logo

Szkoła
Wydział
Katedra

Sprawozdanie

Metody Programowania

Rok akademicki x
Semestr x
Autor: x
Kierunek: x
Grupa x
Laboratorium 6
Data zajęć: x
Temat: Generacja grafów R-MAT o zadanych własnościach
Prowadzący: x

Ocena

Spis treści

1) Wprowadzenie teoretyczne

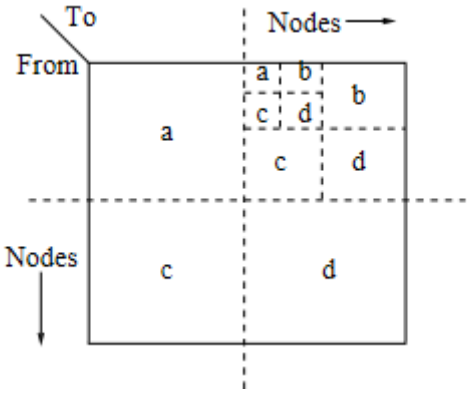
Graf – jest to struktura matematyczna służąca do przedstawiania i badania relacji między obiektami. Graf przedstawia się jako G = (V, E), gdzie V jest zbiorem wierzchołków, E – zbiór krawędzi. Grafy mogą być reprezentowane w postaci graficznej, macierzy sąsiedztwa, macierzy incydencji. Wyróżnia się grafy nieskierowane (macierz sąsiedztwa jest symetryczna) oraz skierowane.
W sprawozdaniu przyjmę, że wierzchołek nie sąsiaduje z samym sobą (na przekątnej głównej macierzy sąsiedztwa są zera).

Graf R-MAT (recursive matrix) – losowy graf otrzymany poprzez rekursywne operacje na macierzy sąsiedztwa. . Na jego wygląd wpływają podawane przez użytkownika parametry: prawdopodobieństwo wyboru danej krawędzi oraz gęstość.

Rozmiar grafu R-MAT wynosi 2^n wierzchołków, gdzie $n = \log_2 N$, N = ilość wierzchołków w prawdziwym grafie. Dzięki temu możliwy jest podział macierzy na coraz mniejsze równe ćwiartki.

Prawdopodobieństwa wyboru danego obszaru (ćwiartki) a,b,c,d muszą spełniać równość a+b+c+d=1.

Gęstość grafu wyraża się stosunkiem ilości aktualnych krawędzi do ilości wszystkich możliwych krawędzi grafu.



Przykładowy R-MAT

2) Problemy do rozwiązania

Należy napisać program generujący graf R-MAT, użytkownik powinien mieć możliwość wpisania parametrów danego rodzaju grafu, zdecydowania czy wynik zapisać do pliku. Należy sprawdzić jego poprawność oraz podać przykładowe wejścia/wyjścia.

Dodatkowo poprzez wykonanie wykresów / tabel zaprezentuję wyniki dla przykładowych danych oraz zbadam wpływ parametrów na czas wykonania programu.

W przypadku wykonywania obliczeń oraz przyrównywania w warunkach zmiennych typu *double* należy uważać na dokładność obliczeń, np. jeśli a=0.15, b=0.15, c=0.35, d=0.35 wykonanie warunku if(a+b+c+d == 1) należy zastąpić if(abs(a+b+c+d – 1) < epsilon), gdzie epsilon oznacza dokładność, abs() – wartość bezwzględna.
Innym rozwiązaniem jest przyjęcie skali prawdopodobieństwa 0 – 100. Dzięki temu uniknie się operowania na zmiennych *double*.
W celu bardziej czytelniejszego wyświetlenia macierzy sąsiedztwa można dodać ozdobniki oznaczające linie pionowe | , poziome -.

3) Program generujący graf R-MAT

a) Pseudokod

wejście: a,b,c,d – prawdopodobieństwo wyboru danej ćwiartki

N – ilość wierzchołków (rozmiar prawdziwej macierzy)

density – gęstość grafu isDirected–rodzaj grafu (0-nieskierowany 1-skierowanyy)

wyjście: tablica A – utworzona macierz sąsiedztwa

Opis działania:

Generacja polega na losowaniu ćwiartki aż zostanie jedna komórka do której wpisywana jest jedynka (jeśli nie leży na głównej przekątnej macierzy oraz wylosowana komórka nie jest już zajęta.)

```
funkcja GenerateRMAT
{
n = pow(2, log2(N));  //rozmiar grafu R-MAT
totalEdges = 0;          //ilość wszystkich możliwych krawędzi
filledEdges = 0;         //ilość 'zajętych' krawędzi
currentDensity = 0;
for(int i = 1; i <= n; i++)       //wyzerowanie macierzy sąsiedztwa oraz policzenie krawędzi
{
for(int j = 1; j <= n; j++)
{
A[i][j] = 0;
totalEdges++;
}
}//koniec for
while(1)
{
filledEdges = 0;
currentDensity = 0;
Generator(a,b,c,d,n, 1, n, 1, n, A, isDirected);          //generuje 1 losową krawędź
for(int i = 1; i <= n; i++)       // obliczenie ilości zapełnionych komórek (ilość utworzonych
```

```
        for(int j = 1; j <= n; j++)                                  //krawędzi)
                if(A[i][j] == 1)
                        filledEdges++;
currentDensity  = filledEdges / totalEdges;  //obliczenie aktualnej gęstości grafu
if(currentDensity >= density)          //jeśli osiągnięto żądaną gęstość to zakończ
        break;
}//koniec while
for(int i = 1; i <= n; i++)                //wypisanie grafu
        for(int j = 1; j <= n; j++)
        {
                std::cout << A[i][j] << ' ';
        std::cout <<std::endl;
        }
for(int i = 1; i <= n; i++)       //obliczenie stopni wierzchołków
        for(int j = 1; j <= n; j++)
                if(A[i][j] == 1)
                        degree[i]++;
mean = 0
minDegree = 10000000000;
maxDegree = 0;
for(int i = 1; i <= n; i++)       //obliczenie średniego stopnia wierzchołków oraz znalezienie
{                                    //największego oraz najmniejszego stopnia
mean += degree[i];
if(degree[i] >= maxDegree)
        maxDegree = degree[i];
if(degree[i] <= minDegree)
        minDegree= degree[i];
}
mean = mean / n;
printf("Srednia wartosc stopnia: %f, minimalny stopnien: %i, maksymalny stopnien: %i \n",
mean, minDegree, maxDegree);
```

```
}//koniec funkcji

//funkcja losuje 1 krawędź
funkcja Generator
{
if(n == 1 && firstX == lastX && firstY == lastY)  //jeśli zostanie 1 komórka
{
if(firstX == firstY)  //sprawdzenie czy wybrana komórka leży na głównej przekątnej macierzy
{
A[firstX][firstY] = 0;
return;
}
if(A[firstX][firstY] == 1)      //sprawdzenie czy krawędź już istnieje
        return;
else
{
if(isDirected)
        A[firstX][firstY] = 1;
else            //jeśli nieskierowany to daj 1 również na symetryczne współrzędne
{
A[firstX][firstY] = 1;
A[firstY][firstX] = 1;
}//koniec else
}//koniec else
return;                 //zakoncz funkcje -> krawędź została wylosowana
}//koniec if

//wylosowanie obszaru i wykonanie odpowiedniego wywołania rekursywnego
```

```
//podziały obszarów pradopodobieństwa
poczatekA = 1;
koniecA = a;
poczatekB = koniecA + 1;
koniecB = koniecA + b;
poczatekC = koniecB + 1;
koniecC = koniecB + c;
poczatekD = koniecC + 1;
koniecD = koniecC + d;
if(a+b+c+d != 100)              //walidacja
{
std::cerr << "ERROR a+b+c+d != 100\n";
exit(0);
}
randInt = rand() % 100 + 1;  //losowanie od 1  do 100
p = randInt
//wybierz obszar
if(p >= poczatekA && p <= koniecA)
        Generator(a,b,c,d,n/2,firstX, (firstX + lastX)/2, firstY, (firstY+lastY)/2, A,
isDirected);
else if(p >= poczatekB && p <= koniecB)
    Generator(a,b,c,d,n/2, (firstX + lastX)/2 + 1, lastX, firstY, (firstY+lastY)/2, A,
isDirected);
else if(p >= poczatekC && p <= koniecC)
    Generator(a,b,c,d,n/2, firstX, (firstX + lastX)/2, (firstY+lastY)/2 + 1, lastY, A,
isDirected);
else if(p >= poczatekD && p <= koniecD)
    Generator(a,b,c,d,n/2, (firstX + lastX)/2 + 1, lastX, (firstY+lastY)/2 + 1, lastY, A,
isDirected);
else
    std::cerr << “error if\n”;
}/koniec funkcji
```

b) C++

Program został napisany w języku C++, obok linijek z kodem (lub nad nimi) po sekwencji „//” znajdują się komentarze do danego fragmentu.

