חישוב אלגברי ואלגוריתמים אלגבריים

בן לי וולק

תוכן העניינים

1 מבוא אלגברי 3

1 שדות 3

2 דוגמאות לשדות סופיים 4

2 כפל מטריצות 6

3 הקדמה 6
3.1 האלגוריתם של שטראסן 6
3.2 מודל החישוב האלגברי 7
3.2.1 דיון במודל החישוב 9
3.2.2 האקספוננט של כפל מטריצות 9

4 פונקציות בילינאריות, טנזורים ודרגה 9
4.1 פונקציות בילינאריות 9
4.2 אלגוריתמים בילינאריים 11
4.3 דרגה של פונקציות בילינאריות ושל טנזור 12

5 פעולות על טנזורים 15
5.1 פרמוטציות 15
5.2 פרמוטציות על הטנזור של כפל מטריצות 16
5.3 כפל טנזורים 17

6 דרגת גבול 19
6.1 קירובים של טנזורים 19
6.2 קירובים של טנזור כפל המטריצות 20
6.3 מחישוב מקורב לחישוב מדוייק 23
6.4 חסם משופר על ω 24
6.5 מה הלאה? 25

3 אלגוריתמים על פולינומים 26

7 בעיית כפל הפולינומים 26

7.1 ייצוג בעזרת שערוכים 26
7.2 שערוך היא פעולה לינארית 27
7.3 שורשי היחידה 28
7.4 חישוב טרנספורם פורייה הדיסקרטי 30
7.5 כפל פולינומים מהיר 31
7.6 חסם תחתון לחישוב DFT 31
7.7 כפל מספרים שלמים 34

8 אלגוריתמים מתקדמים 34

8.1 חלוקת פולינומים 34
8.2 שערוך פולינום במספר נקודות 37
8.3 פירוק לגורמים של מספרים שלמים 38
8.4 אינטרפולציה 40
8.5 הרכבה מודולרית 41

4 בדיקת ראשוניות 43

9 מבחני ראשוניות 43

9.1 משפט פרמה 43
9.2 וריאציה פולינומית על מבחן פרמה 44

10 ארגז כלים אלגברי 45

10.1 אריתמטיקה מודולרית של פולינומים 45
10.2 סדר כפלי 45

11 אלגוריתם AKS לבדיקת ראשוניות 46

11.1 פירוק לגורמים של פולינומים 47
11.2 פולינומים ציקלוטומיים ושורשי יחידה פרימיטיביים 48
11.3 הוכחת נכונות האלגוריתם 49

חלק 1: מבוא אלגברי

1 שדות

בחלק זה של הקורס נגדיר את מושג ה**שדה** ונראה מספר דוגמאות.
שדה הוא מבנה אלגברי אבסטרקטי בעל תכונות מועילות. בקורס אלגברה לינארית נתקלתם בשדה המספרים הממשיים אולם את כל התורה שלמדתם בקורס ניתן ללמד מעל כל שדה.

הגדרה 1.1. שדה היא קבוצה $\mathbb{F}$ עליה מוגדרות שתי פעולות בינאריות הנקראות ״חיבור״ $(+)$ ו״כפל״ ($\cdot$ או $\times$) המקיימות את התכונות הבאות:

1. חוק הקיבוץ (אסוציאטיביות) ביחס החיבור: לכל $a, b, c \in \mathbb{F}$, $(a+b)+c = a+(b+c)$.
2. קיום איבר אדיש לחיבור: קיים בשדה איבר המסומן 0 כך שלכל $a \in \mathbb{F}$, $a+0=a$.
3. קיום איבר הופכי: לכל $a \in \mathbb{F}$ קיים איבר $b \in \mathbb{F}$ כך ש־$a+b=0$. b נקרא האיבר הנגדי של a ומסומן $-a$.
4. חוק החילוף (קומוטטיביות) ביחס לחיבור: לכל $a, b \in \mathbb{F}$, $a+b=b+a$.
5. חוק הקיבוץ ביחס לכפל: לכל $a, b, c \in \mathbb{F}$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
6. קיום איבר אדיש לכפל: קיים בשדה איבר המסומן 1 כך ש־$1 \neq 0$ ולכל $a \in \mathbb{F}$, $a \cdot 1 = a$.
7. קיום איבר הופכי: לכל $a \in \mathbb{F}$ כך ש־$a \neq 0$ קיים איבר $b \in \mathbb{F}$ כך ש־$a \cdot b = 1$. b נקרא האיבר ההופכי של a ומסומן a^{-1}.
8. חוק החילוף ביחס לכפל: לכל $a, b \in \mathbb{F}$, $a \cdot b = b \cdot a$.
9. חוק הפילוג (דיסטריביוטיביות): לכל $a, b, c \in \mathbb{F}$, $a \cdot (b+c) = a \cdot b + a \cdot c$.

המספרים הממשיים הם שדה ביחס לפעולות $+$ ו־$\cdot$ ה״רגילות״. כך גם המספרים המרוכבים והמספרים הרציונליים.

שאלה 1.2. האם המספרים הטבעיים $\mathbb{N} = \{0, 1, 2, \ldots\}$ הם שדה?

שאלה 1.3. האם המספרים השלמים $\mathbb{Z} = \{0, \pm 1, \pm 2, \ldots\}$ הם שדה?

דוגמה 1.4. תהי $\mathbb{F} = \{0, 1\}$ עם הפעולות חיבור וכפל המוגדרות באופן הבא:

$+$	0	1
0	0	1
1	1	0

$\times$	0	1
0	0	0
1	0	1

אזי $\mathbb{F}$ הוא שדה בעל שני איברים המסומן לרוב $\mathbb{F}_2$.

משפט 1.5. האיבר 0 בשדה הוא יחיד.

הוכחה: נניח כי קיימים שני איברים אדישים לחיבור, 0_1 ו־0_2. אזי:

$$0_2 = 0_1 + 0_2 = 0_2 + 0_1 = 0_1$$

(השתמשנו בחוק החילוף ובאדישות של האיברים $0_1, 0_2$). □

באותו אופן ניתן להוכיח כי האיבר 1 הוא יחיד.

משפט 1.6. לכל $a \in \mathbb{F}$, $a \cdot 0 = 0$.

הוכחה: כיוון ש־$0 + 0 = 0$, נקבל בעזרת חוק הפילוג:

$$a \cdot 0 = a \cdot (0+0) = a \cdot 0 + a \cdot 0$$

נסמן $b = a \cdot 0$. כלומר, קיבלנו

$$b = b + b.$$

נחבר לשני הצדדים את האיבר $-b$ ונקבל

$$0 = b + (-b) = b + b + (-b) = b + 0 = b.$$

(השתמשנו בחוק הפילוג ובקיום איבר נגדי) □

משפט 1.7. לכל $a, b \in \mathbb{F}$, אם $a \cdot b = 0$ אז $a = 0$ או $b = 0$.

הוכחה: נניח כי $a \cdot b = 0$ ו־$a \neq 0$. נכפיל את המשוואה ב־a^{-1} ונקבל

$$b = 1 \cdot b = (a^{-1} \cdot a) \cdot b = a^{-1} \cdot (a \cdot b) = a^{-1} \cdot 0 = 0.$$

באותו אופן ניתן לטפל במקרה $b \neq 0$. □

2 דוגמאות לשדות סופיים

כעת נראה קבוצה רחבה של דוגמאות לשדות שהם קבוצה סופית. שדות אלו הם שימושיים ביותר במדעי המחשב. לכל מספר טבעי $n \geq 2$ נסמן $\mathbb{Z}_n = \{0, 1, \ldots n-1\}$. נגדיר עליהם פעולת חיבור וכפל ע״י חיבור וכפל **מודולו** n. כלומר

$$a + b = (a+b) \mod n.$$

כלומר: תחילה נחשב את $a + b$ כמספר שלם. כעת, $a + b \mod n$ הוא שארית חלוקת $a + b$ ב־n.
שימו לב שבמשוואה לעיל, פעולת החיבור בצד שמאל היא ב־$\mathbb{Z}_n$ ובצד ימין היא פעולת החיבור ה״רגילה״ של מספרים שלמים.
באותו אופן

$$a \cdot b = (a \cdot b) \mod n.$$

כלומר: שארית חלוקת $a \cdot b$ ב־n.

תרגיל 2.1. לכל n, הפעולות הנ״ל הן קומוטטיביות, אסוציאטיביות ומקיימות את חוק הפילוג. בנוסף, לכל איבר יש איבר נגדי. האיבר הנגדי של 0 הוא 0 והאיבר הנגדי של $a \neq 0$ הוא $n - a$.

שאלה 2.2. לאיזה ערכים של n מתקיים ש־$\mathbb{Z}_n$ הוא שדה?

נשים לב שהשאלה שקולה לשאלה לאיזה ערכים של n מתקיים שלכל איבר ב־$\mathbb{Z}_n$ יש הופכי כפלי.
האם $\mathbb{Z}_2$ הוא שדה? מסתבר שזאת בדיוק הדוגמה שראינו קודם לשדה בן שני איברים.
האם $\mathbb{Z}_6$ הוא שדה? שימו לב שב־$\mathbb{Z}_6$ מתקיים ש־$2 \cdot 3 = 0$ אבל $2, 3 \neq 0$ וראינו שבשדה זה לא יכול להתקיים. אז $\mathbb{Z}_6$ אינו שדה.

משפט 2.3. $\mathbb{Z}_n$ הוא שדה אם ורק אם n הוא מספר ראשוני.

לצורך ההוכחה נשתמש בלמה היסודית הבאה על מספרים ראשוניים:

למה 2.4. יהי p מספר ראשוני ו־a, b מספרים שלמים. אם $a \cdot b$ מתחלק ב־p אז a מתחלק ב־p או b מתחלק ב־p.

הוכחת משפט 2.3: אם n פריק אז $n = mk$ כאשר $m, k < n$ ולכן, כמו בדוגמה של $\mathbb{Z}_6$ שראינו, $m \cdot k = 0$ ב־$\mathbb{Z}_n$ ו־$m, k \neq 0$ ולכן $\mathbb{Z}_n$ אינו שדה.
נשאר לטפל במקרה ש־n ראשוני ולהראות שבמקרה זה קיים איבר הופכי. יהי $a \in n\mathbb{Z}_n$ כך ש־$a \neq 0$. נביט בפונקציה

$$f_a : \mathbb{Z}_n \to \mathbb{Z}_n, \quad f_a(x) = a \cdot x.$$

נטען שהפונקציה חח"ע. כיוון ש־$\mathbb{Z}_n$ קבוצה סופית, הפונקציה f_a היא עַל, ולכן קיים b כך ש־$f_a(b) = 1$ כלומר $b = a^{-1}$.
נותר אם כן להוכיח ש־f_a פונקציה חח"ע. נניח כי קיימים $x_1, x_2 \in \mathbb{Z}_n$ שונים כך ש־

$$a \cdot x_1 = f_a(x_1) = f_a(x_2) = a \cdot x_2$$

נניח שכמספרים שלמים, $x_1 < x_2$. כלומר, כמספרים שלמים, $a \cdot x_1$ ו־$a \cdot x_2$ נותנים אותה שארית חלוקה ב־n, ולכן $a \cdot (x_2 - x_1)$ מתחלק ב־n. מלמה 2.4 נסיק כי a מתחלק ב־n או $x_2 - x_1$ מתחלק ב־n אבל זה לא ייתכן כיוון ש־n מספר ראשוני. □

חלק 2: כפל מטריצות

פרק זה בקורס יתבסס על מאמר הסקירה Fast Matrix Multiplication מאת Markus Bläser הזמין בקישור הזה.

3 הקדמה

בהינתן שתי מטריצות A, B בגודל $n \times n$ מעל שדה $\mathbb{F}$ נרצה לחשב את מכפלתן $C = AB$. כידוע,

$$.C_{i,j} = \sum_{k=1}^{n} A_{i,k} B_{k,j}$$

לפיכך, כל תא במטריצה C ניתן לחישוב בעזרת n פעולות כפל ו־$n-1$ פעולות חיבור. כיוון שישנם n^2 תאים במטריצה, ניתן לחשב את כל המטריצה C בעזרת n^3 פעולות כפל ו־$n^2 \cdot (n-1)$ פעולות חיבור.

3.1 האלגוריתם של שטראסן

בשנת 1969 המתמטיקאי שטראסן מצא אלגוריתם לכפל של מטריצות 2×2 בעזרת 7 פעולות כפל ו־18 פעולות חיבור. על מנת לחשב את:

$$\begin{pmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix}$$

נחשב את 7 המכפלות הבאות:

$$\begin{aligned} p_1 &= (a_{1,1} + a_{2,2}) \cdot (b_{1,1} + b_{2,2}) \\ p_2 &= (a_{2,1} + a_{2,2}) \cdot b_{1,1} \\ p_3 &= a_{1,1}(b_{1,2} - b_{2,2}) \\ p_4 &= a_{2,2}(-b_{1,1} + b_{2,1}) \\ p_5 &= (a_{1,1} + a_{1,2}) b_{2,2} \\ p_6 &= (-a_{1,1} + a_{2,1})(b_{1,1} + b_{1,2}) \\ p_7 &= (a_{1,2} - a_{2,2})(b_{2,1} + b_{2,2}) \end{aligned}$$

ונשים לב כי

$$.\begin{pmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{pmatrix} = \begin{pmatrix} p_1 + p_4 - p_5 + p_7 & p_3 + p_5 \\ p_2 + p_4 & p_1 + p_3 - p_2 + p_6 \end{pmatrix}$$

מה התועלת בחסכון של פעולת כפל אחת אם אנו נאלצים להוסיף כל כך הרבה פעולות חיבור נוספות? מתברר שהחסכון בפעולת כפל מתבטא כשאנו מפעילים את האלגוריתם של שטראסן באופן **רקורסיבי**. זה נעשה ע״י חלוקת המטריצות A ו־B לארבע בלוקים בגודל $(n/2) \times (n/2)$ ושימוש רקורסיבי באלגוריתם של שטראסן על מנת להכפיל את הבלוקים. האינטואיציה היא שפעולות החיבור של תת־מטריצות הן ״זולות״ יותר מאשר פעולות כפל של תת־מטריצות, ולכן החסכון בפעולת כפל אחת מצטבר.

משפט 3.1. ניתן לחשב את פעולת הכפל של שתי מטריצות $n \times n$ בעזרת $O(n^{\log_2 7}) = O(n^{2.81\ldots})$ פעולות חשבון.

הוכחה: נניח בלי הגבלת הכלליות כי $n = 2^k$ עבור מספר טבעי k (אם זה לא המצב, ניתן לרפד את המטריצה באפסים).
נוכיח, באינדוקציה על k, כי ניתן להכפיל שתי מטריצות בגודל $2^k \times 2^k$ בעזרת 7^k פעולות כפל ו־$6 \cdot (7^k - 4^k)$ פעולות חיבור.
בסיס האינדוקציה, $k = 1$, הוא האלגוריתם של שטראסן שתואר לעיל.
עבור צעד האינדוקציה, נחלק את המטריצות A, B לבלוקים

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix}$$

בגודל $2^{k-1} \times 2^{k-1}$ כל אחד.
ניתן להכפיל את הבלוקים הללו בעזרת 7 פעולות כפל ו־18 פעולות חיבור של מטריצות $2^{k-1} \times 2^{k-1}$.
נספור, בעזרת הנחת האינדוקציה, את מספר הפעולות הכולל.

7 פעולות הכפל של מטריצות $2^{k-1} \times 2^{k-1}$: מהנחת האינדוקציה, כל פעולת כפל של מטריצות כאלה ניתן לבצע בעזרת 7^{k-1} כפלים, ולכן מספר פעולות הכפל הכולל הוא $7 \cdot 7^{k-1} = 7^k$.
עוד מהנחת האינדוקציה, מספר פעולות החיבור שניתן בעזרתם לכפול את המטריצות הוא $7 \cdot 6 \cdot (7^{k-1} - 4^{k-1})$.

18 פעולות החיבור של מטריצות $2^{k-1} \times 2^{k-1}$: את אלו ניתן לבצע בעזרת $18 \cdot (2^{k-1})^2$ פעולות חיבור, בעזרת האלגוריתם הנאיבי שמחבר איבר־איבר.
לפיכך, מספר פעולות הכפל הכולל הוא 7^k, ומספר פעולות החיבור הכולל הוא

$$7 \cdot 6 \cdot (7^{k-1} - 4^{k-1}) + 18 \cdot (2^{k-1})^2 = 6 \cdot (7^k - 7 \cdot 4^{k-1} + 3 \cdot 4^{k-1}) = 6 \cdot (7^k - 4^k).$$

לסיום נשים לב כי $7^k = n^{\log_2 7}$. □

נעיר כי בהמשך הקורס נוכיח פורמלית משפט האומר כי מספיק לספור את מספר פעולות הכפל על מנת לקבל הערכה הדוקה למדי על מספר פעולות החשבון הכולל הדרוש לכפל מטריצות. כלומר, ניתן היה להימנע ממרבית פרטי החישוב המייגעים בהוכחת המשפט.

3.2 מודל החישוב האלגברי

בניתוח האלגוריתם של שטראסן ספרנו אך ורק את מספר פעולות החשבון, והתייחסנו לכל פעולת חיבור, חיסור או כפל כיחידת חישוב "אטומית". כעת נגדיר באופן פורמלי את מודל החישוב האלגברי שאותו ננתח.
תהא $X_1, \ldots, X_m$ קבוצת משתנים, ו־$f_1, \ldots, f_t$ פולינומים במשתנים $X_1, \ldots, X_m$ מעל שדה $\mathbb{F}$ (במקרה של כפל מטריצות, $m = 2n^2$, $t = n^2$).

הגדרה 3.2. **אלגוריתם אלגברי** לחישוב $f_1, \ldots f_t$ הוא רשימה ממוספרת $P_1, \ldots, P_s$ כאשר כל פריט ברשימה הוא אחד מהבאים:

1. משתנה מהקבוצה $X_1, \ldots, X_m$

2. קבוע $c \in \mathbb{F}$

3. צירוף לינארי של שני פריטים קודמים ברשימה: כלומר $P_k = \alpha P_i + \beta P_j$ כאשר $\alpha, \beta \in \mathbb{F}$ ו־$i, j < k$.

4. מכפלה של שני פריטים קודמים ברשימה: כלומר $P_t = P_i \cdot P_j$ כאשר $i, j < k$.

ובנוסף, לכל $1 \le k \le t$ קיים $1 \le j \le s$ כך ש־ $P_j = f_k$.

דוגמה 3.3. נציג שני אלגוריתמים אלגבריים לחישוב $f = X_1^2 - X_2^2$. הראשון:

1. $P_1 = X_1$
2. $P_2 = X_1 \cdot X_1 = X_1^2$
3. $P_3 = X_2$
4. $P_4 = X_2 \cdot X_2 = X_2^2$
5. $P_5 = P_2 - P_4 = X_1^2 - X_2^2 = f$

השני:

1. $P_1 = X_1$
2. $P_2 = X_2$
3. $P_3 = X_1 + X_2$
4. $P_4 = X_1 - X_2$
5. $P_5 = P_3 \cdot P_4 = (X_1 + X_2) \cdot (X_1 - X_2) = X_1^2 - X_2^2 = f$

כלומר, אלגוריתם אלגברי מחשב פונקציה פולינומית במשתני הקלט באופן טבעי בעזרת פעולות כפל וחיבור. כל צעד באלגוריתם מורכב מקריאה של משתנה קלט או מכפל או חיבור של שני ביטויים שכבר חושבו. נשים לב שלא הגדרנו מפורשות את הפלט של האלגוריתם. בתור מוסכמה נאמר שפלט האלגוריתם הוא השורה האחרונה בו (או k השורות האחרונות, במקרה של חישוב קבוצת פונקציות).
כך לדוגמה, אלגוריתם לחישוב כפל של שתי מטריצות בגודל $n \times n$, X, Y, הוא רשימה של s פעולות חשבון אשר n^2 האחרונות מתוכן נדרשות להיות התאים של המטריצה XY.

