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

KRZYSZTOF KOPIEC
173158

ALGORYTMY I STRUKTURY DANYCH

Projekt

kierunek studiów: inżynieria i analiza danych

Opiekun pracy:

Prof. Mariusz Borkowski

Rzeszów 2023

1. Wstęp

Założeniem mojego projektu jest napisanie programu, który dla zadanego grafu skierowanego przy pomocy macierzy sąsiedztwa wyznaczy i wypisze : wszystkich sąsiadów dla każdego wierzchołka grafu, wszystkie wierzchołki, które są sąsiadami każdego wierzchołka, stopnie wychodzące i wchodzące wszystkich wierzchołków, wszystkie wierzchołki izolowane, wszystkie pętle oraz wszystkie krawędzie dwukierunkowe.

2. Pseudokody

2.1 Wszystkie pętle

K01: Ustaw zmienną "i" na 0

K02: Dla "i" od 0 do liczby wierzchołków wykonuj:

K02.1: Jeśli macierz[i][i] jest równe 1:

    Wypisz "Wierzcholek " + i + " posiada petle"

K03: Koniec pętli.

2.2 Wszystkie krawędzie dwukierunkowe

K01: Ustaw zmienne "i" i "j" na 0

K02: Dla "i" od 0 do liczby wierzchołków wykonuj:

K02.1: Dla "j" od 0 do liczby wierzchołków wykonuj:

K02.1.1: Jeśli macierz[i][j] jest równe 1 i macierz[j][i] jest równe 1 oraz "i" jest różne od "j":

Wypisz "Wierzcholek “+i+" posiada krawędź dwukierunkową z wierzchołkiem " + j

K03: Koniec pętli zewnętrznej.

2.3 Wszystkie wierzchołki izolowane

K01: Ustaw zmienne "i", "j" oraz "a" na 0

K02: Dla "i" od 0 do liczby wierzchołków wykonuj:

K02.1: Ustaw "a" na 0

K03: Dla "j" od 0 do liczby wierzchołków wykonuj:

K03.1: Jeśli macierz[i][j] jest równe 0 i macierz[j][i] jest równe 0:

Zwiększ "a" o 1

K03.2: Jeśli "a" jest równe liczbie wierzchołków:

Wypisz "Wierzchołek " + i + " jest wierzchołkiem izolowanym"

K04: Koniec pętli zewnętrznej.

2.4 Wszyscy sąsiedzi dla każdego wierzchołka grafu

K01: Ustaw zmienne "i" i "j" na 0

K02: Dla "i" od 0 do liczby wierzchołków wykonuj:

K02.1: Dla "j" od 0 do liczby wierzchołków wykonuj:

K02.1.1: Jeśli macierz[i][j] jest równe 1 i "i" jest różne od "j":

Wypisz "Sąsiadem wierzchołka " + i + " jest wierzchołek " + j

K03: Koniec pętli zewnętrznej.

2.5 Wszystkie stopnie wychodzące i wchodzące wszystkich wierzchołków

K01: Ustaw zmienne "i", "j", "a" oraz "b" na 0

K02: Dla "i" od 0 do liczby wierzchołków wykonuj:

K02.1: Ustaw "a" i "b" na 0

K03: Dla "j" od 0 do liczby wierzchołków wykonuj:

K03.1: Jeśli macierz[i][j] jest równe 1:

K03.1.1: Zwiększ "a" o 1

K03.2: Jeśli macierz[j][i] jest równe 1:

K03.2.1: Zwiększ "b" o 1

K04: Wypisz "Stopień wchodzący wierzchołka " + i + " to: " + b

K05: Wypisz "Stopień wychodzący wierzchołka " + i + " to: " + a

K06: Koniec pętli zewnętrznej.

2.6 Wszystkie wierzchołki, które są sąsiadami każdego wierzchołka

K01: Ustaw zmienne "i", "j" i "a" na 0

K02: Dla "i" od 0 do liczby wierzchołków wykonuj:

K02.1: Ustaw "a" na 0

K03: Dla "j" od 0 do liczby wierzchołków wykonuj:

K03.1: Jeśli macierz[i][j] jest równe 1:

K03.1.1: Zwiększ "a" o 1

K03.2: Jeśli "a" jest równe liczbie wierzchołków lub "a" jest równe liczbie wierzchołków - 1:

K03.2.1: Wypisz "Wierzchołek " + i + " jest sąsiadem każdego wierzchołka"

K04: Koniec pętli zewnętrznej.

