

WYDZIAŁ
**ELEKTROTECHNIKI
I INFORMATYKI**
POLITECHNIKI RZESZOWSKIEJ

Projekt #3 Algorytmy i struktury danych

Autor: Dawid Stachiewicz

Numer albumy: 173218

Temat: Graf skierowany reprezentowany
przy pomocy listy krawędzi.

Kierunek: Inżynieria i analiza danych

Rzeszów 2022

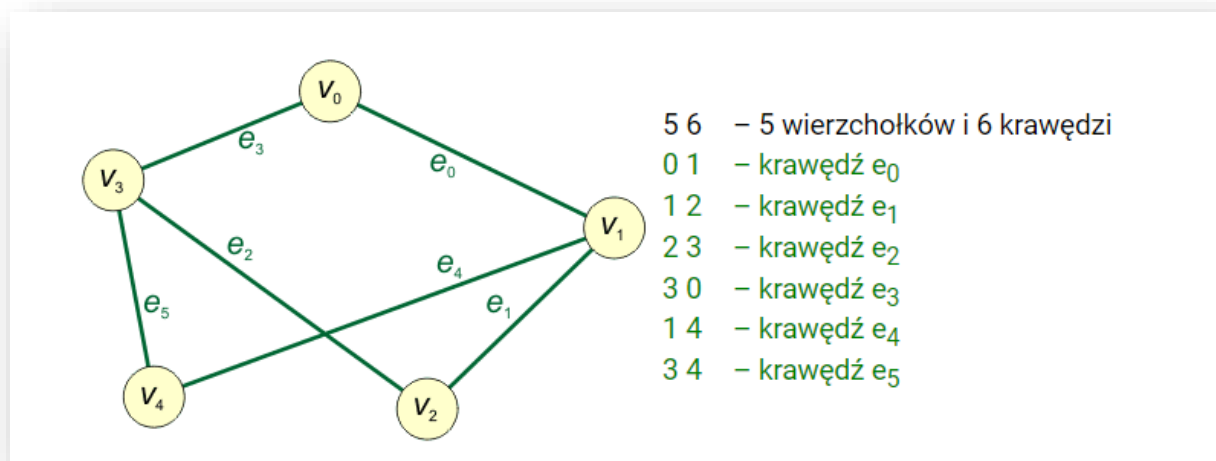
Spis treści

2.Pseudokod funkcji dodajKrawedz.....	4
3.Pseudokod funkcji wypiszSasiadow	4
4.Pseudokod funkcji wypiszSasiadowDlaSasiadow	5
5.Pseudokod funkcji wypiszStopnieWychodzace	5
6.Pseudokod funkcji wypiszStopnieWchodzace	6
7.Pseudokod funkcji wypiszIzolowane	6
8.Pseudokod funkcji wypiszPetle	7
9.Pseudokod funkcji wypiszKrawedzieDwukierunkowe.....	7
10.Kod funkcji dodajKrawedz	8
11.Kod funkcji wypiszSasiadow	8
12.Kod funkcji wypiszSasiadowDlaSasiadow	9
13.Kod funkcji wypiszStopnieWychodzace	9
14.Kod funkcji wypiszStopnieWchodzace.....	10
15.Kod funkcji wypiszIzolowane	10
16.Kod funkcji wypiszPetle.....	11
17.Kod funkcji wypiszKrawedzieDwukierunkowe	11
18. Działanie programu	12
19. O programie	13
20. Podsumowanie.....	13

1. Graf skierowany

Graf skierowany (ang. directed graph) to graf, w którym krawędzie posiadają kierunek. Oznacza to, że każda krawędź prowadzi z jednego wierzchołka do drugiego, co oznacza, że odwiedzanie wierzchołków w określonej kolejności jest ważne. W grafie skierowanym nie ma krawędzi odwrotnych, tj. odwiedzanie wierzchołków w odwrotnej kolejności niż opisana krawędzią jest niemożliwe.

W przeciwieństwie do grafu nieskierowanego (ang. undirected graph) krawędzie nie są zbiorem par niepowtarzających się wierzchołków.



2.Pseudokod funkcji dodajKrawedz

```
funkcja dodajKrawedz(graf g, int z, int dokad)
    alokuj pamięć dla nowej krawędzi
    przypisz z do pola z nowej krawędzi
    przypisz dokad do pola dokad nowej krawędzi
    przypisz następna krawędz z listy krawędzi dla wierzchołka z do następna pola nowej krawędzi
    przypisz nową krawędz do pierwszego elementu listy krawędzi dla wierzchołka z
```

Funkcja ta przyjmuje jako argument wskaźnik na strukturę Graf, która zawiera informacje o grafie, takie jak liczba wierzchołków i tablica krawędzi, oraz liczby całkowite z i dokad, które oznaczają wierzchołek początkowy i końcowy dla nowej krawędzi.

