

TASS - Projekt 1

Bartosz Latosek

Listopad 2024

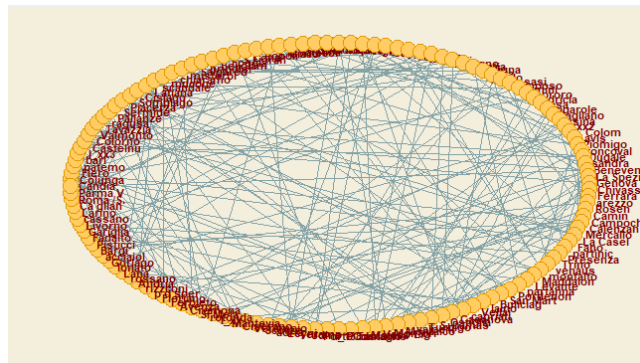
1 Projekt A

Za pomocą Pajeka należy dokonać analizy grafu „małego”, tj. o rozmiarze umożliwiającym wykorzystanie standardowych algorytmów wizualizacji i grupowania.

Zbiór danych = **310790 % 7 = 4** - Sieć energetyczna Włoch.

Uzyskana w programie *Pajek* wizualizacja sieci:

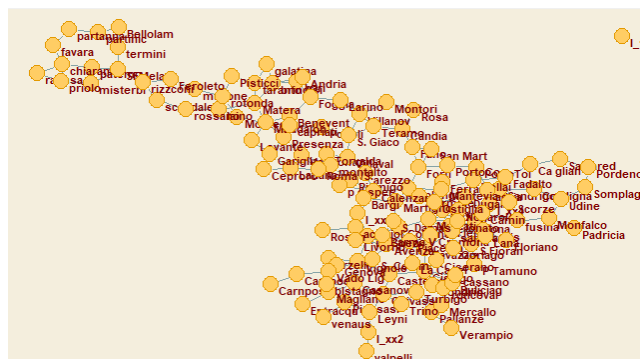
Draw > Network > Layout > Circular > Original



Rysunek 1: Wizualizacja sieci energetycznej Włoch

Schemat sieci wraz ze stopniami wierzchołków:

Draw > Network > Layout > Energy > Kawada-Kamai > SeparateComponents



Rysunek 2: Schemat sieci energetycznej Włoch wraz ze stopniami wierzchołków

1.1 Zbadaj, jaki jest rząd i rozmiar całej sieci, a następnie wyodrębnij największą składową spójną, zbadaj jej rząd i rozmiar

Aby uzyskać te informacje należy użyć funkcji *Info Network* a następnie podać arbitralny zakres uwzględniający wszystkie węzły. W moim przypadku było to 0 - 1000. Uzyskany rząd sieci to **137** a jej rozmiar to **205**.

```
Reading Network --- C:\Users\Bartosz\Desktop\tass\data.net
Working... 344 lines read.
Time spent: 0:00:00

1. C:\Users\Bartosz\Desktop\tass\data.net (137)
Number of vertices (n): 137
-----
Arcs Edges
-----
Total number of lines 0 205
Number of loops 0 0
Number of multiple lines 0 0
-----
Density1 [loops allowed] = 0.02184453
Density2 [no loops allowed] = 0.02200515
Average Degree = 2.99270073

The highest values of lines:
Rank Line Value Line-Id
-----
1 1-67 1.00000 Parma V-La Spezi
```

Rysunek 3: Logi uzyskane za pomocą funkcji *Info Network*

W celu wyznaczenia największej składowej spójnej należy posłużyć się funkcją:

Network > Createpartition > Components > Strong

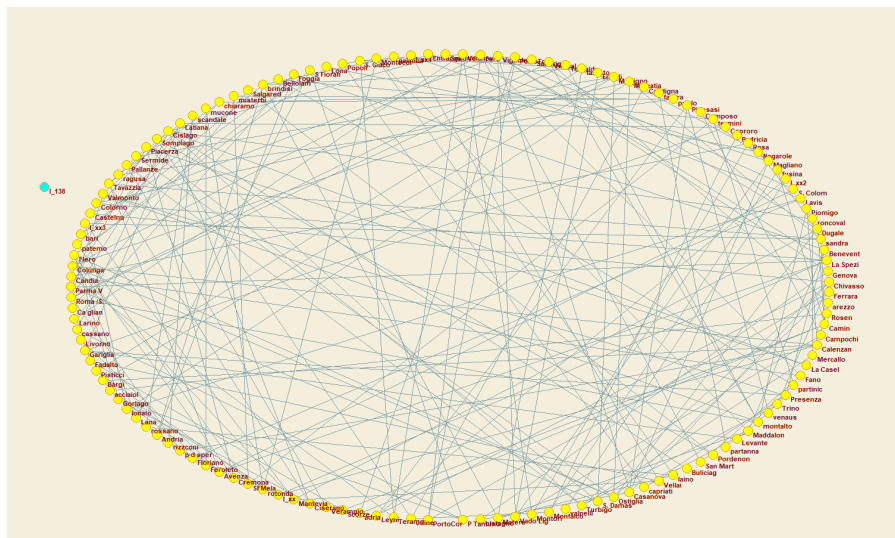
Uzyskana składowa spójna ma 1 węzeł mniej niż cała sieć, a więc jej rząd wynosi **136**. Rozmiar pozostaje bez zmian i wynosi **205**.

```
Strong Components
Working...
Number of components: 2
Size of the largest component: 136 vertices (99.270%).
Time spent: 0:00:00
```

Rysunek 4: Logi uzyskane po wyznaczeniu największej składowej spójnej

1.2 Wykreśl największą składową spójną i skomentuj wynik

Do wykreślenia największej składowej spójnej utworzyłem ją jeszcze raz, analogicznie do poprzedniego podpunktu. Następnie po zaznaczeniu pola wyboru odpowiadającego partycji i namalowaniu jej, uzyskałem poniższy schemat:



Rysunek 5: Wykres największej składowej wspólnej

Na powyższym rysunku widać, że tylko jeden węzeł (po lewej stronie) nie jest połączony z żadnym innym węzłem a więc stanowi oddzielną partycję.

1.3 Przeprowadź grupowanie metodą Warda z metryką d1 (odległość dwóch węzłów to liczba sąsiadów połączonych tylko z jednym z nich)

W celu wykonania zadania przeprowadzono następujące działania:

Cluster > Create Complete Cluster > Wybrano 137 - liczbę węzłów

Następnie, w celu uzyskania dendrogramu:

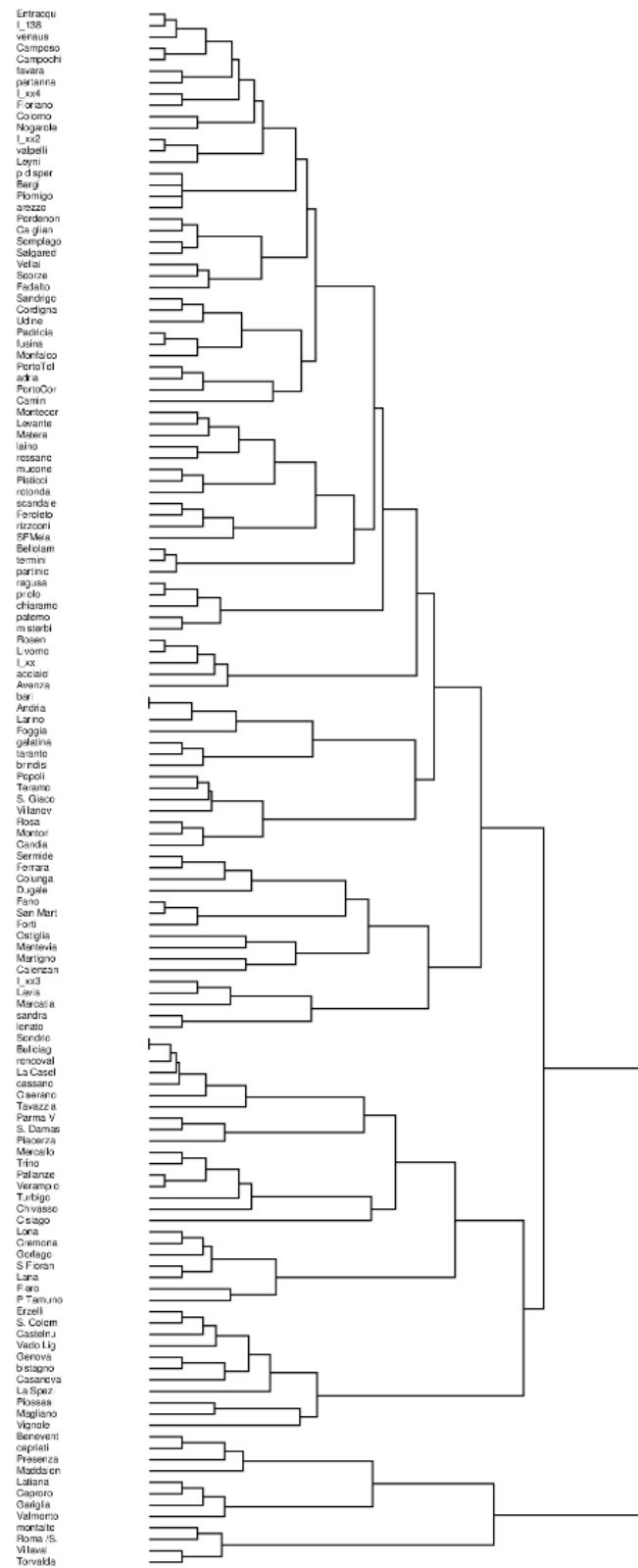
Operations > Network + Cluster > Dissimilarity > Network based > d1 > All