Program został skompilowany oraz uruchomiony na sprzęcie/oprogramowaniu:

procesor CPU: Intel i5-4210H

system operacyjny: Windows 10 Home wersja: 10.0.18363

kompilator: g++ wersja 9.2.0

Kod źródłowy znajduje się w pliku main.cpp

c) Przykładowe wejścia / wyjścia z programu

1.

Wejście

a = 30; b=50; c=15; d=5

N = 15

density 0.5

isDirected = 1

Wyjście

0 1 1 1 | 1 1 1 1

1 0 1 1 | 1 1 1 1

1 1 0 1 | 0 1 0 0

1 0 0 0 | 1 0 1 0

- - - - - - - -

1 0 1 1 | 0 0 1 0

0 1 1 0 | 0 0 0 1

0 1 0 1 | 1 0 0 0

0 0 0 0 | 1 0 0 0

Ilość jedynek w poszczególnych ćwiartkach

A: 10 B: 11 C: 7 D 4

wierzcholek nr 1 ma stopnien: 7

wierzcholek nr 2 ma stopnien: 7

wierzcholek nr 3 ma stopnien: 4

wierzcholek nr 4 ma stopnien: 3

wierzcholek nr 5 ma stopnien: 4

wierzcholek nr 6 ma stopnien: 3

wierzcholek nr 7 ma stopnien: 3

wierzcholek nr 8 ma stopnien: 1

Srednia wartosc stopnia: 4.000000, minimalny stopnien: 1, maksymalny stopnien: 7

2.

Wejście

a = 5; b=20; c=15; d=60

N = 15

density 0.5

isDirected = 0

Wyjście

0 0 0 0 | 0 0 1 0

0 0 0 1 | 0 0 1 1

0 0 0 0 | 0 0 1 1

0 1 0 0 | 1 1 1 1

- - - - - - - -

0 0 0 1 | 0 1 1 1

0 0 0 1 | 1 0 1 1

1 1 1 1 | 1 1 0 1

0 1 1 1 | 1 1 1 0

Ilość jedynek w poszczególnych ćwiartkach

A: 2 B: 9 C: 9 D 12

wierzcholek nr 1 ma stopnien: 1

wierzcholek nr 2 ma stopnien: 3

wierzcholek nr 3 ma stopnien: 2

wierzcholek nr 4 ma stopnien: 5

wierzcholek nr 5 ma stopnien: 4

wierzcholek nr 6 ma stopnien: 4

wierzcholek nr 7 ma stopnien: 7

wierzcholek nr 8 ma stopnien: 6

Srednia wartosc stopnia: 4.000000, minimalny stopnien: 1, maksymalny stopnien: 7

d) Testowanie poprawności

W celu sprawdzenia poprawności wypisze otrzymaną macierz sąsiedztwa, policzę ile jest krawędzi w danych ćwiartkach, następnie sprawdzę czy te ilości zgadzają się w przybliżeniu z wprowadzonymi prawdopodobieństwami.

Rozpocznę od testowania poprawności grafów nieskierowanych

Test I

Wejście:

```
a = 29
b = 1
c = 50
d = 20
N = 17
density = 0.3
isDirected = 0
```

Wyjście z programu

Macierz sąsiedztwa

0 1 1 1 1 0 1 0 | 1 0 1 1 1 1 1 1

1 0 0 0 0 0 0 0 | 0 1 0 0 0 1 0 1

1 0 0 0 0 0 1 1 | 0 0 1 0 0 0 1 1

1 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 1

1 0 0 0 0 1 1 1 | 0 0 0 0 1 1 1 1

0 0 0 0 1 0 0 0 | 0 0 0 0 0 0 1 1

1 0 1 0 1 0 0 0 | 0 0 0 0 0 0 0 0

0 0 1 0 1 0 0 0 | 0 0 0 0 0 0 0 0

- - - - - - - - - - - - - - - - -

1 0 0 0 0 0 0 0 | 0 0 1 0 0 1 1 1

0 1 0 0 0 0 0 0 | 0 0 0 0 0 1 0 0

1 0 1 0 0 0 0 0 | 1 0 0 0 0 0 0 1

1 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 1

1 0 0 0 1 0 0 0 | 0 0 0 0 0 0 0 1

1 1 0 0 1 0 0 0 | 1 1 0 0 0 0 0 0

1 0 1 0 1 1 0 0 | 1 0 0 0 0 0 0 1

1 1 1 1 1 1 0 0 | 1 0 1 1 1 0 1 0

Ilość jedynek w poszczególnych ćwiartkach

A: 20 B: 20 C: 20 D 18

Ocena poprawności: prawdopodobieństwa nie zostały spełnione.

