Zadanie domowe III

Witold Strzeboński

Oznaczenia

Przyjęto następujące oznaczenia:

- $A_{i,k}$ - znalezienie mnożnika dla wiersza i, do odejmowania go od k-tego wiersza,
  $m_{k,i} = M_{k,i} / M_{i,i}$

- $B_{i,j,k}$ - pomnożenie j-tego elementu wiersza i przez mnożnik - do odejmowania od k-tego wiersza,
  $n_{k,i,j} = M_{i,j} * m_{k,i}$

- $C_{i,j,k}$ - odjęcie j-tego elementu wiersza i od wiersza k,
  $M_{k,j} = M_{k,j} - n_{k,i,j}$

Opis funkcji wykorzystanych w programie

init(n)

Funkcja na podstawie zadanego rozmiaru macierzy wyznacza alfabet składający się z niepodzielnych zadań obliczeniowych: A, B, C i zbiór transakcji odpowiadający każdemu zadaniu oraz słowo, które jest ciągiem zadań w kolejności zgodnej z algorytmem Gaussa.

```
def init(n):
    A = []
    T = []

    for i in range(1, n):
        for k in range(i+1, n+1):
            A.append(f"A_{i},{k}")
            T.append((f"m_{k},{i}", (f"M_{k},{i}", f"M_{i},{i}")))
            for j in range(i, n+2):
                A.append(f"B_{i},{j},{k}")
                T.append((f"n_{k},{i},{j}", (f"M_{i},{j}", f"m_{k},{i}")))

                A.append(f"C_{i},{j},{k}")
                T.append((f"M_{k},{j}", (f"M_{k},{j}", f"n_{k},{i},{j}")))

    w = A.copy()

    return A, T, w
```

dependency_relationship(A, T, n)

Funkcja wyznacza relację zależności D i relację niezależności I.

Dla każdej pary transakcji sprawdzam, czy zmienna występująca po lewej stronie równania pierwszej transakcji występuje w równaniu drugiej transakcji. Jeżeli tak, to dodaję tę parę transakcji do zbioru zależności D. Zbiór niezależności I obliczam jako różnicę zbioru wszystkich par i zbioru zależności D.

```
def dependency_relationship(A, T, n):
    D = set()
    for i in range(n):
        x = T[i][0]
        for j in range(n):
            if i == j:
                D.add((A[i], A[i]))
            elif x == T[j][0] or x in T[j][1]:
                D.add((A[i], A[j]))
                D.add((A[j], A[i]))
    I = {(x, y) for x in A for y in A}
    I = sorted(I - D)
    D = sorted(D)

    return D, I
```

dependency_graph(D, w, k)

Funkcja tworzy graf (lista sąsiedztwa) zależności dla słowa w.

Litery w słowie numeruję kolejnymi liczbami naturalnymi i przedstawiam jako wierzchołki grafu. Następnie przechodzę w pętli po słowie i sprawdzam czy dana litera zależy od kolejnych. Jeżeli tak, to tworzę krawędź między wierzchołkami.

```
def dependency_graph(D, w, k):
    G = [[] for _ in range(k)]
    for i in range(k):
        for j in range(i + 1, k):
            if (w[i], w[j]) in D:
                G[i].append(j)

    return G
```

exists_another_path_dfs(G, u, v)

Funkcja sprawdza czy istnieje ścieżka między wierzchołkami u i v bez wykorzystania krawędzi (u, v).

Stosuję do tego algorytm dfs.

```
def exists_another_path_dfs(G, u, v):
    n = len(G)
    Visited = [False for _ in range(n)]
    Visited[u] = True
    for x in G[u]:
        if x != v:
            dfs_visit(G, Visited, x)

    return Visited[v]

def dfs_visit(G, Visited, u):
    Visited[u] = True
    for v in G[u]:
        if not Visited[v]:
            dfs_visit(G, Visited, v)
```

reduce_dependency_graph(G, k)

Funkcja redukuje graf zależności do postaci minimalnej.

Dla każdej krawędzi (u, v) w grafie sprawdzam, czy istnieje ścieżka między wierzchołkami u i v bez wykorzystania tej krawędzi. Jeżeli tak, to usuwam krawędź z grafu.

```
def reduce_dependency_graph(G, k):
    for u in range(k):
        for v in G[u][::]:
            if exists_another_path_dfs(G, u, v):
                G[u].remove(v)
```

foata_classes_bfs(G)

Funkcja dla każdego wierzchołka wyznacza jego numer klasy FNF.

