Kajetan Dreliszak
289050

TASS projekt I

Temat 8: „Powiązania pomiędzy użytkownikami portalu Instagram”

Zadanie 1

Dane do budowy grafu zostały udostępnione w formacie CSV. W celu wczytania grafu do programu Python skorzystano z poniższej instrukcji:

G = nx.read_edgelist('instagraph.csv', delimiter=',', nodetype=int, encoding="utf-8")

Umożliwia ona wczytanie danych z pliku o dowolnym rozszerzeniu, dzięki podaniu informacji przez programistę informacji na temat znaków rozdzielających konkretne wartości w pliku. Wczytanie danych do grafu w powyższy sposób automatycznie określa pewne jego cechy, a mianowicie: jest on traktowany jako graf nieskierowany oraz usuwane są wszystkie powtarzające się krawędzie.

Aby wczytać posiadane dane do Pajeka najpierw wczytałem je do Pythona a następnie zapisałem już utworzony graf w formacie Pajeka.

Zadanie 2

Do wyznaczenia składowych spójnych oraz ich cech wypisanych w zadaniu posłużyłem się następującymi instrukcjami w Pythonie:

```
print("Liczba komponentow w grafie: %d" % len(list(nx.connected_components(G))))
print("Czy graf jest spojny? %s" % nx.is_connected(G))
print("Liczba wierzcholkow w grafie: %d" % len(nx.nodes(G)))
print("Liczba krawedzi w grafie: %d" % len(nx.edges(G)))
print("Stopnie wierzcholkow w grafie: %s" % sorted(nx.degree(G).values(), reverse=True))
```

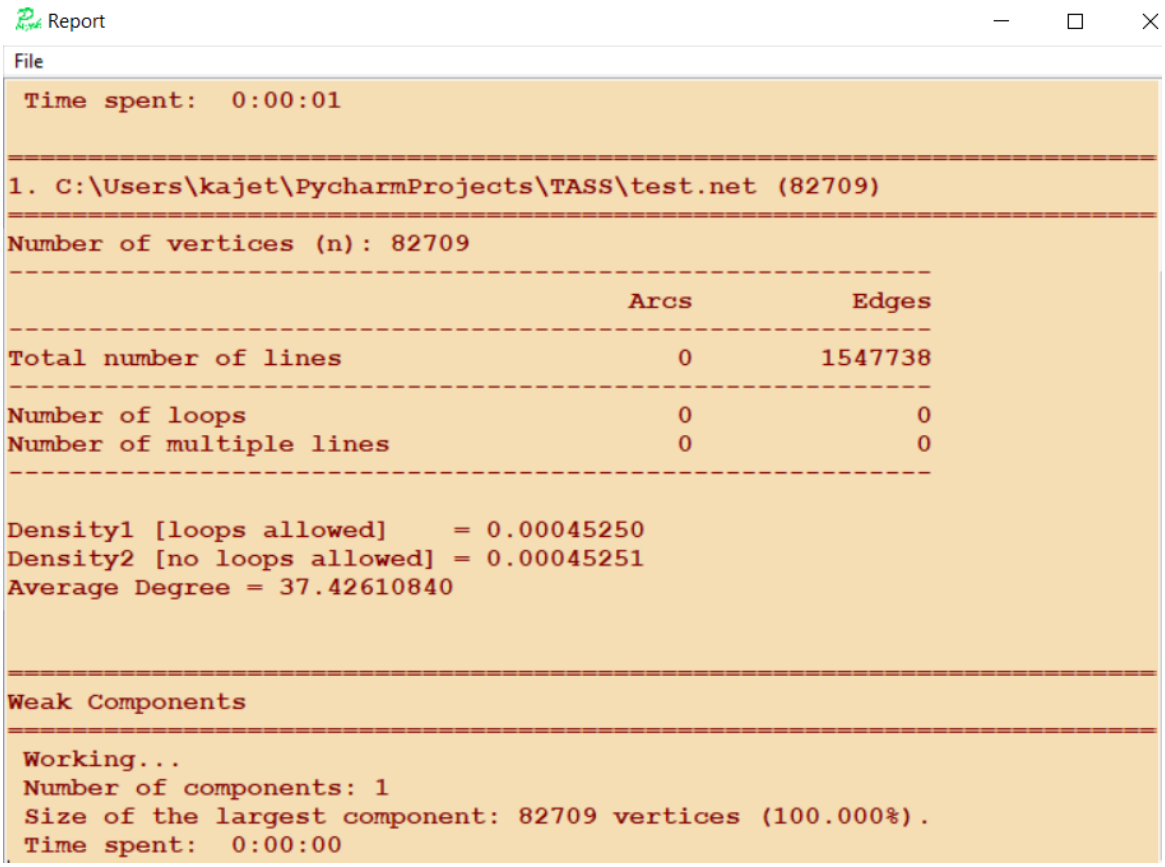
Wyniki:

Liczba komponentow w grafie: 1
Czy graf jest spojny? True
Liczba wierzcholkow w grafie: 82709
Liczba krawedzi w grafie: 1547738
Stopnie wierzcholkow w grafie: [2218, 1300, 375, 355, 329, 324, 320, …]

Analizując otrzymane dane okazało się, że graf składa się tylko z jednego komponentu, co oznaczałoby, że jest on spójny. Przypuszczenie to potwierdziła wartość otrzymana po skorzystaniu z komendy *is.connected()*.

W celu uzyskania w Pajeku odpowiedzi na zadane pytania należało po załadowaniu sieci stworzyć partycję komponentów (kliknąć w zakładkę *Network* następnie *Create Partition* i *Components*). A jeżeli graf składa się tylko z jednego komponentu to oznacza, że cechy grafu są też cechami komponentu.

Kajetan Dreliszak
289050



Zadanie 3 i 4

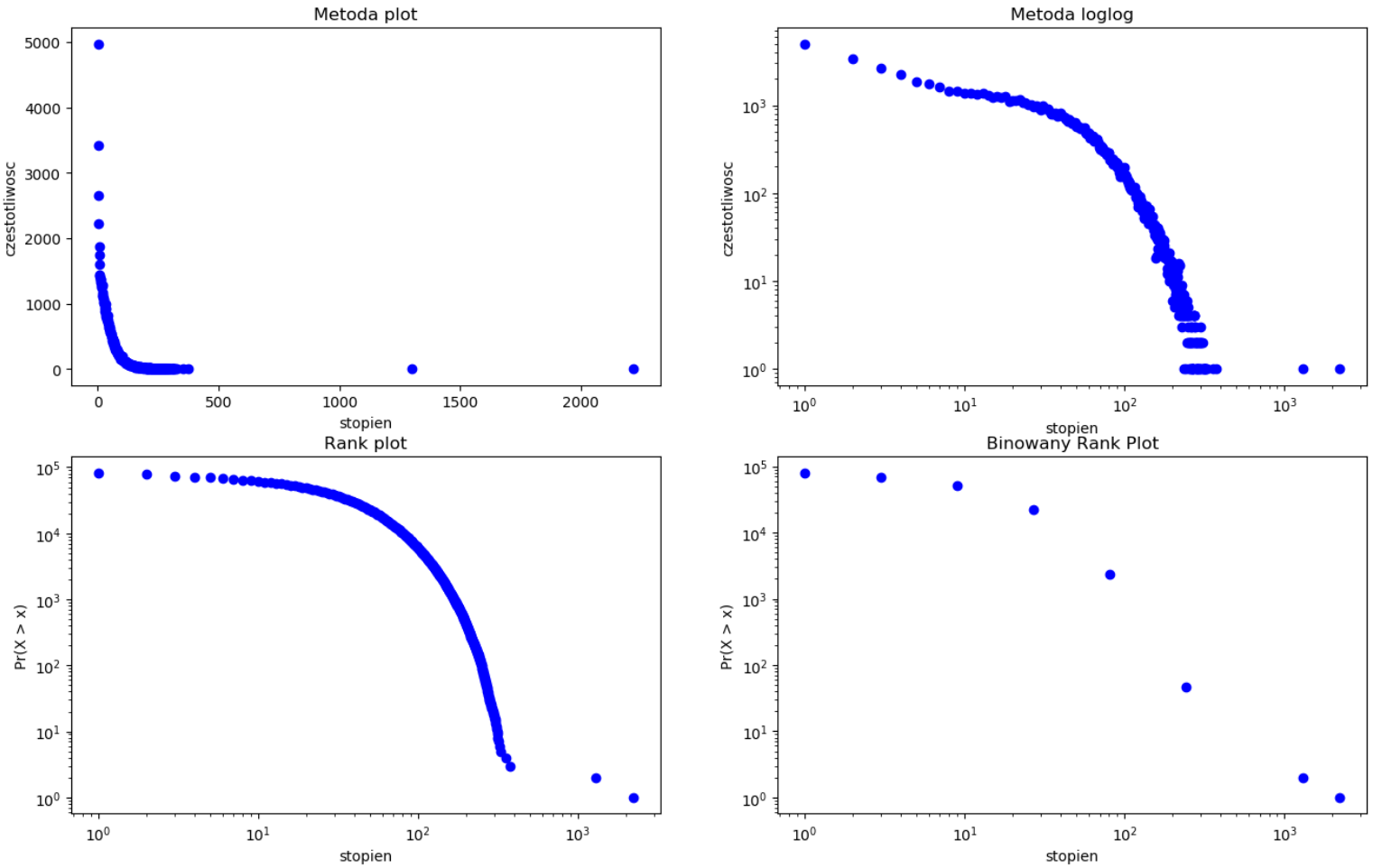
Na poniższych wykresach umieszczono w kolejności: rozkład wierzchołków na osiach w skali liniowej, rozkład wierzchołków na osiach w skali logarytmicznej, wykres rank dla częstotliwości występowania danego stopnia oraz zbinowane przedziały dla stopni wierzchołków, gdzie wielkościami przedziałów były potęgi trójki (niebieski kropki).
Z wykresów rozkładów stopni wyraźnie widać, że występuję ogon – pojedyncze wierzchołki o bardzo dużym stopniu w porównaniu do reszty.
Analizując trzeci z wykresów można zauważyć, że składają się na niego dwie charakterystyki: do „kolana” i od „kolana”. Jednakże zadaniem tego projektu była nauka narzędzi, dlatego postanowiono badać cechy wykresu jako całości.
Współczynnik potęgowy metodą regresji liniowej wyznaczony został za pomocą dwóch instrukcji. Zwróciły one identyczny wynik, więc można założyć, że jest on poprawny.

print('Wsp. potegowy metoda linregress: ', -(linregress(avr_degree, avr_rank)[0] - 1))
print('Wsp. potegowy metoda polyfit: ', -(np.polyfit(avr_degree, avr_rank, 1)[0] - 1))

Wyniki:

Wspolczynnik potegowy metoda linregress: 2.63338921294
Wspolczynnik potegowy metoda polyfit: 2.63338921294

