Document suport pentru tema 2 la PCLP1

1. Indexări în Matlab/Octave

Exemplu 1:

```
>> A
A =

  41  18   7  29  42  40  16
  22  16   6  33  30  29  13
  46  42  13   3   2  32  44
  44  22   2  21  25  43   9
  45  48  27   1  43  32  10
  46  14  34  34  17   9  12
  29  20   4  19  19  11  25

>> A([1, 2, 3], [4, 5, 6])
ans =

  29  42  40
  33  30  29
   3   2  32

>>
```

Primul exemplu este un crop simplu, efectuat după 3 linii și 3 coloane consecutive (reprezintă operația de redimensionare fără a permuta liniile sau coloanele matricei originale).

Primul element din noua matrice este elementul din poziția (1,4) din vechea matrice, al doilea element de pe prima linie din noua matrice este cel din poziția (1,5) din a vechea matrice. Această regulă este valabilă pentru toate cazurile de redimensionare (elementele noii matrice sunt preluate din vechea matrice, iar noii indecși sunt dați prin intermediul celor doi vectori). Am dat indecși consecutivi pentru a explica pas cu pas crop-ul general (+indexare) din Matlab/Octave prin pașii săi componenți: crop (de linii nu neapărat consecutive) + permutare de linii și respectiv coloane, conform ordinii date în vectorii de indecși.

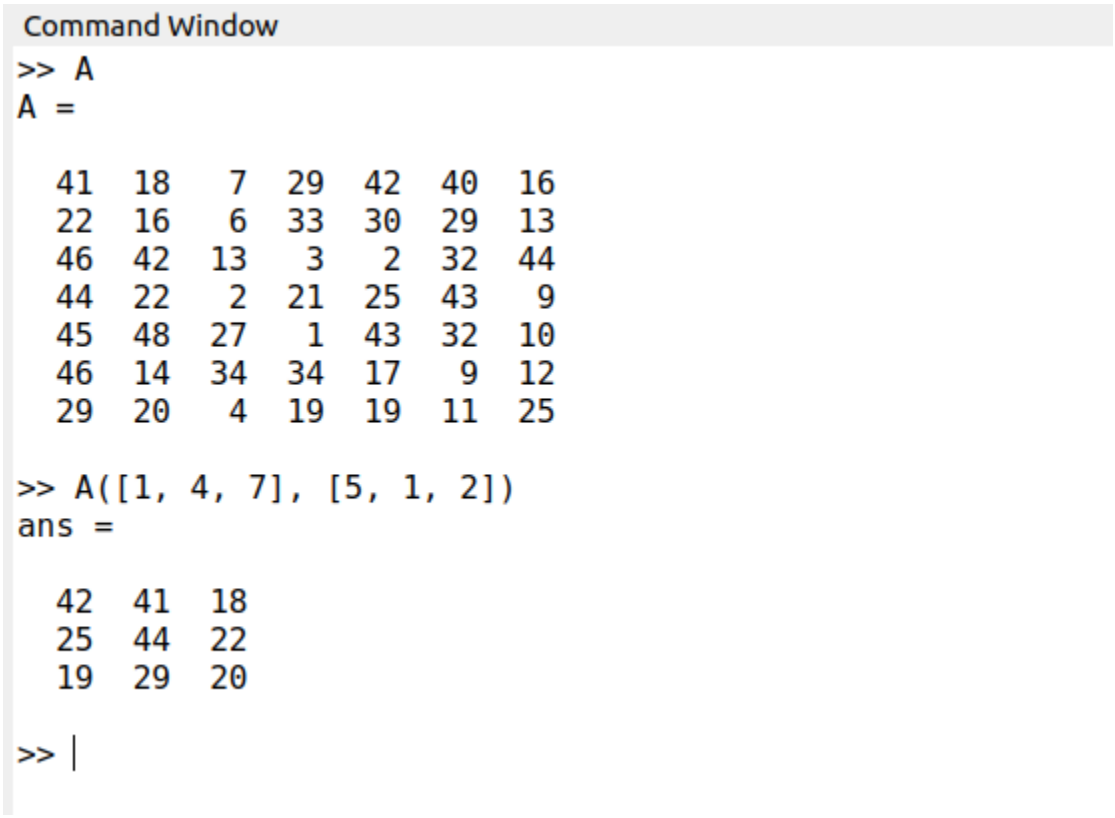
Exemplu 2:

```
Command Window
>> A
A =

  41  18   7  29  42  40  16
  22  16   6  33  30  29  13
  46  42  13   3   2  32  44
  44  22   2  21  25  43   9
  45  48  27   1  43  32  10
  46  14  34  34  17   9  12
  29  20   4  19  19  11  25

>> A([1 4 5], [2 3 7])
ans =

  18   7  16
  22   2   9
  48  27  10

>>
```

Al doilea exemplu reprezintă un crop efectuat după 3 linii și 3 coloane, nu neapărat consecutive (de asemenea, fără a permuta liniile și coloanele originale).

