Szczepan Rzeszutek piątek 18.20

Teoria Współbieżności

ZADANIE 7

1.Niepodzielne operacje

Rozważmy równanie $M \times x = b$ gdzie M to macierz kwadratowa, natomiast x i b to wektory.
Równanie można zapisać za pomocą macierzy:

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

gdzie n to liczba niewiadomych oraz liczba równań.

Operacje

$A_{i,k}$ - 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:

$n_{k,j,i} = M_{i,j} * m_{k,i}$

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

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

2.Alfabet

N - rozmiar macierzy

$$\Sigma = \left\{A_{i,k}, B_{i,j,k}, C_{i,j,k} \mid i = 1 \dots N - 1\ ,\ k = i + 1 \dots N\ ,\ j = i \dots N + 1\right\}$$

3. Relacje

$$D1 = \left\{\left(A_{i,k}, B_{i,j,k}\right) \mid A_{i,k}, B_{i,j,k} \in \Sigma\right\}$$

Operacja $A_{i,k}$ musi być obliczona przed $B_{i,j,k}$, ponieważ wyniku $A_{i,k}$ używamy w mnożeniu w $B_{i,j,k}$.

$$D2 = \left\{\left(B_{i,j,k}, C_{i,j,k}\right) \mid C_{i,j,k}, B_{i,j,k} \in \Sigma\right\}$$

Operacja $B_{i,j,k}$ musi być obliczona przed $C_{i,j,k}$, ponieważ wyniku $B_{i,j,k}$ używamy w odejmowaniu w $C_{i,j,k}$.

$$D3 = \left\{\left(C_{i-1,j,k}, C_{i,j,k}\right) \mid C_{i-1,j,k}, C_{i,j,k} \in \Sigma\, and\ \ i! = j\ and\ i > 1\right\}$$

Operacja $C_{i-1,j,k}$ musi być obliczona przed $C_{i,j,k}$, ponieważ $C_{i,j,k}$ korzysta z $B_{i,j,k}$, które potrzebuje $C_{i-1,j,k}$ do swoich obliczeń.

$$D4 = \left\{\left(C_{i-1,j,i}, B_{i,j,kb}\right) \mid C_{i-1,j,i}, B_{i,j,kb} \in \Sigma\ \wedge\ \ j! = i\ and\ i > 1\right\}$$

Operacja $C_{i-1,j,i}$ musi być obliczona przed $B_{i,j,kb}$, ponieważ korzysta z elementu od którego odejmuje operacja $C_{i-1,j,i}$

$$D5 = \left\{ \left(C_{i-1,i,kc}, A_{i,ka} \right) \mid C_{i-1,i,kc}, A_{i,ka} \in \Sigma \wedge (ka = kc \vee kc = i)\ and\ i >= 2 \right\}$$

Operacja $C_{i-1,i,k\,c}$ musi być obliczona przed $A_{i,ka}$, ponieważ $A_{i,ka}$ korzysta z elementu od którego odejmuje operacja $C_{i-1,i,k\,c}$.

Dzięki takim zabiegom, pomijamy przechodniość i dostajemy relacje zapisaną w postaci krawędzi grafu Diekerta.

$$D = sym\left((D1 \cup D2 \cup D3 \cup D4 \cup D5)^{+} \right) \cup I_{\Sigma}$$

Powyżej zapis wszystkich relacji po dodaniu przechodniości, symetryczności i identyczności.

Relacja niezależności

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

4. Algorytm

N - rozmiar macierzy

$Z_{i,k}$ - operacja odjęcia całego wiersza k od i w celu wyzerowania komórki $M_{i,k}$

$$Z_{i,k} = A_{i,k}, B_{i,i,k}, C_{i,i,k}, B_{i,i+1,k}, C_{i,i+1,k} \dots B_{i,N+1,k}, C_{i,N+1,k}$$

Nasz algorytm wykonuje po kolei operacje:

$Z_{1,2}, Z_{1,3} \dots Z_{1,N}$ -"zerujemy" 1 kolumnę

$Z_{2,3}, Z_{2,4} \dots Z_{2,N}$ -"zerujemy" 2 kolumnę

$Z_{3,4}, Z_{3,4} \dots Z_{3,N}$ -"zerujemy" 3 kolumnę

...

