תרגיל בית מס. 4

בתרגיל זה נממש שלוש פונקציות לכפל מטריצות. בראשונה נבצע את הכפל כולו בחוט יחיד. בשניה נמקבל את החישוב על ידי הקצאת חוט לכל שורה במטריצת התוצאה, ובשלישית נמשיך ונמקבל את החישוב על ידי הקצאת חוט נפרד לחישוב כל אבר בתוצאה. לבסוף נמדוד את זמן הביצוע של המכפלה בכל אחת מן השיטות ונמצא מי מהן היעילה ביותר.

(היש לכם מחשבה מה עשויה להיות התוצאה, עוד לפני שנממש?)

כפל מטריצות

(עבור אלה ששכחו את אלגברה א' מאז השנה שעברה...)

מטריצה היא מלבן של מספרים: M שורות שבכל אחת N מספרים. ניתן לכפול שתי מטריצות זו בזו אם יש להן מימד משותף: אם מספר השורות באחת שווה למספר העמודות בשניה. הכפל אינו קומוטטיבי.

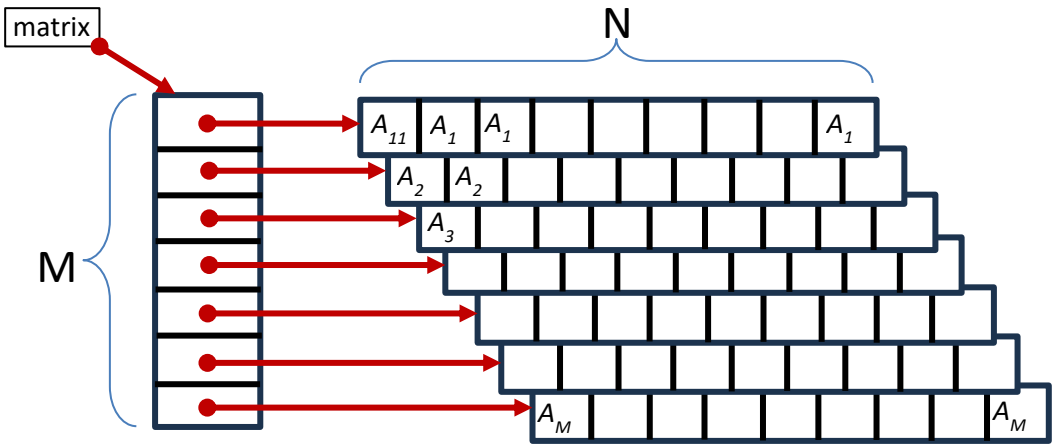
תהי מטריצה A בגודל M×N $\mathbf{A}=\{A_{ij}\}$ ומטריצה B בגודל N×L $\mathbf{B}=\{B_{jk}\}$. המכפלה $\mathbf{C}=\mathbf{A}\times\mathbf{B}$ היא מטריצה בגודל M×L ונתונה על ידי

$$C_{ik} = \sum_{j=1}^{N} A_{ij} \times B_{jk}$$

כלומר, כל אבר במטריצת התוצאה הוא המכפלה הפנימית של וקטור שורה במטריצה אחת ווקטור עמודה בשניה, ומיקומו במטריצת התוצאה הוא בשורה שמספרה כמספר השורה של וקטור השורה ובעמודה שמספרה כמספר וקטור העמודה במטריצות האם שלהם.

איך לייצג מטריצה?

אתם חופשיים לעשות זאת בכל דרך שנראית לכם. ברם, כיון שבסופו של עניין נרצה לבדוק את ביצועי התכנית על מטריצות בגדלים שונים, מומלץ להקצות מקום למטריצות באופן דינאמי, ולא להגדיר אותן עם גדלים קבועים מראש. דרך טובה היא להקצות מערך חד ממדי שגודלו כמספר שורות המטריצה, ואבריו הם מערכים חד ממדיים של double ואורכם הוא כמספר העמודות שבה.



מטריצה בגודל M×N תוקצה, אם כן, באופן הבא:

```
double **matrix = (double**) malloc( M * sizeof(double*) );

for( i = 0; i < M; i++ )
    matrix[i] = (double*) malloc( N * sizeof( double ) );
```

יצירת תוכן

את המטריצות שיצרתם יש למלא בערכים. השתמשו במחולל מספרים אקראיים (random()), למשל). הקוד הבא מייצר מספרים בהתפלגות נורמלית עם ממוצע 0.0 וסטית תקן 1.0. (כדי לקבל ערכים בהתפלגות נורמלית עם ממוצע μ וסטית תקן σ אפשר לכפול כל מספר ב-σ ולהוסיף μ.)

```
#include <math.h>
#define M_PI 3.14159265358979323846
double drand()                  /* uniform distribution (0..1]  */
{
    return( ( random() + 1.0 ) / ( RAND_MAX + 1.0 ) );
}

double normal()                 /* normal distribution, centered on 0, std dev 1 */
{
    return( sqrt( -2 * log( drand() ) ) * cos( 2 * M_PI * drand() ) );
}
```

פונקציות לכפל מטריצות

לשלוש הפונקציות שתכתבו יש חתימה דומה:

```
double **mult_xxx( double **m1, double **m2, int M, int N, int L )
```

(אם תבחרו מבנה אחר להחזקת המטריצה, שנו את ה-"double **" בהגדרה לטיפוס שבחרתם אתם.)
הפונקציה תקבל שתי מטריצות, m1 בגודל M×N ו-m2 בגודל N×L, ותחזיר מטריצה בגודל M×L שהיא המכפלה של m1 ב-m2.