עובדה 3.4. האלגוריתם של שטראסן הוא אלגוריתם במודל החישוב האלגברי.

3.2.1 דיון במודל החישוב

מודל החישוב שהצגנו הוא פשוט ואלגנטי למדי (יש שיאמרו אף פשטני). ניתן להרחיבו במספר דרכים.
הרחבה מתבקשת אחת היא הוספת פעולת חילוק: חילוק (במכנה שונה מאפס) היא חשבון טבעית שלא הרשינו לבצע במודל. מתברר כי לצורך חישוב כפל מטריצות אין צורך בפעולת חילוק. כלומר, בהינתן אלגוריתם שמחשב כפל מטריצות בעזרת s פעולות חשבון, ניתן למצוא אלגוריתם ללא פעולות חילוק שמחשב כפל מטריצות בעזרת $O(s)$ פעולות חשבון. לא נוכיח משפט זה בקורס.
ניתן היה בנוסף להרשות לאלגוריתם לבצע השוואות לאפס או התפצלויות וכו׳ – אלו הם מודלים אחרים ורלוונטיים אך לשם הפשטות לא נדון בהם.
לבסוף, נשים לב כי במודל החישוב שלנו אנחנו סופרים אך ורק פעולות חשבון ולא מתייחסים למספר הביטים הדרוש לשם אחסון תוצאות הביניים ולשיקולים אחרים כמו סיבוכיות זכרון, עמידות לרעש וכן הלאה. כל אלו הם שיקולים חשובים וכמובן שבמימוש האלגוריתמים במחשב אמיתי יש להתייחס אליהם.

3.2.2 האקספוננט של כפל מטריצות

נגדיר:

$$\omega = \inf\{\beta : \text{קיים אלגוריתם אלגברי לכפל מטריצות המבצע } O(n^\beta) \text{ פעולות חשבון}\}.$$

נשים לב שהגדרנו את ω בתור אינפימום ולא בתור מינימום. כלומר, ω הוא מספר כך שלכל $\varepsilon > 0$ קיים אלגוריתם שמבצע $O(n^{\omega+\varepsilon})$ פעולות חשבון. הגדרה זו עשויה להיראות לא טבעית, אך בהמשך הקורס נראה מדוע היא נוחה.

עובדה 3.5. $2 \le \omega \le 3$.

משפט 3.6. (נובע ישירות מהאלגוריתם של שטראסן) $\omega \le \log_2 7 \le 2.81$.

השערה 3.7. $\omega = 2$.

השערה 3.7 היא ההשערה המרכזית בתחום של כפל מטריצות ואחת ההשערות החשובות ביותר בתיאוריה של מדעי המחשב.
בפרק הבא בקורס נראה הגדרה מתמטית אלגנטית מאוד ל-ω.

4 פונקציות בילינאריות, טנזורים ודרגה

4.1 פונקציות בילינאריות

פונקציות לינאריות הן האובייקט המרכזי בקורס באלגברה לינארית. בפרק זה נגדיר מהן פונקציות בילינאריות ונתרגם חלק מהמושגים שנלמדו בקורס באלגברה לינארית לתחום הזה.

הגדרה 4.1. יהי $\mathbb{F}$ שדה. פונקציה $B : \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}^t$ תקרא **בילינארית** אם:

1. לכל $y_0 \in \mathbb{F}^m$, $B(\cdot, y_0) : \mathbb{F}^n \to \mathbb{F}^t$ היא פונקציה לינארית.
2. לכל $x_0 \in \mathbb{F}^n$, $B(x_0, \cdot) : \mathbb{F}^m \to \mathbb{F}^t$ היא פונקציה לינארית.

עובדה 4.2. לכל m, n, t פונקציית כפל המטריצות $B : \mathbb{F}^{m\times n} \times \mathbb{F}^{n\times t} \to \mathbb{F}^{m\times t}$ המוגדרת ע״י $B(X, Y) = X \cdot Y$ היא פונקציה בילינארית.

הוכחה: נובע ישירות מהתכונות של כפל מטריצות. □

כזכור, פונקציה $A : \mathbb{F}^n \to \mathbb{F}^m$ היא לינארית אם ורק אם לכל i ניתן לכתוב את הקואורדינטה ה־i שלה כפולינום ממעלה 1:

$$.A_i(x_1,\dots,x_n) = \sum_{j=1}^{n} a_{i,j} x_j$$

את האיברים $a_{i,j}$ אנו מסדרים בטבלה דו־מימדית בגודל $m \times n$ שנקראת **המטריצה המייצגת** של A (בבסיס הסטנדרטי). התכונות של A (למשל, הדרגה שלה) מלמדות אותנו דברים רבים על הטרנספורמציה הלינארית A.
נרצה למצוא את האובייקט המקביל לפונקציה בילינארית. תחילה ניעזר בעובדה הבאה, שתוכח בתרגילי הבית:

עובדה 4.3. תהי $B : \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}^t$ פונקציה בילינארית. אז לכל $1 \le k \le t$ ניתן לכתוב את הקואורדינטה ה־k של B באופן הבא:

$$B_k(x_1,\dots,x_n,y_1,\dots,y_m) = \sum_{1\le i\le n, 1\le j\le m} c_{i,j,k} x_i y_j$$

כאשר $c_{i,j,k} \in \mathbb{F}$.

ה"טבלה" התלת־מימדית $t = (c_{i,j,k}) \in \mathbb{F}^{n\times m\times t}$ נקראת ה**טנזור** שמתאים לבעייה הבילינארית B.
כאמור, ניתן לחשוב על טנזור כעל קובייה תלת מימדית או לחלופין כעל k מטריצות בגודל $n\times m$ המונחות "אחת על השנייה". כל מטריצה כזו נקראת **פרוסה** של הטנזור (ניתן כמובן להסתכל על פרוסות בכל אחד משלושת המימדים).
כמו במקרה של פונקציות לינאריות, נחקור תכונות בסיסיות של הטנזורים שמתאימים לבעיות בילינאריות (ובפרט לכפל מטריצות) ונראה כיצד הן משפיעות על הסיבוכיות של חישוב כפל מטריצות.
כיוון שכפל מטריצות היא פעולה בילינארית, הרי שקיים לה טנזור מתאים. בתור דוגמה נכתוב תחילה את הטנזור שמתאים לכפל מטריצות 2×2. זוהי פעולה בילינארית $B : \mathbb{F}^{2\times2}\times\mathbb{F}^{2\times2} \to \mathbb{F}^{2\times2}$.
ניתן לחשוב על הטנזור כארבע פרוסות המתאימות לארבע התאים במטריצת המכפלה: $(1,1),(1,2),(2,1),(2,2)$.
כל פרוסה היא מטריצה 4×4 שבה השורות מתאימות למשתני התאים של המטריצה X, דהיינו $x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}$ והעמודות מתאימות למשתני התאים של המטריצה Y באופן אנלוגי.
בקואורדינטה $(1,1)$ נמצאת הפונקציה הבילינארית $\sum_{k=1}^{2} x_{1,k} y_{k,1}$. כלומר, הפרוסה המתאימה היא המטריצה הבאה:

$$\begin{array}{cccc} x_{1,1} & x_{1,2} & x_{2,1} & x_{2,2} \end{array}$$
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{c} y_{1,1} \\ y_{1,2} \\ y_{2,1} \\ y_{2,2} \end{array}$$

באופן דומה ניתן לכתוב את שאר פרוסות הטנזור.
כעת נכתוב הגדרה כללית לטנזור של כפל מטריצות. הדבר המרכזי שמבלבל בהגדרה הוא השימוש המרובה באינדקסים, אולם פרט לכך כל מה שאנו עושים הוא "לארוז" את ההגדרה שאנחנו מכירים ורגילים אליה בטבלה תלת מימדית.

הגדרה 4.4. יהיו m,n,t מספרים טבעיים. נסמן ב־ $\langle m,n,t\rangle$ את הפונקציה הבילינארית $M : \mathbb{F}^{m\times n}\times\mathbb{F}^{n\times t} \to \mathbb{F}^{m\times t}$ המתאימה לכפל של מטריצה $m\times n$ במטריצה $n\times t$. נשתמש באותו סימון לטנזור המתאים,

$$.\langle m,n,t\rangle \in \mathbb{F}^{mn\times nt\times mt}$$

כיוון ש־

$$,M_{i,j}=\sum_{k=1}^{m} x_{i,k}y_{k,j}$$

הטנזור מוגדר באופן הבא:

$$.\langle m,n,t\rangle_{(i,k),(k',j),(i',j')}=\begin{cases}1 & i=i',j=j',k=k' \text{ אם}\\ 0 & \text{אחרת}\end{cases}$$

(כאשר $1\le i,i'\le m$, $1\le k,k'\le n$, $1\le j,j'\le t$).

4.2 אלגוריתמים בילינאריים

כזכור, אלגוריתם אלגברי רשאי לכפול או לחבר כל שתי תוצאות ביניים שכבר חישב. עם זאת, לעיתים נוח יותר לחקור אלגוריתמים אלגבריים מוגבלים בכוחם או בעלי מבנה מיוחד. לדוגמה, בחישוב פונקציה בילינארית

$$B(x_1,\dots,x_n,y_1,\dots,y_n)$$

האלגוריתם האלגברי רשאי לחשב פונקציות מהסגנון x_1x_2 או x_1^2 או אפילו $x_1^{17}\cdot x_2^5$. אינטואיטיבית, זאת נראית כמו עבודה מיותרת כיוון שהפלט צריך להיות פונקציה בילינארית ב־x,y ולכן ניתן לצפות שחישובים כאלה לא יועילו לצורך חישוב B.
במקרה זה מתברר כי האינטואיציה נכונה. נסביר למה הכוונה.

הגדרה 4.5. נאמר שאלגוריתם אלגברי $P_1,\dots,P_s$ הוא **בילינארי** (שימו לב כי כעת זו תכונה של האלגוריתם, ולא של הבעייה החישובית) אם הוא "מכבד" את המבנה הבילינארי של הבעייה. כלומר, אלגוריתם כזה מורכב מארבעה שלבים:
שלב 1: השורות $P_1,\dots,P_{s_1}$ מחשבות פונקציות לינאריות במשתנים $x_1,\dots,x_n$.
שלב 2: השורות $P_{s_1+1},\dots,P_{s_2}$ מחשבות פונקציות לינאריות במשתנים $y_1,\dots,y_m$.
שלב 3: חישוב מכפלות בילינאריות. כל אחת מהשורות $P_{s_2+1},\dots,P_{s_3}$ היא כפל

$$P_\mu\cdot P_\sigma$$

כאשר P_μ חושבה בשלב 1 ו־P_σ בשלב 2.
שלב 4: סכומים פונקציות בילינאריות. כל אחת מהשורות $P_{s_3+1},\dots,P_s$ היא צירוף לינארי

$$\alpha P_\mu+\beta P_\sigma$$

כאשר $\alpha,\beta\in\mathbb{F}$ ו־P_μ,P_σ חושבו בשלב 3 או קודם לכן בשלב 4.

כלומר, אלגוריתם בילינארי מחשב בעייה בילינארית בצורה טבעית וללא "התחכמויות", כלומר חישובי ביניים שאינם בילינאריים. בתרגילי הבית נוכיח את העובדה הבאה:

עובדה 4.6. תהיינה $B_1,\dots,B_k$ קבוצת פונקציות בילינאריות במשתנים $x_1,\dots,x_n,y_1,\dots,y_m$.
נניח כי קיים אלגוריתם אלגברי באורך s לחישוב $B_1,\dots,B_k$. אז קיים אלגוריתם בילינארי באורך $O(s)$ לחישוב $B_1,\dots,B_k$.

נשים לב כי אלגוריתם בילינארי מבצע פעולות כפל רק בשלב 3.

4.3 דרגה של פונקציות בילינאריות ושל טנזור

העובדה שבחישוב פונקציות בילינאריות ניתן להתחשב רק בבעיות בילינאריות נותנת מוטיבציה להגדרה הבאה.

הגדרה 4.7. תהי $B:\mathbb{F}^n\times\mathbb{F}^m\to\mathbb{F}^t$ פונקציה בילינארית. נסמן $B=(B_1,\ldots,B_t)$ כאשר $B_i:\mathbb{F}^n\times\mathbb{F}^m\to\mathbb{F}$ היא פונקציה בילינארית.
ה**דרגה** של B היא המספר המינימלי של מכפלות בילינאריות

$$P_i=\ell_i(x_1,\ldots,x_n)\cdot\ell'_i(y_1,\ldots,y_m),\quad 1\le i\le r$$

כך ש־

$$\{B_1,\ldots,B_t\}\subseteq\operatorname{span}\{P_1,\ldots,P_r\}$$

נסמן ערך זה ב־$\operatorname{rank}(B)$.

אבחנה 4.8. $\operatorname{rank}(\langle 2,2,2\rangle)\le 7$.

הוכחה: כזכור, האלגוריתם של שטראסן מחשב 7 מכפלות בילינאריות. □

למה 4.9. תהי B פונקציה בילינארית. אזי $\operatorname{rank}(B)$ הוא המספר ההכרחי והמספיק של פעולות כפל לחישוב B בעזרת אלגוריתם בילינארי. כלומר, קיים אלגוריתם בילינארי המחשב את B בעזרת $\operatorname{rank}(B)$ פעולות, וכל אלגוריתם בילינארי דורש $\operatorname{rank}(B)$ פעולות.

הוכחה: מהגדרת $\operatorname{rank}(B)$, נובע באופן טבעי אלגוריתם בילינארי לחישוב B המשתמש ב־$\operatorname{rank}(B)$ מכפלות.
נסמן $r=\operatorname{rank}(B)$. אלגוריתם זה יחשב תחילה את $\ell_1(x),\ldots,\ell_r(x)$ ("שלב 1") ולאחר מכן את $\ell'_1(y),\ldots,\ell'_r(y)$ ("שלב 2").
לאחר מכן, האלגוריתם יחשב בעזרת r מכפלות את $P_1,\ldots,P_r$ ("שלב 3").
לבסוף, כיוון שמההגדרה $B_1,\ldots,B_t$ נפרשים ע"י $P_1,\ldots,P_r$, האלגוריתם יכול להשתמש בפעולות לינאריות על מנת לחשב את $B_1,\ldots,B_t$ ("שלב 4").
בכיוון השני, נביט באלגוריתם בילינארי שמחשב את B. כפי שהערנו בעבר אלגוריתם בילינארי מחשב מכפלות בילינאריות בשלב 3, ולכן כל הפונקציות שהוא מחשב בשלב 4 נפרשות ע"י הפונקציות שחושבו בשלב 3. מכאן, שאם נסמן $Q_1,\ldots,Q_\rho$ את הפונקציות שהאלגוריתם מחשב בשלב 3, נקבל כי $B_1,\ldots,B_t\in\operatorname{span}\left\{Q_1,\ldots,Q_\rho\right\}$ ומהגדרת הדרגה נובע ש־$r\le\rho$. □

ניתן גם להגדיר דרגה לטנזור המתאים ל־B. ניזכר כי אם

$$B_k=\sum_{i,j}c_{i,j,k}x_iy_j$$

אז $T=\left\{c_{i,j,k}\right\}$ הוא הטנזור המתאים.
אם $\operatorname{rank}(B)=r$ אז יש r פונקציות לינאריות $\ell_1(x),\ldots,\ell_r(x)$, ו־$r$ פונקציות לינאריות $\ell'_1(y),\ldots,\ell'_r(y)$ כך שלכל $1\le k\le t$ יש r מקדמים $w_{k,1},\ldots,w_{k,r}\in\mathbb{F}$ כך ש־

$$B_k=\sum_{\sigma=1}^{r}w_{k,\sigma}\ell_\sigma(x)\ell'_\sigma(y).\qquad(1)$$

לכל $1 \le \sigma \le r$ נסמן

$$\ell_\sigma = \sum_{i=1}^{n} u_{i,\sigma} x_i, \quad \ell'_\sigma = \sum_{j=1}^{m} v_{j,\sigma} y_j. \tag{2}$$

נציב ב־(1) את (2) ונראה שהמקדם של $x_i y_j$ ב־B_k הוא

$$\sum_{\sigma=1}^{r} u_{\sigma,i} v_{\sigma,j} w_{\sigma,k}.$$

הגדרה 4.10. יהיו $u \in \mathbb{F}^n, v \in \mathbb{F}^m, w \in \mathbb{F}^t$ וקטורים. הטנזור

$$u \otimes v \otimes w \in \mathbb{F}^{n \times m \times t}$$

המוגדר ע״י

$$(u \otimes v \otimes w)_{i,j,k} = u_i v_j w_k$$

נקרא **טנזור מדרגה 1**.
ה**דרגה** של טנזור T היא ה־r המינימלי כך שקיימים r טנזורים מדרגה 1

$$S_1 = (u_1 \otimes v_1 \otimes w_1), \dots, S_r = (u_r \otimes v_r \otimes w_r)$$

כך ש־$T = \sum_{\sigma=1}^{r} S_\sigma$.

מהדיון לעיל נובע שההגדרות הנ״ל שקולות. כלומר, הדרגה של B (כפונקציה בילינארית) היא r אם ורק אם דרגת הטנזור המתאים היא r.

תרגיל 4.11. ודאו זאת.

נעיר כי מבלי לשים לב, הכללנו (לשלושה מימדים, וניתן באופן זהה להכליל לכל d מימדים) את המושג של דרגה של מטריצה. אחת ההגדרות השקולות לדרגה של מטריצה M היא ה־r המינימלי כך שקיימות r מטריצות מדרגה 1, $M_1, \dots, M_r$ כך ש־$M = M_1 + \cdots + M_r$.
מטריצה מדרגה 1 היא מטריצה מהצורה uv^T כאשר u, v וקטורים, כלומר התא ה־(i, j) הוא $u_i v_j$. ניתן (ונהוג לעיתים) במקום uv^T לכתוב $u \otimes v$.
בהמשך הקורס נראה כמה היבטים חשובים בהם דרגה של טנזור שונה מדרגה של מטריצה.
נסיים פרק זה באפיון אלטרנטיבי של ω.

משפט 4.12.

$$\omega = \inf\left\{\beta : \mathrm{rank}(\langle n, n, n\rangle) = O(n^\beta)\right\}.$$

כזכור, $\mathrm{rank}(\langle n, n, n\rangle)$ הוא בדיוק מספר הכפלים הבילינאריים ההכרחי והמספיק על מנת לחשב את $\langle n, n, n\rangle$.
ברור שאלגוריתם שמבצע $O(n^\beta)$ פעולות מבצע לכל היותר מספר זה של פעולות כפל. משפט 4.12 אומר שכפי שרמזנו בתחילת הקורס, על מנת לחסום את הסיבוכיות של כפל מטריצות מספיק לחסום את מספר הכפלים הבילינאריים.
רעיון ההוכחה הוא, כמו באלגוריתם של שטראסן, להשתמש באינדוקציה על מנת להראות שמבחינה אסימפטוטית הגורם המשמעותי בחישוב מספר הפעולות הכולל הוא מספר הכפלים, כיוון שניתן לחבר שתי מטריצות $m \times m$ בעזרת $O(m^2)$ פעולות חיבור.