Funkcja alokuje pamięć dla nowej krawędzi, przypisuje podane z i dokad do odpowiednich pól nowej krawędzi, następnie przypisuje pierwszy element z listy krawędzi dla wierzchołka z do pola następna nowej krawędzi i przypisuje nową krawędź jako pierwszy element listy krawędzi dla wierzchołka z.

3.Pseudokod funkcji wypiszSasiadow

```
funkcja wypiszSasiadow(graf g)
    dla i od 0 do g.liczbaWierzchołkow
        wypisz "Sąsiedzi wierzchołka i: "
        pobierz listę krawędzi dla wierzchołka i
        dla k z listy krawędzi
            wypisz k.dokad
        wypisz nową linię
```

Funkcja ta przyjmuje jako argument wskaźnik na strukturę Graf, która zawiera informacje o grafie, takie jak liczba wierzchołków i tablica krawędzi. Dla każdego wierzchołka od 0 do liczby wierzchołków w grafie, funkcja wypisuje "Sąsiedzi wierzchołka i: " i pobiera listę krawędzi dla tego wierzchołka. Następnie dla każdej krawędzi w liście, funkcja wypisuje numer docelowego wierzchołka. Na końcu dla każdego wierzchołka wypisuje nową linię.

4. Pseudokod funkcji wypiszSasiadowDlaSasiadow

```
funkcja wypiszSasiadowDlaSasiadow(graf g)
  dla i od 0 do g.liczbaWierzchołkow
    wypisz "Sasiadow dla sasiadow wierzchołka i: "
    pobierz listę krawędzi dla wierzchołka i
    dla k z listy krawędzi
      pobierz listę krawędzi dla sąsiada wierzchołka i
      dla k2 z listy krawędzi
        jeśli k2.dokad nie jest równy i
          wypisz k2.dokad
```

Funkcja ta przyjmuje jako argument wskaźnik na strukturę Graf, która zawiera informacje o grafie, takie jak liczba wierzchołków i tablica krawędzi. Dla każdego wierzchołka od 0 do liczby wierzchołków w grafie, funkcja wypisuje "Sąsiedzi sąsiadów dla wierzchołka i: " i pobiera listę krawędzi dla tego wierzchołka. Następnie dla każdej krawędzi w liście, funkcja pobiera listę krawędzi dla sąsiada wierzchołka i i dla każdej krawędzi w liście sąsiada sprawdza czy k2.dokad nie jest równy i i wypisuje numer wierzchołka docelowego dla tej krawędzi. Na końcu dla każdego wierzchołka wypisuje nową linię.

5. Pseudokod funkcji wypiszStopnieWychodzace

```
funkcja wypiszStopnieWychodzace(graf g)
  dla i od 0 do g.liczbaWierzchołkow
    stopien = 0
    pobierz listę krawędzi dla wierzchołka i
    dla k z listy krawędzi
      stopien++
    wypisz "Stopień wychodzący dla wierzchołka i: stopien"
```

Funkcja ta przyjmuje jako argument wskaźnik na strukturę Graf, która zawiera informacje o grafie, takie jak liczba wierzchołków i tablica krawędzi. Dla każdego wierzchołka od 0 do liczby wierzchołków w grafie, funkcja tworzy zmienną stopien i pobiera listę krawędzi dla tego wierzchołka. Następnie dla każdej krawędzi w liście, zwiększa zmienną stopien o 1. Na końcu dla każdego wierzchołka wypisuje "Stopień wychodzący dla wierzchołka i: stopien"

6.Pseudokod funkcji wypiszStopnieWchodzace

```
funkcja wypiszStopnieWchodzace(graf g)
    utwórz tablicę stopnie o rozmiarze g.liczbaWierzchołkow
    dla i od 0 do g.liczbaWierzchołkow
        pobierz listę krawędzi dla wierzchołka i
        dla k z listy krawędzi
            stopnie[k.dokad]++
    dla i od 0 do g.liczbaWierzchołkow
        wypisz "Stopień wchodzący dla wierzchołka i: stopnie[i]"
    zwolnij pamięć z tablicy stopnie
```

Funkcja ta przyjmuje jako argument wskaźnik na strukturę Graf, która zawiera informacje o grafie, takie jak liczba wierzchołków i tablica krawędzi. Funkcja tworzy tablicę stopnie o rozmiarze równym liczbie wierzchołków w grafie, następnie dla każdego wierzchołka od 0 do liczby wierzchołków w grafie, pobiera listę krawędzi dla tego wierzchołka i dla każdej krawędzi zwiększa liczbę stopni wchodzących dla wierzchołka docelowego o 1. Następnie dla każdego wierzchołka od 0 do liczby wierzchołków w grafie wypisuje "Stopień wchodzący dla wierzchołka i: stopnie[i]" i zwalnia pamięć z tablicy stopnie.

