In [4]:
```
import numpy as np
import time
import copy
import matplotlib.pyplot as plt
import networkx as nx
from collections import deque
from collections import defaultdict
import math
import random
import pandas as pd
import time
import matplotlib.colors as mcolors
from matplotlib.colors import LinearSegmentedColormap
```

Zadanie 1 - Metoda Gaussa-Jordana

Złożoność algorytmu O(n^3)

In [ ]:
```
def find_pivot_row(M, col_n):
    # znajdowanie pivota
    pivot_row = np.argmax(np.abs(M[col_n:, col_n])) + col_n
    return pivot_row

def swap_rows(M, B, i, j):
    # zamiana rzędów
    if i != j:
        M[[i, j]] = M[[j, i]]
        B[[i, j]] = B[[j, i]]

def eliminate_gauss(M, B, raw_n):
    # eliminacja w góre i w dół
    n = len(M)
    for i in range(n):
        if i != raw_n:
            factor = M[i, raw_n] / M[raw_n, raw_n]
            M[i] -= factor * M[raw_n]
```

```
            B[i] -= factor * B[raw_n]

def normalize_gauss(M, B, raw_n):
    # normalizacja wiersza
    B[raw_n] /= M[raw_n, raw_n]
    M[raw_n] /= M[raw_n, raw_n]

def solve_Gauss_Jordan(A, B):
    A = A.astype(float).copy()
    B = B.astype(float).copy()

    n = len(A)
    for raw in range(n):
        best_raw = find_pivot_row(A, raw)
        swap_rows(A, B, best_raw, raw)
        normalize_gauss(A, B, raw)  # Przekształcamy element na przekątnej do 1
        eliminate_gauss(A, B, raw)  # Zerujemy wszystkie inne elementy w kolumnie

    return B  # Teraz B zawiera rozwiązania
```

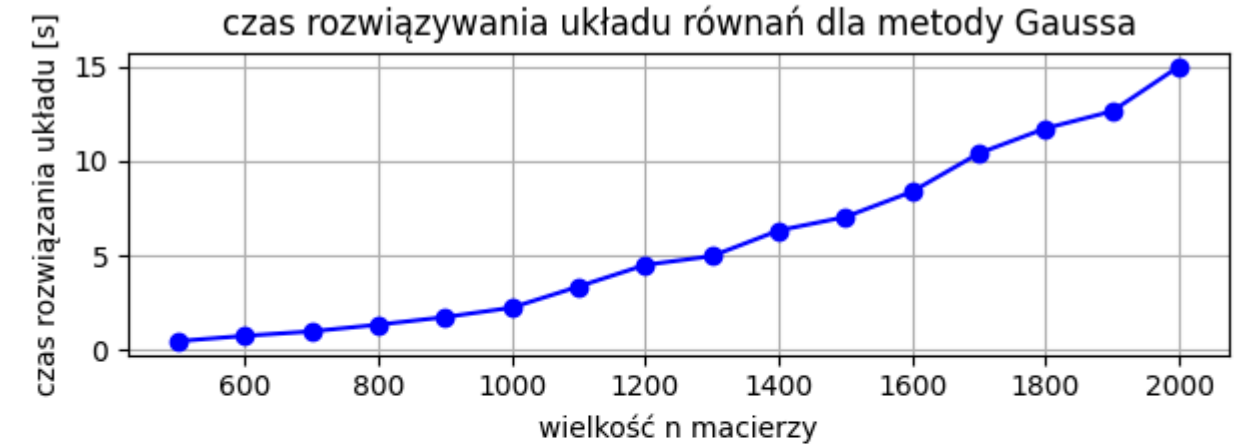
Testy poprawności algorytmu Gaussa

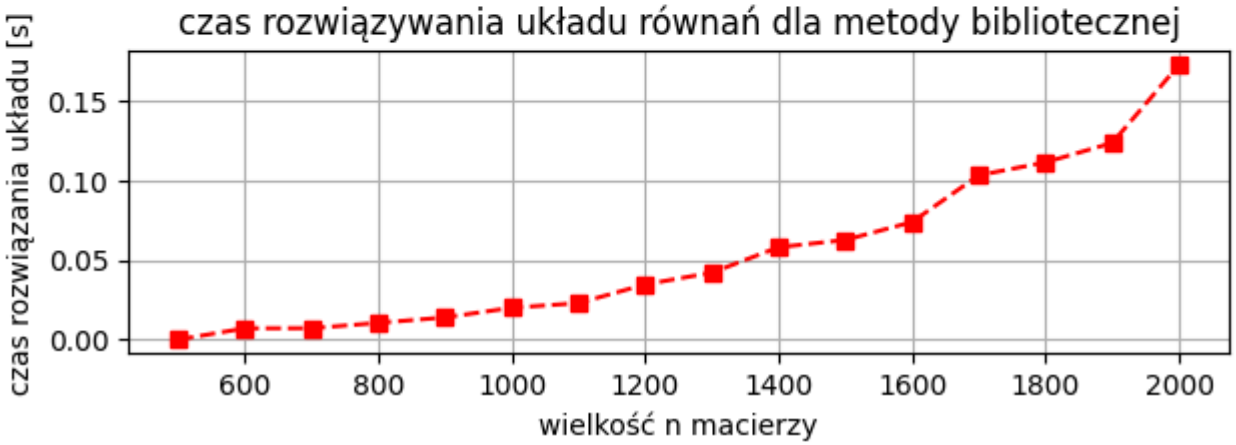
In [45]:

```
def get_random_matrix(n):
    return np.random.uniform(0,1000,size = (n,n))

def get_vector_matrix(n):
    return np.random.uniform(0, 1000, size = n)

def test_gauss_method():
    time_tab_lib = []
    time_tab_gauss = []
    n_tab = []
    result = []
    for n in range(500,2001,100):
        n_tab.append(n)
        A = get_random_matrix(n)
        B = get_vector_matrix(n)
        a_copy = copy.deepcopy(A)
        b_copy = copy.deepcopy(B)
```

```
        start = time.time()
        ans_gauss = solve_Gauss_Jordan(A,B)
        stop = time.time()
        time_tab_gauss.append(stop - start)

        start = time.time()
        ans_lib= np.linalg.solve(a_copy,b_copy)
        stop = time.time()
        time_tab_lib.append(stop - start)
        maxi = -float("inf")
        for i in range(n):
            maxi = max(maxi, abs(ans_gauss[i] - ans_lib[i]))
        result.append({"size": n, "max_error": maxi})

    df = pd.DataFrame(result)
    print(df)

    fig, ax = plt.subplots(2)
    ax[0].plot(n_tab, time_tab_gauss, marker='o', linestyle='-', color='blue')
    ax[1].plot(n_tab, time_tab_lib, marker='s', linestyle='--', color='red')

    ax[0].set_title("czas rozwiązywania układu równań dla metody Gaussa")
    ax[1].set_title("czas rozwiązywania układu równań dla metody bibliotecznej")

    ax[0].set_xlabel("wielkość n macierzy")
    ax[1].set_xlabel("wielkość n macierzy")

    ax[0].set_ylabel("czas rozwiązania układu [s]")
    ax[1].set_ylabel("czas rozwiązania układu [s]")

    ax[0].grid(True)
    ax[1].grid(True)

    plt.tight_layout()
test_gauss_method()
```

	size	max_error
0	500	5.173639e-14
1	600	1.070006e-10
2	700	8.455459e-13
3	800	1.517675e-13
4	900	9.565682e-13
5	1000	3.499423e-13
6	1100	5.301981e-12
7	1200	3.667289e-12
8	1300	2.231992e-12
9	1400	7.929213e-13
10	1500	7.466028e-12
11	1600	1.392220e-12
12	1700	5.087486e-12
13	1800	7.437606e-12
14	1900	1.929124e-12
15	2000	3.135270e-12





Jak widzimy funkcja biblioteczna jest 1000 razy szybsza od mojej implementacji. Powodem takiego stanu rzeczy jest szereg optymalizacji zawartych w kodzie funkcji biliotecznej. Warto także zauważyć, że błąd jaki moje rozwiązanie jest rzędu 10^{-12} co skłania mnie do stwierdzenia poprawności mojej implementacji.

Zadanie 2 - Faktoryzacja LU

Zaimplementowałem faktoryzacje z użyciem pivota. Z tego powodu dodaje macierz permutacji P, a równanie ma postać PA = LU

Rozwiązanie *in situ*, złożoność algorytmu O(n^3)

In [ ]:
```
def eliminate_lu(M, raw_n):
    # eliminacja w dół z równoczesnym zapisaniem mnożnika w macierzy
    n = len(M)
    for i in range(raw_n + 1, n):
        factor = M[i, raw_n] / M[raw_n, raw_n]
        M[i, raw_n] = factor
        M[i, raw_n + 1:] -= factor * M[raw_n, raw_n + 1:]

def LU(A):
    A = A.astype(float).copy()
    n = len(A)
    # P początkowo jest macierzą jednostkową
    P = np.eye(n)

    for raw in range(n):
        best_row = find_pivot_row(A, raw)
        swap_rows(A, P, best_row, raw)
        eliminate_lu(A, raw)

    return P, A
```

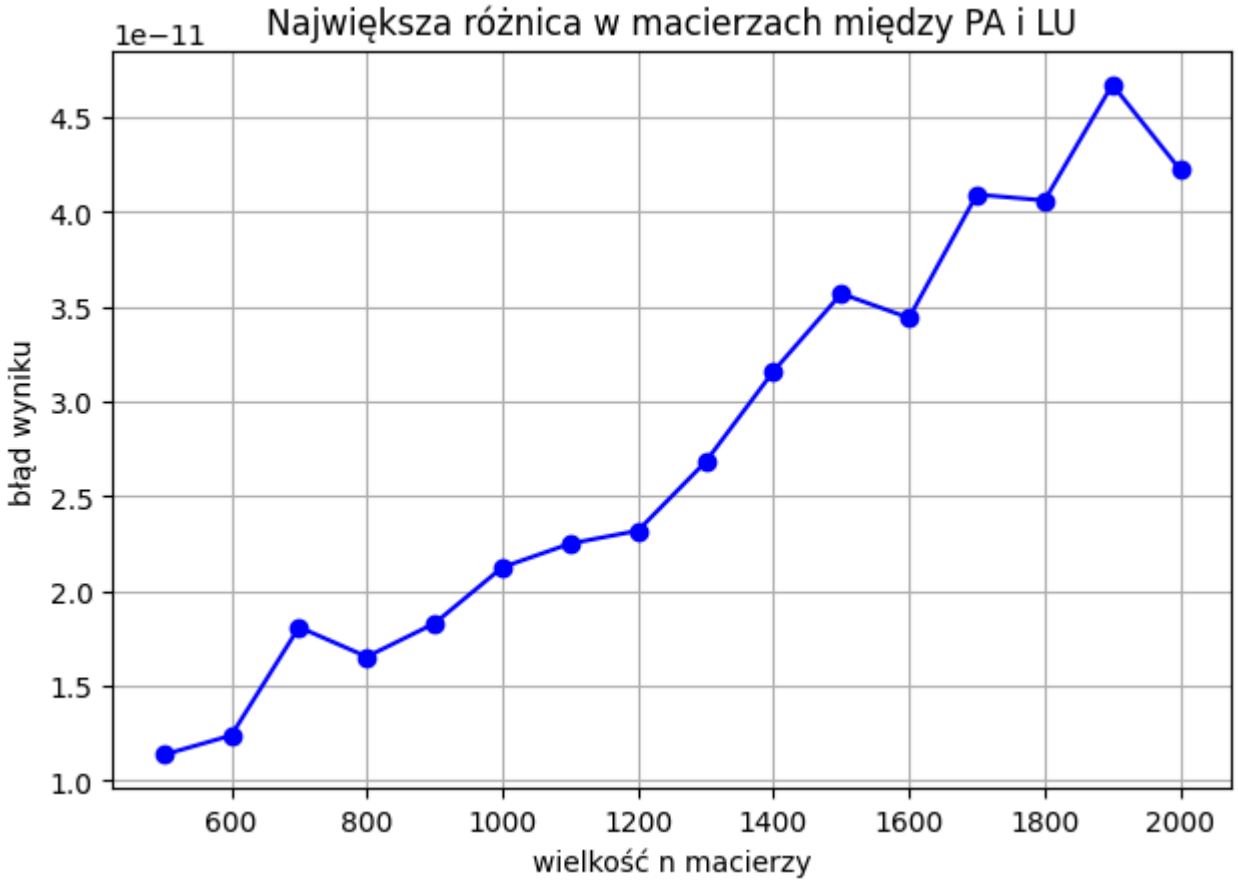
Funkcja `extract_LU` "wyciąga" z macierzy A macierze L i U

In [ ]:
```
def extract_LU(A):
    n = A.shape[0]
    L = np.tril(A, -1) + np.eye(n)
    U = np.triu(A)
    return L, U
```

Test poprawności implementacji, czyli test, czy ||PA-LU|| jest bliskie zero

In [46]:
```
def test_LU():
    difference_t = []
    n_tab=[]
    for n in range(500,2001,100):
        n_tab.append(n)
        A = get_random_matrix(n)
        P, A_lu = LU(copy.deepcopy(A))
```

```
        L, U = extract_LU(A_lu)
        left = np.dot(L,U)
        right = np.dot(P,A)
        maxi = -float("inf")
        for raw in range(n):
            for col in range(n):
                maxi = max(maxi,left[raw][col] - right[raw][col])
        difference_t.append(maxi)

    fig, ax = plt.subplots()
    ax.plot(n_tab, difference_t, marker='o', linestyle='-', color='blue')
    ax.set_title("Największa różnica w macierzach między PA i LU")
    ax.set_xlabel("wielkość n macierzy")
    ax.set_ylabel("błąd wyniku")
    ax.grid(True)
    plt.tight_layout()

test_LU()
```



W porównaniu do metody Gaussa-Jordana precyzja zmalała do 10^{-10}.

Zadanie 3. Analiza obwodu elektrycznego

Aby rozwiązać problem, przyjąłem pewne założenie, którego konsekwentnie się trzymałem. *Zakładałem, że kierunek prądu przepływającego przez krawędź jest zawsze od wierzchołka o mniejszym indeksie do wierzchołka o większym indeksie.*

To założenie pozwala mi ustrukturyzować kierunki oraz odpowienio nimi operować

Rozwiązanie zostało zaimplementowane dwoma metoda:

- Metoda bazaująca na prawa kirhofa
- Metoda potencjałów węzłowych

Po ich zaimplemenotwaniu przeprowadziłem testy działania algorytmów dla różnych grafów

Generowanie grafów i zapis do pliku

Funkcja *simple_print_graph()* to prosta funkcja rysująca graf nieskierowany. Przyjmuje graf w postaci struktury z networkx

In [5]:
```
def simple_print_graph(G):
    pos = nx.spring_layout(G, k=0.2, iterations=50)  # k = siła odpychania

    nx.draw(G, pos, with_labels=True, node_color="lightblue", edge_color="gray", node_size=500)
    plt.show()
```

Funkcje testujące otrzymane wyniki

Przy pomocy funkcji *generate_random_graph* generuje grafy o zadanym przez kroju *shape*. Funkcja przyjmuje rónież napięcie jakie należy przyłożyć do dwóch wierzchołków grafu oraz nazwę pliku do zapisu

In [36]:
```
def generate_random_graph(file_name, V, shape, emf, print=False):
    to_connect_a = random.randint(0, V - 1)
    to_connect_b = random.randint(0, V - 1)
    while to_connect_a == to_connect_b:
        to_connect_b = random.randint(0, V - 1)
    if shape == "random":
        G = nx.erdos_renyi_graph(V, 0.5)
        # Zapis do pliku jako lista krawędzi
        with open(file_name, "w", encoding="utf-8") as f:
            f.write(f"{to_connect_a} {to_connect_b} {emf} \n")
            for u, v in G.edges():
                w = np.random.uniform(1,10)
                f.write(f"{u} {v} {w}\n")
    elif shape == "cubical":
        G = nx.cubical_graph()
```

```
    to_connect_a = random.randint(0, 7)
    to_connect_b = random.randint(0, 7)
    while to_connect_a == to_connect_b:
        to_connect_b = random.randint(0, 7)

    with open(file_name, "w", encoding="utf-8") as f:
        f.write(f"{to_connect_a} {to_connect_b} {emf} \n")
        for u, v in G.edges():
            w = np.random.uniform(1,10)
            f.write(f"{u} {v} {w}\n")
elif shape == "bridge":
    size1 = V // 2
    size2 = V - size1  # Aby zapewnić sumę równą `size`

    # Tworzenie dwóch niezależnych grafów
    G1 = nx.erdos_renyi_graph(size1, 0.6)
    G2 = nx.erdos_renyi_graph(size2, 0.5)

    # Przesunięcie indeksów drugiego grafu, aby uniknąć konfliktów
    mapping = {node: node + size1 for node in G2.nodes()}
    G2 = nx.relabel_nodes(G2, mapping)

    # Połączenie dwóch podgrafów w jeden graf
    G = nx.union(G1, G2)

    # Dodanie losowego mostka między nimi
    bridge_start = random.choice(list(G1.nodes()))
    bridge_end = random.choice(list(G2.nodes()))
    G.add_edge(bridge_start, bridge_end)

    to_connect_a = random.randint(0, size1-1)
    to_connect_b = random.randint(size1, V - 1)
    with open(file_name, "w", encoding="utf-8") as f:
        f.write(f"{to_connect_a} {to_connect_b} {emf} \n")
        for u, v in G.edges():
            w = np.random.uniform(1,10)
            f.write(f"{u} {v} {w}\n")
elif shape == "bridge3":
    size = V // 3  # Aby zapewnić sumę równą `size`
```

```
        # Tworzenie dwóch niezależnych grafów
        G1 = nx.erdos_renyi_graph(size, 0.6)
        G2 = nx.erdos_renyi_graph(size, 0.5)
        G3 = nx.erdos_renyi_graph(size, 0.5)
        # Przesunięcie indeksów drugiego grafu, aby uniknąć konfliktów
        mapping = {node: node + size for node in G2.nodes()}
        G2 = nx.relabel_nodes(G2, mapping)
        mapping = {node: node + 2* size for node in G3.nodes()}
        G3 = nx.relabel_nodes(G3, mapping)
        # Połączenie dwóch podgrafów w jeden graf
        G = nx.union(G1, G2)
        G = nx.union(G,G3)
        # Dodanie losowego mostka między nimi
        bridge_start12 = random.choice(list(G1.nodes()))
        bridge_end12 = random.choice(list(G2.nodes()))

        bridge_start13 = random.choice(list(G1.nodes()))
        bridge_end13 = random.choice(list(G3.nodes()))

        bridge_start23 = random.choice(list(G2.nodes()))
        bridge_end23 = random.choice(list(G3.nodes()))

        G.add_edge(bridge_start12, bridge_end12)
        G.add_edge(bridge_start13, bridge_end13)
        G.add_edge(bridge_start23, bridge_end23)

        to_connect_a = random.randint(0, size-1)
        to_connect_b = random.randint(size, V - 1)
        with open(file_name, "w", encoding="utf-8") as f:
            f.write(f"{to_connect_a} {to_connect_b} {emf} \n")
            for u, v in G.edges():
                w = np.random.uniform(1,10)
                f.write(f"{u} {v} {w}\n")
    elif shape == "grid":
        n = int(math.sqrt(V))  # Wymiary siatki (n x n)
        G = nx.grid_2d_graph(n, n)  # Tworzymy graf siatkowy 2D
        to_connect_a = random.randint(0, n**2 - 1)
        to_connect_b = random.randint(0, n**2 - 1)
        while to_connect_a == to_connect_b:
            to_connect_b = random.randint(0, n**2 - 1)
        # Przekształcamy etykiety wierzchołków na indeksy liczbowe
```

```
        mapping = {node: i for i, node in enumerate(G.nodes())}
        G = nx.relabel_nodes(G, mapping)
        with open(file_name, "w", encoding="utf-8") as f:
            f.write(f"{to_connect_a} {to_connect_b} {emf} \n")
            for u, v in G.edges():
                w = np.random.uniform(1,10)
                f.write(f"{u} {v} {w}\n")

    elif shape == "small_world":
        G = nx.watts_strogatz_graph(V, 10, 0.5)
        with open(file_name, "w", encoding="utf-8") as f:
            f.write(f"{to_connect_a} {to_connect_b} {emf} \n")
            for u, v in G.edges():
                w = np.random.uniform(1,10)
                f.write(f"{u} {v} {w}\n")
    else:
        raise Exception("Nie ma takiego typu grafu")
    if print:
        simple_print_graph(G)
# Przykładowe użycie
```

Funkcja sprawdza, czy krawędzie zapisane w słowniku `edges_dict` tworzą cykl w grafie. Krawędzie są reprezentowane jako pary wierzchołków i powiązane z wartością oporu.

In [7]:
```
def check_cycles(edges_dict):
    # Tworzymy listę wierzchołków z krawędzi, ignorując krawędzie o zerowym oporze
    vertices = []
    for edge, resistance in edges_dict.items():
        if resistance != 0:
            # Dodajemy wierzchołki do listy
            vertices.append(edge[0])
            vertices.append(edge[1])

    # Sortujemy wierzchołki
    vertices.sort()

    # Sprawdzamy, czy krawędzie tworzą cykl
    for i in range(0, len(vertices)//2):
        if vertices[2*i] != vertices[2*i + 1]:
            return False
```

```
    return True
```

Funkcja `check_road` sprawdza, czy istnieje ścieżka w grafie, która zaczyna się w wierzchołku `start_vertex` i kończy w wierzchołku `end_vertex`. Ścieżka jest tworzona z krawędzi zapisanych w słowniku `edges_dict`, gdzie klucze to pary wierzchołków, a wartości to opory (ignorowane w tej funkcji).

In [8]:
```
def check_road(edges_dict, start_vertex, end_vertex):
    # Lista, która przechowuje wszystkie wierzchołki związane z krawędziami w słowniku
    vertices = []

    # Dodajemy wierzchołki, które są częścią krawędzi
    for edge, resistance in edges_dict.items():
        if resistance != 0:
            vertices.append(edge[0])
            vertices.append(edge[1])

    # Sprawdzamy, czy start_vertex i end_vertex są obecne na liście wierzchołków
    if start_vertex not in vertices or end_vertex not in vertices:
        return False

    # Usuwamy start_vertex i end_vertex z listy
    vertices.remove(start_vertex)
    vertices.remove(end_vertex)

    # Sortujemy pozostałe wierzchołki, aby łatwiej porównać je parami
    vertices.sort()

    # Sprawdzamy, czy ścieżka jest możliwa (czy pozostałe wierzchołki tworzą cykl)
    for i in range(0, len(vertices) // 2):
        if vertices[2*i] != vertices[2*i + 1]:
            return False

    return True
```

Funkcja sprawdza, czy natężenie prądu w grafie (reprezentowanym przez `graph`) wchodzącego do wierzchołka `s` oraz wychodzącego z wierzchołka `t` jest zgodne z oczekiwaniami. Dla każdego wierzchołka sumuje natężenie przepływu i sprawdza, czy total_current jest bliska zeru.

In [9]:
```python
def check_intensity(currents_edge_dict, graph, total_current, s ,t):
    n = len(graph)
    Q = deque()
    visited = [False]*n
    visited[s] = True
    Q.append(s)

    while len(Q) > 0:
        # Przechodzę przez każdy wierzchołek grafie
        u = Q.popleft()
        sum = 0
        for v, _ in graph[u]:
            # Rozpatruje każdą krawędź wychodzącą z danego wierzchołka

            if not visited[v]:
                # dodawanie do listy
                visited[v] = True
                Q.append(v)
            # krotka identyfikująca krawędź
            id = (min(u,v),max(u,v))
            # warunki na zmiane kierunku (bo zakładam, że prąd zawsze biegnie
            #  z wierzchołka o niższym numerze, do wierzchołka o wyższym numerze)
            if u<v:
                sum += currents_edge_dict[id]
            else:
                sum -= currents_edge_dict[id]
        # dodaje natężenie wypływające z SEM stosownie do wierzchołka
        if u == s:
            sum -= total_current
        if u == t:
            sum += total_current
        assert np.isclose(sum,0), f"the sum of incoming and outgoing currents is not equal {sum}"
```

Metoda bazująca na prawach kirhofa

Funkcje pomocnicze

Funkcja `load_graph_from_file(file_name)` wczytuje graf z pliku o nazwie `file_name` oraz tworzy graf listowy. na wyjściu zwraca krotke wartości (s, t, E) - wierzchołki, które połączyliśmy siłą elektromotoryczną SEM oraz graf listowy

Funkcja `get_max_number(file_name)` wylicza wielkość grafu z pliku.

In [10]:
```python
def get_max_number(file_name):
    maxi = 0
    with open(file_name, "r", encoding="utf-8") as f:
        # Wczytanie pierwszej linii z dodatkowymi danymi

        _ = f.readline().strip()  # Usuwamy ewentualne białe znaki na początku i końcu

        # Wczytanie pozostałych linii, które zawierają krawędzie
        for line in f:
            u, v, _ = line.strip().split()  # Podział po spacjach
            u, v = int(u), int(v)
            maxi = max(maxi,u,v)
    return maxi

def load_graph_from_file(file_name):
    # graf w postaci listowej
    graph = [[] for _ in range( get_max_number(file_name)+1)]

    with open(file_name, "r", encoding="utf-8") as f:
        # Wczytanie pierwszej linii z dodatkowymi danymi

        s, t, emp = f.readline().strip().split()

        # Wczytanie pozostałych linii, które zawierają krawędzie
        for line in f:
            u, v, weight = line.strip().split()  # Podział po spacjach
            u, v = int(u), int(v)
            weight = float(weight)
            graph[u].append((v,weight))
            graph[v].append((u,weight)) # Przekształcamy każdą linię na dwie liczby
            # graph.add_edge(u, v, weight=weight)  # Dodajemy krawędź do grafu
    return graph, int(s), int(t), float(emp)
```

Funkcja `list_to_weighted_matrix(adj_list)` konwertuje graf zapisany w postaci listy sąsiedztwa, na graf zapisany macierzowo. Pozwala to na szybszy dostęp do wag grafu

In [11]:
```python
def list_to_weighted_matrix(adj_list):
    n = len(adj_list)  # Liczba wierzchołków w grafie
    matrix = np.zeros((n, n), dtype=float)  # matrix sąsiedztwa wypełniona zerami (dla grafu nieważonego można dać int)

    for i, neighbors in enumerate(adj_list):
        for j, weight in neighbors:
            matrix[i][j] = weight  # Ustawiamy wagę krawędzi

    return matrix
```

Funkcja `extract_path` z grafu matrix_graph wyznacza ścieżke od wierzchołka `vertix` przy pomocy tablicy `parents` wyznacza ściezke z tego wierzchołka do wierzchołka początkowego w drzewie przeszukiwań bfs. Flaga reverse odpowiada za "kierunek" ścieżki, która została wyekstraktowana.

Funkcja zwraca edge_weights, który jest słownikiem. Kluszmi w nim są zapisane za pomocą krotek krawędzie wraz z oporem tych krawędzi

In [12]:
```python
def extract_path(matrix_graph, parents, vertix, reverse):
    edge_weights = defaultdict(int)
    # zwykłe cofanie po parentach
    while parents[vertix] != -1:
        parent = parents[vertix]
        # tworze etykiete dla krawędzi za pomocą poniższej krotki
        edge = (min(parent, vertix), max(parent, vertix))

        edge_weights[edge] += matrix_graph[parent][vertix]

        if reverse:
            edge_weights[edge] *= -1
        # założenie, że prąd płynie od wierzchołka od wierzchołka o niższym numerze do wierzchołka o wyższym numerze
        if parent > vertix:
            edge_weights[edge] *= -1

        vertix = parent

    return edge_weights
```

Funkcja `build_matrix` składa macierz na podstawie slowników zawierających informacje o "oczkach" w układzie oraz macierzy rzadkiej trzymającej równania wyprowadzone z I prawa kirhofa dla każdego węzła

In [ ]:
```python
def build_matrix(emf, edge_to_num, eges_num, loops_only_resist, sparse_matrix, loops_with_eps):
    # macierz do uzupełnienia
    matrix = []
    # po "prawej" stronie równania
    B = []

    def add_row(dictionary, value, use_edge_map=True):
        row = np.zeros(eges_num, dtype=float)
        for key, val in dictionary.items():
            row[edge_to_num[key] if use_edge_map else key] = val
        matrix.append(row)
        B.append(value)

    # rozpatruje cykle zawierajace SEM
    for path in loops_with_eps:
        add_row(path, emf)
```

```
    # rozpatruje cykle na samych opornikach
    for loop in loops_only_resist:
        add_row(loop, 0)

    # rozpatruje równia wywodzące się z I prawa kirhofa
    for sparse_row in sparse_matrix:
        add_row(sparse_row, 0, use_edge_map=False)
    return matrix, B
```

Funkcja `test_kirhof_solution` testuje wyliczone pętle oraz natężenia

In [ ]:
```
def test_kirhof_solution(loops_only_resist, loops_with_eps, current_edges, G, total_current, s, t):
    for loop in loops_only_resist:
        if not check_cycles(loop):
            assert False, f"niepoprawna pętla {loop.keys()}"

    for loop in loops_with_eps:
        if not check_road(loop,s,t):
            assert False, f"niepoprawna droga {loop.keys()}"

    check_intensity(current_edges, G, total_current, s, t)
```

Główna funkcja rozwiązujące zadanie za pomocą praw kirhofa

Funkcja `kirchhoff_solve` oblicza natężenia prądu na krawędziach grafu (G), zakładając, że źródło siły elektromotorycznej (SEM) o wartości `emf` jest podłączone między wierzchołkami (s) i (t).

1. Przypisywanie etykiet i wykrywanie cykli

Na początku wykonywane jest przeszukiwanie wszerz (BFS), które pełni trzy kluczowe role:

- **Etykietowanie krawędzi** — każdej krawędzi przypisywany jest unikalny numer.
- **Analiza I prawa Kirchhoffa** — dla każdego wierzchołka gromadzone są informacje o krawędziach z nim połączonych, co później pozwala na utworzenie równań bilansu prądów.

- **Wykrywanie cykli w grafie** — cykle są identyfikowane na podstawie wykrycia połączenia do już odwiedzonego wierzchołka. Dzięki zapisanym przodkom wierzchołków można łatwo odtworzyć pełny cykl.

Jeżeli został znaleziony cykl, jest on odtwarzany za pomocą funkcji `extract_path`, która zwraca słownik, który trzyma informacje o krawędziach w znalezionej ścieżce oraz oporze na tych krawędziach.

Zastosowana metoda gwarantuje znalezienie zbioru cykli, który pokrywa wszystkie krawędzie grafu, co prowadzi do utworzenia **nadokreślonego układu równań** z błędem numerycznym rzędu precyzji maszynowej.

2. Wyszukiwanie "oczek" zawierających SEM

"Oczka" z SEM są szukane w trakcie BFS. Jeżeli zostanie znaleziona pierwsza krawędź w wyszukiwaniu BFS prowadząca do t, odtwarzana jest ścieżka od t do s. Taka ścieżka wraz z SEM tworzą "Oczko"

3. Budowa układu równań

Na podstawie zebranych informacji konstruowana jest macierz układu równań obejmująca:

- **Równania wynikające z I prawa Kirchhoffa** (bilans prądów w węzłach),
- **Równania dla "oczek" zawierających wyłącznie rezystory**,
- **Równania dla "oczek" zawierających źródła SEM**.

Rozwiązanie tego układu dostarcza wartości natężeń prądów na poszczególnych krawędziach grafu.
Warto zaznaczyć, że kolumnami w tej macierzy są natężenia prądu na poszczegolnych krawędziach. Dlatego po rozwiązaniu układu otrzymujemy natężenia dla całego obwodu.

4. Testy poprawności

Po obliczeniach funkcja wykonuje zestaw testów weryfikacyjnych, które obejmują:

- **Sprawdzenie poprawności wykrytych cykli**,
- **Weryfikację spełnienia I prawa Kirchhoffa dla każdego wierzchołka**.

Dzięki tym krokom funkcja `kirchhoff_solve` zapewnia poprawne wyznaczenie rozkładu prądów w obwodzie elektrycznym reprezentowanym przez graf (G).

Macierz, którą rozwiązuje jest macierzą nadokreśloną, co oznacza, że ma za dużo wierszy. Stosuję funkcjie biblioteczną `np.linalg.lstsq(A,B)` do jej rozwiązania. Dodatkowo przyglądam się wartości błędu jaki zwraca ta funkcja. Jestem świadomy sposobu w jaki funkcja `np.linalg.lstsq(A,B)` rozwiązuje układ równań i etapów w jakim to wykonuje tj:

- Dekompozycja macierzy A na $U\Sigma V^T$
- Obliczanie wartości singularnych
- Rozwiązanie układu $x = V\Sigma^+ U^T b$
- Obliczenie błędu i rzędu macierzy

Złożoność tego algorytmu szacuję na O($VE + E^3$). VE z wyszukiwania cykli w tracie bfs oraz E^3 z rozwiązywania macierzy o wymiarach E na E

In [ ]:
```
def kirhof_solve(G,s,t,emf, print_inaccuracy = True, test= True):
    n = len(G)
    # zmieniam na macierz sasiedztwa
    matrix_graph = list_to_weighted_matrix(G)
    # dwa słowniki, bo chce numerować krawędzie w grafie
    edge_to_num = {}
    num_to_edge = {}
    # do tej macierzy będę umieszczać równania otrzymane z pierwszego prawa kirhofa
    sparse_matrix = []
    # lista cykli z SEM (będzie dokładnie jeden)
    loops_with_eps = []
    # zwykły setup dla bfs
    parents = [-1] * n
    Q = deque()
    Q.append(s)

    visited = set()
    visited.add(s)
    loops_only_resist = [] # tablica słowników krawędzi oczek bez SEM
    index = 1 # narastający licznik, którym przyporządkowuje kolejne krawedzie w trakcie przeszukiwania

    while len(Q)>0:
```

```
        u = Q.popleft()
        # tworze pusty wiersz dla pierwszego prawa kirhofa
        raw_in_matrix = {}

        for v,w in G[u]:
            # identyfikuje każdą krawędź za pomocą poniższej krotki
            edge_id = (min(u,v),max(u,v))
            # jeśli to nowa krawędź
            if edge_id not in edge_to_num:
                # nadaje numer
                edge_to_num[edge_id] = index
                num_to_edge[index] = edge_id
                index += 1
                # warunek, na znalezienie cyklu
                if parents[v] != -1:
                    edge_resist_dict = defaultdict(int)
                    # pobieram drogi po parentach jako słowniki z funkcji extract_path()
                    # zarówno dla sąsiada jak i aktualne wierzchołka
                    # w ten sposób otrzymuje cykl "z ogonem", ten ogon
                    # zeruje się samoistnie dzięki defaultdict(int) oraz
                    # odpowiednim dodawaniu i odejmowaniu
                    # (odwrócona kolejność dla v)
                    for d in (extract_path(matrix_graph, parents, u, False), extract_path(matrix_graph, parents, v, True)):
                        for key, value in d.items():
                            edge_resist_dict[key] += value
                    # rozpatruje krawędź łączącą u i v
                    edge_resist_dict[edge_id] = w
                    # trzymam się założenia o założonym kierunku prądu
                    if v<u:
                        edge_resist_dict[edge_id]*=-1
                    # dodaje do tablicy
                    loops_only_resist.append(edge_resist_dict)
                if v == t and t not in visited:
                    edge_resist_dict = extract_path(matrix_graph,parents,u,False)
                    # dodaje krawędź łącząco ścieżke do wierzchołka u z t
                    edge_resist_dict[edge_id] = w
                    # jeżeli idziemy przeciwnie do założonego kierunku prądu to zmieniamy
                    if v < u:
                        edge_resist_dict[edge_id] *= -1
                    loops_with_eps.append(edge_resist_dict)
                # zwykły warunek z bfs
```

```
            if v not in visited:
                visited.add(v)
                parents[v] = u
                Q.append(v)
            # korekta kierunku prądu. Trzymam się założonego kierunku
            if u < v:
                raw_in_matrix[edge_to_num[edge_id]] = -1
            else:
                raw_in_matrix[edge_to_num[edge_id]] = 1
        #polączenie do SEM wierzchołków s i t
        # tyczy się to I prawa kirhofa
        if u==s:
            raw_in_matrix[0] = 1
        if u==t:
            raw_in_matrix[0] = -1
        # dodaje do tablicy
        sparse_matrix.append(raw_in_matrix)


    # buduje graf na podstawie I prawa kirhofa, cykli z SEM oraz bez SEM
    matrix, B = build_matrix(emf, edge_to_num, index, loops_only_resist, sparse_matrix, loops_with_eps)

    # rozwiązuje macierz otrzymując natężenia prądów
    # Uwaga: otrzymany układ jest nadokreślony. Korzystam z funkcji bibliotecznej ze świadomością jej działania

    currents, inaccuracy, _, _ = np.linalg.lstsq(matrix,B)
    # sprawdzam poprawność macierzy ("czy jest bliska układowi oznaczonemu")
    if print_inaccuracy:
        print(f"wymiary matrixy: {len(B)}:{len(matrix[0])}")
        print(f"bład z metody najmniejszych kwadratów wynosi: {inaccuracy}")
    if inaccuracy:
        assert inaccuracy[0]< 10**(-16)
    # końcowo słownik, gdzie kluczami są krawędzie, a wartościami natężenia
    current_edges = {}
    for i,current in enumerate(currents):
        if i==0: continue
        current_edges[num_to_edge[i]] = current

    # to natężenie płynące przez SEM
    total_current = currents[0]
    # sprawdzam poprawność
```

```
    if test:
        test_kirhof_solution(loops_only_resist, loops_with_eps, current_edges, G, total_current, s, t)
    # i zwracam
    return current_edges, total_current
```

Metoda potencjałów wezłowych

Ta metoda charakteryzuje się dużo prostszą implementacją od poprzedniej.

Przyjmuję założenie, że wierzchołek *s* ma potencjał równy SEM, a wierzchołek *t* potencjał równy 0.

Funkcja `find_potential_of_vertix(G,s,t,emf)` wykonuje bfs'a i rozpatruje każdy wierzchołek pod kątem przekształconego I prawa kirhofa. Funkcja buduje macierz, ktorej kolumnami są potencjały odpowiednich wierzchołków.

In [16]:
```
def find_potential_of_vertix(G,s,t,emf):
    n = len(G)
    # tworze tabele wypełnioną zerami
    matrix = np.zeros((n, n), dtype=float)
    Q = deque()
    visited = [False]*n
    visited[s] = True
    # dodaje sąsiadów s, bo mam dany potencjał s
    for v,_ in G[s]:
        Q.append(v)
        visited[v]=True
    # tworze tablice B
    B = np.zeros(n,dtype=float)

    while len(Q)>0:
        u = Q.popleft()
        # znamy potencjał t wiec go pomijam
        if u == t: continue
        for v,w in G[u]:
            # dodawanie do kolejki
            if not visited[v]:
                visited[v] = True
```

```
                Q.append(v)
            # dodawanie wartości do kolejki
            matrix[u][u] += 1/w
            # potencjał t jest zerowy
            if v!=s and v!=t:
                matrix[u][v] -= 1/w
            # potencjał s jest równy emf
            elif v==s:
                B[u] += emf/w
    # dodaje do macierzy co wiem, dla lepszej struktury
    matrix[s][s]=1.0
    B[s] = emf
    matrix[t][t] = 1.0
    B[t] = 0.0
    # zwracam macierze
    return matrix, B
```

Funkcja `nodal_potential_method` implementuje metodę potencjałów węzłowych.

Na początku algorytmu przeprowadzam przeszukiwanie BFS, które iterując po wierzchołkach, tworzy macierz, gdzie kolumny odpowiadają potencjałom węzłów. Numeracja węzłów jest zgodna z kolejnością podaną na wejściu.

Po rozwiązaniu układu równań uzyskuję wartości potencjałów dla każdego wierzchołka. Następnie ponownie wykonuję BFS, tym razem obliczając natężenia prądów między węzłami na podstawie wyznaczonych potencjałów.

Na końcu przeprowadzam testy poprawności rozwiązania, weryfikując zgodność z pierwszym prawem Kirchhoffa.

