שאלה 1

א

$$P(x) = x^3 + 2x^2 - 3x - 1 \Longrightarrow (-1, -3, 2, 1)$$

נעבור לצורה וקטורית

$FFT((-1,-3,2,1), w=i)$

$P(i^0) = P(1) = P_e(1^2) + 1 \cdot P_o(1^2) = P_e(1) + P_o(1) = 1 + (-2) = -1$

$P(i^1) = P(i) = P_e(i^2) + i\,P_o(i^2) = P_e(-1) + i\,P_o(-1) = -3 + i(-4) = -3 - 4i$

$P(i^2) = P(-1) = P_e((-1)^2) - P_o((-1)^2) = P_e(1) - P_o(1) = 1 - (-2) = 3$

$P(i^3) = P(-i) = P_e((-i)^2) - i\,P_o((-i)^2) = P_e(-1) - i\,P_o(-1) = -3 - i(-4) =$
$= -3 + 4i$

$\rightarrow FFT((-1,2), w^2 = -1)$

$P((-1)^0) = P(1) = -1 + 2 \cdot 1 = 1$

$P((-1)^1) = P(-1) = -1 + 2 \cdot (-1) = -3$

$\rightarrow FFT((-3,1), w^2 = -1)$

$P((-1)^0) = P(1) = -3 + 1 \cdot 1 = -2$

$P((-1)^1) = P(-1) = -3 + 1 \cdot (-1) = -4$

$:w^{-1} = -i$

ב

$FFT((-1, -3-4i, 3, -3+4i), w^{-1} = -i)$

$P((-i)^0) = P(1) = P_e(1^2) + 1 \cdot P_o(1^2) = P_e(1) + P_o(1) = 2 + (-6) = -4$

$P((-i)^1) = P(-i) = P_e((-i)^2) - i\,P_o((-i)^2) = P_e(-1) - i\,P_o(-1) = -4 - i(-8i) = -12$

$P((-i)^2) = P(-1) = P_e((-1)^2) - P_o((-1)^2) = P_e(1) - P_o(1) = 2 - (-6) = 8$

$P((-i)^3) = P(i) = P_e((i)^2) + i\,P_o((i^2)) = P_e(-1) + i\,P_o(-1) = -4 + i(-8i) = 4$

$\rightarrow FFT((-1,3), (w^{-1})^2 = (-i)^2 = -1)$

$P((-1)^0) = P(1) = -1 + 3 \cdot 1 = 2$

$P((-1)^1) = P(-1) = -1 + 3 \cdot (-1) = -4$

$\rightarrow FFT((-3-4i, -3+4i), (w^{-1})^2 = -1)$

$P((-1)^0) = P(1) = -3 - 4i + (-3+4i) \cdot 1 = -6$

$P((-1)^1) = P(-1) = -3 - 4i + (-3+4i) \cdot (-1) = -8i$

לא ציינת איזה שורש יחידה מאיזה סדר צריך לבחור
30\30

שאלה 2

פתרון:

רעיון:

נניח שאנחנו עובדים בבסיס 2 ונתונים 2 מספרים x ו-y בני n סיביות ויהי k טבעי כך ש-k מחלק את n. נפרק את x ו-y לסכומים חלקיים באופן הבא:

$$x = x_{\frac{n}{k}-1}\cdot 2^{n-k} + x_{\frac{n}{k}-2}\cdot 2^{n-2k} + \dots + x_1^{k} + x_0$$

$$y = y_{\frac{n}{k}-1}\cdot 2^{n-k} + y_{\frac{n}{k}-2}\cdot 2^{n-2k} + \dots + y_1^{k} + y_0$$

כלומר $x_{\frac{n}{k}-1}$ מייצג את k הסיביות 'השמאליות ביותר', $x_{\frac{n}{k}-2}$ מייצג את k הסיביות 'היותר ימניות' וכן הלאה... כאשר x_0 מייצג את k הסיביות 'הימניות ביותר'. ייצוג דומה עבור y.