Stosuję do tego algorytm bfs z pewną modyfikacją. Najpierw wyznaczam wierzchołki klasy 0. Są to te wierzchołki, które nie mają żadnej krawędzi wchodzącej. Następnie odpalam bfsa, z taką modyfikacją, że dopuszczam kilkukrotne odwiedzenie jednego wierzchołka. Dzięki temu w tablicy D dla każdego wierzchołka otrzymuję długość najdłuższej ścieżki z wierzchołka klasy 0, co jest równocześnie numerami klas FNF.

```
def foata_classes_bfs(G):
    n = len(G)
    D = [0 for _ in range(n)]
    Q = deque()
    for v in range(n):
        for u in range(v):
            if v in G[u]:
                break
        else:
            Q.appendleft(v)

    while Q:
        u = Q.pop()
        for v in G[u]:
            D[v] = D[u] + 1
            Q.appendleft(v)

    return D
```

foata_normal_form(G, w, k)

Funkcja wyznacza postać normalną Foaty FNF([w]) śladu [w] na podstawie numerów klas otrzymanych z powyższej funkcji.

```
def foata_normal_form(G, w, k):
    C = foata_classes_bfs(G)
    Classes = [[] for _ in range(max(C) + 1)]
    for i in range(k):
        Classes[C[i]].append(w[i])
    fnf = ""
    for c in Classes:
        fnf += '('
        c.sort()
        for f in c:
            fnf += f
        fnf += ')'

    return fnf
```

draw_graph(G, k, name)

Funkcja rysuje graf za pomocą biblioteki graphviz i zapisuje go w postaci pliku .gz.

```
def draw_graph(G, k, name):
    dot = Digraph(name)
    for i in range(k):
        dot.node(str(i), w[i])
    for i in range(k):
        for j in G[i]:
            dot.edge(str(i), str(j))
    dot.save()
```

execute_operation(operation, M, m, n)

Funkcja wykonuje operację zgodnie z otrzymanym zadaniem i zapisuje wynik w odpowiedniej macierzy.

```
def execute_operation(operation, M, m, n):
    indexes = operation[2:].split(',')
    match operation[0]:
        case 'A':
            i = int(indexes[0]) - 1
            k = int(indexes[1]) - 1
            m[k][i] = M[k][i] / M[i][i]
        case 'B':
            i = int(indexes[0]) - 1
            j = int(indexes[1]) - 1
            k = int(indexes[2]) - 1
            n[k][i][j] = M[i][j] * m[k][i]
        case 'C':
            i = int(indexes[0]) - 1
            j = int(indexes[1]) - 1
            k = int(indexes[2]) - 1
            M[k][j] = M[k][j] - n[k][i][j]
```

gaussian_elimination_concurrently(M, C)

Funkcja wykonuje współbieżną eliminację Gaussa zgodnie z postacią normalną Foaty. Do uruchamiania zadań w osobnych wątkach wykorzystywana jest pythonowa biblioteka threading.

```
def gaussian_elimination_concurrently(M, C):
    size = len(M)
    m = [[0 for i in range(size - 1)] for k in range(size)]
    n = [[[0 for j in range(size + 1)] for i in range(size - 1)] for k in
range(size)]

    for c in C:
        Threads = []
        for operation in c:
            Threads.append(Thread(target=execute_operation,
args=(operation, M, m, n)))
        for thread in Threads:
            thread.start()
        for thread in Threads:
            thread.join()
```

gaussian_elimination_simple(M)

Funkcja na zadanej macierzy wykonuje zwykłą eliminację Gaussa (niewspółbieżną). Jest ona wykorzystywana do sprawdzenia poprawności implementacji współbieżnego algorytmu.

```
def gaussian_elimination_simple(M):
    rows, cols = M.shape

    for i in range(min(rows, cols - 1)):
        for j in range(i + 1, rows):
            factor = M[j, i] / M[i, i]
            M[j, i:] -= factor * M[i, i:]
```

test(size, M, name)

Funkcja dla zadanej macierzy projektuje i realizuje współbieżny algorytm eliminacji Gaussa oraz sprawdza, czy otrzymany wynik jest prawidłowy.

```
def test(size, M, name):
    A, T, w = init(size)

    n = len(A)
    k = len(w)

    D, I = dependency_relationship(A, T, n)

    G = dependency_graph(D, w, k)
    reduce_dependency_graph(G, k)

    fnf, C = foata_normal_form(G, w, k)

    M_simple = array(M)
    gaussian_elimination_concurrently(M, C)
    gaussian_elimination_simple(M_simple)

    if M == M_simple.tolist():
        print("Uzyskane rozwiązanie jest poprawne")
    else:
        print("Uzyskane rozwiązanie jest niepoprawne")
```

Wyniki działania

W programie jest przeprowadzony test dla danych z pliku znajdującego się na uplu. Dodatkowo użytkownik może wykonać test dla losowej macierzy o rozmiarze wprowadzonym z klawiatury.

Test dla macierzy z pliku na uplu

INPUT

Macierz wejściowa:

2.00 1.00 3.00 6.00

4.00 3.00 8.00 15.00

6.00 5.00 16.00 27.00

OUTPUT

A = ['A_1,2', 'B_1,1,2', 'C_1,1,2', 'B_1,2,2', 'C_1,2,2', 'B_1,3,2', 'C_1,3,2', 'B_1,4,2', 'C_1,4,2', 'A_1,3', 'B_1,1,3', 'C_1,1,3', 'B_1,2,3', 'C_1,2,3', 'B_1,3,3', 'C_1,3,3', 'B_1,4,3', 'C_1,4,3', 'A_2,3', 'B_2,2,3', 'C_2,2,3', 'B_2,3,3', 'C_2,3,3', 'B_2,4,3', 'C_2,4,3']