הוכחה: נסמן

$$\tilde{\omega} = \inf\left\{\beta : \operatorname{rank}(\langle n,n,n\rangle) = O(n^\beta)\right\}$$

ונרצה להראות $\omega = \tilde{\omega}$.

מהגדרת ω ומהדיון לעיל, ברור ש־$\omega \le \tilde{\omega}$. נראה אי שיוויון בכיוון השני.

מהגדרת $\tilde{\omega}$, לכל $\varepsilon > 0$ קיים $m_0 \in \mathbb{N}$ כך שלכל $m \ge m_0$,

$$\operatorname{rank}(\langle m,m,m\rangle) \le K m^{\tilde{\omega}+\varepsilon}. \tag{3}$$

יהי $\varepsilon > 0$ ו־m גדול מספיק כך ש־(3) מתקיים. כלומר, קיים אלגוריתם שמכפיל מטריצות $m \times m$ בעזרת $r := \operatorname{rank}(\langle m,m,m\rangle)$ כפלים בילינאריים. נסמן ב־C את מספר פעולות החיבור שהאלגוריתם זה מבצע.

כעת נכפיל מטריצות $m^i \times m^i$ ע״י חלוקתן לבלוקים בגודל $m^{i-1} \times m^{i-1}$. את הבלוקים נכפיל בעזרת האלגוריתם שמבצע r פעולות כפל ו־C חיבורים.

נסמן ב־$A(i)$ את מספר הפעולות הכולל שהאלגוריתם מבצע. כיוון שיש צורך ב־r פעולות כפל של מטריצות $m^{i-1} \times m^{i-1}$ ו־C פעולות חיבור, נקבל ש־

$$A(i) \le r A(i-1) + C m^{2(i-1)}. \tag{4}$$

אם נפתח את נוסחת הרקורסיה (4), נקבל

$$\begin{aligned}
A(i) &\le r A(i-1) + C m^{2(i-1)} \\
&\le r^2 A(i-2) + r C m^{2(i-2)} + C m^{2(i-1)} \\
&\le r^3 A(i-3) + r^2 C m^{2(i-3)} + r C m^{2(i-2)} + C m^{2(i-1)} \\
&\le \dots \\
&\le r^i A(0) + C m^{2(i-1)} \left(\sum_{j=0}^{i-2} \left(\frac{r}{m^2}\right)^j \right)
\end{aligned}$$

על מנת לחסום את הטור ההנדסי $\sum_{j=0}^{i-2}\left(\frac{r}{m^2}\right)^j$ ניתן להניח תחילה $r > m^2$ (למעשה, ניתן להוכיח כי זה המצב, אך לא נוכיח זאת ונעיר בסוף כיצד לטפל במקרה זה).

$$\begin{aligned}
A(i) &\le r^i A(0) + C m^{2(i-1)} \left(\sum_{j=0}^{i-2} \left(\frac{r}{m^2}\right)^j \right) \\
&= r^i A(0) + C m^{2(i-1)} \cdot \frac{(r/m^2)^{i-1} - 1}{(r/m^2) - 1} \\
&= r^i A(0) + C m^2 \cdot \frac{r^{i-1} - m^{2(i-1)}}{r - m^2} \\
&\le r^i A(0) + C m^2 \cdot \frac{r^{i-1}}{r - m^2} \\
&\le \left(A(0) + \frac{C m^2}{r(r-m^2)} \right) r^i.
\end{aligned}$$

הנקודה המרכזית היא ש־$A(0)+\frac{Cm^2}{r(r-m^2)}$ הוא קבוע אבסולוטי כלשהו.
כעת, עבור n כללי, יהי $n' = m^{\lceil \log_m n \rceil}$. ($n'$ הוא המספר הכי קטן מהצורה m^i שגדול מ־n).
נסמן ב־$C(\langle n,n,n\rangle)$ את מספר הפעולות הכולל הדרוש לחישוב $\langle n,n,n\rangle$. נקבל:

$$\begin{aligned} C(\langle n,n,n\rangle) &\le C(\langle n',n',n'\rangle) \\ &\le A(\lceil \log_m n \rceil) \\ &= O(r^{\lceil \log_m n \rceil}) \\ &= O(r^{\log_m n}) = O(n^{\log_m r}). \end{aligned}$$

כזכור, $r \le K m^{\tilde{\omega}+\varepsilon}$ ולכן $\log_m r \le \tilde{\omega}+\varepsilon+\log_m K$.
נבחר את m גדול דיו כך ש־$\log_m K \le \varepsilon$, ונקבל

$$C(\langle n,n,n\rangle) \le O(n^{\tilde{\omega}+2\varepsilon}).$$

כיוון ש־ε קטן כרצוננו ו־ω היא אינפימום, נקבל $\omega \le \tilde{\omega}$.
נעיר לבסוף שאם $r = m^2$, פתרון נוסחת הרקורסיה נותן

$$A(i) \le r^i A(0) + (i-1) r^{i-1}$$

וע״י חישוב זהה נקבל

$$C(\langle n,n,n\rangle) \le O(r^{\log_m n} + \log_m n\, r^{\log_m n - 1}) = O\left(\frac{\log_m n}{r} n^{\log_m r}\right) = O(n^{\tilde{\omega}+3\varepsilon}).$$

(תרגיל: השלימו את פרטי החישוב) □

5 פעולות על טנזורים

נגדיר מספר פעולות טבעיות על טנזורים ונבחן את השפעתן על הדרגה.

5.1 פרמוטציות

עבור מטריצה A, $\operatorname{rank}(A) = \operatorname{rank}(A^T)$. ניתן לחשוב על A^T כעל הפעלת פרמוטציה שמחליפה בין שורות ועמודות A. באותו אופן ניתן לבצע להפעיל פרמוטציות על טנזורים, רק שכעת יש שלושה מימדים. לדוגמה, בהינתן טנזור $A \in \mathbb{F}^{n\times m\times t}$ ניתן להגדיר טנזור $A' \in \mathbb{F}^{m\times t\times n}$ ע״י

$$A'_{i,j,k} = A_{k,i,j}.$$

באופן כללי יותר, תהי $\pi : \{1,2,3\} \to \{1,2,3\}$ פרמוטציה. עבור טנזור $A \in \mathbb{F}^{n\times m\times t}$, נגדיר

$$(\pi A)_{i_1,i_2,i_3} = A_{i_{\pi(1)},i_{\pi(2)},i_{\pi(3)}}.$$

תרגיל 5.1. לכל פרמוטציה π וטנזור T, $\operatorname{rank}(T) = \operatorname{rank}(\pi T)$.

במטריצות ניתן לבצע גם פרמוטציה על השורות והעמודות, שאף הן משמרות את הדרגה. גם בטנזורים ניתן לעשות פרמוטציות כאלה בכל אחד משלושת המימדים. יהי $A \in \mathbb{F}^{n\times m\times t}$ טנזור, ו־$\sigma : \{1,\ldots,n\} \to \{1,\ldots,n\}$ פרמוטציה. אזי אם נגדיר $A' \in \mathbb{F}^{n\times m\times t}$ ע״י

$$(A')_{i,j,k} = A_{\sigma(i),j,k}$$

מתקיים $\operatorname{rank}(A') = \operatorname{rank}(A)$.

תרגיל 5.2. הוכיחו זאת. ודאו בנוסף שטענה זהה מתקיימת בשני המימדים האחרים.

5.2 פרמוטציות על הטנזור של כפל מטריצות

כזכור, ראינו כי אם $A = \langle m,n,t\rangle$, אזי $A \in \mathbb{F}^{(m\times n)\times(n\times t)\times(m\times t)}$ והטנזור נתון ע״י

$$A_{(i,k),(k',j),(i',j')} = \delta_{i,i'}\delta_{k,k'}\delta_{j,j'}$$

כאשר $\delta_{p,q} = 1$ אם $p = q$ ו־0 אחרת.
ע״י הפעלת פרמוטציה על הרכיב השלישי של הטנזור נהוג יותר בספרות להגדיר

$$A = \langle m,n,t\rangle \in \mathbb{F}^{(m\times n)\times(n\times t)\times(t\times m)}, \quad A_{(i,k),(k',j),(j',i')} = \delta_{i,i'}\delta_{k,k'}\delta_{j,j'}.$$

תהא π הפרמוטציה המוגדרת ע״י $\pi(1) = 3, \pi(2) = 1, \pi(3) = 2$. אזי

$$\pi A =: A' \in \mathbb{F}^{(t\times m)\times(m\times n)\times(n\times t)}$$

מוגדר ע״י

$$\pi A_{(j',i'),(i,k),(k',j)} = A_{(i,k),(k',j),(j',i')} = \delta_{i,i'}\delta_{k,k'}\delta_{j,j'} = \delta_{j,j'}\delta_{i,i'}\delta_{k,k'}.$$

כלומר, $\pi A = \langle t,m,n\rangle$.
באופן דומה, נקבל

$$\operatorname{rank}(\langle m,n,t\rangle) = \operatorname{rank}(\langle t,m,n\rangle) = \operatorname{rank}(\langle n,t,m\rangle).$$

נמשיך הלאה. ע״י פרמוטציות בכל אחד מהמימדים, נגדיר

$$A''_{(k,i),(j,k'),(i',j')} = A_{(i,k),(k',j),(j',i')}, \quad A'' \in \mathbb{F}^{(n\times m)\times(t\times n)\times(m\times t)}.$$

אזי $\operatorname{rank}(A'') = \operatorname{rank}(A)$.
בנוסף, תהא π' הפרמוטציה המוגדרת ע״י $\pi'(1) = 2, \pi'(2) = 1, \pi'(3) = 3$. נגדיר

$$\pi' A'' =: A''' \in \mathbb{F}^{(t\times n)\times(n\times m)\times(m\times t)}.$$

נחשב ונגלה כי

$$(A''')_{(j,k),(k',i),(i',j')} = (A'')_{(k',i),(j,k),(i',j')} = A_{(i,k'),(k,j),(j',i')} = \delta_{i,i'}\delta_{j,j'}\delta_{k,k'}.$$

כלומר, $A''' = \langle t,n,m\rangle$ ו־ $\operatorname{rank}(\langle t,n,m\rangle) = \operatorname{rank}(\langle m,n,t\rangle)$.

מסקנה 5.3.

$$\operatorname{rank}(\langle m,n,t\rangle) = \operatorname{rank}(\langle m,t,n\rangle) = \operatorname{rank}(\langle n,m,t\rangle) =$$
$$\operatorname{rank}(\langle n,t,m\rangle) = \operatorname{rank}(\langle t,m,n\rangle) = \operatorname{rank}(\langle t,n,m\rangle).$$

5.3 כפל טנזורים

נגדיר כעת פעולת כפל של שני טנזורים. **שימו לב!** פעולה זו **אינה קשורה** ואינה אנלוגית לכפל מטריצות. אכן, השימוש במילה ״כפל״ מעט מטעה. פעולת כפל של טנזורים היא אנלוגית למכפלת קרונקר של מטריצות, הקרויה לעיתים (על מנת להגביר את הבלבול) בשם ״מכפלה טנזורית״.
בהינתן מטריצות $A \in \mathbb{F}^{m\times n}, B \in \mathbb{F}^{p\times q}$, המכפלה הטנזורית שלהן,

$$(A \otimes B) \in \mathbb{F}^{mp\times nq}$$

מוגדרת ע״י

$$(A \otimes B)_{(i,j),(i',j')} = A_{i,i'} B_{j,j'}.$$

עבור מטריצות, קל לתאר את $A \otimes B$ בעזרת מטריצת בלוקים. עד כדי פרמוטציה של השורות,

$$A \otimes B = \begin{bmatrix} A_{1,1}B & A_{1,2}B & \cdots & A_{1,n}B \\ A_{2,1}B & A_{2,2}B & \cdots & A_{2,n}B \\ \vdots & \vdots & \cdots & \vdots \\ A_{m,1}B & A_{m,2}B & \cdots & A_{m,n}B \end{bmatrix}.$$

תרגיל 5.4. תהינה A, B מטריצות כנ״ל. אז

$$\operatorname{rank}(A \otimes B) = \operatorname{rank}(A) \cdot \operatorname{rank}(B).$$

הגדרה 5.5. יהיו $A \in \mathbb{F}^{m\times n\times t}$ ו- $B \in \mathbb{F}^{\tilde{m}\times\tilde{n}\times\tilde{t}}$ שני טנזורים. ה**מכפלה הטנזורית** שלהם, המסומנת $A \otimes B$, היא הטנזור

$$A \otimes B \in \mathbb{F}^{m\tilde{m}\times n\tilde{n}\times t\tilde{t}}$$

המוגדר ע״י

$$(A \otimes B)_{(i,\tilde{i}),(j,\tilde{j}),(k,\tilde{k})} = A_{i,j,k} B_{\tilde{i},\tilde{j},\tilde{k}}.$$

נשים לב כי נוח לאנדקס את טנזור המכפלה ע״י זוגות של אינדקסים של הטנזורים המקוריים. זה מעט מבלבל כיוון שבמקרה של כפל מטריצות גם השתמשנו בזוגות של מספרים טבעיים עבור האינדקסים, אולם שם המשמעות שונה.

למה 5.6. יהיו A, B טנזורים כנ״ל. אזי

$$\operatorname{rank}(A \otimes B) \le \operatorname{rank}(A)\operatorname{rank}(B).$$

שימו לב כי בניגוד לתרגיל 5.4, כאן הטענה היא רק קיום של אי שוויון. אכן, בהמשך הקורס נראה דוגמאות שבהן לא מתקיים שוויון.

הוכחה: נסמן $\operatorname{rank}(A) = r, \operatorname{rank}(B) = \tilde{r}$ ונכתוב

$$A = \sum_{p=1}^{r} u_p \otimes v_p \otimes w_p, \quad B = \sum_{q=1}^{\tilde{r}} \tilde{u}_q \otimes \tilde{v}_q \otimes \tilde{w}_q.$$

נגדיר

$$.U_{p,q}=(u_p\tilde{u}_q)_{i,\tilde{i}}=(u_p)_i(\tilde{u}_q)_{\tilde{i}}\in\mathbb{F}^{m\tilde{m}}$$

ובאופן דומה עבור v, w. כעת,

$$U_{p,q}\otimes V_{p,q}\otimes W_{p,q}\in\mathbb{F}^{m\tilde{m}\times n\tilde{n}\times t\tilde{t}}$$

הוא טנזור מדרגה 1 המוגדר ע״י

$$\left(U_{p,q}\otimes V_{p,q}\otimes W_{p,q}\right)_{(i,\tilde{i}),(j,\tilde{j}),(k,\tilde{k})}=(u_p)_i(\tilde{u}_q)_{\tilde{i}}(v_p)_j(\tilde{v}_q)_{\tilde{j}}(w_p)_k(\tilde{w}_q)_{\tilde{k}}$$

נטען כי

$$.A\otimes B=\sum_{p=1}^{r}\sum_{q=1}^{\tilde{r}}U_{p,q}\otimes V_{p,q}\otimes W_{p,q} \tag{5}$$

ולכן $\mathrm{rank}(A\otimes B)\le r\tilde{r}$.
אכן, אם נסמן ב־C את צד ימין של (5), נקבל

$$\begin{aligned}C_{(i,\tilde{i}),(j,\tilde{j}),(k,\tilde{k})}&=\sum_{p=1}^{r}\sum_{q=1}^{\tilde{r}}(u_p)_i(\tilde{u}_q)_{\tilde{i}}(v_p)_j(\tilde{v}_q)_{\tilde{j}}(w_p)_k(\tilde{w}_q)_{\tilde{k}}\\&=\left(\sum_{p=1}^{r}(u_p)_i(v_p)_j(w_p)_k\right)\left(\sum_{q=1}^{\tilde{r}}(\tilde{u}_q)_{\tilde{i}}(\tilde{v}_q)_{\tilde{j}}(\tilde{w}_q)_{\tilde{k}}\right)\\&=A_{i,j,k}B_{\tilde{i},\tilde{j},\tilde{k}}=(A\otimes B)_{(i,\tilde{i}),(j,\tilde{j}),(k,\tilde{k})}\end{aligned}$$

כפי שרצינו להוכיח. □

מה מתקבל מכפל של טנזור כפל מטריצות?

למה 5.7.

$$.\langle m,n,t\rangle\otimes\left\langle\tilde{m},\tilde{n},\tilde{t}\right\rangle=\left\langle m\tilde{m},n\tilde{n},t\tilde{t}\right\rangle$$

בתרגילי הבית תתבקשו להוכיח את הלמה.
כלומר, כפל של שני טנזורי כפל מטריצות נותן טנזור כפל מטריצות גדול יותר!
ממה שהוכחנו עד כה, נובע שחסמים עליונים על הדרגה של כפל מטריצות במימדים קבועים גוררים חסמים עליונים על ω.

משפט 5.8. אם $\mathrm{rank}(\langle m,n,t\rangle)\le r$ אז

$$.\omega\le 3\log_{mnt} r$$

הוכחה: מההנחה וממסקנה 5.3 נובע כי

$$\mathrm{rank}(\langle m,n,t\rangle)\le r,\mathrm{rank}(\langle n,t,m\rangle)\le r,\mathrm{rank}(\langle t,m,n\rangle)\le r$$

ולכן, מלמה 5.6 נובע כי

$$.\mathrm{rank}(\langle m,n,t\rangle\otimes\langle n,t,m\rangle\otimes\langle t,m,n\rangle)\le r^3$$

מלמה 5.7 נובע כי

$$\mathrm{rank}(\langle mnt, mnt, mnt\rangle) = \mathrm{rank}(\langle m,n,t\rangle \otimes \langle n,t,m\rangle \otimes \langle t,m,n\rangle) \le r^3.$$

אם נסמן $N = mnt$, נקבל (שוב, בעזרת למה 5.6)

$$\mathrm{rank}(\langle N^i, N^i, N^i\rangle) \le r^{3i} = (N^{3\log_N r})^i = (N^i)^{3\log_N r}$$

לכל $i \le 1$. לכן, $\omega \le 3\log_N r$. □

דוגמה 5.9. כזכור הוכחנו כי $\mathrm{rank}(\langle 2,2,2\rangle) \le 7$. זה גורר כי $\omega \le 3\log_{2^3} 7 = \log_2 7$ כפי שראינו באלגוריתם של שטראסן.

דוגמה 5.10. ידוע כי $\mathrm{rank}(\langle 70,70,70\rangle) \le 143,640$ מה שגורר $\omega \le 2.80$ (חסם מעט יותר טוב מהאלגוריתם של שטראסן).

6 דרגת גבול

6.1 קירובים של טנזורים

בפרק זה נתקל בתכונה מרכזית של דרגה של טנזור ששונה מהותית מדרגה של מטריצה. נאמר שסדרה של מטריצות $m \times n$ מעל $\mathbb{R}$, $\{A_k\}$, מתכנסת למטריצה $A \in \mathbb{R}^{n\times m}$ אם היא מתכנסת איבר־איבר. כלומר, לכל i, j,

$$\lim_{k\to\infty} (A_k)_{i,j} = A_{i,j}.$$

משפט 6.1. תהי $\{A_k\}$ סדרת מטריצות שמתכנסת למטריצה A.
אם לכל k מתקיים $\mathrm{rank}(A_k) \le r$ אז $\mathrm{rank}(A) \le r$.

