AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Teoria Współbieżności
ćwiczenie 7
"Zastosowanie teorii śladów do szeregowania wątków współbieżnej eliminacji Gaussa"

Przemysław Popowski
Informatyka WI, III rok
Gr 7, wtorek 18:30

Spis treści

1 Cel ćwiczenia

Celem ćwiczenia jest zapoznanie studentów z zastosowaniem teorii śladów do szeregowania wątków z zastosowaniem w klasycznych algorytmach algebry liniowej. Jako przykład omawiany jest algorytm współbieżnej eliminacji Gaussa. Ćwiczenie zakłada znajomość podstaw programowania współbieżnego oraz podstaw z algebry liniowej (poznanych na pierwszym semestrze studiów).

2 Wprowadzenie do ćwiczenia

Rozważmy problem rozwiązywania układów równań liniowych metodą Gaussa. Problem można przedstawić jako $M \cdot x = y$, gdzie M jest macierzą kwadratową o rozmiarze N x N, natomiast x oraz y są wektorami o długości N. Elementy macierzy M będziemy indeksować jako $M_{i,j}$. Elementy wektora y będziemy indeksować jako y_i.

$$\begin{bmatrix} M_{1,1} & M_{1,2} & ... & M_{1,N} \\ M_{2,1} & ... & ... & : \\ : & ... & ... & : \\ M_{N,1} & ... & ... & M_{N,N} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ : \\ x_N \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ : \\ y_N \end{bmatrix}$$

Mamy następujące niepodzielne zadania obliczeniowe za pomocą których można rozwiązać w.w. układ

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

$$m_{k,i} = \frac{M_{k,i}}{M_{i,i}}$$

$B_{i,j,k}$ - pomnożenie j-tego elementu wiersza i przez mnożnik - do odejmowania od k-tego wiersza,

$$n_{k,i} = M_{i,j} \cdot m_{k,i}$$

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

$$M_{k,j} = M_{k,j} - n_{k,i}.$$

Dla uproszczenia zapisu będziemy stosować poniższą notację.

$$\left[\begin{array}{cccc|c} M_{1,1} & M_{1,2} & ... & M_{1,N} & y_1 \\ M_{2,1} & ... & ... & : & y_2 \\ : & ... & ... & : & : \\ M_{N,1} & ... & ... & M_{N,N} & y_N \end{array}\right]$$

3 Algorytm Gaussa

Eliminacja Gaussa polega na sukcesywnym zerowaniu odpowiednich elementów macierzy. Rozważmy macierz o rozmiarze N, gdzie wektor wyrazów wolnych traktowany jest jako dodatkowa kolumna o indeksie $N + 1$

$$\left[\begin{array}{cccc|c} M_{1,1} & M_{1,2} & ... & M_{1,N} & M_{1,N+1} \\ M_{2,1} & ... & ... & : & M_{2,N+1} \\ : & ... & ... & : & : \\ M_{N,1} & ... & ... & M_{N,N} & M_{N,N+1} \end{array}\right]$$

Algorytm przechodzi przez kolejne kolumny macierzy, aby wyzerować elementy znajdujące się pod przekątną. Dla każdej kolumny j, zerowane są elementy $M_{i,j}$, gdzie $i > j$

Poniżej zapisałem pseudo kod realizujący eliminację Gaussa, który ułatwi nam później zrozumienie problemu:

```
1  for i in 1 to n:                          # Przechodzimy po kolumnach
2      for k in i+1 to n:                    # Zerujemy elementy pod przekątną
3          m(k, i) = M[k, i] / M[i, i]       # Wyznaczenie mnożnika -> A(i,k)
4          for j in i to n+1:                # Aktualizacja wiersza k
5              n(k, i) = M[i, j] * m(k, i)   # Obliczenie wyniku -> B(i,j,k)
6              M[k, j] -= n(k, i)            # Odjęcie wyniku od M[k, j] -> C(i,j,k)
```

4 Przykład

Mamy zadany następujący układ równań

$$\begin{bmatrix} 2 & 1 & 3 \\ 4 & 3 & 8 \\ 6 & 5 & 16 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 15 \\ 27 \end{bmatrix}$$

Dla uproszczenia zapisu będziemy stosować poniższą notację.

$$\left[\begin{array}{ccc|c} 2 & 1 & 3 & 6 \\ 4 & 3 & 8 & 15 \\ 6 & 5 & 16 & 27 \end{array}\right]$$

Używamy pierwszego wiersza do „wyprodukowania"zer w pierwszej kolumnie.