Test II

Wejście

```
a = 20
b = 60
c = 10
d = 10
N = 38
```

```
density = 0.3
isDirected = 0
```

Wyjście:

Macierz sąsiedztwa

```
0 0 0 1 0 1 1 1 1 1 0 1 0 1 1 1 |1 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1
0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 |0 0 0 0 1 0 1 0 0 0 1 1 1 1 1 1
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 |0 1 0 0 0 1 0 1 0 0 1 0 0 1 1 1
1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 |0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 1
0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 |1 1 0 0 0 1 1 0 0 1 1 1 0 0 1 1
1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 |0 1 0 1 0 0 0 1 1 0 1 1 1 0 1 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1
1 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 |1 0 0 0 1 0 0 0 1 0 0 1 0 0 1 0
1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 |1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 |1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 1
0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 |1 0 0 0 1 1 1 1 0 0 1 0 0 0 0 1
1 0 0 0 1 0 0 0 0 1 0 0 1 0 1 0 |1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 |0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 |0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0
1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 |1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1
1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 |1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1 0 0 0 1 0 0 1 1 1 1 1 0 0 1 1 |0 0 0 1 0 0 1 1 0 0 0 1 1 1 1 1
0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 1 |0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0
0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 |0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1
1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 |1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0
1 1 0 0 0 0 1 1 0 0 1 1 0 1 0 0 |0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 1 0 1 0 0 0 1 1 1 0 0 1 0 0 |0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 1 1 0 0 0 1 1 1 0 0 0 0 0 |1 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0
1 0 1 0 0 1 0 0 1 0 1 0 1 0 0 0 |1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
1 0 0 1 0 1 1 1 1 0 0 0 0 0 1 0 |0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0
1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 |0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
```

1 1 1 0 1 1 0 0 1 0 1 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

1 1 0 0 1 1 0 1 1 0 0 0 0 0 0 0 | 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0

0 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 | 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 | 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

1 1 1 0 1 1 0 1 1 0 0 0 0 0 0 0 | 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0

1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 0 | 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0

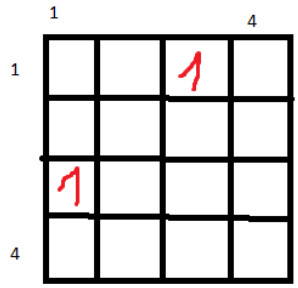
Ilość jedynek w poszczególnych ćwiartkach

A: 62 B: 101 C: 101 D 44

Ocena poprawności: prawdopodobieństwa nie zostały spełnione.

Wniosek do powyższych testów:

W grafie nieskierowanym mniej zgadza się rozkład jedynek bo jego macierz sąsiedztwa musi zachować symetrie.(np. wpisując krawędź w [1][3] należący do ćwiartki c, należy też wpisać w [3][1], które może należeć np. do ćwiartki b). Przykład:



Testowanie poprawności grafów skierowanych

Test I

Wejście

```
a = 25
b = 25
c = 25
d = 25
N =17
density = 0.4
isDirected = 1
```

Wyjście

Macierz sąsiedztwa

0 0 0 0 1 1 0 0 | 0 0 0 1 0 1 0 0

1 0 0 0 0 1 1 1 | 1 1 0 0 0 1 1 0

0 1 0 0 1 1 0 1 | 0 1 1 1 0 0 1 0

1 1 0 0 1 0 0 1 | 0 0 1 0 0 1 1 1

1 0 1 0 0 1 0 0 | 1 0 0 0 1 1 1 1

0 0 1 0 0 0 0 1 | 1 0 1 1 1 1 0 1

0 0 1 1 1 0 0 0 | 1 0 0 0 0 0 0 0

1 0 1 0 0 1 1 0 | 0 1 1 0 1 0 0 0

- - - - - - - - - - - - - - - - -

0 0 1 1 0 0 1 1 | 0 1 0 0 0 0 1 1

0 1 0 0 0 0 0 0 | 0 0 0 1 1 0 1 0

1 0 1 0 0 0 1 0 | 1 0 0 0 1 1 1 0

1 0 0 0 1 0 1 1 | 0 1 0 0 0 0 0 1

0 0 1 1 1 0 0 1 | 0 1 1 0 0 0 0 0

0 0 1 1 1 1 1 1 | 1 0 1 0 0 0 0 1

1 0 0 1 0 1 0 0 | 0 0 0 0 0 1 0 0

0 1 0 1 1 0 1 0 | 0 0 1 0 0 0 0 0

Ilość jedynek w poszczególnych ćwiartkach

A: 26 B: 29 C: 29 D 19

Ocena poprawności: biorąc pod uwagę losowość działa poprawnie

Test II

Wejście

```
a = 50
b = 5
c = 15
d = 30
N =38
```

```
density = 0.3
isDirected = 1
```

Wyjście

Macierz sąsiedztwa

```
0 1 1 0 1 0 1 0 1 0 0 0 0 0 | 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 1 0 1 1 0 0 1 0 0 0 1 1 0 | 1 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0
1 1 0 1 1 0 0 1 0 0 1 1 0 0 0 0 | 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0
1 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 | 1 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0
1 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 | 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 1 0 0 1 0 1 0 0 1 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 1 1 0 0 0 0 1 0 0 0 0 0 | 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0
0 1 1 1 1 1 1 0 0 0 0 0 0 1 0 1 | 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0
1 1 0 0 1 0 0 0 0 1 1 0 1 0 0 0 | 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
1 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 | 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
1 1 1 0 0 0 0 0 1 0 0 1 0 0 0 0 | 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 0 0 0 0 1 1 1 0 0 0 0 0 | 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
1 0 0 0 1 1 1 1 1 1 0 0 0 0 1 0 | 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0
0 1 0 1 1 1 0 0 1 0 0 1 1 0 0 1 | 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
1 0 1 1 1 0 1 1 1 0 1 0 1 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 0 1 1 1 1 1 0 1 1 1 1 0 | 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 | 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0
1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 | 1 0 1 1 0 0 0 0 1 1 0 0 1 0 0 0
1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 | 1 0 0 1 0 0 1 0 1 0 1 1 0 0 0 0
1 0 1 1 0 0 1 0 0 0 1 0 0 0 0 0 | 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0
1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 | 1 1 1 0 0 1 0 0 0 0 0 0 1 1 0 0
0 1 0 0 1 1 0 1 0 0 1 0 0 0 0 0 | 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0
0 0 1 0 1 0 1 1 0 0 0 1 0 0 1 0 | 1 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0
0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 1 | 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0
1 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 | 1 1 1 0 1 0 0 0 0 0 1 0 0 0 0 0
```

0 1 1 0 0 0 0 0 1 1 0 0 1 0 0 0 | 1 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0

1 1 1 0 0 0 0 0 1 0 1 1 0 0 0 0 | 1 1 1 1 0 0 0 1 1 0 0 1 1 0 1 0

1 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 | 1 0 1 1 0 0 0 0 1 1 1 0 0 0 0 1

1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 0 | 1 0 1 0 1 1 1 0 1 1 1 0 0 1 0 0

0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 | 1 0 0 0 1 1 0 0 1 1 0 1 1 0 0 0

0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 | 0 0 0 0 1 0 1 1 0 0 1 0 1 1 0 0

1 0 0 1 0 0 1 0 0 0 0 0 0 1 0 1 | 0 1 0 1 0 1 1 1 0 0 1 1 0 0 1 0

Ilość jedynek w poszczególnych ćwiartkach

A: 108 B: 32 C: 74 D 94

Ocena poprawności: działa poprawnie

4) Tabele i wykres

W celu stworzenia zestawienia uruchomię program z parametrami umieszczonymi w tabeli 1. Następnie wpiszę otrzymane wyniki do tabel 2 i 3. Przy pomocy programu Excel na podstawie otrzymanych z programu danych dotyczących rozkładu stopni wierzchołków stworzę wykres przedstawiający zależność stopnia wierzchołka od jego częstotliwości występowania dla grafów utworzonych przy pomocy parametrów z tabeli 1.