הוכחה: כזכור מאלגברה לינארית, $\mathrm{rank}(B) \le r$ אם ורק אם כל המינורים מסדר $(r+1)\times(r+1)$ של B מתאפסים.
מינורים הם דטרמיננטות של תתי מטריצות, ובפרט הם פונקציות רציפות. לכן, כיוון שלכל k מתקיים שכל המינורים מסדר $(r+1)\times(r+1)$ של A_k מתאפסים, נובע שהם מתאפסים גם בגבול, A. □

מסתבר שהתכונה המתוארת במשפט 6.1 ייחודית למטריצות (שאת דרגתן ניתן לאפיין בדרכים רבות, לרבות בעזרת מינורים) ואינה מתקיימת עבור טנזורים!
נראה דוגמה נגדית. לצורך הוכחה שהיא אכן דוגמה נגדית, נצטרך לפתח שיטה להוכיח חסמים תחתונים על דרגות של טנזורים.
נביט בטנזור שמוגדר ע״י הפונקציה הבילינארית:

$$B: \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^2, \quad B(x_1,x_2,y_1,y_2) = (x_1y_1, x_2y_1 + x_1y_2).$$

פרוסות הטנזור המתאים T הן

$$\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}, \quad \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$$

נטען שדרגת הטנזור היא 3. נזכור שהדרגה היא ה־r המינימלי כך שיש r מכפלות בילינאריות

$$\left\{\ell_i(x_1, x_2)\cdot \ell_i'(y_1, y_2)\right\}_{1\le i\le r} \tag{6}$$

שפורשות את $\left\{x_1y_1, x_2y_1+x_1y_2\right\}$. מכאן ברור שהדרגה היא לכל היותר 3: הקבוצה $\left\{x_1y_1, x_2y_1, x_1y_2\right\}$ פורשת. כעת נראה את החסם התחתון. נשים לב שבכל קבוצה פורשת כמו ב־(6) חייבת להיות פונקציה לינארית ℓ_i (במשתנים x_1, x_2) שתלויה ב־x_2. נניח בלי הגבלת הכלליות שזו ℓ_1 כלומר $\ell_1 = \alpha x_1 + \beta x_2$ ו־$\beta \neq 0$. נציב אם כן $x_2 = -\frac{\alpha}{\beta}x_1$. החלפה זו מאפסת את ℓ_1, וכעת עלינו לחשב את

$$.x_1y_1, -\frac{\alpha}{\beta}x_1y_1 + x_1y_2$$

כעת חייבת להיות פונקציה שתלויה ב־y_2. נניח בה"כ שזו ℓ_2'. נאפס את ℓ_2' ע"י הצבה ב־y_2 כמו קודם. כעת עדיין עלינו לחשב את x_1y_1 וחישוב זה דורש לפחות מכפלה בילינארית אחת. מכאן שדרושות לפחות 3 מכפלות מלכתחילה.

עם זאת, נראה כי ניתן לקרב את הטנזור הזה כרצוננו ע"י סדרת טנזורים מדרגה 2. לכל $\varepsilon > 0$, נגדיר את הפונקציות הבילינארית

$$.B'(x_1, y_1, x_2, y_2) = (x_1y_1, x_1y_2 + x_2y_1 + \varepsilon x_2y_2)$$

נטען כי הדרגה של B' היא 2. אכן, אם נגדיר

$$p_1 = x_1y_1, p_2 = (x_1 + \varepsilon x_2)\cdot(y_1 + \varepsilon y_2)$$

אזי $B' = (p_1, \frac{1}{\varepsilon}p_2 - \frac{1}{\varepsilon}p_1)$.

בשפת הטנזורים, הגדרנו לכל $\varepsilon > 0$ את הטנזור

$$.T_\varepsilon = (1,0)\otimes(1,0)\otimes(1,-\frac{1}{\varepsilon}) + (1,\varepsilon)\otimes(1,\varepsilon)\otimes(0,\frac{1}{\varepsilon})$$

שהוא, מההגדרה, טנזור מדרגה 2, והפרוסות שלו הן

$$.\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}, \quad \begin{bmatrix}0 & 1\\ 1 & \varepsilon\end{bmatrix}$$

לכן, $T_\varepsilon \to T$ כאשר $\varepsilon \to 0$.

6.2 קירובים של טנזור כפל המטריצות

מסתבר כי ניתן להשתמש בתופעה הזאת על מנת לקבל אלגוריתמים משופרים לכפל מטריצות. נניח כי אנחנו רוצים לחשב את הכפל ה"חלקי"

$$\begin{pmatrix}x_{1,1} & x_{1,2}\\ x_{2,1} & x_{2,2}\end{pmatrix}\cdot\begin{pmatrix}y_{1,1} & y_{1,2}\\ y_{2,1} & y_{2,2}\end{pmatrix} = \begin{pmatrix}z_{1,1} & z_{1,2}\\ z_{2,1} & ***\end{pmatrix}$$

(כלומר, אנחנו רוצים לחשב רק את שלושת התאים הראשונים במכפלה)

מסתבר כי $\text{rank}(z_{1,1}, z_{1,2}, z_{2,1}) = 6$ אולם ניתן לקרב אותם במובן שראינו לעיל בעזרת 5 מכפלות בילינאריות בלבד.
את החסם התחתון של 6 על הדרגה ניתן להוכיח בעזרת שיטה דומה לחסם התחתון על הדרגה שראינו בדרגה הקודמת. כעת, נגדיר

$$\begin{aligned}
p_1 &= (x_{1,2} + \varepsilon x_{2,2}) y_{2,1} \\
p_2 &= x_{1,1}(y_{1,1} + \varepsilon y_{1,2}) \\
p_3 &= x_{1,2}(y_{1,1} + y_{2,1} + \varepsilon y_{2,2}) \\
p_4 &= (x_{1,1} + x_{1,2} + \varepsilon x_{2,1}) y_{1,1} \\
p_5 &= (x_{1,2} + \varepsilon x_{2,1})(y_{1,1} + \varepsilon y_{2,2})
\end{aligned}$$

חישוב פשוט (אך מייגע) יראה כי

$$\begin{aligned}
\varepsilon z_{1,1} &= \varepsilon p_1 + \varepsilon p_2 + O(\varepsilon^2) \\
\varepsilon z_{1,2} &= p_2 - p_4 + p_5 + O(\varepsilon^2) \\
\varepsilon z_{2,1} &= p_1 - p_3 + p_5 + O(\varepsilon^2)
\end{aligned}$$

כאשר $O(\varepsilon^2)$ פירושו סכום של איברים שכולם מתחלקים ב־ε^2.
כעת, נשים לב שהקירוב הנ״ל של שלושה תאים מתוך כפל המטריצות מאפשר לנו לקרב כפל של מטריצה 2×2 במטריצה 2×3,

$$\begin{pmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \end{pmatrix} \cdot \begin{pmatrix} y_{1,1} & y_{1,2} & y_{1,3} \\ y_{2,1} & y_{2,2} & y_{2,3} \end{pmatrix} = \begin{pmatrix} z_{1,1} & z_{1,2} & z_{1,3} \\ z_{2,1} & z_{2,2} & z_{2,3} \end{pmatrix}$$

בעזרת שני ״עותקים״ של החישוב הנ״ל, ע״י 10 פעולות כפל בסך הכל. (את $z_{1,1}, z_{1,2}, z_{2,1}$ נחשב בדיוק כמו קודם. את $z_{2,2}, z_{1,3} z_{2,3}$ נחשב בעזרת אלגוריתם זהה עד כדי שינוי שמות המשתנים שיכפיל את המטריצה $\begin{pmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \end{pmatrix}$ במטריצה $\begin{pmatrix} y_{1,2} & y_{1,3} \\ y_{2,2} & y_{2,3} \end{pmatrix}$).
במילים אחרות, ניתן לקרב את $\langle 2,2,3 \rangle$ בעזרת 10 פעולות כפל. אילו הקירוב היה חישוב מדוייק, בעזרת משפט 5.8 היינו מקבלים

$$\omega \le 3 \log_{12} 10 \le 2.78 ,$$

מה שיהווה שיפור לאלגוריתם של שטראסן.
נפנה כעת להוכחת משפט שיגיד שזה אכן המצב: מסתבר שבבעיית כפל המטריצות, קירובים הם אכן ״טובים כמו״ חישוב מדוייק במובן מסויים שנגדיר במדוייק. תחילה, נפתח הגדרות פורמליות שיאפשרו לנו לנתח קירובים בצורה יותר שיטתית.

הגדרה 6.2. יהי $\mathbb{F}$ שדה. נסמן ב־$\mathbb{F}[\varepsilon]$ את קבוצת הפולינומים מעל $\mathbb{F}$ במשתנה ε. כלומר,

$$\mathbb{F}[\varepsilon] = \bigcup_{d \in \mathbb{N}} \left\{ \sum_{i=0}^{d} a_i \varepsilon^i : a_i \in \mathbb{F}, 0 \le i \le d \right\} .$$

על $F[\varepsilon]$ מוגדר כפל וחיבור באופן טבעי.

בניגוד לתחילת פרק זה שבו חשבנו על ε כעל קבוע חיובי קטן, כעת אנו חושבים על ε בתור משתנה פורמלי. הגדרה זו מאפשרת לנו בין היתר לעבוד מעל כל שדה ולא בהכרח מעל שדה הממשיים.

הגדרה 6.3. יהי $h \in \mathbb{N}$ ו־$T \in \mathbb{F}^{m\times n\times t}$ טנזור. נגדיר את $\operatorname{rank}_h(T)$ להיות ה־r המינימלי כך שקיימים וקטורים

$$u_\sigma \in \mathbb{F}[\varepsilon]^m, v_\sigma \in \mathbb{F}[\varepsilon]^n, w_\sigma \in \mathbb{F}[\varepsilon]^t, \quad 1 \le \sigma \le r$$

כך ש־

$$\sum_{\sigma=1}^{r} u_\sigma \otimes v_\sigma \otimes w_\sigma = \varepsilon^h T + O(\varepsilon^{h+1}).$$

בנוסף, נגדיר

$$\overline{\operatorname{rank}}(T) = \min_h \operatorname{rank}_h(T).$$

הערך $\overline{\operatorname{rank}}(T)$ נקרא **דרגת הגבול** (border rank) של הטנזור T.

אבחנה 6.4. להלן מספר עובדות בסיסיות:

1. $\operatorname{rank}_0(T) = \operatorname{rank}(T)$.
2. לכל h, $\operatorname{rank}_h(T) \ge \operatorname{rank}_{h+1}(T)$. כלומר, $\operatorname{rank}_0(T) \ge \operatorname{rank}_1(T) \ge \cdots \ge \overline{\operatorname{rank}}(T)$.
3. עבור $\operatorname{rank}_h(T)$, מספיק לבחור $u_\sigma, v_\sigma, w_\sigma$ שבהם ε מופיע בחזקה לכל היותר h (כיוון שחזקות גבוהות יותר בכל מקרה ייבלעו באיבר השגיאה).
4. תהי $\pi : \{1,2,3\} \to \{1,2,3\}$ פרמוטציה. אזי לכל h, $\operatorname{rank}_h(\pi T) = \operatorname{rank}_h(T)$.

עוד נוכיח עובדה בסיסית על קירובי מכפלות של טנזורים בעזרת קירובים לטנזורים עצמם.

למה 6.5. יהיו $T \in \mathbb{F}^{m\times n\times t}$ ו־$T' \in \mathbb{F}^{m'\times n'\times t'}$ טנזורים. אז

$$\operatorname{rank}_{h+h'}(T\otimes T') \le \operatorname{rank}_h(T)\cdot \operatorname{rank}_{h'} T'.$$

הוכחה: נסמן $r = \operatorname{rank}_h(T)$ ו־$r' = \operatorname{rank}_{h'}(T')$.
מההנחה, קיימים טנזורים

$$S \in \mathbb{F}[\varepsilon]^{m\times n\times t}, \quad S' \in \mathbb{F}[\varepsilon]^{m'\times n'\times t'}$$

כך ש־

$$S = \varepsilon^h T + \varepsilon^{h+1} U$$
$$S' = \varepsilon^{h'} T' + \varepsilon^{h'+1} U'$$

כאשר

$$U \in \mathbb{F}[\varepsilon]^{m\times n\times t}, \quad U' \in \mathbb{F}[\varepsilon]^{m'\times n'\times t'}$$

ובנוסף $\operatorname{rank}(S) = r$ ו־$\operatorname{rank}(S') = r'$. נחשב את $S\otimes S'$. מהגדרת כפל טנזורים, מתקיים ש־

$$\begin{aligned}(S\otimes S')_{(i,i'),(j,j'),(k,k')} &= (\varepsilon^h T_{i,j,k} + \varepsilon^{h+1} U_{i,j,k})\cdot(\varepsilon^{h'} T'_{i',j',k'} + \varepsilon^{h'+1} U'_{i',j',k'}) \\ &= \varepsilon^{h+h'} T_{i,j,k}\cdot T'_{i',j',k'} + O(\varepsilon^{h+h'+1})\end{aligned}$$

כלומר,

$$S\otimes S' = \varepsilon^{h+h'} T\otimes T' + O(\varepsilon^{h+h'+1}),$$

ואילו מלמה 5.6 נובע ש־$\operatorname{rank}(S\otimes S') \le r\cdot r'$ ולכן $\operatorname{rank}_{h+h'}(T\otimes T') \le r\cdot r'$. □

6.3 מחישוב מקורב לחישוב מדוייק

האם ניתן לעבור מקירוב לחישוב מדוייק? מסתבר שכן, ב״תשלום״ שתלוי בפרמטר הקירוב. למרבה המזל התלות היא פולינומית בלבד.

למה 6.6. לכל טנזור $T \in \mathbb{F}^{m\times n\times t}$ ולכל $h \in \mathbb{N}$ מתקיים

$$\operatorname{rank}(T) \le c_h \operatorname{rank}_h(T)$$

כאשר c_h פולינום ב־h (בפרט, נוכיח זאת עבור $c_h = \binom{h+2}{2}$).

הוכחה: נסמן $r = \operatorname{rank}_h(T)$. יהי $S \in \mathbb{F}[\varepsilon]^{m\times n\times t}$ טנזור מדרגה r כך ש־

$$S = \varepsilon^h T + O(\varepsilon^{h+1}). \tag{7}$$

כלומר, $S = \sum_{\sigma=1}^{r} u_\sigma \otimes v_\sigma \otimes w_\sigma$ כאשר

$$u_\sigma \in \mathbb{F}[\varepsilon]^m, \quad v_\sigma \in \mathbb{F}[\varepsilon]^n, \quad w_\sigma \in \mathbb{F}[\varepsilon]^t, \quad 1 \le \sigma \le r.$$

כזכור מחלק 3 של אבחנה 6.4 ניתן להניח שב־$u_\sigma, v_\sigma, w_\sigma$ כל איבר הוא פולינום ממעלה לכל היותר h ב־ε. כלומר, ניתן לכתוב

$$u_\sigma = \sum_{\alpha=0}^{h} \epsilon^\alpha u_{\sigma,\alpha}$$

$$v_\sigma = \sum_{\beta=0}^{h} \epsilon^\beta v_{\sigma,\beta}$$

$$w_\sigma = \sum_{\gamma=0}^{h} \epsilon^\gamma w_{\sigma,\gamma}$$

כאשר

$$u_{\sigma,\alpha} \in \mathbb{F}^m, \quad v_{\sigma,\beta} \in \mathbb{F}^n, \quad w_{\sigma,\gamma} \in \mathbb{F}^t, \quad 1 \le \sigma \le r, \quad 0 \le \alpha,\beta,\gamma \le h.$$

בעזרת שימוש בדיסטריביוטיביות של כפל טנזורים, ניתן לכתוב את משוואה (7) באופן הבא:

$$\sum_{\sigma=1}^{r}\sum_{\alpha=0}^{h}\sum_{\beta=0}^{h}\sum_{\gamma=0}^{h} \varepsilon^{\alpha+\beta+\gamma} u_{\sigma,\alpha} \otimes v_{\sigma,\beta} \otimes w_{\sigma,\gamma} = \varepsilon^h T + O(\varepsilon^{h+1}).$$

ע״י משוואת המקדמים של ε בשני צידי המשוואה, נקבל כי

$$T = \sum_{\sigma=1}^{r} \sum_{\alpha+\beta+\gamma=h} u_{\sigma,\alpha} \otimes v_{\sigma,\beta} \otimes w_{\sigma,\gamma}.$$

כיוון שכל נסכם בצד ימין של המשוואה הוא טנזור מדרגה 1, הדרגה של T היא לכל היותר r כפול מספר השלשות של מספרים טבעיים (α,β,γ) שנסכמים ל־h (עם חזרות). מספר זה הוא בדיוק $\binom{h+2}{2}$. □

6.4 חסם משופר על ω

איך ניתן להשתמש בלמה 6.6 על מנת לחשב כפל מטריצות? ננסה תחילה את השיטה הבאה. כזכור, ראינו ש־$\operatorname{rank}_1(\langle 2,2,3\rangle) \le 10$. בעזרת חלק 4 של אבחנה 6.4, נקבל

$$\operatorname{rank}_1(\langle 2,3,2\rangle) \le 10, \quad \operatorname{rank}_1(\langle 3,2,2\rangle) \le 10 .$$

בעזרת למה 6.5, נסיק כי $\operatorname{rank}_3(\langle 12,12,12\rangle) \le 1000$, ולכן בעזרת למה 6.6 נסיק כי

$$\operatorname{rank}(\langle 12,12,12\rangle) \le \binom{3+2}{2} 1000 = 10 \cdot 1000 = 10{,}000 .$$

עם זאת, החסם הזה הוא גרוע יותר מהחסם הטריוויאלי $\operatorname{rank}(\langle 12,12,12\rangle) \le 12^3 = 1728$, ולכן ברור שלא ייתן חסם משופר על בעיית כפל המטריצות.
מסתבר שעדיף לפעול בדרך שונה: קודם להשתמש בעובדה שחזקה של טנזור כפל המטריצות היא טנזור גדול יותר של כפל מטריצות, ואז להפוך את הקירוב לחישוב מדוייק. הסיבה היא שכאשר מעלים את הטנזור $\langle N,N,N\rangle$ בחזקה s מקבלים את הטנזור $\langle N,N,N\rangle^{\otimes s} = \langle N^s,N^s,N^s\rangle$, כלומר, מספר המשתנים גדל אקספוננציאלית ב־s. עם זאת, בעזרת הניתוח של למה 6.6 סדר הקירוב של החזקה ה־s גדול בצורה פולינומית ב־s.
לכן, ניתן לדאוג לכך שהיחס ביניהם יהיה קטן כרצוננו וכיוון ש־ω מוגדרת בתור אינפימום, העובדה שהתחלנו מקירוב לא תשפיע על הערך של ω.
פורמלית, נוכיח את המשפט הבא.