Złożoność tego algorytmu szacuje na o(V^3)

In [ ]:
```
def nodal_potential_method(G,s,t,emf,test = True):
    # tworze i rozwiązuje układ równań otrzymując potencjały wierzchołków
    matrix, B = find_potential_of_vertix(G,s,t,emf)
    potentials,_,_,_ = np.linalg.lstsq(matrix,B)

    # wykonuje bfs obliczając natężenia
    Q = deque()
    Q.append(s)
    visited = set()
    visited.add(s)
```

```
    # słownik mapujący krawędź na natężenie
    current_edges = {}
    while len(Q)>0:
        u = Q.popleft()
        for v,w in G[u]:
            frs = min(u,v)
            sec = max(u,v)
            edge_id = (frs,sec)
            # jeżeli nie obliczaliśmy tej krawędzi
            if edge_id not in current_edges:
                current_edges[edge_id] = (potentials[frs] - potentials[sec])/w
            # dodawanie do kolejki
            if v not in visited:
                visited.add(v)
                Q.append(v)

    # obliczam natężenie na SEM za pomocą I prawa kirhofa liczonego na węźle t
    total_current = 0
    for v,_ in G[t]:
        if v<t:
            total_current += current_edges[min(v,t),max(v,t)]
        else:
            total_current -= current_edges[min(v,t),max(v,t)]
    # sprawdzam poprwaność
    if test:
        check_intensity(current_edges,G,total_current, s,t)
    return current_edges, total_current
```

Budowanie grafu ze słownika wierzchołków oraz wizualizacja wyników

Funkcja `build_graph(edge_weights)` buduje graf na podstawie słowniku krawędzi na natężenia. Jeżeli wartość natężenia jest ujemna, zmieniam kierunek prądu.

Funkcja `draw_directed_graph` rysuje graf skierowany reprezentujący natężenia w obwodzie. Zaznaczone są dwa wierzchołki do których podpieliśmy SEM

In [95]:
```
def build_graph(edge_currents):
    graph = nx.DiGraph()
```

```
    for (node_u, node_v), current in edge_currents.items():
        # wartość ujemna, zminiam kierunek skierowania grafu
        if current < 0:
            graph.add_edge(node_v, node_u, current=-current)
        else:
            graph.add_edge(node_u, node_v, current=current)

    return graph


def visualize_electrical_circuit(G, source, target, total_curent, voltage, name = "", figsize=(12, 8),
                                 layout='kamada_kawai', seed=42):
    """
    Visualize an electrical circuit graph with continuous color gradient for currents.
    """
    # Validate and normalize current intensities
    edge_currents = nx.get_edge_attributes(G, 'current')
    currents = list(edge_currents.values())

    if not currents:
        raise ValueError("Graph must have edge currents representing current")

    # Advanced layout selection
    def choose_layout(G, layout):
        np.random.seed(seed)

        layouts = {
            'spring': nx.spring_layout,
            'circular': nx.circular_layout,
            'kamada_kawai': nx.kamada_kawai_layout,
            'spectral': nx.spectral_layout,
            'planar': nx.planar_layout,
            'shell': nx.shell_layout,
            'community': lambda g: nx.spring_layout(g, k=0.5),
            'hierarchical': lambda g: nx.spring_layout(g, k=0.9, iterations=50),
            'radial': lambda g: nx.spring_layout(g, k=2, iterations=20)
        }
        if 'Graf siatka 2D' in name.lower():
            # Wymuszenie układu siatki 2D
            rows = int(np.sqrt(len(G.nodes())))
```

```
            cols = len(G.nodes()) // rows
            return {node: (node % cols, node // cols) for node in G.nodes()}
        if 'cykl' in name.lower():
            # Równomierne rozmieszczenie wierzchołków w okręgu w kolejności indeksów
            num_nodes = len(G.nodes())
            return {node: (np.cos(2 * np.pi * i / num_nodes), np.sin(2 * np.pi * i / num_nodes))
                    for i, node in enumerate(sorted(G.nodes()))}
        if "Graf 3-regularny (kubiczny)" in name:
            return layouts['spring'](G)
        return layouts[layout](G)

    # Compute layout
    pos = choose_layout(G, layout)

    # Normalize currents for color mapping
    min_current, max_current = min(currents), max(currents)

    # Create figure with space for colorbar
    fig, (ax, cax) = plt.subplots(
        nrows=1,
        ncols=2,
        figsize=figsize,
        gridspec_kw={'width_ratios': [4, 0.2]}
    )

    # Create a continuous color map
    colors = [
            (0.8, 0.8, 1),   # Jasny szary - najniższe natężenie
            (0.4, 0.7, 0.9),    # Jasnoniebieski
            (0.6, 0.8, 0.6),    # Miętowy zielony
            (0.9, 0.9, 0.5),    # Jasnożółty
            (0.95, 0.7, 0.3),   # Jasnopomarańczowy
            (0.8, 0.2, 0.2)     # Czerwony - najwyższe natężenie
        ]
    cmap = LinearSegmentedColormap.from_list('improved_cmap', colors)

    if np.isclose(min_current, max_current):
        min_current = 0

    norm = mcolors.Normalize(vmin=min_current, vmax=max_current)
```

```
    # Draw nodes
    nx.draw_networkx_nodes(G, pos, node_color='lightgray', node_size=300, ax=ax)

    # Highlight source and target nodes
    nx.draw_networkx_nodes(G, pos, nodelist=[source], node_color='red', node_size=500, ax=ax)
    nx.draw_networkx_nodes(G, pos, nodelist=[target], node_color='green', node_size=500, ax=ax)

    # Draw edges with continuous color gradient
    for (u, v, current) in G.edges(data='current'):
        nx.draw_networkx_edges(G, pos,
                               edgelist=[(u, v)],
                               edge_color=[current],
                               width=0.7 +3*current/max_current,
                               edge_cmap=cmap,
                               edge_vmin=min_current,
                               edge_vmax=max_current,
                               ax=ax)

    # Add node labels if graph is small (less than 20 nodes)
    nx.draw_networkx_labels(G, pos, ax=ax)
    if len(G.nodes()) < 20:


        # Add edge currents for small graphs
        edge_labels = {(u, v): f'{current:.2f}' for (u, v, current) in G.edges(data='current')}
        nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_size=8, ax=ax)

    # Create colorbar
    sm = plt.cm.ScalarMappable(cmap=cmap, norm=norm)
    sm.set_array([])
    plt.colorbar(sm, cax=cax, label='Natężenie prądu')

    # Title and additional information
    ax.set_title(f'{name}\n źródło: {source}, ujście: {target}, napięcie SEM: {voltage:.2f}V \n Natężenie na SEM {total_curent
    ax.axis('off')

    info_text = f"Liczba węzłów: {len(G.nodes())}\nLiczba krawędzi: {len(G.edges())}\n" \

    plt.figtext(0.02, 0.02, info_text, fontsize=12, bbox=dict(facecolor='white', alpha=0.7))
```

```
    plt.tight_layout()
    plt.show()
```

Test wydajności i poprawności

Dla wiarygodnego porównania szybkości obu podejść użyłem tego samego algorytmu rozwiązującego układ równań w obu przypadkach np.linalg.lstsq()

Zbiór testowy

Wygenerowałem różnorodne zbiory testowe o różnej liczbie wierzchołków i krawędzi dla gruntowengo przetestowania obu algorytmów

In [ ]:
```
nums = [25, 50, 75, 100, 125, 150]

for n in nums:
    generate_random_graph(f"graphs_test/random{n}.txt", n, "random", 30, False)
    generate_random_graph(f"graphs_test/bridge{n}.txt", n, "bridge", 30, False)
    generate_random_graph(f"graphs_test/grid{n}.txt", n, "grid", 30, False)
    generate_random_graph(f"graphs_test/small_world{n}.txt", n, "small_world", 30, False)
```

Test poprawności

In [ ]:
```
def test_correctness_generated_graphs(nums):
    # DataFrame na wyniki
    results = []

    # Iteracja przez wygenerowane pliki
    for n in nums:
        for graph_type in ["random", "bridge", "grid", "small_world"]:
            file_name = f"graphs_test/{graph_type}{n}.txt"

            # Załaduj graf
            G, s, t, emf = load_graph_from_file(file_name)

            # Pobieranie wyników z kirhof_solve
```

```
            current_edges_kirhof, total_current1 = kirhof_solve(G, s, t, emf,print_inaccuracy=False)

            # Pobieranie wyników z nodal_potential_method
            current_edges_potential, total_current2 = nodal_potential_method(G, s, t, emf)

            # liczenie błędu wyników
            maxi_err = -float("inf")
            are_consistent = np.isclose(total_current1,total_current2)
            for key in current_edges_kirhof.keys():
                are_consistent = are_consistent and np.isclose(current_edges_kirhof[key],current_edges_potential[key])
                maxi_err = max(maxi_err, abs(current_edges_kirhof[key]-current_edges_potential[key]))
            # Zapisz wyniki w DataFrame
            results.append({
                "typ grafu": graph_type,
                "ilość węzłów": n,
                "maksymalny błąd": maxi_err ,
                "zgodność": are_consistent
            })

    # Tworzymy DataFrame z wynikami
    df_results = pd.DataFrame(results)
    return df_results

# Wygenerowanie wyników dla różnych rozmiarów grafów
df = test_correctness_generated_graphs(nums)

# Zobacz wyniki
display(df)
```

	typ grafu	ilość węzłów	maksymalny błąd	zgodność
0	random	25	1.145750e-13	True
1	bridge	25	4.174439e-14	True
2	grid	25	3.752554e-14	True
3	small_world	25	7.299456e-14	True
4	random	50	2.029557e-13	True
5	bridge	50	1.489919e-13	True
6	grid	50	5.762057e-14	True
7	small_world	50	2.314815e-14	True
8	random	75	1.532940e-13	True
9	bridge	75	8.482104e-14	True
10	grid	75	3.574918e-14	True
11	small_world	75	5.606626e-14	True
12	random	100	9.513501e-13	True
13	bridge	100	1.090239e-13	True
14	grid	100	3.239076e-14	True
15	small_world	100	4.440892e-14	True
16	random	125	1.093153e-12	True
17	bridge	125	9.272791e-14	True
18	grid	125	7.582823e-14	True
19	small_world	125	8.459899e-14	True
20	random	150	9.295871e-13	True
21	bridge	150	1.099459e-13	True

	typ grafu	ilość węzłów	maksymalny błąd	zgodność
22	grid	150	4.707346e-14	True
23	small_world	150	1.235123e-13	True

Dla wszystkich wygenerowanych grafów wyniki obu algorytmów są zgodne. Z testami opartymi o I prawo Kirhofa algorytmy również sobie poradziły

Test szybkości

In [ ]:
```
def test_algorithms_on_generated_graphs(nums):
    # DataFrame na wyniki
    results = []

    # Iteracja przez wygenerowane pliki
    for n in nums:
        for graph_type in ["random", "bridge", "grid", "small_world"]:
            file_name = f"graphs_test/{graph_type}{n}.txt"

            # Załaduj graf
            G, s, t, emf = load_graph_from_file(file_name)

            # Pomiar czasu dla kirhof_solve
            start_time = time.time()
            res_kir,_ = kirhof_solve(G, s, t, emf, False, False)
            end_time = time.time()
            kirhof_time = end_time - start_time

            # Pomiar czasu dla nodal_potential_method
            start_time = time.time()
            _,_ = nodal_potential_method(G, s, t, emf,False)
            end_time = time.time()
            nodal_potential_time = end_time - start_time

            # Zapisz wyniki w DataFrame
            results.append({
                "typ grafu": graph_type,
```

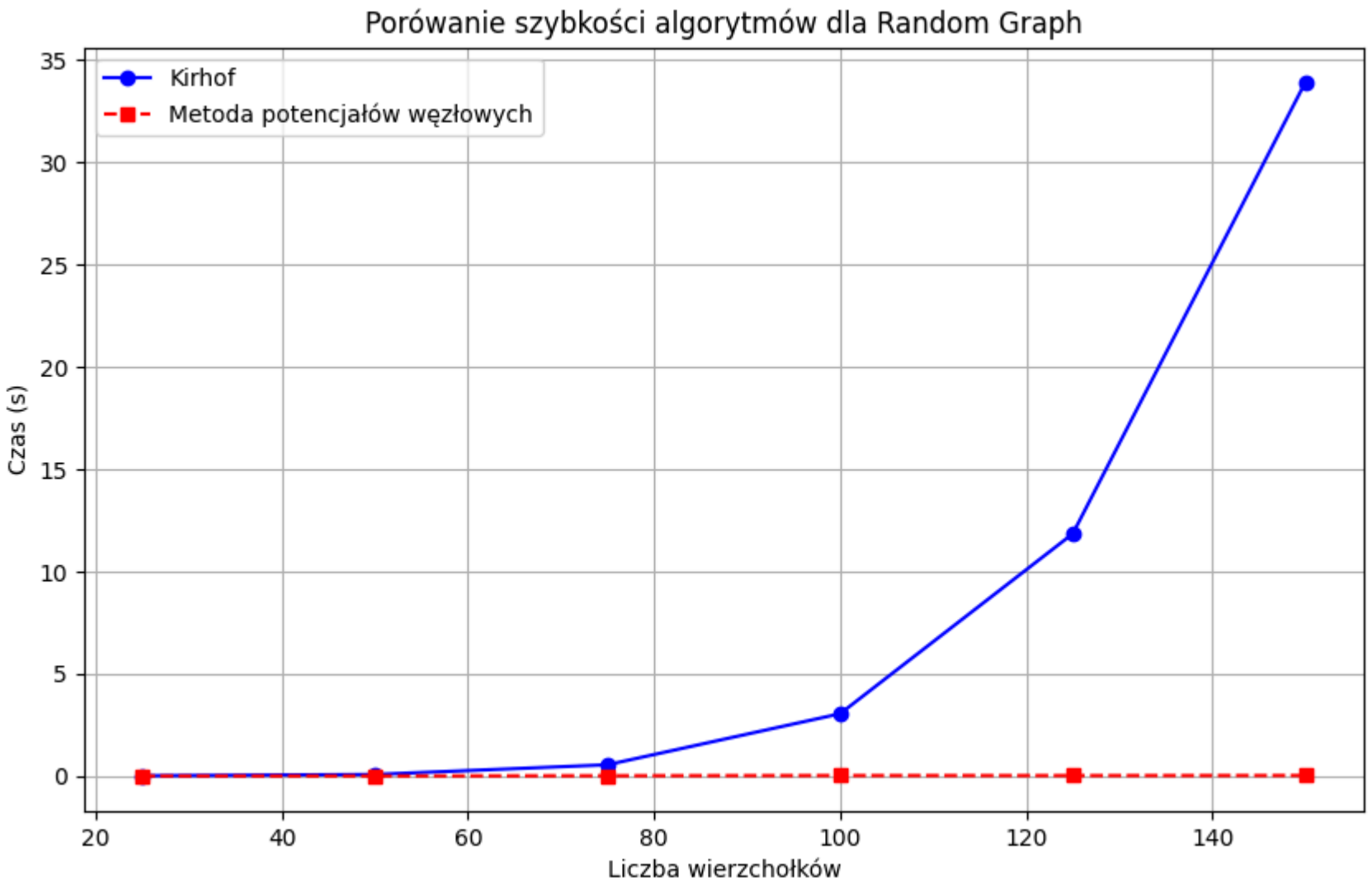
```
                "ilość węzłów": n,
                "ilość krawędzi": len(res_kir),
                "czas_kirhof": kirhof_time,
                "czas_potencjały_węzłowe": nodal_potential_time
            })

    # Tworzymy DataFrame z wynikami
    df_results = pd.DataFrame(results)
    return df_results

# Wygenerowanie wyników dla różnych rozmiarów grafów
df = test_algorithms_on_generated_graphs(nums)

display(df)
```

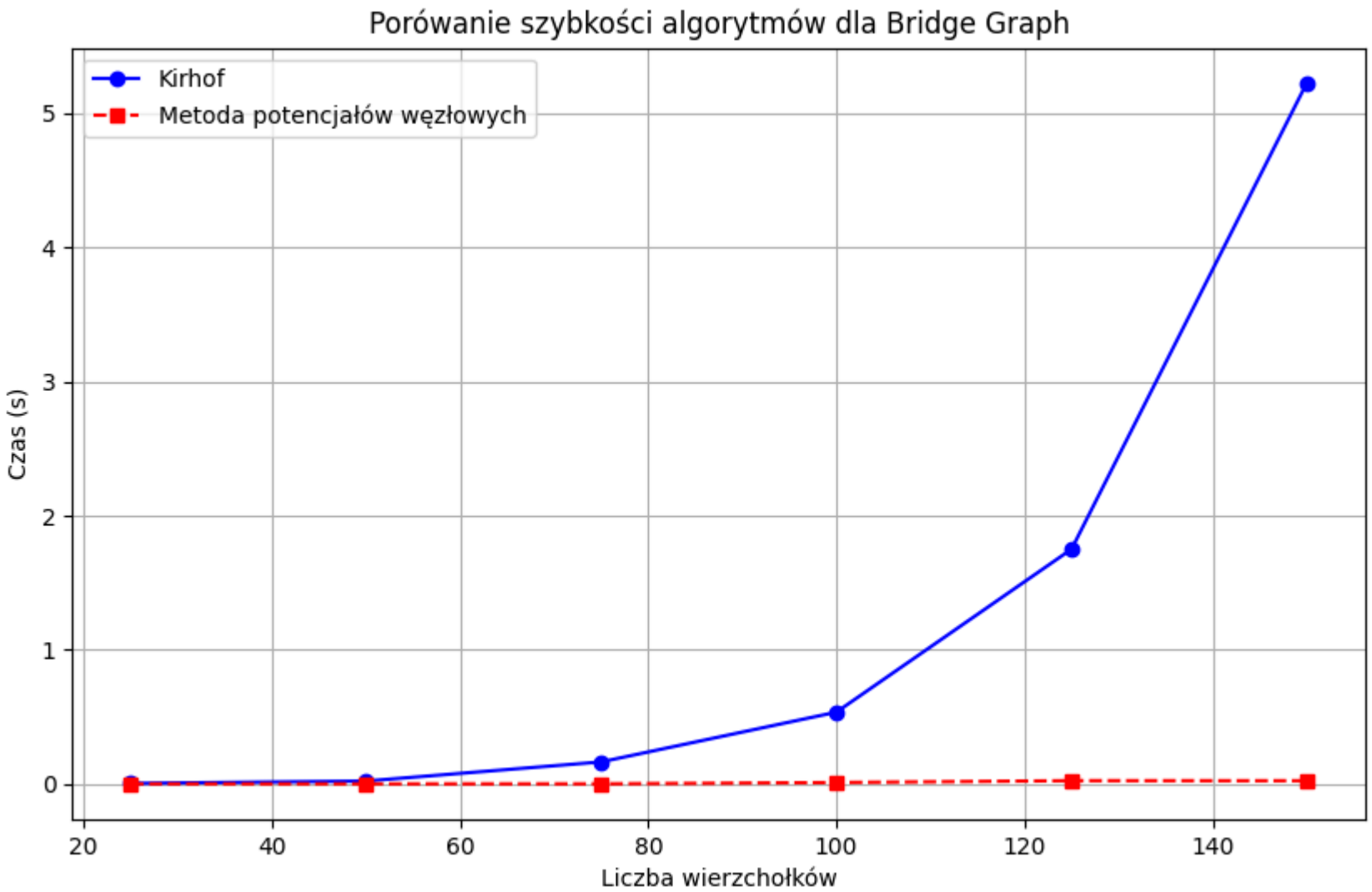
	typ grafu	ilość węzłów	ilość krawędzi	czas_kirhof	czas_potencjały_węzłowe
0	random	25	134	0.011239	0.000000
1	bridge	25	85	0.003665	0.000000
2	grid	25	40	0.001000	0.000000
3	small_world	25	125	0.010506	0.001040
4	random	50	614	0.073124	0.000000
5	bridge	50	312	0.021080	0.000000
6	grid	50	84	0.006951	0.000000
7	small_world	50	250	0.021453	0.000000
8	random	75	1381	0.547750	0.009628
9	bridge	75	774	0.163768	0.000000
10	grid	75	112	0.009781	0.004308
11	small_world	75	375	0.030437	0.000000
12	random	100	2501	3.043290	0.021599
13	bridge	100	1377	0.534682	0.010028
14	grid	100	180	0.012512	0.000000
15	small_world	100	500	0.056171	0.002212
16	random	125	3884	11.845817	0.019110
17	bridge	125	2126	1.750282	0.023953
18	grid	125	220	0.025285	0.013643
19	small_world	125	625	0.095609	0.009031
20	random	150	5560	33.927759	0.023739
21	bridge	150	3005	5.225389	0.023395

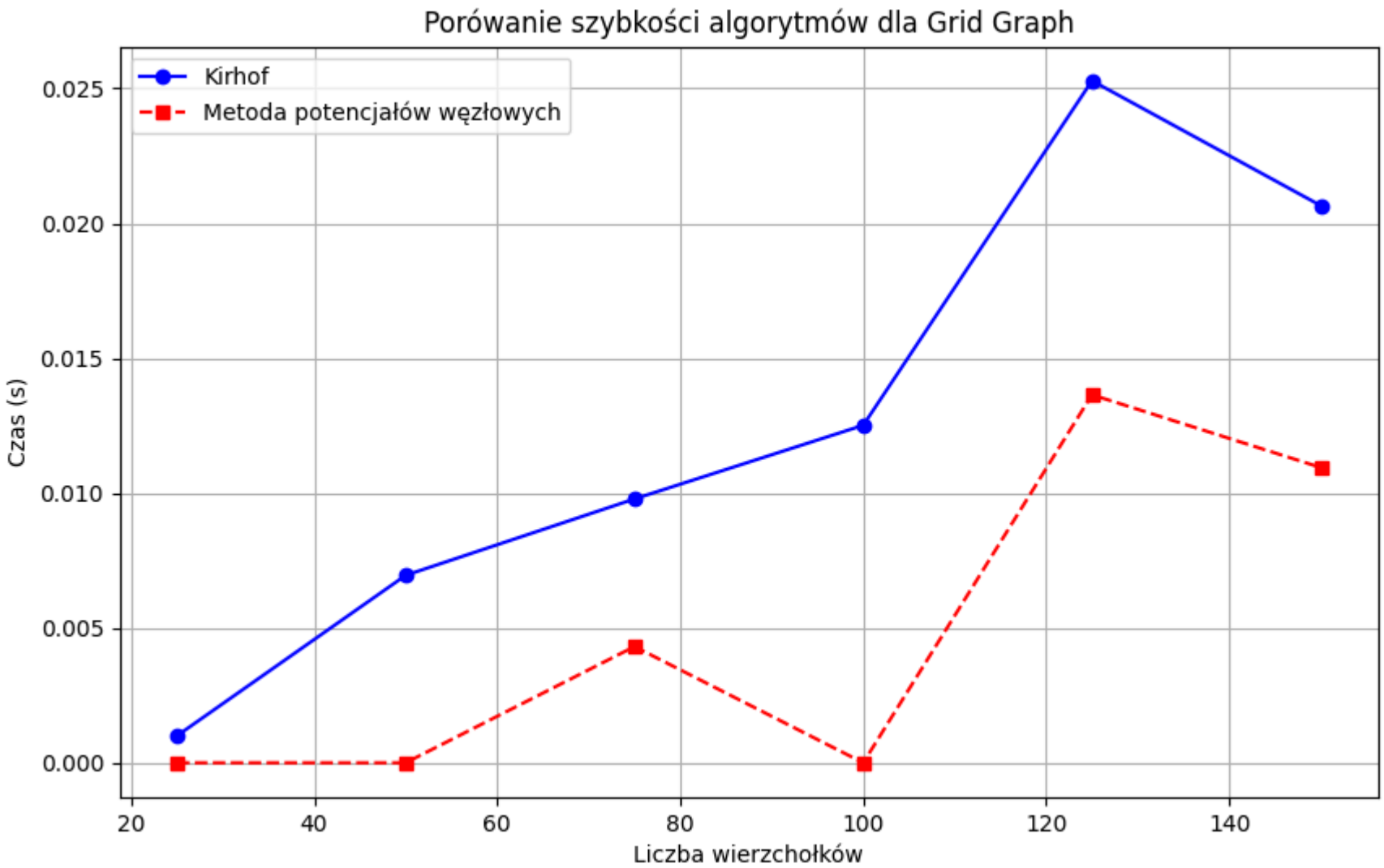
	typ grafu	ilość węzłów	ilość krawędzi	czas_kirhof	czas_potencjały_węzłowe
22	grid	150	264	0.020656	0.010947
23	small_world	150	750	0.155673	0.015108

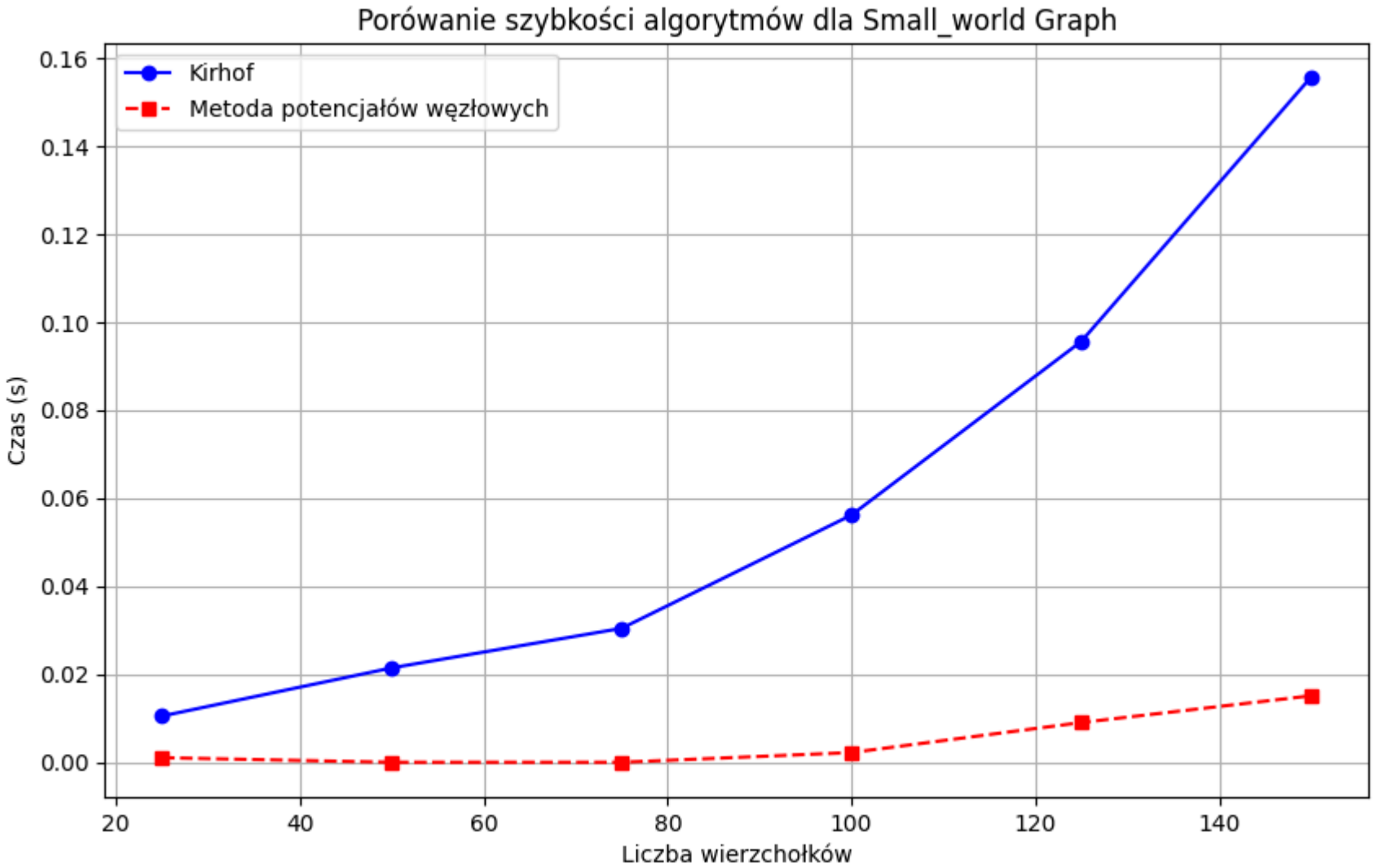
Wizualizacja wyników

In [ ]:
```
def plot_comparison(df):
    graph_types = df['typ grafu'].unique()

    # Tworzenie wykresów dla każdego typu grafu
    for graph_type in graph_types:
        plt.figure(figsize=(10, 6))

        # Filtruj dane dla konkretnego typu grafu
        graph_data = df[df['typ grafu'] == graph_type]

        # Wykres porównujący czasy dla obu algorytmów
        plt.plot(graph_data['ilość węzłów'], graph_data['czas_kirhof'], label='Kirhof', marker='o', linestyle='-', color='blue
        plt.plot(graph_data['ilość węzłów'], graph_data['czas_potencjały_węzłowe'], label='Metoda potencjałów węzłowych', mark

        # Dodanie tytułu, etykiet i legendy
        plt.title(f"Porównanie szybkości algorytmów dla {graph_type.capitalize()} Graph")
        plt.xlabel('Liczba wierzchołków')
        plt.ylabel('Czas (s)')
        plt.legend()
        plt.grid(True)

        # Wyświetlenie wykresu
        plt.show()

# Załóżmy, że `df` to DataFrame z wynikami
plot_comparison(df)
```


Porównanie szybkości algorytmów dla Random Graph
Kirhof
Metoda potencjałów węzłowych
Czas (s)
Liczba wierzchołków
0
5
10
15
20
25
30
35
20
40
60
80
100
120
140


Porównanie szybkości algorytmów dla Bridge Graph
Kirhof
Metoda potencjałów węzłowych
Czas (s)
Liczba wierzchołków
0
1
2
3
4
5
20
40
60
80
100
120
140


Porównanie szybkości algorytmów dla Grid Graph
Kirhof
Metoda potencjałów węzłowych
Czas (s)
0.025
0.020
0.015
0.010
0.005
0.000
20
40
60
80
100
120
140
Liczba wierzchołków



Z analizy wyników wynika, że metoda potencjałów węzłowych jest znacznie szybsza niż metoda oparta na prawach Kirchhoffa. Jedynie dla grafu typu Grid obserwujemy stosunkowo stałą proporcjonalność między czasem wykonania obu algorytmów, podczas gdy dla pozostałych typów grafów zależność ta ma charakter kwadratowy lub wyższy.

Główna część czasu w algorytmie opartym na prawach Kirchhoffa jest poświęcona rozwiązaniu układu równań, którego rozmiar może być bardzo duży. Wielkość tego układu zależy od liczby krawędzi w grafie, co oznacza, że w przypadku grafów o dużej gęstości (gdzie liczba

krawędzi jest ponadliniowa), złożoność obliczeń macierzy staje się dominującym czynnikiem. Złożoność obliczeniowa tej operacji to O(n^3), co wyraźnie wpływa na czas wykonania w przypadku dużych i gęstych grafów.

Wizualizacje dla różnych grafów

Testy jednostkowe dla wybranych grafów

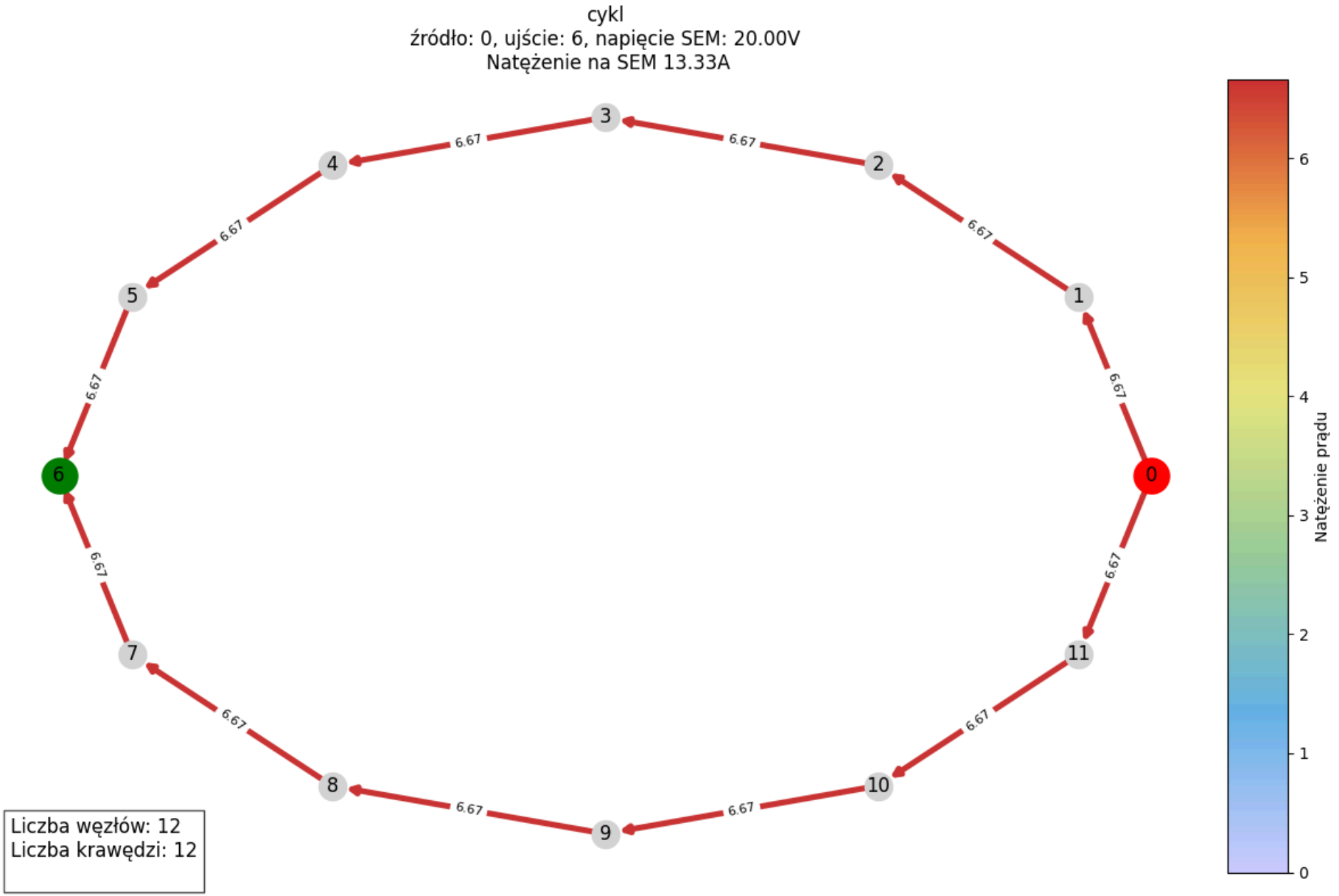
In [ ]:
```
def solve_from_file(file_name, name=""):
    print(name)
    G, s, t, emf = load_graph_from_file(file_name)
    current_edges, total_current =kirhof_solve(G, s, t, emf, False)
    print("wyliczone natężenia:")
    print(current_edges)
    visualize_electrical_circuit(build_graph(current_edges), s,t, total_current, emf, name=name)
```

W celu klarownej wizualizacji wyników algorytmów przygotowałem specjalne rodzaje grafów

Grafy cykli

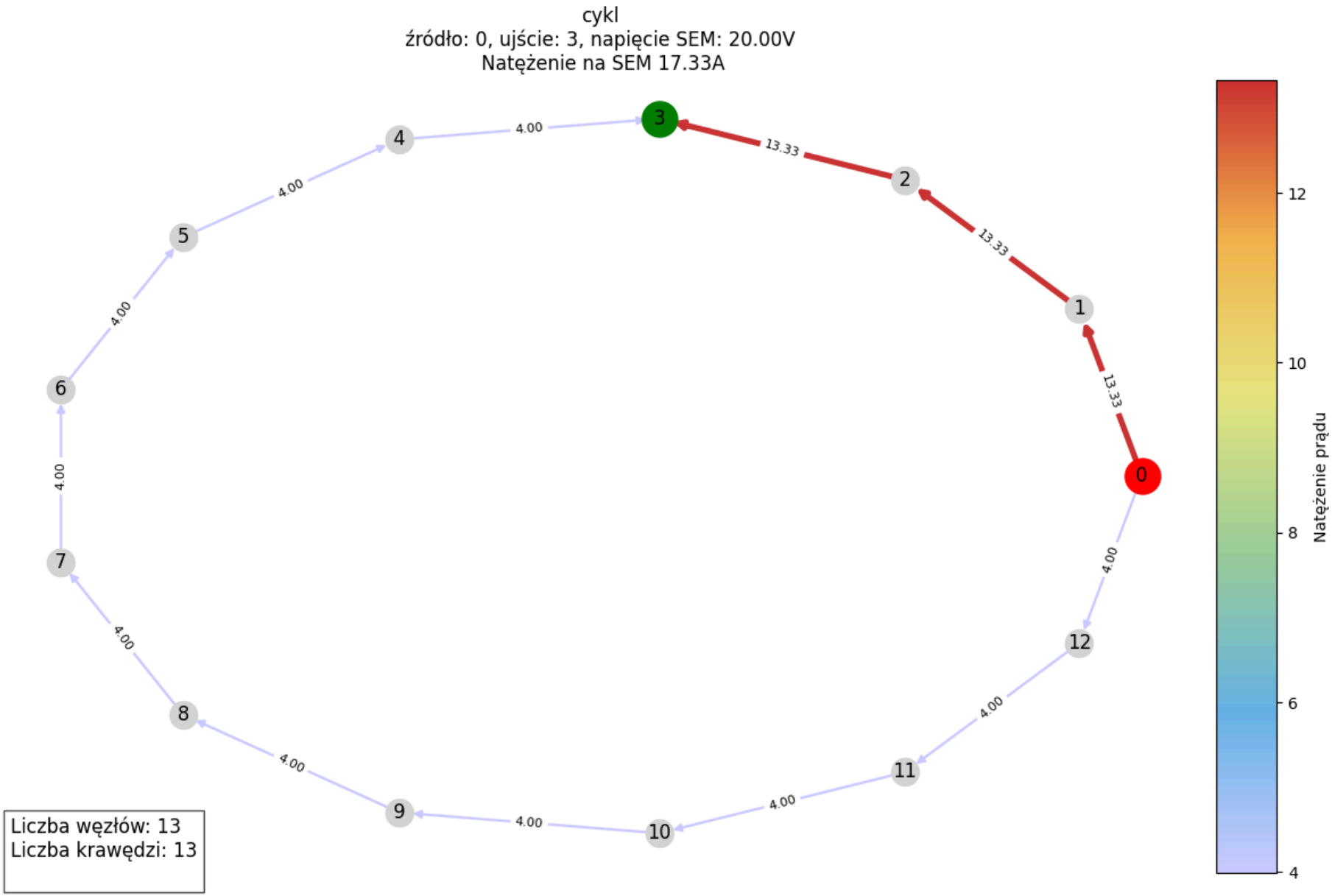
In [98]:
```
solve_from_file("graphs/cycle.txt","cykl")
```

```
cykl
wyliczone natężenia:
{(0, 1): np.float64(6.666666666666671), (0, 11): np.float64(6.666666666666665), (1, 2): np.float64(6.666666666666671), (10, 1
1): np.float64(-6.6666666666666705), (2, 3): np.float64(6.666666666666671), (9, 10): np.float64(-6.666666666666673), (3, 4): n
p.float64(6.666666666666671), (8, 9): np.float64(-6.666666666666673), (4, 5): np.float64(6.66666666666667), (7, 8): np.float64
(-6.66666666666667), (5, 6): np.float64(6.66666666666667), (6, 7): np.float64(-6.66666666666667)}
```



In [99]:
```
solve_from_file("graphs/cycle2.txt","cykl")
```

```
cykl
wyliczone natężenia:
{(0, 1): np.float64(13.333333333333329), (0, 12): np.float64(3.9999999999999964), (1, 2): np.float64(13.33333333333333), (11, 1
2): np.float64(-3.999999999999998), (2, 3): np.float64(13.333333333333332), (10, 11): np.float64(-3.999999999999997), (3, 4): n
p.float64(-3.9999999999999893), (9, 10): np.float64(-4.000000000000002), (4, 5): np.float64(-3.999999999999992), (8, 9): np.flo
at64(-4.0000000000000036), (5, 6): np.float64(-3.9999999999999973), (7, 8): np.float64(-4.0000000000000036), (6, 7): np.float64
(-4.0)}
```

