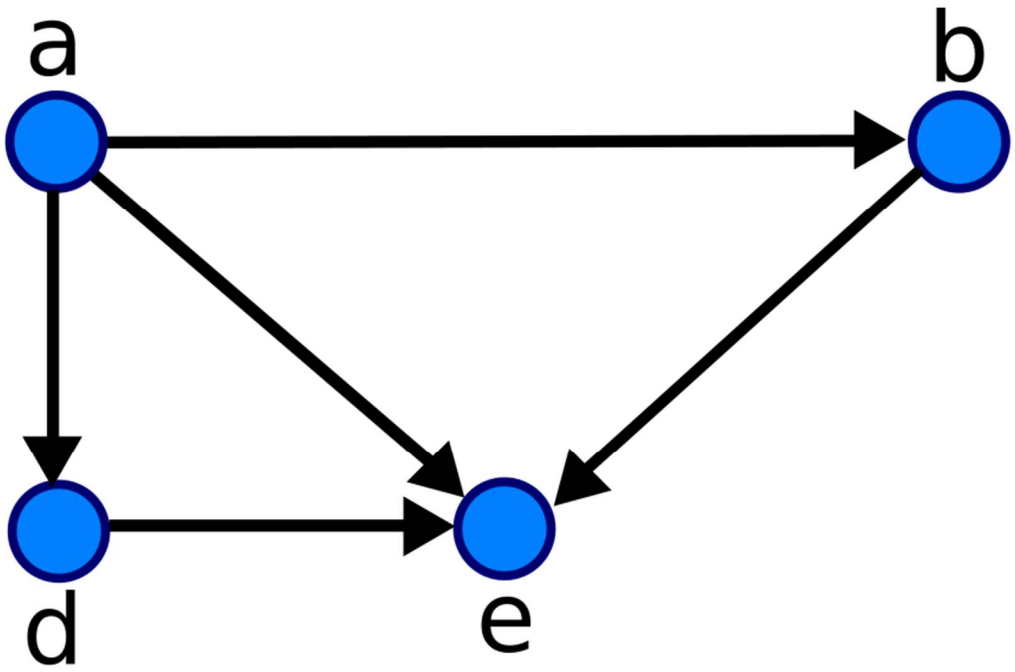
Projekt #3 – funkcje operujące na grafie skierowanym reprezentowanym przez macierz sąsiedztwa.

Rzeszów 2023

Spis treści

1. Czym jest graf skierowany?

Graf skierowany jest rodzajem grafu, który definiujemy jako uporządkowaną parę zbiorów. Pierwszy z nich zawiera wierzchołki grafu, a drugi składa się z krawędzi grafu, czyli uporządkowanych par wierzchołków. Po grafie możemy poruszać się tylko w kierunkach wskazywanych przez krawędzie.



Rysunek 1 - Graf skierowany

2. OPISY ORAZ PSEUDOKOD FUNKCJI

Poniżej opisane zostały następujące funkcje programu:

1) Funkcja pokazująca sąsiadów dla każdego wierzchołka grafu,
2) Funkcja wypisująca wszystkie wierzchołki, które są sąsiadami każdego wierzchołka,
3) Funkcja pokazująca stopnie wychodzące wszystkich wierzchołków,
4) Funkcja pokazująca stopnie wchodzące wszystkich wierzchołków,
5) Funkcja wypisująca wierzchołki izolowane,
6) Funkcja wypisująca pętle,
7) Funkcja wypisująca wszystkie krawędzie dwukierunkowe.

1. Funkcja pokazująca sąsiadów dla każdego wierzchołka grafu,

W grafie reprezentowanym przez macierz sąsiedztwa możemy pokazać dane, posługując się wartościami w naszej macierzy. W tym przypadku posługujemy się dwoma pętlami, które poruszają się po wierszach i kolumnach macierzy. W tym przypadku, gdy w danym wierszu i kolumnie znajdzie się wartość „1” oznacza to, że wierzchołek [i] ma sąsiada [j].

```
void podpunkt_1(char **A, int n)
{
    cout<<"1)"<<endl;
  for( int i = 0; i < n; i++ )
  {
      int c=0;
      //zmienna pozwalajaca policzyc ilu sasiadowa ma wierzcholek
      cout<<"wierzcholek "<<i<<" "<<"sasiedzi to wierzcholki numer"<<":";
      for( int j = 0; j < n; j++ )
    //przeszukujemy i-ty wiersz jesli jest w nim 1 to znalezlismy sasiada
      {
          if(A[i][j]==1)
          {
             c++;
             cout<<j<<",";
          }
      }


      if(c==0)
            cout<<"brak";//jesli nie ma sasiadow
      cout<<endl;
  }
}
```