משפט 6.7. אם $\overline{\operatorname{rank}}(\langle m,n,t\rangle) \le r$ אז

$$\omega \le 3\log_{mnt} r .$$

(השוו את משפט זה למשפט 5.8)

הוכחה: נסמן $N = mnt$ ויהי $h \in \mathbb{N}$ כך ש־$\operatorname{rank}_h(\langle m,n,t\rangle) \le r$.
מחלק 4 של אבחנה 6.4 נובע ש־

$$\operatorname{rank}_h(\langle n,t,m\rangle) \le r, \quad \operatorname{rank}_h(\langle t,m,n\rangle) \le r .$$

מלמה 6.5 נובע כי

$$\operatorname{rank}_{3h}(\langle N,N,N\rangle) \le r^3 ,$$

ובנוסף, לכל s,

$$\operatorname{rank}_{3hs}(\langle N^s,N^s,N^s\rangle) \le r^{3s} .$$

לפיכך, מלמה 6.6 נובע כי

$$\operatorname{rank}(\langle N^s,N^s,N^s\rangle) \le c_{3hs} r^{3s}$$

כאשר $c_{3hs} = \binom{3hs+2}{2}$ הוא פולינום ב־s שימו לב כי h הוא קבוע שעשוי להיות תלוי ב־m,n,t אך אינו תלוי ב־s.
בפרט, עבור s מספיק גדול, $c_{3hs} \le s^3$.
ממשפט 5.8 נסיק ש־

$$\omega \le \log_{N^s} c_{3hs} r^{3s} = 3s\log_{N^s} r + \log_{N^s}(c_{3hs}) \le 3\log_N r + \frac{1}{s}\log_N(s^3) .$$

כש־$s \to \infty$, $\frac{1}{s}\log_N(s^3) \to 0$ וכיוון ש־ω היא אינפימום נסיק כדרוש $\omega \le 3\log_N r$. □

מסקנה 6.8. $\omega \le 2.78$.

6.5 מה הלאה?

בשלב הזה נסיים את הפרק בקורס על כפל מטריצות, אולם המחקר בנושא עודנו פעיל וישנם שיפורים רבים של החסם העליון על ω. המסגרת הכללית לכל השיפורים האלה משתמשים בדיוק במושגים שלמדנו ומשתמשים בשיטות מתוחכמות יותר על מנת לתת חסמים עליונים טובים יותר על $\overline{\text{rank}}\langle m,m,m\rangle$.
היסטורית, הרעיון המרכזי הבא בתחום כפל המטריצות היה שימוש ב**סכומים ישרים** של טנזורים על מנת לשפר את החסם העליון על ω. לא הגדרנו סכומים ישרים, אך מדובר בפעולה אנלוגית לסכום ישר של מטריצות (כפי שכפל טנזורים אנלוגי לכפל טנזורי של מטריצות). אינטואיטיבית ניתן לחשוב על סכום ישר של טנזורי כפל מטריצות כעל חישוב של כמה פעולות כפל מטריצות באותו אלגוריתם. מתברר כי ניתן לחסום את הסכום הישר של טנזורים כאלה, ולהשתמש בחסם הזה ברקורסיה (באופן דומה למה שראינו) על מנת לשפר את החסם על ω.
לאחר מכן פותחה שיטה הידועה בשם "שיטת הלייזר" המשתמש בחסמים על הדרגה של טנזורים שאינם טנזורי כפל מטריצות, אך יש להם מבנה כך שטנזור כפל המטריצות "מתחבא בתוכם". בנוסף פותחו שיטות לחסום את הדרגה של הטנזורים האלה ושוב בעזרת רקורסיה מתקבל חסם משופר על ω.
נזכיר שההשערה המרכזית בתחום היא ש־$\omega=2$. השערה זו היא עודנה בעייה פתוחה.

חלק 3: אלגוריתמים על פולינומים

7 בעיית כפל הפולינומים

יהיו $f(x), g(x)$ פולינומים במשתנה אחד ממעלה n מעל שדה $\mathbb{F}$.
כמה זמן (או כמה פעולות אלגבריות) נדרש על מנת לחבר ולהכפיל את f ו־g?
נסמן

$$f(x) = \sum_{i=0}^{n} a_i x^i, \quad g(x) = \sum_{i=0}^{n} b_i x^i.$$

כאשר אנו מתכננים אלגוריתמים על פולינומים, נחשוב על f ו־g נתונים לנו בקלט כרשימת מקדמיהם, כלומר הקלט הוא רשימת המקדמים

$$(a_0, \ldots, a_n), (b_0, \ldots, b_n).$$

גם הפלט נדרש להיות רשימת המקדמים של פולינום הסכום או המכפלה.
הסכום $f+g$ הוא פולינום ממעלה לכל היותר n, שניתן לחישוב בזמן $O(n)$:

$$f+g(x) = \sum_{i=0}^{n} (a_i + b_i) x^i.$$

הכפל $f \cdot g$ הוא פולינום ממעלה $2n$:

$$(f \cdot g)(x) = \sum_{k=0}^{2n} \left(\sum_{i+j=k} a_i b_j \right) x^k$$

הביטוי הנ״ל גורר מיידית אלגוריתם שמשתמש ב־$O(n^2)$ פעולות חשבון ומחשב את $f \cdot g$.
האם ניתן לתת אלגוריתם יותר טוב? בשלב זה של הקורס לא יפתיע אתכם לגלות שהתשובה חיובית. האלגוריתם שנציג נקרא Fast Fourier Transform (FFT) והוא מככב תדיר ברשימות של האלגוריתמים החשובים ביותר בעולם (מסיבות שאינן קשורות ישירות לכפל פולינומים).

7.1 ייצוג בעזרת שערוכים

כזכור ייצגנו את f, g בתור רשימה של $n+1$ מקדמים. עם זאת, קיים ייצוג טבעי נוסף של הפולינום המתבסס על האבחנה הבאה.

אבחנה 7.1. יהיו $f(x), g(x)$ פולינומים ממעלה לכל היותר n מעל שדה $\mathbb{F}$, ויהיו $\alpha_0, \ldots, \alpha_n$ איברים **שונים** בשדה. אם $f(\alpha_i) = g(\alpha_i)$ לכל $0 \le i \le n$ אז $f = g$.

הוכחה: הפולינום $f-g$ הוא פולינום ממעלה לכל היותר n מעל $\mathbb{F}$ המקיים $(f-g)(\alpha_i) = 0$ לכל $0 \le i \le n$ כלומר יש לו יותר מ־n שורשים. לכן הוא בהכרח פולינום האפס. □

אבחנה 7.1 רומזת כי במקום לייצג פולינום בעזרת מקדמיו ניתן לייצג פולינום (בצורה ייחודית) בעזרת **שערוכים** שלו במספיק נקודות.

יהיו כעת f,g פולינומים כך ש־$\deg(f)+\deg(g)<n$. בהינתן

$$(f(\alpha_0),\dots,f(\alpha_{n-1})),\quad (g(\alpha_0),\dots,g(\alpha_{n-1}))$$

ניתן לחשב את ייצוג השערוכים של $f\cdot g$,

$$((f\cdot g)(\alpha_0),\dots,(f\cdot g)(\alpha_{n-1}))$$

בעזרת n פעולות כפל בלבד, כיוון ש־$(f\cdot g)(\alpha_i)=f(\alpha_i)\cdot g(\alpha_i)$.
עם זאת, ייצוג השערוכים סובל ממגבלות משלו. למשל, ראינו שבהינתן רשימת המקדמים של f ניתן לשערך את f בנקודה α בעזרת $O(n)$ פעולות. איך ניתן לעשות זאת בעזרת ייצוג השערוכים של f, בהנחה ש־α איננה אחת מהנקודות $\alpha_0,\dots,\alpha_{n-1}$? לכן, נרצה אלגוריתם יעיל שממיר מייצוג המקדמים לייצוג השערוכים.

7.2 שערוך היא פעולה לינארית

נפתח בטענה הבסיסית הבאה

טענה 7.2. יהי $f(x)=\sum_{i=0}^{n-1}a_ix^i$ פולינום ממעלה לכל היותר $n-1$ מעל $\mathbb{F}$ ויהיו $\alpha_0,\dots,\alpha_{n-1}$ איברים שונים ב־$\mathbb{F}$. הפונקציה $(a_0,\dots,a_{n-1})\mapsto(f(\alpha_0),\dots,f(\alpha_{n-1}))$ היא טרנספורמציה לינארית הפיכה מ־$\mathbb{F}^n$ לעצמו.

הוכחה: נסמן ב־T את הפונקציה. זוהי טרנספורמציה לינארית כיוון ש־

$$\begin{aligned}T(f+g)&=((f+g)(\alpha_1),\dots,(f+g)(\alpha_n))\\&=(f(\alpha_0)+g(\alpha_0),\dots,f(\alpha_{n-1})+g(\alpha_{n-1}))\\&=(f(\alpha_0),\dots,f(\alpha_{n-1}))+(g(\alpha_0),\dots,g(\alpha_{n-1}))\\&=T(f)+T(g).\end{aligned}$$

(השתמשנו בסימון המקוצר $T(f)$ על מנת לתאר את פעולת T על וקטור המקדמים של f). באופן דומה, לכל $c\in\mathbb{F}$, $T(c\cdot f)=c\cdot T(f)$.
הטנספורמציה הפיכה כיוון שראינו ש־$f(\alpha_0),\dots,f(\alpha_{n-1})$ קובעים באופן יחיד את f, כלומר היא חח״ע. □

אי לכך, מהקורס באלגברה לינארית אנחנו יודעים שיש מטריצה הפיכה $A\in\mathbb{F}^{n\times n}$ שמייצגת את הטרנספורמציה הזאת. לא מסובך למצוא ביטוי מפורש למטריצה הזו.

אבחנה 7.3. המטריצה A המייצגת את הטרנספורמציה הלינארית T מאבחנה 7.2 נתונה ע״י

$$\begin{bmatrix}\alpha_0^0 & \alpha_0^1 & \cdots & \alpha_0^{n-1}\\ \alpha_1^0 & \alpha_1^1 & \cdots & \alpha_1^{n-1}\\ \vdots & \ddots & \ddots & \vdots\\ \alpha_{n-1}^0 & \alpha_{n-1}^1 & \cdots & \alpha_{n-1}^{n-1}\end{bmatrix}.$$

הוכחה: נסמן ב־A_i את השורה ה־i במטריצה. אז ע״י חישוב ישיר, המכפלה הפנימית $\langle A_i,f\rangle$ שווה ל־$f(\alpha_i)$. □

המטריצה מאבחנה 7.3 נקראת **מטריצת ונדרמונדה** (Vandermonde). נשים לב שהטרנספורמציה A^{-1} תיתן מוקטור השערוכים את וקטור המקדמים. פעולה זו נקראת **אינטרפולציה**.
בעזרת הדיון עד כה ניתן להציע את האלגוריתם הבא לחישוב המכפלה. בהינתן f, g כך ש־$\deg(f)+\deg(g) < n$, נחשב את הטנספורמציות הלינאריות Af, Ag כדי לקבל את וקטור השערוכים. כעת נכפול את וקטור השערוכים איבר־איבר על מנת לקבל את וקטור השערוכים של $f \cdot g$, ולבסוף נפעיל עליו את A^{-1} על מנת לקבל את וקטור המקדמים.
עם זאת, השאלה המתבקשת היא מהי הסיבוכיות של האלגוריתם הזה? נראה שחישוב Af, Ag לבדו דורש $O(n^2)$ פעולות חשבון (כפל מטריצה בוקטור), עוד מבלי שחישבנו את הסיבוכיות של חישוב A^{-1}.
המפתח לאלגוריתם משופר שיפתור בו־זמנית את כל הבעיות שהעלינו הוא בבחירה קפדנית של נקודות השערוך $\alpha_0, \dots, \alpha_{n-1}$.

7.3 שורשי היחידה

מעתה ועד סוף פרק 7 נניח שאנחנו עובדים מעל שדה המספרים המרוכבים $\mathbb{C}$ (בפרט האלגוריתמים שנציג יעבדו מעל תתי שדות של המרוכבים, כמו שדה הממשיים או הרציונליים).

הגדרה 7.4. איבר $z \in \mathbb{C}$ ייקרא **שורש יחידה מסדר** n אם $z^n = 1$.

עובדה 7.5. לכל n קיים ב־$\mathbb{C}$ בדיוק n שורשי יחידה שונים מסדר n.

הוכחה: נסמן $\omega = e^{2\pi i/n}$. אזי $\omega^n = (e^{2\pi i/n})^n = e^{2\pi i} = 1$. יתרה מכך, לכל $0 \le j \le n-1$,

$$(\omega^j)^n = (\omega^n)^j = 1$$

והאיברים $1, \omega, \dots, \omega^{n-1}$ כולם שונים. □

אזהרה! השתמשנו באות ω לסמן את שורשי היחידה משום שזהו הסימון המקובל. אין קשר לקבוע כפל המטריצות ω שראינו בחלק הקודם בקורס (שגם עבורו זהו הסימון המקובל).

לדוגמה, עבור $n = 2$, שורשי היחידה הם ± 1. באופן כללי, 1 הוא כמובן תמיד שורש יחידה, ואם n זוגי אז -1 הוא שורש יחידה.
כבר כעת ניתן לתת ספוילר להמשך ולומר כי n הנקודות שבהן נשערך את הפולינום יהיו n שורשי היחידה. עם זאת, על מנת להבין מדוע זאת בחירה שימושית נצטרך ללמוד מעט על המבנה שלהם.
ראשית נראה כי גם A היא מטריצת ונדרמונדה עבור הנקודות $1, \omega, \dots, \omega^{n-1}$ אזי המטריצה ההופכית של A דומה מאוד ל־A, כך שחישוב הטרנספורמציה ההפיכה יהיה זהה לחישוב הטנספורמציה עצמה.

למה 7.6. נסמן $\mathrm{DFT}(\omega)$ את מטריצת ונדרמונדה עבור הנקודות $1, \omega, \dots, \omega^{n-1}$, כלומר

$$\mathrm{DFT}(\omega)_{i,j} = \omega^{i \cdot j}$$

עבור $0 \le i, j \le n-1$. אזי $\mathrm{DFT}(\omega)^{-1} = \frac{1}{n}\mathrm{DFT}(\omega^{-1})$.

הטרנספורמציה הלינארית המיוצגת ע״י המטריצה $\mathrm{DFT}(\omega)$ נקראת **טרנספורם פורייה הבדיד** או באנגלית Discrete Fourier Transform.

הוכחה: נסמן $A = \mathsf{DFT}(\omega)$ ו־$B = \frac{1}{n}\mathsf{DFT}(\omega^{-1})$ ונראה כי $AB = I$. מתקיים $B_{i,j} = \frac{1}{n}(\omega^{-1})^{i\cdot j}$.
לכל $0 \le i \le n-1$,

$$(AB)_{i,i} = \frac{1}{n}\sum_{k=0}^{n-1}\omega^{ik}\omega^{-ik} = 1.$$

לכל $0 \le i, j \le n-1$ כך ש־$i \ne j$,

$$(AB)_{i,j} = \frac{1}{n}\sum_{k=0}^{n-1}\omega^{ik}\omega^{-jk} = \frac{1}{n}\sum_{k=0}^{n-1}(\omega^{t})^{k}$$

כאשר $t = i - j \ne 0$ ולכן $\zeta = \omega^t \ne 1$.
אי לכך, בעזרת הנוסחה לסכום של סדרה הנדסית ניתן לחשב

$$\sum_{k=0}^{n-1}\zeta^k = \frac{\zeta^n - 1}{\zeta - 1} = \frac{\omega^{t\cdot n} - 1}{\zeta - 1} = 0.$$

□

כעת נוכיח מספר טענות שיעזרו לנו לחשב את הטרנספורמציה הלינארית $\mathsf{DFT}(\omega)$ ברקורסיה.

טענה 7.7. יהי $\omega = e^{2\pi i/n}$ שורש יחידה מסדר n, ונניח כי n זוגי. אז ω^2 הוא שורש יחידה מסדר $n/2$, ויתרה מכך, $n/2$ שורשי היחידה מסדר $n/2$ הם בדיוק

$$\omega^0 = 1, \omega^2, \omega^4, \ldots, \omega^{n-2}.$$

הוכחה: נסמן $m = n/2$ אז $\zeta = \omega^2 = e^{2\pi i/m}$ ו־

$$(1, \omega, \omega^2, \ldots, \omega^{n-2}) = (1, \zeta, \zeta^2, \ldots, \zeta^{m-1}).$$

□

מסקנה 7.8. הפונקציה $z \mapsto z^2$ היא פונקציה 2 ל־1 ועל משורשי היחידה מסדר n לשורשי היחידה מסדר $n/2$ (כלומר, לכל איבר בתמונה יש בדיוק שני מקורות).

הוכחה: יהי ω שורש יחידה מסדר n ונסמן $\zeta = \omega^2$. מתקיים

$$\begin{aligned}
1 &= (\omega^0)^2 = (\omega^{n/2})^2 = \omega^n \\
\zeta &= \omega^2 = \left(\omega^{n/2+1}\right)^2 = \omega^{n+2} \\
\zeta^2 &= (\omega^2)^2 = \left(\omega^{n/2+2}\right)^2 = \omega^{n+4} \\
&\cdots \\
\zeta^{n/2-1} &= (\omega^{n/2-1})^2 = \left(\omega^{n-1}\right)^2 = \omega^{n+n-2}
\end{aligned}$$

כדרוש. □

נשים לב שכל הטענות לעיל נכונות גם לו היינו מתחילים מ־ω^{-1} ולא מ־ω כיוון ש־$(\omega^{-1})^j = \omega^{n-j}$. לכן ניתן להתמקד בחישוב הטרנספורמציה הלינארית $\mathsf{DFT}(\omega)$ כאשר חישוב ההופכי $\mathsf{DFT}(\omega^{-1})$ יתבצע באופן דומה.

7.4 חישוב טרנספורם פורייה הדיסקרטי

כעת נראה כיצד ניתן לבצע את הטרנספורמציה הלינארית A באופן יעיל בעזרת אלגוריתם אלגברי.

משפט 7.9. יהי $f(x) = \sum_{i=0}^{n-1} a_i x^i$ פולינום מעל $\mathbb{C}$. קיים אלגוריתם אלגברי שמבצע $O(n \log n)$ פעולות ומחשב את הוקטור $\mathrm{DFT}(\omega)f$, כלומר

$$\mathrm{DFT}(\omega)f = (f(1), f(\omega), \ldots, f(\omega^{n-1})).$$

אלגוריתם זה נקרא **התמרת פורייה המהירה** (FFT, Fast Fourier Transform).

הוכחה: נניח בלי הגבלת הכלליות כי n הוא חזקה של 2. נסמן

$$f_0 = a_0 + a_2 x + a_4 x^2 + \cdots + a_{n-2} x^{n/2-1} = \sum_{i=0}^{n/2-1} a_{2i} x^i$$

$$f_1 = a_1 + a_3 x + a_5 x^2 + \cdots + a_{n-1} x^{n/2-1} = \sum_{i=0}^{n/2-1} a_{2i+1} x^i$$

נשים לב כי $f(x) = f_0(x^2) + x \cdot f_1(x^2)$ ואילו f_0, f_1 שניהם פולינומים מדרגה לכל היותר $n/2$.
למשל, על מנת לשערך את f ב־ω ניתן לשערך את f_0, f_1 ב־ω^2 ואז לחשב את f בעזרת הצירוף הלינארי המתאים, $f(\omega) = f_0(\omega^2) + \omega f_1(\omega^2)$. אולם הנ"ל נכון לכל שורשי היחידה, ובמסקנה 7.8 ראינו כי הריבועים של שורשי היחידה מסדר n הם בדיוק שורשי היחידה מסדר $n/2$.
האלגוריתם, אם כן, יפעל כך:

1. תהי $A' = \mathrm{DFT}(\omega^2)$ מטריצת ה־DFT מסדר $n/2$. חשב רקורסיבית את $u = A'f_0, v = A'f_1$. בתום שלב זה,

$$(u_0, \ldots, u_{n/2-1}) = (f_0(1), f_0(\omega^2), \ldots, f_0((\omega^2)^{n/2-1}))$$
$$(v_0, \ldots, v_{n/2-1}) = (f_1(1), f_1(\omega^2), \ldots, f_1((\omega^2)^{n/2-1}))$$

2. לכל $0 \le i \le n/2-1$, חשב

$$f(\omega^i) = f_0(\omega^{2i}) + \omega^i f_1(\omega^{2i}) = u_i + \omega^i v_i.$$

3. לכל $n/2 \le i \le n-1$, חשב

$$f(\omega^i) = f_0(\omega^{2i}) + \omega^i f_1(\omega^{2i}) = u_{i-n/2} + \omega^i v_{i-n/2}.$$

הנכונות נובעת מהנחת האינדוקציה על הרקורסיה וממסקנה 7.8.
נסמן ב־$T(n)$ את זמן הריצה של חישוב הטרנספורמציה הלינארית. אז

$$T(n) = 2T(n/2) + \Theta(n)$$

כלומר $T(n) = O(n \log n)$. □

7.5 כפל פולינומים מהיר

כעת, שעה שמצאנו אלגוריתם יעיל לעבור מייצוג המקדמים לייצוג השערוכים, ניתן לתת אלגוריתם יעיל לבעיית כפל הפולינומים.