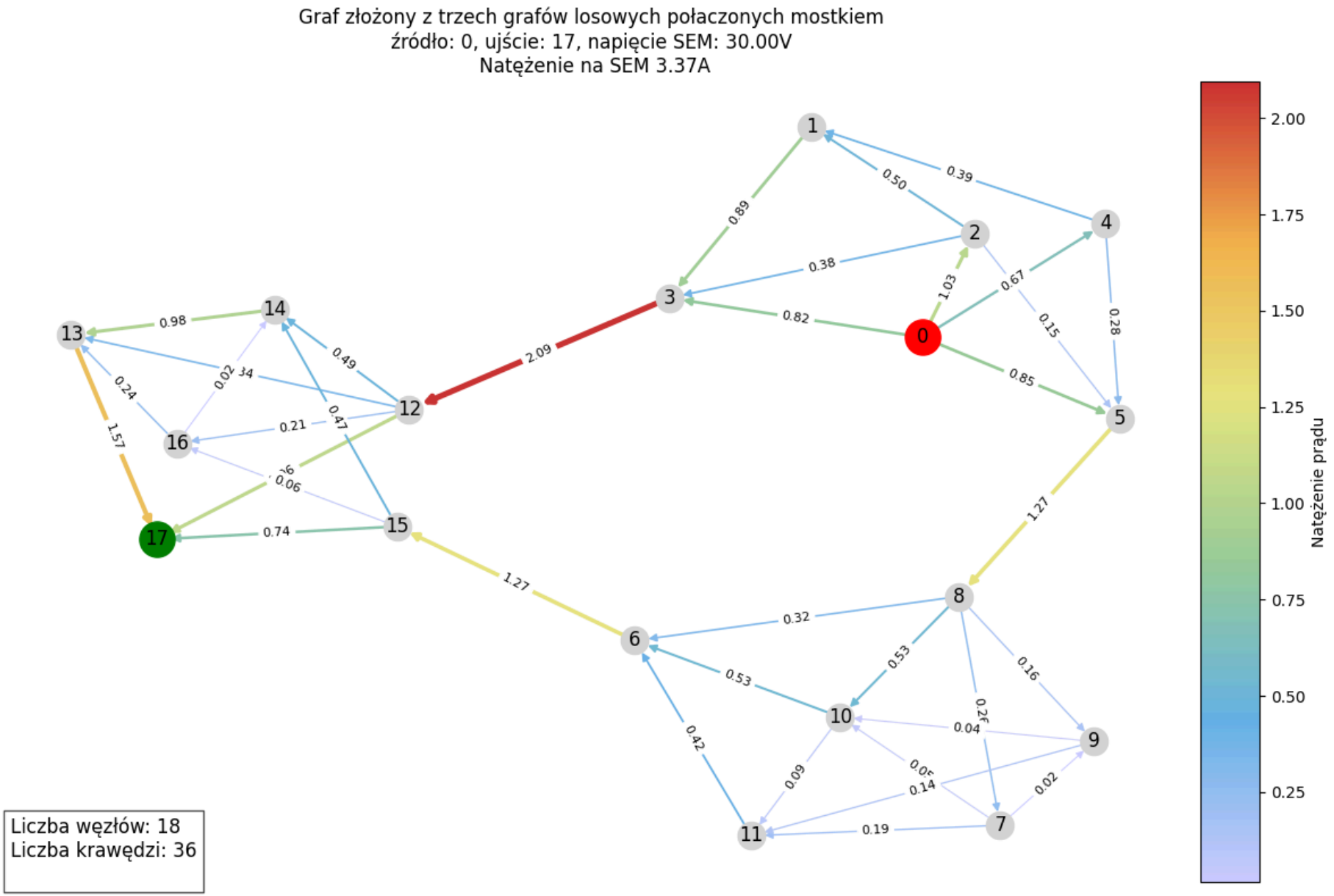


Grafy mostkowe

Graf z 3-ema mostami

In [138…
```
solve_from_file("graphs/bridge3.txt", "Graf złożony z trzech grafów losowych połaczonych mostkiem")
```

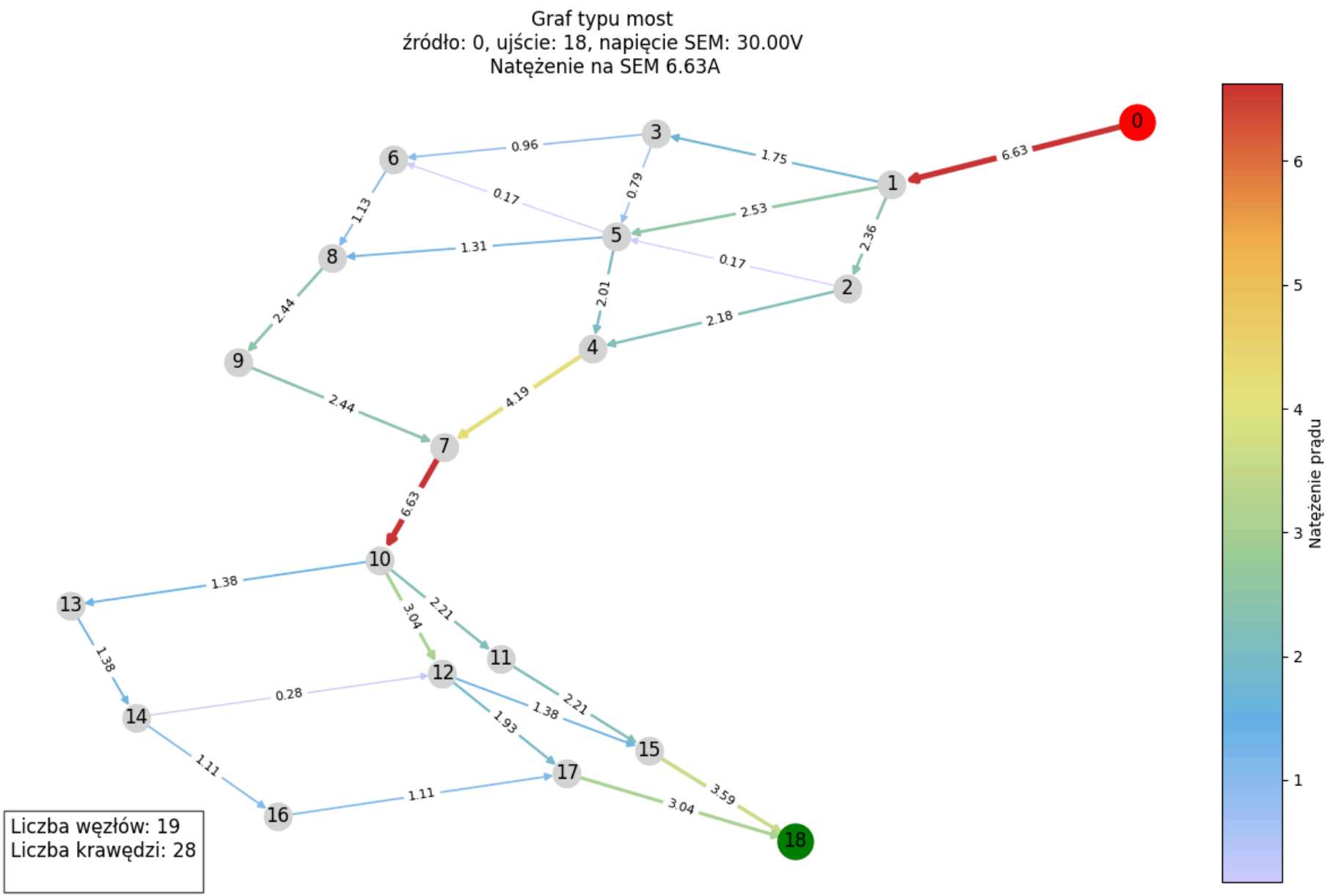
```
Graf złożony z trzech grafów losowych połaczonych mostkiem
wyliczone natężenia:
{(0, 2): np.float64(1.0264171102754944), (0, 3): np.float64(0.8201062973548324), (0, 4): np.float64(0.6707112641146642), (0,
5): np.float64(0.8481204987715723), (1, 2): np.float64(-0.5001308318730667), (2, 3): np.float64(0.38098212876638005), (2, 5): n
p.float64(0.14530414963604968), (1, 3): np.float64(0.8937630906118517), (3, 12): np.float64(2.0948515167330637), (1, 4): np.flo
at64(-0.39363225873878305), (4, 5): np.float64(0.27707900537587926), (5, 8): np.float64(1.270503653783488), (12, 13): np.float6
4(0.3407967607729789), (12, 14): np.float64(0.48582550241535577), (12, 16): np.float64(0.2072083842407827), (12, 17): np.float6
4(1.0610208693039545), (6, 8): np.float64(-0.32294210570160437), (7, 8): np.float64(-0.2557943798998191), (8, 9): np.float64(0.
16153224133148672), (8, 10): np.float64(0.5302349268505815), (13, 14): np.float64(-0.9803440609138109), (13, 16): np.float64(-
0.24472751799855233), (13, 17): np.float64(1.5658683396853474), (14, 15): np.float64(-0.47432001726759804), (14, 16): np.float6
4(-0.02019854123085813), (15, 16): np.float64(0.057717674988632325), (15, 17): np.float64(0.738465961527258), (6, 10): np.float
64(-0.531252461856779), (6, 11): np.float64(-0.4163090862251054), (6, 15): np.float64(1.270503653783488), (7, 9): np.float64(0.
01657097101544572), (7, 10): np.float64(0.049045492294726926), (7, 11): np.float64(0.1901779165896461), (9, 10): np.float64(0.0
39918865935284334), (9, 11): np.float64(0.13818434641164773), (10, 11): np.float64(0.0879468232238126)}
```



Graf z jednym mostem

```
In [101…  solve_from_file("graphs/most.txt", "Graf typu most")
```

```
Graf typu most
wyliczone natężenia:
{(0, 1): np.float64(6.631119728550654), (1, 2): np.float64(2.355792535143002), (1, 5): np.float64(2.5302956858943317), (1, 3):
np.float64(1.7450315075133185), (2, 5): np.float64(0.17450315075132938), (2, 4): np.float64(2.1812893843916625), (3, 5): np.flo
at64(0.7852641783809903), (4, 5): np.float64(-2.0067862336403297), (5, 6): np.float64(0.17450315075132147), (5, 8): np.float64
(1.3087736306349993), (3, 6): np.float64(0.9597673291323289), (4, 7): np.float64(4.188075618031994), (6, 8): np.float64(1.13427
04798836596), (8, 9): np.float64(2.4430441105186596), (7, 9): np.float64(-2.443044110518656), (7, 10): np.float64(6.63111972855
0652), (10, 11): np.float64(2.2103732428502223), (10, 12): np.float64(3.0392632089190474), (10, 13): np.float64(1.3814832767813
932), (11, 15): np.float64(2.2103732428502143), (12, 15): np.float64(1.381483276781384), (12, 14): np.float64(-0.27629665535627
096), (12, 17): np.float64(1.9340765874939374), (13, 14): np.float64(1.3814832767813896), (15, 18): np.float64(3.59185651963160
04), (14, 16): np.float64(1.1051866214251087), (17, 18): np.float64(3.039263208919054), (16, 17): np.float64(1.105186621425115
8)}
```

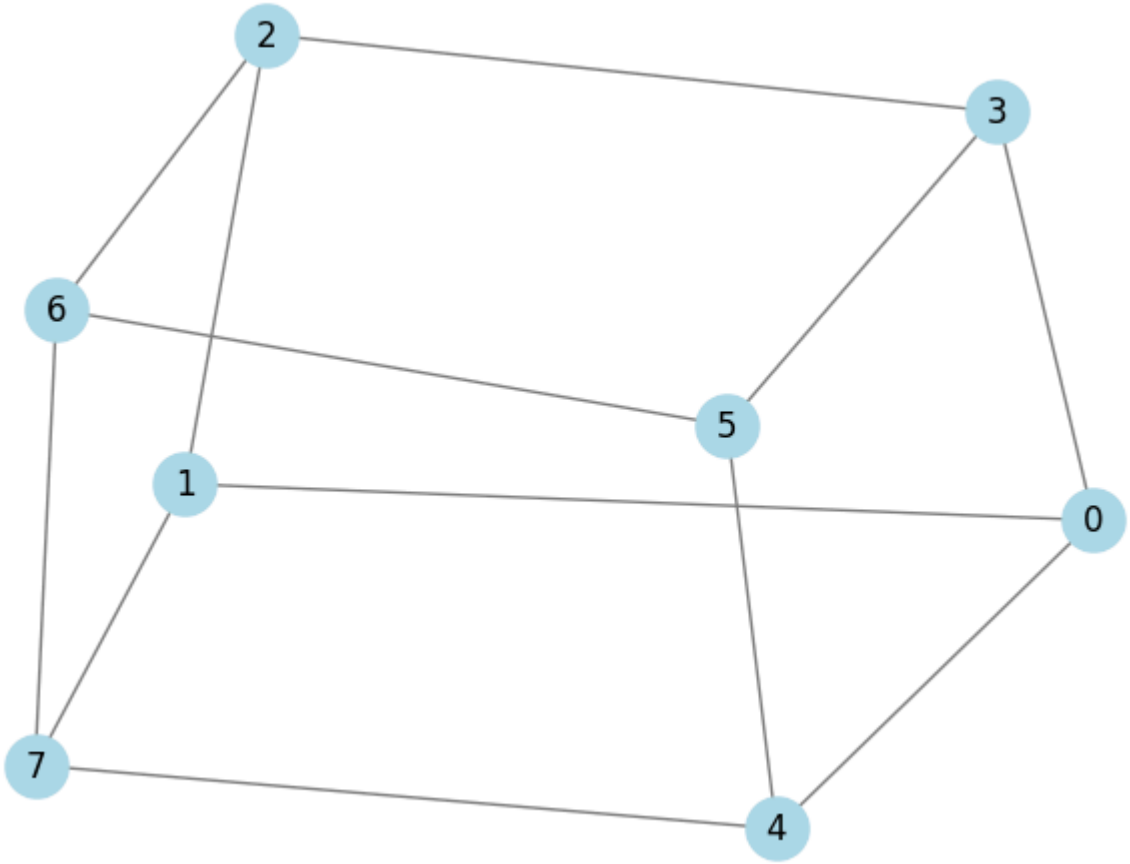


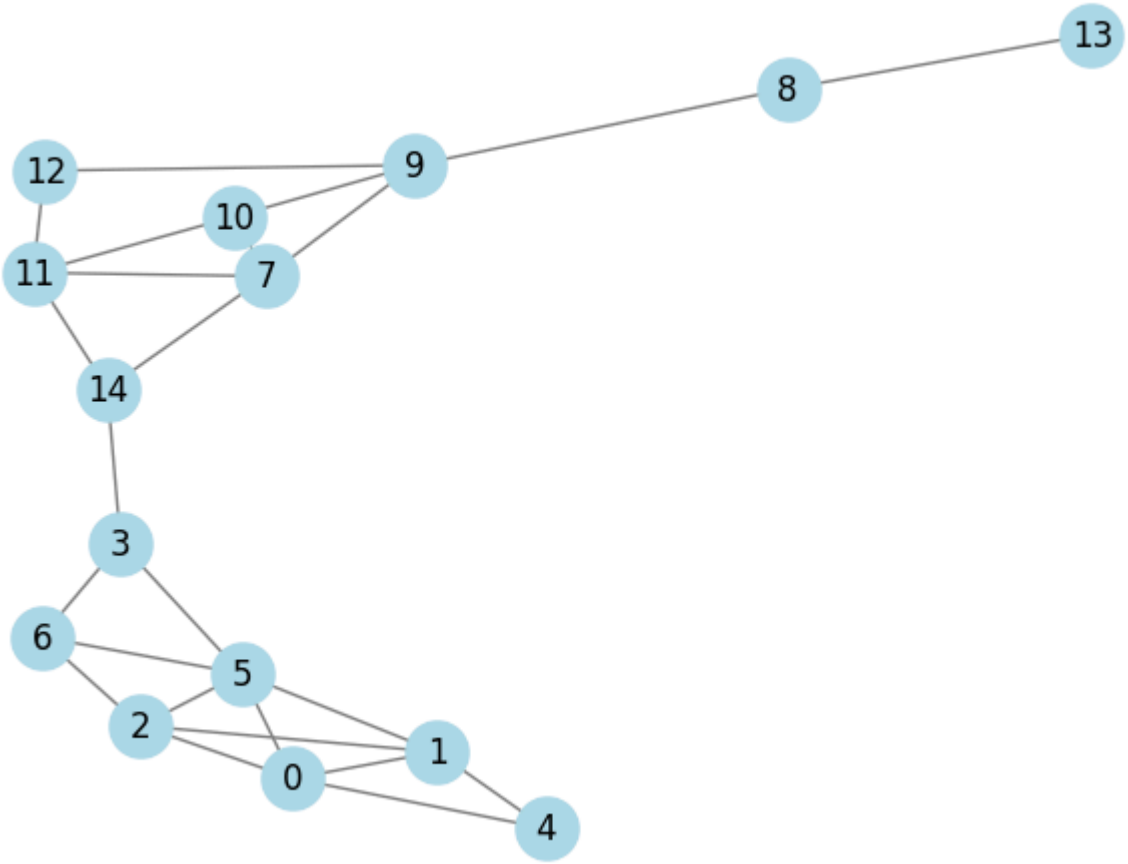
Wymagane testy

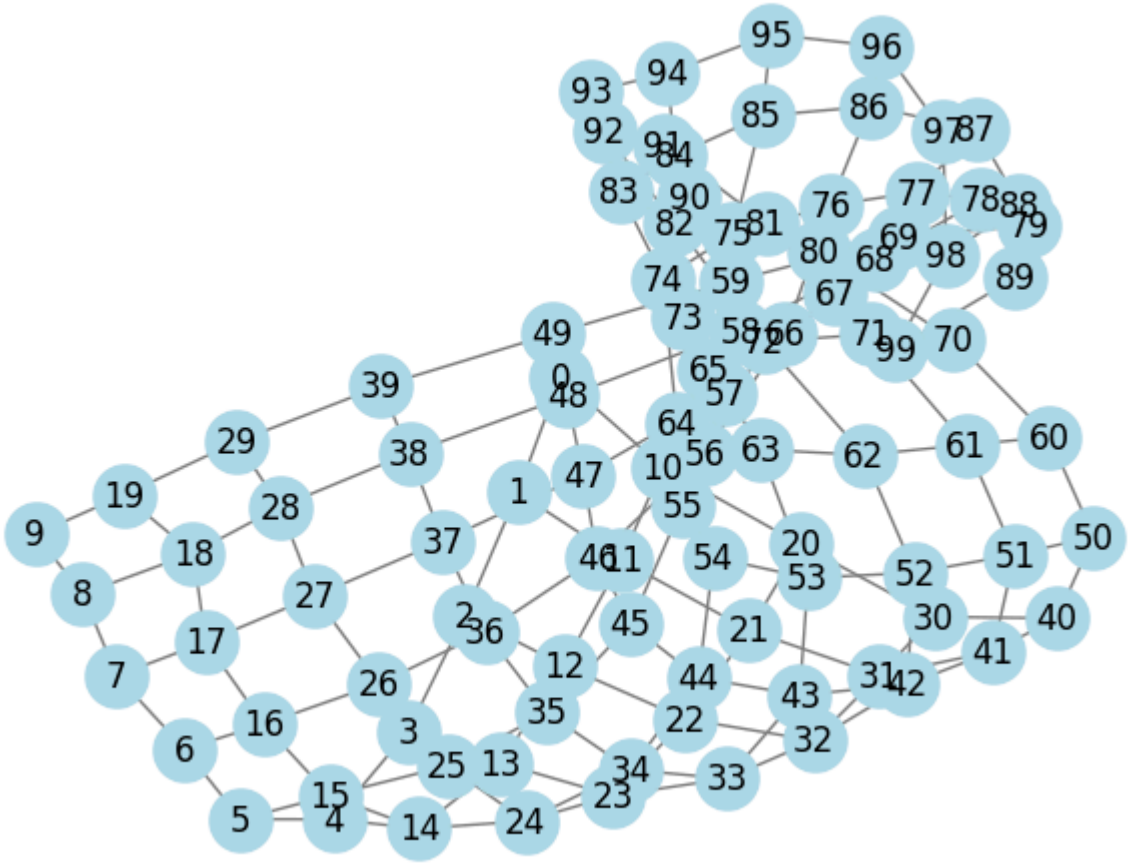
Testy zostały przeprowadzone na poniższych grafach:

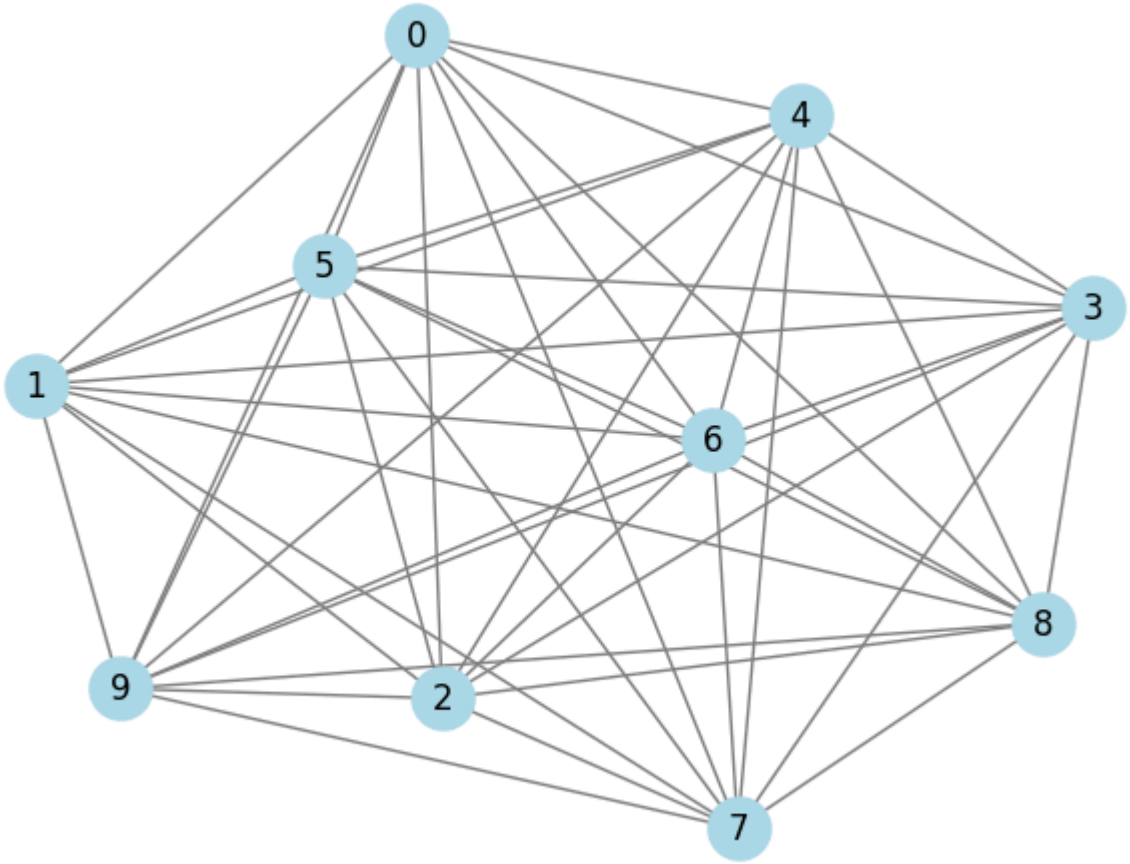
- Spójny graf losowy (20 wierzchołków)
- Graf 3-regularny (kubiczny) (8 wierzchołków)
- Graf złożony z dwóch grafów losowych połaczonych mostkiem (15 wierzchołków)
- Graf siatka 2D (100 wierzchołków)
- Graf typu small-world (10 wierzchołków)

In [85]:
```
generate_random_graph("graphs/random.txt", 20, "random", 30, True)
generate_random_graph("graphs/cubical.txt", 10, "cubical", 30, True)
generate_random_graph("graphs/bridge.txt", 15, "bridge", 30, True)
generate_random_graph("graphs/grid.txt", 100, "grid", 30, True)
generate_random_graph("graphs/small_world.txt", 10, "small_world", 30, True)
```


2
3
6
5
1
0
7
4


13
8
12
9
10
11
7
14
3
6
5
2
1
0
4

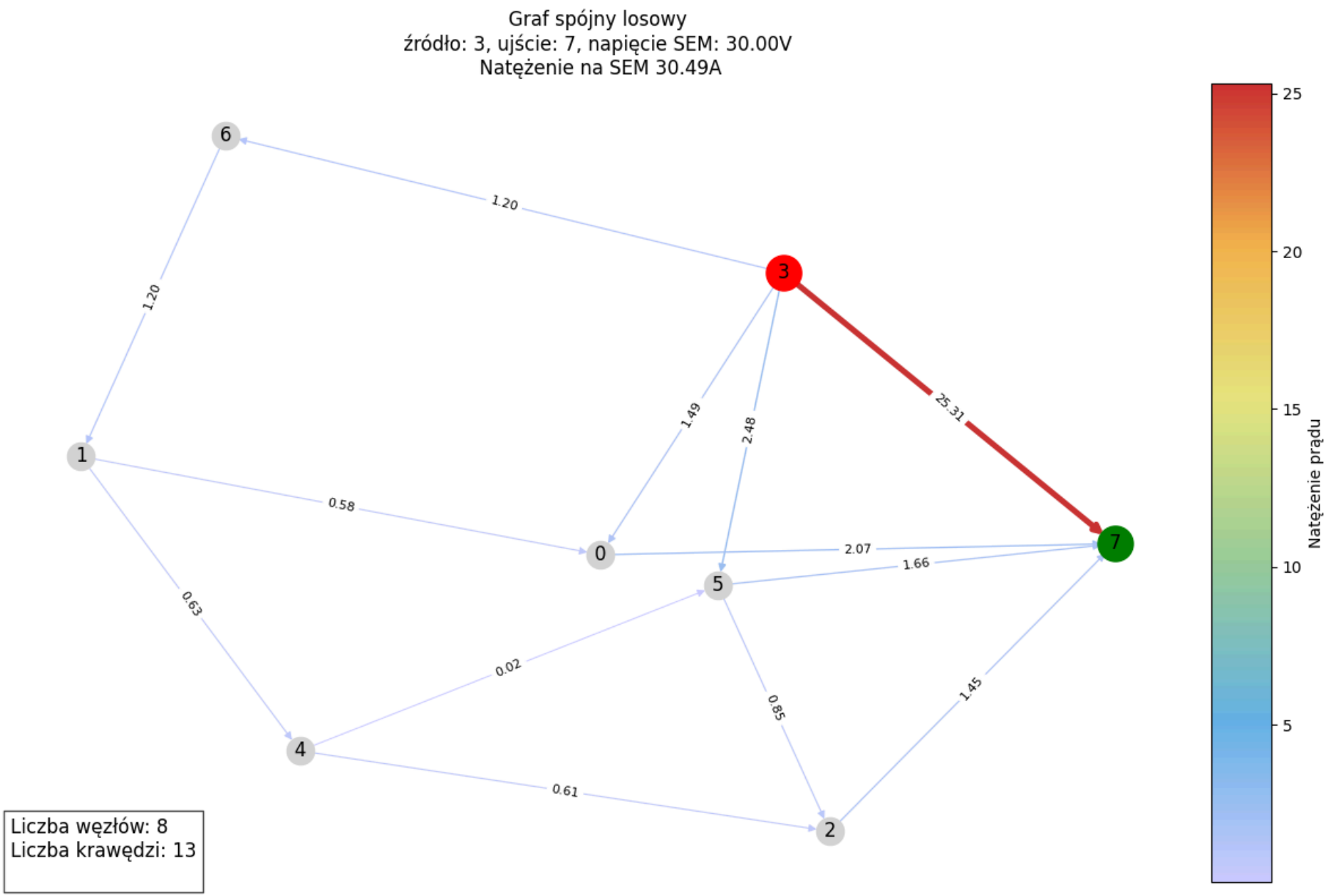

95
96
94
93
85
86
92
97
87
84
83
90
76
77
78
88
82
75
81
80
69
79
68
98
74
59
89
67
73
49
58
66
71
70
72
99
65
39
0
48
57
64
29
38
56
63
62
61
60
19
47
10
1
55
28
9
18
37
46
11
54
20
51
50
53
52
8
27
2
45
21
30
40
17
36
12
44
31
41
7
26
43
42
16
35
22
3
32
6
25
13
34
33
15
23
5
4
14
24



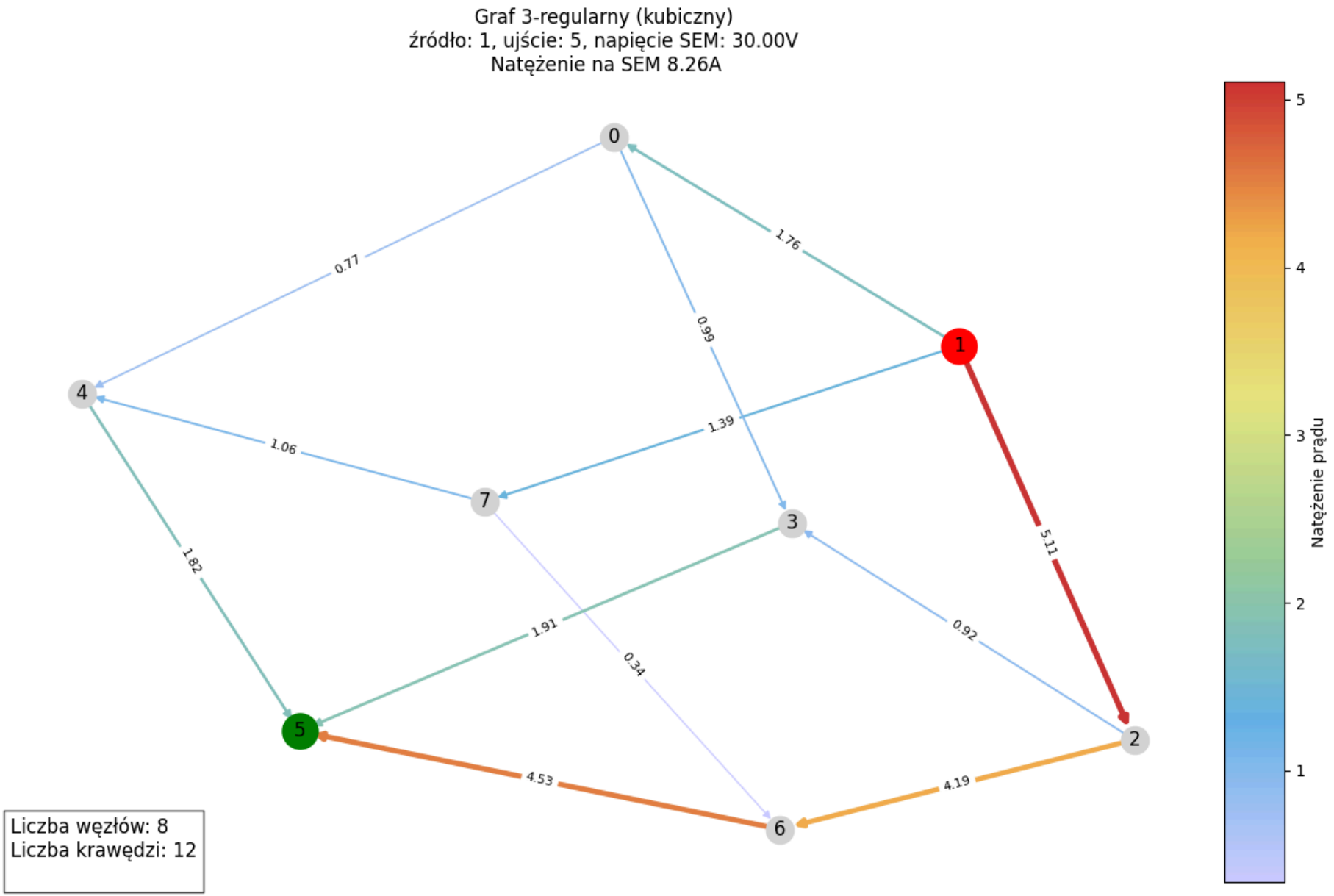
In [131…

```
solve_from_file("graphs/random.txt", "Graf spójny losowy")
solve_from_file("graphs/cubical.txt", "Graf 3-regularny (kubiczny)")
solve_from_file("graphs/bridge.txt", "Graf złożony z dwóch grafów losowych połaczonych mostkiem")
solve_from_file("graphs/grid.txt", "Graf siatka 2D")
solve_from_file("graphs/small_world.txt","Graf typu small-world")
```

```
Graf spójny losowy
wyliczone natężenia:
{(0, 3): np.float64(-1.4944157521211636), (3, 5): np.float64(2.484905885344273), (3, 6): np.float64(1.2027174993407614), (3,
7): np.float64(25.31089266824036), (0, 1): np.float64(-0.5771228981729519), (0, 7): np.float64(2.071538650294114), (2, 5): np.f
loat64(-0.8456682823010958), (4, 5): np.float64(0.017815263091015554), (5, 7): np.float64(1.6570528661342019), (1, 6): np.float
64(-1.2027174993407763), (2, 7): np.float64(1.4534476203778923), (1, 4): np.float64(0.6255946011678295), (2, 4): np.float64(-0.
6077793380768084)}
```

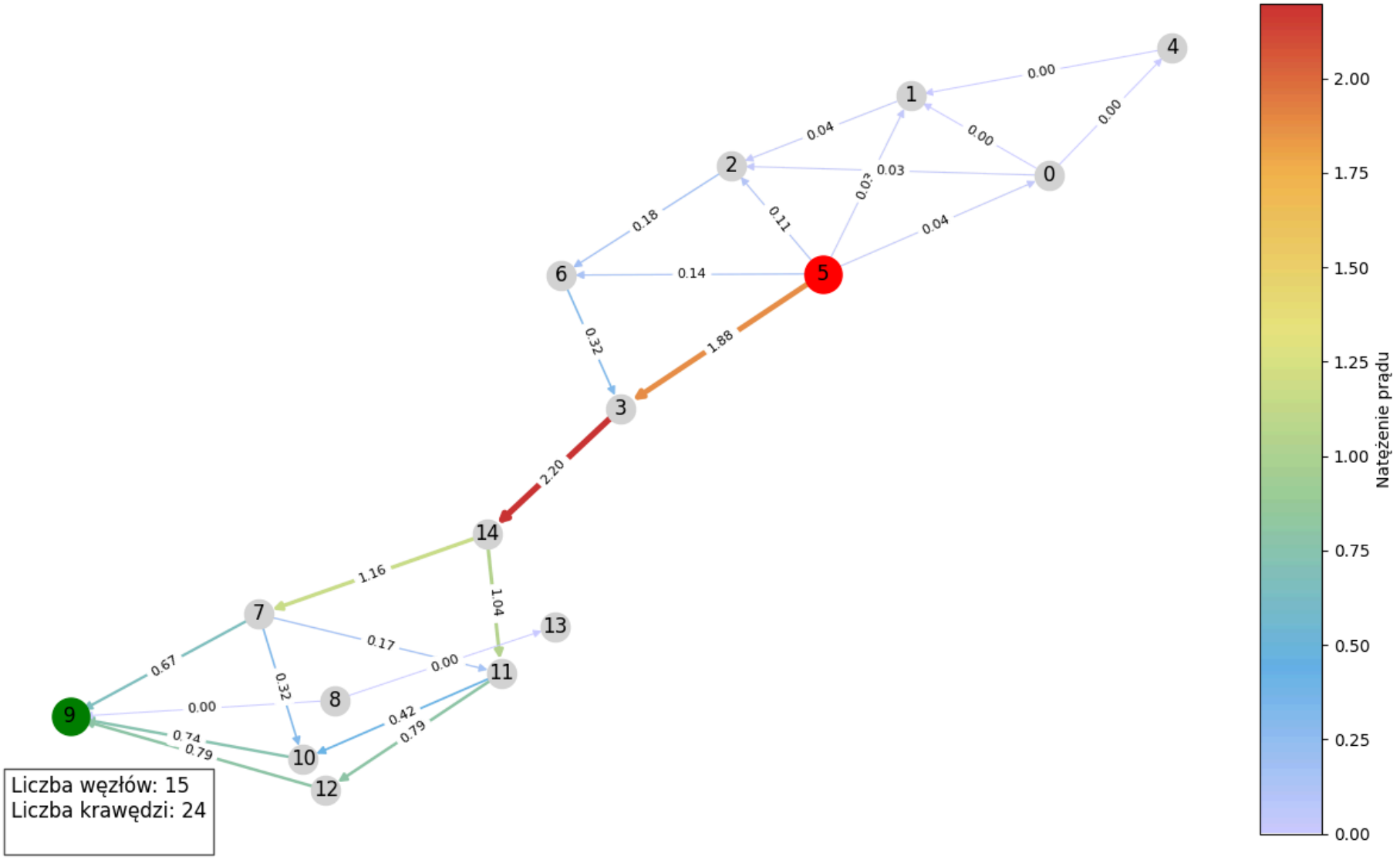

Graf spójny losowy
źródło: 3, ujście: 7, napięcie SEM: 30.00V
Natężenie na SEM 30.49A
6
1.20
3
1.20
1
1.49
2.48
25.31
0.58
0
2.07
7
1.66
5
0.63
0.02
0.85
1.45
4
0.61
2
Liczba węzłów: 8
Liczba krawędzi: 13
Natężenie prądu
25
20
15
10
5

```
Graf 3-regularny (kubiczny)
wyliczone natężenia:
{(0, 1): np.float64(-1.7608092148538317), (1, 2): np.float64(5.110461979645632), (1, 7): np.float64(1.39365021042266), (0, 3):
np.float64(0.9947066714789439), (0, 4): np.float64(0.7661025433749002), (2, 3): np.float64(0.9196045965444296), (2, 6): np.floa
t64(4.190857383101197), (4, 7): np.float64(-1.057804890017584), (6, 7): np.float64(-0.33584532040505266), (3, 5): np.float64(1.
914311268023374), (4, 5): np.float64(1.8239074333924723), (5, 6): np.float64(-4.52670270350626)}
```


Graf 3-regularny (kubiczny)
źródło: 1, ujście: 5, napięcie SEM: 30.00V
Natężenie na SEM 8.26A
0
1
2
3
4
5
6
7
0.77
1.76
0.99
1.06
1.39
1.82
5.11
1.91
0.92
0.34
4.53
4.19
Natężenie prądu
1
2
3
4
5
Liczba węzłów: 8
Liczba krawędzi: 12

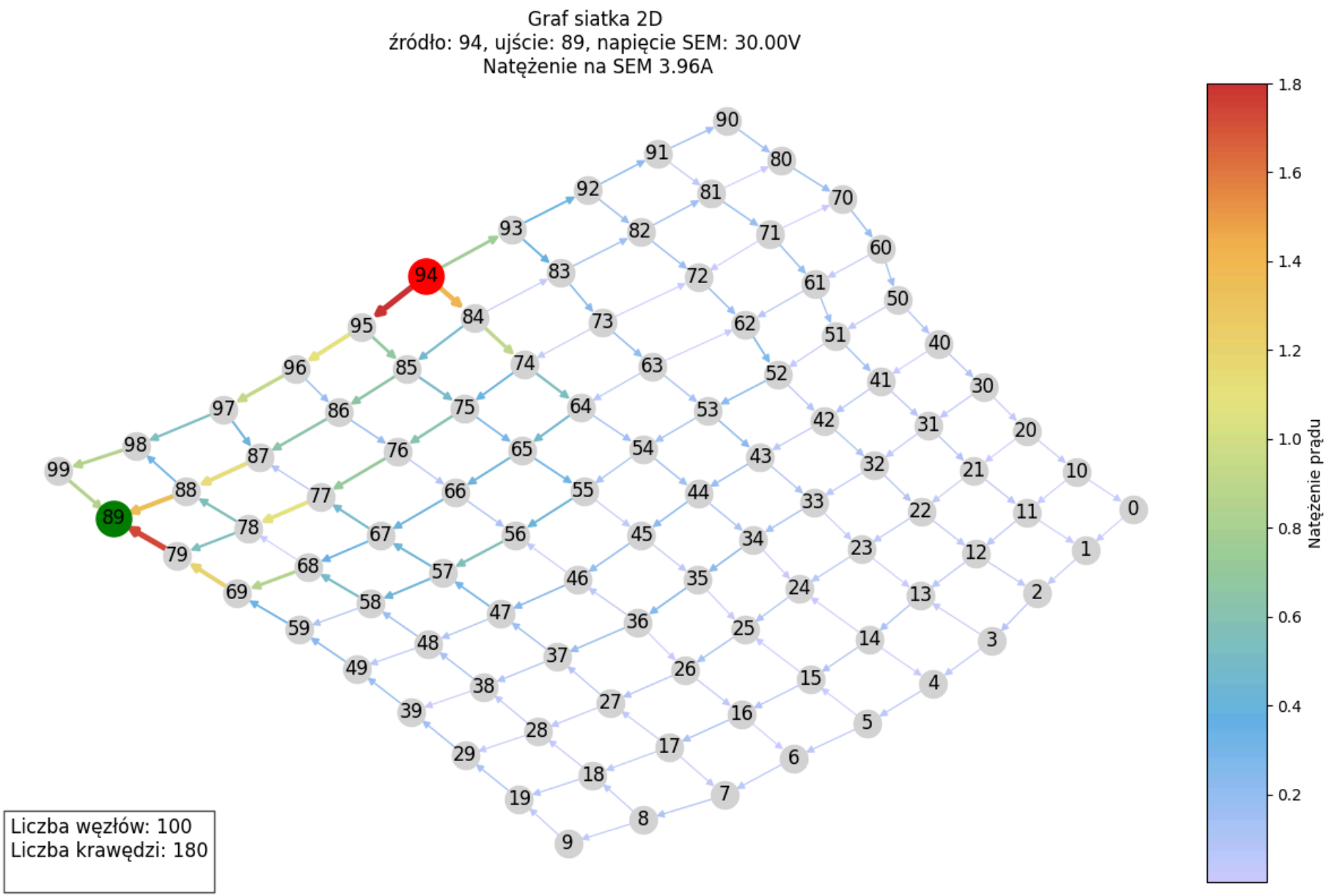
```
Graf złożony z dwóch grafów losowych połaczonych mostkiem
wyliczone natężenia:
{(0, 5): np.float64(-0.03652684229874503), (1, 5): np.float64(-0.0348638473639502 1), (2, 5): np.float64(-0.10666659372184306),
(3, 5): np.float64(-1.8774686921034847), (5, 6): np.float64(0.142371300635558), (0, 1): np.float64(0.002165596727648829), (0,
2): np.float64(0.03186614113539228), (0, 4): np.float64(0.0024951044357014317), (1, 2): np.float64(0.03952454852730661), (1,
4): np.float64(-0.0024951044357042133), (2, 6): np.float64(0.17805728338453786), (3, 6): np.float64(-0.32042858402009294), (3,
14): np.float64(2.1978972761235784), (7, 14): np.float64(-1.1600388367219157), (11, 14): np.float64(-1.037858439401661), (7,
9): np.float64(0.6654049707575196), (7, 10): np.float64(0.32430424885122006), (7, 11): np.float64(0.17032961711317565), (10, 1
1): np.float64(-0.4151823405805059), (11, 12): np.float64(0.793005715934324), (8, 9): np.float64(1.887379141862766e-15), (9, 1
0): np.float64(-0.739486589431734), (9, 12): np.float64(-0.7930057159343251), (8, 13): np.float64(2.185751579730777e-16)}
```

