Πολύ καλή προσπάθεια. Εξαιρετική η ανάλυση για το 7. Δεν υπάρχουν χρονομετρήσεις και αξιολόγηση για το 8 στην αναφορά.

Εργασία 2

Χαράλαμπος Αναστασίου

Οκτώβριος 2024

1 Εισαγωγή

Υλοποιήθηκαν όλα τα Ερωτήματα εκτός από μέρος του υποερωτήματος 2.) του Ερωτήματος 9 που βασίζεται στις ενότητες GvL 1.3.7-8 (λόγω έλλειψης χρόνου θα απαντηθεί στο άμεσο μέλλον).

2 Απαντήσεις Ερωτημάτων

2.1 Ερώτημα 1

Αν $x = [1 : 8]$ και $P_{2,4}$ είναι το μητρώο τέλειας αναδιάταξης mod 2, τότε να γράψετε το διάνυσμα $P_{2,4} \cdot x$.

ΑΠΑΝΤΗΣΗ: Το x είναι ένα διάνυσμα-στήλη μεγέθους 8×1. Το μητρώο τέλειας αναδιάταξης $P_{q,r} = P_{2,4}$ δίνεται από τον ακόλουθο τύπο, ο οποίος έχει αντληθεί από τις διαφάνειες μαθήματος:

$$P_{2,4} = I_8([1:4:8, 2:4:8, 3:4:8, 4:4:8])$$

Προσοχή: Αν I8=eye(8) όπως θεωρώ ότι υποθέτεις, σου λείπει ο 2ος δείκτης! Το σωστό είναι I8 ([1 : 4 : 8, 2 : 4 : 8, 3 : 4 : 8, 4 : 4 : 8],:). Βλ και παρακάτω στο Ερ5 που το έχεις επιλέξει σωστά.

Το $P_{2,4}$ έχει μέγεθος 8×8.
Επομένως, το γινόμενο $P_{2,4} \cdot x$ είναι:

όπως και πριν

$$P_{2,4} \cdot x = I_8([1,5,2,6,3,7,4,8]) \cdot x = \begin{bmatrix} x(1:4:8) \\ x(2:4:8) \\ x(3:4:8) \\ x(4:4:8) \end{bmatrix}$$

που είναι το τελικό διάνυσμα-στήλη μεγέθους 8×1.

2.2 Ερώτημα 2

(Σωστό/Λάθος) Αν $P_{2,4}$ είναι το μητρώο τέλειας αναδιάταξης mod 2, τότε ισχύει:

$$P_{2,4} \cdot P_{2,4}^{\top} = I_8$$

ΑΠΑΝΤΗΣΗ: Σωστό.

Γιατί;

2.3 Ερώτημα 3

(GvL A1.3.4) Έστω το μητρώο

$$A = \begin{pmatrix} 0 & B \\ B^{\top} & 0 \end{pmatrix}$$

όπου το B είναι άνω διδιαγώνιο. Να περιγράψετε τη δομή του $T = PAP^{\top}$, όπου η $P = P_{2,n}$ είναι η μετάθεση τέλειας αναδιάταξης mod 2.

ΑΠΑΝΤΗΣΗ:

```
m = 4;


% Random values for the main diagonal
main_diag = randi(10, 1, m);

% Random values for the first upper diagonal
upper_diag = randi(10, 1, m-1);

% Create the upper bidiagonal matrix B
B = diag(main_diag) + diag(upper_diag, 1);

% Create the matrix A
A = [zeros(m), B; B', zeros(m)];

% Display the final square matrix A
disp('The matrix A is:')
disp(A);

n = size(A,1); % size of A

% Create the identity matrix I
I = eye(n);

% Define the row permutation of I
r = 3;       % You can change this value to experiment
```

```
% r=4;
% r=5;
perm = [];
for i = 1:r
    indices = i:r:n;
    perm = [perm, indices];
end

% Row permutation of I
P = I(perm, :);

T = P * A * P';
D = T';

% I notice that for different values of r,
% the matrix T remains SYMMETRIC !!!
```

Σχόλια κώδικα: Στον παραπάνω κώδικα για τη δημιουργία της μετάθεσης τέλειας αναδιάταξης χρησιμοποιήθηκε ένας βρόγχος for ο οποίος δημιουργεί την μετάθεση (perm) με βάση τον τύπο από τις διαφάνειες του μαθήματος. ΒΡΟΧΟΣ !!!!

2.4 Ερώτημα 4

(GvL A1.3.7 - προσοχή: στο βιβλίο εκ παραδρομής, αντί του συμβόλου της αναστροφής, γράφτηκε $\otimes$). Να επαληθεύσετε ότι, αν $x \in \mathbb{R}^m$, $y \in \mathbb{R}^n$, τότε ισχύει:

$$y \otimes x = \mathrm{vec}(xy^\top).$$

ΑΠΑΝΤΗΣΗ:

- Έχουμε:

$$y \otimes x = \begin{bmatrix} y_1 x \\ y_2 x \\ \vdots \\ y_n x \end{bmatrix} \quad \Rightarrow \quad \text{διάνυσμα-στήλη } mn \text{ στοιχείων}$$

- Επίσης:

$$(xy^\top)_{ij} = x_i y_j \quad \Rightarrow \quad \text{μητρώο } m \times n$$

και

$$\operatorname{vec}(xy^\top) = \begin{pmatrix} x_1y_1 \\ x_2y_1 \\ \vdots \\ x_my_1 \\ x_1y_2 \\ x_2y_2 \\ \vdots \\ x_my_n \end{pmatrix}$$

Είναι εμφανές ότι το $\operatorname{vec}(xy^\top)$ είναι ίσο με το $y \otimes x$, καθώς στους βαθμωτούς ισχύει η αντιμεταθετικότητα ($x_1y_1 = y_1x_1$ κ.ο.κ).