$Z_{N-1,N}$ -"zerujemy" N-1 kolumnę

Pseudokod

```
for(int i=1;i<=N-1;i++){
        for(int k=i+1,k<=N,k++){
                Z_{i,k}
        }
}
```

po rozpisaniu $Z_{i,k}$

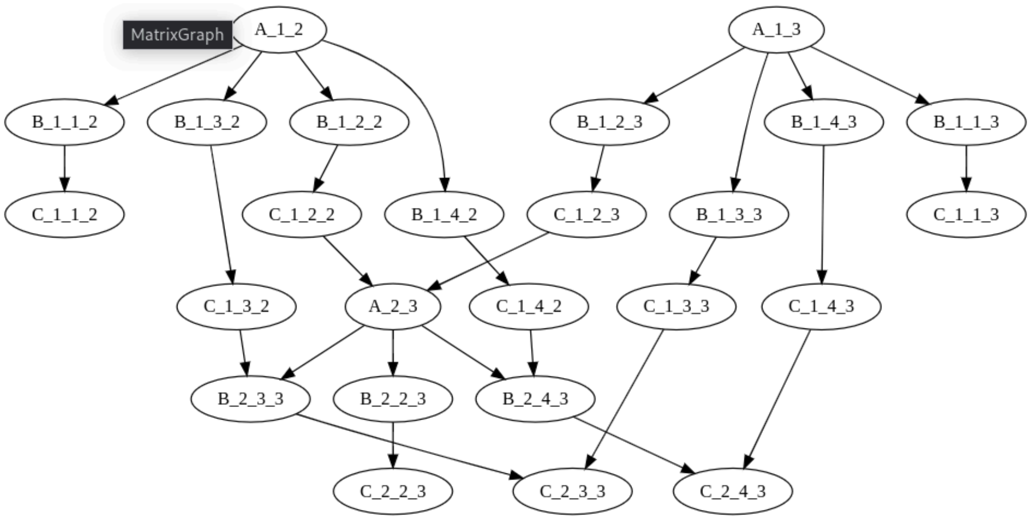
```
for(int i=1;i<N-1;i++){
        for(int k=i+1,k<N,k++){
                A_{i,k}
                for(int j=i,j<=N+1,j++){
                        B_{i,j,k}
                        C_{i,j,k}

                }
        }
}
```

5. Graf Diekerta

Dzięki temu że zapisałem relacje w powyższy sposób to:

$E = D1 \cup D2 \cup D3 \cup D4 \cup D5$



Przykładowy graf Diekerta dla N=3, wygenerowany programem graph.c.

6.Klasy Foaty

Patrząc na graf można łatwo wyznaczyć klasy Foaty:

N - rozmiar macierzy

$$i = 1 \dots N - 1$$

$$FAi = \left\{A_{i,k} | k = i + 1 \dots N\right\}$$
$$FBi = \left\{B_{i,j,k} | \ k = i + 1 \dots N, \ j = i \dots N + 1\right\}$$
$$FCi = \left\{C_{i,j,k} | \ k = i + 1 \dots N, \ j = i \dots N + 1\right\}$$

Wszystkie klasy dla N:

$$\left[FA_1\right], \left[FB_1\right], \left[FC_1\right], \left[FA_2\right], \ \dots, \left[FB_{N-1}\right], \left[FC_{N-1}\right]$$

7.Implementacja

Program napisałem w języku C z użyciem OpenMP.
Implementacja algorytmu znajduje się w pliku gauss.c,
aby skompilować program wystarczy użyć **make gauss.**
Dane wejściowe muszą się znajdować w pliku data.txt, a odpowiedź wygeneruje się w pliku res.txt.