T = [('m_2,1', ('M_2,1', 'M_1,1')), ('n_2,1,1', ('M_1,1', 'm_2,1')), ('M_2,1', ('M_2,1', 'n_2,1,1')), ('n_2,1,2', ('M_1,2', 'm_2,1')), ('M_2,2', ('M_2,2', 'n_2,1,2')), ('n_2,1,3', ('M_1,3', 'm_2,1')), ('M_2,3', ('M_2,3', 'n_2,1,3')), ('n_2,1,4', ('M_1,4', 'm_2,1')), ('M_2,4', ('M_2,4', 'n_2,1,4')), ('m_3,1', ('M_3,1', 'M_1,1')), ('n_3,1,1', ('M_1,1', 'm_3,1')), ('M_3,1', ('M_3,1', 'n_3,1,1')), ('n_3,1,2', ('M_1,2', 'm_3,1')), ('M_3,2', ('M_3,2', 'n_3,1,2')), ('n_3,1,3', ('M_1,3', 'm_3,1')), ('M_3,3', ('M_3,3', 'n_3,1,3')), ('n_3,1,4', ('M_1,4', 'm_3,1')), ('M_3,4', ('M_3,4', 'n_3,1,4')), ('m_3,2', ('M_3,2', 'M_2,2')), ('n_3,2,2', ('M_2,2', 'm_3,2')), ('M_3,2', ('M_3,2', 'n_3,2,2')), ('n_3,2,3', ('M_2,3', 'm_3,2')), ('M_3,3', ('M_3,3', 'n_3,2,3')), ('n_3,2,4', ('M_2,4', 'm_3,2')), ('M_3,4', ('M_3,4', 'n_3,2,4'))]

w = ['A_1,2', 'B_1,1,2', 'C_1,1,2', 'B_1,2,2', 'C_1,2,2', 'B_1,3,2', 'C_1,3,2', 'B_1,4,2', 'C_1,4,2', 'A_1,3', 'B_1,1,3', 'C_1,1,3', 'B_1,2,3', 'C_1,2,3', 'B_1,3,3', 'C_1,3,3', 'B_1,4,3', 'C_1,4,3', 'A_2,3', 'B_2,2,3', 'C_2,2,3', 'B_2,3,3', 'C_2,3,3', 'B_2,4,3', 'C_2,4,3']

D = [('A_1,2', 'A_1,2'), ('A_1,2', 'B_1,1,2'), ('A_1,2', 'B_1,2,2'), ('A_1,2', 'B_1,3,2'), ('A_1,2', 'B_1,4,2'), ('A_1,2', 'C_1,1,2'), ('A_1,3', 'A_1,3'), ('A_1,3', 'B_1,1,3'), ('A_1,3', 'B_1,2,3'), ('A_1,3', 'B_1,3,3'), ('A_1,3', 'B_1,4,3'), ('A_1,3', 'C_1,1,3'), ('A_2,3', 'A_2,3'), ('A_2,3', 'B_2,2,3'), ('A_2,3', 'B_2,3,3'), ('A_2,3', 'B_2,4,3'), ('A_2,3', 'C_1,2,2'), ('A_2,3', 'C_1,2,3'), ('A_2,3', 'C_2,2,3'), ('B_1,1,2', 'A_1,2'), ('B_1,1,2', 'B_1,1,2'), ('B_1,1,2', 'C_1,1,2'), ('B_1,1,3', 'A_1,3'), ('B_1,1,3', 'B_1,1,3'), ('B_1,1,3', 'C_1,1,3'), ('B_1,2,2', 'A_1,2'), ('B_1,2,2', 'B_1,2,2'), ('B_1,2,2', 'C_1,2,2'), ('B_1,2,3', 'A_1,3'), ('B_1,2,3', 'B_1,2,3'), ('B_1,2,3', 'C_1,2,3'), ('B_1,3,2', 'A_1,2'), ('B_1,3,2', 'B_1,3,2'), ('B_1,3,2', 'C_1,3,2'), ('B_1,3,3', 'A_1,3'), ('B_1,3,3', 'B_1,3,3'), ('B_1,3,3', 'C_1,3,3'), ('B_1,4,2', 'A_1,2'), ('B_1,4,2', 'B_1,4,2'), ('B_1,4,2', 'C_1,4,2'), ('B_1,4,3', 'A_1,3'), ('B_1,4,3', 'B_1,4,3'), ('B_1,4,3', 'C_1,4,3'), ('B_2,2,3', 'A_2,3'), ('B_2,2,3', 'B_2,2,3'), ('B_2,2,3', 'C_1,2,2'), ('B_2,2,3', 'C_2,2,3'), ('B_2,3,3', 'A_2,3'), ('B_2,3,3', 'B_2,3,3'), ('B_2,3,3',