Graf złożony z dwóch grafów losowych połaczonych mostkiem
źródło: 5, ujście: 9, napięcie SEM: 30.00V
Natężenie na SEM 2.20A

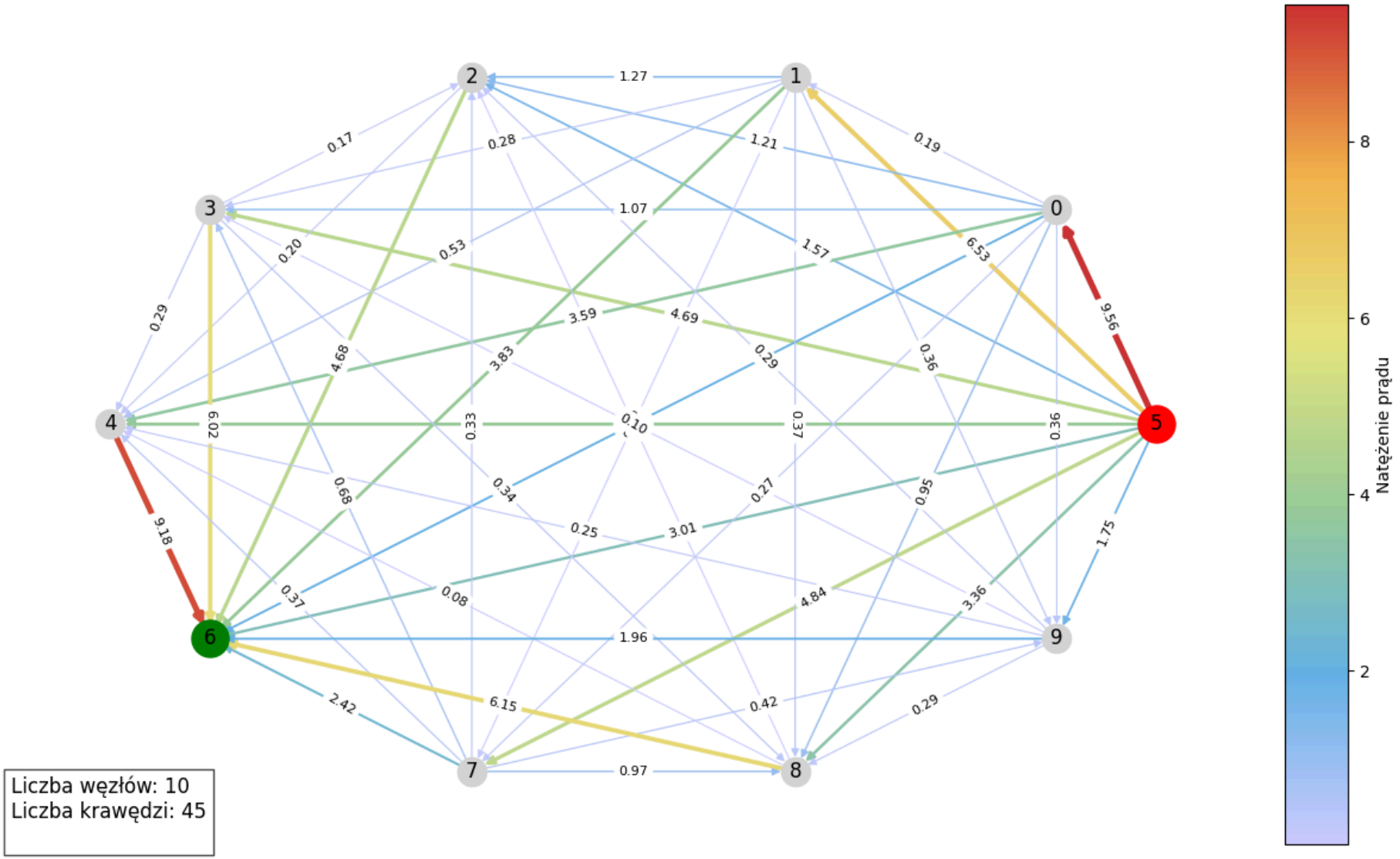


```
Graf siatka 2D
wyliczone natężenia:
{(84, 94): np.float64(-1.4042853260508306), (93, 94): np.float64(-0.755396584819666), (94, 95): np.float64(1.8012822758862856),
(74, 84): np.float64(-0.8975419392869977), (83, 84): np.float64(-0.03320774415136163), (84, 85): np.float64(0.473535642612468
4), (83, 93): np.float64(-0.3701291822382575), (92, 93): np.float64(-0.3852674025814077), (85, 95): np.float64(-0.6853142218320
144), (95, 96): np.float64(1.115968054054274), (64, 74): np.float64(-0.5373188348610308), (73, 74): np.float64(0.02686261258774
1586), (74, 75): np.float64(0.38708571701370775), (73, 83): np.float64(-0.2484098142407658), (82, 83): np.float64(-0.1549271121
4885302), (75, 85): np.float64(-0.5076949574967411), (85, 86): np.float64(0.6511549069477494), (82, 92): np.float64(-0.17136718
811326418), (91, 92): np.float64(-0.21390021446814345), (86, 96): np.float64(-0.18110601816243888), (96, 97): np.float64(0.9348
620358918374), (54, 64): np.float64(-0.14470420372513743), (63, 64): np.float64(0.06788080698262083), (64, 65): np.float64(0.46
04954381185188), (63, 73): np.float64(-0.20797992178152153), (72, 73): np.float64(-0.013567279871503302), (65, 75): np.float64
(-0.2858860497528669), (75, 76): np.float64(0.6088946247575798), (72, 82): np.float64(-0.1722483805527081), (81, 82): np.float6
4(-0.15404591970941034), (76, 86): np.float64(-0.19618430352503327), (86, 87): np.float64(0.6360766215851574), (81, 91): np.flo
at64(-0.050442241915363104), (90, 91): np.float64(-0.16345797255277764), (87, 97): np.float64(-0.4006766937138977), (97, 98): n
p.float64(0.5341853421779359), (44, 54): np.float64(-0.15851502765578004), (53, 54): np.float64(0.14061416161683515), (54, 55):
np.float64(0.12680333768619356), (53, 63): np.float64(-0.13138561608452037), (62, 63): np.float64(-0.00871349871437671), (55, 6
5): np.float64(-0.3246299284951256), (65, 66): np.float64(0.42175155937625863), (62, 72): np.float64(-0.18957125754884602), (7
1, 72): np.float64(0.0037555971246350517), (66, 76): np.float64(-0.08914879878970691), (76, 77): np.float64(0.715930129492903
9), (71, 81): np.float64(-0.19217220828902845), (80, 81): np.float64(-0.012315953335742716), (77, 87): np.float64(0.09287117148
596696), (87, 88): np.float64(1.1296244867850223), (80, 90): np.float64(-0.16345797255277592), (88, 98): np.float64(0.332373198
0731227), (98, 99): np.float64(0.8665585402510529), (34, 44): np.float64(-0.11548314099733858), (43, 44): np.float64(0.17347179
439441932), (44, 45): np.float64(0.21650368105286236), (43, 53): np.float64(-0.21817337674845455), (52, 53): np.float64(0.22740
19222807699), (45, 55): np.float64(-0.038295614435791464), (55, 56): np.float64(0.41313765174552686), (52, 62): np.float64(-0.2
745717667223219), (61, 62): np.float64(0.07628701045910291), (56, 66): np.float64(-0.12167585589052865), (66, 67): np.float64
(0.389224502275436), (61, 71): np.float64(-0.18575270914033026), (70, 71): np.float64(-0.002663902024061693), (67, 77): np.floa
t64(0.4646510490530002), (77, 78): np.float64(1.0877100070599404), (70, 80): np.float64(-0.17577392588851776), (78, 88): np.flo
at64(0.5551206971264884), (88, 89): np.float64(1.3523719858383907), (89, 99): np.float64(-0.8665585402510533), (24, 34): np.flo
at64(-0.02218433079243655), (33, 34): np.float64(0.14087374347120443), (34, 35): np.float64(0.23417255367610762), (33, 43): np.
float64(-0.10809754919455095), (42, 43): np.float64(0.06339596684051416), (35, 45): np.float64(-0.043867032634838765), (45, 4
6): np.float64(0.2109322628538164), (42, 52): np.float64(-0.08446802672436149), (51, 52): np.float64(0.03729818228280808), (46,
56): np.float64(0.025377213195578438), (56, 57): np.float64(0.560190720831636), (51, 61): np.float64(-0.14433952152191645), (6
0, 61): np.float64(0.034873822840691765), (57, 67): np.float64(0.4380788923193189), (67, 68): np.float64(0.3626523455417532),
(60, 70): np.float64(-0.17843782791258123), (68, 78): np.float64(0.011712692684496318), (78, 79): np.float64(0.544302002617947
9), (79, 89): np.float64(1.742033660667336), (14, 24): np.float64(0.008356997151380108), (23, 24): np.float64(0.069702737144153
2), (24, 25): np.float64(0.1002440650879704), (23, 33): np.float64(-0.028885789577299304), (32, 33): np.float64(0.0616619838539
5261), (25, 35): np.float64(-0.020723301517020065), (35, 36): np.float64(0.257316284793929), (32, 42): np.float64(-0.1024437527
9037175), (41, 42): np.float64(0.08137169290652208), (36, 46): np.float64(0.06382565937217824), (46, 47): np.float64(0.24938070
903041365), (41, 51): np.float64(-0.14675457280598742), (50, 51): np.float64(0.03971323356687906), (47, 57): np.float64(0.30462
42199481043), (57, 58): np.float64(0.4267360484604207), (50, 60): np.float64(-0.14356400507188985), (58, 68): np.float64(0.5025
278398806831), (68, 69): np.float64(0.8534674927379395), (69, 79): np.float64(1.1977316580493875), (4, 14): np.float64(-0.01438
8871816820019), (13, 14): np.float64(0.11640092351199327), (14, 15): np.float64(0.09365505454379214), (13, 23): np.float64(-0.0
```

13971757518388188), (22, 23): np.float64(0.05478870508524164), (15, 25): np.float64(0.016560115699932605), (25, 26): np.float64(0.13752748230492245), (22, 32): np.float64(-0.08748754404231074), (31, 32): np.float64(0.04670577510589119), (26, 36): np.float64(-0.006071863744863709), (36, 37): np.float64(0.1874187616768882), (31, 41): np.float64(-0.07860712831278863), (40, 41): np.float64(0.013224248413326284), (37, 47): np.float64(0.16135868771507944), (47, 48): np.float64(0.10611517679738561), (40, 50): np.float64(-0.10385077150501024), (48, 58): np.float64(0.17647087025709807), (58, 59): np.float64(0.10067907883684196), (59, 69): np.float64(0.34426416531145076), (3, 4): np.float64(0.05279754926204383), (4, 5): np.float64(0.06718642107886386), (3, 13): np.float64(0.03058365539912592), (12, 13): np.float64(0.07184551059447875), (5, 15): np.float64(0.004575591408589381), (15, 16): np.float64(0.08167053025244778), (12, 22): np.float64(-0.05797482607337019), (21, 22): np.float64(0.025275987116301774), (16, 26): np.float64(-0.054610983907396854), (26, 27): np.float64(0.08898836214238942), (21, 31): np.float64(-0.08898966382520643), (30, 31): np.float64(0.05708831061831081), (27, 37): np.float64(0.07627019326396095), (37, 38): np.float64(0.10233026722576788), (30, 40): np.float64(-0.09062652309168324), (38, 48): np.float64(0.13548297818328622), (48, 49): np.float64(0.06512728472357673), (49, 59): np.float64(0.24358508647460914), (2, 3): np.float64(0.08338120466116879), (5, 6): np.float64(0.06261082967027289), (2, 12): np.float64(-0.04980064138133232), (11, 12): np.float64(0.0636713259024399), (6, 16): np.float64(-0.008889817390399235), (16, 17): np.float64(0.12739169676944428), (11, 21): np.float64(-0.06915310052764198), (20, 21): np.float64(0.0054394238187366745), (17, 27): np.float64(0.038720266892329304), (27, 28): np.float64(0.05143843577075741), (20, 30): np.float64(-0.03353821247337116), (28, 38): np.float64(0.05686997115446166), (38, 39): np.float64(0.023717260196942272), (39, 49): np.float64(0.1784578017510305), (1, 2): np.float64(0.03358056327983724), (6, 7): np.float64(0.07150064706067297), (1, 11): np.float64(-0.01796288492404645), (10, 11): np.float64(0.012481110298845056), (7, 17): np.float64(-0.030973497604452158), (17, 18): np.float64(0.05769793227266393), (10, 20): np.float64(-0.02809878865463577), (18, 28): np.float64(0.03334542576184651), (28, 29): np.float64(0.02791389037814196), (29, 39): np.float64(0.15474054155408692), (0, 1): np.float64(0.015617678355790796), (7, 8): np.float64(0.10247414466512639), (0, 10): np.float64(-0.015617678355790608), (8, 18): np.float64(0.0550672373660025), (18, 19): np.float64(0.07941974387682063), (19, 29): np.float64(0.12682665117594633), (8, 9): np.float64(0.047406907299124654), (9, 19): np.float64(0.047406907299125084)}


Graf siatka 2D
źródło: 94, ujście: 89, napięcie SEM: 30.00V
Natężenie na SEM 3.96A
Liczba węzłów: 100
Liczba krawędzi: 180
Natężenie prądu
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2

```
Graf typu small-world
wyliczone natężenia:
{(0, 5): np.float64(-9.558905334338135), (1, 5): np.float64(-6.525797365003224), (2, 5): np.float64(-1.5674458156451867), (3,
5): np.float64(-4.6902240050307435), (4, 5): np.float64(-3.8652564381189687), (5, 6): np.float64(3.013812179515585), (5, 7): n
p.float64(4.838903158640609), (5, 8): np.float64(3.356584146336455), (5, 9): np.float64(1.7463676242525397), (0, 1): np.float64
(0.18981000909793785), (0, 2): np.float64(1.214493280407201), (0, 3): np.float64(1.0735509035405824), (0, 4): np.float64(3.5889
397638012706), (0, 6): np.float64(1.9114610984657556), (0, 7): np.float64(0.2712314116825274), (0, 8): np.float64(0.94869158068
57388), (0, 9): np.float64(0.360727286657105), (1, 2): np.float64(1.2727432596661796), (1, 3): np.float64(0.2775139928551136),
(1, 4): np.float64(0.5284159936282667), (1, 6): np.float64(3.829766834171926), (1, 7): np.float64(0.07124762858269572), (1, 8):
np.float64(0.3735467712180765), (1, 9): np.float64(0.3623728939789143), (2, 3): np.float64(-0.17216121383540667), (2, 4): np.fl
oat64(0.19785750803711025), (2, 6): np.float64(4.683926710995301), (2, 7): np.float64(-0.33083260271964765), (2, 8): np.float64
(-0.03258631107255476), (2, 9): np.float64(-0.29152173568622497), (3, 4): np.float64(0.2881819010796855), (3, 6): np.float64(6.
015538170054548), (3, 7): np.float64(-0.6752494602631626), (3, 8): np.float64(0.33589176475630517), (3, 9): np.float64(-0.09523
468803633012), (4, 6): np.float64(9.17728158074451), (4, 7): np.float64(-0.37370258964379016), (4, 8): np.float64(-0.0802328052
3271246), (4, 9): np.float64(-0.25469458120269506), (6, 7): np.float64(-2.421184901402478), (6, 8): np.float64(-6.1521384460567
87), (6, 9): np.float64(-1.9581861454746012), (7, 8): np.float64(0.9651428141172808), (7, 9): np.float64(0.4152698307594757),
(8, 9): np.float64(-0.28510048524816783)}
```



Duży graf na 196 wierzchołków

In [ ]:
```
generate_random_graph("graphs/grid200.txt", 196, "grid", 30, True)
```

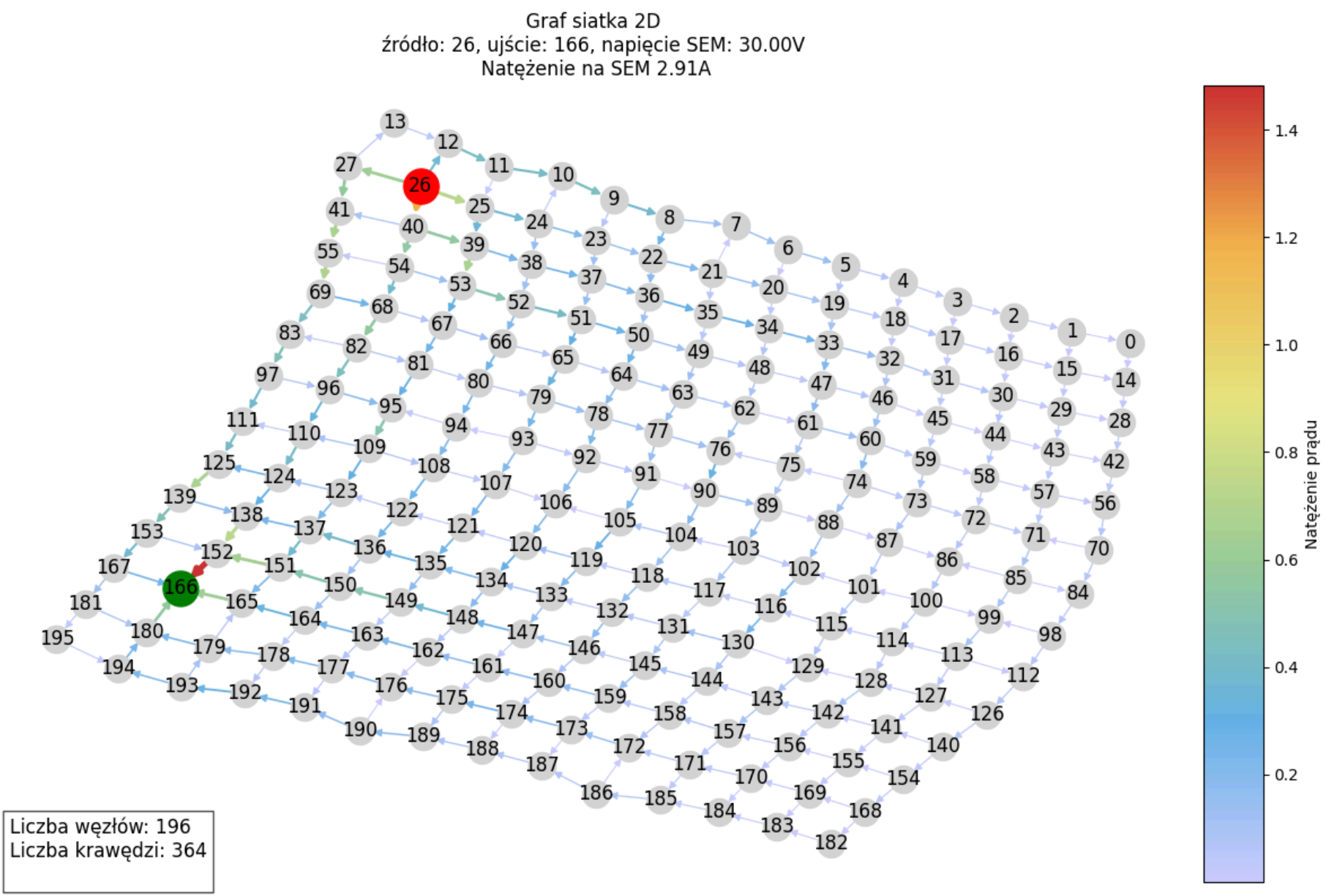
In [ ]:
```
solve_from_file("graphs/grid200.txt", "Graf siatka 2D")
```

Graf siatka 2D
wyliczone natężenia:
{(12, 26): np.float64(-0.37529695523677564), (25, 26): np.float64(-0.7241521128360352), (26, 40): np.float64(1.188090239867099
8), (26, 27): np.float64(0.6231947098013612), (11, 12): np.float64(-0.4262796775905134), (12, 13): np.float64(-0.05098272235372
62), (11, 25): np.float64(0.009049853404212058), (24, 25): np.float64(-0.37614463062533976), (25, 39): np.float64(0.35705733561
49178), (39, 40): np.float64(-0.5517158164963164), (40, 54): np.float64(0.5447007887605222), (40, 41): np.float64(0.09167363461
028233), (13, 27): np.float64(-0.05098272235370907), (27, 41): np.float64(0.5722119874476999), (10, 11): np.float64(-0.41722982
41863105), (10, 24): np.float64(-0.026431073725091753), (23, 24): np.float64(-0.2040137562281323), (24, 38): np.float64(0.14569
980067210914), (38, 39): np.float64(-0.24040450433395652), (39, 53): np.float64(0.6683686477772628), (53, 54): np.float64(-0.13
420240376544457), (54, 68): np.float64(0.38994683406207076), (54, 55): np.float64(0.020551550933024496), (41, 55): np.float64
(0.6638856220579755), (9, 10): np.float64(-0.44366089791138513), (9, 23): np.float64(0.059086727917364384), (22, 23): np.float6
4(-0.1429365997317129), (23, 37): np.float64(0.12016388441379891), (37, 38): np.float64(-0.2550705638482361), (38, 52): np.floa
t64(0.13103374115780386), (52, 53): np.float64(-0.49926934465979006), (53, 67): np.float64(0.3033017068829562), (67, 68): np.fl
oat64(-0.08071181404631296), (68, 82): np.float64(0.5636881187734062), (68, 69): np.float64(-0.2544530987576529), (55, 69): np.
float64(0.6844371729910124), (8, 9): np.float64(-0.38457416999400873), (8, 22): np.float64(0.25166986740385716), (21, 22): np.f
loat64(-0.20639313984214538), (22, 36): np.float64(0.18821332729343151), (36, 37): np.float64(-0.21516487300588963), (37, 51):
np.float64(0.1600695752561309), (51, 52): np.float64(-0.41902470257320845), (52, 66): np.float64(0.21127838324440754), (66, 6
7): np.float64(-0.12670611499853457), (67, 81): np.float64(0.25730740593071055), (81, 82): np.float64(-0.08536747020448586), (8
2, 96): np.float64(0.439246619210961), (82, 83): np.float64(0.03907402935794696), (69, 83): np.float64(0.42998407423334817),
(7, 8): np.float64(-0.1329043025901545), (7, 21): np.float64(-0.0018749596233024646), (20, 21): np.float64(-0.147210289009539
4), (21, 35): np.float64(0.057307891209322344), (35, 36): np.float64(-0.28471511763493085), (36, 50): np.float64(0.118663082664
378), (50, 51): np.float64(-0.2721012029086541), (51, 65): np.float64(0.306993074920659), (65, 66): np.float64(-0.0740595370861
0505), (66, 80): np.float64(0.26392496115684566), (80, 81): np.float64(-0.05387856076682502), (81, 95): np.float64(0.2887963153
683905), (95, 96): np.float64(-0.18944013613202623), (96, 110): np.float64(0.33219754241655214), (96, 97): np.float64(-0.082391
0593375786), (83, 97): np.float64(0.46905810359131056), (6, 7): np.float64(-0.13477926221347145), (6, 20): np.float64(0.0009444
445248389891), (19, 20): np.float64(-0.11375068883497415), (20, 34): np.float64(0.034404044699433416), (34, 35): np.float64(-0.
28706203807340136), (35, 49): np.float64(0.054960970770832646), (49, 50): np.float64(-0.1329179328283522), (50, 64): np.float64
(0.25784635274470286), (64, 65): np.float64(-0.06087126111453203), (65, 79): np.float64(0.32018135089222927), (79, 80): np.floa
t64(-0.07492339109074489), (80, 94): np.float64(0.24288013083294235), (94, 95): np.float64(0.01821807581128978), (95, 109): np.
float64(0.4964545273116621), (109, 110): np.float64(0.10739239366477267), (110, 124): np.float64(0.4048236335901711), (110, 11
1): np.float64(0.03476630249117098), (97, 111): np.float64(0.38666704425370246), (5, 6): np.float64(-0.13383481768862776), (5,
19): np.float64(0.06641000455741326), (18, 19): np.float64(-0.06650377056442695), (19, 33): np.float64(0.11365692282793514), (3
3, 34): np.float64(-0.24104494855940123), (34, 48): np.float64(0.08042113421344514), (48, 49): np.float64(-0.0647931617875674
9), (49, 63): np.float64(0.12308574181163087), (63, 64): np.float64(-0.11807354950081535), (64, 78): np.float64(0.2006440643583
9144), (78, 79): np.float64(-0.14808834375898927), (79, 93): np.float64(0.24701639822397517), (93, 94): np.float64(0.0001385368
7083948625), (94, 108): np.float64(0.22480059189249443), (108, 109): np.float64(-0.08481613522051917), (109, 123): np.float64
(0.3042459984263778), (123, 124): np.float64(0.13581070324874384), (124, 138): np.float64(0.2971402211270135), (124, 125): np.f
loat64(0.24349411571191565), (111, 125): np.float64(0.42143334674486194), (4, 5): np.float64(-0.06742481313123798), (4, 18): n
p.float64(0.02441832248032487), (17, 18): np.float64(-0.06629618240191332), (18, 32): np.float64(0.02462591064282772), (32, 3
3): np.float64(-0.14319959319568784), (33, 47): np.float64(0.2115022781916631), (47, 48): np.float64(-0.0448015983641238), (48,
62): np.float64(0.10041269763689586), (62, 63): np.float64(-0.08994689263889924), (63, 77): np.float64(0.15121239867358036), (7

7, 78): np.float64(-0.1261717640634434), (78, 92): np.float64(0.22256064405397277), (92, 93): np.float64(-0.03555059803412578), (93, 107): np.float64(0.2113272633190155), (107, 108): np.float64(-0.039403996855765386), (108, 122): np.float64(0.270212730257 2806), (122, 123): np.float64(0.05655726132044421), (123, 137): np.float64(0.22499255649806343), (137, 138): np.float64(0.24040 254403196618), (138, 152): np.float64(0.7270596747740166), (138, 139): np.float64(-0.1895169096150527), (125, 139): np.float64 (0.6649274624567982), (3, 4): np.float64(-0.04300649065094572), (3, 17): np.float64(0.012185184224284405), (16, 17): np.float64 (-0.05215076145017526), (17, 31): np.float64(0.026330605175981903), (31, 32): np.float64(-0.07858662510330015), (32, 46): np.fl oat64(0.08923887873519558), (46, 47): np.float64(-0.06949666282366102), (47, 61): np.float64(0.1868072137321251), (61, 62): np. float64(-0.026010532242977447), (62, 76): np.float64(0.16434905803284355), (76, 77): np.float64(-0.13033021193244526), (77, 9 1): np.float64(0.147053950804568), (91, 92): np.float64(-0.0643697749256409), (92, 106): np.float64(0.1937414671624582), (106, 107): np.float64(-0.013386768607500588), (107, 121): np.float64(0.2373444915672874), (121, 122): np.float64(0.0370934142671008 1), (122, 136): np.float64(0.2507488832039277), (136, 137): np.float64(0.3963666540338449), (137, 151): np.float64(0.3809566664 9994153), (151, 152): np.float64(0.6528836024999722), (152, 166): np.float64(1.48305048548014), (152, 153): np.float64(-0.10310 720820614304), (139, 153): np.float64(0.4754105528417499), (2, 3): np.float64(-0.030821306426654718), (2, 16): np.float64(0.006 547852015423037), (15, 16): np.float64(-0.030000830810116154), (16, 30): np.float64(0.028697782655462828), (30, 31): np.float64 (-0.06505538805184785), (31, 45): np.float64(0.039861842227430166), (45, 46): np.float64(-0.02483172520988676), (46, 60): np.fl oat64(0.1339038163489973), (60, 61): np.float64(-0.11689606117053232), (61, 75): np.float64(0.09592168480459547), (75, 76): np. float64(0.030678638452433793), (76, 90): np.float64(0.32535790841769935), (90, 91): np.float64(-0.006659746711207137), (91, 10 5): np.float64(0.20476397901899943), (105, 106): np.float64(0.004426034662965135), (106, 120): np.float64(0.21155427043291472), (120, 121): np.float64(0.04978261613610739), (121, 135): np.float64(0.2500336934362982), (135, 136): np.float64(0.2877330475131 1185), (136, 150): np.float64(0.14211527668322912), (150, 151): np.float64(0.4983686085202073), (151, 165): np.float64(0.226441 67252016803), (165, 166): np.float64(0.6227650633662482), (166, 180): np.float64(-0.563796965611516), (166, 167): np.float64(- 0.2411215032835927), (153, 167): np.float64(0.3723033446356086), (1, 2): np.float64(-0.024273454411225335), (1, 15): np.float64 (0.013917541871582829), (14, 15): np.float64(-0.017143535902779547), (15, 29): np.float64(0.026774836778921604), (29, 30): np.f loat64(-0.028147799846209014), (30, 44): np.float64(0.06560537086112683), (44, 45): np.float64(-0.00980097395810193), (45, 59): np.float64(0.0548925934791961), (59, 60): np.float64(-0.05283809411118324), (60, 74): np.float64(0.197961783408322), (74, 75): np.float64(0.010369668152793688), (75, 89): np.float64(0.07561271450494539), (89, 90): np.float64(-0.1335724003423373), (90, 10 4): np.float64(0.19844525478661085), (104, 105): np.float64(0.08917833031743026), (105, 119): np.float64(0.28951627467349134), (119, 120): np.float64(0.1262061962633527), (120, 134): np.float64(0.28797785056017283), (134, 135): np.float64(0.2155810684917 2312), (135, 149): np.float64(0.1778817144148763), (149, 150): np.float64(0.4944362139521908), (150, 164): np.float64(0.1381828 8211521415), (164, 165): np.float64(0.3240801188667802), (165, 179): np.float64(-0.07224327197930591), (179, 180): np.float64 (0.22105322940723707), (180, 194): np.float64(-0.24307583771978802), (180, 181): np.float64(-0.09966789848448739), (167, 181): np.float64(0.1311818413520297), (0, 1): np.float64(-0.010355912539626774), (0, 14): np.float64(0.010355912539619544), (14, 28): np.float64(0.02749944844238226), (28, 29): np.float64(-0.009310057425194815), (29, 43): np.float64(0.045612579199949384), (43, 44): np.float64(-0.011084601446649078), (44, 58): np.float64(0.06432174337257611), (58, 59): np.float64(-0.007999485366594434), (59, 73): np.float64(0.09973120222381406), (73, 74): np.float64(-0.056003236907075295), (74, 88): np.float64(0.1315888783484476 8), (88, 89): np.float64(-0.02115356693088267), (89, 103): np.float64(0.1880315479163641), (103, 104): np.float64(0.02831689563 227023), (104, 118): np.float64(0.13758382010145837), (118, 119): np.float64(0.04614231007204121), (119, 133): np.float64(0.209 45238848215747), (133, 134): np.float64(0.11765161438794894), (134, 148): np.float64(0.1900483964564198), (148, 149): np.float64 (0.37426385132519974), (149, 163): np.float64(0.0577093517879266), (163, 164): np.float64(0.24329466585018064), (164, 178): n p.float64(0.057397429098637195), (178, 179): np.float64(0.1875998895078174), (179, 193): np.float64(-0.10569661187871639), (19 3, 194): np.float64(0.2115618948522585), (194, 195): np.float64(-0.03151394286753026), (181, 195): np.float64(0.031513942867541

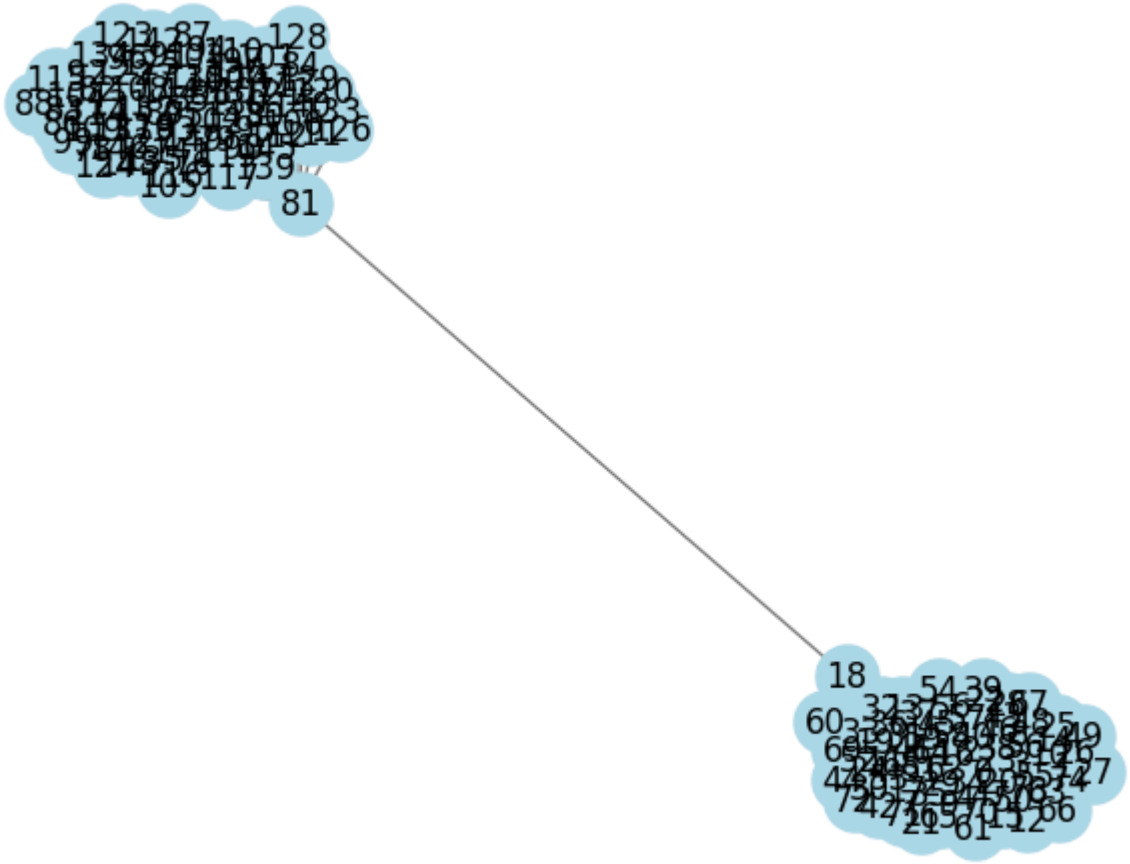
28), (28, 42): np.float64(0.03680950586759644), (42, 43): np.float64(-0.0011449562703346535), (43, 57): np.float64(0.0555522243
7624323), (57, 58): np.float64(-0.019786029974143676), (58, 72): np.float64(0.05253519876501664), (72, 73): np.float64(-0.05468
956480262786), (73, 87): np.float64(0.1010448743282584), (87, 88): np.float64(-0.0062526092478852515), (88, 102): np.float64(0.
14648983603146012), (102, 103): np.float64(-0.05476487466686645), (103, 117): np.float64(0.10494977761721341), (117, 118): np.f
loat64(0.07939776271255711), (118, 132): np.float64(0.17083927274195246), (132, 133): np.float64(0.05623816203623723), (133, 14
7): np.float64(0.14803893613043867), (147, 148): np.float64(0.26349901306407075), (148, 162): np.float64(0.07928355819529179),
(162, 163): np.float64(0.24601012114107199), (163, 177): np.float64(0.06042480707883457), (177, 178): np.float64(0.203186577121
30772), (178, 192): np.float64(0.07298411671211977), (192, 193): np.float64(0.31725850673097467), (42, 56): np.float64(0.037954
46213790552), (56, 57): np.float64(-0.024849648466385665), (57, 71): np.float64(0.050488605884005666), (71, 72): np.float64(-0.
048556147598258256), (72, 86): np.float64(0.05866861596938339), (86, 87): np.float64(-0.008852753090347292), (87, 101): np.floa
t64(0.09844473048576248), (101, 102): np.float64(0.019774743027595472), (102, 116): np.float64(0.22102945372593644), (116, 11
7): np.float64(0.02724498772386339), (117, 131): np.float64(0.052797002628536246), (131, 132): np.float64(0.07038034458309265),
(132, 146): np.float64(0.1849814552888228), (146, 147): np.float64(0.1399934437859042), (147, 161): np.float64(0.02453336685227
9324), (161, 162): np.float64(0.17927760236289894), (162, 176): np.float64(0.012551039417106948), (176, 177): np.float64(0.2036
175587298993), (177, 191): np.float64(0.06085578868742557), (191, 192): np.float64(0.2442743900188213), (56, 70): np.float64(0.
0628041106042853), (70, 71): np.float64(0.0040020736247331845), (71, 85): np.float64(0.10304682710697036), (85, 86): np.float64
(0.015335033993226601), (86, 100): np.float64(0.08285640305297066), (100, 101): np.float64(0.0038441667260899213), (101, 115):
np.float64(0.0825141541842692), (115, 116): np.float64(0.041054870959914565), (116, 130): np.float64(0.23483933696197604), (13
0, 131): np.float64(0.11986172848759902), (131, 145): np.float64(0.10227838653302289), (145, 146): np.float64(0.048935721081733
05), (146, 160): np.float64(0.09392373258465977), (160, 161): np.float64(0.18558066627375114), (161, 175): np.float64(0.0308364
307631292), (175, 176): np.float64(0.18816437140445116), (176, 190): np.float64(-0.0029021479083790264), (190, 191): np.float64
(0.18341860133140808), (70, 84): np.float64(0.058802036979563634), (84, 85): np.float64(-0.0030051485237763695), (85, 99): np.f
loat64(0.08470664458997107), (99, 100): np.float64(-0.010110220772119295), (100, 114): np.float64(0.06890201555475489), (114, 1
15): np.float64(0.03669648180272688), (115, 129): np.float64(0.07815576502708259), (129, 130): np.float64(-0.0077845424043147
8), (130, 144): np.float64(0.10719306607008926), (144, 145): np.float64(0.09528822964967902), (145, 159): np.float64(0.14863089
51009805), (159, 160): np.float64(0.16614291000343578), (160, 174): np.float64(0.07448597631430418), (174, 175): np.float64(0.2
456556837452492), (175, 189): np.float64(0.08832774310392269), (189, 190): np.float64(0.1863207492397671), (84, 98): np.float64
(0.06180718550337777), (98, 99): np.float64(0.0010004079835272334), (99, 113): np.float64(0.09581727334561826), (113, 114): np.
float64(0.03064501925325058), (114, 128): np.float64(0.06285055300523358), (128, 129): np.float64(0.00818098777239244), (129, 1
43): np.float64(0.09412129520379475), (143, 144): np.float64(0.043603908676322294), (144, 158): np.float64(0.0555087450967229
5), (158, 159): np.float64(0.06861089401819669), (159, 173): np.float64(0.05109887911578201), (173, 174): np.float64(0.19006437
909973872), (174, 188): np.float64(0.01889467166875905), (188, 189): np.float64(0.09799300613586423), (98, 112): np.float64(0.0
60806777519852195), (112, 113): np.float64(-0.006037811322909313), (113, 127): np.float64(0.05913444276948049), (127, 128): np.
float64(0.034601266708229154), (128, 142): np.float64(0.08927083194108883), (142, 143): np.float64(0.05505041360967517), (143,
157): np.float64(0.10556780013713697), (157, 158): np.float64(0.08571679517506943), (158, 172): np.float64(0.0726146462535768
3), (172, 173): np.float64(0.1473415160597201), (173, 187): np.float64(0.00837601607576391), (187, 188): np.float64(0.079098334
46709144), (112, 126): np.float64(0.06684458884278519), (126, 127): np.float64(0.01662088691048344), (127, 141): np.float64(0.0
41154062971734434), (141, 142): np.float64(0.023694817556409432), (142, 156): np.float64(0.05791523588782942), (156, 157): np.f
loat64(0.035952485865652126), (157, 171): np.float64(0.05580349082772006), (171, 172): np.float64(0.06671510778649532), (172, 1
86): np.float64(-0.008011762019664167), (186, 187): np.float64(0.07072231839132424), (126, 140): np.float64(0.0502237019322749
2), (140, 141): np.float64(0.010803966986589262), (141, 155): np.float64(0.02826321240190461), (155, 156): np.float64(0.0149320

