**Teoria Współbieżności**

**Zadanie domowe (lab 7)**

**Stas Kochevenko, gr. 5**

**Zastosowaniem teorii śladów do szeregowania wątków współbieżnej eliminacji Gaussa**

1. **Podstawowe niepodzielne zadania obliczeniowe**
   Problem rozwiązywania układów równań liniowych metodą Gaussa można przedstawić w postaci:
   $$M \times x = y,$$
   gdzie:
   - M – macierz kwadratowa
   - x, y – wektory

   $$\begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

   Aby wykonać algorytm eliminacji Gaussa musimy zdefiniować następujące niepodzielne zadania obliczeniowe:
   - $A_{i,k}$ – obliczenie mnożnika, na który trzeby pomnożyć wiersz *i*, aby odjąć go od wiersza *k*:
     $$m_{k,i} = \frac{M_{k,i}}{M_{i,i}}$$
   - $B_{i,j,k}$ – przemnożenie każdego *j*-tego elementa wiersza *i* przez mnożnik $m_{k,i}$, aby odjąć go od wiersza *k*:
     $$n_{k,i} = M_{i,j} * m_{k,i}$$
   - $C_{i,j,k}$ – odjęcie każdego *j*-tego elementa wiersza *i* od wiersza *k*:
     $$M_{k,j} = M_{k,j} - n_{k,i}$$

2. **Identyfikacja ciągu zadań obliczeniowych wykonywanych przez algorytm sekwencyjny**
   Możemy przeprowadzić analizę ciągu zadań obliczeniowych, niezbędnych do wykonania algorytmu. Wykorzystamy przykład z wykładu:

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

Ten układ możemy przepisać w postaci:

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

Teraz musimy doprowadzić macierz do postaci trójkątnej górnej (wyzerować wszystkie elementy poniżej przekątnej), wykonując wcześniej zdefiniowane operacje.

Najpierw wyzerujemy pierwszy element pod przekątnej, czyli w naszym przypadku to $M_{2,1}$. Następnie będziemy zerować kolejne elementy kolejnego wiersza, aż dojdziemy do końca macierzy.

Ciąg zadań sekwencyjnych, niezbędnych do wykonania każdego kroku:

- Oblicz mnożnik $m_{k,i}$ $(A_{i,k})$
- Dla każdego $j$-tego elementa odejmij od elementa w wierszu $k$ element wiersza $i$ przemnożony przez $m_{k,i}$ $(B_{i,j,k} \rightarrow C_{i,j,k})$, czyli to powtarzamy $j$ razy
- Jeśli nie został osiągnięty koniec macierzy, wróć do punktu 1 i powtórz.

Dla naszego przykładu:

- Iteracja 1:

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

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

- Iteracja 2:

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

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

- Iteracja 3:

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

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

Na tym etapie macierz jest w postaci trójkątnej górnej.

## 3. Identyfikacja alfabetu w sensie teorii śladów

Nasz alfabet w sensie teorii sladów będzie wyglądał następująco:

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

Aby zidentyfikować alfabet dla przypadku ogólnego, możemy przyjąć

$\boldsymbol{s}$ – rozmiar macierzy $M$.

Alfabet:

$$\Sigma = \{A_{i,k} \mid 1 \leq i < s,\ i < k \leq s\} \cup$$

$$\{B_{i,j,k} \mid 1 \leq i < s,\ i < k \leq s,\ i \leq j \leq s+1\} \cup$$

$$\{C_{i,j,k} \mid 1 \leq i < s,\ i < k \leq s,\ i \leq j \leq s+1\}$$

## 4. Identyfikacja relacji zależności

Dla naszego przykładu (macierz $3 \times 3$) relacja zależności będzie wyglądała następująco:

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

Mamy dość rzadki przypadek, ponieważ każda akcja w alfabecie jest wykonywana dokładnie jeden raz, ale jest ściśle związana z poszczególnymi kolejnymi akcjami. Relacja jest dość złożona, wiąży się to z tym, że aby móc odjąć musimy najpierw pomnożyć, ale aby móc przemnożyć musimy najpierw mieć mnożnik $m_{k,i}$. Ponadto, aby przejść do obliczenia kolejnego wiersza, musimy mieć obliczone poprzednie.