Kajetan Dreliszak
289050



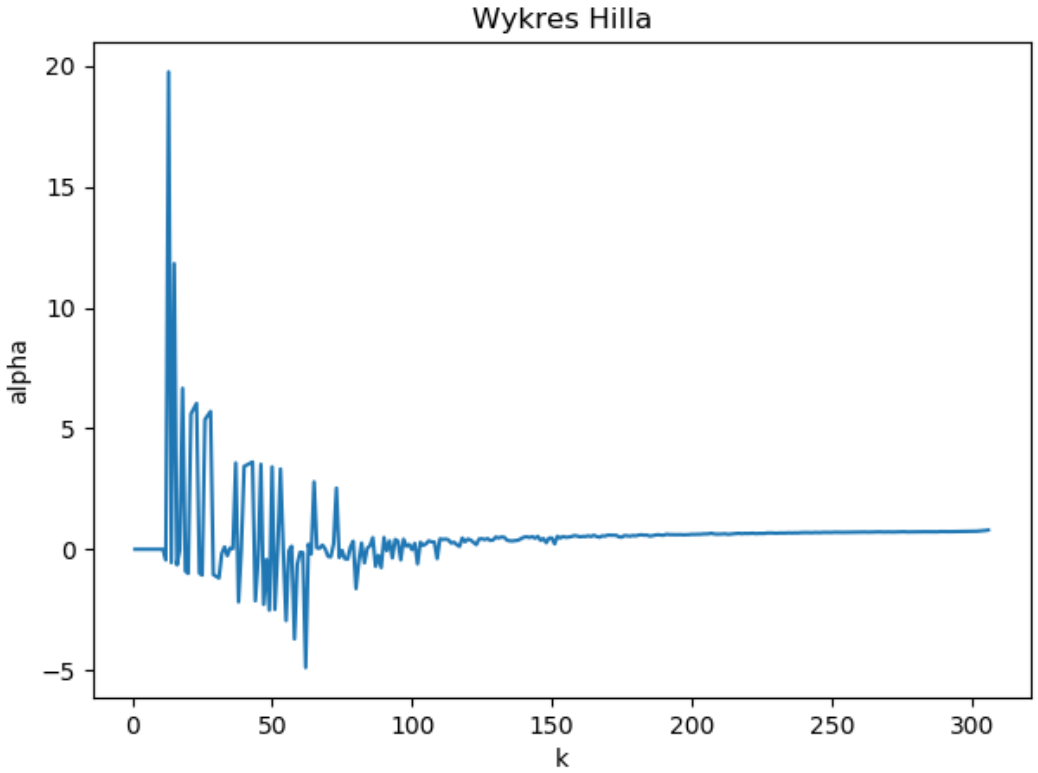
Zadanie 5

Aby wyznaczyć współczynnik potęgowy metodą Hilla należy ekspercko wybrać wartość współczynnika *k* na podstawie analizy rozkładu wierzchołków lub narysować wykres Hilla i znaleźć na nim punkt, od którego charakterystyka przyjmuje stałą wartość.

Wartość alfa została wyznaczona zgodnie ze wzorem podanym na wykładzie. Pojawiające się wartości ujemne wynikają z faktu, że w analizowanej sieci występowały wierzchołki o wyższym stopniu, które były liczniejsze niż wierzchołki o stopniu niższym.

Jak widać na poniższym wykresie w przypadku analizowanej sieci powiązań pomiędzy użytkami portalu Instragram charakterystyka stabilizuje się po odcięciu 150 ostatnich stopni wierzchołków, na poziomie alfa = 0,65.
Biorąc pod uwagę wcześniejszą obserwacją, że charakterystykę rozkładu można podzielić na dwie części wynika z tego, że usuwając drugą część – od „kolana” uzyskano by wykres o niskiej wartości współczynnika potęgowego.

Kajetan Dreliszak
289050



Kod Programu:

```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import linregress
from math import log, pow

def wczytywanie(G):
    # wczytywanie danych
    G = nx.read_edgelist('instagraph.csv', delimiter=',', nodetype=int,
encoding="utf-8")
    return G

def skl_spojne(G):
    # sprawdzanie z ilu komponentów składa się graf
    print("Liczba komponentow w grafie: %d" %
len(list(nx.connected_components(G))))
    # sprawdzanie czy graf jest spojny
    print("Czy graf jest spojny? %s" % nx.is_connected(G))
    """if nx.is_connected(G) == 1:
        # wypisanie z ilu wierzczolkow sklada się pierwszy komponent grafu
        print("Elementy pierwszego komponentu grafu grafu: %s" %
(list(nx.connected_components(G))[0]))"""
    # sprawdzenie liczby wierzcholkow
    print("Liczba wierzcholkow w grafie: %d" % len(nx.nodes(G)))
    # sprawdzenie liczby krawędzi
    print("Liczba krawedzi w grafie: %d" % len(nx.edges(G)))
    # wypisanie stopni wszystkich wierzcholkow
    print("Stopnie wierzcholkow w grafie: %s" %
```