Uzyskany plik .eps został następnie zapisany.

1.4 Wykreśl dendrogram i zaproponuj cięcie

Wykreślony plik .eps (za pomocą serwisu online) wygląda następująco:

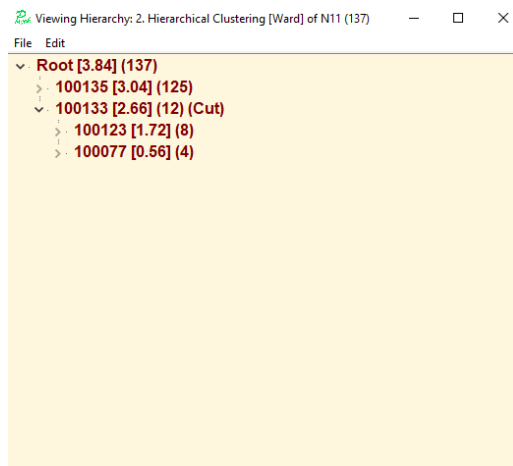
Pajek - Ward [0.00,3.84]



Rysunek 6: Dendrogram sieci

Patrząc na powyższy wykres, najbardziej wyróżnia się węzeł *Csiago*, który ma najwyższy rząd wśród wszystkich węzłów w sieci. W celu wyznaczenia optymalnego miejsca przecięcia można byłoby przeanalizować współczynniki niespójności (ang. *inconsistency*) ale dla uproszczenia przyjmę, że powyższy dendrogram podzielę na 2 grupy, które są ze sobą najmniej połączone. Na dendrogramie obrazowane są one najbardziej wysuniętą na prawo "klamrą".

Cięcia dokonano za pomocą widoku hierarchii:

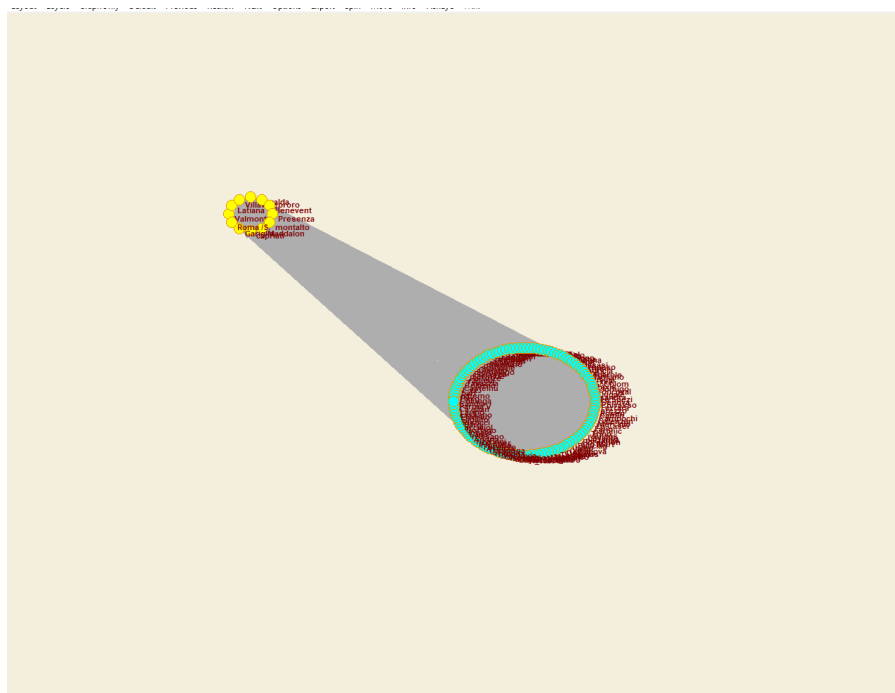


Rysunek 7: Widok hierarchii

1.5 Wykreśl wyodrębnione grupy

W celu uzyskania obrazu przedstawiającego wykreślone grupy wykonano szereg następujących operacji:

Draw > Network + First Partition > Layout > Circular > Using Partition > Different centers



Rysunek 8: Wykreślone grupy

2 Projekt B

Za pomocą `networkx` należy zbadać występowanie w większym grafie wybranych zjawisk charakterystycznych dla sieci złożonych.

Zbiór danych = **310790** % **6** = **2** - Sieć połączeń lotniczych.

Analiza została wykonana przy pomocy *Jupyter Notebook* z wykorzystaniem biblioteki do analizy sieci - *networkx*.

2.1 Zbadaj jaki jest rząd i rozmiar całej sieci: pierwotnej oraz po usunięciu pętli i duplikatów krawędzi

Wykorzystując bibliotekę *networkx* języka *Python* oraz następujący kod, załadowano dane do dwóch różnych rodzajów grafów:

```
1. Analiza rzędu i rozmiaru sieci

multigraph_network = nx.read_edgelist(DATASET, create_using=nx.MultiGraph)
graph_network = nx.read_edgelist(DATASET, create_using=nx.Graph)

[3] ✓ 0.1s

def print_graph_info(graph, graph_name):
    num_nodes = graph.number_of_nodes()
    num_edges = graph.number_of_edges()
    avg_degree = sum(dict(graph.degree()).values()) / num_nodes if num_nodes > 0 else "inf"

    print(f"{graph_name} Info:")
    print(f" - Number of nodes: {num_nodes}")
    print(f" - Number of edges: {num_edges}")
    print(f" - Average degree: {avg_degree:.4f}")
    print(f" - Is directed: {graph.is_directed()}")
    print(f" - Is multigraph: {graph.is_multigraph()}")

# Print info for each network
print_graph_info(multigraph_network, "MultiGraph Network")
print()
print_graph_info(graph_network, "Graph Network")

[15] ✓ 0.0s
```

Rysunek 9: Kod źródłowy ładujący dane do grafów

Uzyskano następujące wyniki:

MultiGraph Network Info:

- Number of nodes: 3425
- Number of edges: 19256
- Average degree: 11.2444
- Is directed: False
- Is multigraph: False

Graph Network Info:

- Number of nodes: 3425
- Number of edges: 19256
- Average degree: 11.2444
- Is directed: False
- Is multigraph: False

Zatem rząd sieci pierwotnej to **3425** a jej rozmiar to **19256**.

Następnie, za pomocą funkcji:

```
nx.number_of_selfloops(multigraph_network)
```

zbadano liczbę pętli w obydwu grafach i uzyskano następujące wyniki:

MultiGraph selfloops

1

Graph selfloops

1

W celu usunięcia pętli i zduplikowanych krawędzi wykorzystano następującą funkcję:

```
graph_network.remove_edges_from(nx.selfloop_edges(graph_network))
```

w wyniku czego otrzymano sieć o następujących właściwościach:

Graph Network Info:

- Number of nodes: 3425
- Number of edges: 19256
- Average degree: 11.2444
- Is directed: False
- Is multigraph: False

Zatem rząd sieci bez duplikatów i pętli to **3425** a jej rozmiar to **19256**.