Drugi wiersz = drugi wiersz - 2 * pierwszy wiersz

$$\left[\begin{array}{ccc|c} 2 & 1 & 3 & 6 \\ 0 & 1 & 2 & 3 \\ 6 & 5 & 16 & 27 \end{array}\right]$$

Trzeci wiersz = trzeci wiersz - 3 * pierwszy wiersz

$$A_{1,3}, B_{1,1,3}, C_{1,1,3}, B_{1,2,3}, C_{1,2,3}, B_{1,3,3}, C_{1,3,3}, B_{1,4,3}, C_{1,4,3}$$

$$\left[\begin{array}{ccc|c} 2 & 1 & 3 & 6 \\ 0 & 1 & 2 & 3 \\ 0 & 2 & 7 & 9 \end{array}\right]$$

Używamy drugiego wiersza do „wyprodukowania” zer w drugiej kolumnie.

Trzeci wiersz = trzeci wiersz - 2 * drugi wiersz

$$A_{2,3}, B_{2,1,3}, C_{2,1,3}, B_{2,2,3}, C_{2,2,3}, B_{2,3,3}, C_{2,3,3}, B_{2,4,3}, C_{2,4,3}$$

$$\left[\begin{array}{ccc|c} 2 & 1 & 3 & 6 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 3 & 3 \end{array}\right]$$

$$\begin{aligned} \Sigma = \{ & A_{1,2}, B_{1,1,2}, C_{1,1,2}, B_{1,2,2}, C_{1,2,2}, B_{1,3,2}, C_{1,3,2}, B_{1,4,2}, C_{1,4,2}, \\ & A_{1,3}, B_{1,1,3}, C_{1,1,3}, B_{1,2,3}, C_{1,2,3}, B_{1,3,3}, C_{1,3,3}, B_{1,4,3}, C_{1,4,3}, \\ & A_{2,3}, B_{2,2,3}, C_{2,2,3}, B_{2,3,3}, C_{2,3,3}, B_{2,4,3}, C_{2,4,3}\} \end{aligned}$$

$$\begin{aligned} D = \mathrm{sym}\Big\{\Big\{ & \big(A_{1,2}, B_{1,1,2}\big), \big(A_{1,2}, B_{1,2,2}\big), \big(A_{1,2}, B_{1,3,2}\big), \big(A_{1,2}, B_{1,4,2}\big), \\ & \big(B_{1,1,2}, C_{1,1,2}\big), \big(B_{1,2,2}, C_{1,2,2}\big), \big(B_{1,3,2}, C_{1,3,2}\big), \big(B_{1,4,2}, C_{1,4,2}\big), \\ & \big(A_{1,3}, B_{1,1,3}\big), \big(A_{1,3}, B_{1,2,3}\big), \big(A_{1,3}, B_{1,3,3}\big), \big(A_{1,3}, B_{1,4,3}\big), \\ & \big(B_{1,1,3}, C_{1,1,3}\big), \big(B_{1,2,3}, C_{1,2,3}\big), \big(B_{1,3,3}, C_{1,3,3}\big), \big(B_{1,4,3}, C_{1,4,3}\big), \\ & \big(A_{2,3}, B_{2,2,3}\big), \big(A_{2,3}, B_{2,3,3}\big), \big(A_{2,3}, B_{2,4,3}\big), \\ & \big(B_{2,2,3}, C_{2,2,3}\big), \big(B_{2,3,3}, C_{2,3,3}\big), \big(B_{2,4,3}, C_{2,4,3}\big), \\ & \big(C_{1,2,2}, A_{2,3}\big), \big(C_{1,2,2}, B_{2,2,3}\big), \big(C_{1,2,3}, B_{2,3,3}\big), \big(C_{1,3,2}, B_{2,3,3}\big), \\ & \big(C_{1,3,3}, C_{2,3,3}\big), \big(C_{1,4,3}, C_{2,4,3}\big)\Big\}^{+}\Big\} \cup I_{\Sigma} \end{aligned}$$