7.Pseudokod funkcji wypiszIzolowane

```
funkcja wypiszIzolowane(graf g)
    izolowane = 0
    dla i od 0 do g.liczbaWierzchołkow
        jeśli g.krawedzie[i] == NULL
            wypisz "Wierzchołek i jest izolowany"
            izolowane++
    jeśli izolowane == 0
        wypisz "Brak izolowanych wierzchołków"
```

Funkcja ta przyjmuje jako argument wskaźnik na strukturę Graf, która zawiera informacje o grafie, takie jak liczba wierzchołków i tablica krawędzi. Dla każdego wierzchołka od 0 do liczby wierzchołków w grafie, funkcja sprawdza czy dla danego wierzchołka nie ma żadnych krawędzi i jeśli nie ma, wypisuje "Wierzchołek i jest izolowany" oraz zwiększa liczbę izolowanych wierzchołków o 1. Na końcu jeśli liczba izolowanych wierzchołków jest równa 0, funkcja wypisuje "Brak izolowanych wierzchołków".

8.Pseudokod funkcji wypiszPetle

```
funkcja wypiszPetle(graf g)
  dla i od 0 do g.liczbaWierzchołkow
    pobierz listę krawędzi dla wierzchołka i
    dla k z listy krawędzi
      jeśli k.dokad == i
        wypisz "Pętla w wierzchołku i"
        przerwij
```

Funkcja ta przyjmuje jako argument wskaźnik na strukturę Graf, która zawiera informacje o grafie, takie jak liczba wierzchołków i tablica krawędzi. Dla każdego wierzchołka od 0 do liczby wierzchołków w grafie, funkcja pobiera listę krawędzi dla danego wierzchołka i dla każdej krawędzi z listy sprawdza czy krawędź zwraca do wierzchołka z którego się wychodzi ($k.dokad == i$), jeśli tak to wypisuje "Pętla w wierzchołku i" i przerywa działanie pętli.

9.Pseudokod funkcji wypiszKrawedzieDwukierunkowe

```
funkcja wypiszKrawedzieDwukierunkowe(graf g)
  dla i od 0 do g.liczbaWierzchołkow
    pobierz listę krawędzi dla wierzchołka i
    dla k z listy krawędzi
      pobierz listę krawędzi dla wierzchołka k.dokad
      dla k2 z listy krawędzi
        jeśli k2.dokad == i
          wypisz "Krawędź dwukierunkowa między wierzchołkami i i k.dokad"
          przerwij
```

Funkcja ta przyjmuje jako argument wskaźnik na strukturę Graf, która zawiera informacje o grafie, takie jak liczba wierzchołków i tablica krawędzi. Dla każdego wierzchołka od 0 do liczby wierzchołków w grafie, funkcja pobiera listę krawędzi dla danego wierzchołka i dla każdej krawędzi z listy pobiera listę krawędzi dla wierzchołka do którego krawędź prowadzi ($k.dokad$) i dla każdej krawędzi z tej listy sprawdza czy krawędź prowadzi z powrotem do wierzchołka z którego się wychodzi ($k2.dokad == i$), jeśli tak to wypisuje "Krawędź dwukierunkowa między wierzchołkami i i k.dokad" i przerywa działanie pętli.

10. Kod funkcji dodajKrawedz

```
35 // Funkcja dodająca krawędź do grafu skierowanego
36 // g - wskaźnik na graf
37 // z - wierzchołek początkowy krawędzi
38 // dokad - wierzchołek końcowy krawędzi
39 void dodajKrawedz(struct Graf* g, int z, int dokad, bool dwukierunkowa) {
40     // Alokacja pamięci dla nowej krawędzi
41     struct Krawedz* k = (struct Krawedz*) malloc(sizeof(struct Krawedz));
42     k->z = z;
43     k->dokad = dokad;
44     // Dodanie nowej krawędzi na początek listy krawędzi dla wierzchołka z
45     k->nastepna = g->krawedzie[z];
46     g->krawedzie[z] = k;
47
48     if(dwukierunkowa){
49         // Dodanie krawędzi zwrotnej
50         struct Krawedz* k2 = (struct Krawedz*) malloc(sizeof(struct Krawedz));
51         k2->z = dokad;
52         k2->dokad = z;
53         k2->nastepna = g->krawedzie[dokad];
54         g->krawedzie[dokad] = k2;
55     }
56 }
57
```

