# עבודה מספר 2 באלגוריתמים מתקדמים:

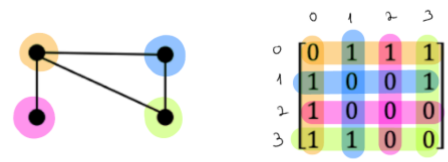
## שאלה מספר 1:

*נתון $G$ גרף לא מכוון ללא קשתות מקבילות או לולאות עצמיות.*
*משולש ב$G$ הוא קבוצה של 3 קודקודים נפרדים $v_1, v_2, v_3$ כך שכל זוג מהם מחובר באמצעות קשת.*
*צריך לתכנן אלגוריתם שבהינתן גרף $G$ עם $n$ קודקודים, סופר את מספר המשולשים ב$G$ .*
*זמן הריצה של האלגוריתם שלך צריך להיות קטן יותר באופן אסימפטוטית מ $n^3$.*
*רמז: להשתמש בכפל מטריצה מהיר.*

## פתרון:

נציג את הגרף $G$ כמטריצת שכנים $A$ בגודל $n \times n$ כאשר אם יש קצה בין קודקודים נסמן $A[i][j] = 1$ , אחרת 0.

דוגמה:



נוכל לפתור את הבעיה על ידי כפל מטריצות כפי שלמדנו בשיעור, וכדי להוריד את זמן הריצה לקטן יותר באופן אסימפטוטית מ$n^3$ ניעזר באלגוריתם של שטרסן.

### מה שנעשה:

נסמן את תוצאת כפל מטריצה $A \times A = B$ וגם נסמן $A \times B = C$ ובסה"כ $C_{i,j} = \sum_{k=1}^{n} A_{i,k} B_{k,j}$
האלגוריתם של שטרסן מחשב את הכפל מטריצה בצורה הזו:
נניח ויש לנו מטריצת $2 \times 2$

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

נחשב 7 כפלים:

$$\begin{aligned}
p_1 &= (a_{11} + a_{22}) \cdot (b_{11} + b_{22}) \\
p_2 &= (a_{21} + a_{22}) \cdot b_{11} \\
p_3 &= a_{11} \cdot (b_{12} - b_{22}) \\
p_4 &= a_{22} \cdot (-b_{11} + b_{21}) \\
p_5 &= (a_{11} + a_{12}) \cdot b_{22} \\
p_6 &= (-a_{11} + a_{21}) \cdot (b_{11} + b_{12}) \\
p_7 &= (a_{12} - a_{22}) \cdot (b_{21} + b_{22})
\end{aligned}$$

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} p_1 + p_4 - p_5 + p_7 & p_3 + p_5 \\ p_2 + p_4 & p_1 + p_3 - p_2 + p_6 \end{pmatrix}$$

איבר מהמטריצה A אל מול איבר מהמטריצה B
התועלת האמיתית של האלגוריתם היא העובדה שניתן להשתמש בו רקורסיבית. יש לנו מטריצה גדולה אז נחלק לארבעה בלוקים. והרעיון להשתמש במטריצה של טרסן אבל כל פעם שצריך לכפול 2 בלוקים נעשה את זה בצורה רקורסיבית.
חיבור זו פעולה זולה אבל כפל זה מה שנשלם עליו ולכן העובדה שחסכנו בכפל היא הפקטור המשמעותי.
ואז יצא לנו שביצענו כפל מטריצות בגודל $n \times n$ בזמן $O(n^{\log_2 7}) = O(n^{2.81\cdots})$ שזה טוב יותר מ $O(n^3)$ וזהו שיפור

ואז במטריצה C הערך $C[i][j]$ מייצג את מספר הנתיבים באורך 2 מקודקוד i לקודקוד j.

אם ל$C[i][j]$ יש ערך $k$ זה אומר שיש $k$ נתיבים באורך 2 שמתחילים מקודקוד $i$ ומסתיימים בקודקוד $j$.
משולשים נספרים פעמיים במטריצה C כי כל משולש סופר פעם אחת בנתיב $i \to j \to k$ ופעם נוספת בנתיב $i \to k \to j$
אז כדי לספור את כמות המשולשים האמיתית צריך לחלק את הערך ב2.

