ממן 13

שאלה 1

א. נתון הפולינום הבא:

$p(x) = x^3 + 4x^2 + 2x + 3$

וקטור המקדמים של הפולינום p הוא: (3,2,4,1)

נציג את ריצת FFT על וקטור המקדמים של p עם הפרמטר ω_4. כאשר מתקיים:

$$\omega_4 = e^{\frac{2\pi i}{4}}$$

ω_4 הוא שורש יחידה פרימיטיבי מסדר 4 ו-4 הוא חזקה של 2 כנדרש לריצת FFT.

<u>בריצה הראשונה של FFT נקבל את הקריאות הרקורסיביות הבאות:</u>

$(f_e(1), f_e(\omega_4^2)) \leftarrow FFT((3,4), \omega_4^2)$

$(f_o(1), f_o(\omega_4^2)) \leftarrow FFT((2,1), \omega_4^2)$

<u>לאחר מכן בקריאה הרקורסיבית הראשונה נקבל:</u>

$(f_e(1)) \leftarrow FFT((3), \omega_4^4)$

$(f_o(1)) \leftarrow FFT((4), \omega_4^4)$

ומהן נגיע לתנאי העצירה כי n=1 ואז נחזיר $f_e(1)=3$ ו-$f_o(1)=4$.

כעת כאשר האלגוריתם קיבל תשובה ל-2 הקריאות הרקורסיביות ל-FFT שלעיל נתחיל בריצת לולאת ה-for מ-k=0 עד n-1 (1 = 2-1).

- עבור k=0:

$$f((\omega_4^2)^k) = f((\omega_4^2)^0) = f(1) \leftarrow f_e((\omega_4^2)^{2k}) + (\omega_4^2)^k * f_o((\omega_4^2)^{2k}) = f_e(1) + 1*f_o(1) = 3+4 = 7$$

- עבור k=1:

$$f((\omega_4^2)^k) = f((\omega_4^2)^1) = f(\omega_4^2) \leftarrow f_e((\omega_4^2)^{2k}) + (\omega_4^2)^k * f_o((\omega_4^2)^{2k}) =$$

$$f_e(\omega_4^4) + (\omega_4^2)^1 * f_o(\omega_4^4) = f_e(1) + (\omega_4^2)*f_o(1) = 3+(-1)*4 = -1$$

ונחזיר $(f(1), f(\omega_4^2)) = (7,-1)$.

<u>בקריאה הרקורסיבית השנייה נקבל:</u>

$(f_e(1)) \leftarrow FFT((2), \omega_4^4)$

$(f_o(1)) \leftarrow FFT((1), \omega_4^4)$

ומהן נגיע לתנאי העצירה שוב ונחזיר $f_e(1)=2$ ו-$f_o(1)=1$.

כעת כאשר האלגוריתם קיבל תשובה ל-2 הקריאות הרקורסיביות ל-FFT שלעיל נתחיל בריצת לולאת ה-for מ-k=0 עד n-1 (1 = 2-1).

- עבור k=0:

$f((\omega_4^2)^k) = f((\omega_4^2)^0) = f(1) \leftarrow f_e((\omega_4^2)^{2k}) + (\omega_4^2)^k * f_o((\omega_4^2)^{2k}) = f_e(1) + f_o(1) = 2+1 = 3$

- עבור k=1:

$f((\omega_4^2)^k) = f((\omega_4^2)^1) = f(\omega_4^2) \leftarrow f_e((\omega_4^2)^{2k}) + (\omega_4^2)^k * f_o((\omega_4^2)^{2k}) = f_e(\omega_4^4) +$

$(\omega_4^2)^1 * f_o(\omega_4^4) = f_e(1) + (\omega_4^2)*f_o(1) = 2+(-1)*1 = 1$

ונחזיר $(f(1), f(\omega_4^2)) = (3,1)$.

<u>כעת לאחר ש-2 הקריאות הרקורסיביות ל-FFT סיימו את ריצתן נחזור לקריאה הראשונה ל-FFT:</u>

קיבלנו ש:

$(f_e(1), f_e(\omega_4^2)) \leftarrow (7,-1)$

$(f_o(1), f_o(\omega_4^2)) \leftarrow (3,1)$

נתחיל בריצת לולאת ה-for מ-k=0 עד n-1 (3 = 4-1).