$$x\cdot y = \left(x_{\frac{n}{k}-1}\cdot 2^{n-k} + x_{\frac{n}{k}-2}\cdot 2^{n-2k} + \dots + x_0\right)\cdot\left(y_{\frac{n}{k}-1}\cdot 2^{n-k} + y_{\frac{n}{k}-2}\cdot 2^{n-2k} + \dots + y_0\right)$$

שזה שקול לחישוב הקונבולוציה $(x_0, x_1, \dots, x_{\frac{n}{k}-1}) * (y_0, y_1, \dots, y_{\frac{n}{k}-1})$

לכן המטרה לחשב את הקונבולוציה.

אלגוריתם:

קלט: שני מספרים בינאריים בני n סיביות (נניח גם ש-$k|n$ כאשר $k=\log n$ ו-n חזקה של 2).

שלבי האלגוריתם:

1. represent:
$$x = x_{\frac{n}{k}-1}\cdot 2^{n-k} + x_{\frac{n}{k}-2}\cdot 2^{n-2k} + \dots + x_1^{k} + x_0;\quad y = y_{\frac{n}{k}-1}\cdot 2^{n-k} + y_{\frac{n}{k}-2}\cdot 2^{n-2k} + \dots + y_1^{k} + y_0$$

2. $(x_0', x_1', \dots, x'_{2(\frac{n}{k}-1)}) \leftarrow FFT((x_0, \dots, x_{\frac{n}{k}-1}), \omega)$
$(y_0', y_1', \dots, y'_{2(\frac{n}{k}-1)}) \leftarrow FFT((y_0, \dots, y_{\frac{n}{k}-1}), \omega)$

3. for $j \leftarrow 0, \dots, 2\cdot(\frac{n}{k}-1)$ do
$z_j \leftarrow x_j' \cdot y_j'$

4. $(z_0', z_1', \dots, z'_{2(\frac{n}{k}-1)}) \leftarrow FFT((z_0, z_1, \dots, z_{2(\frac{n}{k}-1)}), \omega^{-1})$
(נניח גם שהמקדמים בחישוב כבר מתוקנים)

5. return $z'_{2(\frac{n}{k}-1)}\cdot 2^{2n-2k} + z'_{2(\frac{n}{k}-1)-1}\cdot 2^{2n-3k} + \dots + z_1\cdot 2^k + z_0$

שאלה 2

הוכחת נכונות:

נתבונן בכפל:

$$x\cdot y=\left(x_{\frac{n}{k}-1}\cdot 2^{n-k}+x_{\frac{n}{k}-2}\cdot 2^{n-2k}+\ldots+x_1\cdot 2^k+x_0\right)\cdot\left(y_{\frac{n}{k}-1}\cdot 2^{n-k}+y_{\frac{n}{k}-2}\cdot 2^{n-2k}+\ldots+y_1\cdot 2^k+y_0\right)=$$

$$=\left(x_{\frac{n}{k}-1}\cdot y_{\frac{n}{k}-1}\right)\cdot 2^{2n-2k}+\left(x_{\frac{n}{k}-1}\cdot y_{\frac{n}{k}-2}+x_{\frac{n}{k}-2}\cdot y_{\frac{n}{k}-1}\right)\cdot 2^{2n-3k}+\ldots+x_0\cdot y_0 \quad (*)$$

מכאן נובע שעלינו למצוא את המקדמים של $(*)$ ואז נוכל להרכיב את המספר $x\cdot y$.
נשים לב שהמקדמים שמתקבלים ב-$(*)$ הם בדיוק אותם המקדמים שמתקבלים מכפל פולינומים:

$$x(t)=x_{\frac{n}{k}-1}\,t^{\frac{n}{k}-1}+x_{\frac{n}{k}-2}\cdot t^{\frac{n}{k}-2}+\ldots+x_1 t+x_0$$

$$y(t)=y_{\frac{n}{k}-1}\,t^{\frac{n}{k}-1}+y_{\frac{n}{k}-2}\cdot t^{\frac{n}{k}-2}+\ldots+y_1 t+y_0$$