האלגוריתם עצמו:
1. ניצור מטריצה C ריקה בגודל $n \times n$
2. נבצע כפל מטריצה $A \times A = B$ בסיבוכיות של $n^2$
3. כעת נבצע כפל מטריצה $A \times B = C$ על ידי האלגוריתם של שטרסן (*ראה פירוט מעלה) בסיבוכיות של $O\left(n^{log_2 7}\right) = O(n^{2.81\dots})$
4. נסכום את הערכים של$C[i][j]$ ונחלק ב2 כדי לקבל ספירה כוללת של משולשים ב$G$

**הראינו אלגוריתם לפתרון הבעיה וזמן הריצה של אלגוריתם זה קטן באופן אסימפטוטית מ$n^3$** *מ.ש.ל*

## <u>שאלה מספר 2</u>:

*מטריצה A בגודל $n \times n$ נקראת "טופוליץ מטריקס" אם היא קבועה באלכסוניה. כלומר קיימים $2n-1$ מספרים $a_0, a_1, \dots, a_{2n-1}$ כך ש:*

$$A = \begin{pmatrix} a_{n-1} & a_{n-2} & a_{n-3} & \cdots & \cdots & a_0 \\ a_n & a_{n-1} & a_{n-2} & \ddots & \ddots & a_1 \\ a_{n+1} & a_n & a_{n-1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & a_{n-2} & a_{n-3} \\ \vdots & \ddots & \ddots & a_n & a_{n-1} & a_{n-2} \\ a_{2n-2} & \cdots & \cdots & a_{n+1} & a_n & a_{n-1} \end{pmatrix}$$

*הראה אלגוריתם שבהינתן מטריצה טופוליץ $A \in \mathbb{C}^{n\times n}$ וקטור $v \in \mathbb{C}^n$ מחשב את המכפלה שלהם $Av$ בגודל של $O(nlogn)$ באמצעות פעולות אריתמטיות.*

## פתרון:

תהא A מטריצת טופוליץ $n \times n$ ויהיה וקטור $v = [x_0, x_1, \dots, x_{n-1}]$ נדרש לחשב:

$$\begin{bmatrix} a_{n-1} & \dots & a_2 & a_1 & a_0 \\ a_n & a_{n-1} & \dots & a_2 & a_1 \\ \vdots & a_n & & & a_2 \\ & & & & \vdots \\ a_{2n-2} & & & a_n & a_{n-1} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{bmatrix} = \begin{bmatrix} \sum_{k=0}^{n-1} x_k a_{n-1-k} \\ \sum_{k=0}^{n-1} x_k a_{n-k} \\ \sum_{k=0}^{n-1} x_k a_{n+1-k} \\ \vdots \\ \sum_{k=0}^{n-1} x_k a_{n-1+j-k} \\ \vdots \\ \sum_{k=0}^{n-1} x_k a_{2n-2-k} \end{bmatrix}$$

כדי לחשב את מכפלת המטריצה-וקטור ביעילות נשתמש בטכניקות כפל והערכה של פולינום.
את מכפלת הפולינומים בפועל $P(x) \times Q(x)$ נחשב באמצעות FFT אלגוריתם שלמדנו בכיתה.

### האלגוריתם

1. בהינתן מטריצת טופוליץ $A \in \mathbb{C}^{n\times n}$ *וגם וקטור* $v \in \mathbb{C}^n$
   נסמן את המרכיבים של A $a = [a_0, a_1, \dots, a_{2n-2}]$ ואת האלמנטים של v $v = [v_0, v_1, \dots, v_{n-1}]$ .
   נשייך את A לפולינום הבא: $P(x) = a_0 + a_1x + a_2x^2 + \dots + a_{2n-2}x^{2n-2}$
   נשייך את v לפולינום הבא: $Q(x) = v_0 + v_1x + v_2x^2 + \dots + v_{n-1}x^{n-1}$

2. צור פולינום $R(x) = R_0 + R_1x + R_2x^2 + \dots + R_{3n-3}x^{3n-3}]$ כאשר כל מקדם $R_i$ יתאים למכפלה $Av$

3. נאריך את הפולינומים עצמם $P(x), Q(x)$ לגודל של $3n-2$. כדי שיהיו בדרגה $3n-3$ ולרפד באפסים את השאר.