'C_1,3,2'), ('B_2,3,3', 'C_2,3,3'), ('B_2,4,3', 'A_2,3'), ('B_2,4,3', 'B_2,4,3'), ('B_2,4,3', 'C_1,4,2'), ('B_2,4,3', 'C_2,4,3'), ('C_1,1,2', 'A_1,2'), ('C_1,1,2', 'B_1,1,2'), ('C_1,1,2', 'C_1,1,2'), ('C_1,1,3', 'A_1,3'), ('C_1,1,3', 'B_1,1,3'), ('C_1,1,3', 'C_1,1,3'), ('C_1,2,2', 'A_2,3'), ('C_1,2,2', 'B_1,2,2'), ('C_1,2,2', 'B_2,2,3'), ('C_1,2,2', 'C_1,2,2'), ('C_1,2,3', 'A_2,3'), ('C_1,2,3', 'B_1,2,3'), ('C_1,2,3', 'C_1,2,3'), ('C_1,2,3', 'C_2,2,3'), ('C_1,3,2', 'B_1,3,2'), ('C_1,3,2', 'B_2,3,3'), ('C_1,3,2', 'C_1,3,2'), ('C_1,3,3', 'B_1,3,3'), ('C_1,3,3', 'C_1,3,3'), ('C_1,3,3', 'C_2,3,3'), ('C_1,4,2', 'B_1,4,2'), ('C_1,4,2', 'B_2,4,3'), ('C_1,4,2', 'C_1,4,2'), ('C_1,4,3', 'B_1,4,3'), ('C_1,4,3', 'C_1,4,3'), ('C_1,4,3', 'C_2,4,3'), ('C_2,2,3', 'A_2,3'), ('C_2,2,3', 'B_2,2,3'), ('C_2,2,3', 'C_1,2,3'), ('C_2,2,3', 'C_2,2,3'), ('C_2,3,3', 'B_2,3,3'), ('C_2,3,3', 'C_1,3,3'), ('C_2,3,3', 'C_2,3,3'), ('C_2,4,3', 'B_2,4,3'), ('C_2,4,3', 'C_1,4,3'), ('C_2,4,3', 'C_2,4,3')]

FNF([w]) = (A_1,2 A_1,3)(B_1,1,2 B_1,2,2 B_1,3,2 B_1,4,2 B_1,1,3 B_1,2,3 B_1,3,3 B_1,4,3)(C_1,1,2 C_1,2,2 C_1,3,2 C_1,4,2 C_1,1,3 C_1,2,3 C_1,3,3 C_1,4,3)(A_2,3)(B_2,2,3 B_2,3,3 B_2,4,3)(C_2,2,3 C_2,3,3 C_2,4,3)

Macierz po współbieżnej eliminacji Gaussa:

2.00 1.00 3.00 6.00

0.00 1.00 2.00 3.00

0.00 0.00 3.00 3.00

Macierz po zwykłej eliminacji Gaussa:

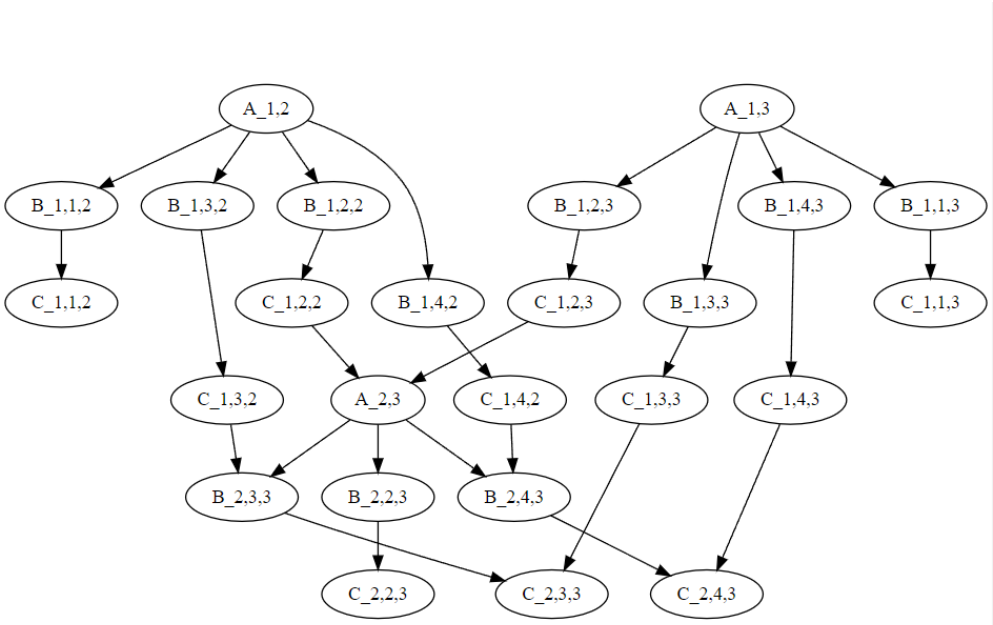
2.00 1.00 3.00 6.00

0.00 1.00 2.00 3.00

0.00 0.00 3.00 3.00

Uzyskane rozwiązanie jest poprawne

Zredukowany graf zależności



Test dla losowej macierzy o rozmiarze 4

INPUT

Macierz wejściowa:

55.48 92.80 29.24 82.81 39.08

61.85 86.48 34.45 41.10 50.48

12.88 88.33 11.06 80.54 10.24

59.83 67.21 92.61 0.06   4.57

OUTPUT

A = ['A_1,2', 'B_1,1,2', 'C_1,1,2', 'B_1,2,2', 'C_1,2,2', 'B_1,3,2', 'C_1,3,2', 'B_1,4,2', 'C_1,4,2', 'B_1,5,2', 'C_1,5,2', 'A_1,3', 'B_1,1,3', 'C_1,1,3', 'B_1,2,3', 'C_1,2,3', 'B_1,3,3', 'C_1,3,3', 'B_1,4,3', 'C_1,4,3', 'B_1,5,3', 'C_1,5,3', 'A_1,4', 'B_1,1,4', 'C_1,1,4', 'B_1,2,4', 'C_1,2,4', 'B_1,3,4', 'C_1,3,4', 'B_1,4,4', 'C_1,4,4', 'B_1,5,4', 'C_1,5,4', 'A_2,3', 'B_2,2,3', 'C_2,2,3', 'B_2,3,3', 'C_2,3,3', 'B_2,4,3', 'C_2,4,3', 'B_2,5,3', 'C_2,5,3', 'A_2,4', 'B_2,2,4', 'C_2,2,4', 'B_2,3,4', 'C_2,3,4', 'B_2,4,4', 'C_2,4,4', 'B_2,5,4', 'C_2,5,4', 'A_3,4', 'B_3,3,4', 'C_3,3,4', 'B_3,4,4', 'C_3,4,4', 'B_3,5,4', 'C_3,5,4']

T = [('m_2,1', ('M_2,1', 'M_1,1')), ('n_2,1,1', ('M_1,1', 'm_2,1')), ('M_2,1', ('M_2,1', 'n_2,1,1')), ('n_2,1,2', ('M_1,2', 'm_2,1')), ('M_2,2', ('M_2,2', 'n_2,1,2')), ('n_2,1,3', ('M_1,3', 'm_2,1')), ('M_2,3', ('M_2,3', 'n_2,1,3')), ('n_2,1,4', ('M_1,4', 'm_2,1')), ('M_2,4', ('M_2,4', 'n_2,1,4')), ('n_2,1,5', ('M_1,5', 'm_2,1')), ('M_2,5', ('M_2,5', 'n_2,1,5')), ('m_3,1', ('M_3,1', 'M_1,1')), ('n_3,1,1', ('M_1,1', 'm_3,1')), ('M_3,1', ('M_3,1', 'n_3,1,1')), ('n_3,1,2', ('M_1,2', 'm_3,1')), ('M_3,2', ('M_3,2', 'n_3,1,2')), ('n_3,1,3', ('M_1,3', 'm_3,1')), ('M_3,3', ('M_3,3', 'n_3,1,3')), ('n_3,1,4', ('M_1,4', 'm_3,1')), ('M_3,4', ('M_3,4', 'n_3,1,4')), ('n_3,1,5', ('M_1,5', 'm_3,1')), ('M_3,5', ('M_3,5', 'n_3,1,5')), ('m_4,1', ('M_4,1', 'M_1,1')), ('n_4,1,1', ('M_1,1', 'm_4,1')), ('M_4,1', ('M_4,1', 'n_4,1,1')), ('n_4,1,2', ('M_1,2', 'm_4,1')), ('M_4,2', ('M_4,2', 'n_4,1,2')), ('n_4,1,3', ('M_1,3', 'm_4,1')), ('M_4,3', ('M_4,3', 'n_4,1,3')), ('n_4,1,4', ('M_1,4', 'm_4,1')), ('M_4,4', ('M_4,4', 'n_4,1,4')), ('n_4,1,5', ('M_1,5', 'm_4,1')), ('M_4,5', ('M_4,5', 'n_4,1,5')), ('m_3,2', ('M_3,2', 'M_2,2')), ('n_3,2,2', ('M_2,2', 'm_3,2')), ('M_3,2', ('M_3,2', 'n_3,2,2')), ('n_3,2,3', ('M_2,3', 'm_3,2')), ('M_3,3', ('M_3,3', 'n_3,2,3')), ('n_3,2,4', ('M_2,4', 'm_3,2')), ('M_3,4', ('M_3,4', 'n_3,2,4')), ('n_3,2,5', ('M_2,5', 'm_3,2')), ('M_3,5', ('M_3,5', 'n_3,2,5')), ('m_4,2', ('M_4,2', 'M_2,2')), ('n_4,2,2', ('M_2,2', 'm_4,2')), ('M_4,2', ('M_4,2', 'n_4,2,2')), ('n_4,2,3', ('M_2,3', 'm_4,2')), ('M_4,3', ('M_4,3', 'n_4,2,3')), ('n_4,2,4', ('M_2,4', 'm_4,2')), ('M_4,4', ('M_4,4', 'n_4,2,4')), ('n_4,2,5', ('M_2,5', 'm_4,2')), ('M_4,5', ('M_4,5', 'n_4,2,5')), ('m_4,3', ('M_4,3', 'M_3,3')), ('n_4,3,3', ('M_3,3', 'm_4,3')), ('M_4,3', ('M_4,3', 'n_4,3,3')), ('n_4,3,4', ('M_3,4', 'm_4,3')), ('M_4,4', ('M_4,4', 'n_4,3,4')), ('n_4,3,5', ('M_3,5', 'm_4,3')), ('M_4,5', ('M_4,5', 'n_4,3,5'))]

w = ['A_1,2', 'B_1,1,2', 'C_1,1,2', 'B_1,2,2', 'C_1,2,2', 'B_1,3,2', 'C_1,3,2', 'B_1,4,2', 'C_1,4,2', 'B_1,5,2', 'C_1,5,2', 'A_1,3', 'B_1,1,3', 'C_1,1,3', 'B_1,2,3', 'C_1,2,3', 'B_1,3,3', 'C_1,3,3', 'B_1,4,3', 'C_1,4,3', 'B_1,5,3', 'C_1,5,3', 'A_1,4', 'B_1,1,4', 'C_1,1,4', 'B_1,2,4', 'C_1,2,4',