Kajetan Dreliszak
289050

```
sorted(nx.degree(G).values(), reverse=True))
    # sprawdzenie stopnia grafu
    print("Stopien grafu: %d" % max(nx.degree(G).values()))


def regresja_liniowa(degree, freq):

    rank = []

    plt.figure("Rozklad stopni wierzcholkow", figsize=(16, 10))

    plt.subplot(221)
    plt.plot(degree, freq, 'bo', ls='None')
    plt.title("Metoda plot")
    plt.ylabel("czestotliwosc")
    plt.xlabel("stopien")

    plt.subplot(222)
    plt.loglog(degree, freq, 'bo', lw=0)
    plt.title("Metoda loglog")
    plt.ylabel("czestotliwosc")
    plt.xlabel("stopien")

    #wyliczanie rank
    for value in sorted(freq, reverse=False):
        if rank:
            rank.append(rank[-1] + value)
        else:
            rank.append(value)

    plt.subplot(223)
    plt.loglog(degree, sorted(rank, reverse=True), 'bo')
    plt.title("Rank plot")
    plt.ylabel("Pr(X > x)")
    plt.xlabel("stopien")

    avr_rank = []
    avr_degree = []
    avr = 0
    step = 0
    index = -1

    #binowanie co potege 3
    for value in sorted(degree, reverse=False):
        index = index + 1
        if value >= 3 ** step:
            if avr_rank:
                avr_rank[-1] = avr_rank[-1] / avr
            step = step + 1
            avr_rank.append(sorted(rank, reverse=True)[index])
            avr = 1
            avr_degree.append(value)
        else:
            avr_rank[-1] = avr_rank[-1] + sorted(rank, reverse=True)[index]
            avr = avr + 1

    fit = np.polyfit(np.log(avr_degree), np.log(avr_rank), 1)
    fit_fn = np.poly1d(fit)

    plt.subplot(224)
    plt.loglog(avr_degree, avr_rank, 'bo')
```

Kajetan Dreliszak
289050

```
    #, fit_fn(avr_degree), avr_degree, '--k')
    plt.title("Binowany Rank Plot")
    plt.ylabel("Pr(X > x)")
    plt.xlabel("stopien")

    print('Wspolczynnik potegowy metoda linregress: ', -
(linregress(np.log(avr_degree), np.log(avr_rank))[0] - 1))
    print('Wspolczynnik potegowy metoda polyfit: ', -
(np.polyfit(np.log(avr_degree), np.log(avr_rank), 1)[0] - 1))

    print(np.polyfit(np.log(avr_degree), np.log(avr_rank), 1))
    print(linregress(np.log(avr_degree), np.log(avr_rank)))

    print('Freq: ', freq)
    print('Rank: ', rank)
    print('Rank R: ', sorted(rank, reverse=True))
    print('Avr_Rank: ', avr_rank)
    print('Avr_Degree: ', avr_degree)
    print('Degree: ', degree)
    print('Degree R: ', sorted(degree, reverse=False))


def metoda_hilla(freq):

    alpha = []
    k_list = list(range(1, len(freq)))

    for k in k_list:
        suma = 0
        for i in range(0, k):
            suma = suma + log(freq[-i - 1] / freq[-k - 1])
            #print(suma)

        #print(freq[-i - 1], freq[-k - 1], log(freq[-i - 1] / freq[-k -
1]))
        print(suma, 1/k, 1/k * suma)
        gamma = (1 / k) * suma
        #print(k)
        print('Gamma: ', gamma)
        if gamma == 0:
            alpha.append(0)
        else:
            alpha.append(1 + pow(gamma, -1))
        print('Alfa: ', alpha)

    # print(alpha)
    # print(k_list)

    plt.figure("Wykres Hilla", figsize=(7, 5))

    plt.plot(k_list, alpha)
    plt.title("Wykres Hilla")
    plt.ylabel("alpha")
    plt.xlabel("k")


def stopnie_czestotliwosc(G):

    degree = []
    freq = []
```

Kajetan Dreliszak
289050

```
    for value in sorted(nx.degree(G).values()):
        if degree and degree[-1] == value:
            freq[-1] = freq[-1] + 1
        else:
            degree.append(value)
            freq.append(1)

    return degree, freq


G = nx.Graph()
G = wczytywanie(G)

skl_spojne(G)

degree, freq = stopnie_czestotliwosc(G)

regresja_liniowa(degree, freq)

metoda_hilla(freq)

plt.show()
```