*** כעת לקחתי שקופית מהמצגת לחישוב כפל פולינומים בעזרת שורשי היחידה עם FFT וזה הרעיון הכללי ותמללתי למילים שלי למטה בשלבים 4-6:*

Algorithm for multiplying polynomials $f, g$ such that $\deg f + \deg g < n$:

1. Compute $\hat{f} = \left(f(1), f(\omega), \ldots, f(\omega^{n-1})\right)$ and similarly $\hat{g}$
2. Compute $\widehat{f \cdot g} = \left(\hat{f}_0 \cdot \hat{g}_0, \ldots, \hat{f}_{n-1} \cdot \hat{g}_{n-1}\right)$
   (this is the evaluation vector of $f \cdot g$ on $1, \omega, \ldots, \omega^{n-1}$)
3. Compute $f \cdot g$ using the inverse Fourier transform on $\widehat{f \cdot g}$
   (this is done as in step 1 using $A_{\omega^{-1}}$)

Total time complexity: $O(n \log n)$

4. כעת כשהפולינומים באותו הגודל $3n-3$ אפשר לחשב עם התמרת פורייה ולהחיל את אלגוריתם $FFT$.
$F(P(x)), F(Q(x))$
5. נחשב את המכפלה מבחינת האלמנט של $F(P(x)), F(Q(x))$ כדי לקבל $F(R(x))$
6. נחשב $FFT^{-1}$ כדי לקבל את המקדמים של הפולינום $R(x)$. המקדמים המתקבלים מתאימים למרכיבי הווקטור $Av$.

הראינו אלגוריתם לפתרון הבעיה. על ידי שימוש באלגוריתמים $\boldsymbol{FFT}$ וגם $\boldsymbol{FFT^{-1}}$ המספר הכולל של פעולות אריתמטיות שנדרשות עבור אלגוריתם זה הוא $\boldsymbol{O(nlogn)}$ מה שמספק דרך יעילה לחישוב $\boldsymbol{Av}$ עבור מטריצת טופוליץ ווקטור. מ.ש.ל

## <u>שאלה מספר 3:</u>

*נותנים לך קובייה עם $n$ פאות. בכל הטלת קובייה ההסתברות שהיא תיפול בצד ה$i$ הוא $p_i$ עבור כל $1 \leq i \leq n$.*
*(לכן אנחנו חייבים ש$p_i \geq 0$ וגם שיתקיים $\sum_{i=1}^{n} p_i = 1$)*
*כעת נשחק משחק בו נטיל את הקובייה פעמיים ונסכום את שתי התוצאות שנקבל. אנחנו רוצים לחשב את ההתפלגות של התוצאות משחק. עבור כל $2 \leq s \leq 2n$ אנחנו רוצים לחשב את ההסתברות שסכום התוצאות של שתי התוצאות הוא $s$.*

א. *תכנן אלגוריתם שמחשב את ההסתברויות האלו באמצעות $O(nlogn)$.*

### פתרון:

כמו בתרגיל הקודם נייצג את ההסתברויות כמקדמים בפולינום ונבצע כפל פולינומים באמצעות FFT ולאחר מכן נקרא את המקדמים כדי לקבוע את ההסתברויות של התוצאות.
נגדיר את הפונקציה היוצרת $G(x) = p_1x + p_2x^2 + \cdots + p_nx^n$ כאשר $p_i$ מייצגת את ההסתברות שהקוביות יפלו בצד הi.
בדי לחשב את התפלגות התוצאות עלינו לחשב את המקדמים של הפולינום $G(x)^2$ כי הוא מייצג התפלגות הסכומים להטלת הקובייה פעמיים.