Macierz M i macierze pomocnicze

```
double **matrix = (double **)malloc(n * sizeof(double *));
double **diff_matrix = (double **)malloc(n * sizeof(double *));
double **multi_matrix = (double **)malloc(n * sizeof(double *));
for (size_t i = 0; i < n; i++) {
    matrix[i] = (double *)malloc((n + 1) * sizeof(double));
    diff_matrix[i] = (double *)malloc((n + 1) * sizeof(double));
    multi_matrix[i] = (double *)malloc((n + 1) * sizeof(double));
}
```

diff_matrix zawiera elementy $n_{k,j,i} = M_{i,j} * m_{k,i}$. Na potrzeby implementacji $n_{k,j,i}$ zmieniłem na $n_{k,j}$ i nadpisuje ją dla każdego wiersza.
multi_matrix elementy $m_{k,i} = M_{k,i} / M_{i,i}$

Funkcje A, B, C

```
void A (int k,int i,double **matrix,double **multi_matrix){
        multi_matrix[k][i] = matrix[k][i] / matrix[i][i];

    }

void B(int i,int j,int k, double **matrix,double **multi_matrix,double **diff_matrix){
            diff_matrix[k][j] = matrix[i][j] *multi_matrix[k][i] ;

}

void C(int i,int j,int k, double **matrix,double **diff_matrix){
             matrix[k][j] -= diff_matrix[k][j];


}
```

Funkcje odpowiadają po kolei za operacje: $A_{i,k}, B_{i,j,k}, C_{i,j,k}$

Scheduler

```
void scheduler(int n, double **matrix, double **diff_matrix, double **multi_matrix) {

        for (int i = 0; i < n - 1; i++) {

            #pragma omp parallel for
            for (int k = i + 1; k < n; k++) {
                A(k, i, matrix, multi_matrix);
            }

            #pragma omp parallel for collapse(2)
            for (int j = i; j <= n; j++) {
                for (int k = i + 1; k < n; k++) {
                    B(i, j, k, matrix, multi_matrix, diff_matrix);
                }
            }

            #pragma omp parallel for collapse(2)
            for (int j = i; j <= n; j++) {
                for (int k = i + 1; k < n; k++) {
                    C(i, j, k, matrix, diff_matrix);
                }
            }
        }
    }
```

Dla każdej klasy Foaty, obliczam wartości przy pomocy omp parallel for.

Wyniki

Dla przykładu:

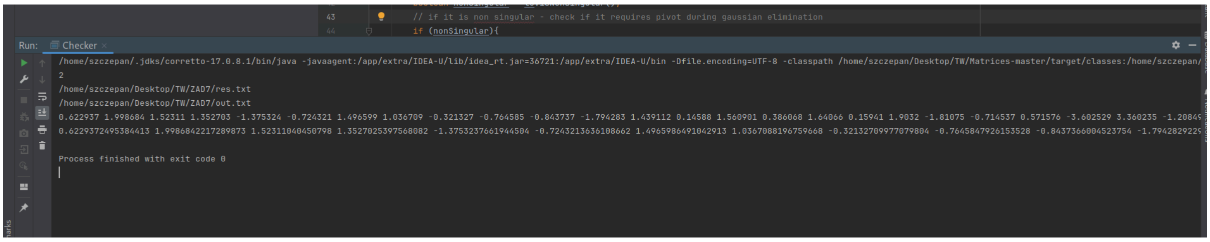
```
data.txt
15
0.3598381494 0.8409342858 0.2078457000 0.3076592230 0.5746280031 0.3193307736 0.3614672566 0.7279397259 0.9533114347 0.0125489458 0.3668514230 0.082205188
0.0990509280 0.8448001488 0.7335959037 0.4410333892 0.0004641690 0.2571903929 0.2827449953 0.1344851542 0.8379886466 0.9408241372 0.7638297249 0.775472664
0.0611367836 0.5240200388 0.3451626587 0.9028476472 0.9895839527 0.4061565322 0.5826884804 0.1881432569 0.8365774747 0.2832440536 0.0622547836 0.552720097
0.6823218116 0.4663162978 0.8145247654 0.1093526633 0.6058778448 0.1784634465 0.9460446959 0.9433581349 0.9184359435 0.7168164496 0.6029857273 0.478833164
0.9368129044 0.5912644209 0.9445518145 0.3562214748 0.6420745397 0.5789400685 0.5413422112 0.7382183675 0.1538579404 0.6128888873 0.7117459899 0.948382682
0.5352875036 0.1358311358 0.5173959241 0.3176994038 0.4072910492 0.2258247967 0.8745417488 0.9285354458 0.2594445028 0.4034480317 0.2240388131 0.624611312
0.4114219336 0.0139796539 0.7853549691 0.6315148523 0.8068024749 0.5601392847 0.3525633491 0.2650392302 0.0876001591 0.6359691166 0.2433253886 0.766014740
0.2785960288 0.3268870842 0.7827245496 0.2605054719 0.4557001634 0.2953058685 0.2070036651 0.5655408288 0.7126389005 0.2165884607 0.0632256706 0.109115234
0.5963010280 0.5848972130 0.7296366203 0.1246918985 0.5640114466 0.5862422241 0.9399262573 0.5094661313 0.2112823140 0.4058047461 0.9899567962 0.854449511
0.3812462110 0.2216356655 0.5895190368 0.6389691158 0.1072504053 0.0889349987 0.3579798574 0.6291786687 0.3718725101 0.6017271235 0.8620720106 0.557131776
0.0948736058 0.0646352080 0.0830413107 0.0143082636 0.4444321599 0.0114416364 0.1665537668 0.9500438661 0.7895594488 0.9186577436 0.0865250001 0.597267205
0.6646218739 0.8314134551 0.5496519350 0.9347766200 0.4076652435 0.8008724220 0.6403601923 0.8993992383 0.4330066054 0.0393853043 0.9755776891 0.535488538
0.7115397145 0.7280899996 0.8476419914 0.6123640034 0.9775477666 0.6008731913 0.7461419270 0.0064400154 0.7794512955 0.2574379281 0.1902886965 0.054847003
0.5973289158 0.2003489442 0.1237908044 0.2352716256 0.9280816961 0.3936953738 0.9808339366 0.5414606953 0.8976471080 0.5914889976 0.6838673279 0.096369416
0.9991857941 0.6751889463 0.5910245339 0.3281276648 0.9021790219 0.7498552792 0.5425374148 0.0792096039 0.7344354286 0.8225782968 0.1015977843 0.654152394
0.6384948161 0.8550308482 0.4348680819 0.0930752870 0.7562935026 0.1305411126 0.3408784191 0.3158685483 0.9779708263 0.9447345766 0.8344664526 0.030832693
```

Wynik:

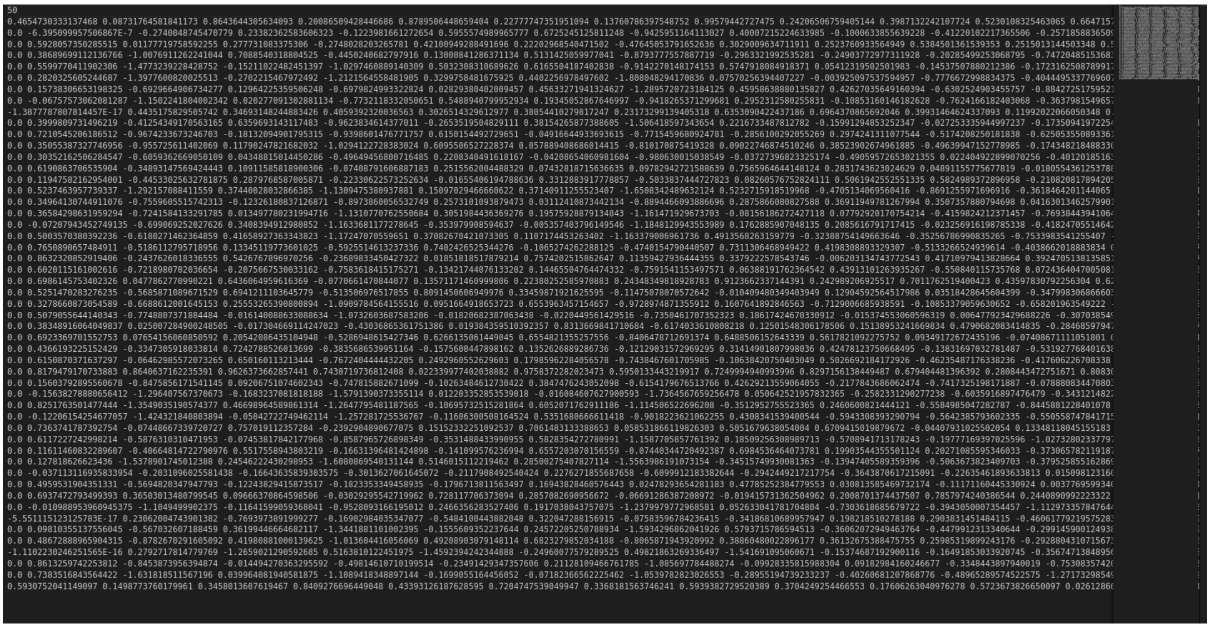
```
res.txt
15
1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
-0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 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
-0.553560 -0.050677 -0.347317 -1.654257 1.255114 -0.381452 -0.953928 -0.347704 -0.148746 0.646361 1.147145 -0.066972 0.603727 0.872925 0.630139
```