2.2 Wyodrębnij największą składową spójną, zbadaj jej rząd i rozmiar

W pierwszej kolejności należy, za pomocą funkcji:

```
nx.is_connected(graph_network)
```

sprawdzić, czy cały graf jest spójny (największa składowa spójna to po prostu cały graf).

Otrzymaliśmy wynik *False*, a więc szukamy największej składowej spójnej za pomocą poniższego kodu:

```
connected_components = list(nx.connected_components(graph_network))
largest_component = max(connected_components, key=len)
largest_subgraph = graph_network.subgraph(largest_component)
```

```
order = largest_subgraph.number_of_nodes()
size = largest_subgraph.number_of_edges()
```

W wyniku otrzymaliśmy:

```
Order: 3397
Size: 19230
```

zatem największa składowa spójna ma rząd **3397** i rozmiar **19230**.

2.3 Wyznacz aproksymacje średniej długości ścieżki, operując na próbie losowej 100, 1000 i 10 tys. par wierzchołków

Do wyznaczenia aproksymacji średniej długości ścieżki wykorzystano poniższy kod:

```

  ~ Aproksymacje średniej długości ścieżki

import random

def average_shortest_path_length(G, num_samples):
    nodes = list(G.nodes())
    total_length = 0
    valid_pairs = 0

    while valid_pairs < num_samples:
        u, v = random.sample(nodes, 2)
        if nx.has_path(G, u, v):
            total_length += nx.shortest_path_length(G, u, v)
            valid_pairs += 1

    return total_length / valid_pairs

[17] ✓ 0.0s
```

Rysunek 10: Kod źródłowy do aproksymacji średniej długości ścieżki

W każdym kroku wybierane są 2 węzły z grafu (dla których istnieje ścieżka), między którymi następnie wyznaczana jest najkrótsza ścieżka. W wyniku otrzymujemy (wyniki różnią się przy każdym uruchomieniu programu):

Aproksymacja średniej długości ścieżki (dla 100 próbek): 3.88
Aproksymacja średniej długości ścieżki (dla 1000 próbek): 4.08
Aproksymacja średniej długości ścieżki (dla 10000 próbek): 4.10

2.4 Wyznacz liczbę rdzeni o największym możliwym rzędzie, o drugim możliwie największym rzędzie o trzecim możliwie największym rzędzie; jakie to są rzędy?

Za pomocą poniższego kodu wyznaczamy i sortujemy malejąco zbiór unikalnych rzędów rdzeni.

```
core_numbers = nx.core_number(largest_subgraph)
unique_cores = sorted(set(core_numbers.values()), reverse=True)
```

Rysunek 11: Kod źródłowy do znalezienia rdzeni

Następnie, za pomocą poniższego kodu wyznaczamy rząd dla 3 największych rdzeni dla grafu:

```
three_largest_cores = unique_cores[0:3]

for i, core in enumerate(three_largest_cores):
    no_nodes = sum(1 for v in core_numbers.values() if v == core)
    print(f"{i + 1}. największy rząd rdzenia [{core}], liczba wierzchołków: {no_nodes}")
```

✓ 0.0s

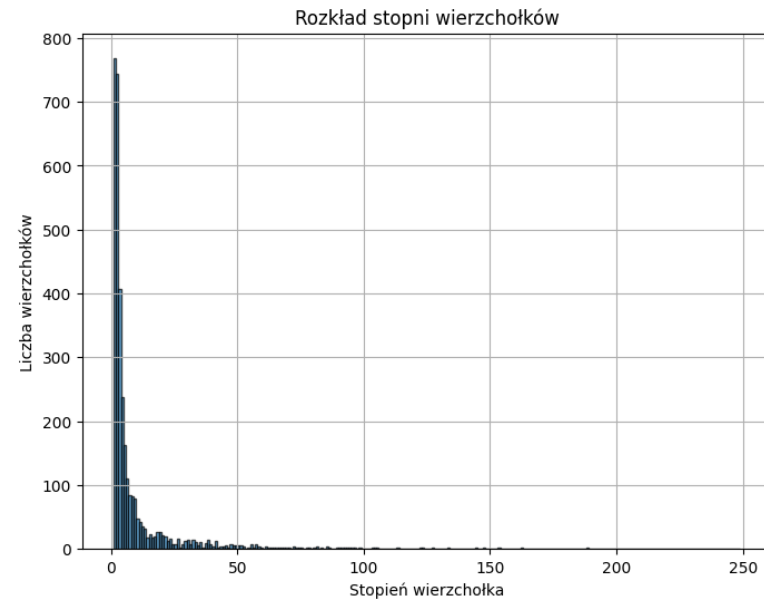
Rysunek 12: Kod źródłowy do wyznaczenia rzędów rdzeni