00213604397), (156, 170): np.float64(0.036894750235769956), (170, 171): np.float64(0.055423609195165764), (171, 185): np.float64(0.044511992236390324), (185, 186): np.float64(0.07873408041099492), (140, 154): np.float64(0.03941973494568276), (154, 155): np.float64(0.009036116331204908), (155, 169): np.float64(0.022367328519508865), (169, 170): np.float64(0.026533356372633248), (170, 184): np.float64(0.008004497413244766), (184, 185): np.float64(0.03422208817460682), (154, 168): np.float64(0.03038361861448868), (168, 169): np.float64(0.015879325931267178), (169, 183): np.float64(0.011713298078132376), (183, 184): np.float64(0.02621759076136247), (168, 182): np.float64(0.014504292683217552), (182, 183): np.float64(0.014504292683229003)}



In [146…

```
generate_random_graph("graphs/bridge_big.txt", 150, "bridge", 30, True)
solve_from_file("graphs/bridge_big.txt", "Graf duży złożony z dwóch grafów losowych połaczonych mostkiem")
```


81
18

```
Graf duży złożony z dwóch grafów losowych połaczonych mostkiem
wyliczone natężenia:
{(0, 56): np.float64(-0.07530343246347251), (2, 56): np.float64(-0.13073980625993822), (3, 56): np.float64(-0.2736469739322864), (8, 56): np.float64(-0.5056374500948838), (9, 56): np.float64(-0.06527238505723111), (10, 56): np.float64(-0.06804216807512793), (11, 56): np.float64(-0.12430341127079217), (12, 56): np.float64(-0.39081660907233917), (13, 56): np.float64(-0.13863425841154672), (16, 56): np.float64(-0.1098555351486588), (18, 56): np.float64(-0.3523570941343496), (19, 56): np.float64(-0.2017546445325215), (21, 56): np.float64(-0.13690651209552412), (23, 56): np.float64(-0.09952937936008689), (25, 56): np.float64(-0.1015788048165731), (26, 56): np.float64(-0.12411534834207678), (27, 56): np.float64(-0.06555240105799401), (28, 56): np.float64(-0.07068195836856782), (31, 56): np.float64(-0.1860179445831286), (34, 56): np.float64(-0.06524217545595837), (36, 56): np.float64(-0.2774896696006215), (37, 56): np.float64(-0.11750669100706493), (38, 56): np.float64(-0.3509154114332418), (39, 56): np.float64(-0.06994382795138793), (40, 56): np.float64(-0.11663821377592108), (41, 56): np.float64(-0.08581102759585382), (42, 56): np.float64(-0.0745378964301126), (45, 56): np.float64(-0.09379047757127339), (46, 56): np.float64(-0.23430097102515665), (47, 56): np.float64(-0.09148381302491752), (51, 56): np.float64(-0.06763680862070774), (53, 56): np.float64(-0.10646547546785456), (54, 56): np.float64(-0.14943491761702524), (55, 56): np.float64(-0.12166916014818922), (56, 57): np.float64(0.13845576627089434), (56, 58): np.float64(0.09926427890491582), (56, 59): np.float64(0.11688438543602353), (56, 60): np.float64(0.06389956146436026), (56, 62): np.float64(0.274614528528526), (56, 63): np.float64(0.08532159227091335), (56, 64): np.float64(0.136464628926368), (56, 66): np.float64(0.1259001478988333), (56, 67): np.float64(0.08352968327945662), (56, 68): np.float64(0.08700726390324037), (56, 71): np.float64(0.06659817693296607), (56, 72): np.float64(0.2824273180115691), (0, 1): np.float64(0.0026736661192461055), (0, 2): np.float64(0.0010590173295066117), (0, 7): np.float64(0.000819985076294443), (0, 8): np.float64(-0.005419458408063377), (0, 9): np.float64(0.00021989240160816056), (0, 11): np.float64(0.004526432915131537), (0, 12): np.float64(-0.022374190962191655), (0, 14): np.float64(0.0015533608848212477), (0, 17): np.float64(0.0012181171727225096), (0, 19): np.float64(-0.0006987769497912263), (0, 20): np.float64(0.00907536938148033), (0, 21): np.float64(0.002529492231210641), (0, 22): np.float64(0.0010610709744848547), (0, 23): np.float64(-0.0004747516475252084), (0, 25): np.float64(-0.001436842088634863), (0, 26): np.float64(-0.0003238776443780213), (0, 27): np.float64(-0.00014351019657922706), (0, 30): np.float64(0.008034112921371286), (0, 33): np.float64(0.0017995583770316602), (0, 34): np.float64(0.0030289946709501), (0, 35): np.float64(0.0008571700314921811), (0, 37): np.float64(0.0008386912425037788), (0, 38): np.float64(0.0014436664474263084), (0, 40): np.float64(0.0038249639747841464), (0, 41): np.float64(0.004456284942381067), (0, 42): np.float64(0.0030391076896858936), (0, 43): np.float64(0.005624195694641432), (0, 44): np.float64(0.0026330444412951256), (0, 45): np.float64(0.0015953720532730575), (0, 48): np.float64(0.008506852812765999), (0, 49): np.float64(0.0010123134112487422), (0, 50): np.float64(0.00856697546175416), (0, 51): np.float64(0.005533043415708606), (0, 52): np.float64(0.001701171637505697), (0, 54): np.float64(0.0001627419476345887), (0, 57): np.float64(0.0006374890862148225), (0, 58): np.float64(0.0038196837968039244), (0, 60): np.float64(0.0012282352143572202), (0, 61): np.float64(0.001541887550680759 7), (0, 62): np.float64(-0.005408891201862026), (0, 63): np.float64(-0.0007440787455120232), (0, 64): np.float64(0.007531318319437025), (0, 65): np.float64(0.00196774403182688), (0, 66): np.float64(-0.00363037395072642), (0, 67): np.float64(-2.6308522540074957e-05), (0, 70): np.float64(0.0015552680792570261), (0, 71): np.float64(0.006612221664026873), (0, 72): np.float64(-0.0024311693893143615), (0, 74): np.float64(0.006127148767479558), (1, 2): np.float64(-0.0001609867745252804 5), (2, 5): np.float64(0.0030388462936723464), (2, 6): np.float64(0.00041406622366672413), (2, 7): np.float64(-0.0004910652051045798), (2, 8): np.float64(-0.008217892540639692), (2, 9): np.float64(-0.0016814102768358855), (2, 11): np.float64(0.00015695510944696808), (2, 14): np.float64(0.0028827920188133544), (2, 15): np.float64(-0.00021714292862138059), (2, 16): np.float64(-0.003788295130226729), (2, 17): np.float64(-0.0003436913042165164), (2, 18): np.float64(0.17622229054528968), (2, 19): np.float64(-0.002738370226923895 4), (2, 21): np.float64(0.0015804241046994318), (2, 24): np.float64(0.0012393215074590143), (2, 25): np.float64(-0.0029633980515254657), (2, 27): np.float64(-0.0013662749935518643), (2, 31): np.float64(-0.0040572861937202016), (2, 32): np.float64
```

(-0.0032278383587742523), (2, 34): np.float64(0.00023498068293785877), (2, 36): np.float64(-0.011439518176180802), (2, 37): np.float64(-0.0008784623602570049), (2, 40): np.float64(0.0007634913931756223), (2, 41): np.float64(0.0018844876276693436), (2, 42): np.float64(0.0016277910446980718), (2, 44): np.float64(0.0005707047438584099), (2, 45): np.float64(-7.075722916190859e-06), (2, 46): np.float64(-0.0028164889445418443), (2, 47): np.float64(-3.0086273682317927e-05), (2, 48): np.float64(0.00023500177077360603), (2, 52): np.float64(0.0006738910754652784), (2, 53): np.float64(0.00010019007556445884), (2, 54): np.float64(-0.004292128831278856), (2, 55): np.float64(-0.005549817819113889), (2, 57): np.float64(-0.0009250322967648394), (2, 58): np.float64(-0.00033648033206934864), (2, 59): np.float64(0.005800351179989175), (2, 62): np.float64(-0.003836196329604309), (2, 63): np.float64(-0.0029811762002085 3), (2, 64): np.float64(-8.609196702177742e-05), (2, 65): np.float64(0.0038248764889420536), (2, 67): np.float64(-0.0025513207367123753), (2, 68): np.float64(-0.00042901046089888477), (2, 70): np.float64(-0.001075004767849502), (2, 72): np.float64(-0.0043241137531582945), (2, 73): np.float64(-0.00041313343851054617), (2, 74): np.float64(0.00145117854692407 6), (1, 3): np.float64(-0.0028245077425149588), (3, 4): np.float64(0.0015953608055183325), (3, 6): np.float64(0.002428967447452 1996), (3, 7): np.float64(0.005224859808049252), (3, 8): np.float64(-0.004342551858930745), (3, 9): np.float64(0.00306676179967 93773), (3, 12): np.float64(-0.00651812342040715), (3, 13): np.float64(0.011789744225948556), (3, 14): np.float64(0.0037781409537281365), (3, 15): np.float64(0.0037173459677434448), (3, 16): np.float64(0.011004624736287601), (3, 17): np.float64(0.005316916507973344), (3, 18): np.float64(0.0758365265176233), (3, 19): np.float64(0.0018031240862689125), (3, 20): np.float64(0.009374467933634952), (3, 22): np.float64(0.0036730666286709348), (3, 23): np.float64(0.0009584353819548274), (3, 25): np.float64(0.002219437543674411), (3, 26): np.float64(0.0014163438525399863), (3, 27): np.float64(0.00131211854834354), (3, 28): np.float64(0.0037747350377072735), (3, 29): np.float64(0.004788894914267744), (3, 31): np.float64(0.0019839785727050296), (3, 32): np.float64(0.0023164628413622464), (3, 34): np.float64(0.0032544819210895406), (3, 35): np.float64(0.002820338162607733), (3, 36): np.float64(-0.0039032912025129974), (3, 37): np.float64(0.001495026910530462), (3, 38): np.float64(0.0041373858144154235), (3, 41): np.float64(0.01253992855660729), (3, 42): np.float64(0.02586571516922783), (3, 43): np.float64(0.004206027950930067), (3, 44): np.float64(0.0036152972118568267), (3, 47): np.float64(0.018086723469493), (3, 48): np.float64(0.00408387240085027), (3, 49): np.float64(0.005723339076165178), (3, 50): np.float64(0.0047832510433894115), (3, 52): np.float64(0.01053662522168614), (3, 53): np.float64(0.002957697849152678), (3, 54): np.float64(0.0027718684570441415), (3, 55): np.float64(0.0004271449960018064), (3, 57): np.float64(0.0032503913007684527), (3, 58): np.float64(0.008915211590496679), (3, 59): np.float64(0.002909847108130256), (3, 65): np.float64(0.003250220168576529), (3, 66): np.float64(-0.0003742266556804357), (3, 67): np.float64(0.0015359718612757954), (3, 70): np.float64(0.002006106474500516), (3, 72): np.float64(-0.0005921274995956386), (1, 8): np.float64(-0.004980744460359554), (4, 8): np.float64(-0.02139638386426646), (5, 8): np.float64(-0.010266917388600957), (7, 8): np.float64(-0.007232734267359434), (8, 9): np.float64(0.005372097650416957), (8, 10): np.float64(0.005139315829823384), (8, 14): np.float64(0.034297493547116774), (8, 16): np.float64(0.009122621301227993), (8, 19): np.float64(0.004442698562698235), (8, 20): np.float64(0.008208017347474433), (8, 21): np.float64(0.006801358466851831), (8, 24): np.float64(0.007037776762554862), (8, 25): np.float64(0.012500345515811353), (8, 26): np.float64(0.02064719670227873), (8, 28): np.float64(0.005252962600585989), (8, 29): np.float64(0.04840260844848495), (8, 30): np.float64(0.012850293369604962), (8, 32): np.float64(0.004150822086744938), (8, 33): np.float64(0.007025824884325039), (8, 35): np.float64(0.008945402582728438), (8, 36): np.float64(0.0024598469608709786), (8, 39): np.float64(0.008716501757492428), (8, 40): np.float64(0.008638103614790445), (8, 41): np.float64(0.006055703209014079), (8, 45): np.float64(0.009993593339974509), (8, 48): np.float64(0.004837183056327074), (8, 49): np.float64(0.004499994079176808), (8, 52): np.float64(0.01840912899986592), (8, 54): np.float64(0.022284835854985266), (8, 55): np.float64(0.014580330399416298), (8, 58): np.float64(0.02259285189986506), (8, 62): np.float64(0.0032557979179102), (8, 66): np.float64(0.013734103760660634), (8, 67): np.float64(0.008896376663367626), (8, 68): np.float64(0.004726921072436226), (8, 69): np.float64(0.04144040084430689), (8, 70): np.float64(0.004866322337584683), (8, 71): np.float64(0.02040905692215735), (8, 72): np.float64(0.009801689550995089), (8, 73): np.float64(0.005737109428705006), (8, 74): np.float64(0.007648079977903755), (4, 9): np.float64(-0.0004602990897775441), (6, 9): np.f

loat64(-0.0033213619653970077), (7, 9): np.float64(-0.001457362359633139), (9, 10): np.float64(-7.113681835234362e-05), (9, 1
1): np.float64(0.0013593951523423065), (9, 14): np.float64(0.0015956185154246008), (9, 15): np.float64(0.0025898716562744967),
(9, 17): np.float64(0.0028030955909849557), (9, 21): np.float64(0.0031050412218336866), (9, 23): np.float64(-0.0002312754376627
964), (9, 24): np.float64(0.0013728985226274268), (9, 25): np.float64(-0.0009158226551348283), (9, 26): np.float64(-0.000319017
83032021406), (9, 27): np.float64(-0.00013334539068390286), (9, 29): np.float64(0.006098447078402701), (9, 31): np.float64(-0.0
0300076494169507806), (9, 33): np.float64(0.0009473943241448347), (9, 34): np.float64(0.0021476312515101843), (9, 35): np.float
64(0.0006507829172896235), (9, 36): np.float64(-0.0023676893316003293), (9, 37): np.float64(0.0007068891950931473), (9, 38): n
p.float64(0.0005163713006835768), (9, 40): np.float64(0.002867751082065713), (9, 41): np.float64(0.002633173542811306), (9, 4
2): np.float64(0.0061813914314326444), (9, 43): np.float64(0.0016780425519107582), (9, 47): np.float64(0.0020910558736306865),
(9, 48): np.float64(0.010509437425538469), (9, 49): np.float64(0.0006579142298411341), (9, 51): np.float64(0.0045256230164377 5
2), (9, 52): np.float64(0.004052964976343161), (9, 53): np.float64(0.0010448293130289445), (9, 57): np.float64(0.00030133748156
80466), (9, 58): np.float64(0.0009943438544329811), (9, 61): np.float64(0.0014140600612380727), (9, 63): np.float64(-0.00126103
94206658715), (9, 65): np.float64(0.0044322785689286485), (9, 66): np.float64(-0.0017885782375766214), (9, 68): np.float64(0.00
07105685734565324), (9, 69): np.float64(0.0018429144434683983), (9, 71): np.float64(0.004568250128382461), (1, 10): np.float64
(-0.006563278460263298), (5, 10): np.float64(-0.004178762690761421), (6, 10): np.float64(-0.0015366999232606933), (7, 10): np.f
loat64(-0.0007755936293068836), (10, 11): np.float64(0.006033201784745199), (10, 12): np.float64(-0.006232558863724032), (10, 1
4): np.float64(0.009068985725825944), (10, 15): np.float64(0.0016476773007032327), (10, 16): np.float64(0.0010786224705712315),
(10, 17): np.float64(0.0008802025079712489), (10, 19): np.float64(-0.0012138747035625125), (10, 20): np.float64(0.0052887239173
88321), (10, 23): np.float64(-0.0005587253424712567), (10, 24): np.float64(0.0018137663856881657), (10, 25): np.float64(-0.0005
612253635337458), (10, 26): np.float64(-0.0003948578173591781), (10, 27): np.float64(-0.00024599476636881696), (10, 28): np.flo
at64(0.00015181382516121856), (10, 32): np.float64(0.002876690686817546), (10, 33): np.float64(0.0010598891953154106), (10, 3
4): np.float64(0.001505189096413558), (10, 35): np.float64(0.0032801012691601864), (10, 36): np.float64(-0.01010097940346904),
(10, 37): np.float64(0.00020201768288341008), (10, 38): np.float64(0.0012232669926213408), (10, 40): np.float64(0.0038150270369
453694), (10, 43): np.float64(0.0019044047822307363), (10, 44): np.float64(0.0015667145079723754), (10, 45): np.float64(0.00103
3595369781405), (10, 46): np.float64(-0.0019557782856758346), (10, 47): np.float64(0.002290877690667081), (10, 49): np.float64
(0.003797764674957843), (10, 50): np.float64(0.0027293274335156146), (10, 51): np.float64(0.0014731562608209531), (10, 55): np.
float64(-0.001392719129551191), (10, 58): np.float64(0.0036394038806257557), (10, 61): np.float64(0.0014270658659138904), (10,
67): np.float64(-7.159852649825789e-05), (10, 68): np.float64(0.0011795856115887652), (10, 69): np.float64(0.0103569676161330 2
1), (10, 73): np.float64(0.0034781690137950494), (10, 74): np.float64(0.007982115999082775), (4, 11): np.float64(0.000678343245
9312397), (6, 11): np.float64(-0.00021044716316134796), (7, 11): np.float64(0.0013554422248327576), (11, 13): np.float64(-0.000
87977277830117), (11, 14): np.float64(0.000360545897771964), (11, 15): np.float64(-0.0001902129362703744), (11, 17): np.float64
(-0.00031814595129634057), (11, 18): np.float64(0.1541050676932936), (11, 19): np.float64(-0.004489310133976894), (11, 20): np.
float64(-1.4136278727251933e-05), (11, 21): np.float64(0.004557414801391857), (11, 24): np.float64(0.0003370567484538488), (11,
25): np.float64(-0.013222981889200545), (11, 28): np.float64(-0.001015120984202116), (11, 29): np.float64(0.01040818705964555 3
), (11, 30): np.float64(0.003890825124910071), (11, 31): np.float64(-0.0022912107388304735), (11, 32): np.float64(-0.000626471
0711790222), (11, 33): np.float64(-0.0012534607524713928), (11, 34): np.float64(1.4513168939561457e-05), (11, 35): np.float64(-
0.0005454694543480437), (11, 40): np.float64(0.0009733087531983166), (11, 41): np.float64(0.0017805097826946198), (11, 42): np.
float64(0.0018774974438353477), (11, 43): np.float64(0.0009526690776860439), (11, 45): np.float64(-0.00011730625081414169), (1
1, 48): np.float64(0.00018882969451030672), (11, 50): np.float64(0.001189136760171106), (11, 52): np.float64(0.0037954141097655
374), (11, 53): np.float64(-0.00010597782373073085), (11, 54): np.float64(-0.0009738883093961631), (11, 55): np.float64(-0.0019
634896628346653), (11, 58): np.float64(-0.0005628080145623722), (11, 61): np.float64(-0.0009242119805139566), (11, 62): np.floa

t64(-0.008254842170495427), (11, 63): np.float64(-0.002888194620811851), (11, 65): np.float64(0.0017200761193585572), (11, 66): np.float64(-0.0036729981283586837), (11, 67): np.float64(-0.00651001617006004), (11, 68): np.float64(-0.0004944516195633791), (11, 69): np.float64(0.0004677613977345152), (11, 70): np.float64(-0.0006813942763447406), (11, 71): np.float64(0.004580386836781311), (11, 73): np.float64(-0.0010005939342660854), (1, 12): np.float64(-0.016555402339263857), (4, 12): np.float64(-0.005789295927154897), (5, 12): np.float64(-0.012317176535972717), (7, 12): np.float64(-0.008138648931107667), (12, 16): np.float64(0.004724913922266055), (12, 17): np.float64(0.005846849325633016), (12, 20): np.float64(0.01819609391560136), (12, 21): np.float64(0.006379437252850069), (12, 23): np.float64(0.003890154689466959), (12, 24): np.float64(0.005049190793799404), (12, 25): np.float64(0.004020756510859663), (12, 26): np.float64(0.005093269888207195), (12, 27): np.float64(0.007038742636177505), (12, 29): np.float64(0.006323109864152637), (12, 30): np.float64(0.009101979363282298), (12, 31): np.float64(0.0225867222735125), (12, 34): np.float64(0.036400576160381115), (12, 35): np.float64(0.009035128635655145), (12, 38): np.float64(0.004738993564087851), (12, 39): np.float64(0.004462807275589627), (12, 40): np.float64(0.021458168343185015), (12, 42): np.float64(0.009230178554928714), (12, 43): np.float64(0.011246926780651492), (12, 46): np.float64(0.001927115300843857), (12, 47): np.float64(0.005130468694032760), (12, 48): np.float64(0.005398918030621645), (12, 51): np.float64(0.008210203300160783), (12, 52): np.float64(0.010626556316709813), (12, 53): np.float64(0.012881262337543695), (12, 59): np.float64(0.005733452378833681), (12, 61): np.float64(0.005016797330211627), (12, 62): np.float64(0.0022726415068147726), (12, 63): np.float64(0.003010133328307847), (12, 65): np.float64(0.008691929345181313), (12, 66): np.float64(0.00277554386002102), (12, 68): np.float64(0.004221989192098212), (12, 70): np.float64(0.004624676119055665), (12, 72): np.float64(0.0060992150346157945), (12, 73): np.float64(0.008143326783535985), (12, 74): np.float64(0.023302983484378894), (4, 13): np.float64(3.832999214664321e-05), (5, 13): np.float64(-0.004700368904273982), (6, 13): np.float64(-0.0008820218578711564), (13, 14): np.float64(0.0009181270371610916), (13, 15): np.float64(0.000748392156130040 3), (13, 16): np.float64(0.00013062738585582648), (13, 17): np.float64(0.0004182987955755074), (13, 18): np.float64(0.10361639323566585), (13, 19): np.float64(-0.00237079557870541), (13, 26): np.float64(-0.002828665746803719), (13, 28): np.float64(-0.0005473796827211506), (13, 29): np.float64(0.003602350301802516), (13, 31): np.float64(-0.0008996114962594437), (13, 33): np.float64(0.0006924649893221884), (13, 36): np.float64(-0.002627431699203369), (13, 38): np.float64(6.340879618220996e-05), (13, 39): np.float64(0.0008573330349465609), (13, 40): np.float64(0.0013021230579704124), (13, 41): np.float64(0.0038620641173599322), (13, 43): np.float64(0.0049272119061612284), (13, 45): np.float64(0.004162154584742896), (13, 46): np.float64(-0.002588703165813 7), (13, 47): np.float64(0.0007721906005254535), (13, 48): np.float64(0.0009816437753549995), (13, 50): np.float64(0.005778399847234 2), (13, 51): np.float64(0.0017560437520443453), (13, 53): np.float64(0.004992678546652668), (13, 54): np.float64(-0.0007903301163103675), (13, 55): np.float64(-0.0018505362070807216), (13, 59): np.float64(0.002313537101742627), (13, 60): np.float64(0.0004301753436052464), (13, 61): np.float64(0.0006745393860340324), (13, 62): np.float64(-0.002881464335761985), (13, 64): np.float64(0.0006210333997095857), (13, 65): np.float64(0.002891384069094361), (13, 67): np.float64(-0.0007644305299850232), (13, 69): np.float64(0.0012144072583460434), (13, 70): np.float64(0.00011076060353488559), (13, 71): np.float64(0.014072763458921339), (13, 73): np.float64(0.0002390111063027 3028), (1, 16): np.float64(-0.0014285294182927977), (7, 16): np.float64(-0.0005635288725790475), (14, 16): np.float64(-0.002527573012252749), (16, 17): np.float64(0.00040168388448539737), (16, 18): np.float64(0.09556503010168793), (16, 19): np.float64(-0.0012850784930528666), (16, 20): np.float64(0.0010018474989499525), (16, 21): np.float64(0.010856497157679324), (16, 24): np.float64(0.0015220818481345572), (16, 25): np.float64(-0.0014405136203986858), (16, 27): np.float64(-0.0007519862322146025), (16, 28): np.float64(-0.0008710984041629734), (16, 30): np.float64(0.018953864998926 22), (16, 31): np.float64(-0.002124973642901354), (16, 34): np.float64(0.0013373860986587059), (16, 35): np.float64(4.649267585477765e-05), (16, 36): np.float64(-0.0037768099185031363), (16, 37): np.float64(-0.0008011993400961967), (16, 38): np.float64(-2.1318692750802787e-06), (16, 39): np.float64(0.0007418038088006143), (16, 40): np.float64(0.003564958144196485), (16, 41): np.float64(0.009126690127586897), (16, 42): np.float64(0.00810472439971492), (16, 45): np.float64(0.0015063717223375407), (16, 46): np.float64(-0.008413923073455764), (16, 48): np.float64(0.002539138072498859), (16, 49): np.float64(0.0003609187205636089),

(16, 51): np.float64(0.0013716346621186965), (16, 52): np.float64(0.001209205081167852), (16, 55): np.float64(-0.007011739753198178), (16, 58): np.float64(0.00040839089996669607), (16, 59): np.float64(0.001120077257069156), (16, 61): np.float64(0.0011896915429961531), (16, 62): np.float64(-0.0036261433509805377), (16, 64): np.float64(0.00047316858451695317), (16, 65): np.float64(0.0014575206157194183), (16, 66): np.float64(-0.0020764756684675863), (16, 67): np.float64(-0.0016391249622976906), (16, 68): np.float64(0.00031134454533547385), (16, 69): np.float64(0.001459195128265321), (16, 70): np.float64(7.025079416441676e-05), (16, 72): np.float64(-0.0032697515107132156), (1, 18): np.float64(0.07959027252147685), (5, 18): np.float64(0.23609028696203305), (6, 18): np.float64(0.08213971363965783), (14, 18): np.float64(0.13079106879372604), (18, 19): np.float64(-0.08141788143819204), (18, 20): np.float64(-0.069606397389677), (18, 21): np.float64(-0.3116577968758403), (18, 24): np.float64(-0.0885178449159614), (18, 26): np.float64(-0.0726061748906325), (18, 29): np.float64(-0.25068588794805335), (18, 30): np.float64(-0.39971675738901197), (18, 31): np.float64(-0.08959522315390943), (18, 34): np.float64(-0.1092244948537039), (18, 37): np.float64(-0.07963891303733116), (18, 38): np.float64(-0.3010126769446561), (18, 39): np.float64(-0.08273032503316434), (18, 40): np.float64(-0.23749866599452066), (18, 41): np.float64(-0.26776775241209805), (18, 42): np.float64(-0.2664615721630717), (18, 43): np.float64(-0.08032245874225108), (18, 44): np.float64(-0.09590868908823638), (18, 45): np.float64(-0.118230315384218), (18, 47): np.float64(-0.15958654440254386), (18, 48): np.float64(-0.07564509051090863), (18, 50): np.float64(-0.14633622264434007), (18, 51): np.float64(-0.08098060447603962), (18, 52): np.float64(-0.13502694824391456), (18, 53): np.float64(-0.1535353792558341), (18, 54): np.float64(-0.13233254539707666), (18, 57): np.float64(-0.08831308644706885), (18, 58): np.float64(-0.12267827735347474), (18, 59): np.float64(-0.21639040439312768), (18, 60): np.float64(-0.0743301389722173), (18, 64): np.float64(-0.1515254940979447), (18, 65): np.float64(-0.11317764865413402), (18, 68): np.float64(-0.0741256045266795), (18, 69): np.float64(-0.10114781648263371), (18, 71): np.float64(-0.31546006003971916), (18, 72): np.float64(-0.07393410958437824), (18, 74): np.float64(-0.1005494383464481), (18, 81): np.float64(6.803979985627775), (5, 19): np.float64(-0.00521433344641369), (14, 19): np.float64(-0.0019158856011329097), (19, 21): np.float64(0.013207200092186209), (19, 22): np.float64(0.0017385273110909283), (19, 23): np.float64(0.0005728629591654578), (19, 24): np.float64(0.003325575937335503), (19, 28): np.float64(0.0006807207136015807), (19, 31): np.float64(0.0003532357040786854), (19, 32): np.float64(0.004147065761337745), (19, 33): np.float64(0.003665010136991103), (19, 36): np.float64(-0.003944472285242039), (19, 37): np.float64(0.0009152682278725109), (19, 38): np.float64(0.007982336734251888), (19, 41): np.float64(0.004552046447603976), (19, 42): np.float64(0.0037734129837662246), (19, 43): np.float64(0.0023488170393070633), (19, 45): np.float64(0.0037397144533328763), (19, 46): np.float64(-0.0012118681317522927), (19, 47): np.float64(0.007564385929688837), (19, 50): np.float64(0.0031174842483175005), (19, 51): np.float64(0.006484258334034929), (19, 52): np.float64(0.004672837101162029), (19, 53): np.float64(0.001915344358045026), (19, 55): np.float64(-0.0007076883416471193), (19, 58): np.float64(0.016438352368489 11), (19, 59): np.float64(0.0028655876813288733), (19, 60): np.float64(0.002059525886776715), (19, 62): np.float64(-0.0008554059905702579), (19, 63): np.float64(0.00018552164741725117), (19, 64): np.float64(0.004596814481742787), (19, 67): np.float64(0.0012085484710142016), (19, 68): np.float64(0.004252458836807809), (19, 69): np.float64(0.0022507193712830572), (19, 70): np.float64(0.0017710919853592558), (19, 71): np.float64(0.009270059337266235), (19, 72): np.float64(-0.0017916912277736305), (19, 73): np.float64(0.0012855073468807134), (19, 74): np.float64(0.009021511964594442), (1, 21): np.float64(0.003069190075233727), (4, 21): np.float64(0.0022155558467722588), (5, 21): np.float64(-0.0026528697261048104), (14, 21): np.float64(0.002765304425950253), (15, 21): np.float64(0.004272024442094168), (17, 21): np.float64(0.0027344282646598996), (20, 21): np.float64(0.002207310324395239), (21, 23): np.float64(-0.006304775491243737), (21, 24): np.float64(-0.0013708108145320756), (21, 26): np.float64(-0.0050233893385656205), (21, 27): np.float64(-0.008322636152987766), (21, 30): np.float64(0.004212926751002715), (21, 31): np.float64(-0.01186281696697551), (21, 33): np.float64(-0.00611194625439986), (21, 34): np.float64(-0.002486956794425366), (21, 35): np.float64(-0.0039897833580966775), (21, 37): np.float64(-0.002460389571165163), (21, 39): np.float64(-0.003417363225058875), (21, 40): np.float64(-0.001312878395219867), (21, 41): np.float64(-0.00027834328917927066), (21, 42): np.float64(-8.470791089880718e-05), (21, 44): np.float64(-0.0014043421358264532), (21, 46): np.float64(-0.015344330674031651), (21, 50): n

p.float64(-0.00030001637233889654), (21, 53): np.float64(-0.006031976100269802), (21, 59): np.float64(-0.0047090027391318325), (21, 60): np.float64(-0.00158627045967737), (21, 62): np.float64(-0.006514713345823684), (21, 65): np.float64(-0.0010282945355768 89565), (21, 72): np.float64(-0.024321050601654622), (21, 74): np.float64(-0.0010696080225126932), (4, 23): np.float64(-0.00089 81434951079824), (5, 23): np.float64(-0.005191243187169167), (6, 23): np.float64(-0.002045085235882113), (14, 23): np.float64(-0.004759021483502247), (15, 23): np.float64(-0.0014318559452013776), (17, 23): np.float64(-0.0011595829060784047), (20, 23): np.float64(-0.0018666059715152841), (22, 23): np.float64(-0.003597121301086571), (23, 24): np.float64(0.008013698142064289), (23, 25): np.float64(-0.0002934537651038683), (23, 26): np.float64(0.00020082976251660088), (23, 27): np.float64(0.0001229956094699658), (23, 28): np.float64(0.0007594140179697219), (23, 31): np.float64(-0.00047997135853393777), (23, 32): np.float64(0.0018937461275328493), (23, 33): np.float64(0.0013569108794336295), (23, 34): np.float64(0.0014193568940800477), (23, 35): np.float64(0.0008214556041544059), (23, 37): np.float64(0.0007156060514635508), (23, 39): np.float64(0.0016296973310724794), (23, 40): np.float64(0.006263537319231305), (23, 45): np.float64(0.003156254368971192), (23, 46): np.float64(-0.001537327355269903), (23, 47): np.float64(0.010139791763128778), (23, 50): np.float64(0.013052209055590997), (23, 52): np.float64(0.006134203497336271), (23, 53): np.float64(0.009597582466699232), (23, 55): np.float64(-0.002063244345592397), (23, 57): np.float64(0.0007860104587121394), (23, 59): np.float64(0.0019554611449186013), (23, 61): np.float64(0.0010807602874582864), (23, 62): np.float64(-0.00244524469384223), (23, 63): np.float64(-0.0008030762824845268), (23, 64): np.float64(0.0025455452584800782), (23, 65): np.float64(0.003081676490358875), (23, 67): np.float64(0.00035130338553698853), (23, 68): np.float64(0.001123227248599539), (23, 71): np.float64(0.008312902265094561), (23, 72): np.float64(-0.002180047689884699), (23, 73): np.float64(0.0017208350070943138), (1, 25): np.float64(-0.004385642983118942), (7, 25): np.float64(-0.0011743709693582197), (17, 25): np.float64(-0.002963446095528055), (20, 25): np.float64(-0.0016436448241050564), (22, 25): np.float64(-0.001409164825557083), (24, 25): np.float64(-0.004911646002912913), (25, 26): np.float64(0.0004193993297049982), (25, 27): np.float64(0.0012611436446389676), (25, 28): np.float64(0.0005193476980961458), (25, 31): np.float64(0.00031381572467613345), (25, 32): np.float64(0.0018983665694841422), (25, 33): np.float64(0.0022192065786857647), (25, 34): np.float64(0.010372088007042288), (25, 36): np.float64(-0.005659654860975242), (25, 39): np.float64(0.001810290232392849), (25, 40): np.float64(0.011588996425597558), (25, 41): np.float64(0.0041279228390584), (25, 43): np.float64(0.006084319704654814), (25, 46): np.float64(-0.002560779658974283), (25, 47): np.float64(0.003816423675600155), (25, 48): np.float64(0.0037731718083283895), (25, 49): np.float64(0.0013833623441064117), (25, 51): np.float64(0.003418683371853372 4), (25, 57): np.float64(0.0011682015001112263), (25, 58): np.float64(0.0015725563498667253), (25, 60): np.float64(0.0013338277283372856), (25, 62): np.float64(-0.0009958436426927536), (25, 63): np.float64(-9.269438906875241e-05), (25, 64): np.float64(0.010658947730234942), (25, 66): np.float64(-0.0024692058044469486), (25, 67): np.float64(0.0005671990235893382), (25, 69): np.float64(0.01318607091206319), (25, 70): np.float64(0.0017753826094162339), (25, 71): np.float64(0.006489686446316403), (25, 74): np.float64(0.00501695935549899), (4, 26): np.float64(-0.0031664807207161107), (6, 26): np.float64(-0.001842740613892124 7), (7, 26): np.float64(-0.0009747675616502209), (14, 26): np.float64(-0.008629344365809552), (17, 26): np.float64(-0.001794737292435613 3), (22, 26): np.float64(-0.0034673894656255403), (26, 28): np.float64(0.00023478651240991028), (26, 29): np.float64(0.013366923483301546), (26, 31): np.float64(-0.0008549348617748687), (26, 34): np.float64(0.0015656781966128133), (26, 35): np.float64(0.0011788447803629952), (26, 37): np.float64(0.00038219144243956766), (26, 39): np.float64(0.0022194489978222477), (26, 40): np.float64(0.005375197822657286), (26, 41): np.float64(0.0028196836544163726), (26, 43): np.float64(0.00202443431586157), (26, 45): np.float64(0.0015112444476747323), (26, 46): np.float64(-0.004192879388133546), (26, 47): np.float64(0.001711446349989196 6), (26, 48): np.float64(0.0030412736970062133), (26, 49): np.float64(0.00405979163662877), (26, 51): np.float64(0.0010405262821043316), (26, 53): np.float64(0.005093667085202604), (26, 55): np.float64(-0.0006235148297410886), (26, 57): np.float64(0.001216588524991101), (26, 60): np.float64(0.001642016696280183), (26, 61): np.float64(0.0009606898441267019), (26, 66): np.float64(-0.0014791921199866784), (26, 67): np.float64(8.176543770886991e-05), (26, 68): np.float64(0.0009971400108460635), (26, 70): np.float64(0.0009372671492821081), (26, 73): np.float64(0.0009200541147256923), (26, 74): np.float64(0.00529080530642057

6), (1, 27): np.float64(-0.0020771822223834098), (4, 27): np.float64(-0.0021356587131778956), (5, 27): np.float64(-0.0036456521993274227), (14, 27): np.float64(-0.0065739479283798636), (15, 27): np.float64(-0.0035465972859230754), (20, 27): np.float64(-0.0012027974999383544), (27, 29): np.float64(0.004786265055896576), (27, 32): np.float64(0.001736189778630221), (27, 33): np.float64(0.0009436552178296766), (27, 37): np.float64(0.00025568156920450136), (27, 38): np.float64(0.004263206512048354), (27, 40): np.float64(0.0023092395805271704), (27, 45): np.float64(0.001140523854510457), (27, 46): np.float64(-0.00563182408016164), (27, 47): np.float64(0.0017149228576644321), (27, 48): np.float64(0.0013636868693045845), (27, 49): np.float64(0.005426225719602117), (27, 50): np.float64(0.003057852364613171), (27, 52): np.float64(0.004560650088573396), (27, 53): np.float64(0.0011850152815681666), (27, 54): np.float64(0.00021700736258834606), (27, 57): np.float64(0.000554610402354213), (27, 61): np.float64(0.0009346867539123 44), (27, 63): np.float64(-0.00040968375778397504), (27, 64): np.float64(0.002685681950856788), (27, 65): np.float64(0.002199464880968457), (27, 67): np.float64(0.00026501905380785023), (27, 70): np.float64(0.0010138939312740356), (27, 73): np.float64(0.002485209699170729), (27, 74): np.float64(0.00808463696804541), (1, 28): np.float64(-0.004708740866784062), (5, 28): np.float64(-0.0034386979377260887), (6, 28): np.float64(-0.0014237504365571636), (14, 28): np.float64(-0.005880571922609833), (22, 28): np.float64(-0.002947252450845924), (24, 28): np.float64(-0.0013774027814699148), (28, 30): np.float64(0.026570333394011676), (28, 31): np.float64(-0.0004537006449025775), (28, 32): np.float64(0.0008995043880994502), (28, 35): np.float64(0.0006077477796573218), (28, 36): np.float64(-0.011914718735078334), (28, 37): np.float64(0.0002309561634490656), (28, 39): np.float64(0.001544100301119944), (28, 41): np.float64(0.002678785323123826), (28, 42): np.float64(0.006076840878051099), (28, 43): np.float64(0.0022200941433999748), (28, 44): np.float64(0.001539343716728004), (28, 46): np.float64(-0.002495321229052144), (28, 52): np.float64(0.004916247279285533), (28, 53): np.float64(0.007791082890792755), (28, 58): np.float64(0.00518041279378137), (28, 59): np.float64(0.0017220304468907705), (28, 61): np.float64(0.001153485448419903), (28, 63): np.float64(-0.0005359336691059156), (28, 64): np.float64(0.001081558421170371), (28, 66): np.float64(-0.0013115273494767252), (28, 67): np.float64(-0.0002859392560134096 6), (28, 69): np.float64(0.0018337942328657429), (28, 71): np.float64(0.007917082781293636), (28, 74): np.float64(0.0028794638085200444), (1, 31): np.float64(-0.008365141965138718), (4, 31): np.float64(-0.000770046707487762), (5, 31): np.float64(-0.003926670396423412), (6, 31): np.float64(-0.003690713359245715), (7, 31): np.float64(-0.003535629922258035), (14, 31): np.float64(-0.0070925408801793734), (17, 31): np.float64(-0.004966598008074642), (22, 31): np.float64(-0.0021533649674153896), (24, 31): np.float64(-0.0018638258515671466), (31, 35): np.float64(0.0008844280435659705), (31, 39): np.float64(0.0036391627991097116), (31, 40): np.float64(0.008648002569970795), (31, 41): np.float64(0.002782530077059379), (31, 44): np.float64(0.0017821962481903602), (31, 45): np.float64(0.0014148665332075032), (31, 47): np.float64(0.0014767535853792949), (31, 48): np.float64(0.0020812474124117337), (31, 49): np.float64(0.0014836397707207792), (31, 50): np.float64(0.013464959663258419), (31, 54): np.float64(0.0008273258062439283), (31, 55): np.float64(-0.0010387272059208346), (31, 57): np.float64(0.0006226702556113163), (31, 58): np.float64(0.001733321909102732), (31, 59): np.float64(0.010740769087870784), (31, 62): np.float64(-0.00303796868636042), (31, 64): np.float64(0.001699991132039885), (31, 65): np.float64(0.012083646148951402), (31, 66): np.float64(-0.0011308374800431296), (31, 67): np.float64(0.00038641087659442943), (31, 68): np.float64(0.0015665487250334082), (31, 70): np.float64(0.0012966525184340697), (31, 72): np.float64(-0.0014369191234822687), (5, 34): np.float64(-0.020189692128048203), (7, 34): np.float64(0.0006386683101694956), (14, 34): np.float64(-0.0003981369045736419), (15, 34): np.float64(0.00023798135875417322), (17, 34): np.float64(0.0008783462854277501), (20, 34): np.float64(2.8716587583881173e-05), (22, 34): np.float64(0.00024531860754913883), (24, 34): np.float64(-0.00038690852333396535), (32, 34): np.float64(0.0008887316452961218), (33, 34): np.float64(0.000541033297546519), (34, 35): np.float64(-0.0028039558312507435), (34, 37): np.float64(-0.0009620034229062186), (34, 39): np.float64(-0.0001281493909621241), (34, 40): np.float64(0.0033955662786245453), (34, 41): np.float64(0.003176206079340514), (34, 43): np.float64(0.0006459653776849913), (34, 44): np.float64(0.0023975795130232362), (34, 49): np.float64(-0.0007084876160498089), (34, 50): np.float64(0.002612890386801908), (34, 51): np.float64(-0.0003332810276368445), (34, 52): np.float64(0.0037267327805838813), (34, 54): np.float64(-0.00106188104854792), (34, 55): np.float64(-0.0018217221683748002), (34, 57): np.float64(-0.00110