Sprawdzarka

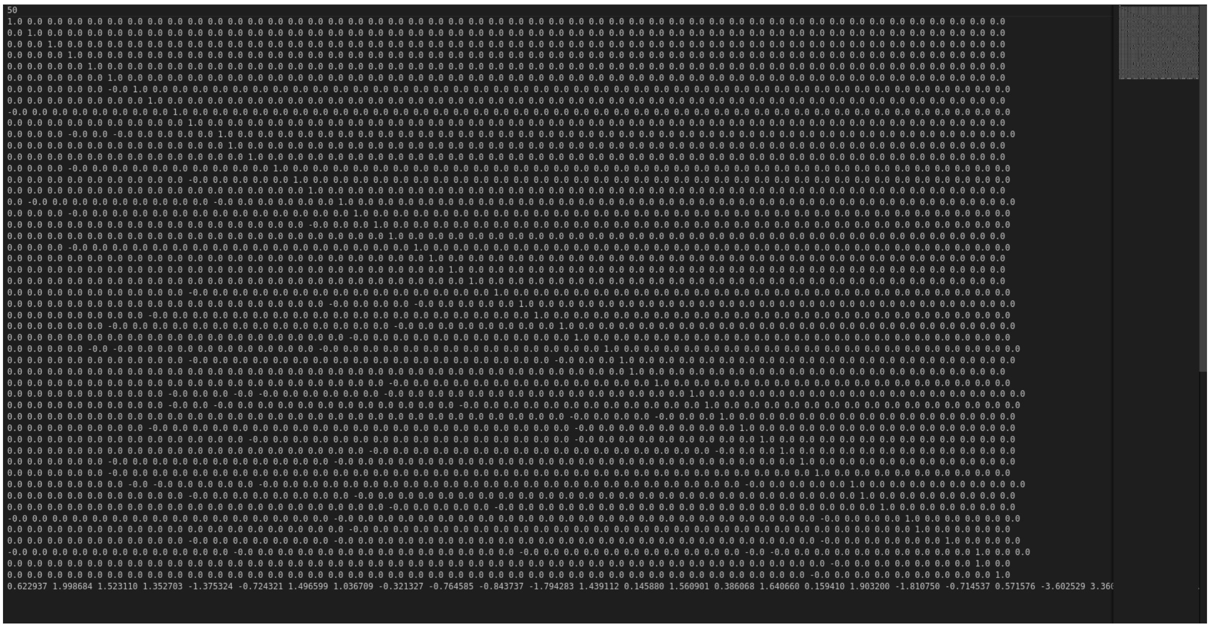
dla macierzy N=50



dane:



wynik:



Program graph.c

Program po koleji wypisuje wszystkie krawędzie z relacji D1,D2,D3,D4,D5 w celu wygenerowania grafu Diekerta i przy okazji sprawdzeniu poprawności zapisu relacji oraz krawędzi.