Θα μπορούσες να το διαμορφώσεις πιο τυπικά και όχι "είναι προφανές"

2.5 Ερώτημα 5

Να αποδείξετε αυτό που αναφέρεται στο σύγγραμμα (εξίσωση GvL 1.3.5) ότι ενώ για γενικά μητρώα $B \in \mathbb{R}^{m_1 \times m_2}$, $C \in \mathbb{R}^{n_1 \times n_2}$ ισχύει ότι $B \otimes C \neq C \otimes B$ (το αναφέραμε και δείξαμε παράδειγμα στη Διάλεξη 4), υπάρχουν μητρώα μετάθεσης P, Q τέτοια ώστε $P(B \otimes C)Q = C \otimes B$. Να επιβεβαιώστε ότι τα μητρώα μετάθεσης είναι αυτά που αναφέρονται στο βιβλίο.

ΑΠΑΝΤΗΣΗ: Στο βιβλίο αναφέρονται τα P και Q ως $P = P_{m_1,m_2}$ και $Q = P_{n_1,n_2}$ τέτοια ώστε

$$P(B \otimes C)Q^T = C \otimes B.$$

Έστω το πλήθος γραμμών $M = m_1n_1$ και το πλήθος στηλών $N = m_2n_2$ του γινομένου $B \otimes C$.

Έχουμε ότι:

$$\begin{aligned} P(B \otimes C)Q^T &= P_{m_1,m_2}(B \otimes C)P^T_{n_1,n_2} \\ &= I_M\left([(1 : m_2 : M), (2 : m_2 : M), \ldots, (m_2 : m_2 : M)], :\right) \\ &\quad (B \otimes C) \\ &\quad I_N\left([(1 : n_1 : N), (2 : n_1 : N), \ldots, (n_1 : n_1 : N)], :\right) \end{aligned}$$

Ο όρος με το I_M είναι μεγέθους $M \times M$, ο όρος $B \otimes C$ είναι $M \times N$ και ο όρος με το I_N είναι μεγέθους $N \times N$. Επομένως το αποτέλεσμα θα είναι ένα $M \times N$ μητρώο.

Αν κοιτάξουμε προσεκτικά τον παραπάνω τύπο και συγκρίνουμε με το αναλυτικό παράδειγμα με το μητρώο A στην αρχή της σελ. 27 του συγγράμματος, παρατηρούμε πως στην πραγματικότητα το αποτέλεσμα πρόκειται για μεταθέσεις γραμμών του $B \otimes C$ ανά m_2 και μεταθέσεις στηλών ανά n_1. Αυτές οι ενέργειες οδηγούν στο γινόμενο $C \otimes B$.

2.6 Ερώτημα 6

(GvL A1.3.8 - προσοχή στο βιβλίο εκ παραδρομής αντί του συμβόλου της αναστροφής, γράφτηκε $\otimes$.) Να δείξετε ότι αν $B \in \mathbb{R}^{p\times p}$ και $C \in \mathbb{R}^{q\times q}$ και

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix},$$

όπου $x_i \in \mathbb{R}^q$, τότε

$$x^T(B \otimes C)x = \sum_{i=1}^{p}\sum_{j=1}^{p} \beta_{ij}(x_i^T C x_j)$$

Ακολουθούν τα μεγέθη για τον κάθε όρο στην παραπάνω έκφραση:

$$\begin{aligned} x^T &\rightarrow 1 \times pq, \\ (B \otimes C) &\rightarrow pq \times pq, \\ x &\rightarrow pq \times 1. \end{aligned}$$

Άρα το τελικό αποτέλεσμα είναι βαθμωτός.

Εάν εκτελέσουμε πρώτα το γινόμενο $(B \otimes C)x$, τότε έχουμε:

$$\begin{bmatrix} \beta_{11}C & \cdots & \beta_{1p}C \\ \beta_{21}C & \cdots & \beta_{2p}C \\ \vdots & \ddots & \vdots \\ \beta_{p1}C & \cdots & \beta_{pp}C \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix}$$

όπου το πρώτο μητρώο είναι ένα μπλοκ μητρώο που αποτελείται από τα στοιχεία $\beta_{ij}C$, και πολλαπλασιάζεται με το διάνυσμα-στήλη x.