האלגוריתם:

1. בהינתן הסתברויות $p_1, p_2, \ldots, p_n$ עבור הקוביות עם n פאות, ניצור מערך P בגודל 2n ונרפד אותו באפסים.
2. נחיל את FFT על המערך P כדי לקבל את התמרת פורייה שלו באופן הבא$F(P)$ .
3. נחשב את הריבוע של $F(P)$ מבחינת האלמנט כדי לקבל $F(P)^2$
4. נחיל את $FFT^{-1}$ על $F(P)^2$ כדי לקבל את המקדמים של הפולינום $G(x)^2$
5. המקדמים שקיבלנו מייצגים את ההסתברויות לתוצאות של משחק הקוביות.
6. אחרי שהשגנו את המקדמים ננרמל אותם על ידי חלוקת כל מקדם במספר הכולל של התוצאות שהוא 2n כדי לקבל את ההסתברויות בפועל.

**הראינו אלגוריתם לפתרון הבעיה. על ידי שימוש באלגוריתמים FFT וגם $FFT^{-1}$ המספר הכולל של פעולות אריתמטיות שנדרשות עבור אלגוריתם זה הוא $O(nlogn)$ מה שמספק דרך יעילה לחישוב ההסתברויות מ.ש.ל**

ב. *נשחק אותו משחק רק עכשיו נטיל$2^k$ פעמים את הקובייה במקום פעמיים. בהנחה ש $2^k \leq n$ ונסכום את התוצאות. תכנן אלגוריתם שמחשב את ההסתברויות של כל התוצאות של המשחק הזה בהכי פחות פעולות שנוכל.*

### פתרון:

כדי לחשב עבור הגרסה המורחבת של משחק הקוביות שבה נטיל $2^k$ פעמים ונסכום את התוצאות נשתמש באלגוריתם שכתבנו בסעיף א. נריץ את האלגוריתם מסעיף א k פעמים ונקבל את $F(P)^{2k}$ שזו בעצם התוצאה הרצויה.
הרי שאלגוריתם בסעיף א לוקח $O(nlogn)$ זמן ריצה. הכפלת p בעצמה פי $2^k = 2^{2^{logk}}$ שווה ערך להכפלת תוצאת הכפל הקודמת עם עצמה עבור $logk$ פעמים.
כל כפל לוקח $O(n)$ מה שלוקח בסה"כ $O(nlogn)$.
בסה"כ האלגוריתם פועל $O(nlogn) + O(nlogk) = O(nlogn)$ עבור $n \leq k$

## שאלה מספר 4:

*בשאלה זו נתכנן אלגוריתם מקבילי שמחשב את הדרגה של מטריצה בגודל $n \times m$ מעל $\mathbb{R}$.*
*נניח ללא אובדן הכלליות ש $m < n$.*

א. צריך להראות ש $rank(A) = rank(A^T A)$

### פתרון:

*דוגמה איך זה יראה:*

$$A_{m\times n} = \begin{bmatrix} a_{[0,0]} & \cdots & r_{[0,n]} \\ \vdots & \ddots & \vdots \\ r_{[m,0]} & \cdots & r_{[m,n]} \end{bmatrix} \qquad {A_{n\times m}}^T = \begin{bmatrix} a_{[0,0]} & \cdots & r_{[0,m]} \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ r_{[n,0]} & \cdots & r_{[n,m]} \end{bmatrix}$$

דרגה של מטריצה A מוגדרת ומסומנת כך$\mathrm{rank(A) = dimImA}$ כלומר מספר העמודות של A שהן בלתי תלויות לינארית.
קיים משפט בלינארית שאומר שהתמונה של אופרטור לינארי איזומורפית למקור חלקי הגרעין. המקור של שני האופרטורים הלינאריים זהה אז אם נראה שהגרעין זהה אז התמונות זהות ואז הדרגה שווה.
נציג את זה:
נשים לב שעבור מטריצה $\mathrm{A_{n\times m}}$ ועבור $\mathrm{A^T}$ מטריצה הפיכה, מתקיים לפי ההגדרה

$$\mathrm{rank(A) = dimImA \ \times rank\left(A^TA\right) = dimIm\left(A^TA\right)}$$