W wyniku otrzymujemy:

1. największy rząd rdzenia [31], liczba wierzchołków: 93
2. największy rząd rdzenia [30], liczba wierzchołków: 45
3. największy rząd rdzenia [29], liczba wierzchołków: 11

2.5 Wykreśl rozkład stopni wierzchołków

Za pomocą biblioteki *matplotlib* wykreślony został następujący rozkład stopni wierzchołków:



Rysunek 13: Rozkład stopni wierzchołków

2.6 Wyznacz wykładnik rozkładu potęgowego metodą regresji dla dopełnienia dystrybuanty rozkładu stopni, dla przedziałów rozlokowanych logarytmicznie

Za pomocą poniższego kodu wyznaczamy i normalizujemy dystrybuantę rozkładu stopni a następnie narzucamy ją na przedziały rozlokowane logarytmicznie:

```
# Dystrybuanta rozkładu stopni
degrees = [graph_network.degree(n) for n in graph_network.nodes()]

degree_counts = np.bincount(degrees)
k_values = np.arange(len(degree_counts)) # stopnie k
cum_counts = np.cumsum(degree_counts[::-1])[::-1] # dopełnienie dystrybuanty
cum_counts_normalized = cum_counts / float(len(degrees))

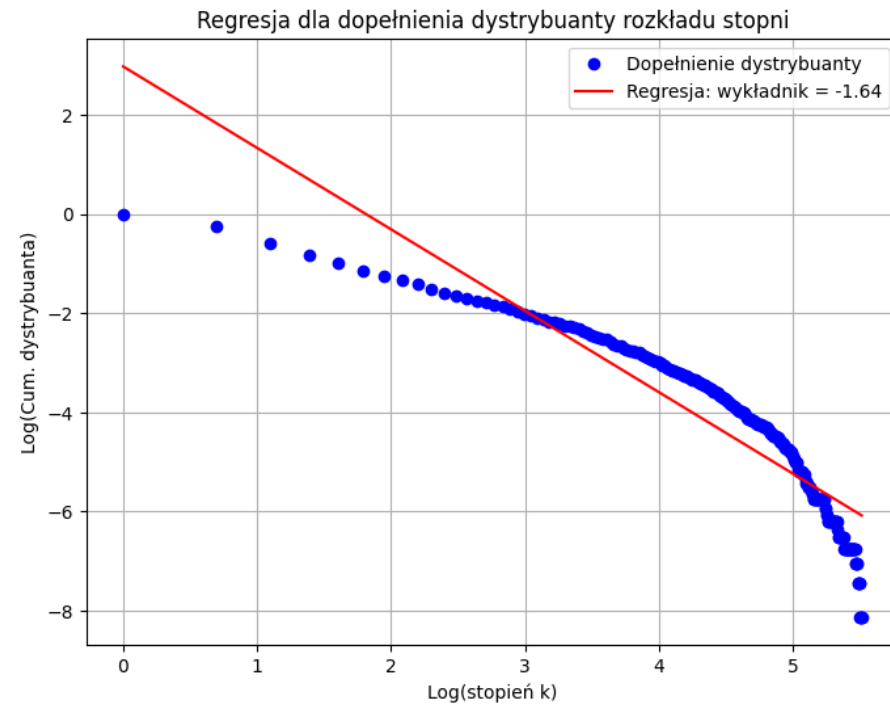
log_k = np.log(k_values[k_values > 0]) # bez log(0)
log_cum_counts = np.log(cum_counts_normalized[k_values > 0])
```

Rysunek 14: Kod służący do wyznaczenia dystrybuanty

Następnie, przy użyciu modułu *stats* biblioteki *scipy* wyznaczamy metodą regresji liniowej wykładnik rozkładu potęgowego (w tym przypadku będzie to wartość *slope*).

```
slope, intercept, r_value, p_value, std_err = st.linregress(log_k, log_cum_counts)
```

Wizualizacja działania regresji liniowej przeprowadzonej na dystrybuancie przedstawiona jest na poniższym rysunku:



Rysunek 15: Wizualizacja regresji liniowej

Wyznaczony wykładnik rozkładu potęgowego (*slope*) wynosi **-1.64**.