Η εξίσωση που περιγράφει τον πολλαπλασιασμό αυτόν είναι:

$$\sum_{i=1}^{p}\sum_{j=1}^{p}(\beta_{ij}C)x_j$$

Αυτός ο πολλαπλασιασμός δίνει ένα διάνυσμα-στήλη διαστάσεων $pq \times 1$, το οποίο στη συνέχεια πολλαπλασιάζεται με το

$$x^T = \begin{bmatrix} x_1^T & x_2^T & \cdots & x_p^T \end{bmatrix}$$

δίνοντας την τελική εξίσωση:

$$\sum_{i=1}^{p}\sum_{j=1}^{p} x_i^T(\beta_{ij}C)x_j$$

2.7 Ερώτημα 7

Δίνονται $A \in \mathbb{R}^{n\times n}$, $u, v \in \mathbb{R}^n$ διανύσματα (στήλες) και k ακέραιος τέτοιος ώστε $k \leq n$. Να δείξετε ότι υπάρχουν μητρώα $U, V \in \mathbb{R}^{n\times k}$ ώστε να ισχύει

$$(A + uv^T)^k = A^k + UV^T$$

και να δείξετε πώς να το υπολογίσετε αποδοτικά. Ποιό είναι το αντίστοιχο Ω;

ΑΠΑΝΤΗΣΗ:

Δίνονται:

- Ένα μητρώο $A \in \mathbb{R}^{n\times n}$.
- Δύο διανύσματα $u, v \in \mathbb{R}^n$.
- Ένας ακέραιος $k \leq n$.

Ο στόχος είναι να δείξουμε ότι υπάρχουν μητρώα $U, V \in \mathbb{R}^{n\times k}$ έτσι ώστε:

$$(A + uv^T)^k = A^k + UV^T,$$

και να υπολογίσουμε την έκφραση αυτή με αποδοτικό τρόπο.

Υπολογισμός των Πρώτων Δυνάμεων

Θα υπολογίσουμε αρχικά τις πρώτες δυνάμεις για k=1,2,3 ώστε να φτάσουμε επαγωγικά σε κάποιον τελικό τύπο που ενδεχομένως να αποκαλύψει τρόπους αποδοτικού υπολογισμού.

Για $k = 1$, έχουμε:

$$(A + uv^T)^1 = A + uv^T.$$

Αυτή είναι απλώς η αρχική μορφή του $A + uv^T$, χωρίς πρόσθετους όρους.

Για $k = 2$:

Χρησιμοποιώντας τη σχέση:

$$(A + uv^T)^2 = (A + uv^T)(A + uv^T),$$

αναπτύσσουμε το γινόμενο:

$$(A + uv^T)^2 = A \cdot A + A \cdot uv^T + uv^T \cdot A + uv^T \cdot uv^T.$$

Αναλυτικά, κάθε όρος είναι:

$$\begin{aligned} A \cdot A &= A^2, \\ A \cdot uv^T &= Au \cdot v^T, \\ uv^T \cdot A &= u \cdot (v^T A), \\ uv^T \cdot uv^T &= u \cdot (v^T u) \cdot v^T = (v^T u) \cdot (uv^T), \end{aligned}$$

που είναι το μητρώο uv^T scaled κατά τον βαθμωτό $v^T u$. Τελικά, για $k = 2$, προκύπτουν οι όροι:

$$(A + uv^T)^2 = A^2 + \boxed{Au \cdot v^T + u \cdot (v^T A) + (v^T u) \cdot (uv^T).}$$

Αυτό είναι μητρώο τάξης 2 (δεν το δείχνεις)

Για $k = 3$, έχουμε:

Χρησιμοποιώντας ομοίως τη σχέση:

$$(A + uv^T)^3 = (A + uv^T) \cdot (A + uv^T)^2.$$

Αντικαθιστώντας το $(A + uv^T)^2$ από πριν, έχουμε:

$$(A + uv^T)^3 = (A + uv^T) \cdot (A^2 + Au \cdot v^T + u \cdot (v^T A) + (v^T u) \cdot (uv^T)).$$

Αναπτύσσοντας το γινόμενο αυτό, κάθε όρος περιέχει μια διαδοχική εφαρμογή του A στους όρους u και v. Προκύπτουν τελικά όροι της μορφής $A^i u \cdot (A^j v)^T$, όπως:

$$\begin{aligned} A^3&, \\ A^2 u \cdot v^T&, \\ Au \cdot (v^T A)&, \\ u \cdot (v^T A^2)&. \end{aligned}$$

Εξαιρετική ανάλυση

Επαγωγικά λοιπόν, κάθε επόμενη δύναμη $(A + uv^T)^k$ θα περιέχει συνδυασμούς όρων της μορφής $A^i u \cdot (A^j v)^T$, όπου οι δείκτες i και j αντιστοιχούν σε διαδοχικές εφαρμογές του A ή του A^T στα διανύσματα u και v, αντίστοιχα. Αυτό είναι κρίσιμο για την κατασκευή των μητρώων U και V όπως εξηγείται στη συνέχεια.

Για να συλλάβουμε όλους τους όρους που εμφανίζονται στις δυνάμεις του $A + uv^T$ μέχρι την τάξη k, ορίζουμε τα μητρώα U και V ως εξής:

- **Ορισμός του U:**

$$U = [u, Au, A^2u, \ldots, A^{k-1}u] \in \mathbb{R}^{n \times k}.$$

Κάθε στήλη του U είναι της μορφής $A^i u$, όπου $i = 0, 1, \ldots, k-1$, και αντιστοιχεί σε κάθε διαδοχική εφαρμογή του A πάνω στο u.

- **Ορισμός του V:**

$$V = [v, A^T v, (A^2)^T v, \ldots, (A^{k-1})^T v] \in \mathbb{R}^{n \times k}.$$

Κάθε στήλη του V είναι της μορφής $(A^j)^T v$, όπου $j = 0, 1, \ldots, k-1$, και αντιστοιχεί σε κάθε διαδοχική εφαρμογή του A^T πάνω στο v.