Rysunek 2 - Kod funkcji nr. 1

```
1  Funkcja "podpunkt_1"
2
3  wypisuje na ekran "1)" i nowa linie
4  dla i od 0 do n-1 wykonuj:
5  tworzy zmienna c i ustawia ja na 0
6  wypisuje na ekran "wierzcholek " i aktualna wartosc i oraz
7  " sasiedzi to wierzcholki numer:"
8  dla j od 0 do n-1 wykonuj:
9  jesli element A[i][j] jest rowny 1, to:
10 zwieksza c o 1
11 wypisuje na ekran j i przecinek
12 jesli c jest rowne 0, to wypisuje na ekran "brak"
13 wypisuje nowa linie
14 konczy dzialanie funkcji
```

Rysunek 3 - Pseudokod funkcji nr.1

2. Funkcja wypisująca wszystkie wierzchołki, które są sąsiadami każdego wierzchołka,

Aby znaleźć wierzchołek, który jest sąsiadem każdego wierzchołka, należy zliczyć ilość sąsiadów. Jeśli jest taka sama jak ilość krawędzi, to wierzchołek spełnia nasz warunek.

```
void podpunkt_2(char **A, int n,int m)
{
    int c=0;
    cout<<"2)"<<endl;
  for( int i = 0; i < n; i++ )
  {
      int b=0;
      for( int j = 0; j < n; j++ )
        //przeszukujemy wiersz i-ty
      {
          if(A[i][j]==1)
          //jesli komorka jest rowna 1 wierzcholek jest sasiadanem
          {
             b++;//zliczamy sasiadow
          }

      }
       if(b==m)
       {
           c++;
           cout<<"wierzcholek "<<i<<" jest sasiadem kazdego wierzcholka"<<endl;
           }
  }
  if(c==0)
cout<<"brak wierzcholkow,ktore sa sasiadami kazdego wierzcholka"<<endl;

}
```

Rysunek 4 - Kod funkcji nr.2

```
1  podpunkt_2(tablica dwuwymiarowa znakow A,
2  |liczba naturalna n, liczba naturalna m)
3  zmienna calkowita c = 0
4  wypisz "2)"
5  dla kazdej liczby naturalnej i od 0 do n-1
6  zmienna calkowita b = 0
7  dla kazdej liczby naturalnej j od 0 do n-1
8  //przeszukujemy wiersz i-ty
9  jezeli A[i][j] jest rowne 1
10 //jeśli komórka jest równa 1 wierzchołek jest sąsiadem
11 b++; //zliczamy sąsiadów
12 koniec petli
13 jezeli b jest rowne m
14 c++;
15 wypisz "wierzchołek " i " jest sąsiadem każdego wierzchołka"
16 koniec petli
17 jezeli c jest rpwne 0
18 wypisz "brak wierzchołków, które są sąsiadami każdego wierzchołka"
```

Rysunek 5 - Pseudokod funkcji nr.2

3. Funkcja zliczająca stopnie wychodzące wszystkich wierzchołków.

Aby zliczyć stopnie wychodzące wszystkich wierzchołków, przeszukujemy wiersz w poszukiwaniu wartości „1”. Po znalezieniu inkrementujemy zmienną zliczającą stopnie wychodzące.

```
void podpunkt_3(char **A, int n)
{
    cout<<"3)"<<endl;
  for(int i = 0; i < n; i++ )
  {
      int wychodzacy=0;
      cout<<"wierzcholek "<<i<<" "<<"stopien wychodzacy"<<":";

      for( int j = 0; j < n; j++ )//przeszukujemy wiersz
      {
          if(A[i][j]==1)
          {
              wychodzacy++;//zliczmy sasiadow
          }
      }
      cout<<wychodzacy<<endl;
  }
}
```