Możemy teraz określić ogólną zależność pomiędzy obliczeniami w trakcie realizacji algorytmu Gaussa.

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

  Ten zbiór wskazuje na zależność pomiędzy $A_{i,k}$ (mnożnik) a wartością $B_{i,j,k}$.

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

  Ten zbiór opisuje zależność pomiędzy zmienną $B_{i,j,k}$ (element macierzy używany w mnożeniu) a $C_{i,j,k}$ (wynik odejmowania).

- $D_3 = \{(B_{ix,j,kx}, C_{iy,j,ky}) \mid B_{ix,j,kx}, C_{iy,j,ky} \in \Sigma \wedge i_x = k_y\}$

  Ten zbiór opisuje zależność pomiędzy zmienną $B_{ix,j,kx}$ a $C_{iy,j,ky}$ w sytuacji, gdy są one wykorzystywany w tej samej operacji (np. przy mnożeniu lub odejmowaniu).

- $D_4 = \{(A_{ix,kx}, C_{iy,jy,ky}) \mid A_{ix,kx}, C_{iy,jy,ky} \in \Sigma \wedge i_x = j_y \wedge (i_x = k_y \vee k_x = k_y)\}$

  Ten zbiór opisuje zależność pomiędzy zmienną $A_{ix,kx}$ a $C_{iy,j,ky}$ w sytuacji, gdy indeks wiersza A jest równy indeksowi kolumny C. To oznacza, że A dostarcza mnożnik, który jest niezbędny do zaktualizowania wartości w C.

- $D_5 = \{(C_{ix,j,k}, C_{iy,j,k}) \mid C_{ix,j,k}, C_{iy,j,k} \in \Sigma\}$

  Ten zbiór określa, że zmiany tego samego elementu pewnego wiersza są między sobą zależne.

- $D = sym((D_1 \cup D_2 \cup D_3 \cup D_4 \cup D_5)^+) \cup I_\Sigma$

  Możemy również zdefiniować relację niezależności, która będzie po prostu różnicą:

- $I = \Sigma^2 - \mathrm{D}$

## 5. Algorytm eliminacji Gaussa

Możemy przedstawić algorytm eliminacji w postaci ciągu pewnych operacji.

Pryzjmujemy $\boldsymbol{s}$ – rozmiar macierzy $M$.

Niech $\boldsymbol{w_{i,k}} = (A_{i,k}, B_{i,i,k}, C_{i,i,k}, B_{i,i+1,k}, C_{i,i+1,k}, \ldots, B_{i,s,k}, C_{i,s,k})$

To oznacza wyzerowanie elementu macierzy $M_{i,k}$ (w wyniku odejmowania i-tego wiersza od k-tego, czyli wykonania wcześniej zdefiniowanych akcji).

Na podstawie tego możemy przedstawić cały algorytm jako konkatenację ciągów:

$$(w_{1,2}, w_{1,3}, w_{1,4}, \ldots, w_{1,s}, w_{2,3}, w_{2,4}, \ldots w_{s-2,s-1}, w_{s-2,s}, w_{s-1,s}).$$