Primul element din noua matrice este elementul din vechea matrice de la poziția (1,2), cel de-al doilea element de pe prima linie din matrice este elementul din poziția (1,3) din vechea matrice etc. Acesta reprezintă un exemplu mai complicat (crop de linii nu neapărat consecutive, însă în ordinea în care ele apar în matrice). Această operație se aseamănă cu operația de selectare a unui minor din algebra liniară (toate elementele aflate la intersecția dintre o mulțime de coloane și o mulțime de linii, doar că aici „minorul” (sau, mai bine zis, matricea asociată minorului) nu este neapărat o matrice pătratică).

Exemplu 3:



Al treilea exemplu este de fapt operația generalizată de redimensionare în Matlab/Octave, în care putem observa toate operațiile componente care au loc: crop (de coloane nu neapărat consecutive) și permutare de coloane.

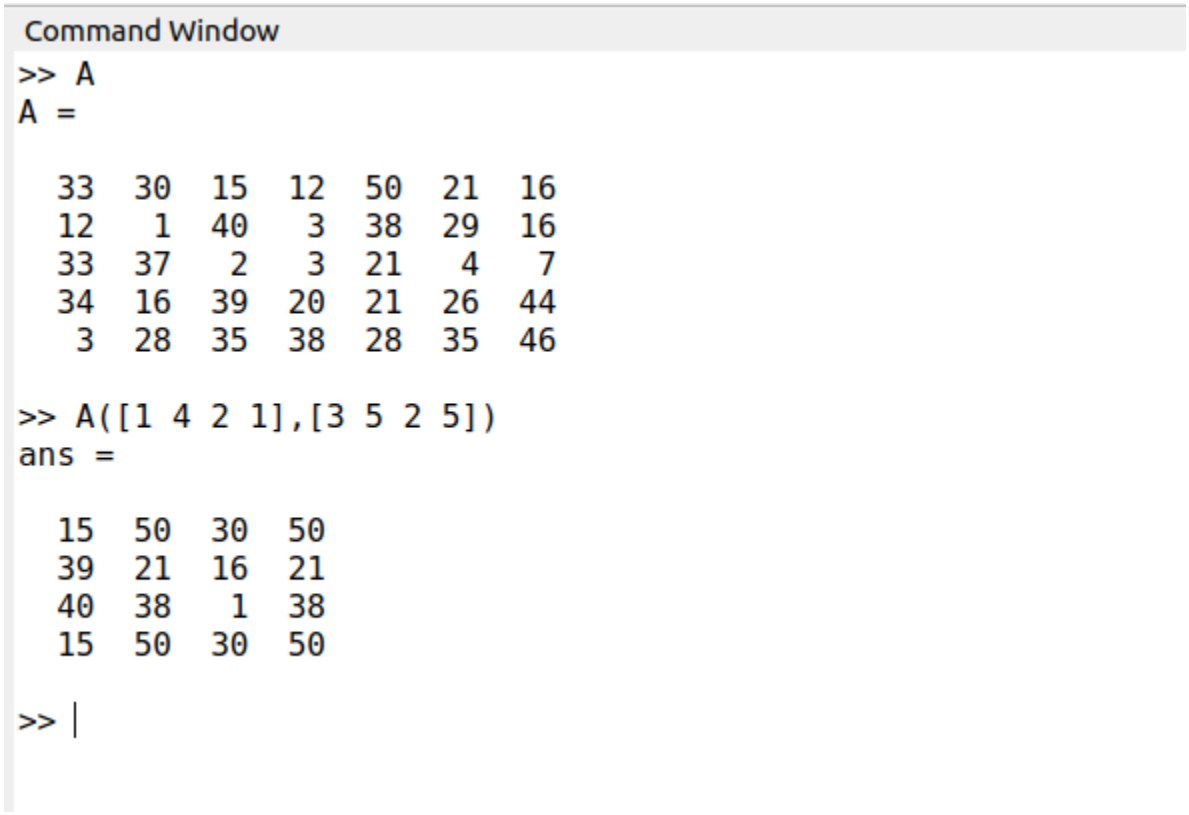
Observații:

- Din fericire, operația de redimensionare poate fi făcută în limbajul C folosindu-ne de valorile indecșilor deja primiți de la STDIN (**hint!**). Cu alte cuvinte, implementarea voastră nu ar trebui să țină cont dacă vectorul de indecși pentru linii/coloane este sortat sau nu.