$$t = [w]_{\equiv_{I^+}} = \Big[\langle A_{1,2}, B_{1,1,2}, C_{1,1,2}, B_{1,2,2}, C_{1,2,2}, B_{1,3,2}, C_{1,3,2}, B_{1,4,2}, C_{1,4,2},$$
$$A_{1,3}, B_{1,1,3}, C_{1,1,3}, B_{1,2,3}, C_{1,2,3}, B_{1,3,3}, C_{1,3,3}, B_{1,4,3}, C_{1,4,3},$$
$$A_{2,3}, B_{2,2,3}, C_{2,2,3}, B_{2,3,3}, C_{2,3,3}, B_{2,4,3}, C_{2,4,3}\Big]_{\equiv_{I^+}}$$

Postać Normalna Foaty:

$$t = \langle [A] \rangle_{\equiv_{I^+}} = \big[\{A_{1,2}, A_{1,3}\}\big]_{\equiv_{I^+}}$$
$$\frown \big[\{B_{1,1,2}, B_{1,2,2}, B_{1,3,2}, B_{1,1,3}, B_{1,2,3}, B_{1,3,3}, B_{1,4,3}\}\big]_{\equiv_{I^+}}$$
$$\frown \big[\{C_{1,1,2}, C_{1,2,2}, C_{1,3,2}, C_{1,1,3}, C_{1,2,3}, C_{1,3,3}, C_{1,4,3}\}\big]_{\equiv_{I^+}}$$
$$\frown \big[\{A_{2,3}\}\big]_{\equiv_{I^+}} \frown \big[\{B_{2,2,3}, B_{2,3,3}, B_{2,4,3}\}\big]_{\equiv_{I^+}}$$
$$\frown \big[\{C_{2,2,3}, C_{2,3,3}, C_{2,4,3}\}\big]_{\equiv_{I^+}} =$$
$$[F_1]_{\equiv_{I^+}} \frown [F_2]_{\equiv_{I^+}} \frown [F_3]_{\equiv_{I^+}} \frown [F_4]_{\equiv_{I^+}} \frown [F_5]_{\equiv_{I^+}} \frown [F_6]_{\equiv_{I^+}}.$$

5 Zadanie 1

5.1 Treść zadania

Proszę zlokalizować niepodzielne czynności wykonywane przez algorytm, nazwać je oraz zbudować alfabet w sensie teorii śladów (dla macierzy o rozmiarze N).

5.2 Niepodzielne czynności

Mamy następujące niepodzielne zadania obliczeniowe za pomocą których można rozwiązać układy podane we wprowadzeniu do ćwiczenia oraz w przykładzie.

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

$$m_{k,i} = \frac{M_{k,i}}{M_{i,i}}$$

- $B_{i,j,k}$ - pomnożenie j-tego elementu wiersza i przez mnożnik - do odejmowania od k-tego wiersza,

$$n_{k,i} = M_{i,j} \cdot m_{k,i}$$

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

$$M_{k,j} = M_{k,j} - n_{k,i}.$$

5.3 Budowa alfabetu

Korzystając z pseudokodu algorytmu eliminacja Gaussa z punktu 3, zbudowałem poniższy alfabet w sensie teorii śladów, dla macierzy o rozmiarze N:

$$\Sigma = \{A_{i,k} \mid 1 \leq i < N,\ i < k \leq N\} \cup$$
$$\{B_{i,j,k} \mid 1 \leq i < N,\ i \leq j \leq N+1,\ i < k \leq N\} \cup$$
$$\{C_{i,j,k} \mid 1 \leq i < N,\ i \leq j \leq N+1,\ i < k \leq N\}$$

lub upraszczając:

$$\Sigma = \{A_{i,k},\ B_{i,j,k},\ C_{i,j,k} \mid 1 \leq i \leq N,\ i \leq j \leq N+1,\ i < k \leq N\}$$

Oznaczenia:
i - indeks aktualnie zerowanej kolumny
j - indeks kolumny aktualnie mnożonego bądź odejmowanego elementu
k - indeks wiersza, od którego odejmujemy i-ty wiersz

6 Zadanie 2

6.1 Treść zadania

Proszę skonstruować relację zależności dla alfabetu, opisującego algorytm eliminacji Gaussa (dla macierzy o rozmiarze N).