נשים לב שמטריצה הפיכה היא ריבועית. נסמן ב$\mathrm{F^n}$ את הפעולה של המטריצה על כל איברי המרחב.
ולפיכך מתקיים :

$$\mathrm{Im\left(A^TA\right) = Im\left(A^TA(F^n)\right) = Im\left(A^T(A(F^n))\right) = Im(A(F^n)) = Im(A)}$$

משוויון המרחבים נובע כמובן שהמדדים שווים כלומר דרגות המטריצות שוות.
**ולכן ש $\mathbf{rank(A) = rank\left(A^TA\right)}$ מ.ש.ל**

ב. *סמן $A^T A = B \in \mathbb{R}^{m\times m}$. נניח שיש לB ערכים עצמיים$\lambda_1, \dots, \lambda_m$ תן לk להיות המספר של $\lambda_i = 0$. הראה ש* $rank(B) = m - k$
*רמז: השתמש בעובדה שB סימטרי ומכאן הוא בר אלכסון, כך הריבוי האלגברי של כל ערך עצמי שווה לגיאומטרי.*

### פתרון:

יש לנו ערכים עצמיים של B שמוגדרים באופן הבא: $\mathrm{0 \le \lambda_1 \le \lambda_2 \le \lambda_k \le \cdots \le \lambda_m}$ .
מכיוון שB סימטרית וניתנת לאלכסון. המשמעות היא שהמכפלה האלגברי של כל ערך שווה למכפלה הגאומטרי שלו.
אם ערך עצמי $\mathrm{\lambda_i = 0}$, המכפלה שלו היא גם אפס מכיוון שמרחב האפס הקשור לערך עצמי אפס הוא טריוואלי.
לכן עבור כל ערך עצמי $\mathrm{\lambda_i = 0}$ המכפלה הגאומטרית היא אפס.
הדרגה של מטריצה שווה למספר הערכים העצמיים שאינם אפס. לכן אם k ערכים עצמיים של B שווים לאפס, הדרגה של Bהיא m-k מכיוון שישנם m ערכים עצמיים סהכ.
**לפיכך הראינו ש $\boldsymbol{rank(B) = m - k}$ מ.ש.ל**

ג. *תן ל* $p(x) = \det(xI - A) = (x - \lambda_1)(x - \lambda_2) \dots (x - \lambda_m)$ הפולינום האפייני של $B$. הראה ש $x^k$ הוא החזקה הקטנה ביותר של $x$ עם מקדם שאינו אפס של ל $p(x)$
*השתמש בעובדה זו כדי לעצב אלגוריתם שמחשב דרגה $A$ בזמן מקביל* $O(\log^2 n)$

## פתרון:

יידוע ש $p(x) = \prod_{i=1}^{m}(x - \lambda_i)$ הוא הפולינום האפייני של $B$. מכיון ש $\lambda_1 = \lambda_2 = \cdots = \lambda_k = 0$ *נוכל לכתוב ש* $p(x) = x^k \prod_{i=k+1}^{m}(x - \lambda_i) = x^k \sum_{i=0}^{m-k} a_i x^i$ , $a_0 = \prod_{i=k+1}^{m} \lambda_i \neq 0$
ולכן $x^k$ הוא החזקה הקטנה ביותר של x עם מקדם none-zero (מקדם שאינו אפס) בפולינום $p(x)$.

האלגוריתם שמחשב דרגה A בזמן *מקביל* $O(\log^2 n)$ :

1. בהינתן מטריצה $A \in \mathbb{R}^{n \times m}$ נחשב את $B \leftarrow A^T A \in \mathbb{R}^{m \times m}$ בזמן מקבילי של $O(\mathrm{m}^2)$
2. מצא את הפולינום האפייני של $B$ שהוא $p(x)$ בזמן של $O(\log^2 n)$
3. מצא את $k$ החזקה הקטנה ביותר של $x$ עם מקדם שאינו אפס של $p(x)$ בזמן של $O(\log n)$
4. החזר את *m-k* בזמן של $O(1)$

הראינו אלגוריתם לחישוב דרגה A בזמן מקבילי של $\boldsymbol{O(\log^2 n)}$ וכמובן משתמש בהוכחה שהוכחנו מעל. מ.ש.ל.