Έτσι, το γινόμενο UV^T περιέχει όλους τους συνδυασμούς των όρων $A^i u \cdot (A^j v)^T$ που εμφανίζονται στις διαδοχικές δυνάμεις του $(A + uv^T)^k$, καλύπτοντας όλους τους απαραίτητους όρους που προκύπτουν από την σχέση που ανέλυσα παραπάνω με επαγωγή.

Το γινόμενο UV^T παράγει ακριβώς αυτούς τους όρους, ενώ ο όρος A^k διατηρεί τη βασική δύναμη του A χωρίς την επίδραση του uv^T.

Σωστά αλλά με διαφορετική σειρά.
π.χ. δεν ισχύει για k=2 όπως το έχεις γράψει.

Επομένως, έχουμε:

$$(A + uv^T)^k = A^k + UV^T.$$

Η κατασκευή των μητρώων U και V επιτρέπει τον αποδοτικό υπολογισμό της έκφρασης με πολυπλοκότητα $\mathcal{O}(kn^2)$, καθώς απαιτεί μόνο k πολλαπλασιασμούς με το μητρώο A και το ανάστροφό του, σε αντίθεση με την πλήρη εξαντλητική επέκταση του $(A + uv^T)^k$, η οποία θα ήταν πολύ πιο κοστοβόρα.

Συμπέρασμα: Απεδείχθη επομένως ότι υπάρχουν

παραπάνω, ώστε:

$$(A + uv^T)^k = A^k + UV$$

με αποδοτικό τρόπο υπολογισμού που εκμεταλλεύ

στο A, ώστε να προκύψει ένας αναδρομικός τύπος

αποτελέσματος ο οποίος μας γλιτώνει από περίπλοκ

ΑΠΑΝΤΗΣΗ: Εξετάζουμε τους όρους για $k = 2, 3$. Προσοχή - η γνωστή έκφραση για το $(a+b)^k$ δεν μπορεί να χρησιμοποιηθεί καθώς δεν ισχύει η μεταθετική ιδιότητα για τον πολλαπλασιασμό μητρώων, δηλ. $Auv^\top \neq vv^\top A$.

Αν $k = 2$ και θέσουμε E_2 το μητρώο αναστροφής, δηλ. το αντιταυτοτικό μητρώο $E_2 = [0,1;1,0]$ θα έχουμε:

$$\begin{aligned}(A + uv^\top)^2 &= A^2 + Auv^\top + uv^\top A + (v^\top u)uv^\top \\ &= A^2 + \\ &= A^2 + [u, Au][(A^\top + vu^\top)v, v]^\top\end{aligned}$$

Ισχυριζόμαστε ότι

$$\begin{aligned}(A + uv^\top)^k &= A^k + U_k V_k^\top, \text{ όπου} \\ U_k &= [u, Au, ..., A^{k-1}u]E_k, V_k = [v, (A^\top + vu^\top)v, ..., (A^\top + vu^\top)^{k-1}v]\end{aligned}$$

Είναι προφανώς σωστό αν $k = 1$ και έστω ότι ισχύει για $k = 2, ..., s$. Τότε

$$\begin{aligned}(A + uv^\top)^{s+1} &= (A + uv^\top)(A + uv^\top)^s = A(A^s + U_s V_s^\top) + uv^\top (A + uv^\top)^s \\ &= A^{s+1} + A[u, Au, ..., A^{s-1}u]E_s V_s^\top + uv^\top (A + uv^\top)^s\end{aligned}$$

Επικεντρωνόμενοι στους τελευταίους 4 όρους:

$$\begin{aligned}&= [Au, A^2u, ..., A^s u]E_s V^\top + uv^\top (A + uv^\top)^s \\ &= [u, Au, ..., A^s u]E_{s+1}[v, (A^\top + vu^\top)v, ..., (A^\top + vu^\top)^s v]^\top \\ &= U_{s+1}V_{s+1}^\top, \text{ αποδεικνύοντας το ζητούμενο.}\end{aligned}$$

2.8 Ερώτημα 8

Δίνεται

$$A = \begin{bmatrix} I_n & xy^T \\ yx^T & -I_n \end{bmatrix}$$

όπου τα $x, y \in \mathbb{R}^n$ είναι διανύσματα στήλες και I_n είναι το ταυτοτικό μητρώο συμβατού μεγέθους (πρόκειται για ειδική περίπτωση Χαμιλτονιανού μητρώου, βλ. GvL ενότητα 1.3.10). Έστω επίσης τυχαίο $B \in \mathbb{R}^{n \times m}$.

(α) Να κατασκευάσετε συνάρτηση MATLAB η οποία δοθέντων των x, y κατασκευάζει το A. Η συνάρτηση να λέγεται `HamIbuild(x,y)` όπου x, y είναι ισομεγέθη διανύσματα (στήλες).

(β) Για $n = 2^{[6:12]}$ να εκτελέσετε τις εντολές:

```
x = rand(n, 1);
y = rand(n, 1);
I = eye(n);
B = rand(n, 1);
A = HamIbuild(x, y);
C = mtimes(A, B)
```

και να χρονομετρήσετε τα runtimes της τελευταίας εντολής (πολλαπλασιασμού) με αξιόπιστο τρόπο.