6.2 Relacja zależności

Według pseudokodu z punktu 3, najpierw musimy wyznaczyć mnożnik, zatem zaczniemy od relacji zależności między czynnością A a B (operacja A musi być wykonana przed B):

$$D_1 = \{(A_{i,k},\ B_{i',j',k'}) \ \mid \ A_{i,k} \in \Sigma,\ B_{i',j',k'} \in \Sigma\}$$

Następnie, zanim odejmiemy elementy od siebie, musimy wykonać operację mnożenia, więc C jest zależne od B:

$$D_2 = \{(B_{i,j,k},\ C_{i',j',k'}) \ \mid \ B_{i,j,k} \in \Sigma,\ C_{i',j',k'} \in \Sigma\}$$

Operacja C działa na elementach macierzy $M_{k,j}$, zatem będzie zależna od A, B, a także od samej siebie.

Zacznę od zależności C od samej siebie:

$$D_3 = \{(C_{i,j,k},\ C'_{i',j',k'}) \ \mid \ C_{i,j,k} \in \Sigma,\ C'_{i',j',k'} \in \Sigma,\ i < i'\}$$

Teraz zależność C od B:

$$D_4 = \{(C_{i,j,k},\ B_{i',j',k'}) \ \mid \ C_{i,j,k} \in \Sigma,\ B_{i',j',k'} \in \Sigma,\ i < i'\}$$

I na koniec C od A:

$$D_5 = \{(C_{i,j,k},\ A_{i',j'}) \ \mid \ C_{i,j,k} \in \Sigma,\ A_{i',j'} \in \Sigma,\ i < i'\}$$

Zatem z tego wszystkiego, otrzymujemy relację zależności:

$$D = sym\{\{D_1 \cup D_2 \cup D_3 \cup D_4 \cup D_5\}^+\} \cup I_\Sigma$$

6.3 Relacja niezależności

Relację niezależności możemy obliczyć posiadając tylko alfabet i relację zależności. Wzór prezentuje się następująco:

$$I = \Sigma^2 - D$$

7 Zadanie 3

7.1 Treść zadania

Proszę przedstawić algorytm eliminacji Gaussa w postaci ciągu symboli alfabetu (dla macierzy o rozmiarze N).

7.2 Rozwiązanie

Rozwiązanie zaczerpnę z napisanej przeze mnie aplikacji w Pythonie:

```
    # w dół
     for i in range(1, N + 1):
        for k in range(i + 1, N + 1):
            A(i, k)
            for j in range(i, N + 2):
                B(i, j, k)
                C(i, j, k)
    # normalizacja
    for i in range(1, N + 1):
        diag = matrix[i][i]
        for j in range(i, N + 2):
            matrix[i][j] /= diag
    # w górę
    for i in range(N, 0, -1):
        for k in range(i):
            A(i, k)
            for j in range(i, N + 2):
                B(i, j, k)
                C(i, j, k)
```

8 Zadanie 4

8.1 Treść zadania

Proszę wygenerować graf zależności Diekerta (dla macierzy o rozmiarze N).

8.2 Rozwiązanie

Zbiór wierzchołków grafu zależności Diekerta to zbiór niepodzielnych zadań obliczeniowych, więc $V = \Sigma$. Natomiast zbiór krawędzi to zbiór bezpośrednich zależności. Zatem, żeby móc stworzyć taki graf, potrzebujemy pozbyć się zależności pochodnych z relacji z zadania 2.

