Projektowanie i Eksploatacja Sieci Komputerowych
Rozkład stopni wierzchołków w grafach

Jako podstawę do tego ćwiczenia wykorzystaj albo program z poprzednich zajęć, albo kod przedstawiony poniżej. Zwróć szczególną uwagę na funkcję addRandomByProb.

```
Import random
class graf:
    g = {}
    vertexSteps = {}
    sortedSteps = {}
    allSteps = 0
    def __init__(self):
        self.g = {}

    def addVertex(self, newVertexNumber):
        if newVertexNumber not in self.g:
            self.g[newVertexNumber] = []

    def addRandom(self):
        vertexCount = len(self.g)
        newVertexCount = vertexCount+1
        randVertex = random.randint(1,vertexCount)
        self.addVertex(self, newVertexCount)
        self.g[newVertexCount].append(randVertex)
        self.g[randVertex].append(newVertexCount)

    def countSteps(self):
        for i in self.g:
            self.vertexSteps[i] = len(self.g[i])
            self.allSteps = self.allSteps + len(self.g[i])

    def createProbTable(self):
        tmp = sorted(self.vertexSteps.items(), key = lambda kv:(kv[1], kv[0]))
        for x in tmp:
            self.sortedSteps[x[0]] = x[1]
        print(self.sortedSteps)

    def addRandomByProb(self):
        vertexCount = len(self.g)
        newVertexCount = vertexCount+1
        self.countSteps(self)
        self.sortedSteps = {}
        self.createProbTable(self)
        randNum = random.randint(0,self.allSteps -1)
        selected = -1
        for i in self.sortedSteps:
            if randNum <= self.sortedSteps[i]:
                selected = i
                break
            else:
                randNum = randNum - self.sortedSteps[i]
```

```
        if selected == -1:
            selected = len(self.sortedSteps)
        self.addVertex(self, newVertexCount)
        self.g[newVertexCount].append(selected)
        self.g[selected].append(newVertexCount)

    def printGraph(self):
        print(self.g)
```

Funkcja addRandomByProb dodaje nowy wierzchołek i dołącza go z większym prawdopodobieństwem do wierzchołka o wyższym stopniu. Jeśli korzystasz z programu z poprzedniego ćwiczenia, upewnij się, że Twój program posiada podobną funkcję i ich działanie jest zbieżne.

1. Dodaj do programu funkcję generateHistogram, która w każdej linii wyjścia wydrukuje stopien_wierzcholka; ilosc_wierzcholkow_o_tym_stopniu
2. Wykorzystując funkcję addRandom oraz addRandomByProb (lub ich odpowiedniki z poprzedniego ćwiczenia), wygeneruj grafy o 100 wierzchołkach.
3. Używając funkcji generateHistogram wypisz do plików tekstowych rozkłady stopni wierzchołków wygenerowanych grafów (jeden dla addRandom i jeden dla addRandomByProb).
4. Zaimportuj wygenerowane pliki do arkusza kalkulacyjnego
5. Posortuj dane po drugiej z kolumn (z ilością wierzchołków o danym stopniu) od najwyższej do najniższej.
6. Do sprawozdania załącz wykres, na którym umieścisz tylko dane z drugiej kolumny. Wykres powinien zawierać dwie serie danych, jedną z funkcji addRandom a drugą z addRandomByProb
7. W sprawozdaniu opisz, czym różnią się wykresy dla obu serii danych. Jakimi liniami trendu można je przybliżyć? Podaj równania tych linii wraz ze współczynnikiem dopasowania R^2, w razie konieczności rozdziel wykres na dwa osobne i dopasuj linię niezależnie dla każdego
8. Powtórz procedurę z punktów 2-7 dla 200 i 400 wierzchołków. Czy przy wyższych ilościach wierzchołków różnice w kształcie wykresów są mniej czy bardziej widoczne?
9. Wygeneruj dla 100, 200 i 400 wierzchołków i obu rodzajów generacji grafów reprezentacje w formacie DOT, a do sprawozdania załącz wizualizację wygenerowanych grafów. Spróbuj dobrać metodę wizualizacji rozmieszczającą wierzchołki w możliwie najbardziej czytelny sposób. Jakie zauważasz różnice pomiędzy poszczególnymi grafami? Czy ilość wierzchołków wzmacnia je czy osłabia? Odpowiedzi na te pytania, wraz z wizualizacjami, umieść w sprawozdaniu.