(γ) Να υλοποιήσετε εξειδικευμένο αλγόριθμο πολλαπλασιασμού μητρώων σαν και το παραπάνω με μητρώα $2^n \times s$ όπου η δεύτερη διάσταση s είναι πολύ μικρότερη του n. Η συνάρτηση να λέγεται `HAMM(x,y,B)`. Να συγκρίνετε τα runtimes της με τα αποτελέσματα του (β). Σχολιάστε τα αποτελέσματα.

ΑΠΑΝΤΗΣΗ:

Δες το αρχείο .mlx για τα α), β)

(γ) Η συνάρτηση `HAMM(x, y, B)` που υλοποίησα στο αρχείο `.mlx` εκμεταλλεύεται τη δομή του μητρώου A ώστε να μειώσει τον αριθμό των πράξεων (πολλαπλασιασμών) που απαιτούνται για τον υπολογισμό του γινομένου μητρώου επί μητρώο $A \cdot B$, ο οποίος χρειάζεται κανονικά $O(n^3)$ πράξεις.

Αρχικά, το μητρώο αποτελέσματος C αρχικοποιείται ως ένα μητρώο $2n \times s$ με μηδενικά στοιχεία. Έπειτα, προστίθεται σε αυτό το μητρώο B ως εξής:
- το πάνω μπλοκ $n \times s$ του B προστίθεται αυτούσιο,
- το κάτω μπλοκ $n \times s$ του B αφαιρείται από το C (ώστε να προστεθεί αλλά με αρνητικά στοιχεία λόγω του πολλαπλασιασμού με το $-I_n$).

Έτσι, έχουμε εκμεταλλευτεί την ύπαρξη των ταυτοτικών μητρώων στις γωνίες του μητρώου A και έχουμε γλιτώσει πολλαπλασιασμούς.

Στη συνέχεια, πρέπει στο παραπάνω μητρώο C να προσθέσουμε το αποτέλεσμα από τον πολλαπλασιασμό του xy^T με κάποια στοιχεία του B και του yx^T με κάποια άλλα στοιχεία του B. Αντί να εκτελέσουμε πρώτα τους πολλαπλασιασμούς των x, y που θα μας έδιναν ένα μητρώο $n \times n$, εκτελούμε πρώτα τον πολλαπλασιασμό ενός διανύσματος ορισμένων στοιχείων του B με το y^T και x^T αντίστοιχα και τον βαθμωτό που προκύπτει τον πολλαπλασιάζουμε με το x ή το y αντίστοιχα, και το τελικό διάνυσμα το προσθέτουμε στις σωστές θέσεις του μητρώου C.

Οι «σωστές θέσεις» που αναφέρω παραπάνω προσδιορίζονται με τη μορφή δεικτών στον επισυναπτόμενο κώδικα. Ακολουθεί εξήγηση:

- Το άνω αριστερό ταυτοτικό μητρώο πολλαπλασιάζεται με το επάνω $n \times s$ μπλοκ στοιχείων του B.
- Το κάτω δεξιό ταυτοτικό μητρώο με αρνητικό πρόσημο πολλαπλασιάζεται με τα υπόλοιπα στοιχεία του B (κάτω $n \times s$ μπλοκ).

Το γινόμενο xy^T πολλαπλασιάζεται με το κάτω $n \times s$ μπλοκ του B, ενώ το γινόμενο yx^T πολλαπλασιάζεται με το άνω $n \times s$ μπλοκ του B.

Αντί να πολλαπλασιάσω πρώτα το $n \times n$ μητρώο xy^T με το κάτω $n \times s$ μπλοκ του B, πολλαπλασιάζω πρώτα την κάθε στήλη j του κάτω $2n \times s$ μπλοκ με το y^T, και το βαθμωτό αυτό αποτέλεσμα με το x. Το διάνυσμα-στήλη που προκύπτει για κάθε j, το προσθέτω στο $C(1:n, j)$. Ομοίως για την περίπτωση του yx^T με το μόνο που αλλάζει να είναι το μπλοκ του B με το οποίο ασχολούμαι.

Παρατήρηση: Παρατηρώ πως για μεγάλα n, η συνάρτηση `HAMM` είναι σημαντικά ταχύτερη (δες και αποτελέσματα στο `.mlx` αρχείο).

Είναι αναμενόμενο αλλά γατί;

2.9 Ερώτημα 9

Δίνονται γενικά μητρώα A_i, $i = 1, 2, 3$ με διαστάσεις $n_i \times m_i$ και έστω ότι θέλουμε να υπολογίσουμε το $(A_1 \otimes A_2 \otimes A_3)x$ για δοθέν διάνυσμα x.

1. Ποιά διάσταση πρέπει να έχει το x ώστε ο πολλαπλασιασμός να είναι καλά ορισμένος;

2. Να εξετάσετε διάφορους τρόπους υλοποίησης του πολλαπλασιασμού με ή χωρίς τη χρήση της συνάρτησης `kron` και να αποφανθείτε σχετικά με τον ισχυρισμό ότι το Ω παραμένει το ίδιο ανεξαρτήτως τρόπου πολλαπλασιασμού. Αν ισχύει, να το δείξετε, αν όχι, να υλοποιήσετε αλγόριθμο που επιτυγχάνει όσο το δυνατόν μικρότερο Ω.

ΑΠΑΝΤΗΣΗ:

1.) Το διάνυσμα x πρέπει να έχει διάσταση $m_1 m_2 m_3$ ώστε ο παραπάνω πολλαπλασιασμός να είναι καλά ορισμένος.