ולכן עלינו למצוא את וקטור הקונבולוציה:

$$(x_0,x_1,\ldots,x_{\frac{n}{k}-1})*(y_0,y_1,\ldots,y_{\frac{n}{k}-1})$$

במהלך השלבים 2, 3 ו-4 האלגוריתם הנ"ל מחשב את וקטור הקונבולוציה ונכונות החישוב נובעת מנכונות האלג' FFT (נציין גם שניתן שניתן להניח כי ההכפלות שמתבצעות במהלך הקריאות הרקורסיביות אינן מגדילות את אורכם של המספרים).
לאחר שיש בידינו את המקדמים ניתן להרכיב את תוצאת הכפל לפי $(*)$ וזה מה שמבצע האלג' הנ"ל בשלב 5.

ניתוח זמן ריצה:

נסמן $m=\frac{n}{k}$. ניתן להציג את המספרים משלב 1 בזמן $O(n)$.
לפי FFT (תוך כדי שגם מתחשבים בגודל ייצוג המקדמים ובמקרה שלנו מקדמים בגודל k ולכן ניתן לממש הכפלות בין שני מקדמים בזמן $\Theta(k^2)$)
שלבים 2 ו-4 מתבצעים בזמן $O(k^2 m\log m)$. בשלב 3 מתבצעות $O(m)$ איטרציות ובכל איטרציה מתבצעות $\Theta(k^2)$ פעולות פרימיטיביות (כלומר כל איטרציה עורכת זמן $\Theta(k^2)$).

בסה"כ מתבצעות:

$$O(n)+O(k^2 m\log m)+\Theta(k^2)\cdot O(m)=O(n)+O(k^2 m)+O(k^2 m\log m)=$$

$$=O(n)+O(k^2 m\log m)\underset{k=\log n \text{ עבור}}{=}O(n)+O\left(\log^2 n\cdot\frac{n}{\log n}\cdot\log\left(\frac{n}{\log n}\right)\right)=$$

$$=O(n)+O(\log n\cdot n\cdot(\log n-\log\log n))=O(n)+O(n\log^2 n)=$$

$$=O(n\log^2 n)$$

מעולה
30/30

שאלה 3

פתרון:

נסמן $u=(a_n, \frac{1}{n}a_{n-1}, \frac{1}{n-1}\cdot\frac{1}{n}a_{n-2}, \dots, a_0)$; $V=(n!, 1\cdot n!\cdot x_0, \frac{1}{2}\cdot 1\cdot n!\cdot x_0^2, \frac{1}{3}\cdot\frac{1}{2}\cdot 1\cdot n!\cdot x_0^3, \dots, x_0^n)$

ראשית בעזרת n פעולות כפל נוכל לחשב $n!$. בהינתן מקדמי הפולינום f ונקודה x_0 נוכל לבנות בעזרת $O(n)$ איטרציות של לולאה את הוקטור V ובאופן דומה את u ונשים לב שלפי הגדרת הקונבולוציה מתקיים $u*V=(n!a_n, (n-1)!a_{n-1}+n!a_n x_0, \dots, a_0+\dots+a_n x_0^n, \dots)=(f^{(n)}(x_0), f^{(n-1)}(x_0), \dots, f^{(0)}(x_0)\dots)$

כלומר $n+1$ הכניסות הראשונות של וקטור הקונבולוציה הן בדיוק הנגזרות בנקודה x_0. נוכל להשתמש ב-FFT לחישוב הקונבולוציה $u*V$ ולהחזיר $n+1$ הכניסות הראשונות. נכונות נובעת מוקטור הקונבולוציה ונכונות FFT וזמן ריצה כולל הוא $\Theta(n)+\Theta(n\log n)=\Theta(n\log n)$