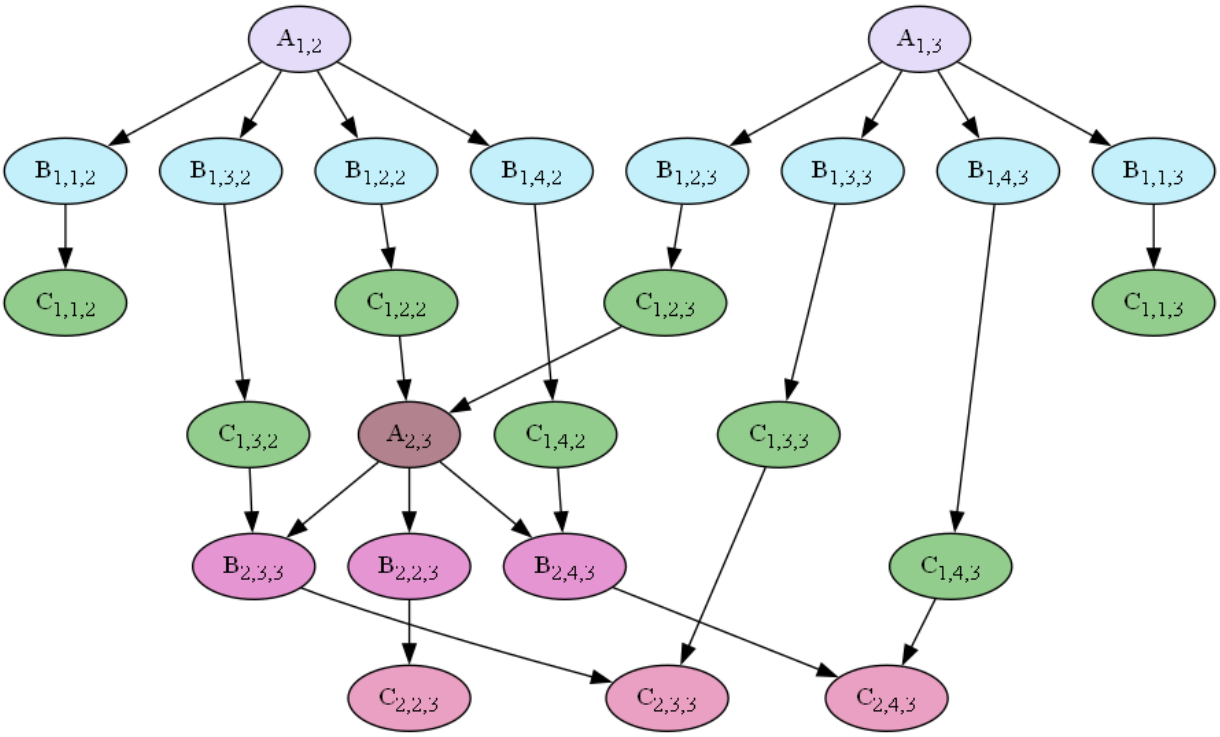
Później zostanie podstawienie wstecz.

## 6. Wyprowadzenie grafu zależności Diekerta

Krawędzie w grafie Diekerta – to zbiór wszystkich bezpośrednich zależności. Korzystamy ze wcześniej zdefiniowanego D:

- $E_1 = \{(A_{i,k}, B_{i,j,k}) \mid A_{i,k}, B_{i,j,k} \in \Sigma\}$
- $E_2 = \{(B_{i,j,k}, C_{i,j,k}) \mid B_{i,j,k}, C_{i,j,k} \in \Sigma\}$
- $E_3 = \{(B_{ix,j,kx}, C_{iy,j,ky}) \mid B_{ix,j,kx}, C_{iy,j,ky} \in \Sigma \wedge i_x = k_y \wedge i_x = i_y + 1 \wedge i_y \neq j\}$
- $E_4 = \{(A_{ix,kx}, C_{iy,j,ky}) \mid A_{ix,kx}, C_{iy,jy,ky} \in \Sigma \wedge i_x = j \wedge i_x = i_y + 1 \wedge (i_x = k_y \vee k_x = k_y)\}$
- $E_5 = \{(C_{ix,j,k}, C_{iy,j,k}) \mid C_{ix,j,k}, C_{iy,j,k} \in \Sigma \wedge i_x = i_y - 1 \wedge i_y \neq j\}\}$
- $E = ((D_1 \cup D_2 \cup D_3 \cup D_4 \cup D_5)^+) \cup I_\Sigma$

Co daje następujący graf Diekerta (przykładowo dla macierzy o rozmiarte 3x3):



## 7. Obliczenia klas Foaty

Dla przykładu z wykładu:

$$
\begin{aligned}
t = [\langle A \rangle]_{\equiv_I^+} &= [\{A_{1,2}, A_{1,3}\}]_{\equiv_I^+} \\
&\frown [\{B_{1,1,2}, B_{1,2,2}, B_{1,3,2}, B_{1,4,2}, B_{1,1,3}, B_{1,2,3}, B_{1,3,3}, B_{1,4,3}\}]_{\equiv_I^+} \\
&\frown [\{C_{1,1,2}, C_{1,2,2}, C_{1,3,2}, C_{1,4,2}, C_{1,1,3}, C_{1,2,3}, C_{1,3,3}, C_{1,4,3}\}]_{\equiv_I^+} \\
&\frown [\{A_{2,3}\}]_{\equiv_I^+} \frown [\{B_{2,2,3}, B_{2,3,3}, B_{2,4,3}\}]_{\equiv_I^+} \\
&\frown [\{C_{2,2,3}, C_{2,3,3}, C_{2,4,3}\}]_{\equiv_I^+} = \\
&[F_1]_{\equiv_I^+} \frown [F_2]_{\equiv_I^+} \frown [F_3]_{\equiv_I^+} \frown [F_4]_{\equiv_I^+} \frown [F_5]_{\equiv_I^+} \frown [F_6]_{\equiv_I^+}
\end{aligned}
$$

Klasy Foaty przyjmują prostą postać, która wynika ze zbudowanego grafu w poprzednim punkcie.

Dla kolejnych n = 1, 2, 3, … s – 1 klas Foaty mamy:

$$F_{An} = \{A_{n,k} \mid n \leq \mathrm{k} \leq \mathrm{s}\}$$
$$F_{Bn} = \{B_{n,j,k} \mid n \leq \mathrm{k} \leq \mathrm{s} \wedge n < \mathrm{j} \leq \mathrm{s}+1\}$$
$$B_{Cn} = \{C_{n,j,k} \mid n \leq \mathrm{k} \leq \mathrm{s} \wedge n < \mathrm{j} \leq \mathrm{s}+1\}$$

Wszystkie klasy Foaty:

$$[F_{A1}]\,[F_{B1}]\,[F_{C1}]\,[F_{A2}]\,[F_{B2}]\,[F_{C2}]\,\ldots\,[F_{As-1}]\,[F_{Bs-1}]\,[F_{Cs-1}]$$

## 8. Implementacja współbieżnej eliminacji Gaussa

Program, realizujący dane podejście został napisany w języku C++.

Pliki:

- diekert.cpp – do generowania grafu Diekerta
- elimination.cpp – do współbieżnego obliczenia wyniku eliminacji Gaussa
- in.txt, out.txt – przykładowe wejście do programu elimination oraz oczekiwane wyjście
- diekert_example – graf Diekerta, wygenerowany dla macierzy rozmiaru 3 x 3

Zależności:

Przed uruchomieniem programu należy zainstalować graphvis:

- sudo apt install graphviz

Uruchomienie programu:

Abu uruchomić generator grafu Diekerta diekert.cpp należy wpisać:

- g++ -o diekert diekert.cpp
- ./diekert **n**

gdzie n – to rozmiar macierzy.

Abu uruchomić współbieżne obliczenia eliminacji Gaussa elimination.cpp należy wpisać:

- g++ elimination.cpp -o elimination
- ./elimination **file**

gdzie **file** – to opcjonalna nazwa pliku wejściowego, np. „in.txt”.

Domyślnie zostanie użyty plik „in.txt”.

