Teoria Współbieżności

Współbieżna eliminacja Gaussa z wykorzystaniem Teorii Śladów

Aga Patro

1. Cel ćwiczenia

Celem ćwiczenia było zapoznanie studentów z zastosowaniem teorii śladów do szeregowania wątków z zastosowaniem w klasycznych algorytmach algebry liniowej. Jako przykład omawiany był algorytm współbieżnej eliminacji Gaussa.

2. Opis zadań

2.1 Część teoretyczna

W tej części należało dla zadanej wielkości macierzy N:

1. Zlokalizować niepodzielne czynności wykonywane przez algorytm, nazwać je oraz zbudować alfabet w sensie teorii śladów.
2. Wyznaczyć relacje zależności dla alfabetu D opisującego algorytm eliminacji Gaussa.
3. Przedstawić algorytm eliminacji Gaussa w postaci ciągu symboli alfabetu.
4. Wygenerować graf zależności Diekerta
5. Przedstawić ciąg symboli opisujący algorytm jako postać normalną Foaty.

2.2 Część implementacyjna

Natomiast w tej części należało:

1. Zaprojektować i zaimplementować współbieżny algorytm eliminacji Gaussa.
2. Zweryfikować poprawność algorytmu.

3. Sposób realizacji zadań

3.1 Część teoretyczna

W celu realizacji zadania napisałam odpowiednie funkcje i klasy w języku Python 3.10. By narysować reprezentację grafu Diekerta użyłam modułu *Digraph* z biblioteki *graphviz*. Wyniki poszczególnych zagadnień zapisywane są do pliku "*theory_result.txt*" po zakończeniu działania programu, natomiast rysunki grafów zapisywane są do odpowiednich plików .png i .pdf w katalogu *"graphs"*.

Ponadto, by "skomunikować" część teoretyczną z częścią implementacyjną, postać normalną FNF oraz wczytane wartości elementów macierzy zapisuję do pliku .json, który "odbiera" część implementacyjna.

3.2 Część implementacyjna

By zrealizować część implementacyjną, napisałam odpowiednie klasy i metody w języku Java 19. W celu weryfikacji poprawności mojego rozwiązania, stworzyłam randomowe macierze o rozmiarach 2, 3, 4 (ich zapis znajduje się w katalogu "*examples*"), oraz przeprowadziłam dla nich eksperymenty. Otrzymane wyniki porównałam z macierzami otrzymanymi przez kalkulatory dostępne online.

4. Rozwiązanie teoretyczne

Mamy zadany układ równań w postaci:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Wyraźmy dany układ postacią:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & | & a_{1(n+1)} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} & | & a_{2(n+1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & | & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} & | & a_{n(n+1)} \end{bmatrix}$$

gdzie $a_{i(n+1)} = b_i$.

Algorytm eliminacji Gaussa polega na:

1. $A_{i,k} -$ znalezieniu mnożnika dla wiersza i, do odejmowania go od k-tego wiersza,
$$m_{k,i} = M_{k,i}/M_{i,i}$$
2. $B_{i,j,k} -$ pomnożeniu j-tego elementu wiersza i przez mnożnik - do odejmowania od k-tego wiersza,
$$n_{k,i} = M_{i,j} * m_{k,i}$$
3. $C_{i,j,k} -$ odjęcie j-tego elementu wiersza i od wiersza k,
$$M_{k,j} = M_{k,j} - n_{k,i}$$

Alfabet możemy zdefiniować jako:

$$\Sigma = \{A_{i,k}, B_{i,j,k}, C_{i,j,k}\}$$

gdzie: $i = \{1, 2,.., N), k = \{i + 1,...., N\}, j = \{i + 1, \dots , N + 1\}, N -$ rozmiar macierzy

Algorytm sekwencyjny możemy zapisać jako:

$$t = A_{m,k}, B_{m,m,k}, C_{m,m,k},..., B_{m,i,k}, C_{m,i,k}, ..., B_{m,(n+1),k}, C_{m,(n+1),k}$$

gdzie: $k = \{2, 3,..., N\}, m = \{1, 2,..., k - 1\}$

Możemy zauważyć, że występują poniższe zależności pomiędzy operacjami:

- $A_{i,k}$ jest zależne od $C_{i-1,i,i}$ oraz $C_{i-1,i,k}$
- $B_{i,j,k}$ jest zależne od $A_{i,k}$ oraz $C_{i-1,j,k-1}$
- $C_{i,j,k}$ jest zależne od $B_{i,j,k}$ oraz $C_{i-1,j,k}$