- עבור k=0:

$f((\omega_4)^k) = f((\omega_4)^0) = f(1) \leftarrow f_e((\omega_4)^{2k}) + (\omega_4)^k * f_o((\omega_4)^{2k}) = f_e(1) + f_o(1) = 7+3 = \mathbf{10}$

- עבור k=1:

$f((\omega_4)^k) = f((\omega_4)^1) = f(\omega_4) \leftarrow f_e((\omega_4)^{2k}) + (\omega_4)^k * f_o((\omega_4)^{2k}) = f_e(\omega_4)^2 + (\omega_4)^1 * f_o(\omega_4)^2 =$
$f_e(\omega_4^2) + (\omega_4)*f_o(\omega_4^2) = -1 + i*1 = \mathbf{-1 + i}$

- עבור k=2:

$f((\omega_4)^k) = f((\omega_4)^2) = f(\omega_4^2) \leftarrow f_e((\omega_4)^{2k}) + (\omega_4)^k * f_o((\omega_4)^{2k}) = f_e(\omega_4)^4 + (\omega_4)^2 * f_o(\omega_4)^4 =$
$f_e(\omega_4^4) + (\omega_4^2)*f_o(\omega_4^4) = f_e(1) + (-1)*f_o(1) = 7 - 3 = \mathbf{4}$

- עבור k=3:

$f((\omega_4)^k) = f((\omega_4)^3) = f(\omega_4^3) \leftarrow f_e((\omega_4)^{2k}) + (\omega_4)^k * f_o((\omega_4)^{2k}) =$

$f_e((\omega_4)^{2(k-n/2)}) + (\omega_4)^k * f_o((\omega_4)^{2(k-n/2)}) = f_e((\omega_4)^2) + (\omega_4)^3 * f_o((\omega_4)^2) = -1 + (-i)*1 =$

$\mathbf{-1 - i}$

ולסיכום האלגוריתם יחזיר:

$\mathbf{(f(1), f(\omega_4), f(\omega_4^2), f(\omega_4^3)) = (10,-1+i,4,-1-i)}$

ב. נציג כעת את ריצת האלגוריתם FFT על הוקטור שהתקבל בסעיף הקודם עם הפרמטר $.(\omega_4)^{-1}$

<u>בריצה הראשונה של FFT נקבל את הקריאות הרקורסיביות הבאות:</u>

$(f_e(1), f_e(\omega_4^{-2})) \leftarrow FFT((10, 4), \omega_4^{-2})$

$(f_o(1), f_o(\omega_4^{-2})) \leftarrow FFT((-1+i,-1-i), \omega_4^{-2})$

<u>לאחר מכן בקריאה הרקורסיבית הראשונה נקבל:</u>

$(f_e(1)) \leftarrow FFT((10), \omega_4^{-4})$

$(f_o(1)) \leftarrow FFT((4), \omega_4^{-4})$

ומהן נגיע לתנאי העצירה כי n=1 ואז נחזיר $f_e(1)=10$ ו-$f_o(1)=4$.

כעת כאשר האלגוריתם קיבל תשובה ל-2 הקריאות הרקורסיביות לFFT שלעיל נתחיל בריצת לולאת ה-for מ-k=0 עד n-1 (1 = 2-1).

- עבור k=0:

$$f(1) \leftarrow f_e((\omega_4^{-2})^{2k}) + (\omega_4^{-2})^k * f_o((\omega_4^{-2})^{2k}) = f_e(1) + 1*f_o(1) = 10 +4 = 14$$

- עבור k=1:

$$f(\omega_4^{-2}) \leftarrow f_e((\omega_4^{-2})^{2k}) + (\omega_4^{-2})^k * f_o((\omega_4^{-2})^{2k}) =$$

$$f_e(\omega_4^{-4}) + (\omega_4^{-2})^1 * f_o(\omega_4^{-4}) = f_e(1) + (\omega_4^{-2})*f_o(1) = 10+(-1)*(4) = 6$$

ונחזיר $.(f(1), f(\omega_4^{-2})) = (14,6)$

<u>בקריאה הרקורסיבית השנייה נקבל:</u>

$(f_e(1)) \leftarrow FFT(-1+i, \omega_4^{-4})$

$(f_o(1)) \leftarrow FFT(-1-i, \omega_4^{-4})$

ומהן נגיע לתנאי העצירה כי n=1 ואז נחזיר $f_e(1)=-1+i$ ו-$f_o(1)=-1-i$.