11. Kod funkcji wypiszSasiadow

```
58 // Funkcja wypisująca wszystkich sąsiadów dla każdego wierzchołka w grafie
59 void wypiszSasiadow(struct Graf* g) {
60     for (int i = 0; i < g->liczbaWierzchołkow; i++) {
61         printf("Sąsiedzi wierzchołka %d: ", i);
62         // Pobranie listy krawędzi dla wierzchołka i
63         struct Krawedz* k = g->krawedzie[i];
64         while (k != NULL) {
65             printf("%d ", k->dokad);
66             k = k->nastepna;
67         }
68         printf("\n");
69     }
70 }
71
```


12.Kod funkcji wypiszSasiadowDlaSasiadow

```
72 // Funkcja wypisująca wszystkie wierzchołki, które są sąsiadami każdego wierzchołka w grafie
73 void wypiszSasiadowDlaSasiadow(struct Graf* g) {
74     for (int i = 0; i < g->liczbaWierzchołkow; i++) {
75         printf("Sąsiedzi sąsiadow dla wierzchołka %d: ", i);
76         // Pobranie listy krawędzi dla wierzchołka i
77         struct Krawedz* k = g->krawedzie[i];
78         while (k != NULL) {
79             // Pobranie listy krawędzi dla sąsiada wierzchołka i
80             struct Krawedz* k2 = g->krawedzie[k->dokad];
81             while (k2 != NULL) {
82                 if (k2->dokad != i) {
83                     printf("%d ", k2->dokad);
84                 }
85                 k2 = k2->nastepna;
86             }
87             k = k->nastepna;
88         }
89         printf("\n");
90     }
91 }
```

13.Kod funkcji wypiszStopnieWychodzace

```
93 // Funkcja wypisująca stopnie wychodzące dla każdego wierzchołka w grafie
94 void wypiszStopnieWychodzace(struct Graf* g) {
95     for (int i = 0; i < g->liczbaWierzchołkow; i++) {
96         int stopien = 0;
97         // Pobranie listy krawędzi dla wierzchołka i
98         struct Krawedz* k = g->krawedzie[i];
99         while (k != NULL) {
100             stopien++;
101             k = k->nastepna;
102         }
103         printf("Stopień wychodzący dla wierzchołka %d: %d\n", i, stopien);
104     }
105 }
106
```

14. Kod funkcji wypiszStopnieWchodzace

```
107 // Funkcja wypisująca stopnie wchodzące dla każdego wierzchołka w grafie
108 void wypiszStopnieWchodzace(struct Graf* g) {
109     // Tworzenie tablicy przechowującej stopnie wchodzące dla każdego wierzchołka
110     int* stopnie = (int*) calloc(g->liczbaWierzchołkow, sizeof(int));
111     for (int i = 0; i < g->liczbaWierzchołkow; i++) {
112         // Pobranie listy krawędzi dla wierzchołka i
113         struct Krawedz* k = g->krawedzie[i];
114         while (k != NULL) {
115             stopnie[k->dokad]++;
116             k = k->nastepna;
117         }
118     }
119     for (int i = 0; i < g->liczbaWierzchołkow; i++) {
120         printf("Stopien wchodzacy dla wierzchołka %d: %d\n", i, stopnie[i]);
121     }
122     free(stopnie);
123 }
124
```

15. Kod funkcji wypiszIzolowane

```
125 // Funkcja wypisująca wszystkie wierzchołki izolowane w grafie
126 void wypiszIzolowane(struct Graf* g) {
127     int izolowane = 0;
128     for (int i = 0; i < g->liczbaWierzchołkow; i++) {
129         if (g->krawedzie[i] == NULL) {
130             printf("Wierzchołek %d jest izolowany\n", i);
131             izolowane++;
132         }
133     }
134     if (izolowane == 0) {
135         printf("Brak izolowanych wierzchołkow\n");
136     }
137 }
138
```