Dlatego **relację zależności** możemy przedstawić jako:

$$D = sym\{(A_{i,k}, C_{i-1,i,i}), (A_{i,k}, C_{i-1,i,k}), (B_{i,j,k}, A_{i,k}), (B_{i,j,k}, C_{i-1,j,k-1}), (C_{i,j,k}, B_{i,j,k}), (C_{i,j,k}, C_{i-1,j,k})\} \cup I_{\Sigma}$$

gdzie: $i = \{1, 2,.., N\}$, $k = \{i + 1,...., N\}$, $j = \{i + 1, ... , N + 1\}$, $N -$ rozmiar macierzy

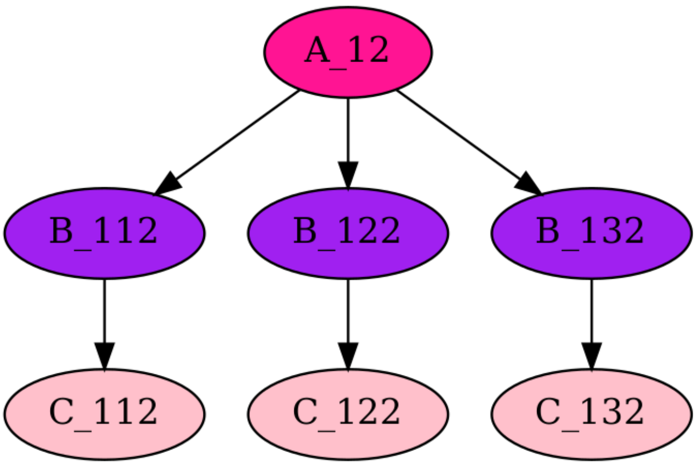
Do wyznaczenia **grafu Diekerta** oraz **postaci normalnej Foaty** użyłam programów z zadania domowego 2.

5. Otrzymane wyniki

5.1 Macierz o rozmiarze $n = 2$

5.1.1 Część teoretyczna

- Alfabet $A = \{A\_12, B\_112, B\_122, B\_132, C\_112, C\_122, C\_132\}$
- Algorytm sekwencyjny $\{A\_12, B\_112, C\_112, B\_122, C\_122, B\_132, C\_132\}$
- Relacja zależności D:
  $\{(B\_112, A\_12), (C\_122, B\_122), (C\_132, B\_132), (B\_132, A\_12), (B\_122, A\_12), (C\_112, B\_112)\}$
- FNF: $(A\_12)(B\_112, B\_122, B\_132)(C\_112, C\_122, C\_132)$

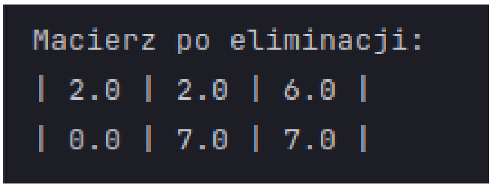


Rysunek 5.1.1.1 Graf zależności w postaci minimalnej dla $n = 2$

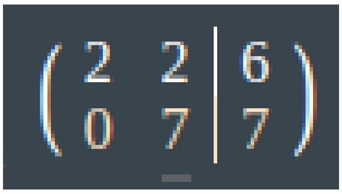
5.1.2 Testy części implementacyjnej

W celu przetestowania wygenerowałam macierz 2x3:

$$\begin{bmatrix} 2.0 & 2.0 & 6.0 \\ 1.0 & 8.0 & 10.0 \end{bmatrix}$$



Rysunek 5.1.2.1 Wynik eliminacji gaussa otrzymany przez mój program



Rysunek 5.1.2.2 Wynik eliminacji gaussa otrzymany przez kalkulator online

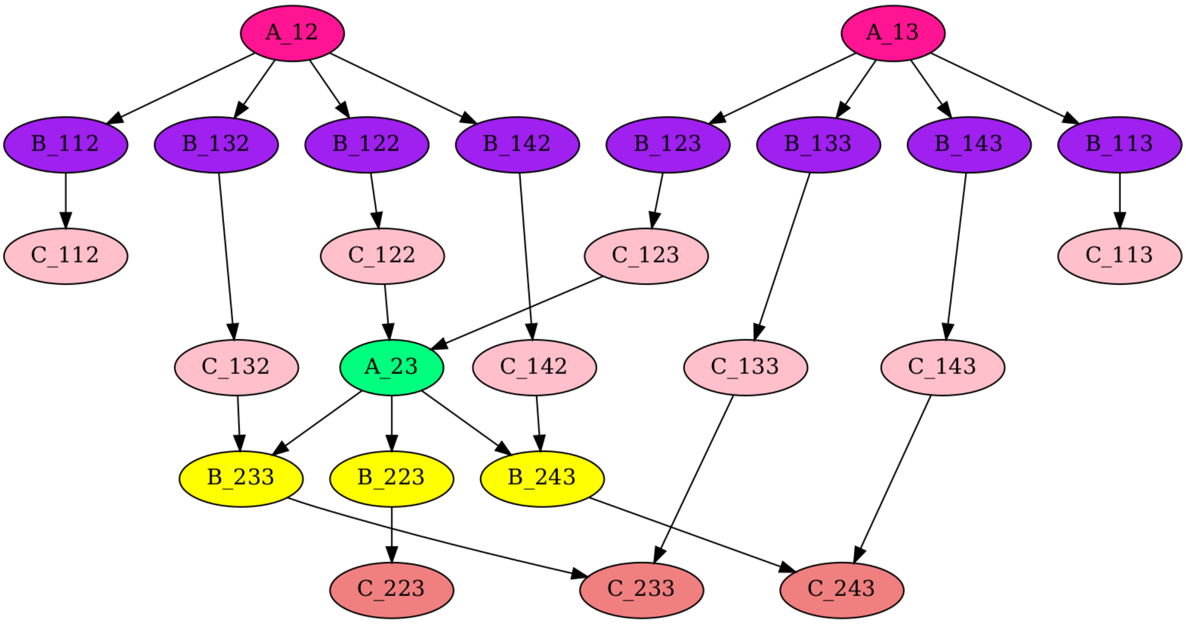
Jak widać na powyższych rysunkach, dla $n = 2$ program działa prawidłowo.

5.2 Macierz o rozmiarze $n = 3$

5.2.1 Część teoretyczna

- Alfabet $A = \{A\_12, A\_13, A\_23, B\_112, B\_113, B\_122, B\_123, B\_132, B\_133, B\_142, B\_143, B\_223, B\_233, B\_243, C\_112, C\_113, C\_122, C\_123, C\_132, C\_133, C\_142, C\_143, C\_223, C\_233, C\_243\}$
- Algorytm sekwencyjny: $\{A\_12, B\_112, C\_112, B\_122, C\_122, B\_132, C\_132, B\_142, C\_142, A\_13, B\_113, C\_113, B\_123, C\_123, B\_133, C\_133, B\_143, C\_143, A\_23, B\_223, C\_223, B\_233, C\_233, B\_243, C\_243\}$
- Relacja zależności D:
  $\{(B\_233, A\_23), (C\_243, C\_143), (B\_112, A\_12), (C\_223, B\_223), (B\_133, A\_13), (B\_233, C\_132), (C\_113, B\_113), (C\_223, C\_123), (C\_233, B\_233), (C\_122, B\_122), (C\_112, B\_112), (B\_113, A\_13), (B\_132, A\_12), (B\_243, A\_23), (B\_243, C\_142), (B\_223, C\_122), (B\_223, A\_23), (A\_23, C\_123), (A\_23, C\_122), (C\_142, B\_142), (C\_133, B\_133), (B\_123, A\_13), (C\_143, B\_143), (C\_233, C\_133), (C\_243, B\_243), (C\_123, B\_123), (C\_132, B\_132), (B\_122, A\_12), (B\_142, A\_12), (B\_143, A\_13)\}$

- FNF:
  $(A\_12, A\_13)(B\_112, B\_122, B\_132, B\_142, B\_113, B\_123, B\_133, B\_143)$
  $(C\_112, C\_122, C\_132, C\_142, C\_113, C\_123, C\_133, C\_143)(A\_23)$
  $(B\_223, B\_233, B\_243)(C\_223, C\_233, C\_243)$

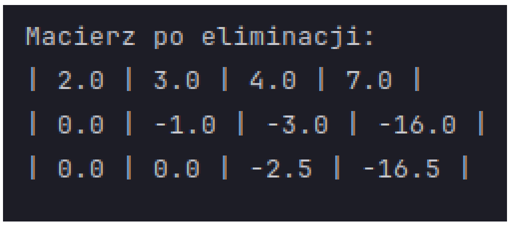


Rysunek 5.2.1.1 Graf zależności w postaci minimalnej dla $n = 3$

5.2.2 Testy części implementacyjnej

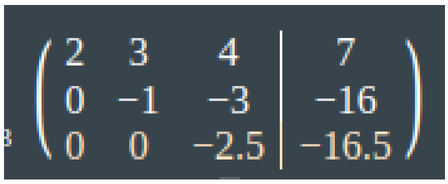
W celu przetestowania wygenerowałam macierz 3x4:

$$\begin{bmatrix} 2.0 & 3.0 & 4.0 & 7.0 \\ 6.0 & 8.0 & 9.0 & 5.0 \\ 5.0 & 8.0 & 9.0 & 9.0 \end{bmatrix}$$



```
Macierz po eliminacji:
| 2.0 | 3.0 | 4.0 | 7.0 |
| 0.0 | -1.0 | -3.0 | -16.0 |
| 0.0 | 0.0 | -2.5 | -16.5 |
```