79315286421968), (34, 58): np.float64(-0.0017462902180722256), (34, 61): np.float64(-0.0004240980952343887), (34, 64): np.float64(-0.0009055579780135966), (34, 65): np.float64(0.005923845395813239), (34, 66): np.float64(-0.005892348238749878), (34, 68): np.float64(-0.0009110079132875051), (34, 69): np.float64(0.0005713255402916687), (34, 70): np.float64(-0.0007062941137119511), (34, 72): np.float64(-0.005059051708882417), (34, 73): np.float64(-0.001119046112895177), (34, 74): np.float64(0.000536653554228419), (1, 36): np.float64(-0.010616671057468446), (4, 36): np.float64(-0.011824726047047405), (6, 36): np.float64(-0.01654743879970995), (20, 36): np.float64(-0.0046349267072574585), (22, 36): np.float64(-0.004708152482610619), (24, 36): np.float64(-0.026665637427787502), (29, 36): np.float64(-0.004387875339505709), (30, 36): np.float64(-0.009876208145383807), (35, 36): np.float64(-0.0048587378451534265), (36, 37): np.float64(0.01547356389565392), (36, 38): np.float64(0.0033732316967350654), (36, 39): np.float64(0.0064781334135107374), (36, 40): np.float64(0.009139169296067869), (36, 42): np.float64(0.008068805833715302), (36, 45): np.float64(0.0028844918176609913), (36, 46): np.float64(0.0009726568714751805), (36, 48): np.float64(0.0034001184467638613), (36, 49): np.float64(0.0030622288894146493), (36, 51): np.float64(0.0033622052029226427), (36, 53): np.float64(0.0033040727560086428), (36, 54): np.float64(0.0024836647146531943), (36, 55): np.float64(0.0016732046339703243), (36, 57): np.float64(0.015705392155367606), (36, 59): np.float64(0.0036824717215362505), (36, 60): np.float64(0.004077887994379685), (36, 61): np.float64(0.0031298974205485348), (36, 64): np.float64(0.0052083862004480026), (36, 66): np.float64(0.0028369621796518326), (36, 67): np.float64(0.008339113732511684), (36, 68): np.float64(0.003342844686950926), (36, 70): np.float64(0.0048917225069546036), (36, 71): np.float64(0.019780639732223174), (36, 72): np.float64(0.0006497112973546441), (1, 37): np.float64(-0.009330866174555561), (6, 37): np.float64(-0.0017301127728077373), (7, 37): np.float64(-0.0012304296336370462), (14, 37): np.float64(-0.00446395530984341), (15, 37): np.float64(-0.0011883699128909363), (20, 37): np.float64(-0.001372206988497816), (22, 37): np.float64(-0.0014332936636459234), (29, 37): np.float64(-0.002961772829508872), (30, 37): np.float64(-0.00785260406151246), (32, 37): np.float64(-0.0008123563050973849), (33, 37): np.float64(-0.0008975703589322939), (35, 37): np.float64(-0.0007280430968565465), (37, 44): np.float64(0.008192916000075257), (37, 45): np.float64(0.0013161564406832117), (37, 46): np.float64(-0.0050315488675281685), (37, 48): np.float64(0.0012437653203864008), (37, 49): np.float64(0.0012274209458821362), (37, 51): np.float64(0.0013499796825410394), (37, 52): np.float64(0.012979993552385771), (37, 54): np.float64(-0.00019039714404740554), (37, 55): np.float64(-0.001920819623996029), (37, 57): np.float64(7.711635137016008e-05), (37, 58): np.float64(0.0014581531061325331), (37, 59): np.float64(0.001983790445237568), (37, 60): np.float64(0.0007893976841445027), (37, 63): np.float64(-0.0007422383347085904), (37, 68): np.float64(0.0030659162879277087), (37, 69): np.float64(0.0025559401211637913), (37, 72): np.float64(-0.008375507418961706), (5, 38): np.float64(-0.00591861891197058), (6, 38): np.float64(-0.00558122478078199), (17, 38): np.float64(-0.0010750892661009798), (24, 38): np.float64(-0.0026040293371335636), (29, 38): np.float64(-0.004221296836850082), (30, 38): np.float64(-0.013243760927430528), (33, 38): np.float64(-0.0014532421154099865), (35, 38): np.float64(-0.0001366272188271461), (38, 41): np.float64(0.005508480856475188), (38, 42): np.float64(0.0019509905509109858), (38, 43): np.float64(0.004596678479480714), (38, 45): np.float64(0.0008856049999400603), (38, 46): np.float64(-0.003132599642839123), (38, 47): np.float64(0.0006425397726445717), (38, 48): np.float64(0.0011303388908512615), (38, 50): np.float64(0.01514366046023071), (38, 52): np.float64(0.0027905539062893833), (38, 55): np.float64(-0.0021298853028389035), (38, 57): np.float64(-0.0006868542481139096), (38, 58): np.float64(0.001492015771130892), (38, 59): np.float64(0.006469162872560578), (38, 60): np.float64(0.0007769215574490447), (38, 62): np.float64(-0.007559652029435026), (38, 65): np.float64(0.0021009019282175675), (38, 66): np.float64(-0.002404530743561304), (38, 68): np.float64(9.886911153211135e-05), (38, 70): np.float64(0.00024235273580794773), (38, 71): np.float64(0.012890921826064838), (38, 73): np.float64(0.0009550322436914132), (38, 74): np.float64(0.0016470770872434844), (1, 39): np.float64(-0.00019661031596486232), (4, 39): np.float64(0.0007660907687658647), (5, 39): np.float64(-0.00246598480952032), (6, 39): np.float64(-0.0004969737287019828), (22, 39): np.float64(0.0002248951535792356), (24, 39): np.float64(-0.000696828596865018), (32, 39): np.float64(0.0006051394807596426), (39, 41): np.float64(0.0017260858299998013), (39, 42): np.float64(0.01126593082759863), (39, 43): np.float64(0.0006030797792376905), (39, 44): np.float64(0.0032515288011004404), (39, 45): np.float64(9.33418078049022e-06), (39, 46): np.float64(-0.0034

36634070363381), (39, 47): np.float64(-1.1581252396126016e-05), (39, 48): np.float64(0.0002715192728329035), (39, 49): np.float64(-0.00042591340576597956), (39, 50): np.float64(0.002877447787077111), (39, 51): np.float64(-0.00015290429918060183), (39, 53): np.float64(0.0002676886652807546), (39, 59): np.float64(0.0010520731632596477), (39, 62): np.float64(-0.0025606982161278614), (39, 64): np.float64(-3.63691955783534e-05), (39, 67): np.float64(-0.0015061157150656841), (39, 73): np.float64(-0.0005147314004513856), (39, 74): np.float64(0.0008272564539568913), (14, 40): np.float64(0.0006748065477842237), (15, 40): np.float64(0.0015611282187081496), (24, 40): np.float64(0.00034269859897306874), (29, 40): np.float64(-0.0018562624005549742), (32, 40): np.float64(0.001440725641979704), (33, 40): np.float64(0.0009138689307610975), (35, 40): np.float64(0.0013319199167719617), (40, 41): np.float64(0.0036380603313605907), (40, 42): np.float64(0.005234919856409764), (40, 43): np.float64(-0.0002188683728448395 2), (40, 45): np.float64(-0.0026008371657551712), (40, 47): np.float64(-0.0007339494916679782), (40, 48): np.float64(-0.0014711693407430764), (40, 49): np.float64(-0.0010108883813213186), (40, 51): np.float64(-0.001900865710013523), (40, 52): np.float64(-7.871190562314092e-05), (40, 53): np.float64(-0.0007313779685640835), (40, 54): np.float64(-0.002144771454506229), (40, 57): np.float64(-0.0023426198539692102), (40, 58): np.float64(-0.001239088252948133), (40, 59): np.float64(-5.011419249341813e-05), (40, 60): np.float64(-0.0007670332624482519), (40, 63): np.float64(-0.002097277758148544), (40, 64): np.float64(-0.0009593873139744962), (40, 65): np.float64(0.0005846743531362549), (40, 66): np.float64(-0.004004805047202194), (40, 68): np.float64(-0.006295424735980292), (40, 69): np.float64(-0.0007335896720111006), (40, 70): np.float64(-0.003169018475827625), (40, 72): np.float64(-0.005970696651282179), (1, 41): np.float64(0.0015025656893649596), (4, 41): np.float64(0.017255765933738794), (5, 41): np.float64(-0.001159852330032866), (6, 41): np.float64(0.0017202384468947012), (14, 41): np.float64(0.0010015291758590234), (15, 41): np.float64(0.0025287099465477487), (20, 41): np.float64(0.0024892195518086786), (24, 41): np.float64(0.002591806408749464 3), (32, 41): np.float64(0.00261207543844176), (33, 41): np.float64(0.0016435378847728848), (41, 43): np.float64(-0.0007960164973953818), (41, 45): np.float64(-0.001676212080612699), (41, 46): np.float64(-0.004954049511554317), (41, 47): np.float64(-0.002651112539261465), (41, 48): np.float64(-0.00259747949501349), (41, 50): np.float64(-0.00023801401845425843), (41, 51): np.float64(-0.001413232854312019), (41, 52): np.float64(-0.004789597220600233), (41, 53): np.float64(-0.0013904271460883706), (41, 54): np.float64(-0.0034101700036213507), (41, 55): np.float64(-0.005975105669133002), (41, 57): np.float64(-0.003784837735998479), (41, 58): np.float64(-0.001757211085734826), (41, 59): np.float64(-0.0007098355242860237), (41, 60): np.float64(-0.002910205818737694), (41, 62): np.float64(-0.009765970721849088), (41, 64): np.float64(-0.0020352452149445272), (41, 65): np.float64(-0.0005076556149800626), (41, 66): np.float64(-0.012028230934305625), (41, 67): np.float64(-0.0034708502176541496), (41, 71): np.float64(0.0022162909601221212), (41, 72): np.float64(-0.004888270352548428), (41, 73): np.float64(-0.007167389318004915), (4, 42): np.float64(0.008094468852827025), (6, 42): np.float64(0.0033220175471198486), (7, 42): np.float64(0.0019067771000868408), (15, 42): np.float64(0.0019182514590143417), (17, 42): np.float64(0.003134376126992782), (24, 42): np.float64(0.004602462411821163), (30, 42): np.float64(-0.008290661282684854), (33, 42): np.float64(0.0028236525403307453), (35, 42): np.float64(0.0026145965157837103), (42, 44): np.float64(-0.0013028217569480557), (42, 45): np.float64(-0.0015699041544129616), (42, 46): np.float64(-0.015221235085382302), (42, 48): np.float64(-0.003968285210311028), (42, 51): np.float64(-0.001615877152383214), (42, 53): np.float64(-0.005328627171490543), (42, 57): np.float64(-0.004075137253687018), (42, 58): np.float64(-0.005375942632662125), (42, 61): np.float64(-0.0019085455284565992), (42, 62): np.float64(-0.007161889230733352), (42, 63): np.float64(-0.0050725268809375754), (42, 64): np.float64(-0.0034615243848241314), (42, 65): np.float64(-0.0010169061138550603), (42, 66): np.float64(-0.01480785829636752), (42, 69): np.float64(-0.001994438611834124), (42, 70): np.float64(-0.0019123241748382092), (42, 71): np.float64(0.003925808579926545), (42, 72): np.float64(-0.005630267221454232), (42, 74): np.float64(-0.002086833428029402), (4, 45): np.float64(0.0009085672489343655), (6, 45): np.float64(-0.0019809116670373534), (7, 45): np.float64(0.0006477448723201839), (14, 45): np.float64(-0.0013960879867936633), (15, 45): np.float64(5.512517842130687e-05), (17, 45): np.float64(0.001563880615390126 4), (22, 45): np.float64(0.00018070435917707435), (29, 45): np.float64(-0.0076373873082252255), (30, 45): np.float64(-0.013396924312876101), (33, 45): np.float64(0.0012351295973769218), (35, 45): np.float64(0.0006350120217823446), (44, 45): np.float64(-

0.0035469509551746695), (45, 46): np.float64(-0.018500432332185374), (45, 48): np.float64(0.0008400996763382988), (45, 49): np.float64(-0.0003597206034554813), (45, 50): np.float64(0.0011989789987652362), (45, 53): np.float64(0.00038861618177657447), (45, 55): np.float64(-0.0017682738302804717), (45, 57): np.float64(-0.0008757297070020804), (45, 59): np.float64(0.0012139496312977638), (45, 60): np.float64(-0.00023219063899529866), (45, 62): np.float64(-0.003972721225083365), (45, 63): np.float64(-0.002022572053348024), (45, 64): np.float64(-0.00010600140431615662), (45, 65): np.float64(0.006276816729264952), (45, 67): np.float64(-0.0027270320004261586), (45, 68): np.float64(-0.0005517089108171625), (45, 70): np.float64(-0.0014850989927791976), (45, 71): np.float64(0.0038890271234169674), (1, 46): np.float64(-0.003400800710834459), (4, 46): np.float64(-0.0027710940347986557), (6, 46): np.float64(-0.020916433382031942), (7, 46): np.float64(-0.003311513993764251), (14, 46): np.float64(-0.010614412223639173), (22, 46): np.float64(-0.010150170731768793), (24, 46): np.float64(-0.005140506799309706), (29, 46): np.float64(-0.009698795298878779), (30, 46): np.float64(-0.007646794267529079), (35, 46): np.float64(-0.003032875678033003), (46, 48): np.float64(0.002814636228153351), (46, 50): np.float64(0.020072482926914815), (46, 53): np.float64(0.0036618225685619904), (46, 54): np.float64(0.0033838464190207107), (46, 55): np.float64(0.0009652054819980902), (46, 57): np.float64(0.0024477586183144057), (46, 58): np.float64(0.0026335341596267583), (46, 59): np.float64(0.003326276190309389), (46, 60): np.float64(0.0033727866830811555), (46, 62): np.float64(0.0007254069535081577), (46, 64): np.float64(0.003838543436964946), (46, 67): np.float64(0.0017162790545379228), (46, 69): np.float64(0.005023544076137463), (46, 72): np.float64(0.0007283417226937931), (46, 73): np.float64(0.002389960150679519), (46, 74): np.float64(0.004391197909399857), (5, 47): np.float64(-0.00658133965410817), (15, 47): np.float64(2.7809436308059578e-05), (20, 47): np.float64(-8.633797521188085e-05), (22, 47): np.float64(0.00035402869895615526), (29, 47): np.float64(-0.004292193162676975), (33, 47): np.float64(0.0003206350251944619), (35, 47): np.float64(0.0012623719730709688), (44, 47): np.float64(-0.0006390943170868825), (47, 50): np.float64(0.001660487136168964), (47, 51): np.float64(-0.0003984436465492424), (47, 52): np.float64(0.0009069058882851195), (47, 53): np.float64(0.00012389262117690078), (47, 54): np.float64(-0.001219515621003344), (47, 55): np.float64(-0.008553237992117265), (47, 59): np.float64(0.0006342748556149604), (47, 62): np.float64(-0.014850329327219981), (47, 64): np.float64(-8.378092445259486e-06), (47, 65): np.float64(0.0017432296774360383), (47, 66): np.float64(-0.002492976001132045), (47, 68): np.float64(-0.0004841834209469462), (47, 70): np.float64(-0.0006596400144271232), (47, 72): np.float64(-0.003183054768169826), (47, 73): np.float64(-0.0003200950888354383), (47, 74): np.float64(0.0013750631464862254), (7, 51): np.float64(0.000789713158978674), (14, 51): np.float64(-0.0018858840044795192), (15, 51): np.float64(-0.0001589192579819723), (17, 51): np.float64(0.0006140376235799294), (20, 51): np.float64(-0.00021266972928143225), (24, 51): np.float64(-0.0046055830564140305), (29, 51): np.float64(-0.014265231181395853), (33, 51): np.float64(-3.426077654751479e-06), (43, 51): np.float64(-0.0007153077686368088), (44, 51): np.float64(-0.0007032970813487476), (48, 51): np.float64(-0.00039788236128981605), (49, 51): np.float64(0.0005103834731433657), (50, 51): np.float64(-0.010515633260617972), (51, 52): np.float64(0.006981482390181238), (51, 53): np.float64(0.0002541158092551311), (51, 54): np.float64(-0.0007922422046856993), (51, 55): np.float64(-0.004884099324483752), (51, 57): np.float64(-0.004141101772651247), (51, 58): np.float64(-0.0001695093234355242), (51, 61): np.float64(-0.00026954893067202175), (51, 62): np.float64(-0.010519808649372542), (51, 63): np.float64(-0.0023884134557054745), (51, 64): np.float64(0.00032914436288845975), (51, 65): np.float64(0.0026092773845972444), (51, 67): np.float64(-0.002378265971589815), (51, 68): np.float64(-0.00033081510169719253), (51, 70): np.float64(-0.0004876633930572623), (51, 71): np.float64(0.007572533146395163), (51, 72): np.float64(-0.0035678277551121663), (1, 53): np.float64(-0.0005770755131260367), (4, 53): np.float64(0.0007829714170298506), (6, 53): np.float64(-0.0004995106929851569), (7, 53): np.float64(0.001019412341344936), (15, 53): np.float64(0.00016212008784271953), (20, 53): np.float64(-0.0001234759447771674), (24, 53): np.float64(-0.001763780120610906), (29, 53): np.float64(-0.0014990741051927271), (32, 53): np.float64(0.0007933386116709323), (33, 53): np.float64(0.0002552933040123538), (35, 53): np.float64(0.0020894211565066416), (44, 53): np.float64(-0.0006878705392317494), (50, 53): np.float64(-0.002534623543776019), (53, 54): np.float64(-0.0016686151320352022), (53, 55): np.float64(-0.0020986047225778296), (53, 57): np.float64(-0.005840300187461815), (53, 58): np.float64(-0.0004971346070162107), (53, 59): np.float64(0.0006040531855013354), (53, 60): np.float64(-

0.0002674459529077521), (53, 63): np.float64(-0.001739880277739941), (53, 64): np.float64(-7.201771279106077e-05), (53, 65): n
p.float64(0.002477848082219473), (53, 68): np.float64(-0.0015323220163324144), (53, 69): np.float64(0.0007683728794348417), (5
3, 70): np.float64(-0.0009765256428557755), (53, 71): np.float64(0.0031609873718184862), (1, 54): np.float64(-0.001152967770730
9066), (4, 54): np.float64(-0.0009979339420546557), (5, 54): np.float64(-0.012825264354476278), (6, 54): np.float64(-0.00515457
5134076782), (17, 54): np.float64(-0.0015972308372406358), (20, 54): np.float64(-0.0009306563608868156), (22, 54): np.float64(-
0.001705019392103820 4), (33, 54): np.float64(-0.0012318142432163556), (43, 54): np.float64(-0.002784000647465004), (44, 54): n
p.float64(-0.0015023052887825057), (49, 54): np.float64(-0.0016321076539341324), (52, 54): np.float64(-0.0019679441097401017),
(54, 55): np.float64(-0.0009221177342473511), (54, 58): np.float64(0.0017881821921417263), (54, 59): np.float64(0.0018326619346
400893), (54, 60): np.float64(0.0020772292910901107), (54, 63): np.float64(-0.0006708153026192937), (54, 64): np.float64(0.0017
211813828812135), (54, 67): np.float64(-0.00025118045644560304), (54, 72): np.float64(-0.008587162855401415), (54, 74): np.floa
t64(0.002228924730409613), (1, 55): np.float64(-0.002799015407320286), (4, 55): np.float64(-0.002932402241623203), (7, 55): np.
float64(-0.0016669256646227144), (14, 55): np.float64(-0.01138677657644807), (15, 55): np.float64(-0.002849167851600484), (17,
55): np.float64(-0.0017641193086911373), (20, 55): np.float64(-0.002055074627660856), (22, 55): np.float64(-0.00460469922017855
45), (29, 55): np.float64(-0.006628271201140542), (30, 55): np.float64(-0.005895871831493602), (43, 55): np.float64(-0.00721585
7905813254), (48, 55): np.float64(-0.0023076786083093366), (52, 55): np.float64(-0.006672284312736492), (55, 57): np.float64(0.
0014650206826331382), (55, 58): np.float64(0.005569682611939811), (55, 59): np.float64(0.003026532013715351), (55, 61): np.floa
t64(0.001650196641884519), (55, 62): np.float64(-0.0015370581440655887), (55, 63): np.float64(0.0014796398710706531), (55, 66):
np.float64(-0.004346499227685462), (55, 67): np.float64(0.0009783792243129826), (55, 69): np.float64(0.0034523200920534235), (5
5, 70): np.float64(0.0021507100207310637), (55, 71): np.float64(0.014372633452541703), (1, 57): np.float64(-0.0023380719563805 1
3), (4, 57): np.float64(-0.00030066570900835434), (5, 57): np.float64(-0.005253727878583344), (14, 57): np.float64(-0.005407016
359276638), (15, 57): np.float64(-0.002296263599848507), (20, 57): np.float64(-0.003945163632503418), (24, 57): np.float64(-0.0
018994686664191654), (30, 57): np.float64(-0.004662414321201327), (32, 57): np.float64(-0.0002661507510817819), (35, 57): np.fl
oat64(-0.0006526555774300574), (43, 57): np.float64(-0.008032675299731987), (48, 57): np.float64(-0.0019669719813211477), (52,
57): np.float64(-0.002931833066337741), (57, 59): np.float64(0.007175328734251913), (57, 60): np.float64(0.000852055376690133
5), (57, 62): np.float64(-0.0022133486360745715), (57, 67): np.float64(-0.000740706281152581), (57, 68): np.float64(0.002137639
5427742684), (57, 69): np.float64(0.007406135018411346), (57, 70): np.float64(0.0003905112915922016), (57, 72): np.float64(-0.0
06153227694413187), (57, 73): np.float64(0.0012040229013382101), (57, 74): np.float64(0.0045842330053655), (1, 58): np.float64
(-0.0028681793215435333), (4, 58): np.float64(0.0003609810339879 07), (5, 58): np.float64(-0.0028422238052024703), (6, 58): np.f
loat64(-0.0009558928325356688), (14, 58): np.float64(-0.0012600221167299928), (15, 58): np.float64(-0.000734711642901784), (17,
58): np.float64(-0.00019162988568764672), (20, 58): np.float64(-0.00038538612260113827), (22, 58): np.float64(-0.00026975065948
33479), (29, 58): np.float64(-0.004071614349548067), (30, 58): np.float64(-0.003897895061129437), (32, 58): np.float64(0.000346
43560932559587), (35, 58): np.float64(0.00016855813599677845), (43, 58): np.float64(-0.0033838875128461223), (44, 58): np.float
64(-0.002678878381341161), (48, 58): np.float64(-0.0005843904632650037), (49, 58): np.float64(0.0003796970237192005 6), (50, 5
8): np.float64(-0.003167618918791433), (52, 58): np.float64(-0.004316576820762411), (58, 59): np.float64(0.002170774036699143
6), (58, 60): np.float64(0.0009936235144668787), (58, 61): np.float64(8.33116655635978e-05), (58, 63): np.float64(-0.0069862704
77867707), (58, 64): np.float64(0.00032729428608863987), (58, 65): np.float64(0.0035103848651679725), (58, 67): np.float64(-0.0
012560438217482136), (58, 72): np.float64(-0.0024180637917369436), (58, 73): np.float64(-1.225574850552872e-05), (58, 74): np.f
loat64(0.0015773765174743055), (4, 59): np.float64(0.0013395624227373882), (5, 59): np.float64(-0.0024342096052007494), (6, 5
9): np.float64(0.00046123348526901797), (7, 59): np.float64(0.00890777088809519), (14, 59): np.float64(0.0009173877029302433),
(15, 59): np.float64(0.0014995294879722395), (17, 59): np.float64(0.0011881257559759808), (20, 59): np.float64(0.00062600289278
81232), (22, 59): np.float64(0.001554543841553612), (29, 59): np.float64(-0.0012714599234711257), (30, 59): np.float64(-0.01157

6344670235404), (33, 59): np.float64(0.0009338711460039669), (48, 59): np.float64(0.002027264185326997), (49, 59): np.float64(0.0015139318727404128), (50, 59): np.float64(-0.0008729593311693283), (59, 60): np.float64(-0.0019116773949231635), (59, 62): np.float64(-0.006913508634074225), (59, 63): np.float64(-0.002895761479369445), (59, 65): np.float64(0.00017973142188274036), (59, 68): np.float64(-0.0014709797217372417), (59, 69): np.float64(-0.00013003433336535088), (59, 70): np.float64(-0.001368199619550799 6), (59, 71): np.float64(0.004102252208538065), (59, 72): np.float64(-0.019903686233866468), (59, 73): np.float64(-0.0015163953040405046), (7, 60): np.float64(0.0012568509595556636), (22, 60): np.float64(-0.0001780825844836922 2), (24, 60): np.float64(-0.001542911211562845), (29, 60): np.float64(-0.005579972724061418), (32, 60): np.float64(0.0005807323578926899), (35, 60): np.float64(0.0006728348568020674), (43, 60): np.float64(-0.0012221645916470193), (44, 60): np.float64(-0.003002693593727447 3), (52, 60): np.float64(-0.002827500356569765), (60, 62): np.float64(-0.0020940141391285086), (60, 64): np.float64(0.000415253 25610113664), (60, 68): np.float64(-0.0003801747989205118), (60, 72): np.float64(-0.008255689270796448), (4, 62): np.float64(-0.002320665389375845), (5, 62): np.float64(-0.007391858882908332), (6, 62): np.float64(-0.008148431852806165), (7, 62): np.float64(-0.01285836486007113), (15, 62): np.float64(-0.017822647973117795), (17, 62): np.float64(-0.003011892131273618), (20, 62): np.float64(-0.003497028412689044), (22, 62): np.float64(-0.008464512290012027), (24, 62): np.float64(-0.003714242070084934), (29, 62): np.float64(-0.003841205592625657), (30, 62): np.float64(-0.0073269846608320244), (32, 62): np.float64(-0.00205879418717 26204), (44, 62): np.float64(-0.017287022652661712), (48, 62): np.float64(-0.022259238658780364), (49, 62): np.float64(-0.00255 16790782719842), (52, 62): np.float64(-0.010474521787215876), (61, 62): np.float64(-0.003263161754918556), (62, 63): np.float64(0.0015868152999211434), (62, 65): np.float64(0.012404899393016814), (62, 66): np.float64(-7.408838403195934e-05), (62, 67): np.float64(0.0011951756650314766), (62, 68): np.float64(0.0029695581592846664), (62, 69): np.float64(0.003130060834245312), (62, 70): np.float64(0.0030355562900937602), (62, 72): np.float64(-0.00046032020618046757), (62, 73): np.float64(0.010234080203158972), (62, 74): np.float64(0.003548672716036302), (1, 63): np.float64(-0.004010125128063364), (15, 63): np.float64(-0.003527463052672238), (17, 63): np.float64(-0.002263136509285733), (22, 63): np.float64(-0.0017559337240943378), (29, 63): np.float64(-0.003639000643338009), (30, 63): np.float64(-0.006888290313450386), (32, 63): np.float64(-0.0035907745885523906), (33, 63): np.float64(-0.006549478548141047), (35, 63): np.float64(-0.0021050078892354824), (48, 63): np.float64(-0.004111754189225956), (49, 63): np.float64(-0.0016132339844031295), (52, 63): np.float64(-0.0035430655838008442), (63, 64): np.float64(0.006477264527761917), (63, 66): np.float64(-0.0009514908074382944), (63, 70): np.float64(0.0012413830707658517), (63, 71): np.float64(0.0056407752591476346), (63, 72): np.float64(-0.007796140210880681), (63, 73): np.float64(0.00299168305208856), (63, 74): np.float64(0.006051330265631511), (1, 64): np.float64(-0.0002410622407747175), (4, 64): np.float64(0.0008432671157996664), (5, 64): np.float64(-0.01285840262738297), (6, 64): np.float64(-0.00041839250171829903), (7, 64): np.float64(0.00041845885468589456), (20, 64): np.float64(-0.00019482361859101617), (22, 64): np.float64(0.00012288602482006942), (24, 64): np.float64(-0.0005052257115490993), (29, 64): np.float64(-0.008572188438001999), (30, 64): np.float64(-0.015142344660816123), (32, 64): np.float64(0.0005831154351058741), (33, 64): np.float64(0.00013592195316499913), (35, 64): np.float64(0.0012355786495208876), (43, 64): np.float64(-0.0014552337007700334), (44, 64): np.float64(-0.0008575045608856479), (50, 64): np.float64(-0.0018316111554567947), (52, 64): np.float64(-0.0008625566407421236), (61, 64): np.float64(0.0011830234461498196), (64, 65): np.float64(0.0019355942224447825), (64, 66): np.float64(-0.012269684016780413), (64, 67): np.float64(-0.0010649654435119847), (64, 68): np.float64(-0.000636922814199818 6), (64, 69): np.float64(0.0015037523180770685), (64, 70): np.float64(-0.0008624454880754573), (64, 73): np.float64(-0.0005510466776483283), (64, 74): np.float64(0.0010083118188160326), (1, 66): np.float64(-0.002645746885780955), (5, 66): np.float64(-0.004646799616161579), (20, 66): np.float64(-0.005686716900750976), (33, 66): np.float64(-0.0021600386059252743), (35, 66): np.float64(-0.0017921342457183273), (44, 66): np.float64(-0.01986914711701752), (48, 66): np.float64(-0.005618403194451064), (49, 66): np.float64(-0.0036030795048017025), (50, 66): np.float64(-0.004287248012669764), (61, 66): np.float64(-0.003905431148370556), (66, 70): np.float64(0.006258790642238021), (66, 71): np.float64(0.005295484133676993), (66, 72): np.float64(-0.0011272861288830205), (66, 74): np.float64(0.003399096728817493), (5, 67): np.float64(-0.015105400882233489), (6, 67): np.float64(-0.00874

2755670318646), (7, 67): np.float64(-0.0012159662138683358), (14, 67): np.float64(-0.001919428008483036), (17, 67): np.float64(-0.0016979148752585104), (20, 67): np.float64(-0.001329963403607929), (22, 67): np.float64(-0.003066892676135166), (24, 67): np.float64(-0.01303365954166933), (30, 67): np.float64(-0.015285230424118052), (32, 67): np.float64(-0.0005470417820933263), (33, 67): np.float64(-0.0022023205037396076), (43, 67): np.float64(-0.0032624956906929484), (44, 67): np.float64(-0.007147364924844353), (48, 67): np.float64(-0.002682336468288116), (49, 67): np.float64(-0.0015234527541782932), (50, 67): np.float64(-0.002632965687266004), (61, 67): np.float64(-0.00100353429195801), (67, 68): np.float64(0.001408603318326248), (4, 68): np.float64(0.000312083166910557), (5, 68): np.float64(-0.004927778035829198), (6, 68): np.float64(-0.0011371800781416615), (7, 68): np.float64(-9.106690268401014e-05), (14, 68): np.float64(-0.005622629681994365), (20, 68): np.float64(-0.004517595851928584), (22, 68): np.float64(-0.0009610914102683283), (24, 68): np.float64(-0.0036244087818173305), (29, 68): np.float64(-0.002163550976262075), (32, 68): np.float64(0.0011038987559213777), (49, 68): np.float64(-5.664713818948757e-05), (52, 68): np.float64(-0.0016669984954674735), (61, 68): np.float64(-0.0003873722555728305), (68, 69): np.float64(0.0010616018910204336), (68, 70): np.float64(-1.5016538985602023e-05), (68, 71): np.float64(0.0038179404277984643), (68, 73): np.float64(0.00012854983855274405), (68, 74): np.float64(0.0027444594813009733), (1, 71): np.float64(0.012461900017677288), (4, 71): np.float64(0.0033673569163991265), (6, 71): np.float64(0.003701322278743514), (7, 71): np.float64(0.005956662227244003), (14, 71): np.float64(0.003593200555279965), (15, 71): np.float64(0.0070599633295177185), (20, 71): np.float64(0.0058480820168384335), (24, 71): np.float64(0.0026419756984223633), (29, 71): np.float64(0.0018754130394706887), (30, 71): np.float64(-0.001442919583861481), (33, 71): np.float64(0.0030640542259160065), (43, 71): np.float64(0.0029656631482421025), (44, 71): np.float64(0.002418150480742592), (49, 71): np.float64(0.005837764175925037), (61, 71): np.float64(0.0032009489314784637), (65, 71): np.float64(0.0021014476767455535), (69, 71): np.float64(0.004767977699168108), (70, 71): np.float64(0.004361815355426269), (71, 72): np.float64(-0.0061834008557169985), (4, 72): np.float64(-0.00420886509748222), (5, 72): np.float64(-0.0066491110221438), (7, 72): np.float64(-0.008056645113518817), (20, 72): np.float64(-0.0043090710281883885), (24, 72): np.float64(-0.004474684559381874), (29, 72): np.float64(-0.015869390815000283), (30, 72): np.float64(-0.00952758597278391), (35, 72): np.float64(-0.006506013935556697), (44, 72): np.float64(-0.007596739766025438), (65, 72): np.float64(-0.0059322051662232585), (70, 72): np.float64(-0.004936184563491574), (72, 73): np.float64(0.009799145073035033), (1, 4): np.float64(-0.0007382353659157043), (1, 6): np.float64(0.00010381904118515695), (1, 7): np.float64(-0.0008563133733661269), (1, 14): np.float64(0.0014047128455994043), (1, 15): np.float64(-0.0005075883096547981), (1, 17): np.float64(-0.001126638621607446), (1, 29): np.float64(0.0016601715917160084), (1, 32): np.float64(-0.0007113902754231373), (1, 35): np.float64(-0.0007683949070999972), (1, 43): np.float64(0.0005349792172468792), (1, 49): np.float64(-0.003083696456998953), (1, 50): np.float64(0.0023729724557771665), (1, 61): np.float64(-0.0010625200843423586), (1, 65): np.float64(0.0009476540050873946), (1, 69): np.float64(0.0007266112967179126), (1, 73): np.float64(-0.0006190555286771398), (4, 7): np.float64(0.00047727843041551007), (6, 7): np.float64(-0.0015120438248832978), (7, 20): np.float64(0.003692260416572461), (7, 24): np.float64(0.000819376728898599), (7, 29): np.float64(0.004082927485959505), (7, 30): np.float64(0.015531693828532934), (7, 32): np.float64(-0.00048143773403559124), (7, 33): np.float64(0.0004330021041080812), (7, 35): np.float64(-9.286878418585286e-05), (7, 44): np.float64(0.0011251119313763413), (7, 48): np.float64(0.0012229234344429954), (7, 61): np.float64(0.0010097965655727536), (7, 65): np.float64(0.0016159359358382812), (7, 69): np.float64(0.0017105372570808584), (7, 73): np.float64(0.0001776517750348964), (7, 74): np.float64(0.002201837924740284), (6, 14): np.float64(0.00011243696504603803), (14, 15): np.float64(-0.000731304860241003), (14, 20): np.float64(-0.0005348317264857312), (14, 22): np.float64(-0.0006942834004882487), (14, 29): np.float64(0.004929922182923405), (14, 43): np.float64(0.0003928319342594087), (14, 44): np.float64(0.0001650031653948609), (14, 48): np.float64(-0.0011694132432091744), (14, 49): np.float64(-0.0028230156364577397), (14, 50): np.float64(0.0022958147763468325), (14, 61): np.float64(-0.0014053850443021291), (14, 65): np.float64(0.0007971449916101259), (14, 69): np.float64(0.00022431181098653995), (14, 73): np.float64(-0.003784580773630664), (14, 74): np.float64(0.0002999373794504 67), (4, 17): np.float64(0.00041179883445070895), (6, 17): np.float64(-0.0008738262926229886), (15, 17): np.float64(-0.00020287319719103722), (17, 22): np.float64(0.0003416158888133

```
9254), (17, 24): np.float64(0.0023597544533775286), (17, 29): np.float64(0.0031470420034093063), (17, 30): np.float64(0.0119966
24380113308), (17, 32): np.float64(-0.0004852371463417757), (17, 35): np.float64(-0.000358015498556632), (17, 44): np.float64
(0.0012101710331357), (17, 49): np.float64(-0.0003128545612344464), (17, 50): np.float64(0.006293380776188209), (17, 52): np.fl
oat64(0.0008497736031919311), (17, 61): np.float64(-2.018546424878274e-05), (17, 69): np.float64(0.001858226036533725), (17, 7
3): np.float64(-7.632580793656816e-05), (4, 20): np.float64(0.0006668322470068401), (15, 20): np.float64(0.000686742769190805
7), (20, 22): np.float64(-0.0003144389497558802), (20, 24): np.float64(0.0029094342710479326), (20, 29): np.float64(0.002394890
626975759), (20, 30): np.float64(0.004844800438024664), (20, 32): np.float64(-0.0006557810358041134), (20, 35): np.float64(-0.0
02182753078467879), (20, 43): np.float64(0.0006521447783074521), (20, 52): np.float64(0.0010820347841711832), (20, 61): np.floa
t64(-0.0005479803656042542), (20, 65): np.float64(0.004899911259140889), (20, 70): np.float64(-0.0012091137673257634), (20, 7
3): np.float64(-0.00039481303447512316), (20, 74): np.float64(0.001351468333197928), (4, 22): np.float64(0.000536827069900433
2), (5, 22): np.float64(-0.02047502229848269), (15, 22): np.float64(-8.258818620292287e-05), (22, 30): np.float64(0.02781682564
162889), (22, 32): np.float64(-0.0007157508497265782), (22, 33): np.float64(-0.00013525803226637596), (22, 35): np.float64(-0.0
006758937915752591), (22, 48): np.float64(0.00034862404401961333), (22, 52): np.float64(0.0007586441310943173), (22, 65): np.fl
oat64(0.007284506328671512), (22, 69): np.float64(0.0009209955015910927), (22, 73): np.float64(-0.0016284027759796837), (15, 3
0): np.float64(0.008371751463674143), (24, 30): np.float64(0.014080981406663681), (30, 33): np.float64(-0.0039202906730957 5),
(30, 35): np.float64(-0.005900601738425054), (30, 43): np.float64(-0.010134932878564477), (30, 48): np.float64(-0.0081699518773
39232), (30, 50): np.float64(-0.015956253676569252), (30, 52): np.float64(-0.003059945713665372), (30, 61): np.float64(-0.00839
3848189833093), (30, 65): np.float64(-0.0033450594984088923), (30, 69): np.float64(-0.010110542489476022), (30, 73): np.float64
(-0.006250725169414884), (30, 74): np.float64(-0.016264757905269104), (5, 33): np.float64(-0.005811269807438656), (6, 33): np.f
loat64(-0.0008745608999256294), (24, 33): np.float64(-0.0005905414854754777), (32, 33): np.float64(0.0010021284364790815), (33,
48): np.float64(0.0004987259868093337), (33, 49): np.float64(-0.00022865216324147755), (33, 52): np.float64(0.0022268598251227 4
5), (33, 65): np.float64(0.0015798927961620524), (33, 69): np.float64(0.0010017833216858973), (4, 35): np.float64(0.000804860909
73054747), (5, 35): np.float64(-0.0030346900148127804), (24, 35): np.float64(-0.0017740077962524413), (29, 35): np.float64(-0.0
02228920139249278), (32, 35): np.float64(0.0001512004446485 8443), (35, 43): np.float64(0.0033994501572141025), (35, 44): np.flo
at64(0.005306982777941545), (35, 48): np.float64(0.0006785939727331259), (35, 61): np.float64(0.0003071932038564848), (35, 69):
np.float64(0.002118604652764952), (35, 70): np.float64(3.7702330297428764e-05), (35, 74): np.float64(0.0036818746077658467),
(5, 43): np.float64(-0.002101536823373332), (6, 43): np.float64(0.0004915065448584879), (15, 43): np.float64(0.0045399472750497
225), (24, 43): np.float64(0.00018685010974051368), (29, 43): np.float64(-0.0012569937662522991), (32, 43): np.float64(0.001279
5675516191472), (43, 48): np.float64(-0.0004607388980554395), (43, 49): np.float64(-0.004216835553492624), (43, 65): np.float64
(0.00027663243523873044), (43, 69): np.float64(-7.113042300352283e-05), (43, 70): np.float64(-0.0014971563041517057), (43, 73):
np.float64(-0.003450503249059853), (43, 74): np.float64(0.00023503003336423377), (15, 44): np.float64(0.001566851889183507), (2
4, 44): np.float64(6.884032774277287e-05), (29, 44): np.float64(-0.003338650960346257), (32, 44): np.float64(0.0011541214089895
969), (44, 49): np.float64(-0.0011860098153858485), (44, 50): np.float64(0.001308083661075319), (44, 52): np.float64(0.00017089
53019937838), (44, 61): np.float64(-0.002091515575839094), (44, 65): np.float64(0.00034492660204353206), (44, 69): np.float64
(0.00013448702335194522), (44, 73): np.float64(-0.0013882447231061594), (24, 48): np.float64(-0.0008161325112337146), (29, 48):
np.float64(-0.0036187901343087622), (48, 49): np.float64(-0.0013987347569652693), (48, 50): np.float64(0.0013771712608267128),
(48, 52): np.float64(0.0005299811443451402), (48, 61): np.float64(-0.000747965293387864), (48, 69): np.float64(0.00047356571125
154584), (48, 73): np.float64(-0.0011051421933836695), (48, 74): np.float64(0.0008450667458499765), (5, 49): np.float64(-0.0061
3964246153362), (15, 49): np.float64(-0.0005610758167019968), (24, 49): np.float64(-0.002200316877800249), (29, 49): np.float64
(-0.006054824808230569), (49, 50): np.float64(0.0048335916782865605), (49, 70): np.float64(-0.0001109235265791358), (4, 50): n
p.float64(0.0024766856668190116), (5, 50): np.float64(-0.0014146956848483563), (29, 50): np.float64(-0.0026236550819046848), (5
```