3. Wniosek

Program ten wykorzystuje macierz sąsiedztwa do reprezentowania grafu oraz wypisywania na jej podstawie potrzebnych elementów jak w tym przypadku wierzchołków, sąsiadów, krawędzi czy pętli. Zaletą jest różnorodność możliwości oraz typów informacji, które możemy z tego programu zdobyć, a wadą mocna nieefektywność z powodu dużej ilości pętli

4. Kod

```
#include <iostream>
#include <iomanip>

using namespace std;
void petle(int wierzcholki, int **macierz){
   int i;
   for(i = 0; i < wierzcholki; i++){
            if(macierz[i][i] == 1){ //sprawdzamy czy wierzcholek z ktorego wychodzi krawedz i do ktorego to ten sam
                  cout << "Wierzcholek  " << i << "  posiada petle" << endl;
```

```
            }
        }
}

void krawedziedwukierunkowe(int wierzcholki, int **macierz){
   int i,j;
   for(i = 0; i < wierzcholki; i++){
      for(j = 0; j < wierzcholki; j++){
         if(macierz[i][j] == 1 && macierz[j][i] == 1 && i != j){ //sprawdzamy czy wierzcholki posiadaja krawedzie w obie strony, pomijamy "petle"
            cout << "Wierzcholek  " << i << "  posiadaja krawedz dwukierunkowa" << endl;
         }
      }
   }
}

void wierzcholekizolowany(int wierzcholki, int **macierz){
   int i,j,a;
   for(i = 0; i < wierzcholki; i++){
      a = 0;
      for(j = 0; j < wierzcholki; j++){
         if(macierz[i][j] == 0 && macierz[j][i] == 0){
            a++; //jesli wierzcholek posiada same zera w macierzy, jest wierzcholkiem izolowanym
```

```
        }
      }
      if(a == wierzcholki){
        cout << "Wierzcholek  " << i << "  jest wierzcholkiem izolowanym" <<
endl;
      }
   }
}

void sadziedzi(int wierzcholki, int **macierz){
   int i,j;
   for(i = 0; i < wierzcholki; i++){
      for(j = 0; j < wierzcholki; j++){
        if(macierz[i][j] == 1 && i != j){ //sprawdzamy do jakiego wierzholka
wychodzi krawedz z sprawdzanego wierzcholka
          cout << "Sasiadem wierzcholka  " << i << "  jest wierzcholek  " << j <<
endl;
        }
      }
   }
}

void stopien(int wierzcholki, int **macierz){
   int i,j, a,b;
   for(i = 0; i < wierzcholki; i++){
```

```
    a = 0, b = 0;
    for(j = 0; j < wierzcholki; j++){ // zliczamy wszyskie krawedzie wchodzace i wychodzace z wierzcholka
        if(macierz[i][j] == 1 ){
            a++;
        }
        if(macierz[j][i] == 1){
            b++;
        }
    }
    cout << "Stopien wchodzacy wierzcholka " << i << " to:  " << b << endl;
    cout << "Stopien wychodzacy wierzcholka " << i << " to:  " << a << endl;
  }
}

void sasiedzikazdego(int wierzcholki, int **macierz){
  int i,j, a;
  for(i = 0; i < wierzcholki; i++){
    a = 0;
    for(j = 0; j < wierzcholki; j++){
        if(macierz[i][j] == 1 ){ //sprawdzamy jaki wierzcholek posiada same jedynki w macierzy
            a++;
        }
    }
```

```
      if(a == wierzcholki || a == wierzcholki-1){
         cout << "Wierzcholek " << i << " jest sasiadem kazdego wierzcholka" <<
endl;
      }
   }
}

int main( )
{
   int wierzcholki, krawedzie, **macierz;
   int a,b;
   //wprowadzanie ilosci wierzcholkow oraz krawedzi
   cout << "Podaj liczbe wierzcholkow:";
   cin >> wierzcholki;
   cout << "Podaj liczbe krawedzi:";
   cin >> krawedzie;
   //tworzenie tablicy wskaznikow
   macierz = new int *[wierzcholki];

   for(int i = 0; i < wierzcholki; i++){
      macierz[i] = new int [wierzcholki];
   }
   //tworzenie macierzy sasiedztwa wypelnionej na start zerami
   for(int i = 0; i < wierzcholki;i++){
      for(int j = 0; j< wierzcholki;j++){
```

```
      macierz[i][j] = 0;
   }
}
//wprowadzanie krawedzi
for(int i = 0; i < krawedzie; i++){
   cout << "Podaj wierzcholek z ktorego wychodzi krawedz: ";
   cin >> a;
   cout << "Podaj wierzcholek do ktorego wchodzi krawedz: ";
   cin >> b;
   macierz[a][b] = 1;
}
//wywolanie funkcji
petle(wierzcholki,macierz);
krawedziedwukierunkowe(wierzcholki,macierz);
wierzcholekizolowany(wierzcholki,macierz);
sadziedzi(wierzcholki,macierz);
stopien(wierzcholki,macierz);
sasiedzikazdego(wierzcholki,macierz);

//wypisanie macierzy
for(int i =0; i < wierzcholki; i++){
   cout << setw ( 3 ) << i;
   for( int j = 0; j < wierzcholki; j++ ) cout << setw ( 3 ) << ( int ) macierz [ i ][ j ];
   cout << endl;
```

```
    }
    //zwolnienie pamieci
    for(int i = 0; i < wierzcholki; i++){
        delete [] macierz[i];
    }
    delete [] macierz;
    return 0;
}
```