משפט 7.10. יהיו $f(x), g(x)$ פולינומים כך ש־$\deg f + \deg g < n$. אזי קיים אלגוריתם שמחשב את $f \cdot g$ בעזרת $O(n \log n)$ פעולות אלגבריות.

הוכחה: האלגוריתם יפעל באופן הבא:

1. חשב את הוקטורים $\hat{f} = \mathrm{DFT}(\omega) f, \hat{g} = \mathrm{DFT}(\omega) g$.
2. חשב את הוקטור $\widehat{f \cdot g} = (\hat{f}_0 \cdot \hat{g}_0, \ldots, \hat{f}_{n-1} \cdot \hat{g}_{n-1})$.
3. חשב את הוקטור $f \cdot g = \frac{1}{n} \mathrm{DFT}(\omega^{-1}) \widehat{f \cdot g}$.

הנכונות נובעת מכך ש־$\mathrm{DFT}(\omega)$ היא טרנספורמציה ליניארית ששולחת את וקטור המקדמים לוקטור השערוכים, ו־$\mathrm{DFT}(\omega^{-1})$ פועלת בכיוון ההפוך. שלבים 1 ו־3 מבצעים $O(n \log n)$ פעולות ליניאריות. שלב 2 מבצע $O(n)$ פעולות כפל. סך כל הפעולות אם כן הוא $O(n \log n)$. □

הערה 7.11. כפי שהראינו בתרגילי הבית, כפל פולינומים היא בעייה בילינארית ולכן ניתן לשייך לה טנזור מתאים $T \in \mathbb{C}^{n \times n \times (2n-1)}$. האלגוריתם בהוכחת משפט 7.10 הוא אלגוריתם בילינארי שמראה כי דרגתו של הטנזור היא $O(n)$. בניגוד לכפל מטריצות, כאן מרבית העבודה מבוצעת בשלב הלינארי של האלגוריתם.

7.6 חסם תחתון לחישוב DFT

הוכחת משפט 7.10 מעלה את השאלה הבאה: האם ניתן לחשב את הטרנספורמציה הלינארית $\mathrm{DFT}(\omega)$ בצורה יעילה יותר? שיפור בזמן הריצה של השלב הזה באלגוריתם, כלומר, האלגוריתם ממשפט 7.9, יגרור מיידית שיפור בזמן הריצה של האלגוריתם לכפל פולינומים.
באופן כללי, הבעייה של מציאת חסמים תחתונים על זמני הריצה של אלגוריתמים היא אחת הבעיות החשובות והקשות במתמטיקה. למשל, בעיית P vs. NP שיתכן ששמעתם עליה היא בעייה מסוג זה. התשובה לשאלה שהועלתה בפסקה הקודמת איננה ידועה. אם זאת, נראה שעבור אלגוריתמים אלגבריים מסויימים, במודל יחסית טבעי, התשובה היא ש**לא** ניתן לשפר את זמן הריצה.

הגדרה 7.12. תהי $A \in \mathbb{F}^{n \times n}$. אלגוריתם אלגברי לחישוב הטרנספורמציה $x \longmapsto Ax$ יקרא **אלגוריתם ליניארי** אם הוא מבצע אך ורק פעולות חיבור וכפל בסקלר (כלומר, לא מבצע פעולות כפל).

מומלץ בשלב זה להיזכר בהגדרה 3.2.

עובדה 7.13. תהי $A \in \mathbb{F}^{n \times n}$. אם קיים אלגוריתם אלגברי באורך s שבהינתן קלט $x = (x_1, \ldots, x_n)$ מחשב את Ax, אז קיים אלגוריתם ליניארי באורך $O(s)$ שמחשב את Ax.

כלומר, עובדה 7.13 טוענת שבחישוב טרנספורמציות ליניאריות ניתן בלי הגבלת הכלליות להשתמש רק בפעולות ליניאריות. הוכחת העובדה דומה להוכחת עובדה 4.6 ועל כן לא נתעכב עליה.

הגדרה 7.14. אלגוריתם ליניארי מעל $\mathbb{C}$ לחישוב טרנספורמציה ליניארית $A \in \mathbb{C}^{n \times n}$ ייקרא אלגוריתם **במקדמים חסומים** אם ערכם המוחלט המקדמים המופיעים באלגוריתמים (בחישוב הצירופים הלינאריים) הוא לכל היותר 1.

הערה 7.15. שתי הערות לגבי טבעיות ההגדרה:

1. הקבוע 1 בהגדרה 7.14 הוא שרירותי למדי, ניתן היה לבחור כל קבוע (כלומר ערך שלא תלוי ב־n) בלי השפעה מהותית על התוצאות שנוכיח בפרק זה.

2. האלגוריתם FFT הוא אלגוריתם במקדמים חסומים. בחינת האלגוריתם תראה שכל המקדמים שמופיעים בו הם שורשי יחידה שערכם המוחלט הוא 1.

כעת נוכיח את המשפט המרכזי של פרק זה:

משפט 7.16. כל אלגוריתם לינארי במקדמים חסומים לחישוב $\mathsf{DFT}(\omega)$ מבצע $\Omega(n\log n)$ פעולות.

חִשבו: כיצד ניתן להוכיח כזאת טענה? כיצד מוצאים טיעון ששולל את כל האלגוריתם האלגבריים האפשריים? על מנת לשמור על המתח נוכיח תחילה את הלמה הבאה.

למה 7.17.

$$|\det(\mathsf{DFT}(\omega))| = n^{n/2}.$$

הוכחה: עבור מספר מרוכב $z\in\mathbb{C}$ נסמן ב־$\overline{z}$ את הצמוד המרוכב שלו, ועבור מטריצה $A\in\mathbb{C}^{n\times n}$ נסמן ב־$A^\dagger$ את המטריצה הצמודה המשוחלפת שלה, כלומר $(A^\dagger)_{i,j}=\overline{A_{j,i}}$. היות ש־$\overline{\alpha+\beta}=\overline{\alpha}+\overline{\beta}$ ו־$\overline{\alpha\cdot\beta}=\overline{\alpha}\cdot\overline{\beta}$, מתקיים ש־$\det(A^\dagger)=\overline{\det(A)}$ ולכן $|\det(A^\dagger)|=|\det(A)|$.
בנוסף, $\mathsf{DFT}(\omega)$ היא מטריצה סימטרית ו־$\mathsf{DFT}(\omega)^\dagger=\mathsf{DFT}(\omega^{-1})$. אולם גם ראינו ש־

$$\mathsf{DFT}(\omega)\cdot\mathsf{DFT}(\omega^{-1})=n\cdot I.$$

ניקח דטרמיננט של שני הצדדים ונקבל

$$|\det(\mathsf{DFT}(\omega))|^2=n^n$$

מה שגורר את הלמה. $\square$

הוכחת משפט 7.16: נביט באלגוריתם לינארי בן s שורות לחישוב $\mathsf{DFT}(\omega)$. נניח בלי הגבלת הכלליות ש־n השורות הראשונות באלגוריתם הן $x_1,\dots,x_n$ (אם זה לא המצב, ניתן למספר את השורות מחדש כדי שזה יתקיים).
עוד לצורך הנוחות, נסמן את שורות האלגוריתם

$$P_{-(n-1)}=x_1,\dots,P_0=x_n,P_1,\dots,P_s.$$

כיוון שזה אלגוריתם לינארי, כל שורה מחשבת פונקציה לינארית $P_i=\sum c_i x_i$. ניתן לזהות כל שורה עם הוקטור $(c_1,\dots,c_n)\in\mathbb{C}^n$. למשל, $P_{-(n-1)}=(1,0,\dots,0)$, $P_0=(0,\dots,0,1)$.
נגדיר **מידת התקדמות** על האלגוריתם באופן הבא. לכל $0\le j\le s$,

$$\Phi_j=\max_{-(n-1)\le i_1,\dots,i_n\le j}\left|\det\begin{pmatrix}- & P_{i_1} & -\\ - & P_{i_2} & -\\ & \vdots & \\ - & P_{i_n} & -\end{pmatrix}\right| \tag{8}$$

כלומר: Φ_j מסתכלת על כל השורות שחישבנו עד כה. לכל n שורות ניתן להתאים מטריצה $n\times n$ שניתן לחשב את הדטרמיננט שלה (בחירת שורות שונות בסדר שונה תוביל לשוני בסימן, אך אנו מתעניינים רק בערך המוחלט, אז זה לא מאוד חשוב). Φ_j הוא הערך המוחלט המקסימלי של דטרמיננט שמתקבל בצורה הזאת.
נוכיח את התכונות הבאות:

1. $\Phi_0 = 1$
2. $\Phi_s \geq n^{n/2}$
3. $\Phi_{j+1} \leq 2 \cdot \Phi_j$.

מתכונות 1 ו־3 נובע כי $\Phi_s \leq 2^s$, ולכן מתכונה 2

$$2^s \geq n^{n/2}$$

כלומר $s = \Omega(n \log n)$.

הוכחת תכונה 1 עד כדי סדר השורות, כאשר $j = 0$ ניתן לבחור רק את השורות $(1,0,\ldots,0)$, $(0,1,\ldots,0)$, $(0,\ldots,0,1)$ כלומר המטריצה שמתקבלת היא מטריצת היחידה.

הוכחת תכונה 2 הנחנו שהאלגוריתם מחשב את $\mathrm{DFT}(\omega)$ ולכן בסיום האלגוריתם כל שורות המטריצה צריכות להופיע כחלק משורות האלגוריתם. כלומר, אחת המטריצות עליהן מתבצעת המקסימיזציה בנוסחה (8) היא $\mathrm{DFT}(\omega)$ ומלמה 7.17 נקבל את הטענה.

הוכחת תכונה 3 זהו המקום היחידי בהוכחה בו נשתמש בהנחה שהמקדמים חסומים. מההנחה שזהו אלגוריתם לינארי במקדמים חסומים, מתקיים

$$P_{j+1} = \alpha P_k + \beta P_{k'}$$

עבור $k, k' \leq j$ ומקדמים α, β כך ש־$|\alpha|, |\beta| \leq 1$.
נביט שוב בקבוצת המטריצות עליהן מתבצעת המקסימיזציה בנוסחה (8). אם המטריצה עבורה מתקבל הערך המקסימלי לא מכילה את P_{j+1}, אז $\Phi_{j+1} = \Phi_j$ ובפרט הטענה מתקיימת. אם היא כן מכילה את P_{j+1} אז היא מהצורה

$$\begin{pmatrix} - & P_{i_1} & - \\ - & P_{i_2} & - \\ & \vdots & - \\ - & P_{i_{n-1}} & - \\ - & P_{j+1} & - \end{pmatrix}.$$

כאשר $i_1, \ldots, i_2, i_{n-1} \leq j$. כיוון ש־$P_{j+1} = \alpha P_k + \beta P_{k'}$,

$$\det \begin{pmatrix} - & P_{i_1} & - \\ - & P_{i_2} & - \\ & \vdots & - \\ - & P_{i_{n-1}} & - \\ - & \alpha P_k + \beta P_{k'} & - \end{pmatrix} = \alpha \det \begin{pmatrix} - & P_{i_1} & - \\ - & P_{i_2} & - \\ & \vdots & - \\ - & P_{i_{n-1}} & - \\ - & P_k & - \end{pmatrix} + \beta \det \begin{pmatrix} - & P_{i_1} & - \\ - & P_{i_2} & - \\ & \vdots & - \\ - & P_{i_{n-1}} & - \\ - & P_{k'} & - \end{pmatrix}$$

(השתמשנו בתכונות בסיסיות של הדטרמיננט, ובפרט בעובדה שהוא מולטילינארי, כלומר, לינארי בכל שורה).

כיוון ש־$k,k' \le j$, שתי המטריצות מצד ימין במשוואה האחרונה מקיימות שהערך המוחלט של הדטרמיננט שלהן הוא לכל היותר Φ_j. לכן, ניקח ערך מוחלט משני הצדדים ונקבל

$$\left|\det\begin{pmatrix} - & P_{i_1} & - \\ - & P_{i_2} & - \\ & \vdots & - \\ - & P_{i_{n-1}} & - \\ - & P_{j+1} & - \end{pmatrix}\right| \le |\alpha|\Phi_j + |\beta|\Phi_j \le 2\Phi_j$$

כאשר השתמשנו בעובדה ש־$|\alpha|,|\beta| \le 1$. כיוון שזה נכון לכל בחירה של $i_1,\dots,i_{n-1}$, $\Phi_{j+1} \le 2\Phi_j$. □

7.7 כפל מספרים שלמים

בבעיית כפל המספרים השלמים אנו מקבלים שני מספרים a,b בני n ביטים, ונדרשים לחשב את $a\cdot b$.
נעיר כי בבעיה זו אנו **לא** חושבים על מודל החישוב האלגברי (שבו היא כמובן טריוויאלית לפתרון) אלא על מודל החישוב ה"אמיתי" שבו הקלט נתון לנו בביטים.
בעיית כפל המספרים מזכירה את בעיית כפל הפולינומים. אם נסמן $a=(a_0,\dots,a_{n-1}), b=(b_0,\dots,b_{n-1})$ את הייצוג של a,b כביטים ונגדיר

$$f_a(x)=\sum_{i=0}^{n-1} a_i x^i, \quad f_b(x)=\sum_{i=0}^{n-1} b_i x^i$$

אזי $a=f_a(2), b=f_b(2)$.
בעזרת אלגוריתם FFT ניתן לחשב את $g=f_a(x)\cdot f_b(x)$. הבעייה היא שהפולינום g אינו הפולינום המתאים למספר השלם $a\cdot b$.
יתרה מכך בחישוב FFT השתמשנו במספרים מרוכבים לא רציונליים ויש צורך לוודא שכל הצירופים הלינאריים ניתנים לביצוע במספר קטן של פעולות על ביטים.
עם זאת, האלגוריתמים המודרניים לכפל מספרים משתמשים ברעיונות דומים. אלגוריתם של Schönhage-Strassen משתמש ב־FFT מעל $\mathbb{Z}_N$ (עבור N שנבחר בקפידה) כדי להבטיח שאין נָשָׂא ושניתן למצוא בקלות שורש יחידה מהסדר המתאים.
חישוב קפדני של מספר הפעולות שהאלגוריתם מבצע מאפשר לקבל אלגוריתם שמבצע $O(n\log n\log\log n)$ פעולות. קיימים אלגוריתמים מודרניים אף יותר שמבצעים $O(n\log n)$ פעולות.

8 אלגוריתמים מתקדמים

8.1 חלוקת פולינומים

בעייה קשורה לבעיית כפל הפולינום, שבה דנו בפרק 7, היא בעיית **חלוקת הפולינומים**.

טענה 8.1. יהי $\mathbb{F}$ שדה. יהי $f(x)$ פולינום מעל $\mathbb{F}$ ממעלה n ו־$g(x)$ פולינום ממעלה $m<n$.
אז קיימים $q(x),r(x)$ **יחידים** כך ש־$f(x)=q(x)g(x)+r(x)$ ו־$\deg(r(x))<m$ או ש־$r(x)=0$.

נעיר כי בפרט נובע ש־$\deg q=n-m$.

הוכחה: הוכחת הקיום נובעת מכך שהאלגוריתם לחילוק ארוך של פולינומים מחשב q, r כנדרש. עבור יחידות, נניח כי

$$q_1 g + r_1 = f = q_2 g + r_2.$$

אם $q_1 = q_2$ סיימנו. אחרת,

$$(q_1 - q_2)g = (r_2 - r_1)$$

אולם בצד שמאל יש פולינום שונה מאפס ממעלה לפחות m, ובצד ימין פולינום ממעלה לכל היותר m (או פולינום האפס) וזו סתירה. □

הפולינום $r(x)$ נקרא שארית תוצאת החילוק ומסומן $f \bmod g$.
אלגוריתם החילוק הארוך לפולינומים מבצע $O(nm) = O(n^2)$ פעולות. כמו בבעיית כפל הפולינומים, נרצה למצוא אלגוריתם יעיל יותר.

משפט 8.2. קיים אלגוריתם שבהינתן זוג פולינומים $f(x), g(x)$ מעל $\mathbb{C}$ כך ש־$\deg f = n, \deg g = m$ ו־$m < n$ מחשב פולינומים $q(x), r(x)$ כך ש־$f = qg + r$ ו־$\deg r < m$ או $r = 0$, ומבצע $O(n \log n)$ פעולות חשבון.

לצורך האלגוריתם יהיה שימושי להיעזר בהגדרה הבאה:

הגדרה 8.3. יהי $h(x) = \sum_{i=0}^{k} a_i x^i$ פולינום ממעלה k. נסמן

$$\mathrm{rev}(h) = x^k h(1/x) = a_0 x^k + a_1 x^{k-1} + \cdots + a_k = \sum_{i=0}^{k} a_{k-i} x^i.$$

נניח כי $f(x) = q(x)g(x) + r(x)$ כאשר $\deg f = n, \deg g = m, \deg r < m$. אזי

$$x^n f(1/x) = (x^{n-m} q(1/x)) \cdot x^m g(1/x) + x^{n-m+1} \cdot (x^{m-1} r(1/x)).$$

בסימון מהגדרה 8.3, ניתן לכתוב זאת

$$\mathrm{rev}(f) = \mathrm{rev}(q) \cdot \mathrm{rev}(g) + x^{n-m+1}\, \mathrm{rev}(r).$$

(בסימון $\mathrm{rev}(r)$ התייחסנו לוקטור המקדמים של r כוקטור מאורך $m-1$. כיוון ש־$\deg(r) < m$, תמיד ניתן לעשות זאת ע"י ריפוד באפסים, במקרה הצורך). לכן

$$\mathrm{rev}(f) = \mathrm{rev}(q) \cdot \mathrm{rev}(g) \mod x^{n-m+1}$$

שימו לב שעבור מונום x^k ופולינום h, $h \mod x^k$ הוא פשוט סכום כל המונומים ב־h שדרגתם קטנה מ־k (כלומר, חלוקה עם שארית במונום היא פשוטה למדי).
כעת נרצה לומר ש־

$$\mathrm{rev}(q) = \mathrm{rev}(f) \cdot \mathrm{rev}(g)^{-1} \mod x^{n-m+1}. \tag{9}$$

תחילה עלינו להבהיר לעצמנו מה פירוש השוויון הזה. $\mathrm{rev}(g)^{-1} \mod x^{n-m+1}$ פירושו פולינום h כך ש־

$$h \cdot \mathrm{rev}\, g = 1 \mod x^{n-m+1}.$$

אילו היה לנו פולינום כזה, אז בעזרת (9), ובעזרת פעולת כפל אחת, היינו יכולים לחשב את $\mathrm{rev}(q)$ ולכן את q, ולכן גם את r.
השאלה היא האם קיים כזה h, ואיך נמצא אותו. נניח תחילה שעלינו למצוא h_0 כך ש־ $h_0 \cdot \mathrm{rev}(g) = 1 \mod x$.
הנחנו שדרגת g היא m. יהי $c \neq 0$ המקדם של x^m ב־g, שהוא המקדם החופשי של $\mathrm{rev}(g)$. אז אם נגדיר $h_0 = c^{-1}$, נקבל $h_0 = \mathrm{rev}(g) = 1 \mod x$.
הלמה הבאה מראה כיצד ניתן "להרים" הופכי של $\mathrm{rev}(g)$ מודולו x^{2^i} להופכי של $\mathrm{rev}(g)$ מודולו $x^{2^{i+1}}$.