Rysunek 6 - Kod funkcji nr.3

```
1  podpunkt_3(tablica dwuwymiarowa znakow A, liczba naturalna n)
2  wypisz "3)"
3  dla kazdej liczby naturalnej i od 0 do n-1
4  zmienna calkowita wychodzacy = 0
5  wypisz "wierzchołek " i "stopień wychodzący :"
6  dla kazdej liczby naturalnej j od 0 do n-1
7  //przeszukujemy wiersz
8  jezeli A[i][j] jest rowne 1
9  wychodzacy++; //zliczamy sąsiadów
10 wypisz wychodzacy i przejdz do nowej linii
11 koniec petliiadami kazdego wierzcholka"
```

Rysunek 7 - Pseudokod funkcji nr.3

4. Funkcja zliczająca stopnie wchodzące wszystkich wierzchołków.

Aby zliczyć stopnie wychodzące wszystkich wierzchołków, przeszukujemy kolumnę w poszukiwaniu wartości „1”. Po znalezieniu inkrementujemy zmienną zliczającą stopnie wchodzące.

```
void podpunkt_4(char **A, int n)
{
    cout<<"4)"<<endl;
  for(int i = 0; i < n; i++ )
  {
      int wchodzacy=0;
      cout<<"wierzcholek "<<i<<" "<<"stopien wchodzacy"<<":";

      for(int j = 0; j < n; j++ )//przeszukujemy kolumne
      {
          if(A[j][i]==1)
          {
             wchodzacy++;
          }
      }
      cout<<wchodzacy<<endl;
  }
}
```

Rysunek 8 - Kod funkcji nr.4

```
1  podpunkt_4(tablica dwuwymiarowa znakow A, liczba naturalna n)
2  wypisz "4)"
3  dla kazdej liczby naturalnej i od 0 do n-1
4  zmienna calkowita wchodzacy = 0
5  wypisz "wierzchołek " i "stopień wchodzący :"
6  dla kazdej liczby naturalnej j od 0 do n-1
7  //przeszukujemy wiersz
8  jezeli A[j][i] jest rowne 1
9  wchodzacy++; //zliczamy sąsiadów
10 wypisz wchodzacy i przejdz do nowej linii
11 koniec petliiadami kazdego wierzcholka"
```

Rysunek 9 - Pseudokod funkcji nr.4

5. Funkcja zliczająca wierzchołki izolowane.

Wierzchołki izolowane znajdziemy obliczając, czy posiada on jakiekolwiek krawędzie. Jeżeli liczba krawędzi jest równa 0, to wierzchołek jest izolowany.

```
void podpunkt_5(char **A, int n)
{
    int c=0;
    cout<<"5)"<<endl;
  for(int i = 0; i < n; i++ )
  {
      int b=0;
      for(int j = 0; j < n; j++ )
      {
          if(A[j][i]==1 || A[i][j]==1)
        //przeszukujemy kolumne i wiersza to lub to musi byc rowne jeden
          {
             b++;
          }
      }
      if(b==0)
      {
          cout<<"wierzcholek "<<i<<" jest izolowany"<<endl;
          c++;
      }
  }
  if(c==0) cout<<"brak wierzcholkow izolowanych"<<endl;
}
```

Rysunek 10 - Kod funkcji nr.5

```
1  podpunkt_5(tablica dwuwymiarowa znakow A, liczba naturalna n)
2  zmienna calkowita c = 0
3  wypisz "5)"
4  dla kazdej liczby naturalnej i od 0 do n-1
5  zmienna calkowita b = 0
6  dla kazdej liczby naturalnej j od 0 do n-1
7  //przeszukujemy kolumnę i wiersz
8  jezeli A[j][i] jest rowne 1 lub A[i][j] jest rowne 1
9  b++;
10 koniec petli
11 jezeli b jest rowne 0
12 wypisz "wierzchołek " i " jest izolowany"
13 c++;
14 koniec petli
15 jezeli c jest rowne 0
16 wypisz "brak wierzchołków izolowanych"
```

Rysunek 11 - Pseudokod funkcji nr.5

6. Funkcja zliczająca pętle

Aby obliczyć pętlę należy sprawdzić, czy w i-tej kolumnie i i-tym wierszu znajdziemy wartość „1”.

```
void podpunkt_6(char **A, int n)
{
    cout<<"6)"<<endl;
  int c=0;
  for(int i = 0; i < n; i++ )
//w celu sprawdzenia wartosci i-tej kolumny i-tego wiersza
  {
      if(A[i][i]==1)
        {
            cout<<"wierzcholek "<<i<<" posiada petle"<<endl;
            c++;//zliczanie ilosci petli
        }
  }
  if(c==0) cout<<"brak petli"<<endl;//jesli nie ma petli
}
```