Tabela 1: Parametry losowanych grafów

graf	a	b	c	d	N	density	isDirected
1	25	25	25	25	68	0.3	1
2	45	15	15	25	68	0.3	1
3	55	15	15	15	68	0.3	1

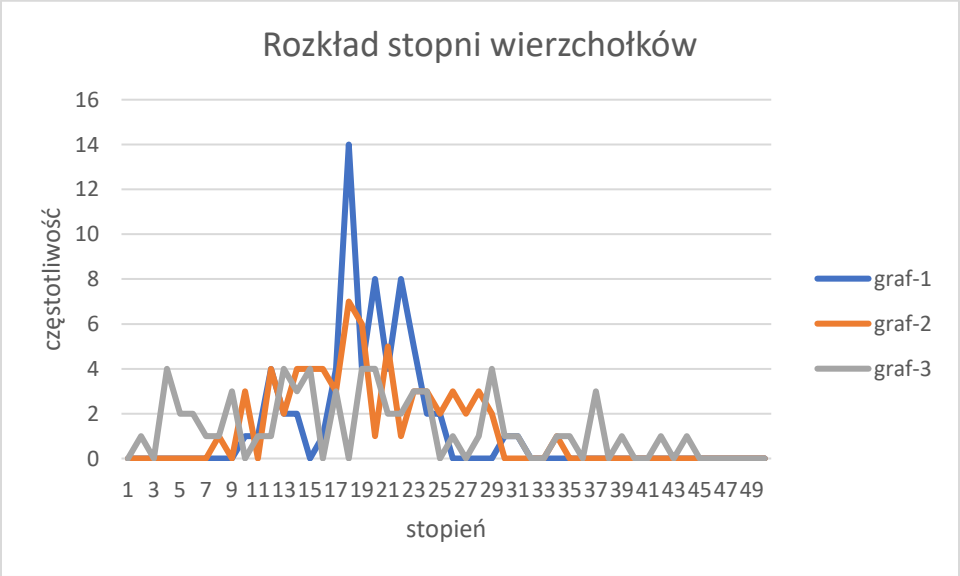
Tabela 2: Właściwości otrzymanych grafów

graf	ilość wierzchołków	ilość krawędzi	średni stopień	min. stopnień	maks. stopnień	czas generowania
1	64	1229	19,2	10	31	203s
2	64	1229	19,20	8	34	163s
3	64	1229	19,2	2	44	221s

Tabela 3: Ilość krawędzi w poszczególnych ćwiartkach

graf	a	b	c	d
1	291	311	318	309
2	431	252	243	303
3	477	261	265	226

Wykres przedstawiający rozkład stopni wierzchołków



Wnioski do zestawienia

Grafy o tych samych N oraz gęstości mają tą samą wartość średniego stopnia wierzchołków, natomiast różnią się minimalne i maksymalne wartości stopni. Na czas generowania grafów wpływa wielkość grafu oraz jego gęstość. Ilość krawędzi w poszczególnych ćwiartkach jest proporcjonalna do ustalonych prawdopodobieństw.

5) Wnioski

Program działa poprawnie.
Program obsługuje zarówno generowanie grafów skierowanych jak i nieskierowanych.
Im większa ma być gęstość oraz wielkość grafu tym program wykonuje się dłużej, bo musi wylosować i wpisać więcej jedynek w macierz sąsiedztwa grafu.
Zgodnie z założeniem, że wierzchołek nie może sąsiadować z samym sobą główna przekątna macierzy sąsiedztwa pozostaje wypełniona zerami.
W celu reprezentowania prawdopodobieństwa została użyta skala 0 – 100, dzięki temu nie występują problemy z niedokładnością zmiennych *double*.

W grafach nieskierowanym rozkład krawędzi nie jest tak dokładny jak w grafach skierowanych, bo w nieskierowanych macierz sąsiedztwa musi zachować symetrię.

6) Literatura

[1] https://pl.wikipedia.org/wiki/Graf_(matematyka)

[2] https://en.wikipedia.org/wiki/Directed_graph

(dostęp do powyższych: 13.05.2021)

[3] R-MAT: A Recursive Model for Graph Mining – Deepayan Chakrabarti, Yiping Zhan, Christos Faloutsos

[4] Algorytmy bez tajemnic – Thomas H. Cormen (str. 83 – 87, 90 - 92)