'B_1,3,4', 'C_1,3,4', 'B_1,4,4', 'C_1,4,4', 'B_1,5,4', 'C_1,5,4', 'A_2,3', 'B_2,2,3', 'C_2,2,3', 'B_2,3,3', 'C_2,3,3', 'B_2,4,3', 'C_2,4,3', 'B_2,5,3', 'C_2,5,3', 'A_2,4', 'B_2,2,4', 'C_2,2,4', 'B_2,3,4', 'C_2,3,4', 'B_2,4,4', 'C_2,4,4', 'B_2,5,4', 'C_2,5,4', 'A_3,4', 'B_3,3,4', 'C_3,3,4', 'B_3,4,4', 'C_3,4,4', 'B_3,5,4', 'C_3,5,4']

D = [('A_1,2', 'A_1,2'), ('A_1,2', 'B_1,1,2'), ('A_1,2', 'B_1,2,2'), ('A_1,2', 'B_1,3,2'), ('A_1,2', 'B_1,4,2'), ('A_1,2', 'B_1,5,2'), ('A_1,2', 'C_1,1,2'), ('A_1,3', 'A_1,3'), ('A_1,3', 'B_1,1,3'), ('A_1,3', 'B_1,2,3'), ('A_1,3', 'B_1,3,3'), ('A_1,3', 'B_1,4,3'), ('A_1,3', 'B_1,5,3'), ('A_1,3', 'C_1,1,3'), ('A_1,4', 'A_1,4'), ('A_1,4', 'B_1,1,4'), ('A_1,4', 'B_1,2,4'), ('A_1,4', 'B_1,3,4'), ('A_1,4', 'B_1,4,4'), ('A_1,4', 'B_1,5,4'), ('A_1,4', 'C_1,1,4'), ('A_2,3', 'A_2,3'), ('A_2,3', 'B_2,2,3'), ('A_2,3', 'B_2,3,3'), ('A_2,3', 'B_2,4,3'), ('A_2,3', 'B_2,5,3'), ('A_2,3', 'C_1,2,2'), ('A_2,3', 'C_1,2,3'), ('A_2,3', 'C_2,2,3'), ('A_2,4', 'A_2,4'), ('A_2,4', 'B_2,2,4'), ('A_2,4', 'B_2,3,4'), ('A_2,4', 'B_2,4,4'), ('A_2,4', 'B_2,5,4'), ('A_2,4', 'C_1,2,2'), ('A_2,4', 'C_1,2,4'), ('A_2,4', 'C_2,2,4'), ('A_3,4', 'A_3,4'), ('A_3,4', 'B_3,3,4'), ('A_3,4', 'B_3,4,4'), ('A_3,4', 'B_3,5,4'), ('A_3,4', 'C_1,3,3'), ('A_3,4', 'C_1,3,4'), ('A_3,4', 'C_2,3,3'), ('A_3,4', 'C_2,3,4'), ('A_3,4', 'C_3,3,4'), ('B_1,1,2', 'A_1,2'), ('B_1,1,2', 'B_1,1,2'), ('B_1,1,2', 'C_1,1,2'), ('B_1,1,3', 'A_1,3'), ('B_1,1,3', 'B_1,1,3'), ('B_1,1,3', 'C_1,1,3'), ('B_1,1,4', 'A_1,4'), ('B_1,1,4', 'B_1,1,4'), ('B_1,1,4', 'C_1,1,4'), ('B_1,2,2', 'A_1,2'), ('B_1,2,2', 'B_1,2,2'), ('B_1,2,2', 'C_1,2,2'), ('B_1,2,3', 'A_1,3'), ('B_1,2,3', 'B_1,2,3'), ('B_1,2,3', 'C_1,2,3'), ('B_1,2,4', 'A_1,4'), ('B_1,2,4', 'B_1,2,4'), ('B_1,2,4', 'C_1,2,4'), ('B_1,3,2', 'A_1,2'), ('B_1,3,2', 'B_1,3,2'), ('B_1,3,2', 'C_1,3,2'), ('B_1,3,3', 'A_1,3'), ('B_1,3,3', 'B_1,3,3'), ('B_1,3,3', 'C_1,3,3'), ('B_1,3,4', 'A_1,4'), ('B_1,3,4', 'B_1,3,4'), ('B_1,3,4', 'C_1,3,4'), ('B_1,4,2', 'A_1,2'), ('B_1,4,2', 'B_1,4,2'), ('B_1,4,2', 'C_1,4,2'), ('B_1,4,3', 'A_1,3'), ('B_1,4,3', 'B_1,4,3'), ('B_1,4,3', 'C_1,4,3'), ('B_1,4,4', 'A_1,4'), ('B_1,4,4', 'B_1,4,4'), ('B_1,4,4', 'C_1,4,4'), ('B_1,5,2', 'A_1,2'), ('B_1,5,2', 'B_1,5,2'), ('B_1,5,2', 