Rysunek 12 - Kod funkcji nr.6

```
1  podpunkt_6(tablica dwuwymiarowa znakow A, liczba naturalna n)
2  wypisz "6)"
3  zmienna calkowita c = 0
4  dla kazdej liczby naturalnej i od 0 do n-1
5  //w celu sprawdzenia wartości i-tej kolumny i-tego wiersza
6  jezeli A[i][i] jest rowne 1
7  wypisz "wierzchołek " i " posiada pętle"
8  c++; //zliczanie ilości pętli
9  koniec petli
10 jesli c jest rowne 0
11 wypisz "brak pętli"
```

Rysunek 13 - Pseudokod funkcji nr.6

7. Funkcja znajdująca krawędzie dwukierunkowe

```
void podpunkt_7(char **A, int n)
{
    int a,c=0;
    cout<<"7)"<<endl;
  for(int i = 0; i < n; i++ )
  {
      int a;
      if(i==a) i++;
      //aby nie wypisywac dwa razy tego samego tylko w innej kolejnosci wierzcholkow
      for(int j = 0; j < n; j++ )
      {
          if(A[j][i]==1&&A[i][j]==1&&j!=i)
          {
             cout<<"wierzcholek "<<i<<" i "<<j<<" laczy krawedz dwukierunkowa"<<endl;
             a=j;
             c++;//zlicenie ilosci krawedzi dwukierunkowych
          }
      }
  }
  if(c==0) cout<<"nie ma wierzcholkow ktore maja krawedz dwukierunkowa"<<endl;
  //aby zaznaczyc ze ich nie ma
}
```

Rysunek 14 - Kod funkcji nr.7

```
1  podpunkt_7(tablica dwuwymiarowa znakow A, liczba naturalna n)
2  zmienna calkowita a, zmienna calkowita c = 0
3  wypisz "7)"
4  dla kazdej liczby naturalnej i od 0 do n-1
5  zmienna calkowita a
6  jesli i jest rowne a to i++;
7  dla kazdej liczby naturalnej j od 0 do n-1
8  jesli A[j][i] jest rowne 1 i A[i][j] jest rowne 1 i j jest rozne od i
9  wypisz "wierzchołek " i " i " j " łączy krawędź dwukierunkową"
10 a = j
11 c++;
12 koniec petli
13 jesli c jest rowne 0
14 wypisz "nie ma wierzchołków, które mają krawędź dwukierunkową"
```

Rysunek 15 - Pseudokod funkcji nr.7

3. Funkcja main.

```
int main( )
{
  int n, m,a,b;
  char ** A;
  cout<<"Podaj ilosc wierzcholkow"<<endl;
  cin >> n;//Czytamy liczbe wierzcholkow
  cout<<"Podaj ilosc krawedzi"<<endl;
  cin >> m;//  wczytujemy krawedzi
  cout<<"podaj liste krawedzi"<<endl;
  A = new char *[n];  // Tworzymy tablice wskaznikow

  for(int i = 0; i < n; i++ )
    A [i] = new char [n]; // Tworzymy wiersze
  // Macierz wypelniamy zerami
  for(int i = 0; i < n; i++ )
    for(int j = 0; j < n; j++ ) A [ i ][ j ] = 0;
  // Odczytujemy kolejne definicje krawędzi
  for(int i = 0; i < m; i++ )
  { //cout<<"podaj wierzcholek startowy numer:"<<i<<endl;
    cin >> a;// Wierzcholek startowy krawedzi
    //cout<<"podaj wierzcholek koncowy numer:"<<i<<endl;
    cin>> b;// Wierzcholek koncowy krawedzi
    A [a][b] = 1; // krawedz a,b
  }
  cout << endl;
  //wywolania funkcji
  podpunkt_1(A,n);
  podpunkt_2(A,n,m);
  podpunkt_3(A,n);
  podpunkt_4(A,n);
  podpunkt_5(A,n);
  podpunkt_6(A,n);
  podpunkt_7(A,n);
  return 0;
}
```

Rysunek 16 - Kod funkcji main