- Aveți în vedere faptul că indexarea în exemplele de mai sus a fost făcută de la 1, specific Matlab/Octave. În cadrul testelor veți avea numai exemple de indexări de la 0 (adaptat limbajului C), așa că nu este nevoie să țineți cont de exemplele de mai sus ca model pentru implementare, ci doar ca să înțelegeți mai bine mecanismul de crop/permutare.

- Pentru a înțelege exact cum are loc operația pe mai multe exemple, vă încurajăm să vă construiți propriile exemple folosind https://octave-online.net/.

- Octave permite și situații în care indecșii se pot repeta, după cum este evident în următoarea situație:



Astfel, implementarea pentru cazul general ar trebui să țină cont și de acest aspect.

2. Exemplu pentru algoritmul lui Strassen

După cum se poate observa (fie este un fapt cunoscut, fie din implementarea înmulțirii de matrice), pentru înmulțirea a două matrice pătratice de ordin n sunt necesare aproximativ n^3operații aritmetice. În limbaj de complexitate, se spune că o astfel de operație se efectuează în complexitate $O(n^3)$ (notația de complexitate va fi discutată pe larg anul viitor, la disciplinele Analiza Algoritmilor și Proiectarea Algoritmilor). Pentru matrice foarte mari (de dimensiune de ordinul peste10^4) nu mai este fezabilă înmulțirea clasică, de aceea se încearcă folosirea unor metode alternative, care să reducă complexitatea. Una dintre aceste metode este algoritmul lui Strassen de înmulțire a matricelor, care se bazează pe următoarea idee (pentru simplitate, presupunem că matricele sunt pătratice și au dimensiunea 2^n, situație care va fi garantată în cadrul testelor) :

- Se partiționează fiecare matrice în 4 blocuri, cu dimensiunea $m / 2$, unde m este dimensiunea matricelor, considerată număr par.
- O abordare ca cea din figura următoare nu ar fi de folos, deoarece am obține 8 înmulțiri de matrice de dimensiune $m / 2$, idee care ar duce la aceeași complexitate.

$$A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}$$

$$B = \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}$$

$$C = A \cdot B = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \cdot \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix} = \begin{bmatrix} A_1 \cdot B_1 + A_2 \cdot B_2 & A_1 \cdot B_2 + A_2 \cdot B_4 \\ A_3 \cdot B_1 + A_4 \cdot B_3 & A_3 \cdot B_2 + A_4 \cdot B_4 \end{bmatrix}$$

- Principala idee a algoritmului lui Strassen este să definească următoarele matrice auxiliare:

$$
\begin{aligned}
M_1 &= (A_1 + A_4) \cdot (B_1 + B_4) \\
M_2 &= (A_3 + A_4) \cdot B_1 \\
M_3 &= A_1 \cdot (B_2 - B_4) \\
M_4 &= A_4 \cdot (B_3 - B_1) \\
M_5 &= (A_1 + A_2) \cdot B_4 \\
M_6 &= (A_3 - A_1) \cdot (B_1 + B_2) \\
M_7 &= (A_2 - A_4) \cdot (B_3 + B_4)
\end{aligned}
$$

- Astfel, matricea rezultat poate fi scrisă ca:

$$
C = \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 - M_2 + M_3 + M_6 \end{bmatrix}
$$

- Avantajul acestei forme este că sunt necesare doar 7 înmulțiri de matrice de dimensiune $m / 2$. Astfel, se poate demonstra că, pentru o înmulțire de 2 matrice de ordin n, algoritmul nostru face aproximativ $n^{log_2 7}$ pași (adică o complexitate mai bună decât înmulțirea clasică).
- Prin aplicarea recursivă a algoritmului, fiecare dintre cele 7 matrice auxiliare reprezintă tot un produs de matrice, care poate fi soluționat recursiv aplicând din nou algoritmul lui Strassen. Recurența merge până când matricele de înmulțit sunt numere.