0, 52): np.float64(-0.0013474386365306481), (50, 61): np.float64(-0.002423984100881707), (50, 69): np.float64(-0.0008326256072053735), (50, 70): np.float64(-0.0016455119250763368), (50, 73): np.float64(-0.012560449877857156), (50, 74): np.float64(-0.0019529435718279795), (4, 52): np.float64(0.0038176482969028164), (5, 52): np.float64(-0.004604348641664747), (6, 52): np.float64(0.00042875272985618137), (15, 52): np.float64(0.0033273281063965618), (24, 52): np.float64(0.00022633787818832855), (52, 61): np.float64(-0.0012032503059193512), (52, 65): np.float64(0.0009390964355661594), (52, 69): np.float64(-0.00022007465924145408), (52, 70): np.float64(-0.0010666609539470668), (24, 61): np.float64(-0.0010839808209327597), (32, 61): np.float64(0.000455820381848546), (61, 69): np.float64(0.0011241087213632835), (61, 70): np.float64(-0.000212213654955111), (61, 74): np.float64(0.0027864921769920506), (4, 65): np.float64(0.007251849028954133), (5, 65): np.float64(-0.0024322508048487436), (6, 65): np.float64(0.0030797495639288725), (24, 65): np.float64(0.001375610534609078), (29, 65): np.float64(-0.0015259726656259513), (65, 70): np.float64(-0.0021543818634878246), (65, 73): np.float64(-0.0011056240299351545), (65, 74): np.float64(-0.00020209420657302962), (4, 70): np.float64(0.00037941374838624446), (5, 70): np.float64(-0.003658319331906629), (6, 70): np.float64(-0.0011435477759678685), (15, 70): np.float64(-0.0013000246809680548), (24, 70): np.float64(-0.0009034896613232864), (29, 70): np.float64(-0.0037287268308010026), (69, 70): np.float64(-0.005544532055071549), (70, 73): np.float64(0.00012012768173051285), (70, 74): np.float64(0.0015368289354421533), (4, 74): np.float64(0.0012864910049190273), (5, 74): np.float64(-0.0024098284828872564), (6, 74): np.float64(0.00037767631371124727), (15, 74): np.float64(0.0007352837084321406), (24, 74): np.float64(0.0003404558807581875), (29, 74): np.float64(-0.0029210692443655943), (73, 74): np.float64(0.0010535925690949698), (5, 15): np.float64(-0.004468165276289296), (5, 24): np.float64(-0.007643082081170897), (5, 69): np.float64(-0.002075632002808714), (4, 6): np.float64(0.0013734171029662577), (6, 15): np.float64(-0.0005236827089740092), (6, 29): np.float64(0.0014102381175541763), (6, 73): np.float64(-0.0007342998659680518), (4, 15): np.float64(0.0012036498469669738), (15, 24): np.float64(0.0016717075463423897), (15, 73): np.float64(-0.0012508573927907632), (24, 29): np.float64(0.0014216831198824522), (24, 32): np.float64(-0.0017605823461348561), (4, 32): np.float64(-7.67571925382108e-06), (29, 32): np.float64(-0.0031103462364895696), (32, 69): np.float64(0.001286071714257741), (32, 73): np.float64(0.0011283521665401265), (29, 73): np.float64(-0.001838526243909445), (69, 73): np.float64(-0.0032819973272307074), (4, 69): np.float64(0.0031873659911008977), (29, 69): np.float64(-0.0010346204151111646), (76, 81): np.float64(-0.12265599151272548), (77, 81): np.float64(-0.13453123484044238), (80, 81): np.float64(-0.11989789398994243), (81, 83): np.float64(0.25073692902526584), (81, 84): np.float64(0.16297223896888505), (81, 85): np.float64(0.09105222952036553), (81, 90): np.float64(0.11699135178065932), (81, 91): np.float64(0.6050141544466843), (81, 92): np.float64(0.08698196362742809), (81, 95): np.float64(0.49886136352378846), (81, 98): np.float64(0.08928685650704934), (81, 99): np.float64(0.23932099995162862), (81, 100): np.float64(0.07082306186569554), (81, 102): np.float64(0.16092997629591183), (81, 104): np.float64(0.1781807838812652), (81, 105): np.float64(0.2844177453010811), (81, 106): np.float64(0.17814061745310003), (81, 107): np.float64(0.07667877394134574), (81, 108): np.float64(0.07090015386305222), (81, 110): np.float64(0.15885793799145909), (81, 114): np.float64(0.13474405553221852), (81, 116): np.float64(0.09086385303513142), (81, 117): np.float64(0.08228173453718889), (81, 118): np.float64(0.10771067661544484), (81, 119): np.float64(0.15520824690030965), (81, 120): np.float64(0.1480565972836597), (81, 121): np.float64(0.09686892533570549), (81, 126): np.float64(0.5997155061251591), (81, 127): np.float64(0.10595193297041583), (81, 130): np.float64(0.16214158843919887), (81, 132): np.float64(0.08917005910551294), (81, 135): np.float64(0.3576336224108628), (81, 136): np.float64(0.10786587878574191), (81, 139): np.float64(0.12934710525073986), (81, 140): np.float64(0.10031644150532357), (81, 141): np.float64(0.41153440194052143), (81, 145): np.float64(0.1092826569042634), (81, 149): np.float64(0.11805444466261047), (75, 76): np.float64(-0.0037754584608318976), (76, 77): np.float64(-4.514140114072099e-05), (76, 80): np.float64(0.014748588981103514), (76, 86): np.float64(0.002258878210037062), (76, 88): np.float64(0.004249315920830152), (76, 89): np.float64(0.004812740416293877), (76, 91): np.float64(-0.013715424750558746), (76, 92): np.float64(-0.0003469171199100088), (76, 93): np.float64(0.001140659033599688 4), (76, 94): np.float64(0.0025232869490545624), (76, 95): np.float64(-0.012685426305635471), (76, 98): np.float64(0.0003937669862071308), (76, 99): np.float64(-0.001895267280891957), (76, 101): np.float64(0.002295121904950318), (76, 103): np.float64(0.0

062815033946499), (76, 104): np.float64(0.0036910243593759034), (76, 108): np.float64(0.0025631870541815965), (76, 111): np.float64(0.004715455879411064), (76, 113): np.float64(0.00395769283911833), (76, 116): np.float64(-0.0033238951346072987), (76, 117): np.float64(0.011280892802922616), (76, 127): np.float64(0.002206082555895099), (76, 129): np.float64(0.005996781585719832 5), (76, 131): np.float64(0.008833503123669435), (76, 133): np.float64(0.0016819163347934288), (76, 134): np.float64(0.0059962053303639325), (76, 135): np.float64(-0.0048242090386265025), (76, 137): np.float64(0.005236312542629879), (76, 140): np.float64(-0.004301138456857189), (76, 141): np.float64(-0.010492358709704795), (76, 144): np.float64(0.0027330977313071666), (76, 145): np.float64(-0.0006436576854016948), (76, 146): np.float64(0.07209283716705034), (76, 148): np.float64(0.0014651178320279135), (75, 77): np.float64(-0.006586033595048959), (77, 78): np.float64(0.013013940119403654), (77, 80): np.float64(0.003935404007111382), (77, 84): np.float64(-5.2344472216669236e-05), (77, 85): np.float64(0.02584707461860751), (77, 86): np.float64(0.004235679643963 52), (77, 89): np.float64(0.013523909735092628), (77, 90): np.float64(0.004045299951282098), (77, 91): np.float64(-0.006703932813233895), (77, 92): np.float64(-0.0003725340763315861), (77, 93): np.float64(0.0015691504946525755), (77, 94): np.float64(0.0027527520083843356), (77, 96): np.float64(0.00233157866915529), (77, 99): np.float64(-0.0011458038730352412), (77, 101): np.float64(0.000735605929620491), (77, 102): np.float64(0.0009545220027241315), (77, 103): np.float64(0.0033900201895659962), (77, 104): np.float64(0.004630373476599077), (77, 106): np.float64(-0.002304589228454817), (77, 108): np.float64(0.004450757188053878), (77, 112): np.float64(0.007303372348286029), (77, 114): np.float64(-0.0006898039167883938), (77, 118): np.float64(-0.0015434221534339064), (77, 120): np.float64(0.0014476244493542804), (77, 121): np.float64(0.0036171475034358306), (77, 122): np.float64(0.003933390221773833), (77, 123): np.float64(0.0016058687887297639), (77, 127): np.float64(0.0013349508532617203), (77, 129): np.float64(0.0010934511941035775), (77, 131): np.float64(0.004569127946148329), (77, 132): np.float64(0.026161487285788453), (77, 134): np.float64(0.0024219181737323613), (77, 135): np.float64(-0.011796574696684794), (77, 137): np.float64(0.0020563881202601358), (77, 138): np.float64(0.0027829286697517532), (77, 144): np.float64(0.0038000112873146858), (77, 145): np.float64(-0.003607525372579907), (77, 147): np.float64(0.004551929617880923), (77, 149): np.float64(0.004020925952972048), (78, 80): np.float64(-0.001537467276603829), (80, 82): np.float64(2.2505856565454486e-05), (80, 84): np.float64(-0.002521435033359458), (80, 85): np.float64(0.018234686944295694), (80, 88): np.float64(0.004977838863935261), (80, 89): np.float64(0.0038634187113047083), (80, 95): np.float64(-0.006719767334757689), (80, 97): np.float64(-0.0016052090759283023), (80, 98): np.float64(-0.006667994454872874), (80, 99): np.float64(-0.005868246393831058), (80, 100): np.float64(-0.0030958160768933926), (80, 101): np.float64(-0.0021989862203519064), (80, 102): np.float64(-0.002411826987172591), (80, 104): np.float64(-0.0008821208215020229), (80, 105): np.float64(-0.007621321533515746), (80, 106): np.float64(-0.010854126000061116), (80, 110): np.float64(-0.0037119812533455403), (80, 113): np.float64(0.0011587150463154496), (80, 118): np.float64(-0.002869169930946719), (80, 120): np.float64(-0.001508490090481183), (80, 125): np.float64(-0.0010515186065447463), (80, 126): np.float64(-0.037260737598012114), (80, 127): np.float64(-0.002183536949339153), (80, 128): np.float64(0.01000271742092037), (80, 130): np.float64(-0.00842291491971244), (80, 135): np.float64(-0.006776627965892136), (80, 136): np.float64(-0.0029354012756256376), (80, 140): np.float64(-0.014682092482085256), (80, 144): np.float64(0.0008064441244869227), (80, 145): np.float64(-0.0034514301246781095), (80, 146): np.float64(0.236814978359853), (80, 147): np.float64(-0.0025547611805837736), (80, 148): np.float64(-0.001989965311888741), (80, 149): np.float64(0.0010085919952450624), (75, 83): np.float64(-0.005720885490147328), (82, 83): np.float64(-0.005492682779675001), (83, 85): np.float64(0.009269826228161181), (83, 88): np.float64(0.017046328962329334), (83, 89): np.float64(0.009725130082467736), (83, 91): np.float64(-0.010359932907544214), (83, 92): np.float64(0.006210352172317402), (83, 96): np.float64(0.017750961660226064), (83, 97): np.float64(0.002305974590072936), (83, 98): np.float64(0.003905919784617608), (83, 99): np.float64(7.099451075268737e-05), (83, 102): np.float64(0.005098132065344204), (83, 106): np.float64(-0.0007325392663600293), (83, 107): np.float64(0.001986907146963551), (83, 108): np.float64(0.0064975181342592486), (83, 112): np.float64(0.0038043691867803535), (83, 114): np.float64(0.00043520958277850155), (83, 119): np.float64(0.010123983295764707), (83, 122): np.float64(0.004393807449330431), (83, 123): np.float64(0.00405538281149934), (83, 125): np.float64(0.00462563268379108), (83, 130): np.float64(0.002663752547942196), (83,

131): np.float64(0.0228690079769734), (83, 132): np.float64(0.006819618790835624), (83, 134): np.float64(0.004272906369254386), (83, 135): np.float64(-0.0036695162718699916), (83, 139): np.float64(0.001879505027647657), (83, 141): np.float64(-0.0029532312710027534), (83, 142): np.float64(0.007155326296846152), (83, 144): np.float64(0.005654162286618922), (83, 145): np.float64(0.002110981105899516), (83, 146): np.float64(0.0907801593556456), (83, 149): np.float64(0.005726730367176502), (78, 84): np.float64(-0.002738921769530121), (84, 85): np.float64(0.005110917792785528), (84, 87): np.float64(0.015730119190977637), (84, 90): np.float64(0.010780446524321429), (84, 91): np.float64(-0.01433533246538112), (84, 95): np.float64(-0.007996781212970816), (84, 97): np.float64(0.0032188831029646616), (84, 98): np.float64(0.0004709630798791202), (84, 100): np.float64(-0.0008399079847639675), (84, 102): np.float64(0.0027879630414810053), (84, 106): np.float64(-0.0020584727486733937), (84, 107): np.float64(-0.0005090051839943442), (84, 110): np.float64(0.0001508514119262606), (84, 111): np.float64(0.023059854331850888), (84, 113): np.float64(0.0059642491706575), (84, 115): np.float64(0.034118416235450845), (84, 117): np.float64(0.034726897257536554), (84, 119): np.float64(0.0019532557569479935), (84, 120): np.float64(0.006056457309121346), (84, 121): np.float64(0.0032961088128746754), (84, 122): np.float64(0.005732248965552882), (84, 125): np.float64(0.0037660311676876693), (84, 126): np.float64(-0.007379976746714345), (84, 130): np.float64(-0.0027826632143257138), (84, 132): np.float64(0.006066017646554712), (84, 133): np.float64(0.0014442901616822724), (84, 134): np.float64(0.002345755812995854), (84, 136): np.float64(-0.0005666326091731954), (84, 137): np.float64(0.010987097884114281), (84, 138): np.float64(0.0033679749441742264), (84, 139): np.float64(0.0015043171818189013), (84, 140): np.float64(-0.0005270770566759721), (84, 142): np.float64(0.0008336022723125753), (84, 143): np.float64(0.004142257863221866), (84, 144): np.float64(0.003994730521369403), (84, 147): np.float64(0.003045679476196595), (75, 85): np.float64(0.005131871432185469), (79, 85): np.float64(0.0038947442026364076), (82, 85): np.float64(0.0039024337627790264), (85, 87): np.float64(-0.0038799365886792504), (85, 88): np.float64(-0.002503846816021976), (85, 89): np.float64(-0.0017312485151184682), (85, 92): np.float64(-0.007817496592781723), (85, 93): np.float64(-0.010079664341863878), (85, 94): np.float64(-0.017829330864804933), (85, 96): np.float64(-0.01553886413787585), (85, 99): np.float64(-0.019123945440512616), (85, 100): np.float64(-0.0159081220554639), (85, 101): np.float64(-0.004854898910841659), (85, 106): np.float64(-0.010781812834218578), (85, 108): np.float64(-0.004672721335374952), (85, 111): np.float64(-0.006142723740461574), (85, 115): np.float64(-0.0005346512976934089), (85, 116): np.float64(-0.006927205321645175), (85, 119): np.float64(-0.019406734971053427), (85, 120): np.float64(-0.004812019465590958), (85, 121): np.float64(-0.0050957168700098915), (85, 122): np.float64(-0.012775797627439387), (85, 127): np.float64(-0.00713680193816025), (85, 130): np.float64(-0.009779956450188618), (85, 133): np.float64(-0.014103875143868709), (85, 134): np.float64(-0.0036737009117786238), (85, 138): np.float64(-0.0038210438554020457), (85, 139): np.float64(-0.006279120916836128), (85, 140): np.float64(-0.01029081419883807), (85, 142): np.float64(-0.008695022927292857), (85, 143): np.float64(-0.005162411119060186), (85, 145): np.float64(-0.009075136494858504), (85, 146): np.float64(0.4142137913964868), (85, 149): np.float64(-0.003335385210942663), (75, 90): np.float64(-0.0042699270542513315), (86, 90): np.float64(-0.00475737921923123), (90, 93): np.float64(-0.0016011038713291503), (90, 94): np.float64(-0.0008912419861378543), (90, 95): np.float64(-0.014808203376316635), (90, 96): np.float64(-0.002997833029684385), (90, 97): np.float64(-0.0015907511153109952), (90, 98): np.float64(-0.001745134930764757), (90, 99): np.float64(-0.0026583128221999105), (90, 100): np.float64(-0.002153912379517574), (90, 101): np.float64(-0.0011684458259077356), (90, 102): np.float64(-0.000828121064779066), (90, 103): np.float64(0.0016510817848759236), (90, 104): np.float64(0.000570946533208515 7), (90, 105): np.float64(-0.0038204354565099827), (90, 107): np.float64(-0.00268388623982581), (90, 111): np.float64(0.0033603328011466574), (90, 112): np.float64(0.00409071831566057), (90, 113): np.float64(0.0031492165689036094), (90, 114): np.float64(-0.004233946292012059), (90, 116): np.float64(-0.002185786760959436), (90, 117): np.float64(0.006417837856769511), (90, 119): np.float64(0.0003552913262434377), (90, 121): np.float64(0.0002833392888399544), (90, 123): np.float64(-0.002369469842542435), (90, 127): np.float64(-0.0007863125987378325), (90, 129): np.float64(-0.006172479374049008), (90, 131): np.float64(0.0018212749201863938), (90, 132): np.float64(0.002805765412558661), (90, 133): np.float64(-0.0006672785750992949), (90, 135): np.float64(-0.004997767504654145), (90, 136): np.float64(-0.00596707417151221), (90, 137): np.float64(-1.756727878160673e-05), (90, 138): n

p.float64(-3.259599799334961e-05), (90, 139): np.float64(-0.001369401831510652), (90, 140): np.float64(-0.017477591564221448), (90, 143): np.float64(-0.00027007395653313895), (90, 145): np.float64(-0.0027621280922437447), (90, 146): np.float64(0.18145984028016288), (90, 148): np.float64(-0.0009027843654587931), (90, 149): np.float64(0.003983787198778875), (75, 91): np.float64(-0.037391139441621354), (82, 91): np.float64(-0.02180559253878407), (87, 91): np.float64(-0.021269526395436483), (89, 91): np.float64(-0.01616472288623193), (91, 92): np.float64(0.0055590435219881875), (91, 93): np.float64(0.028844408613932747), (91, 95): np.float64(0.003260598188046862), (91, 96): np.float64(0.010767522360979068), (91, 97): np.float64(0.032703367777141897), (91, 98): np.float64(0.017328941464526147), (91, 100): np.float64(0.04319422685477563), (91, 101): np.float64(0.011081673349724921), (91, 103): np.float64(0.021175706471338162), (91, 104): np.float64(0.01080105792660087), (91, 106): np.float64(0.009418759916833835), (91, 109): np.float64(0.02444969469800193), (91, 110): np.float64(0.006756362267170635), (91, 113): np.float64(0.05782431681998474), (91, 115): np.float64(0.010429757893264972), (91, 116): np.float64(0.0055205358309375644), (91, 124): np.float64(0.007524191841352248), (91, 127): np.float64(0.009982981028239784), (91, 128): np.float64(0.009454264249483615), (91, 129): np.float64(0.021946117955292867), (91, 130): np.float64(0.027715386741669547), (91, 131): np.float64(0.025671276101020435), (91, 134): np.float64(0.007963787188041426), (91, 135): np.float64(0.017875145724355654), (91, 143): np.float64(0.010319555550926255), (91, 145): np.float64(0.004642273646486634), (91, 147): np.float64(0.006805711565785745), (91, 149): np.float64(0.014251884699951893), (75, 92): np.float64(-0.00868766765773395), (87, 92): np.float64(-0.0033200613392425377), (88, 92): np.float64(-0.007914861850565627), (92, 93): np.float64(0.005624547233183589), (92, 94): np.float64(0.0017087624647447778), (92, 95): np.float64(-0.003960939708946921), (92, 96): np.float64(0.006852760920096006), (92, 98): np.float64(0.003160749965882869), (92, 100): np.float64(-0.00013148015789401016), (92, 101): np.float64(0.001226026507102114), (92, 103): np.float64(0.003469821540117942), (92, 104): np.float64(0.0022156200853441902), (92, 105): np.float64(-0.0027951445010299526), (92, 107): np.float64(-0.00018576483256775738), (92, 109): np.float64(0.008457783136051476), (92, 110): np.float64(0.0007070105826975408), (92, 114): np.float64(-0.0008025473039535412), (92, 117): np.float64(0.0070091213404161945), (92, 118): np.float64(-1.6932951551181687e-05), (92, 122): np.float64(0.0031506802298559644), (92, 123): np.float64(0.0026258155129412395), (92, 124): np.float64(0.0005544922595902731 8), (92, 132): np.float64(0.006558663347801937), (92, 136): np.float64(-0.0006048323275441918), (92, 138): np.float64(0.0024005275517792564), (92, 140): np.float64(-0.0007891412628787844), (92, 142): np.float64(0.001731178191380361), (92, 144): np.float64(0.014980419688844941), (92, 149): np.float64(0.0071541928373988345), (79, 95): np.float64(-0.0056970966694027125), (87, 95): np.float64(-0.007796558445436166), (89, 95): np.float64(-0.010842546871017489), (93, 95): np.float64(-0.004972287815651486), (95, 96): np.float64(0.02556115485576774), (95, 97): np.float64(0.005782551329034003), (95, 98): np.float64(0.004338061396030104), (95, 107): np.float64(0.011949792753444136), (95, 108): np.float64(0.011058777068836938), (95, 110): np.float64(0.005490925462440755), (95, 111): np.float64(0.014684300568805006), (95, 112): np.float64(0.024566569912246705), (95, 113): np.float64(0.012360937180111723), (95, 114): np.float64(0.002753909643915588), (95, 116): np.float64(0.014614749116633846), (95, 119): np.float64(0.025975417978204175), (95, 121): np.float64(0.010952302821872777), (95, 124): np.float64(0.003043193289842684), (95, 125): np.float64(0.006038202019356124), (95, 127): np.float64(0.005539020954030143), (95, 129): np.float64(0.008136906018904072), (95, 132): np.float64(0.01089983330251854), (95, 135): np.float64(-0.0004195115064492518), (95, 136): np.float64(0.003195286584304084), (95, 138): np.float64(0.0046525265683250985), (95, 139): np.float64(0.0048914135106175516), (95, 140): np.float64(0.00257671228109375 4), (95, 141): np.float64(-0.0007531949601808318), (95, 142): np.float64(0.016799655707061364), (95, 144): np.float64(0.005957632711786549), (95, 145): np.float64(0.013968495635263327), (95, 146): np.float64(0.12441356558013052), (95, 147): np.float64(0.015491065781599363), (95, 148): np.float64(0.03212210040616528), (75, 98): np.float64(-0.014254086919113871), (79, 98): np.float64(-0.005996399746906227), (82, 98): np.float64(-0.011634423447785749), (87, 98): np.float64(-0.004641003013108994), (96, 98): np.float64(-0.002218806708714675), (97, 98): np.float64(-0.0009456919464992966), (98, 100): np.float64(-0.0007274113015085233), (98, 101): np.float64(0.0006234957897663327), (98, 102): np.float64(0.003922650457285208), (98, 103): np.float64(0.018407940470659193), (98, 105): np.float64(-0.005122465864123693), (98, 106): np.float64(-0.008376095774378318), (98, 10

7): np.float64(-0.005686445659498814), (98, 108): np.float64(0.0023210056937383546), (98, 109): np.float64(0.006903311065148369), (98, 110): np.float64(-0.0008622641552532685), (98, 111): np.float64(0.003550651393007559), (98, 113): np.float64(0.013604170693809202), (98, 114): np.float64(-0.0010175999276525426), (98, 115): np.float64(0.01048898804216074), (98, 116): np.float64(-0.0007992436111878783), (98, 120): np.float64(0.0014524555358693743), (98, 121): np.float64(0.0017508816333703624), (98, 122): np.float64(0.002295883009754754), (98, 123): np.float64(0.002022788669977286), (98, 125): np.float64(0.001163363003114716), (98, 126): np.float64(-0.01586096523024862), (98, 127): np.float64(0.0018407078084813502), (98, 128): np.float64(0.0074625665566051165), (98, 129): np.float64(0.0005954367118971689), (98, 130): np.float64(-0.0032008819805854263), (98, 131): np.float64(0.004337538760354137), (98, 132): np.float64(0.006466478648999238), (98, 134): np.float64(0.0012548675948928327), (98, 135): np.float64(-0.0037166418668745536), (98, 136): np.float64(-0.0019453000200853313), (98, 137): np.float64(0.003182554982139686), (98, 138): np.float64(0.003524300800853455), (98, 140): np.float64(-0.003018643327086988), (98, 144): np.float64(0.015276063493807035), (98, 148): np.float64(0.0010686307098740657), (98, 149): np.float64(0.007598945209366769), (75, 99): np.float64(-0.004623210738666506), (82, 99): np.float64(-0.021484812379593977), (87, 99): np.float64(-0.0037120225284821037), (88, 99): np.float64(-0.0066963373300412575), (93, 99): np.float64(-0.0025162363743391407), (96, 99): np.float64(-0.00394748167645224), (99, 103): np.float64(0.003778158838293458), (99, 104): np.float64(0.004488090196397798), (99, 105): np.float64(-0.0024012336882403233), (99, 108): np.float64(0.005016031152217829), (99, 112): np.float64(0.014774715566897965), (99, 113): np.float64(0.008191342316044935), (99, 115): np.float64(0.008021411319566504), (99, 116): np.float64(0.0010950819013219243), (99, 119): np.float64(0.005779613876085120 5), (99, 121): np.float64(0.0028036537764163155), (99, 122): np.float64(0.004004527278344164), (99, 124): np.float64(0.001632943226161556), (99, 125): np.float64(0.003972626825603424), (99, 127): np.float64(0.003238787391368406), (99, 131): np.float64(0.013890160067802998), (99, 132): np.float64(0.011214514207881733), (99, 135): np.float64(-0.013650893459136525), (99, 139): np.float64(0.0026169957215718864), (99, 142): np.float64(0.0028853447882886928), (99, 143): np.float64(0.007376590190337076), (99, 146): np.float64(0.07278305004712846), (99, 147): np.float64(0.0018798630436695706), (99, 148): np.float64(0.002328943040255307), (75, 100): np.float64(-0.006497511353245501), (79, 100): np.float64(-0.004932234178485718), (82, 100): np.float64(-0.006671846424160443), (86, 100): np.float64(-0.009253852431145605), (87, 100): np.float64(-0.004867618682689479), (88, 100): np.float64(-0.015592516643701858), (89, 100): np.float64(-0.008704160675522977), (93, 100): np.float64(-0.0013070097184794667), (96, 100): np.float64(-0.002445454191833529), (97, 100): np.float64(-0.0017822626546264656), (100, 101): np.float64(0.0016926191438056547), (100, 102): np.float64(0.001397315238071415), (100, 105): np.float64(-0.007340030663847816), (100, 106): np.float64(-0.001651471099150661), (100, 108): np.float64(0.011847172691857852), (100, 109): np.float64(0.005207454736568988), (100, 110): np.float64(0.00112916045961935), (100, 113): np.float64(0.003919857075624936), (100, 114): np.float64(-0.0002878718973319353), (100, 118): np.float64(0.0004699102643840803), (100, 120): np.float64(0.003170502646073235), (100, 125): np.float64(0.0018583238373388886), (100, 127): np.float64(0.002853062374917617), (100, 130): np.float64(-6.873484465222706e-05), (100, 135): np.float64(-0.003899845185063083), (100, 136): np.float64(-0.0005979643331295636), (100, 138): np.float64(0.005259893806584409), (100, 139): np.float64(0.0027404753847390852), (100, 140): np.float64(-0.00016135023987793993), (100, 141): np.float64(-0.00664600787393355), (100, 143): np.float64(0.002933520058968672), (100, 144): np.float64(0.005568156121719228), (100, 145): np.float64(-0.0002879758927337232), (78, 102): np.float64(-0.0024028370293892876), (79, 102): np.float64(-0.002518602495577537), (87, 102): np.float64(-0.0021865266595083245), (88, 102): np.float64(-0.0029699521287985613), (93, 102): np.float64(0.0003405333849728208), (101, 102): np.float64(0.0002866720240743889), (102, 103): np.float64(0.0022132543487551927), (102, 106): np.float64(-0.007335450445363197), (102, 109): np.float64(0.010389610731814143), (102, 112): np.float64(0.0015474054455751386), (102, 113): np.float64(0.020860542429821563), (102, 114): np.float64(-0.0015101263353659836), (102, 116): np.float64(-0.0017611581459890877), (102, 118): np.float64(-0.0013290458093280723), (102, 122): np.float64(0.004745660979951784), (102, 123): np.float64(0.00174859352494701), (102, 124): np.float64(-0.0009956738870124729), (102, 126): np.float64(-0.009022644314194812), (102, 127): np.float64(0.0003820161567986269), (102, 129): np.float64(-0.0007886402360781756), (102, 130): np.float64(-0.001663165416918230

3), (102, 132): np.float64(0.004127798351630421), (102, 133): np.float64(0.0006287758176335012), (102, 134): np.float64(0.00057
76131208501386), (102, 137): np.float64(0.0007596310399890217), (102, 140): np.float64(-0.002085953881710778), (102, 142): np.f
loat64(-0.0007877467779326384), (102, 144): np.float64(0.0033222338948385613), (102, 145): np.float64(-0.008312748843173677),
(102, 146): np.float64(0.14693266028812388), (102, 147): np.float64(-0.00024354389302396154), (78, 104): np.float64(-0.00303454
5161005404), (82, 104): np.float64(-0.0007151431066193122), (89, 104): np.float64(-0.005465471226136038), (94, 104): np.float64
(0.0005918614190233434), (96, 104): np.float64(0.0027535131016464235), (104, 107): np.float64(-0.003932320089876062), (104, 11
0): np.float64(-0.005609009258585809), (104, 116): np.float64(-0.003119618609454608), (104, 117): np.float64(0.0036779863737373
2), (104, 118): np.float64(-0.003086655623930735), (104, 119): np.float64(-7.979642267777679e-05), (104, 122): np.float64(0.001
2760987063111847), (104, 123): np.float64(-0.003152015876885938), (104, 128): np.float64(0.0018239451755995787), (104, 130): n
p.float64(-0.0038991897821787965), (104, 131): np.float64(0.002387114654552576), (104, 134): np.float64(-0.000646068016802788
3), (104, 136): np.float64(-0.003558584542968004), (104, 144): np.float64(0.005643700840269525), (104, 145): np.float64(-0.0031
825874951490985), (104, 146): np.float64(0.21314764956056487), (104, 147): np.float64(-0.0013128687666746485), (104, 149): np.f
loat64(0.0014482098383354815), (75, 105): np.float64(-0.011160726542052711), (78, 105): np.float64(-0.01753583003477024), (82,
105): np.float64(-0.007552520621173587), (86, 105): np.float64(-0.012620353133279963), (89, 105): np.float64(-0.006875485420661
678), (93, 105): np.float64(-0.007911457325430525), (105, 106): np.float64(0.0003120530931838342), (105, 107): np.float64(0.002
7003638768074768), (105, 114): np.float64(0.0023298692809765004), (105, 116): np.float64(0.0069206514546429265), (105, 118): n
p.float64(0.0021396228108034455), (105, 119): np.float64(0.01062688877868729), (105, 120): np.float64(0.005912643701451718), (1
05, 124): np.float64(0.005817419867211037), (105, 127): np.float64(0.014169825762442315), (105, 131): np.float64(0.007345125823
463181), (105, 132): np.float64(0.04096895039236418), (105, 133): np.float64(0.004221391253170173), (105, 136): np.float64(0.00
34608911567152777), (105, 139): np.float64(0.023695062814068194), (105, 141): np.float64(-0.002463430158673249), (105, 144): n
p.float64(0.03250495168573275), (105, 145): np.float64(0.004070204205517853), (105, 147): np.float64(0.0036905838179360157), (1
05, 148): np.float64(0.017498769606929854), (105, 149): np.float64(0.005738901293028825), (75, 106): np.float64(-0.013670249555
472854), (79, 106): np.float64(-0.007667306203285756), (93, 106): np.float64(-0.005510160432755882), (94, 106): np.float64(-0.0
053479680309676016), (96, 106): np.float64(-0.004736419961653364), (97, 106): np.float64(-0.003682939311763618), (106, 107): n
p.float64(0.01310737170503487), (106, 109): np.float64(0.017331144948665365), (106, 110): np.float64(0.0022603458823708864), (1
06, 111): np.float64(0.013714826709734583), (106, 112): np.float64(0.021189198835162366), (106, 116): np.float64(0.003123414068
2653887), (106, 118): np.float64(0.0042231069500951865), (106, 124): np.float64(0.0021121019394184813), (106, 126): np.float64
(-0.013355415663861721), (106, 129): np.float64(0.0029549625741542985), (106, 131): np.float64(0.005139423069007384), (106, 13
4): np.float64(0.0061780159060755265), (106, 135): np.float64(-0.0012147024018417528), (106, 137): np.float64(0.007343687003315
886), (106, 142): np.float64(0.0025135603655454582), (106, 143): np.float64(0.016540787679374835), (75, 107): np.float64(-0.006
925972611427888), (87, 107): np.float64(-0.007287177326147029), (89, 107): np.float64(-0.004127024668044682), (101, 107): np.fl
oat64(-0.0009831440703355616), (103, 107): np.float64(-0.019081744409840225), (107, 108): np.float64(0.0034802375970144086), (1
07, 110): np.float64(0.0006933175019924865), (107, 114): np.float64(-0.0006439809151468294), (107, 116): np.float64(-4.60298795
5115941e-05), (107, 118): np.float64(0.00012845911305462644), (107, 119): np.float64(0.01968409459587572), (107, 120): np.float
64(0.006479616543262692), (107, 121): np.float64(0.0029201067103025976), (107, 124): np.float64(0.0006535433149785 62), (107, 12
6): np.float64(-0.00666129429074119), (107, 128): np.float64(0.004739962635055204), (107, 131): np.float64(0.0034356974261332 7
4), (107, 134): np.float64(0.002286154567002374), (107, 137): np.float64(0.005084588552076534), (107, 140): np.float64(-0.00073
11989014130549), (107, 141): np.float64(-0.0037234956394442257), (107, 147): np.float64(0.0016249056317777 98), (107, 148): np.f
loat64(0.007842607996179892), (107, 149): np.float64(0.007773431773610733), (79, 108): np.float64(0.001065350655547071), (86, 1
08): np.float64(-0.00034976135679135127), (89, 108): np.float64(-0.005146232189553389), (93, 108): np.float64(0.002875266692158
26), (97, 108): np.float64(0.0027581985665102063), (101, 108): np.float64(0.0032856225660602926), (103, 108): np.float64(-0.001