W zależnościach A od B oraz B od C (D_1 *i* D_2), nie ma zależności pochodnych, ponieważ są to operacje wykonywane podczas zerowania elementów pod przekątną. Możemy więc użyć ich jako podzbiory krawędzi.

$$E_1 = D_1 \qquad E_2 = D_2$$

W D_3 oraz D_4 interesują nas tylko zależności od i do $i+1$, bez tych w kolumnie $j \neq i+1$, ponieważ jest ona zależna od A, a my nie chcemy przechodniości.

$$E_3 = \{(C_{i,j,k},\ C'_{i',j',k'}) \ \mid \ C_{i,j,k} \in \Sigma,\ C'_{i',j',k'} \in \Sigma,\ i' = i+1,\ j' \neq i+1,\ j' = j,\ k' = k\}$$

$$E_4 = \{(C_{i,j,k},\ B_{i',j',k'}) \ \mid \ C_{i,j,k} \in \Sigma,\ B_{i',j',k'} \in \Sigma,\ i' = i+1,\ j' \neq i+1,\ j' = j,\ k = i+1\}$$

Sprowadzając do prostszej postaci:

$$E_3 = \{(C_{i,j,k},\ C'_{i+1,j,k}) \ \mid \ C_{i,j,k} \in \Sigma,\ C'_{i+1,j,k} \in \Sigma,\ j \neq i+1,\}$$

$$E_4 = \{(C_{i,j,i+1},\ B_{i+1,j,k'}) \ \mid \ C_{i,j,i+1} \in \Sigma,\ B_{i+1,j,k'} \in \Sigma,\ j \neq i+1\}$$

W ostatniej relacji, musimy wziąć zależności tylko od i do $i+1$.

$$E_5 = \{(C_{i,j,k},\ A_{i',j'}) \ \mid \ C_{i,j,k} \in \Sigma,\ A_{i',j'} \in \Sigma,\ j = i' = i+1,\ k = j' \vee k = i+1\}$$

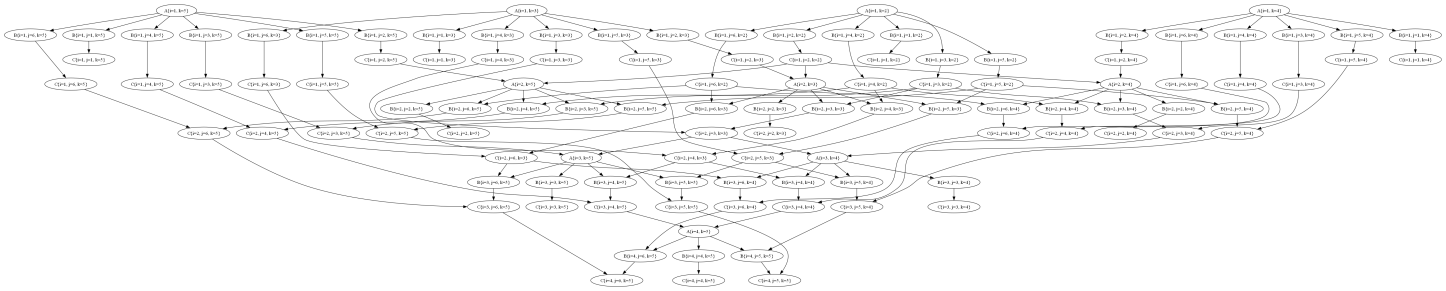
Upraszczając:

$$E_5 = \{(C_{i,i+1,k},\ A_{i+1,j'}) \ \mid \ C_{i,j,k} \in \Sigma,\ A_{i',j'} \in \Sigma,\ k = j' \vee k = i+1\}$$