2.7 Wyznacz wykres Hilla

Do wykreślenia wykresu *Hilla* dla zadanego grafu wykorzystany został niniejszy kod:

```
def hill_estimator(data, k):
    if k <= 0 or k >= len(data):
        raise ValueError(f"K [{k}] musi być w przedziale (0, {len(data)})")
    x_k = data[k]
    y_k = (1 / k) * sum(np.log(data[:k] / x_k))
    return 1 + pow(y_k, -1)

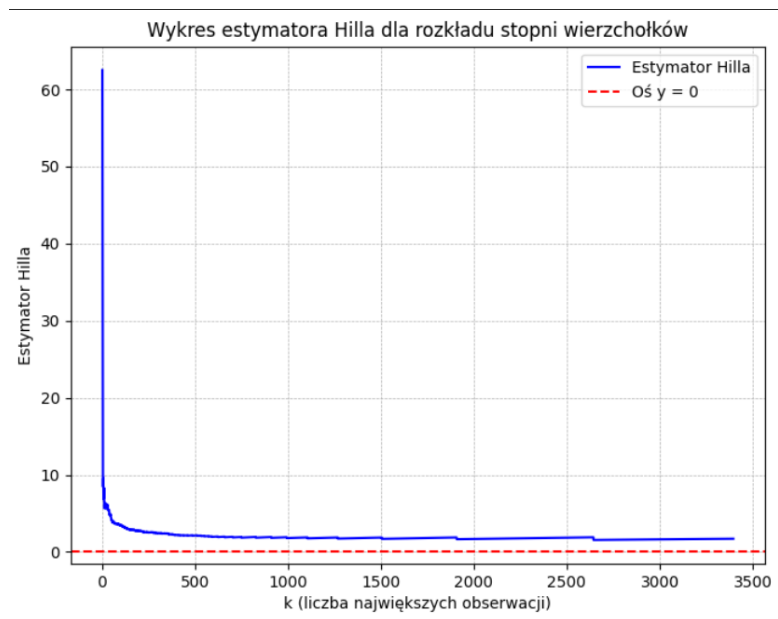
degree_sequence = np.array(sorted([d for _, d in largest_subgraph.degree()], reverse=True)) # posortowanie stopni wierzchołków malejąco
k_values = range(1, len(degree_sequence)) # wybór k (liczba największych obserwacji)

# obliczanie estymatora Hilla dla różnych wartości k
hill_values = [hill_estimator(degree_sequence, k) for k in k_values]

# rysowanie wykresu
plt.figure(figsize=(8, 6))
plt.plot(k_values, hill_values, 'b-', label="Estymator Hilla")
plt.xlabel("k (Liczba największych obserwacji)")
plt.ylabel("Estymator Hilla")
plt.title("Wykres estymatora Hilla dla rozkładu stopni wierzchołków")
plt.axhline(y=0, color="red", linestyle="--", label="0ś y = 0")
plt.legend()
plt.grid(True, linestyle="--", linewidth=0.5)
plt.show()
```

Rysunek 16: Kod służący do wyznaczenia rozkładu stopni wierzchołków

W pierwszej kolejności definiowana jest funkcja, za pomocą której wyznaczone będą wartości estymatora *Hilla* dla danych obserwacji. W pierwszym kroku, stopnie wierzchołków sortowane są malejąco, następnie wybierane są wartości k obserwacji (przyjąłem liczbę równą liczbie węzłów w grafie). W ostatnim kroku dla każdej wartości obserwacji wyznaczane są wartości estymatora *Hilla*. Wykreślony wykres Hilla wygląda następująco:



Rysunek 17: Wykres Hilla