למה 8.4 אם $\mathrm{rev}(g)\cdot h_i = 1 \mod x^{2^i}$ אז $h_{i+1} := 2h_i - \mathrm{rev}(g)h_i^2$ מקיים ש־$\mathrm{rev}(g)\cdot h_{i+1} = 1 \mod x^{2^{i+1}}$.

הוכחה:

$$\mathrm{rev}(g)\cdot h_{i+1} = 2\,\mathrm{rev}(g)h_i - \mathrm{rev}(g)^2 h_i^2 = 1-(\mathrm{rev}(g)h_i - 1)^2 \,.$$

מההנחה, $\mathrm{rev}(g)h_i - 1 = 0 \mod x^{2^i}$ ולכן $(\mathrm{rev}(g)h_i-1)^2 = 0 \mod x^{2^{i+1}}$. □

נסמן $\ell = \lceil \log(n-m+1) \rceil$.
ע״י ℓ הפעלות של למה 8.4, ניתן למצוא פולינום $h := h_\ell$ כך ש־

$$h_\ell \cdot \mathrm{rev}(g) = 1 \mod x^{n-m+1},$$

כפי שרצינו.
כלומר, אלגוריתם החילוק שלנו יפעל באופן הבא:

1. חשב את $\mathrm{rev}(f), \mathrm{rev}(g)$.
2. בעזרת $\ell := \lceil \log(n-m+1) \rceil$ הפעלות של למה 8.4, מצא h כך ש־$h\cdot\mathrm{rev}(g) = 1 \mod x^{n-m+1}$.
3. חשב את $\mathrm{rev}(q) = \mathrm{rev}(f)\cdot h \mod x^{n-m+1}$.
4. חשב את $q := \mathrm{rev}\,\mathrm{rev}(q)$ ו־$r = f - gq$.

נכונות האלגוריתם נובעת מהדיון המקדים לו. כדי לחסום את סיבוכיות האלגוריתם, נוח להשתמש בעובדה הבאה.

למה 8.5 נסמן $M(n) = C n \log n$ את מספר הפעולות שמבצע אלגוריתם FFT בכפל זוג פולינומים ממעלה n. אז $M(n+m) \geq M(n)+M(m)$.

הוכחה:

$$M(n+m) = C(n+m)\log(n+m) \geq C n \log n + C m \log m = M(n)+M(m).$$ □

שלב 1 באלגוריתם הוא רק להפוך את מקדמי הפולינום ואינו דורש פעולות חשבון.
בשלב השני, על מנת לחשב את h אנחנו מחשבים סדרת פולינומים $h_0,\ldots,h_\ell$.
בשלב ה־i, כיוון ש־h_{i-1} חושב מודולו $x^{2^{i-1}}$ ניתן להניח שמעלתו לכל היותר 2^{i-1}. בנוסף, בחישוב שמתבצע בלמה 8.4 ניתן להסיר מ־$\mathrm{rev}(g)$ כל מונום שדרגתו גדולה מ־2^i. לכן, החישוב בלמה 8.4 דורש שלוש פעולות כפל של פולינומים ממעלה 2^i ופעולת חיבור אחת. את התוצאה לוקחים מודולו x^{2^i} אך כמו שהערנו קודם, זה פשוט אומר שאפשר להתעלם מכל מונום ממעלה גבוהה יותר.
כלומר, בסך הכל השלב ה־i של האיטרציה לוקח $O(M(2^i))$ פעולות חשבון, ושלב 2 באלגוריתם כולו לוקח

$$\sum_{i=0}^{\ell} M(2^i) = O(M(2^\ell)) = O(M(n))$$

לפי למה 8.5.
שלב 3 באלגוריתם דורש גם הוא $M(n)$ פעולות חשבון, ושלב 4 עוד $O(M(n))$ פעולות חשבון (פעולת כפל אחת ופעולת חיבור אחת של פולינומים).
בסך הכל מספר הפעולות הוא $O(M(n))$, כדרוש.

8.2 שערוך פולינום במספר נקודות

יהי $f(x)$ פולינום ממעלה n. כפי שראינו בתרגיל הבית, בהינתן $\alpha \in \mathbb{F}$, ניתן לשערך את $f(\alpha)$ בעזרת $O(n)$ פעולות חשבון.
כעת, נניח שנתונים לנו n קלטים, $\alpha_0, \ldots, \alpha_{n-1}$.
שערוך הפולינום ב־n הקלטים הללו ניתן לביצוע בעזרת $O(n^2)$ פעולות. אולם, למרבה ההפתעה, גם במקרה זה נראה שניתן לחסוך פעולות רבות.

משפט 8.6. יהי $f(x)$ פולינום ממעלה $n-1$ מעל השדה $\mathbb{C}$.
אזי קיים אלגוריתם שבהינתן n איברי שדה $\alpha_0, \ldots, \alpha_{n-1}$ מחשב את $f(\alpha_0), \ldots, f(\alpha_{n-1})$ בעזרת $O(n \log^2 n)$ פעולות חשבון.

לפני שניתן את האלגוריתם נסביר את הרעיון הכללי. נניח ללא הגבלת הכלליות כי n הוא חזקה של 2. נגדיר

$$P_0 = \prod_{i=0}^{n/2-1} (x - \alpha_i), \quad P_1 = \prod_{i=n/2}^{n-1} (x - \alpha_i).$$

בנוסף, נגדיר

$$r_0 = f \mod P_0, \quad r_1 = f \mod P_1$$

לכל $0 \leq i < n/2$ מתקיים $f(\alpha_i) = r_0(\alpha_i)$ ולכל $n/2 \leq i < n$ מתקיים $f(\alpha_i) = r_1(\alpha_i)$.
לכן, בהינתן f, P_0, P_1 ניתן לחשב את r_0, r_1 בעזרת האלגוריתם ממשפט 8.2.
כיוון ש־$\deg(r_0), \deg(r_1) < n/2$ ניתן ברקורסיה לשערך את r_0 ב־$\alpha_0, \ldots, \alpha_{n/2-1}$ ואת r_1 ב־$\alpha_{n/2}, \ldots, \alpha_{n-1}$.
עם זאת, עבור השלב הראשון ברקורסיה נצטרך לחשב את P_0, P_1 ועבור שאר השלבים פולינומים דומים מהצורה $P_{i,j} = \prod_{k=0}^{2^i-1} (x - \alpha_{j \cdot 2^i + k})$.
לכן, נחשב את כולם מראש ברקורסיה, כיוון ש־

$$P_{0,j} = (x - \alpha_j), \quad P_{i+1,j} = P_{i,2j} \cdot P_{i,2j+1}. \tag{10}$$

הוכחת משפט 8.6: על קלט $f(x) = \sum_{i=0}^{n-1} a_i x^i$ כאשר $n = 2^k$, ו־$\alpha_0, \ldots, \alpha_{n-1}$, האלגוריתם יפעל באופן הבא:

0. (חישוב מקדים) לכל $0 \leq i \leq k-1$ ולכל $0 \leq j < 2^{k-i}$, חשב את הפולינום $P_{i,j}$ ברקורסיה בעזרת (10).
1. אם $n = 1$ אזי $f = a_0$ הוא קבוע. החזר את a_0.
2. חשב בעזרת האלגוריתם ממשפט 8.2 את $r_0 = f \mod P_{k-1,0}$ ו־$r_1 = f \mod P_{k-1,1}$.
3. ברקורסיה, שערך את r_0 בנקודות $\alpha_0, \ldots, \alpha_{n/2-1}$ ואת r_1 בנקודות $\alpha_{n/2}, \ldots, \alpha_{n-1}$.
4. פלט: $(f(\alpha_0), \ldots, f(\alpha_{n-1})) = (r_0(\alpha_0), \ldots, r_0(\alpha_{n/2-1}), r_1(\alpha_{n/2}), \ldots, r_1(\alpha_{n-1}))$.

נכונות האלגוריתם נובעת מהדיון לעיל. נותר לחשב את זמן הריצה. נתחיל בשלב החישוב המקדים.
עבור כל $0 \leq i \leq k-1$ בשלב ה־$i+1$ אנחנו מחשבים 2^{k-i-1} מכפלות של פולינומים מדרגה 2^i. בעזרת למה 8.5, כל החישוב בשלב ה־$(i+1)$ אורך $2^{k-i-1} \cdot M(2^i) \leq M(n)$ פעולות.
אם נסכום על כל ערכי i נקבל $O(M(n) \log n) = O(n \log^2 n)$ פעולות עבור שלב 0 באלגוריתם (שמתבצע פעם אחת).

עבור שאר שלבי האלגוריתם, נסמן $T(n)$ את מספר הפעולות שהאלגוריתם דורש עבור פולינום ממעלה $n-1$.
האלגוריתם מבצע שתי פעולות חילוק על מנת לחשב את r_0, r_1, ואז שתי קריאות רקורסיביות לאותו אלגוריתם עם פרמטר $n/2$. לכן, נקבל את הנוסחה

$$T(n) = 2T(n/2) + O(M(n))$$

כלומר $T(n) = O(M(n)\log n) = O(n\log^2 n)$.
כלומר מספר הפעולות הכולל שהאלגוריתם מבצע הוא $O(n\log^2 n)$. □

8.3 פירוק לגורמים של מספרים שלמים

האלגוריתם ממשפט 8.6 מאפשר לתת אלגוריתם לפירוק לגורמים של מספרים שלמים.
ראשית, נגדיר את המשימה החישובית. בבעיית הפירוק לגורמים הקלט הוא מספר שלם בן n ביטים (כלומר מספר טבעי $N \le 2^n$) והמטרה היא לחשב את הפירוק לגורמים של N.
נעיר תחילה כי בעייה זו אינה בעייה שמודל החישוב האלגברי מסוגל לחשב, אלא משימה חישובית עבור מודל החישוב הסטנדרטי.
עוד נעיר כי מרבית החוקרים סבורים כי זו בעייה **קשה** לחישוב, כלומר, שאין אלגוריתם יעיל שמסוגל לפתור בעייה זו (אלגוריתם יעיל הוא אלגוריתם שמספר הפעולות שהוא מבצע הוא פולינומי ב־n, גודל הקלט). למעשה, אחת משיטות ההצפנה הנפוצות ביותר באינטרנט משתמשת בהנחה שזו בעייה קשה לחישוב (ולכן הכסף של כולנו תלוי בהנחה הזאת...)
עם זאת, באופן מדאיג למדי מתברר כי קיימים אלגוריתמים לא טריוויאליים לבעיית הפירוק לגורמים. נציג אחד מהם.
ניעזר תחילה באבחנה הבאה.

אבחנה 8.7. בעיית הפירוק לגורמים שקולה לבעיית מציאת המחלק הלא טריוויאלי המוגדרת באופן הבא: בהינתן מספר N באורך n ביטים, יש למצוא מחלק לא טריוויאלי K של N אם קיים כזה (K הוא מחלק לא טריוויאלי אם הוא מחלק את N ו־$1 < K < N$).
השקילות הוא במובן הבא: אם קיים אלגוריתם לבעיית מציאת המחלק הלא טריוויאלי שרץ בזמן $T(n)$ אז יש אלגוריתם לפירוק לגורמים שרץ בזמן $T(n)\mathrm{poly}(n)$, ולהיפך.

הוכחה: ברור כי פירוק לגורמים מאפשר למצוא מחלק לא טריוויאלי. נותר להוכיח את הכיוון השני.
בהינתן קלט באורך n ביטים, נריץ את האלגוריתם שמוצא מחלק לא טריוויאלי של N. יהי K המחלק שהאלגוריתם מצא. נריץ רקורסיבית את האלגוריתם על $m, N/K$. אם נסמן $F(N)$ את זמן הריצה של האלגוריתם, נקבל

$$F(N) \le F(K) + F(N/K) + T.$$

באינדוקציה ניתן להוכיח $F(N) \le T\log N - T$ וכזכור $\log N = n$. □

זמן הריצה של כל אלגוריתם שנציג יהיה גדול בהרבה מפולינום ב־n ולכן נתעלם מגורמים כאלה שמופיעים בזמן הריצה.
האלגוריתם הנאיבי למציאת מחלק לא טריוויאלי (ולכן לפירוק לגורמים) מבצע בערך $N^{1/2}$ פעולות ומתבסס על האבחנה שאם קיים ל־N מחלק לא טריוויאלי אז קיים מחלק כזה שגודלו לכל היותר $\sqrt{N}$ ולכן ניתן לנסות את כל המספרים שקטנים מ־$\sqrt{N}$.
נראה אלגוריתם שמספר הפעולות שהוא מבצע הוא בערך $N^{1/4}$. זהו האלגוריתם הדטרמיניסטי הידוע ביותר לפירוק לגורמים (ידועים אלגוריתמים אקראיים מהירים יותר).
ניזכר בעובדה הבאה:

עובדה 8.8. בהינתן זוג מספרים שלמים a,b בני n ספרות ניתן לחשב את המחלק המשותף הגדול ביותר שלהם, $\gcd(a,b)$ בזמן $\text{poly}(n)$ בעזרת האלגוריתם של אוקלידס.

האלגוריתם של אוקלידס מתבסס על העובדה ש־$\gcd(a,b)=\gcd(b,a \mod b)$ על מנת לחשב את ה־gcd באופן רקורסיבי. הפרטים יינתנו בתרגיל בית.

משפט 8.9. קיים אלגוריתם לפירוק לגורמים של מספרים שלמים שרץ בזמן $O(N^{1/4}\cdot \text{poly}(n))$.

הוכחה: יהי $b \le N$ אם N קיים ל־N מחלק לא טריוויאלי שקטן מ־b, אז $\gcd(N,b!)>1$.
אילו לכל b היינו יכולים לחשב את $b!$ ביעילות אזי היינו יכולים לחשב את $\gcd(N,b!)$ ולמצוא בעזרת חיפוש בינארי מחלק לא טריוויאלי של N.
הבעייה היא שלא ברור איך לחשב את $b!$ ללא שימוש ב־b פעולות כפל. בעייה נוספת היא שמספר הביטים ב־$b!$ עשוי להיות עצום. אבל זאת בעייה שקל לפתור כי אפשר פשוט לחשב את $b! \mod N$ שהוא מספר בן לכל היותר n ביטים.
על מנת להתגבר על הבעייה הזאת, נשתמש ברעיון שמבוסס על משפט 8.6.
נסמן $c=\lceil N^{1/4}\rceil$ (מעתה נשמיט את סימני ה־$\lceil\cdot\rceil$ כדי להקל על הקריאה) ונגדיר את הפולינום

$$f(x)=\prod_{i=1}^{c}(x+i).$$

f הוא פולינום מדרגה c, ונשים לב כי

$$\sqrt{N}!=\prod_{j=0}^{c-1}f(c\cdot j).$$

כלומר: על מנת למצוא מחלק לא טריוויאלי של N, ניתן לחשב את c הערכים $f(0),f(c),f(2c)\ldots,f((c-1)c)$ בעזרת משפט 8.6, מה שדורש $O(c\,\text{poly}(n))=O(N^{1/4}\,\text{poly}(n))$ פעולות. עבור כל אחד מהם נחשב את הערך $g_j:=\gcd(N,f(j\cdot c))$. אם N אינו ראשוני קיים j עבורו $g_j>1$. יהי j' המספר המינימלי עבורו זה מתקיים.
כיוון ש־

$$f(j'\cdot c)=(j'c+1)(j'c+2)\cdots(j'c+c),$$

אם j' הוא המינימלי כך ש־$g_{j'}>1$ אז לפחות אחד מבין c המספרים $\{jc+1,\ldots,jc+c\}$ הוא מחלק לא טריוויאלי של N, ולכן האלגוריתם יבדוק את כולם ויחזיר את המינימלי מביניהם שמחלק את N.
כפי שהערנו קודם, את הערכים $\{f(j\cdot c)\}_{0\le j\le c-1}$ יש לחשב מודולו N. הדבר דורש שימוש באלגוריתם ממשפט 8.6 מעל החוג $\mathbb{Z}_N$ ולא מעל $\mathbb{C}$, מה שבתורו דורש גירסה של FFT מעל $\mathbb{Z}_N$ אך ניתן לעשות זאת.
בעייה נוספת היא איך לחשב את מקדמי הפולינום f. מימושים נאיביים של הכפל ייתנו אלגוריתם שרץ בזמן $O(c^2)=O(N^{1/2})$. את בעייה זו ניתן לפתור באופן דומה לשלב החישוב המקדים בהוכחת משפט 8.6.
לכל $0\le k\le\lceil\log c\rceil-1$ ו־$0\le \ell<2^{\lceil\log c\rceil-k}$ נגדיר את $f_{k,\ell}$ באופן הבא:

$$f_{0,\ell}=(x+1+\ell),\quad 0\le\ell\le c-1$$
$$f_{k,\ell}=f_{k-1,2\ell}\cdot f_{k-1,2\ell+1}$$

כך נקבל את $f=f_{\lceil\log c\rceil}$ בעזרת $O(N^{1/4}\,\text{poly}(n))$ פעולות בלבד. □

8.4 אינטרפולציה

אינטרפולציה היא הבעייה ההפוכה לשערוך. בהינתן רשימת ערכים $(u_0, v_0, \dots, u_{n-1}, v_{n-1})$ כאשר $n = 2^k$ נרצה למצוא את הפולינום היחיד $f \in \mathbb{F}[x]$ מדרגה n כך ש־$f(u_i) = v_i$ לכל i.
מנוסחת לגראנז׳ (שאם לא הכרתם, ניתן לוודא ישירות),

$$f(x) = \sum_{i=0}^{n-1} v_i \cdot \prod_{j=0, j\neq i}^{n-1} \frac{x-u_j}{u_i-u_j}.$$

חישוב ישירות של f לפי נוסחה זו יתן אלגוריתם שרץ בזמן $O(n^2)$ (בערך). נראה אלגוריתם מהיר יותר.

משפט 8.10. קיים אלגוריתם לאינטרפולציה שמבצע $O(M(n)\log n)$ פעולות חישוב.