Exemplu de aplicare al algoritmului lui Strassen

Fie matricele de înmulțit A și B din exemplul următor:

$$A = \begin{bmatrix} -4 & 1 & 5 & -2 \\ -1 & 2 & 1 & 2 \\ 3 & 0 & 5 & -1 \\ 2 & 1 & 1 & 4 \end{bmatrix}$$

$$B = \begin{bmatrix} 2 & -2 & -4 & -1 \\ 0 & 2 & 0 & 3 \\ 1 & 3 & -3 & -2 \\ 1 & -4 & -5 & -2 \end{bmatrix}$$

- Etapa I: Partiționăm matricele pe blocuri:

Partiționarea pentru A:

$$A_1 = \begin{bmatrix} -4 & 1 \\ -1 & 2 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 5 & -2 \\ 1 & 2 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} 5 & -1 \\ 1 & 4 \end{bmatrix}$$

Partiționarea pentru B:

$$B_1 = \begin{bmatrix} 2 & -2 \\ 0 & 2 \end{bmatrix}$$

$$B_2 = \begin{bmatrix} -4 & -1 \\ 0 & 3 \end{bmatrix}$$

$$B_3 = \begin{bmatrix} 1 & 3 \\ 1 & -4 \end{bmatrix}$$

$$B_4 = \begin{bmatrix} -3 & -2 \\ -5 & -2 \end{bmatrix}$$

- Se calculează cele 7 matrice auxiliare, folosind formulele prezentate anterior. Acestea nu se calculează folosind înmulțirea clasică, ci prin **aplicarea recursivă a algoritmului lui Strassen** (în cazul nostru, aplicarea încă o dată a algoritmului ar fi fost banală - matricele degenerau în numere, de aceea am scris direct rezultatul acestor înmulțiri de matrice). Vezi următoarea pagină.
- După calcularea matricelor de forma M_k, se pot calcula blocurile matricei C și deci matricea C per total.

$$M_1=(A_1+A_4)\cdot(B_1+B_4)=\begin{bmatrix}1 & 0\\ 0 & 6\end{bmatrix}\cdot\begin{bmatrix}-1 & -4\\ -5 & 0\end{bmatrix}=\begin{bmatrix}-1 & -4\\ -30 & 0\end{bmatrix}$$

$$M_2=(A_3+A_4)\cdot B_1=\begin{bmatrix}8 & -1\\ 3 & 5\end{bmatrix}\cdot\begin{bmatrix}2 & -2\\ 0 & 2\end{bmatrix}=\begin{bmatrix}16 & -18\\ 6 & 4\end{bmatrix}$$

$$M_3=A_1\cdot(B_2-B_4)=\begin{bmatrix}-4 & 1\\ -1 & 2\end{bmatrix}\cdot\begin{bmatrix}-1 & 1\\ 5 & 5\end{bmatrix}=\begin{bmatrix}9 & 1\\ 11 & 9\end{bmatrix}$$

$$M_4=A_4\cdot(B_3-B_1)=\begin{bmatrix}5 & -1\\ 1 & 4\end{bmatrix}\cdot\begin{bmatrix}-1 & 5\\ 1 & -6\end{bmatrix}=\begin{bmatrix}-6 & 31\\ 3 & -5\end{bmatrix}$$

$$M_5=(A_1+A_2)\cdot B_4=\begin{bmatrix}1 & -1\\ 0 & 4\end{bmatrix}\cdot\begin{bmatrix}-3 & -2\\ -5 & -2\end{bmatrix}=\begin{bmatrix}2 & 0\\ -20 & -8\end{bmatrix}$$

$$M_6=(A_3-A_1)\cdot(B_1+B_2)=\begin{bmatrix}7 & -1\\ 3 & -1\end{bmatrix}\cdot\begin{bmatrix}-2 & -3\\ 0 & 5\end{bmatrix}=\begin{bmatrix}-14 & -26\\ -6 & -14\end{bmatrix}$$

$$M_7=(A_2-A_4)\cdot(B_3+B_4)=\begin{bmatrix}0 & -1\\ 0 & -2\end{bmatrix}\cdot\begin{bmatrix}-2 & 1\\ -4 & -6\end{bmatrix}=\begin{bmatrix}4 & 6\\ 8 & 12\end{bmatrix}$$

- Am calculat cele 7 matrice ajutătoare, iar acum folosim expresiile de mai sus pentru a calcula blocurile matricei (de forma C_k).

$$C_1 = M_1 + M_4 - M_5 + M_7 = \begin{bmatrix} -5 & 33 \\ 1 & 1 \end{bmatrix}$$

$$C_2 = M_3 + M_5 = \begin{bmatrix} 11 & 1 \\ -9 & 1 \end{bmatrix}$$

$$C_3 = M_2 + M_4 = \begin{bmatrix} 10 & 13 \\ 9 & -15 \end{bmatrix}$$

$$C_4 = M_1 - M_2 + M_3 + M_6 = \begin{bmatrix} -22 & -11 \\ -31 & -9 \end{bmatrix}$$

$$C = \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} -5 & 33 & 11 & 1 \\ 1 & 1 & -9 & 1 \\ 10 & 13 & -22 & -11 \\ 9 & -15 & -31 & -9 \end{bmatrix}$$

- Se poate verifica faptul că $C = A \cdot B$.