'C_1,5,2'), ('B_1,5,3', 'A_1,3'), ('B_1,5,3', 'B_1,5,3'), ('B_1,5,3', 'C_1,5,3'), ('B_1,5,4', 'A_1,4'), ('B_1,5,4', 'B_1,5,4'), ('B_1,5,4', 'C_1,5,4'), ('B_2,2,3', 'A_2,3'), ('B_2,2,3', 'B_2,2,3'), ('B_2,2,3', 'C_1,2,2'), ('B_2,2,3', 'C_2,2,3'), ('B_2,2,4', 'A_2,4'), ('B_2,2,4', 'B_2,2,4'), ('B_2,2,4', 'C_1,2,2'), ('B_2,2,4', 'C_2,2,4'), ('B_2,3,3', 'A_2,3'), ('B_2,3,3', 'B_2,3,3'), ('B_2,3,3', 'C_1,3,2'), ('B_2,3,3', 'C_2,3,3'), ('B_2,3,4', 'A_2,4'), ('B_2,3,4', 'B_2,3,4'), ('B_2,3,4', 'C_1,3,2'), ('B_2,3,4', 'C_2,3,4'), ('B_2,4,3', 'A_2,3'), ('B_2,4,3', 'B_2,4,3'), ('B_2,4,3', 'C_1,4,2'), ('B_2,4,3', 'C_2,4,3'), ('B_2,4,4', 'A_2,4'), ('B_2,4,4', 'B_2,4,4'), ('B_2,4,4', 'C_1,4,2'), ('B_2,4,4', 'C_2,4,4'), ('B_2,5,3', 'A_2,3'), ('B_2,5,3', 'B_2,5,3'), ('B_2,5,3', 'C_1,5,2'), ('B_2,5,3', 'C_2,5,3'), ('B_2,5,4', 'A_2,4'), ('B_2,5,4', 'B_2,5,4'), ('B_2,5,4', 'C_1,5,2'), ('B_2,5,4', 'C_2,5,4'), ('B_3,3,4', 'A_3,4'), ('B_3,3,4', 'B_3,3,4'), ('B_3,3,4', 'C_1,3,3'), ('B_3,3,4', 'C_2,3,3'), ('B_3,3,4', 'C_3,3,4'), ('B_3,4,4', 'A_3,4'), ('B_3,4,4', 'B_3,4,4'), ('B_3,4,4', 'C_1,4,3'), ('B_3,4,4', 'C_2,4,3'), ('B_3,4,4', 'C_3,4,4'), ('B_3,5,4', 'A_3,4'), ('B_3,5,4', 'B_3,5,4'), ('B_3,5,4', 'C_1,5,3'), ('B_3,5,4', 'C_2,5,3'), ('B_3,5,4', 'C_3,5,4'), ('C_1,1,2', 'A_1,2'), ('C_1,1,2', 'B_1,1,2'), ('C_1,1,2', 'C_1,1,2'), ('C_1,1,3', 'A_1,3'), ('C_1,1,3', 'B_1,1,3'), ('C_1,1,3', 'C_1,1,3'), ('C_1,1,4', 'A_1,4'), ('C_1,1,4', 'B_1,1,4'), ('C_1,1,4', 'C_1,1,4'), ('C_1,2,2', 'A_2,3'), ('C_1,2,2', 'A_2,4'), ('C_1,2,2', 'B_1,2,2'), ('C_1,2,2', 'B_2,2,3'), ('C_1,2,2', 'B_2,2,4'), ('C_1,2,2', 'C_1,2,2'), ('C_1,2,3', 'A_2,3'), ('C_1,2,3', 'B_1,2,3'), ('C_1,2,3', 'C_1,2,3'), ('C_1,2,3', 'C_2,2,3'), ('C_1,2,4', 'A_2,4'), ('C_1,2,4', 'B_1,2,4'), ('C_1,2,4', 'C_1,2,4'), ('C_1,2,4', 'C_2,2,4'), ('C_1,3,2', 'B_1,3,2'), ('C_1,3,2', 'B_2,3,3'), ('C_1,3,2', 'B_2,3,4'), ('C_1,3,2', 'C_1,3,2'), ('C_1,3,3', 'A_3,4'), ('C_1,3,3', 'B_1,3,3'), ('C_1,3,3', 'B_3,3,4'), ('C_1,3,3', 'C_1,3,3'), ('C_1,3,3', 'C_2,3,3'), ('C_1,3,4', 'A_3,4'), ('C_1,3,4', 'B_1,3,4'), ('C_1,3,4', 'C_1,3,4'), ('C_1,3,4', 'C_2,3,4'), ('C_1,3,4', 'C_3,3,4'), ('C_1,4,2', 'B_1,4,2'), ('C_1,4,2', 'B_2,4,3'), ('C_1,4,2', 'B_2,4,4'), ('C_1,4,2', 'C_1,4,2'),