2.) Σκέφτηκα 2 τρόπους υπολογισμού του παραπάνω γινομένου. Ο πρώτος τρόπος αξιοποιεί τη συνάρτηση kron του Matlab. Εκτελεί αρχικά το γινόμενο $(A_1 \otimes A_2)$ και το αποτέλεσμα αυτού πολλαπλασιάζεται με τη σειρά του κατά Kronecker με το A_3. Στο τέλος εκτελείται το γινόμενο του τελικού μητρώου-αποτελέσματος με το διάνυσμα x (δες αρχείο .mlx για τον κώδικα).

Ω;

Ο δεύτερος τρόπος βασίζεται στις ενότητες 1.3.7-8 του Συγγράματος.

2.10 Ερώτημα 10

Να υλοποιήσετε σε MATLAB κώδικα αντίστοιχο του Strassen που χρησιμοποιεί τη μέθοδο Winograd (βλ. διάλεξη 5). Να συγκρίνετε συστηματικά τις επιδόσεις του Strassen.

ΑΠΑΝΤΗΣΗ:

Δες το αρχείο .mlx για την πλήρη απάντηση.

ΕΡΓΑΣΙΑ 2 - ΚΩΔΙΚΑΣ

ΕΡΩΤΗΜΑ 3

```
%{
m = 4;


% Random values for the main diagonal
main_diag = randi(10, 1, m);

% Random values for the first upper diagonal
upper_diag = randi(10, 1, m-1);

% Create the upper bidiagonal matrix B
B = diag(main_diag) + diag(upper_diag, 1)

% Create the matrix A
A = [zeros(m), B; B', zeros(m)];

% Display the final square matrix A
disp('The matrix A is:')
disp(A);

n = size(A,1); % size of A

% Create the identity matrix I
I = eye(n);

% Define the row permutation of I
r = 3;      % You can change this value to experiment
% r=4;
% r=5;
perm = [];
for i = 1:r
    indices = i:r:n;
    perm = [perm, indices];
end

% Row permutation of I
P = I(perm, :)

T = P * A * P'
D = T'

% I notice that for different values of r,
% the matrix T remains SYMMETRIC !!!
```

ΕΡΩΤΗΜΑΤΑ 8α) και 8β)

```
% Ορισμός των n
n_values = 2.^(6:12);    % n = [64, 128, 256, 512, 1024, 2048, 4096]

% Δοκιμή για κάθε n
for i = 1:length(n_values)
    n = n_values(i); % Τρέχουσα τιμή του n

    % Δημιουργία τυχαίων διανυσμάτων και μήτρας
    x = rand(n, 1);
    y = rand(n, 1);
    I = eye(n);
    B = rand(2*n, m);

    % Δημιουργία της μήτρας A
    A = HamIbuild(x, y);

    % Ορισμός της συνάρτησης πολλαπλασιασμού για timeit
    multiplyFunc = @() mtimes(A, B);

    % Χρονομέτρηση της εκτέλεσης του πολλαπλασιασμού
    elapsedTime = timeit(multiplyFunc);

    % Εμφάνιση των αποτελεσμάτων
    fprintf('n = %d, Elapsed time for multiplication: %.6f seconds\n', n,
elapsedTime);
end
```

ΕΡΩΤΗΜΑ 8γ)

```
% Ορισμός των n
n_values = 2.^(6:12);    % n = [64, 128, 256, 512, 1024, 2048, 4096]
s = 3; % ή οποιοδήποτε μικρό αριθμό για s

% Δοκιμή για κάθε n
for i = 1:length(n_values)
    n = n_values(i); % Τρέχουσα τιμή του n

    % Δημιουργία τυχαίων διανυσμάτων και μήτρας
    x = rand(n, 1);
    y = rand(n, 1);
    B = rand(2*n, s); % Μητρώο 2n x s

    % Χρονομέτρηση της εκτέλεσης της HAMM
    hammFunc = @() HAMM(x, y, B);
    elapsedTimeHAMM = timeit(hammFunc);

    % Εμφάνιση των αποτελεσμάτων
    fprintf('n = %d, Elapsed time for HAMM multiplication: %.6f seconds\n', n,
elapsedTimeHAMM);
end
%}
```

Χρονομετρήσεις και αξολόγηση;

ΕΡΩΤΗΜΑ 9β)

```
n1 = randi(10);
n2 = randi(10);
n3 = randi(10);
m1 = randi(10);
m2 = randi(10);
m3 = randi(10);

A1 = rand(n1,m1);
A2 = rand(n2,m2);
A3 = rand(n3,m3);

x = rand(m1*m2*m3,1);


% 1ος τρόπος
A_temp = kron(A1,A2);
A_final = kron(A_temp, A3);
R = A_final * x
```

```
R = 96×1
    4.2308
    6.6305
    6.4367
    7.0741
    4.1912
    6.3992
    6.2555
    6.9137
    5.0453
    7.5748
      ⋮
```

```

% 2ος τρόπος
                ????