16.Kod funkcji wypiszPetle

```
139 // Funkcja wypisująca wszystkie pętle w grafie
140 void wypiszPetle(struct Graf* g) {
141     for (int i = 0; i < g->liczbaWierzchołkow; i++) {
142         struct Krawedz* k = g->krawedzie[i];
143         while (k != NULL) {
144             if (k->dokad == i) {
145                 printf("Petla w wierzchołku %d\n", i);
146                 break;
147             }
148             k = k->nastepna;
149         }
150     }
151 }
152
```

17.Kod funkcji wypiszKrawedzieDwukierunkowe

```
152 // Funkcja wypisująca wszystkie krawędzie dwukierunkowe w grafie
153 void wypiszKrawedzieDwukierunkowe(struct Graf* g) {
154     for (int i = 0; i < g->liczbaWierzchołkow; i++) {
155         struct Krawedz* k = g->krawedzie[i];
156         while (k != NULL) {
157             struct Krawedz* k2 = g->krawedzie[k->dokad];
158             while (k2 != NULL) {
159                 if (k2->dokad == i) {
160                     printf("Krawedz dwukierunkowa między wierzchołkami %d i %d\n", i, k->dokad);
161                     break;
162                 }
163                 k2 = k2->nastepna;
164             }
165             k = k->nastepna;
166         }
167     }
168 }
169
```

18. Działanie programu

 C:\Users\dawid\OneDrive\Dokumenty\Projekty\Graf_skierowany.exe

```
Sasiedzi wierzcholka 0: 2 1
Sasiedzi wierzcholka 1: 2 0
Sasiedzi wierzcholka 2: 3
Sasiedzi wierzcholka 3: 4 3 3
Sasiedzi wierzcholka 4:
Sasiedzi wierzcholka 5:

Sasiedzi sasiadow dla wierzcholka 0: 3 2
Sasiedzi sasiadow dla wierzcholka 1: 3 2
Sasiedzi sasiadow dla wierzcholka 2: 4 3 3
Sasiedzi sasiadow dla wierzcholka 3: 4 4
Sasiedzi sasiadow dla wierzcholka 4:
Sasiedzi sasiadow dla wierzcholka 5:

Stopien wychodzacy dla wierzcholka 0: 2
Stopien wychodzacy dla wierzcholka 1: 2
Stopien wychodzacy dla wierzcholka 2: 1
Stopien wychodzacy dla wierzcholka 3: 3
Stopien wychodzacy dla wierzcholka 4: 0
Stopien wychodzacy dla wierzcholka 5: 0

Stopien wchodzacy dla wierzcholka 0: 1
Stopien wchodzacy dla wierzcholka 1: 1
Stopien wchodzacy dla wierzcholka 2: 2
Stopien wchodzacy dla wierzcholka 3: 3
Stopien wchodzacy dla wierzcholka 4: 1
Stopien wchodzacy dla wierzcholka 5: 0

Wierzcholek 4 jest izolowany
Wierzcholek 5 jest izolowany

Petla w wierzcholku 3

Krawedz dwukierunkowa miedzy wierzchołkami 0 i 1
Krawedz dwukierunkowa miedzy wierzchołkami 1 i 0
Krawedz dwukierunkowa miedzy wierzchołkami 3 i 3
Krawedz dwukierunkowa miedzy wierzchołkami 3 i 3

-----
Process exited after 0.04461 seconds with return value 0
Press any key to continue . . .
```

19. O programie

Ten program reprezentuje implementację grafu skierowanego w języku C. Zawiera on funkcje do tworzenia grafu, dodawania krawędzi oraz wypisywania sąsiadów dla każdego wierzchołka. Graf jest reprezentowany przez strukturę zawierającą liczbę wierzchołków oraz tablicę list krawędzi dla każdego wierzchołka. Krawędzie są reprezentowane przez strukturę zawierającą informację o wierzchołku początkowym i końcowym oraz wskaźnik na następną krawędź wychodzącą z tego samego wierzchołka. Program umożliwia także tworzenie krawędzi dwukierunkowych.

20. Podsumowanie

Ten program reprezentuje graf skierowany przy użyciu list sąsiedztwa. Zaletą tego podejścia jest efektywne wykorzystanie pamięci, szczególnie w przypadku grafów o dużej liczbie krawędzi. Dodawanie krawędzi jest także proste i szybkie. Wada to mniejsza wydajność przy szukaniu krawędzi między dwoma wierzchołkami w porównaniu do macierzy sąsiedztwa. Inna zaleta tego programu jest to, że umożliwia łatwe dodawanie krawędzi dwukierunkowych. Możliwe jest również łatwe iterowanie po wszystkich krawędziach wychodzących z danego wierzchołka, co może być przydatne w przypadku wykonywania różnych algorytmów grafowych.