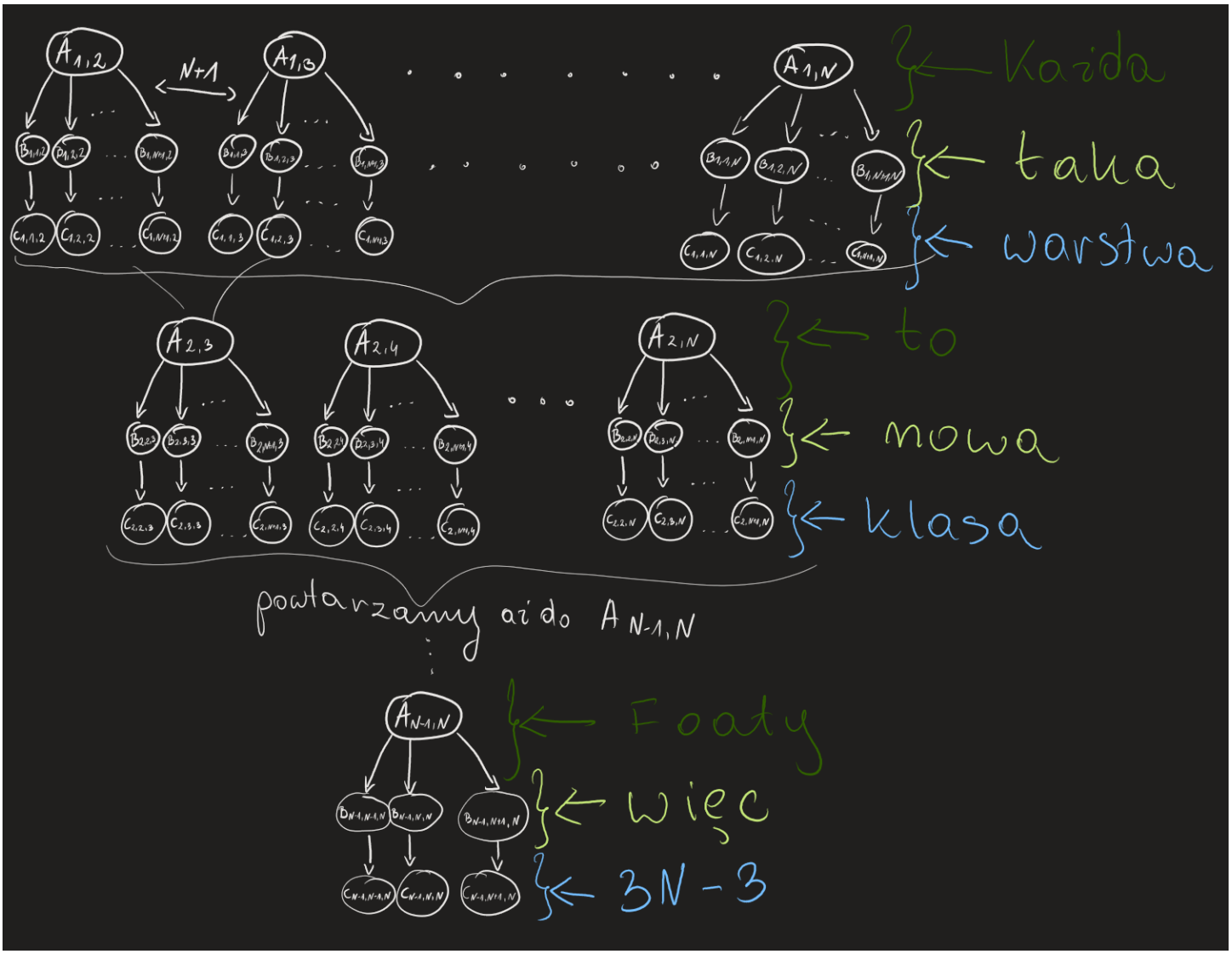
Podsumowując, nasz zbiór krawędzi E wygląda następująco:

$$E = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5$$

8.3 Przykładowy graf zależności Diekerta - 5x5



8.4 Graf zależności Diekerta - NxN



9 Zadanie 5

9.1 Treść zadania

Proszę przekształcić ciąg symboli opisujący algorytm do postaci normalnej Foaty (dla macierzy o rozmiarze N).

9.2 Postać normalna Foaty

Możemy zauważyć, że graf Diekerta układa się warstwami i każda taka warstwa będzie oddzielną klasą Foaty - jeden rodzaj operacji dla jednej kolumny to nowa klasa Foaty. Wszystkich klas będzie $3 * (N - 1)$, gdzie N to rozmiar macierzy.

$$F_{A_i} = \{A_{i,k} \mid k < i < N\}$$
$$F_{B_i} = \{B_{i,j,k} \mid k < i < N, 1 < j \leq N + 1\}$$
$$F_{C_i} = \{C_{i,j,k} \mid k < i < N, 1 < j \leq N + 1\}$$

Teraz mając wzory poszczególnych klas Foaty, możemy je pogrupować po 3

$$F_i = [F_{A_i}]_{\equiv_{I^+}} \frown [F_{B_i}]_{\equiv_{I^+}} \frown [F_{C_i}]_{\equiv_{I^+}}$$