```

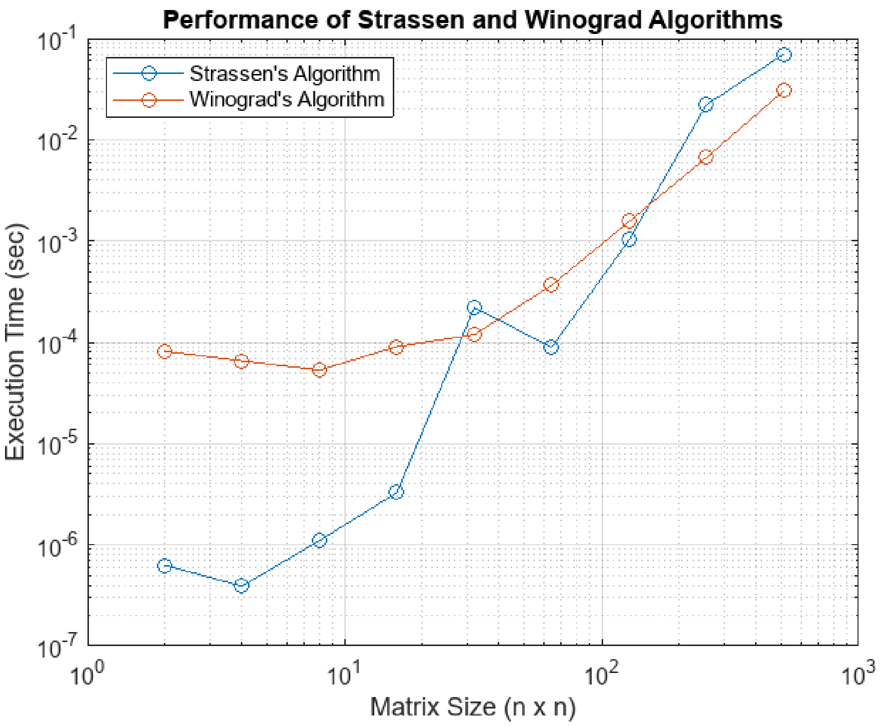
ΕΡΩΤΗΜΑ 10

```
% Define a range of matrix sizes to test
sizes = 2.^(1:9);
nmin = 64;           % Minimum size for Strassen's algorithm

% Arrays to store execution times
strassen_times = zeros(size(sizes));
winograd_times = zeros(size(sizes));

% Loop through each size and measure performance
for i = 1:length(sizes)
    n = sizes(i);

    A = rand(n);
    B = rand(n);
```

```
    strassen_time = @(A, B) strass(A, B, n, nmin);
    winograd_time = @(A, B) winograd(A, B);

    % Measure the performance of Strassen's algorithm
    strassen_times(i) = timeit(@() strassen_time(A, B));

    % Measure the performance of Winograd's algorithm
    winograd_times(i) = timeit(@() winograd_time(A, B));
end
```

```
Warning: The measured time for F may be inaccurate because it is running too fast. Try measuring
something that takes longer.
Warning: The measured time for F may be inaccurate because it is running too fast. Try measuring
something that takes longer.
Warning: The measured time for F may be inaccurate because it is running too fast. Try measuring
something that takes longer.
Warning: The measured time for F may be inaccurate because it is running too fast. Try measuring
something that takes longer.
```

```
% Plot the results
figure;
loglog(sizes, strassen_times, '-o', 'DisplayName', 'Strassen''s Algorithm');
hold on;
loglog(sizes, winograd_times, '-o', 'DisplayName', 'Winograd''s Algorithm');
hold off;


xlabel('Matrix Size (n x n)');
ylabel('Execution Time (sec)');
title('Performance of Strassen and Winograd Algorithms');
legend('Location', 'northwest');
grid on;
```



Παρατήρηση με βάση το γράφημα:

Αφού έτρεξα τον παραπάνω κώδικα του Ερωτήματος 10 κάμποσες φορές, διαπίστωσα από το γράφημα ότι για n μεγαλύτερο από περίπου 2^7 παρατηρώ καλύτερη απόδοση για τον αλγόριθμο Winograd συγκριτικά με τον Strassen (αναμενόμενο).