('C_1,4,3', 'B_1,4,3'), ('C_1,4,3', 'B_3,4,4'), ('C_1,4,3', 'C_1,4,3'), ('C_1,4,3', 'C_2,4,3'), ('C_1,4,4', 'B_1,4,4'), ('C_1,4,4', 'C_1,4,4'), ('C_1,4,4', 'C_2,4,4'), ('C_1,4,4', 'C_3,4,4'), ('C_1,5,2', 'B_1,5,2'), ('C_1,5,2', 'B_2,5,3'), ('C_1,5,2', 'B_2,5,4'), ('C_1,5,2', 'C_1,5,2'), ('C_1,5,3', 'B_1,5,3'), ('C_1,5,3', 'B_3,5,4'), ('C_1,5,3', 'C_1,5,3'), ('C_1,5,3', 'C_2,5,3'), ('C_1,5,4', 'B_1,5,4'), ('C_1,5,4', 'C_1,5,4'), ('C_1,5,4', 'C_2,5,4'), ('C_1,5,4', 'C_3,5,4'), ('C_2,2,3', 'A_2,3'), ('C_2,2,3', 'B_2,2,3'), ('C_2,2,3', 'C_1,2,3'), ('C_2,2,3', 'C_2,2,3'), ('C_2,2,4', 'A_2,4'), ('C_2,2,4', 'B_2,2,4'), ('C_2,2,4', 'C_1,2,4'), ('C_2,2,4', 'C_2,2,4'), ('C_2,3,3', 'A_3,4'), ('C_2,3,3', 'B_2,3,3'), ('C_2,3,3', 'B_3,3,4'), ('C_2,3,3', 'C_1,3,3'), ('C_2,3,3', 'C_2,3,3'), ('C_2,3,4', 'A_3,4'), ('C_2,3,4', 'B_2,3,4'), ('C_2,3,4', 'C_1,3,4'), ('C_2,3,4', 'C_2,3,4'), ('C_2,3,4', 'C_3,3,4'), ('C_2,4,3', 'B_2,4,3'), ('C_2,4,3', 'B_3,4,4'), ('C_2,4,3', 'C_1,4,3'), ('C_2,4,3', 'C_2,4,3'), ('C_2,4,4', 'B_2,4,4'), ('C_2,4,4', 'C_1,4,4'), ('C_2,4,4', 'C_2,4,4'), ('C_2,4,4', 'C_3,4,4'), ('C_2,5,3', 'B_2,5,3'), ('C_2,5,3', 'B_3,5,4'), ('C_2,5,3', 'C_1,5,3'), ('C_2,5,3', 'C_2,5,3'), ('C_2,5,4', 'B_2,5,4'), ('C_2,5,4', 'C_1,5,4'), ('C_2,5,4', 'C_2,5,4'), ('C_2,5,4', 'C_3,5,4'), ('C_3,3,4', 'A_3,4'), ('C_3,3,4', 'B_3,3,4'), ('C_3,3,4', 'C_1,3,4'), ('C_3,3,4', 'C_2,3,4'), ('C_3,3,4', 'C_3,3,4'), ('C_3,4,4', 'B_3,4,4'), ('C_3,4,4', 'C_1,4,4'), ('C_3,4,4', 'C_2,4,4'), ('C_3,4,4', 'C_3,4,4'), ('C_3,5,4', 'B_3,5,4'), ('C_3,5,4', 'C_1,5,4'), ('C_3,5,4', 'C_2,5,4'), ('C_3,5,4', 'C_3,5,4')]

FNF([w]) = (A_1,2 A_1,3 A_1,4)(B_1,1,2 B_1,2,2 B_1,3,2 B_1,4,2 B_1,5,2 B_1,1,3 B_1,2,3 B_1,3,3 B_1,4,3 B_1,5,3 B_1,1,4 B_1,2,4 B_1,3,4 B_1,4,4 B_1,5,4)(C_1,1,2 C_1,2,2 C_1,3,2 C_1,4,2 C_1,5,2 C_1,1,3 C_1,2,3 C_1,3,3 C_1,4,3 C_1,5,3 C_1,1,4 C_1,2,4 C_1,3,4 C_1,4,4 C_1,5,4)(A_2,3 A_2,4)(B_2,2,3 B_2,3,3 B_2,4,3 B_2,5,3 B_2,2,4 B_2,3,4 B_2,4,4 B_2,5,4)(C_2,2,3 C_2,3,3 C_2,4,3 C_2,5,3 C_2,2,4 C_2,3,4 C_2,4,4 C_2,5,4)(A_3,4)(B_3,3,4 B_3,4,4 B_3,5,4)(C_3,3,4 C_3,4,4 C_3,5,4)

Macierz po współbieżnej eliminacji Gaussa:

55.48	92.80	29.24	82.81	39.08
0.00	-16.98	1.86	-51.23	6.90
0.00	0.00	11.57	-140.21	28.32
0.00	0.00	0.00	706.47	-191.65

Macierz po zwykłej eliminacji Gaussa:

55.48	92.80	29.24	82.81	39.08
0.00	-16.98	1.86	-51.23	6.90
0.00	0.00	11.57	-140.21	28.32
0.00	0.00	0.00	706.47	-191.65

Uzyskane rozwiązanie jest poprawne

Zredukowany graf zależności