כעת כאשר האלגוריתם קיבל תשובה ל-2 הקריאות הרקורסיביות לFFT שלעיל נתחיל בריצת לולאת ה-for מ-k=0 עד n-1 (1 = 2-1).

- עבור k=0:

$$f(1) \leftarrow f_e(1) + 1*f_o(1)) = -1+i -1-i = -2$$

- עבור k=1:

$$f(\omega_4^{-2}) \leftarrow f_e(1) + (\omega_4^{-2})*f_o(1) = -1+i+(-1)*(-1-i) = 2i$$

ונחזיר $.(f(1), f(\omega_4^{-2})) = (-2, 2i)$

<u>כעת לאחר ש-2 הקריאות הרקורסיביות ל-FFT סיימו את ריצתן נחזור לקריאה הראשונה ל-FFT:</u>

קיבלנו ש:

$(f_e(1), f_e(\omega_4^{-2})) \leftarrow (14,6)$

$(f_o(1), f_o(\omega_4^{-2})) \leftarrow (-2, 2i)$

נתחיל בריצת לולאת ה-for מ-k=0 עד n-1 (3 = 4-1).

- עבור k=0:

$$f(1) \leftarrow f_e(1) + 1 * f_o(1) = (14 + -2) = 12$$

- עבור k=1:

$$f((\omega_4)^{-1}) \leftarrow f_e((\omega_4)^{-2}) + (\omega_4)^{-1} * f_o((\omega_4)^{-2}) = 6 + -i*2i = 8$$

- עבור k=2:

$$f((\omega_4)^{-3}) \leftarrow f_e(\omega_4)^{-4} + (\omega_4)^{-2} * f_o(\omega_4)^{-4} = f_e(1) + (-1)*f_o(1) = 14 -1*-2 = 16$$

- עבור k=3:

$$f((\omega_4)^{-4}) \leftarrow f_e((\omega_4)^{-2k}) + (\omega_4)^{-k} * f_o((\omega_4)^{-2k} =$$

$$f_e((\omega_4)^{-2(k-n/2)}) + (\omega_4)^{-k} * f_o((\omega_4)^{-2(k-n/2)}) = f_e((\omega_4)^{-2}) + (\omega_4)^{-3} * f_o((\omega_4)^{-2}) =$$

$$6 + i*2i = 4$$

ולסיכום האלגוריתם יחזיר:

$$\mathbf{(f(1), f(\omega_4^{-1}), f(\omega_4^{-2}), f(\omega_4^{-3})) = (12,8,16,4) = 4*(3,2,4,1)}$$

זוהי הסדרה המקורית שהפעלנו עליה FFT מוכפלת ב-n=4.

__שאלה 2__

כפל מספרים שלמים בגישת FFT:

פתרו את הבעיה בזמן ריצה $\theta(nlog^2n)$.

הציגו אלגוריתם משופר ל-Karatsuba, שמחלק כל קלט ל-(n/k) בלוקים בגודל k.
היעזרו ב-FFT לפתרון תתי-הבעיות המתקבלות.
הניחו לשם פשטות כי ההכפלות שמתבצעות במהלך הקריאות הרקורסיביות אינן מגדילות את אורכם של המספרים, ולכן ניתן לממש הכפלות אלו בצורה תמימה תוך $\theta(k^2)$ פעולות על ביטים. בחרו לבסוף את גודלם של הבלוקים להיות $k = logn$.

__תשובה__

נחלק את המספר a ל-$\frac{n}{k}$ בלוקים בגודל k כך שיתקבל-

$$a = \sum_{i=0}^{\frac{n}{k}-1} \mathrm{a}_i * 2^{ik}$$

נסמן $x = 2^{ik}$ ונקבל פולינום ממעלה $\frac{n}{k} - 1$ בנקודה $x = 2^k$ -

$$a = \sum_{i=0}^{\frac{n}{k}-1} \mathrm{a}_i * x = A(2^k)$$

נתאר את B באותו האופן, נקבל -

$$b = \sum_{i=0}^{\frac{n}{k}-1} \mathrm{b}_i * x = B(2^k)$$

כעת, כשנרצה למצוא את תוצאת המכפלה, נצטרך למצוא –

$$c = a * b = A(2^k) * B(2^k) = C(2^k) = \sum_{i=0}^{\frac{2n}{k}-2} \mathrm{c}_i * 2^{ik}$$

זוהי בעיית כפל פולינומים. עלינו למצוא את מקדמי הפולינום C. נניח לפי הנתון בשאלה שהמקדמים אינם גדולים מ-k סיביות.