1. הפונקציה הראשונה, תיקרא mult_by_loop ותבצע את הכפל בעצמה, באמצעות שלוש לולאות מקוננות, לפי הנוסחה שלעיל.
2. הפונקציה השניה, mult_by_row, תחלק את העבודה ל-M חוטים, כאשר כל אחד מהם יהיה אחראי למלא שורה אחת (מתוך M) במטריצת התוצאה. יהיה עליה להקצות מקום למטריצת התוצאה מראש, ואז להעביר לחוטים את כל המידע הבא:
   - מחוונים למטריצות m1 ו-m2;
   - את הממדים של שתי המטריצות: M, N ו-L
   - מחוון למטריצת התוצאה

   לבסוף יהיה עליה לחכות לכל החוטים עד שיסתיימו.
3. הפונקציה השלישית, mult_by_element, תחלק את העבודה ל-M*L חוטים, כאשר כל חוט אחראי לחישוב של אבר אחד בלבד במטריצת התוצאה. גם פונקציה זו צריכה להעביר לידיעת החוטים את אותו המידע כמו mult_by_row, אבל על כל חוט לחשב רק אבר אחד בתוצאה.

מדידת זמן

כבר ביצענו בעבר מטלות עם מדידת זמן. אני ממליץ לכתוב שתי פונקציות, אחת שדוגמת את השעון (באמצעות gettimeofday()) ושומרת את ערכו, והשניה שדוגמת אותו שוב ומחשבת כמה זמן חלף מאז הפעם הקודמת, ומדפיסה זאת בצורה קריאה. כך תוכלו למדוד את זמן הביצוע של שלוש פונקציות המכפלה מבלי לשכפל יותר מידי קוד. (בדוגמת ה-main() שלהלן הן נקראות set_timer() ו-print_elapsed_time(), בהתאמה. האחרונה, כפי שנתן ללמוד משמה, גם מדפיסה את הזמן שחלף.)

main() לדוגמא

```
int main( int argc, char *argv[] )
{
   if( argc < 4 )    {
      printf( "Usage: %s M N L to multiply an MxN matrix by an NxL matrix\n", argv[0] );
      return( EXIT_SUCCESS );
   }

   int m = atoi( argv[1] );          //  ממדי המטריצות הם פרמטרים
   int n = atoi( argv[2] );
   int l = atoi( argv[3] );

   double **m1 = make_matrix( m, n );
   print_matrix( m1, m, n, "Matrix 1" );

   double **m2 = make_matrix( n, l );
   print_matrix( m2, n, l, "Matrix 2" );

   set_timer();
   double **m3 = mult_by_loop( m1, m2, m, n, l );
   print_elapsed_time();
   print_matrix( m3, m, l, "m1 x m2 by loop" );

   set_timer();
   double **m4 = mult_by_row( m1, m2, m, n, l );
   print_elapsed_time();
   print_matrix( m4, m, l, "m1 x m2 by row" );

   set_timer();
   double **m5 = mult_by_element( m1, m2, m, n, l );
   print_elapsed_time();
   print_matrix( m5, m, l, "m1 x m2 by element" );

   return( EXIT_SUCCESS );
}
```

את התרגיל יש לבצע בזוגות בבית על מחשב אישי או על מחשב במעבדה במכללה. יש לקבץ את קוד התכנית ואת הפלט שלה באמצעות ZIP, RAR או תוכנה דומה. על הפלט לכלול את תוכן המטריצות ואת תוצאת המכפלה שלהן בכל אחת מהשיטות. שם הקובץ שתיצרו צריך להיות בפורמט הבא:

Ex2.zip_123456789_Ploni_Almoni_987654321_Israel_Israeli

כלומר: השם באנגלית ומספר הזהות של כל אחד מהמגישים, והמחרוזת "Ex4", מופרדים באמצעות קו תחתון ביניהם. את הקובץ (היחיד) יש להעלות באמצעות moodle לאתר הקורס. נא ציינו, בנוסף, את שמותיכם (בעברית) ואת מספרי הזהות שלכם בגוף ההגשה בקובץ ReadMe.txt. את התרגיל יש להגיש עד יום 21 במאי, 2025.

בהצלחה!