%% Συναρτήσεις Ερωτημάτων

Συναρτήσεις Ερωτήματος 10

```
function C = strass(A, B, n, nmin)

    % Ελέγχουμε αν το μέγεθος της μήτρας είναι μικρότερο ή ίσο με το ελάχιστο
μέγεθος
    if n <= nmin
        % Αν n <= nmin, υπολογίζουμε το C = AB με συμβατικό πολλαπλασιασμό
        C = A * B;
    else
        % Διαχωρίζουμε τα μητρώα A και B σε υπομητρώα
        m = n / 2;      % Το μέγεθος των υπομητρώων
        u = 1:m;        % Οι γραμμές και στήλες του πρώτου υπομητρώου
        v = m + 1:n;    % Οι γραμμές και στήλες του δεύτερου υπομητρώου

        % Υπολογίζουμε τα P1 έως P7 με αναδρομή
        P1 = strass(A(u, u) + A(v, v), B(u, u) + B(v, v), m, nmin);
        P2 = strass(A(v, u) + A(v, v), B(u, u), m, nmin);
        P3 = strass(A(u, u), B(u, v) - B(v, v), m, nmin);
        P4 = strass(A(v, v), B(v, u) - B(u, u), m, nmin);
        P5 = strass(A(u, u) + A(u, v), B(v, v), m, nmin);
        P6 = strass(A(v, u) - A(u, u), B(u, u) + B(u, v), m, nmin);
        P7 = strass(A(u, v) - A(v, v), B(v, u) + B(v, v), m, nmin);

        % Συνδυάζουμε τα αποτελέσματα για να υπολογίσουμε το C
        C = zeros(n);                      % Αρχικοποιούμε το μητρώο C
        C(u, u) = P1 + P4 - P5 + P7;   % Υπολογισμός του C(u, u)
        C(u, v) = P3 + P5;             % Υπολογισμός του C(u, v)
        C(v, u) = P2 + P4;             % Υπολογισμός του C(v, u)
        C(v, v) = P1 + P3 - P2 + P6;   % Υπολογισμός του C(v, v)
    end
end
```

```matlab
function C = winograd(A, B)
    % winograd: Πολλαπλασιασμός μητρώων χρησιμοποιώντας τη μέθοδο Winograd
    % C = winograd(A, B)
    %
    % Είσοδος:
    % A, B - μητρώες n x n που πρόκειται να πολλαπλασιαστούν (πρέπει να είναι
τετράγωνες και ίδιου μεγέθους)
    %
    % Έξοδος:
    % C - αποτέλεσμα του πολλαπλασιασμού A * B

    % Λαμβάνουμε το μέγεθος της μήτρας
    [n, m] = size(A);

    if n ~= m || n ~= size(B, 1) || m ~= size(B, 2)
        error('Οι μήτρες πρέπει να είναι τετράγωνες και ίδιου μεγέθους.');
    end

    % Αρχικοποίηση του μητρώου C
    C = zeros(n, n);

    % Βασική περίπτωση: αν το μητρώο είναι 1x1
    if n == 1
        C = A * B;      % Συμβατικός πολλαπλασιασμός
    else
        half = n / 2;  % Διάσταση των υπομητρώων

        % Υπομητρώα
        A11 = A(1:half, 1:half);
        A12 = A(1:half, half+1:end);
        A21 = A(half+1:end, 1:half);
        A22 = A(half+1:end, half+1:end);

        B11 = B(1:half, 1:half);
        B12 = B(1:half, half+1:end);
        B21 = B(half+1:end, 1:half);
        B22 = B(half+1:end, half+1:end);

        % I. Προσθέσεις
        S1 = A21 + A22;
        S2 = S1 - A11;
        S3 = A11 - A21;
        S4 = A12 - S2;

        S5 = B12 - B11;
        S6 = B22 - S5;
        S7 = B22 - B12;
        S8 = S6 - B21;

        % II. Πολλαπλασιασμοί
        M1 = S2 * S6;
        M2 = A11 * B11;
        M3 = A12 * B21;
        M4 = S3 * S7;
```

```
        M5 = S1 * S5;
        M6 = S4 * B22;
        M7 = A22 * S8;

        % III. Προσθέσεις
        T1 = M1 + M2;
        T2 = T1 + M4;

        C11 = M2 + M3;
        C12 = T1 + M5 + M6;
        C21 = T2 - M7;
        C22 = T2 + M5;

        % Συγκέντρωση των αποτελεσμάτων στο μητρώο αποτελέσματος C
        C(1:half, 1:half) = C11;
        C(1:half, half+1:end) = C12;
        C(half+1:end, 1:half) = C21;
        C(half+1:end, half+1:end) = C22;
    end
end
```

Συνάρτηση Ερωτήματος 8α)

```
function A = HamIbuild(x, y)
    % Ελέγχουμε αν x και y είναι ισομεγέθη διανύσματα
    if length(x) ~= length(y)
        error('Τα διανύσματα x και y πρέπει να είναι ισομεγέθη.');
    end

    % Διαστάσεις
    n = length(x);

    % Δημιουργία των συνιστωσών της μήτρας A
    I_n = eye(n);          % Μονάδα μήτρας I_n
    xyT = x * y';          % xy^T
    yxT = y * x';          % yx^T    Μπορούσες να αποφύγεις το κόστος των υπολογισμών καθώς
                                     (y*x')' = x*y'

    % Κατασκευή της μήτρας A
    A = [I_n, xyT;
         yxT, -I_n];
end
```

Συνάρτηση Ερωτήματος 8γ)

```
function C = HAMM(x, y, B)
    % Ελέγχουμε αν x και y είναι ισομεγέθη διανύσματα
    if length(x) ~= length(y)
        error('Τα διανύσματα x και y πρέπει να είναι ισομεγέθη.');
    end

    % Δηλώνουμε τις διαστάσεις
    n = length(x);
```

```
    s = size(B, 2);

    % Δημιουργία του μητρώου αποτελέσματος C
    C = zeros(2*n, s);

    % Αρχικοποίηση του μητρώου C με τις τιμές που προκύπτουν από τις
    % πράξεις πολλ/σμού των μπλοκ In  και -In με τα αντίστοιχα μπλοκ του B
    C(1:n, :) = B(1:n, :);
    C(n+1:end, :) = C(n+1:end, :) - B(n+1:end, :);
                     μη απαραίτητο
    % Πρόσθεση στο παραπάνω C του γινομένου x * ( y' * B(n+1:end, j) ) για
    % κάθε στήλη j
    for j = 1:s
        C(1:n, j) = C(1:n, j) + x * ( y' * B(n+1:end, j) );
    end

    % Ομοίως με επάνω πρόσθεση του γινομένου y * ( x' * B(1:n,j) )
    for j = 1:s
        C(n+1:end, j) = C(n+1:end, j) + y * ( x' * B(1:n,j) );
    end



end
```

Χρονομετρήσεις;