## 9. Wynik działania eliminacji Gaussa dla pryzkładowych danych

Dla macierzy rozmiaru 15x15:

```
15
0.3598381494 0.8409342858 0.2078457000 0.3076592230 0.5746280031 0.3193307736 0.3614672566 0.7279397259 0.9533114347 0.0125489458 0.3668514230 0.0822051884 0.6728263458 0.2414697087 0.8907866449
0.0990509280 0.8448001488 0.7335959037 0.4410333892 0.0004641690 0.2571903929 0.2827449953 0.1344851542 0.8379886466 0.9408241372 0.7638297249 0.7754726642 0.3612390876 0.4475851775 0.6895253650
0.0611367836 0.5240200388 0.3451626587 0.9028476472 0.9895839527 0.4061565322 0.5826884804 0.1881432569 0.8365774747 0.2832440536 0.0622547836 0.5527200972 0.3878068150 0.8118781254 0.9632170592
0.6823218116 0.4663162978 0.8145247654 0.1093526633 0.6058778448 0.1784634465 0.9460446959 0.9433581349 0.9184359435 0.7168164496 0.6029857273 0.4788331646 0.0059115238 0.0046892389 0.7971097761
0.9368129044 0.5912644209 0.9445518145 0.3562214748 0.6420745397 0.5789400685 0.5413422112 0.7382183675 0.1538579404 0.6128888873 0.7117459899 0.9483826822 0.5363043470 0.9832042774 0.1617978656
0.5352875036 0.1358311358 0.5173959241 0.3176994038 0.4072910492 0.2258247967 0.8745417488 0.9285354458 0.2594445028 0.4034480317 0.2240388131 0.6246113122 0.7297402803 0.6790216676 0.6353992729
0.4114219336 0.0139796539 0.7853549691 0.6315148523 0.8068024749 0.5601392847 0.3525633491 0.2650392302 0.0876001591 0.6359691166 0.2433253886 0.7660147408 0.9022450545 0.0176539539 0.5241959248
0.2785960288 0.3268870842 0.7827245496 0.2605054719 0.4557001634 0.2953058685 0.2070036651 0.5655408288 0.7126389005 0.2165884607 0.0632256706 0.1091152343 0.7542658297 0.5096093700 0.1980688909
0.5963010280 0.5848972130 0.7296366203 0.1246918985 0.5640114466 0.5862422241 0.9399262573 0.5094661313 0.2112823140 0.4058047461 0.9899567962 0.8544495118 0.8785187571 0.3366953742 0.4019945394
0.3812462110 0.2216356655 0.5895190368 0.6389691158 0.1072504053 0.0889349987 0.3579798574 0.6291786687 0.3718725101 0.6017271235 0.8620720106 0.5571317762 0.6181986809 0.9641170863 0.6162011510
0.0948736058 0.0646352080 0.0830413107 0.0143082636 0.4444321599 0.0114416364 0.1665537668 0.9500438661 0.7895594488 0.9186577436 0.0865250001 0.5972672054 0.0137319643 0.0647808592 0.4413314562
0.6646218739 0.8314134551 0.5496519350 0.9347766200 0.4076652435 0.8008724220 0.6403601923 0.8993992383 0.4330066054 0.0393853043 0.9755776891 0.5354885381 0.8611343274 0.8454461847 0.9415865965
0.7115397145 0.7280899996 0.8476419914 0.6123640034 0.9775477666 0.6008731913 0.7461419270 0.0064400154 0.7794512955 0.2574379281 0.1902886965 0.0548470032 0.7312017753 0.9393672943 0.8445678922
0.5973289158 0.2003489442 0.1237908044 0.2352716256 0.9280816961 0.3936953738 0.9808339366 0.5414606953 0.8976471080 0.5914889976 0.6838673279 0.0963694169 0.3441834018 0.1717851197 0.5220487224
0.9991857941 0.6751889463 0.5910245339 0.3281276648 0.9021790219 0.7498552792 0.5425374148 0.0792096039 0.7344354286 0.8225782968 0.1015977843 0.6541523947 0.8510025307 0.1997580345 0.7838164701
0.6384948161 0.8550308482 0.4348680819 0.0930752870 0.7562935026 0.1305411126 0.3408784191 0.3158685483 0.9779708263 0.9447345766 0.8344664526 0.0308326939 0.6042533948 0.8313023673 0.6430521759
```

Spodziewany wynik:

```
15
1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
-0.5535598627 -0.0506765114 -0.3473165237 -1.6542570708 1.2551143962 -0.3814515519 -0.9539278208 -0.3477037635 -0.1487458824 0.6463610025 1.1471447558 -0.0669724262 0.6037270621 0.8729251919 0.6301394491
```

Wynik działania algorytmu:

```
15
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
-0.55356 -0.0506765 -0.347317 -1.65426 1.25511 -0.381452 -0.953928 -0.347704 -0.148746 0.646361 1.14714 -0.0669724 0.603727 0.872925 0.630139
```

**Wnioski**

W trakcie wykonywania tego zadania domowego udało się zlokalizować niepodzielne czynności wykonywane przez algorytm eliminacji Gaussa, skonstruować relacje zależności dla alfabetu, wygenerować graf zależności Diekerta, wyznaczyć klasy Foaty. Został zaimplementowany współbieżny algorytm eliminacji Gaussa na podstawie wyznaczonych zależności oraz krawędzi grafa Diekerta. Przeprowadziłem testowanie na kilku przykładowych danych. Otrzymane wyniki po wykonaniu algorytmu są zgodne ze spodziewanymi.