Finalnie wyznaczamy ogólną postać klas Foaty:

$$[F_1]_{\equiv_{I^+}} \frown [F_2]_{\equiv_{I^+}} \frown ... \frown [F_{3N-3}]_{\equiv_{I^+}}$$

10 Zadanie implementacyjne

10.1 Treść zadania

Proszę zaprojektować i zaimplementować współbieżny algorytm eliminacji Gaussa. W szczególności proszę zwrócić uwagę na implementację jak najlepiej odwzorowującą graf zależności lub postać normalną Foaty. Sugerowane rozwiązania to wykorzystanie biblioteki GALOIS, biblioteki CILK, CUDA lub odpowiednich języków programowania.

Program ma działać dla zadanych rozmiarów macierzy N.

10.2 Zastosowany język programowania

Językiem programowania, użytym przeze mnie do implementacji problemu jest:
Python v. 3.12

Dodatkowo program wykorzystuje biblioteki:
Decimal - zwiększenie precyzji do 20 miejsc po przecinku
concurrent.futures - możliwość wielowątkowości w Pythonie
subprocess - pozwala uruchomić program "Graphviz"aby wygenerować graf Diekerta z pliku .dot

Do rysowania grafów zależności Diekerta z gotowego pliku .dot, wykorzystałem aplikację Graphviz dostępną pod linkiem:
https://graphviz.org/download/

10.3 Struktura katalogu

Plik **GaussElimination.py** - główna aplikacja rozwiązująca dane zadanie.

Folder **Tests** - folder zawierający 16 testów wygenerowanych z aplikacji sprawdzającej.

Folder **Results** - folder zawierający rozwiązania testów przez moją aplikację.

Folder **TaskTests** - są to wszystkie testy podane razem z treścią zadania w serwisie UPeL.

Plik **test.txt** - przykładowa macierz do uruchomienia programu z opcji 2 lub 3.

Plik **res.txt** - rozwiązanie powyższej przykładowej macierzy, przydatne do opcji 3.

Plik **graph.dot** - plik będący reprezentacją grafu Diekerta w formie tekstowej, dla podanego w opcji 4 rozmiaru (w przykładzie N=5).

Plik **graph.png** - reprezentacja grafu Diekerta w formie graficznej, wygenerowana z tekstu przez aplikację "Graphviz".

10.4 Jak uruchomić

Uruchamiamy w dowolnym środowisku Python 3+. Na konsoli wyświetli się wpierw możliwość wybrania interesującej nas opcji:
0 - uruchomienie testów podanych razem z zadaniem na UPeLu.
1 - uruchomienie 16 testów wygenerowanych w aplikacji podanej przez Pana Profesora Macieja Woźniaka.
(https://github.com/macwozni/Matrices)
2 - uruchomienie programu dla samemu dodanego pliku .txt z macierzą.
3 - uruchomienie programu dla samemu dodanych plików .txt, jednego z macierzą i jednego z rozwiązaniem, aby porównać wyniki.
4 - uruchomienie generatora grafu Diekerta dla podanego przez użytkownika rozmiaru N.

10.5 Testy

Zarówno w opcji 0 jak i opcji 1, uruchomione zostają testy sprawdzające poprawność wykonanego zadania. Testy dla opcji 0, były załączone razem z zadaniem w serwisie UPeL.

Natomiast w opcji 1, wygenerowane zostało 16 testów w wyżej wspomnianej przeze mnie aplikacji. Pliki zawierające macierz do rozwiązania zostały nazwane test0.txt - test15.txt, natomiast ich rozwiązania służące do porównania mojego rozwiązania nazywają się out0.txt - out15.txt. W programie ustawiona jest domyślnie obsługa 16 testów, więc aby sprawdzić nowe testy, należałoby podmienić aktualne znajdujące się w folderze Tests. Ewentualnie w kodzie programu GaussElimination.py w funkcji run() zmienić wartość test(15) na większą, w zależności od ilości dodanych testów.

Rozwiązania testów z opcji 1 znajdują się w folderze Results, a z opcji 0, razem z testami w folderze TaskTests.