Rysunek 5.2.2.1 Wynik eliminacji gaussa otrzymany przez mój program



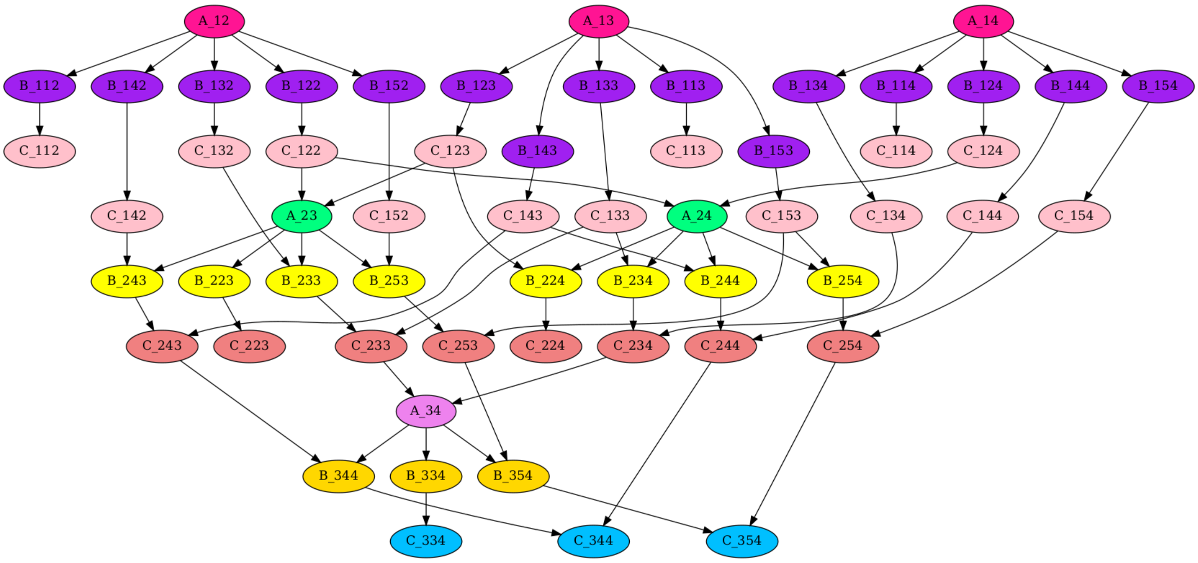
$$\left(\begin{array}{ccc|c} 2 & 3 & 4 & 7 \\ 0 & -1 & -3 & -16 \\ 0 & 0 & -2.5 & -16.5 \end{array}\right)$$

Rysunek 5.2.2.2 Wynik eliminacji gaussa otrzymany przez kalkulator online

Jak widać na powyższych rysunkach, dla $n = 3$ program działa prawidłowo.

5.3 Macierz o rozmiarze $n = 4$

5.3.1 Część teoretyczna



Rysunek 5.3.1.1 Graf zależności w postaci minimalnej dla $n = 4$

5.3.2 Testy części implementacyjnej

W celu przetestowania wygenerowałam macierz 4x5:

$$\begin{bmatrix} 5.0 & 8.0 & 9.0 & 0.0 & 7.0 \\ 7.0 & 5.0 & 4.0 & 3.0 & 9.0 \\ 56.0 & 7.0 & 87.0 & 8.0 & 90.0 \\ 6.0 & 5.0 & 4.0 & 3.0 & 8.0 \end{bmatrix}$$

```
Macierz po eliminacji:
| 5.0 | 8.0 | 9.0 | 0.0 | 7.0 |
| 0.0 | -6.2 | -8.599999 | 3.0 | -0.8000002 |
| 0.0 | 0.0 | 100.774185 | -31.967743 | 22.258064
| 0.0 | 0.0 | 0.0 | 0.6411649 | 0.28617123 |
```

Rysunek 5.2.3.1 Wynik eliminacji gaussa otrzymany przez mój program

$$\left(\begin{array}{cccc|c} 5 & 8 & 9 & 0 & 7 \\ 0 & -6.2 & -8.6 & 3 & -0.8 \\ 0 & 0 & 100.774 & -31.968 & 22.258 \\ 0 & 0 & 0 & 0.641 & 0.286 \end{array}\right)$$

Rysunek 5.3.2.2 Wynik eliminacji gaussa otrzymany przez kalkulator online

Jak widać na powyższych rysunkach, dla $n = 4$ program działa prawidłowo.