3090933330642597), (108, 109): np.float64(0.0017972434316833338), (108, 112): np.float64(0.00037488434047912436), (108, 113): np.float64(0.0019830881234082294), (108, 114): np.float64(-0.0054479902124924704), (108, 115): np.float64(0.00764255546963026), (108, 116): np.float64(-0.005860388267291169), (108, 118): np.float64(-0.005558032484667371), (108, 121): np.float64(-0.0022699288604209112), (108, 122): np.float64(0.0005704150643376378), (108, 123): np.float64(-0.002283723617500269), (108, 124): np.float64(-0.0024317771160364643), (108, 125): np.float64(-0.0007963454658046949), (108, 128): np.float64(0.00419761118262516), (108, 129): np.float64(-0.002113997234511964), (108, 130): np.float64(-0.004905989076585834), (108, 134): np.float64(-0.000925632762314373), (108, 136): np.float64(-0.0094415901371002), (108, 137): np.float64(-0.0008838062230627129), (108, 138): np.float64(-0.0009841961977943138), (108, 141): np.float64(-0.015015585149474528), (108, 142): np.float64(-0.006384726017613082), (108, 143): np.float64(-0.0012708510718045139), (108, 145): np.float64(-0.003522814919709465), (108, 146): np.float64(0.1752431358659407), (108, 147): np.float64(-0.0020510211918887255), (108, 148): np.float64(-0.0030190667633342113), (75, 110): np.float64(-0.01778645069805949), (86, 110): np.float64(-0.005562209034338785), (88, 110): np.float64(-0.007209993404909405), (89, 110): np.float64(-0.009823734237574482), (93, 110): np.float64(-0.0027087387440021073), (96, 110): np.float64(-0.0014832890065434015), (97, 110): np.float64(-0.003664840082835596), (103, 110): np.float64(-0.0032200831555489486), (110, 112): np.float64(0.005020411742836464), (110, 113): np.float64(0.00535280393594996), (110, 115): np.float64(0.006801979266117174), (110, 117): np.float64(0.04874928450540959), (110, 120): np.float64(0.0024032147513909008), (110, 121): np.float64(0.0018492485761774963), (110, 122): np.float64(0.0045378397711715954), (110, 123): np.float64(0.0014624296393211684), (110, 124): np.float64(0.0002097581834595898), (110, 128): np.float64(0.004196743376884612), (110, 129): np.float64(0.000829998954019224), (110, 130): np.float64(-0.0007967036903263285), (110, 132): np.float64(0.004200651290428679), (110, 133): np.float64(0.005991551979960056), (110, 134): np.float64(0.011250522613256481), (110, 136): np.float64(-0.0008257662760675122), (110, 139): np.float64(0.0004623679156831196), (110, 141): np.float64(-0.016186544806268677), (110, 143): np.float64(0.0036871720708634825), (110, 149): np.float64(0.025206354728387424), (75, 114): np.float64(-0.006290235301002118), (78, 114): np.float64(-0.004577018083859118), (79, 114): np.float64(-0.002390647628663221), (86, 114): np.float64(-0.005067979156998009), (87, 114): np.float64(-0.0033808424553040013), (89, 114): np.float64(-0.004870799187140253), (93, 114): np.float64(-0.0013123953876043415), (96, 114): np.float64(-0.0036725083778621336), (97, 114): np.float64(-0.003849806275094905), (114, 115): np.float64(0.029594001125623638), (114, 119): np.float64(0.006723950593731189), (114, 121): np.float64(0.0029712321607348125), (114, 123): np.float64(0.003318196980600764), (114, 125): np.float64(0.003547610283852038), (114, 127): np.float64(0.003676866631538916), (114, 130): np.float64(0.001527348789495258), (114, 131): np.float64(0.014337537372673749), (114, 135): np.float64(-0.004357346602728432), (114, 138): np.float64(0.007466120243870418), (114, 139): np.float64(0.007233056600744503), (114, 140): np.float64(0.0001706237143067489), (114, 141): np.float64(-0.002749828409528165), (114, 142): np.float64(0.001677959837017385), (114, 144): np.float64(0.0037970703364387706), (114, 149): np.float64(0.011282545758664442), (78, 116): np.float64(-0.008623571054169825), (82, 116): np.float64(-0.011556528969759875), (86, 116): np.float64(-0.0029409147402545415), (89, 116): np.float64(-0.005329805135305248), (93, 116): np.float64(-0.0019116560743351642), (103, 116): np.float64(-0.005387535254472339), (112, 116): np.float64(-0.005109590373619642), (113, 116): np.float64(-0.009686885675430586), (116, 117): np.float64(0.006023827358862789), (116, 119): np.float64(0.0021537958376152697), (116, 122): np.float64(0.005237569111119779), (116, 125): np.float64(0.0018800368145245707), (116, 127): np.float64(0.0019725375599252398), (116, 129): np.float64(0.004352878938370591), (116, 131): np.float64(0.006262719954632392), (116, 135): np.float64(-0.0032702489341978193), (116, 137): np.float64(0.005899447052768292), (116, 139): np.float64(0.0011682713776861942), (116, 143): np.float64(0.0023692313330083946), (116, 144): np.float64(0.012332689732349177), (116, 145): np.float64(-0.0002683311959209658), (116, 148): np.float64(0.0014540474581078951), (75, 117): np.float64(0.002685750130435924), (78, 117): np.float64(0.023552028354921944), (79, 117): np.float64(0.01771518771710278), (86, 117): np.float64(0.0034779876527869383), (89, 117): np.float64(0.005291164093012096), (93, 117): np.float64(0.016992046685723548), (97, 117): np.float64(0.018381707873978706), (111, 117): np.float64(0.003123261754591227), (112, 117): np.float64(0.003858622717894285), (115, 117): np.float64(0.0010231208288324178), (117, 120): n

p.float64(-0.007406511423587963), (117, 121): np.float64(-0.03175887030686103), (117, 130): np.float64(-0.05330195456228463), (117, 132): np.float64(-0.0028490030584397055), (117, 133): np.float64(-0.004321194764655189), (117, 135): np.float64(-0.026682800139513133), (117, 137): np.float64(-0.005686770914793484), (117, 138): np.float64(-0.012754371831368091), (117, 139): np.float64(-0.01265256107709278), (117, 140): np.float64(-0.006753935215720148), (117, 141): np.float64(-0.011482054963176835), (117, 143): np.float64(-0.0218460242673234), (117, 144): np.float64(-0.005515214670261043), (117, 146): np.float64(0.5078446862853969), (117, 148): np.float64(-0.008564959248230844), (78, 118): np.float64(-0.016063992438725118), (82, 118): np.float64(-0.006485647463296135), (86, 118): np.float64(-0.0028211195532011684), (88, 118): np.float64(-0.01535799349147763), (97, 118): np.float64(-0.009764546979775107), (103, 118): np.float64(-0.007974852225511121), (111, 118): np.float64(-0.00925580140544386), (112, 118): np.float64(-0.004961763937754865), (115, 118): np.float64(-0.016616727713002193), (118, 119): np.float64(0.002803092512473953), (118, 121): np.float64(0.01002525728166517), (118, 124): np.float64(0.001008603869005492), (118, 129): np.float64(0.00206005676920844), (118, 130): np.float64(-0.00018350308815106764), (118, 131): np.float64(0.003208737868371983), (118, 135): np.float64(-0.01827564980263113), (118, 138): np.float64(0.002253695557598269), (118, 140): np.float64(-0.0005446519692770337), (118, 142): np.float64(0.0021673678582792622), (118, 143): np.float64(0.001968682198317523), (118, 144): np.float64(0.003145000554043776), (118, 145): np.float64(-0.00030165799681687556), (118, 147): np.float64(0.0016310399796406671), (82, 119): np.float64(-0.0012489697098126), (86, 119): np.float64(-0.0017180120711046112), (93, 119): np.float64(0.0016253748467788401), (94, 119): np.float64(0.0022496368141997334), (97, 119): np.float64(0.0011029806887088071), (101, 119): np.float64(0.003849864151370707), (109, 119): np.float64(-0.00781103612501361), (111, 119): np.float64(-0.00431191004728644), (112, 119): np.float64(-0.004343870541183217), (119, 121): np.float64(0.0002853673966797398), (119, 126): np.float64(-0.010513739460973715), (119, 127): np.float64(-0.0016410730314283483), (119, 128): np.float64(0.00275619894493601), (119, 129): np.float64(-0.0023842498898735214), (119, 131): np.float64(0.0017913602979441167), (119, 132): np.float64(0.007718213234049567), (119, 133): np.float64(-0.0014391442528078843), (119, 135): np.float64(-0.004876600287329542), (119, 137): np.float64(-0.0009045003360349482), (119, 140): np.float64(-0.003940070879966015), (119, 141): np.float64(-0.016862822186175536), (119, 142): np.float64(-0.007177358842475211), (119, 145): np.float64(-0.01691492221913649), (119, 146): np.float64(0.262937525352023), (119, 147): np.float64(-0.0010694362284174926), (119, 149): np.float64(0.003530410453875107), (79, 120): np.float64(-0.0012873362222491158), (87, 120): np.float64(-0.0033922231377439267), (89, 120): np.float64(-0.0021685187943945483), (94, 120): np.float64(0.00019309210263815038), (97, 120): np.float64(0.00048611932829410997), (101, 120): np.float64(0.0033048305253492245), (103, 120): np.float64(-0.008394960146022618), (109, 120): np.float64(-0.002891395537077655), (111, 120): np.float64(-0.02270282075880407), (112, 120): np.float64(-0.0011274949000915015), (113, 120): np.float64(-0.003535397156511331), (120, 121): np.float64(0.0005697950284599702), (120, 122): np.float64(0.002132343465835663), (120, 125): np.float64(0.00038391572031126807), (120, 126): np.float64(-0.012987587723164714), (120, 128): np.float64(0.003714827234654496), (120, 132): np.float64(0.0036605757165723185), (120, 136): np.float64(-0.00536084401992683), (120, 138): np.float64(0.0005783148654571977), (120, 140): np.float64(-0.0019498864616749752), (120, 142): np.float64(-0.0005726720198618451), (120, 143): np.float64(6.567098416151227e-06), (120, 144): np.float64(0.002904175674740893 5), (120, 145): np.float64(-0.0022630229836483246), (120, 146): np.float64(0.1289630197274985), (120, 147): np.float64(-0.0009439278934800342), (120, 148): np.float64(-0.0017493978921652644), (120, 149): np.float64(0.002649791005888856), (87, 121): np.float64(-0.0006498600833234162), (89, 121): np.float64(-0.002508187182395804), (93, 121): np.float64(0.003335836955420703), (96, 121): np.float64(0.0019168215870484441), (101, 121): np.float64(0.0013814296434437863), (103, 121): np.float64(-0.00159080212211743), (112, 121): np.float64(-0.0005696362458723414), (113, 121): np.float64(-0.0014477267106404003), (121, 123): np.float64(-0.0028053153187078436), (121, 125): np.float64(-0.0008913907341962536), (121, 127): np.float64(-0.0007533797212860571), (121, 128): np.float64(0.005428652895459087), (121, 131): np.float64(0.006812359453664873), (121, 132): np.float64(0.014320617691532176), (121, 133): np.float64(-0.0005411687344748286), (121, 136): np.float64(-0.0043928892536458), (121, 138): np.float64(-0.0006463611668209929), (121, 140): np.float64(-0.0027819333752401292), (121, 141): np.float64(-0.0060526371932529315), (121, 142): np.float64(-0.0

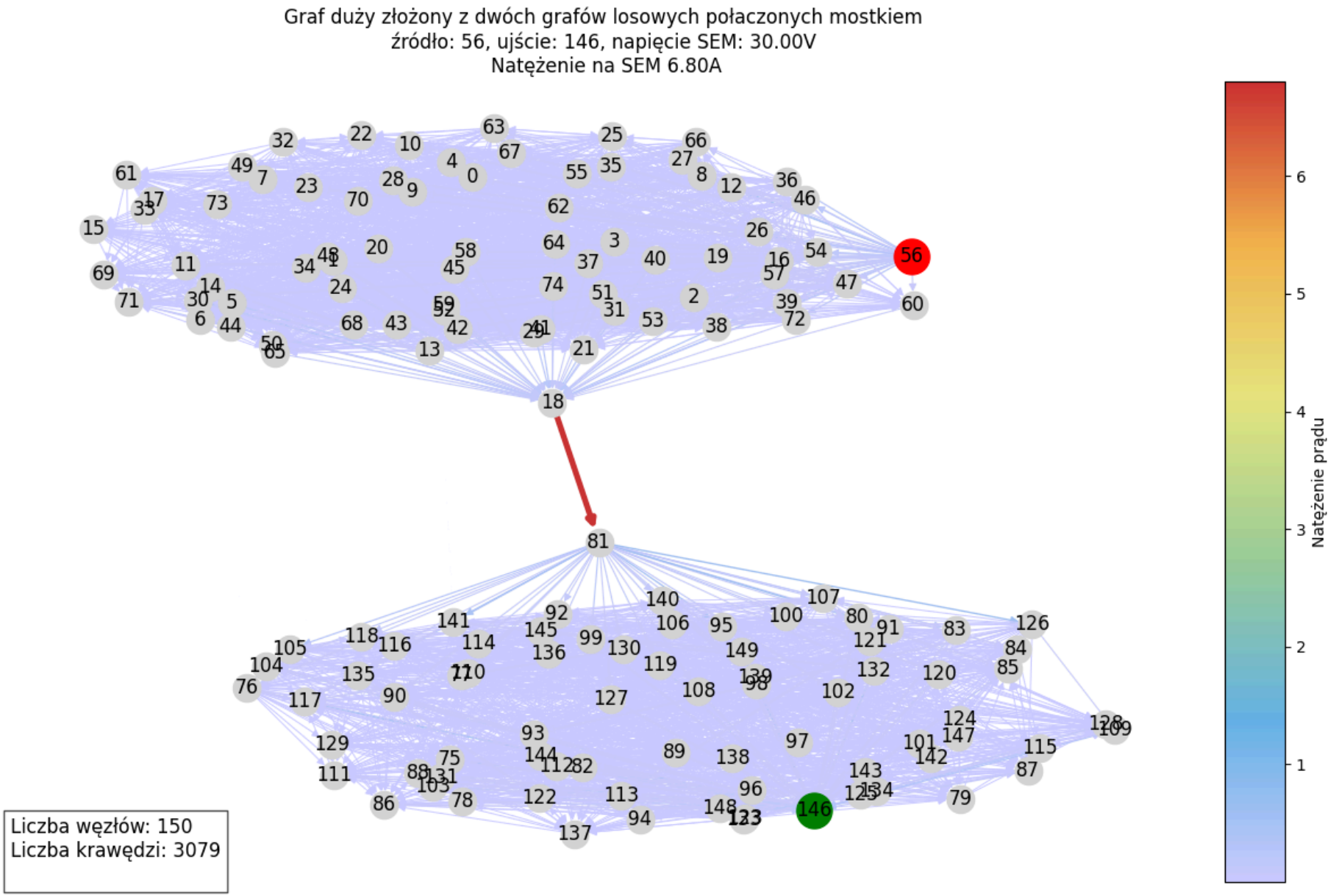
02730681256919461), (121, 143): np.float64(-0.0007434628902466535), (121, 144): np.float64(0.0010608970585706209), (121, 146): np.float64(0.09814284509159289), (121, 147): np.float64(-0.00448942641524381), (78, 126): np.float64(-0.025543483733301738), (87, 126): np.float64(-0.010241017821089781), (93, 126): np.float64(-0.06966891508839472), (101, 126): np.float64(-0.04752421730420683), (109, 126): np.float64(-0.0701656906357896), (111, 126): np.float64(-0.015220097064437044), (112, 126): np.float64(-0.009236066703098053), (122, 126): np.float64(-0.02656149071878493), (123, 126): np.float64(-0.0256143906799583), (124, 126): np.float64(-0.01760606838783628), (126, 129): np.float64(0.022124329865223828), (126, 132): np.float64(0.017064551588327426), (126, 133): np.float64(0.022107640366110206), (126, 138): np.float64(0.011199341451545502), (126, 139): np.float64(0.009216298911887327), (126, 140): np.float64(0.03284688280720561), (126, 141): np.float64(0.006161029037364165), (126, 143): np.float64(0.01448965641202241), (126, 145): np.float64(0.014487313656754895), (126, 147): np.float64(0.00804809175189293), (126, 149): np.float64(0.011546571112008262), (75, 127): np.float64(-0.0029110899062966804), (78, 127): np.float64(-0.0017389163096266013), (79, 127): np.float64(-0.0013103743585192596), (82, 127): np.float64(-0.0017976019468079858), (86, 127): np.float64(-0.0013536022247582866), (87, 127): np.float64(-0.0018803532878078181), (88, 127): np.float64(-0.0031596258247239472), (89, 127): np.float64(-0.009220125574885198), (96, 127): np.float64(1.4404551298858552e-05), (101, 127): np.float64(0.000608610465010382 3), (103, 127): np.float64(-0.009244030079582131), (109, 127): np.float64(-0.013557535724271806), (112, 127): np.float64(-0.002025911246478506), (122, 127): np.float64(-0.0029834460427164877), (124, 127): np.float64(0.001421126407667441), (125, 127): np.float64(-0.00047491935654036263), (127, 129): np.float64(-0.0006502424069547135), (127, 130): np.float64(-0.002136068424766038), (127, 134): np.float64(0.00031663605946053016), (127, 141): np.float64(-0.004847948613980615), (127, 146): np.float64(0.09418058458712915), (127, 147): np.float64(-0.0005374753297395014), (127, 148): np.float64(-0.0004414574438656133), (127, 149): np.float64(0.005150248922049682), (87, 130): np.float64(-0.012166540679737119), (93, 130): np.float64(-0.00358622894483014), (96, 130): np.float64(-0.0031392464169121494), (97, 130): np.float64(-0.003605158409780954), (101, 130): np.float64(-0.0015695764363549646), (103, 130): np.float64(-0.008524111804570127), (111, 130): np.float64(-0.004633252026120373), (112, 130): np.float64(-0.006483679572160159), (113, 130): np.float64(-0.015249173942736853), (115, 130): np.float64(-0.006450499664356331), (122, 130): np.float64(-0.014073814716382037), (123, 130): np.float64(-0.014719475742306896), (124, 130): np.float64(-0.0006386136389607805), (128, 130): np.float64(-0.004594284202965178), (129, 130): np.float64(-0.0020376171391484504), (130, 133): np.float64(0.002355210614268012 3), (130, 135): np.float64(-0.002868102059693627), (130, 137): np.float64(0.003292359939947423), (130, 139): np.float64(0.00086111243938798287), (130, 141): np.float64(-0.01990666476535732), (130, 142): np.float64(0.0020791231928275534), (130, 144): np.float64(0.004131107167306711), (130, 145): np.float64(-0.00013478463895161275), (130, 147): np.float64(0.004517262434349734), (130, 148): np.float64(0.001463856278007052), (130, 149): np.float64(0.005644585173758307), (78, 132): np.float64(0.0024144005163465095), (79, 132): np.float64(0.0028659068748168705), (82, 132): np.float64(0.0019461121407691139), (88, 132): np.float64(0.0006823902453603607), (89, 132): np.float64(0.0025644392636819447), (93, 132): np.float64(0.026289035107492316), (96, 132): np.float64(0.003650664165115949), (97, 132): np.float64(0.004949030625338048), (103, 132): np.float64(0.004676222726636762), (109, 132): np.float64(0.0004610385154849254), (112, 132): np.float64(0.006447778676228709), (113, 132): np.float64(0.009498493543957076), (115, 132): np.float64(-0.00034851350681518135), (123, 132): np.float64(0.003166437914091511), (125, 132): np.float64(0.015688433899928866), (131, 132): np.float64(0.002635058543375449), (132, 134): np.float64(-0.004577684891292218), (132, 138): np.float64(-0.003089023106176186), (132, 140): np.float64(-0.005057347855044646), (132, 141): np.float64(-0.0674212719700866), (132, 143): np.float64(-0.004047346553070229), (132, 146): np.float64(0.4453881961798761), (132, 147): np.float64(-0.00569542927755082), (132, 148): np.float64(-0.009371262155339623), (132, 149): np.float64(-0.0031671081645966445), (75, 135): np.float64(-0.0063489523821519175), (79, 135): np.float64(-0.00516456459980991), (82, 135): np.float64(-0.01070814838076963), (86, 135): np.float64(-0.0055088718542679205), (88, 135): np.float64(-0.00836621249050277), (89, 135): np.float64(-0.023126866875233104), (94, 135): np.float64(-0.0062223441981257175), (103, 135): np.float64(-0.005888942607699826), (109, 135): np.float64(-0.00786368921002857), (113, 135): np.float64(-0.011816517614630755), (115, 135): np.float64(-0.014540910561548237), (122, 135): np.float64(-

0.05422486392220074), (131, 135): np.float64(-0.026335848139697533), (135, 137): np.float64(0.0064118705811381255), (135, 138): np.float64(0.006155074973495165), (135, 140): np.float64(0.0034819172044542766), (135, 141): np.float64(-0.0018642779383388416), (135, 143): np.float64(0.008603465202327368), (135, 144): np.float64(0.007063744154201005), (135, 145): np.float64(0.011911664196512487), (135, 147): np.float64(0.01021675208065757), (135, 148): np.float64(0.010314181828428404), (135, 149): np.float64(0.011800605292436877), (82, 136): np.float64(-0.004365352184423443), (86, 136): np.float64(-0.004897518786881295), (89, 136): np.float64(-0.0043188861635770045), (93, 136): np.float64(-0.0031290455988664005), (94, 136): np.float64(-0.00484100309028239 1), (96, 136): np.float64(-0.0020379493182353207), (97, 136): np.float64(-0.003671274184554376), (101, 136): np.float64(-0.0017297273981179116), (103, 136): np.float64(-0.007144508824497362), (111, 136): np.float64(-0.0052144512789446635), (112, 136): np.float64(-0.0037148414857829195), (115, 136): np.float64(-0.007849852001143309), (124, 136): np.float64(-0.0016936860465376644), (125, 136): np.float64(-0.0028282017348912228), (131, 136): np.float64(-0.004052233670937068), (133, 136): np.float64(-0.0046236406246823805), (134, 136): np.float64(-0.005463889246934754), (136, 137): np.float64(0.0021885373504782714), (136, 138): np.float64(0.013316440678867735), (136, 140): np.float64(0.0002382143069259995), (136, 141): np.float64(-0.010384762734545071), (136, 142): np.float64(0.0013906863189610697), (75, 139): np.float64(-0.0042561688747031505), (78, 139): np.float64(-0.003257846727453102), (79, 139): np.float64(-0.001564406482818572), (82, 139): np.float64(-0.0023527715731009877), (86, 139): np.float64(-0.002643826840283384), (89, 139): np.float64(-0.008937309861249784), (93, 139): np.float64(-0.00027983313064050444), (101, 139): np.float64(-0.000934504509073933), (109, 139): np.float64(-0.00785996829647757), (111, 139): np.float64(-0.006896328407488644), (112, 139): np.float64(-0.002345332550056553), (122, 139): np.float64(-0.014690668820319866), (123, 139): np.float64(-0.000759505289393765), (128, 139): np.float64(-0.007364701479274825), (129, 139): np.float64(-0.000623996394968125), (133, 139): np.float64(-0.002231532924009349), (139, 140): np.float64(-0.0025146455788798 02), (139, 146): np.float64(0.09990879761776537), (139, 147): np.float64(0.000354971601317238), (139, 148): np.float64(0.0005670844640990691), (82, 140): np.float64(-0.013563668426317782), (93, 140): np.float64(-0.0018413729646636658), (94, 140): np.float64(-0.0017685082038498212), (97, 140): np.float64(-0.001452410396057274), (101, 140): np.float64(-0.003000519454504744), (112, 140): np.float64(-0.004236864646908806), (115, 140): np.float64(-0.007337253956309165), (128, 140): np.float64(-0.003939148449297595), (129, 140): np.float64(-0.009621385347079516), (131, 140): np.float64(-0.009489057720348408), (140, 141): np.float64(-0.016835252869250562), (140, 142): np.float64(0.00725460797027778), (140, 144): np.float64(0.003294833088143728), (140, 145): np.float64(-0.0001405371514361896), (140, 147): np.float64(0.006819208491432285), (140, 149): np.float64(0.005380270017378557), (78, 141): np.float64(-0.007377319057466474), (79, 141): np.float64(-0.009662960319832855), (82, 141): np.float64(-0.0120508766761425), (88, 141): np.float64(-0.0078692314306011402), (93, 141): np.float64(-0.007093065819873792), (94, 141): np.float64(-0.005418893922489059), (97, 141): np.float64(-0.008428345028238587), (109, 141): np.float64(-0.017887612077277754), (112, 141): np.float64(-0.012681881036068278), (113, 141): np.float64(-0.0067172143425040165), (124, 141): np.float64(-0.0041762342563674785), (125, 141): np.float64(-0.005984762709404724), (131, 141): np.float64(-0.007097173054844113), (138, 141): np.float64(-0.010106067205256758), (141, 142): np.float64(0.006152961737394656), (141, 144): np.float64(0.008777364252146423), (141, 145): np.float64(0.008214131785551639), (141, 149): np.float64(0.05535796602259428), (75, 145): np.float64(-0.012820075620109), (88, 145): np.float64(-0.005501156113056359), (93, 145): np.float64(-0.0012612828659941705), (94, 145): np.float64(-0.0025991925884298922), (97, 145): np.float64(-0.0027715871200997134), (111, 145): np.float64(-0.025851496006052462), (112, 145): np.float64(-0.008685910067013032), (124, 145): np.float64(-0.0008364274522263725), (125, 145): np.float64(-0.014509626474456177), (128, 145): np.float64(-0.005189456699352796), (129, 145): np.float64(-0.0012658352388598593), (133, 145): np.float64(-0.003981581129877854), (134, 145): np.float64(-0.012378390244403641), (138, 145): np.float64(-0.006089345888555017), (144, 145): np.float64(-0.0049343745514545245), (145, 147): np.float64(0.0014765651724306622), (145, 149): np.float64(0.0036661567974059484), (75, 149): np.float64(-0.0028871780969166073), (78, 149): np.float64(0.00041956209122533415), (79, 149): np.float64(0.0050722441153146815), (88, 149): np.float64(-0.0006863279519982313), (96, 149): np.float64(0.0041402041074865), (97, 149): np.float64(0.003607749929376896), (109, 149): np.float64(-0.001712712747045651

2), (111, 149): np.float64(-0.0019358267604882044), (112, 149): np.float64(0.000637385376225047), (113, 149): np.float64(-0.000
2448670921914646), (115, 149): np.float64(-0.0015315961126262764), (124, 149): np.float64(0.006254957832554158), (129, 149): n
p.float64(0.007250804164836249), (133, 149): np.float64(0.0020395885410304637), (134, 149): np.float64(0.0017075386959434873),
(138, 149): np.float64(0.0041943448323212545), (143, 149): np.float64(0.0022674683445497346), (144, 149): np.float64(0.00045141
85084948158), (146, 149): np.float64(-0.3405178155154739), (75, 78): np.float64(-0.0014852904983755821), (75, 89): np.float64(-
0.0006643336995427259), (75, 96): np.float64(-0.014553978734583504), (75, 97): np.float64(-0.020719318256718137), (75, 101): n
p.float64(-0.0029729148259591887), (75, 109): np.float64(0.0007662269964590309), (75, 111): np.float64(0.00010631065391674325),
(75, 113): np.float64(-0.0012804878918243206), (75, 123): np.float64(-0.010269679132488435), (75, 125): np.float64(-0.013122102
29604999), (75, 129): np.float64(-0.005468318082043058), (75, 134): np.float64(-0.014266401854706743), (75, 138): np.float64(-
0.013055091929129558), (75, 142): np.float64(-0.0028123814500002624), (75, 146): np.float64(0.2892542327071316), (75, 147): np.
float64(-0.013109256768173707), (75, 148): np.float64(-0.007301816201611556), (79, 86): np.float64(0.001671454744020362), (86,
88): np.float64(0.0015569883748193095), (86, 89): np.float64(0.0017414489272556842), (86, 96): np.float64(-0.001346532125408297
1), (86, 113): np.float64(0.001131530296940602), (86, 115): np.float64(0.007434040988816833), (86, 123): np.float64(-0.0023142123473395867), (86, 125): np.float64(-0.0007865370784920044), (86, 129): np.float64(-0.012445911064865643), (86, 133): np.float64(-0.005880246589362813), (86, 134): np.float64(-0.0017633832442202413), (86, 137): np.float64(-0.0013034910810306504), (86, 138): np.float64(-0.0037120458617239034), (86, 142): np.float64(-0.0017981332765907554), (86, 144): np.float64(0.0009530070031806021), (86, 146): np.float64(0.08422756366276687), (86, 148): np.float64(-0.0015106612369620081), (78, 88): np.float64(0.0023540448926142087), (79, 88): np.float64(0.0019281014816166847), (82, 88): np.float64(0.0017126576924436992), (87, 88): np.float64(0.0037625052889162435), (88, 93): np.float64(-0.0029245670793472314), (88, 94): np.float64(-0.003305987588429664), (88, 122): np.float64(-0.008275378231965643), (88, 124): np.float64(-0.005656243202739552), (88, 125): np.float64(-0.0035875413548068385), (88, 134): np.float64(-0.012820751605289468), (88, 138): np.float64(-0.00747859328932755), (88, 142): np.float64(-0.0033398066540479337), (88, 143): np.float64(-0.003826283479053796), (88, 146): np.float64(0.17218479801965217), (88, 148): np.float64(-0.005243892458154723), (79, 89): np.float64(0.0023239986218475272 7), (82, 89): np.float64(0.000997195237145927), (87, 89): np.float64(0.0011812070459862817), (89, 93): np.float64(-0.005902475040969122), (89, 97): np.float64(-0.002586084367393616), (89, 111): np.float64(0.001926740245032044), (89, 112): np.float64(-0.001874428298375584), (89, 113): np.float64(-1.9991150165616252e-05), (89, 115): np.float64(0.0017787441899624389), (89, 122): np.float64(-0.0005397974270736251), (89, 125): np.float64(-0.002489571181512864), (89, 131): np.float64(-0.0002782705502758264), (89, 133): np.float64(-0.002545672693570742), (89, 134): np.float64(-0.01110501972938771), (89, 142): np.float64(-0.002817354272000008), (89, 146): np.float64(0.1887697078713975), (89, 147): np.float64(-0.003501576159932989), (89, 148): np.float64(-0.0032512236841759096), (78, 93): np.float64(-0.005175850462989835), (82, 93): np.float64(-0.001980638129351237), (93, 94): np.float64(0.0010481032543755656), (93, 96): np.float64(0.0005332542896534372), (93, 101): np.float64(-4.2467661086493624e-05), (93, 103): np.float64(0.012525105795258746), (93, 109): np.float64(0.02294793105188112), (93, 111): np.float64(0.008391693681459488), (93, 115): np.float64(0.009319727558129534), (93, 122): np.float64(0.006850690967169837), (93, 124): np.float64(-0.0009424128287242355), (93, 125): np.float64(0.007823607023344052), (93, 133): np.float64(0.003366430222101254), (93, 142): np.float64(0.00023837160887575634), (93, 147): np.float64(0.0002510437666306577), (93, 148): np.float64(0.0007549803337573998), (79, 94): np.float64(-0.000680214085188343), (87, 94): np.float64(-0.0020193547500486782), (94, 96): np.float64(-8.43971562157195e-05), (94, 97): np.float64(-0.001285268976185795), (94, 101): np.float64(-0.0014327215580782506), (94, 103): np.float64(0.0026717364448965784), (94, 109): np.float64(0.0036511470437961355), (94, 111): np.float64(0.0023803817337826106), (94, 122): np.float64(0.0030683963418030686), (94, 124): np.float64(-0.002287537419445193), (94, 125): np.float64(0.0005454241796329763), (94, 128): np.float64(0.0023624764289237807), (94, 134): np.float64(0.00016104724274339398), (94, 137): np.float64(0.00037777339088667603), (94, 143): np.float64(0.0007197114349360727), (94, 147): np.float64(-0.003081850632016145 3), (94, 148): np.float64(-0.0012962233991990705), (79, 101): np.float64(-0.001083099544294223), (82, 101): np.

float64(-0.0020986571581476164), (87, 101): np.float64(-0.007598652680689761), (101, 111): np.float64(0.003496127052185083), (101, 112): np.float64(0.0022080830392854716), (101, 113): np.float64(0.00406450094660088), (101, 115): np.float64(0.0038886525627443063), (101, 122): np.float64(0.004008005158757827), (101, 123): np.float64(0.0005211558416810003), (101, 125): np.float64(0.0009346566544415793), (101, 128): np.float64(0.005470757724217574), (101, 129): np.float64(0.00012823721816935144), (101, 131): np.float64(0.002481812813653651), (101, 133): np.float64(0.005866113207279126), (101, 134): np.float64(0.0013380065016880824), (101, 137): np.float64(0.001207674514351919), (101, 138): np.float64(0.0012402695142318197), (101, 148): np.float64(0.0003743052876642707 3), (82, 103): np.float64(0.0005201769015199152), (87, 103): np.float64(0.0002894735613619629), (96, 103): np.float64(0.011068677203597796), (103, 109): np.float64(0.0059304269678183), (103, 111): np.float64(0.0009210806086520566), (103, 112): np.float64(-0.0020326663427646303), (103, 113): np.float64(0.0004467997803598226), (103, 122): np.float64(-0.00011139453528521322), (103, 123): np.float64(-0.002208539297172107), (103, 125): np.float64(-0.008477571887929948), (103, 128): np.float64(0.0014673138252963517), (103, 133): np.float64(-0.0039335413529152105), (103, 134): np.float64(-0.0016615500314261266), (103, 138): np.float64(-0.0012874237535745183), (103, 146): np.float64(0.1714741642001432), (78, 111): np.float64(0.005467284017874583), (79, 111): np.float64(0.003916358642836008), (96, 111): np.float64(0.0035226831638104712), (97, 111): np.float64(0.005385289354971754), (111, 112): np.float64(-0.0033333845057285684), (111, 113): np.float64(-0.0005641481308935682), (111, 123): np.float64(-0.0067709468862066635), (111, 129): np.float64(-0.0029501966652045854), (111, 131): np.float64(-0.005246271514716721), (111, 133): np.float64(-0.0024935256493687314), (111, 137): np.float64(-0.002551454391412278), (111, 138): np.float64(-0.0040147736477131515), (111, 144): np.float64(-0.0024453243539575343), (111, 146): np.float64(0.21572539484366574), (82, 113): np.float64(0.001636658532897851), (87, 113): np.float64(0.0007623556181391893), (96, 113): np.float64(0.002725691447108753), (97, 113): np.float64(0.0024151230565950983), (113, 124): np.float64(-0.004088795919060853), (113, 128): np.float64(0.00160279135827323), (113, 134): np.float64(-0.0026899523306914176), (113, 138): np.float64(-0.007278271264881474), (113, 143): np.float64(-0.0031958787537718406), (113, 146): np.float64(0.2087920173259617), (113, 147): np.float64(-0.004297656719620583), (97, 129): np.float64(-0.000547248681258896), (112, 129): np.float64(-0.009928323698595372), (122, 129): np.float64(-0.01429722988512026), (125, 129): np.float64(-0.0011123561781141965), (129, 131): np.float64(0.0050752927295735105), (129, 133): np.float64(0.0024958284632381903), (129, 134): np.float64(0.0007878845649406488), (129, 138): np.float64(0.001084529707906438), (129, 142): np.float64(4.521450155914141e-05), (129, 143): np.float64(0.005347213775340642), (129, 144): np.float64(0.002687046698625119), (129, 147): np.float64(0.00013498390244075277), (82, 131): np.float64(0.0006259371920960781), (87, 131): np.float64(0.0009617218262692203), (97, 131): np.float64(0.0028039797899989047), (112, 131): np.float64(0.0019060811669638186), (115, 131): np.float64(-0.001441918513149609), (123, 131): np.float64(0.0024695599370066264), (131, 137): np.float64(-0.005841937322788542), (131, 138): np.float64(-0.0037494285518612667), (131, 142): np.float64(-0.003128549849563804), (131, 143): np.float64(-0.00202686285864526), (131, 146): np.float64(0.20441550766659158), (131, 148): np.float64(-0.003259585347298912), (78, 133): np.float64(-0.0027830996170994326), (87, 133): np.float64(-0.0017629335674754464), (109, 133): np.float64(-0.007551034626983476), (112, 133): np.float64(-0.0046426780862173864), (122, 133): np.float64(-0.0016443371693702635), (125, 133): np.float64(-0.0002664029322537724), (128, 133): np.float64(-0.007496544881601445), (133, 138): np.float64(0.00031786718217805557), (133, 143): np.float64(-3.4407319441511896e-05), (133, 147): np.float64(-0.0033998239420756447), (82, 134): np.float64(-0.0014511889556345222), (96, 134): np.float64(0.00020818026437125381), (112, 134): np.float64(-0.0015634051369110907), (115, 134): np.float64(-0.009073504305360679), (123, 134): np.float64(0.0008568102049033416), (124, 134): np.float64(0.004618687607154197), (134, 137): np.float64(0.0004050426476097231 6), (134, 138): np.float64(0.0002636949586368969), (134, 144): np.float64(0.003342115147629319), (134, 148): np.float64(-0.0012593566125550096), (78, 137): np.float64(-0.0035642735249529207), (79, 137): np.float64(-0.00011511281263517763), (96, 137): np.float64(0.00040344296063356787), (97, 137): np.float64(0.0019522625280693215), (109, 137): np.float64(-0.02079855743849731), (112, 137): np.float64(-0.001343907461239681), (115, 137): np.float64(-0.01706271371836016), (122, 137): np.float64(-0.0022712150481325917), (123, 137): np.float64(0.0006557726742357571), (124, 137): np.float64(0.0015520432807342186), (125, 137): np.float6

4(0.0002456516731169084), (137, 138): np.float64(-4.56576547523048e-05), (137, 142): np.float64(-0.0017180810523425261), (137, 143): np.float64(-0.000320480327315714), (137, 148): np.float64(-0.0010189497987703 95), (87, 144): np.float64(0.0002455425381006163), (96, 144): np.float64(0.0038997260536571573), (122, 144): np.float64(0.0001888624453121763), (123, 144): np.float64(0.0033159027687304533), (124, 144): np.float64(0.002586256774876124), (125, 144): np.float64(0.0012513075609014857), (128, 144): np.float64(-0.0009742527460479372), (138, 144): np.float64(0.0025011727899600135), (142, 144): np.float64(0.0019946319259333), (143, 144): np.float64(0.004238198989705058), (144, 146): np.float64(0.1750350072126977), (144, 147): np.float64(-0.005534585837336851), (78, 146): np.float64(0.11591760201674228), (82, 146): np.float64(0.1507378587937343), (87, 146): np.float64(0.12078618087077654), (109, 146): np.float64(0.30942687579824546), (112, 146): np.float64(0.16338636046342705), (115, 146): np.float64(0.24014910019905875), (122, 146): np.float64(0.17885869280607117), (128, 146): np.float64(0.10907378291165547), (78, 148): np.float64(-0.012807058624516223), (79, 148): np.float64(-0.0012377508797508344), (87, 148): np.float64(-0.0017330871849728145), (115, 148): np.float64(-0.0043973903988467015), (122, 148): np.float64(-0.004889946571881558), (138, 148): np.float64(-0.001381595345726061), (143, 148): np.float64(-0.0007102136054578426), (147, 148): np.float64(8.301928864625144e-05), (78, 79): np.float64(-0.005516433624641996), (78, 82): np.float64(-0.0020753445577047698), (78, 87): np.float64(-0.00010793802044888713), (78, 96): np.float64(-0.0033501411454533742), (78, 97): np.float64(-0.0019999190503161204), (78, 115): np.float64(0.00230006638316603), (78, 123): np.float64(-0.0015170265872667885), (78, 138): np.float64(-0.004900707131604414), (78, 142): np.float64(-0.002666797628974252), (79, 96): np.float64(-0.0005777642162316181), (87, 96): np.float64(-0.0015975296008995947), (96, 97): np.float64(-0.0004270214293860373), (96, 109): np.float64(0.0066725354198328), (96, 112): np.float64(0.0014018165801152394), (96, 115): np.float64(0.004537131623244799), (96, 125): np.float64(0.0008294662848563074), (96, 138): np.float64(0.0007459758724647331), (96, 147): np.float64(-0.0006325646892012785), (79, 112): np.float64(0.0011158501217948267), (97, 112): np.float64(0.006183569636155102), (112, 124): np.float64(-0.0027879060988979885), (112, 128): np.float64(0.0010494878881861197), (112, 138): np.float64(-0.003629770844504088), (112, 143): np.float64(-0.0015263957320959772), (87, 122): np.float64(0.0006389816949601381), (97, 122): np.float64(0.004075065476315961), (109, 122): np.float64(-0.003558370447728714), (122, 124): np.float64(-0.0028072781949892377), (122, 125): np.float64(-0.0019484068884968208), (122, 138): np.float64(-0.0019076285246316961), (122, 143): np.float64(-0.0013563631255425577), (82, 123): np.float64(-0.006915498282748708), (87, 123): np.float64(-0.004934836975431356), (97, 123): np.float64(0.0007363934895389872), (115, 123): np.float64(-0.0034356311720201054), (123, 138): np.float64(0.0005831433506117112), (123, 147): np.float64(-0.0008345252150521054), (79, 138): np.float64(-0.0003545232416920644), (87, 138): np.float64(-0.005385085123941179), (128, 138): np.float64(-0.004521324280319855), (138, 142): np.float64(-0.0019099726889617377), (138, 147): np.float64(-0.002663833050701986), (79, 147): np.float64(-0.0010948808793838673), (82, 147): np.float64(-0.0025280751554191184), (97, 147): np.float64(-0.0015211116875828577), (109, 147): np.float64(-0.0066121533001450355), (128, 147): np.float64(-0.0027468546137497606), (82, 97): np.float64(-0.002487392652128502), (82, 124): np.float64(-0.007372813949146497), (82, 128): np.float64(0.0013936337231320827), (82, 143): np.float64(-0.0012046517668557492), (79, 97): np.float64(-0.0009912483708651252), (97, 142): np.float64(-0.0003027521120046534), (97, 143): np.float64(0.0014710680312467598), (79, 125): np.float64(-0.0002470108384486056), (87, 125): np.float64(-0.0037447359059140655), (109, 125): np.float64(-0.005606526974957511), (115, 125): np.float64(-0.0029036404322149534), (125, 143): np.float64(-0.00029312689580379225), (79, 128): np.float64(0.0020871999790657712), (128, 142): np.float64(-0.0030360649596148197), (79, 142): np.float64(-0.0010522269344808105), (87, 143): np.float64(-0.0013181762182467225), (79, 115): np.float64(0.00859642824533112), (109, 115): np.float64(0.0006384365499099349), (115, 143): np.float64(-0.0031263828166917993), (79, 143): np.float64(-0.0004652885243723592 4), (109, 143): np.float64(-0.022145557494632415), (79, 124): np.float64(-0.0016941973903064965)}



Wszystkie wizualizacje zgadzają się z oczekiwanymi wynikami algrytmów

Podsumowanie

- Oba algorytmy zostały zaimplementowane w sposób poprawny na co wskazują testy oraz wizualizacje.
- Czas wykonania obu algorytmów jest odmienny, z racji dużej różnicy złożoności: O(E^3) dla metody opartej o "oczka" oraz O(V^3) dla metody potencjałów węzłowych
- Dzięki odpowiedniemu szukaniu cykli ilość równań dla metody opartej o "oczka" jest niewiele większa od potrzebnej. Przyspiesza to działanie algorytmu.
- Różnice w wynikach obu metod są rzędu 10^{-13}, co jest bardzo małą wartością, rzędu epsilona maszynowego. Ewentualne większe błędy obliczeniowe może wykazywać metoda potencjałów węzłowych przez liczenie odwrotności oporów, które umieszczamy w macierzy.