כעת נריץ את האלגוריתם הידוע לפתרון כפל פולינומים ואז נציב את הערך 2^k כדי לקבל את c.

<u>אלגוריתם</u>

1. נחלק את שני מספרי הקלט ל-$\frac{n}{k}$ בלוקים בעלי k סיביות לכל מקדם. נקבל את הוקטורים הבאים –
$$a\_vec = \left(a_0, a_1, \dots, a_{\frac{n}{k}-1}\right)$$
$$b\_vec = (b_0, b_1, \dots, b_{\frac{n}{k}-1})$$
2. נריץ FFT על כל אחד מהוקטורים לקבלת $a\_vec\_result, b\_vec\_result$.
3. חישוב $c\_vec\_result = a\_vec\_result * b\_vec\_result$.
4. מציאת מקדמי הפולינום C על ידי ביצוע FFT^{-1} על c_vec_result.
5. הצבת $x = 2^k$ בפולינום C והחזרת התוצאה.

<u>נכונות</u>

הנכונות נובעת מהתיאור המפורט מעלה (לפני האלגוריתם) ומנכונות חישוב מכפלת פולינומים (קונבולוצייה) שנמצא בספר הלימוד.

<u>סיבוכיות זמן</u>

1. חלוקת המספרים לוקטור מקדמים - $O(\frac{n}{k})$.
2. הרצת FFT עבור וקטור מקדמים באורך $\frac{2n}{k}$ כאשר בכל קריאה רקורסיבית מבצעים $\theta(k^2)$ פעולות עבור $\frac{n}{k}$ קבוצות של סיביות.
נוסחת הנסיגה תהיה (נציג $k = logn$):
$$T(n) < 2T\left(\frac{n}{2}\right) + ck^2\frac{n}{k} = 2T\left(\frac{n}{2}\right) + ckn = 2T\left(\frac{n}{2}\right) + cnlogn$$
$$= 2T\left(\frac{n}{2}\right) + O(nlogn)$$
מדובר במקרה ב' המורחב של שיטת האב. הפתרון הוא- $T(n) = \theta(nlog^2n)$.
3. חישוב c_vec_result – לכפול $\frac{n}{k}$ מקדמים כאשר כל מקדם בעל k סיביות, כלומר עלות כל מכפלה היא ck^2 ומתקבל- $\frac{cn}{k}k^2 = cnk = O(nk) = O(nlogn)$.

ולבסוף-
$$T(n) = O\left(\frac{n}{k}\right) + \theta(nlog^2n) + O(nlogn) = \theta(nlog^2n)$$

שאלה 3

❖ הרעיון הכללי

נשים לב שעבור פולינום כללי ממעלה n הנתון ע"י $P(x)=\sum_{k=0}^{n}a_kx^k$ ניתן לקבל נוסחה פשוטה לנגזרת:

$P'(x)=\sum_{k=1}^{n}ka_kx^{k-1}=\sum_{k=0}^{n-1}(k+1)a_{k+1}x^k$ המתקבלת כפולינום ממעלה n-1 הנתון באותה תצורה. בהכללה פשוטה באינדוקציה על מעלת הפולינום ניתן לקבל את הנוסחה הפשוטה והשימושית $P^{(t)}(x)=\sum_{k=t}^{n}\frac{k!}{(k-t)!}a_kx^{k-t}$. ננסה לקבל מנוסחה זו נוסחה שמזכירה צורה של קונבולוציה:

$$\boxed{P^{(t)}(x)=\sum_{k=t}^{n}\frac{k!}{(k-t)!}a_kx^{k-t}=\sum_{k=t}^{n}k!a_k\cdot\frac{x^{k-t}}{(k-t)!}}$$

כעת אם נגדיר את שתי הסדרות $b_k=(n-k)!a_{n-k},c_k=\frac{{x_0}^k}{k!}$ לכל $k=0,...,n$. נשים לב שמתקיים:

$P^{(t)}(x_0)=\sum_{k=t}^{n}k!a_k\cdot\frac{{x_0}^{k-t}}{(k-t)!}=\sum_{k=t}^{n}b_{n-k}\cdot c_{k-t}\underset{def:j=k-t}{=}\sum_{j=0}^{n-t}b_{n-t-j}\cdot c_j=\sum_{i+j=n-t}b_i\cdot c_j=(b*c)_{n-t}$ כאשר $b=\left\langle n!a_n,(n-1)!a_{n-1},\cdots,(0)!a_0\right\rangle,c=\left\langle\frac{{x_0}^0}{0!},\frac{{x_0}^1}{1!},\cdots,\frac{{x_0}^n}{n!}\right\rangle$ הם שני הוקטורים בקונבולוציה שבשוויון האחרון. כלומר, נקבל שהנגזרת ה t-ית ($t=0,...,n$) היא האיבר ה $n-t$ בקונבולוציה $b*c$.

❖ האלגוריתם

[האלגוריתם מקבל את הדרגה n , את המקדמים $a_0,...,a_n$ ואת הנקודה x_0 בה רוצים לחשב את הנגזרות]

(1) תחילה נבנה את הוקטורים b ו c כך:

$k\leftarrow 1,t\leftarrow 1$

עבור $i\leftarrow 0...n$ בצע

$b_{n-i}\leftarrow t\cdot a_i$

$c_i\leftarrow\frac{k}{t}$

$t\leftarrow t\cdot(i+1)$

$k\leftarrow k\cdot x_0$

(2) נחשב את הקונבולוציה $d\leftarrow b*c$ באמצעות FFT

(3) עבור $i\leftarrow 0...n$ בצע

הצג כפלט ש d_{n-i} הוא ערך הנגזרת ה i-ית בנקודה x_0

❖ נכונות האלגוריתם

בלולאה שב(1) באיטרציה ה-i-ית t הוא !i (עד השורה השלישית בלולאה) ו-k הוא x_0^i (עד השורה הרביעית בלולאה). לפיכך $b_{n-i} \leftarrow i!\cdot a_i$ ולכן $b_k \leftarrow (n-k)!\cdot a_{n-k}$ לכל k וכן $c_k \leftarrow \frac{x_0^k}{k!}$ כמו שרצינו להגדיר ברעיון הכללי. כמו שהסברנו ברעיון הכללי שהנגזרת ה-t-ית ($t=0,...,n$) היא האיבר ה-$n-t$ בקונבולוציה $b*c$. ולכן הלולאה שב(3) מוציאה נכונה את הפלט הדרוש.

❖ ניתוח סיבוכיות

ב(1) הלולאה מתבצעת n+1 פעמים וכל איטרציה לוקחת $O(1)$ ולכן סה"כ (1) לוקח $O(n)$. (2) כידוע לוקח $\Theta(n\log n)$. (3) כמובן לוקח גם הוא $O(n)$. לכן בסה"כ האלגוריתם לוקח $\Theta(n\log n)$.

שאלה 4

כאמור המימוש הפשוט של כפל מטריצות הוא ב $\Theta(n^3)$ והוא נגזר מהעובדה שבכל שלב של הרקורסיה אנו מפצלים את כל אחת מהמטריצות לארבע מטריצות קטנות יותר (שזה בסך הכל 8 פעולות כפל, כלומר 8 קריאות רקורסיביות) ומבצעים 4 חיבורים (שכל חיבור הוא בין כל שני איברים במטריצה שזה כמות האיברים במטריצה כלומר n^2) ולכן $T(n) = 8T(n/2) + 4n^2 = \Theta(n^3)$ (לפי שיטת האב שזמן הריצה הוא $n^{\log_b a} = n^{\log_2 8} = \Theta(n^3)$)

האלגוריתם הנתון מצמצם את כמות פעולות הכפל ל7 במקום ל8 בתמורה להעלאה משמעותית של כמות החיבורים והחיסורים (ל18 במקום 4). מכיון ש18 הוא קבוע שמכפיל את פעולת החיבור, ולכן אסימפטוטית הוא לא נחשב, נוסחת הנסיגה שנקבל עכשיו היא $T(n) = 7T(n/2) + 18n^2 = n^{\log_2 7} = \Theta(n^{2.81})$ מ.ש.ל.