הוכחה: נשתמש בנוסחת לגראנז׳. נסמן $s_i = \prod_{j=0, j\neq i}^{n-1} \frac{1}{u_i-u_j}$ ונכתוב את נוסחת לגראנז׳ בצורה

$$f(x) = \sum_{i=0}^{n-1} v_i s_i \prod_{j=0, j\neq i}^{n-1} (x-u_j).$$

אם נסמן

$$P_{k,0} = \prod_{i=0}^{n-1} (x-u_i)$$

אז לכל $0 \le i \le n-1$,

$$\frac{1}{s_i} = P'_{k,0}(u_i)$$

כאשר $P'_{k,0}$ היא הנגזרת של $P_{k,0}$.
הסימון $P_{k,0}$ אינו מקרי. זהו אותו $P_{k,0}$ שהראינו איך לחשב ברקורסיה בהוכחת משפט 8.6.
שלב החישוב המקדים באלגוריתם אם כן יתבצע באופן הבא: תחילה, נחשב את כל הפולינום $P_{i,j}$ כמו בהוכחת משפט 8.6. לאחר מכן, נחשב את הנגזרת $P'_{k,0}$ בזמן $O(n)$. כאן, בעזרת האלגוריתם ממשפט 8.6 נשערך את $P'_{k,0}$ בנקודות $u_0, \dots, u_{n-1}$ על מנת לקבל את $\frac{1}{s_0}, \dots, \frac{1}{s_{n-1}}$.
בעזרת n פעולות חילוק נוספות נקבל את $s_0, \dots, s_{n-1}$ ואת $v'_i := v_i \cdot s_i$.
לאחר שלב החישוב המקדים, שדורש $O(M(n)\log n)$ פעולות, נרצה לחשב את הפולינום

$$f(x) = \sum_{i=0}^{n-1} v'_i \prod_{j=0, j\neq i}^{n-1} (x-u_j).$$

אם נגדיר

$$r_0(x) = \sum_{i=0}^{n/2-1} v'_i \prod_{j=0, j\neq i}^{n/2-1} (x-u_j), \quad r_1(x) = \sum_{i=n/2}^{n-1} v'_i \prod_{j=n/2, j\neq i}^{n-1} (x-u_j)$$

אז נקבל כי

$$f = r_0 P_{k-1,1} + r_1 P_{k-1,0} \tag{11}$$

כאשר את $P_{k-1,1}, P_{k-1,0}$ כבר חישבנו בשלב החישוב המקדים.
בעיית החישוב של r_0, r_1 דומה לבעיית החישוב של f אולם כעת מספר הנסכמים והגורמים במכפלה הוא $n/2$.
לכן, ניתן להפעיל אלגוריתם רקורסיבי לחישוב r_0, r_1. נשים לב שהאלגוריתם הרקורסיבי דורש שימוש בפולינומים $P_{i,j}$ שחישבנו בשלב החישוב המקדים, אך אינו זקוק לחשב אותם מחדש, וכן לא זקוק לחשב מחדש את הערכים v_i' שחישבנו בשלב החישוב המקדים.
לאחר חישוב r_0, r_1, ניתן להשתמש ב־(11) כדי לקבל את f.
נסמן $T(n)$ את סיבוכיות השלב הרקורסיבי. נקבל את נוסחת הרקורסיה $T(n) = 2T(n/2) + O(M(n))$ כלומר $T(n) = O(M(n)\log n)$ כנדרש. □

8.5 הרכבה מודולרית

נציג אלגוריתם לבעייה הבאה: בהינתן פולינומים $f, g, h \in \mathbb{F}[x]$ כך ש־$\deg(g), \deg(h) < \deg(f) = n$, נרצה לחשב את הפולינום $g(h(x)) \mod f$.
הבעייה עשויה להיראות קצת מוזרה, אך נעיר כי פתרון יעיל לבעייה זו היא חלק משמעותי באלגוריתמים לפירוק לגורמים של פולינומים. עם זאת, אנו מציגים אותה בעיקר מכיוון שניתן לקבל אלגוריתם יעיל עבורה בעזרת שילוב אלגנטי של האלגוריתמים משני החלקים בקורס.
מה האלגוריתם הנאיבי לבעייה? כזכור ראינו בתרגיל הבית שניתן בהינתן קלט x לשערך את הפולינום $g(x)$ בעזרת n כפלים ו־n חיבורים. שיטה זו נקראת **כלל הורנר**.
כאן אנחנו נדרשים לשערך את g בנקודה $h(x)$, כלומר כל פעולת חשבון היא פעולת חשבון על פולינומים מודולו f. לכן, יש צורך ב־$O(nM(n))$ פעולות חשבון. נראה אלגוריתם יעיל יותר שמשתמש בכפל מטריצות מהיר.

משפט 8.11. קיים אלגוריתם לבעיית ההרכבה המודולרית שמבצע לכל היותר $\sqrt{n}$ פעולות של כפל מטריצות בגודל $\sqrt{n} \times \sqrt{n}$, ובנוסף $O(\sqrt{n} \cdot (M(n) + n))$ פעולות כפל וחיבור.

אם $\omega = 2$ אז האלגוריתם מבצע בערך $O(n^{3/2})$ פעולות. בעזרת החסם העליון הידוע $\omega < 2.38$ אפשר עדיין לקבל אלגוריתם טוב יותר מהאלגוריתם הנאיבי.

הוכחה: האלגוריתם יפעל באופן הבא:

1. נגדיר $m = \lceil \sqrt{n} \rceil$. (מעתה והלאה נשמיט את סימני ה־$\lceil \cdot \rceil$).
   נסמן $g = \sum_{i=0}^{m} g_i(x) x^{mi}$ כאשר $g_0, \ldots, g_{m-1} \in \mathbb{F}[x]$ פולינומים ממעלה לכל היותר $m-1$.
2. לכל $0 \le i \le m$, חשב את $h^i \mod f$.
3. תהי $A \in \mathbb{F}^{m\times n}$ מטריצה שהשורה ה־i בה היא המקדמים של $h^i \mod f$ עבור $0 \le i \le m-1$.
   תהי $B \in \mathbb{F}^{m\times m}$ מטריצה שהשורה ה־i בה היא המקדמים של g_i, עבור $0 \le i \le m-1$.
   חשב את המטריצה $BA \in \mathbb{F}^{m\times n}$ בעזרת $n/m \le m$ כפלים של מטריצות $m \times m$.
4. יהי r_i הפולינום שמקדמיו הם השורה ה־i של המטריצה BA. חשב את

$$b = \sum_{i=0}^{m-1} r_i(x)(h(x)^m)^i \mod f$$

   בעזרת כלל הורנר, והחזר את b.

נטען כי האלגוריתם אכן מחזיר את $g(h(x)) \mod f$.
יהי $0 \le i \le m-1$ ונסמן $g_i = \sum_{j=0}^{m-1} g_{i,j} x^j$ כאשר $g_{i,j} \in \mathbb{F}$.
אזי $r_i = \sum_{j=0}^{m-1} g_{i,j} \cdot (h^j \mod f) = g_i(h) \mod f$, ו־

$$b = \sum_{i=0}^{m-1} r_i (h^m)^i \mod f = \sum_{i=0}^{m-1} g_i(h) h^{mi} \mod f = g(h) \mod f.$$

ננתח את סיבוכיות האלגוריתם. שלב 2 דורש m מכפלות מודולו f, ולכן דורש בסך הכל $O(M(n) \cdot \sqrt{n})$ פעולות.
שלב 3 דורש m מכפלות של מטריצות $m \times m$.
שלב 4 דורש m כפלים וחיבורים מודולו f, ולכן הסיבוכיות שלו דומה לזה של שלב 2. □

איור 1: כפל המטריצות המתבצע באלגוריתם מהוכחת משפט 8.11

$$\begin{pmatrix} - & g_0 & - \\ - & g_1 & - \\ & \vdots & \\ - & g_{m-1} & - \end{pmatrix} \cdot \begin{pmatrix} - & h^0 \mod f & - \\ - & h^1 \mod f & - \\ & \vdots & \\ - & h^{m-1} \mod f & - \end{pmatrix} = \begin{pmatrix} - & r_0 & - \\ - & r_1 & - \\ & \vdots & \\ - & r_{m-1} & - \end{pmatrix}$$

חלק 4: בדיקת ראשוניות

בפרק הבא בקורס נדון באלגוריתם **יעיל** ו**דטרמיניסטי** לבדיקת ראשוניות.
כלומר, ניתן אלגוריתם שבהינתן מספר N מכריע בעזרת $O(\log^C N)$ פעולות (עבור קבוע C כלשהו) האם N הוא מספר ראשוני. זכרו כי האלגוריתם הנאיבי רץ בזמן $O(\sqrt{N})$ ואילו אלגוריתם הפירוק לגורמים שהצגנו (שיכול להשתמש בפרט כאלגוריתם לבדיקת ראשוניות) רץ בזמן בערך $O(N^{1/4})$.
בעיית בדיקת הראשוניות אינה בעייה אלגברית שניתן לנסח במודל החישוב האלגברי, אך האלגוריתם שנציג עבורה משתמש ברעיונות אלגבריים שלמדנו בקורס.
בדיקת ראשוניות היא פעולה נפוצה ושימושית, ובספרות ובשוק קיימים אלגוריתמים נוספים לבדיקת ראשוניות. האלגוריתם המשמש בפועל במרבית השימושים הוא אלגוריתם אקראי, כלומר, אלגוריתם שמשתמש בביטים מקריים ויכול לטעות בהסתברות נמוכה (על פני האקראיות של האלגוריתם). עם זאת, ניתן באמצעים סטנדרטיים לוודא שהסתברות השגיאה כל כך נמוכה כך שהיעילות הרבה של האלגוריתם מפצָה על הסיכוי הקלוש לטעות.

9 מבחני ראשוניות

על מנת להציג את האלגוריתם נסקור תחילה את העקרונות שעומדים מאחוריו. בתחילת הקורס ראינו כי לכל n, $\mathbb{Z}_n$ הוא שדה אם ורק אם n הוא ראשוני. כלומר, מעבר להגדרתם היבשה, מספרים ראשוניים גוררים תכונות אלגבריות נחמדות שניתן אולי לנסות לבדוק בזמן יעיל. התכונה הראשונה אותה נוכיח נקראת לעיתים "משפט פרמה הקטן" (וזאת כדי להבדיל מהמשפט המפורסם שנודע בשם המשפט האחרון של פרמה, שהוא מעט יותר קשה להוכחה).

9.1 משפט פרמה

למה 9.1. יהי p ראשוני ויהי $a \in \mathbb{Z}_p$. אזי מתקיים $a^p = a \mod p$.

על מנת להוכיח את 9.1 נוכיח תחילה את הטענה הבאה.

טענה 9.2. יהי p ראשוני. אזי לכל $a, b \in \mathbb{Z}_p$ מתקיים $(a+b)^p = a^p + b^p$ כאשר כל החיבורים והכפלים מתקיימים ב־$\mathbb{Z}_p$.

הוכחה: באופן שקול, יש להוכיח כי לכל זוג מספרים טבעיים a, b,

$$(a+b)^p = a^p + b^p \mod p.$$

נפתח סוגריים בביטוי $(a+b)^p$ לפי נוסחת הבינום, ונקבל

$$(a+b)^p = \sum_{k=0}^{p} \binom{p}{k} a^k b^{p-k}.$$

את הסכום בצד ימין של המשוואה הראשונה נפריד לכמה חלקים. כאשר $k=0$ מקבלים את a^p וכאשר $k=p$ מקבלים את b^p.
נראה שאם $1 < k < p$ מתקיים $\binom{p}{k} = 0 \mod p$ כלומר $\binom{p}{k}$ מתחלק ב־p.

אכן,

$$\binom{p}{k} = \frac{p!}{k! \cdot (p-k)!}.$$

נשים לב ש־p מחלק את המונה אבל כיוון ש־p ראשוני ואין לו מחלקים לא טריוויאליים, p לא מחלק את המכנה, ולכן הוא מחלק את $\binom{p}{k}$. □

הוכחת למה 9.1: באינדוקציה על k. עבור $k = 0, 1$ הטענה טריוויאלית. עבור צעד האינדוקציה,

$$k^p = ((k-1)+1)^p = (k-1)^p + 1^p = k-1+1 = k \mod p$$

כאשר השוויון השני נובע מטענה 9.2 והשוויון השלישי נובע מהנחת האינדוקציה. □

האם למה 9.1, מספקת מבחן ראשוניות יעיל? תחילה יש לשים לב שבהינתן a ניתן לחשב את a^N בזמן $O(\log N)$ כפי שראינו בתרגיל הבית הראשון, ובפרט לבדוק האם $a^N = a \mod N$.
עם זאת, השוויון הזה עשוי להתקיים גם עבור N פריק. יתרה מכך, ישנם מספרים הידועים בשם מספרי קרמייקל שעבורם מתקיים השוויון $a^N = a \mod N$ לכל a שזר ל־N (כלומר, שאין להם גורם משותף גדול מ־1). מתברר אפילו שיש אינסוף מספרים כאלה. לכן, למה 9.1 אינה מספקת מבחן ראשוניות יעיל.

9.2 וריאציה פולינומית על מבחן פרמה

כזכור, השתמשנו בסימון $\mathbb{Z}_N[x]$ עבור קבוצת הפולינומים מעל $\mathbb{Z}_N$, והערנו כי זו קבוצה בעלת מבנה אלגברי שנקראת **חוג**. למעשה, גם $\mathbb{Z}_N$ הוא חוג לכל N (והוא שדה רק כש־N ראשוני), אך ניתן להגדיר פולינומים מעל חוג ולא רק מעל שדה.
נציג כעת קריטריון דומה ללמה 9.1 המתייחס לזהויות ב־$\mathbb{Z}_N[x]$ במקום ב־$\mathbb{Z}_N$.

למה 9.3. יהי N מספר טבעי ויהי a זר ל־N.
אזי $(x+a)^N = x^N + a$, כשוויון בחוג $\mathbb{Z}_N[x]$, אם ורק אם N ראשוני.

הוכחה: אם N ראשוני הטענה נובעת ישירות מטענה 9.2.
אם N פריק, יהי p גורם ראשוני של N. נסמן את p^k את החזקה הגבוהה ביותר של p שמחלקת את N. המקדם של x^p בפולינום $(x+a)^N$ הוא $\binom{N}{p}a^{N-p}$.
a זר ל־N ולכן רק צריך לוודא ש־$\binom{N}{p} \neq 0 \mod N$. אכן,

$$\binom{N}{p} = \frac{N!}{p!(N-p)!} = \frac{N(N-1)\cdots(N-p+1)}{p(p-1)\cdots 1}.$$

החזקה הגבוהה ביותר של p שמחלקת את המונה היא p^k, ו־p מחלק את המונה, לכן החזקה הגבוהה ביותר של p שמחלקת את $\binom{N}{p}$ היא p^{k-1}, כלומר p^k לא מחלק את $\binom{N}{p}$ ובפרט N לא מחלק את $\binom{N}{p}$. □

למה 9.3 מעלה את הרעיון הבא. בהינתן קלט N, נבחר, באקראי או בצורה מתוחכמת, $a < N$ כלשהו. נבדוק בעזרת אלגוריתם לחישוב gcd האם a זר ל־N. אם לא, הרי שמצאנו מחלק לא טריוויאלי של N ובפרט N אינו ראשוני. אך אם כן, נבדוק האם $(x+a)^N = x^N + a$ בחוג $\mathbb{Z}_N[x]$.
הבעייה היא שהשוויון $(x+a)^N = x^N + a$ הוא שוויון של פולינומים. לפולינום $(x+a)^N$ יש במקרה הגרוע N מקדמים ולכן נצטרך לשמור את כולם על מנת לחשב אותו כרשימת מקדמים, מה שגורר אלגוריתם שרץ לפחות בזמן לינארי ב־N במקום לוגריתמי ב־N.
האלגוריתם שנציג מתגבר על הקושי הזה ע״י חישוב זהויות פולינומיות מדרגה נמוכה יותר שניתן לוודא באופן ישיר. בפרקים הבאים ניתן את הפרטים המלאים.

10 ארגז כלים אלגברי

10.1 אריתמטיקה מודולרית של פולינומים

כזכור מפרק 8.1, בהינתן פולינומים $f, g \in \mathbb{F}[x]$ ניתן להגדיר באופן יחיד את הפולינום $f \mod g \in \mathbb{F}[x]$, ובנוסף מעלתו קטנה ממעלת g. לא ציינו זאת בפרק, אך עובדה זו וגם אלגוריתם החילוק נכונים גם עבור פולינומים מעל חוג R כל עוד g הוא פולינום **מתוקן** (monic), כלומר, המקדם המוביל שלו הוא 1.

הגדרה 10.1. יהי R חוג ויהי $h(x) \in R[x]$ פולינום מתוקן ממעלה d.
נסמן ב־$R[x]/(h(x))$ את קבוצת הפולינומים ממעלה לכל היותר $d-1$ מעל R עליה מוגדרות פעולות החיבור והכפל הבאות. לכל f, g נגדיר את החיבור שלהם ב־$R[x]/(h(x))$ להיות החיבור הרגיל כפולינומים, ואת הכפל ב־$R[x]/(h(x))$ להיות $f \cdot g \mod h$.

זוהי בדיוק ההגדרה האנלוגית בעולם הפולינומים להגדרת $\mathbb{Z}_N$ (כאשר הפולינום h ממלא את תפקיד N). בנוסף, בפרק 8.1 עבדנו מעל החוג $\mathbb{F}[x]/(x^k)$ עבור k מתאים (אם כי לא השתמשנו בסימון הזה).
האלגוריתמים המהירים שלמדנו בקורס לכפל וחילוק של פולינומים מאפשרים לבצע אריתמטיקה יעילה בחוג $R[x]/(h(x))$. עם זאת, באלגוריתם שלנו הסיטואציה אף תהיה פשוטה יותר כיוון שהאלגוריתם עצמו יבצע אך ורק חיבורים וכפלים מודולו הפולינום $x^r - 1$ (עבור r שייבחר בהמשך). נשים לב שלכל k,

$$x^k = x^{k \bmod r} \mod x^r - 1,$$

כיוון שאם נכתוב $k = m \cdot r + b$ אזי

$$x^k = x^{mr+b} = (x^r)^m \cdot x^b = 1 \cdot x^b \mod x^r - 1.$$

לכן, אריתמטיקה מודולו $x^r - 1$ היא פשוטה במיוחד מבחינה אלגוריתמית.
כעת כבר ניתן לחשוף כי האלגוריתם שנציג לבדיקת ראשוניות יבדוק את הזהות

$$(x+a)^N = x^N + a \mod x^r - 1$$

מעל $\mathbb{Z}_N[x]$ כאשר r יבחר להיות $O(\log^C N)$ עבור קבוע כלשהו C. כעת מספר המקדמים הוא פולינום ב־$\log N$ וניתן להשוות את הפולינומים ישירות. ברור שאם N ראשוני, כיוון שהשוויון מתקיים ב־$\mathbb{Z}_N[x]$, הוא גם יתקיים מודולו $x^r - 1$. מה שלא ברור עדיין הוא למה המבחן הזה ייכשל עבור מספרים פריקים.
לצורך כך, נציג עוד רקע מתמטי.

10.2 סדר כפלי

על מנת שהמבחן יצליח יש לבחור r מתאים, אם כי בשלב זה עדיין לא ברור מה התכונה שנזדקק לה.

הגדרה 10.2. **הסדר הכפלי** של N מודולו r, המסומן $\text{ord}_r(N)$, הוא המספר המינימלי k כך ש־$N^k = 1 \mod r$.

לא נוכיח זאת כעת אבל מסתבר ש־$\text{ord}_r(N)$ הוא סופי אם ורק אם r זר ל־N. בנוסף, אם הסדר סופי אז הוא בוודאי לכל היותר r.

למה 10.3. יהי N גדול דיו. אזי קיים $r \le 100 \log^5 N$ כך ש־$\text{ord}_r(N) \ge 5 \log^2 N$.

לצורך הוכחת למה 10.3, נוכיח תחילה עובדה בסיסית בתורת המספרים.
עבור k מספרים טבעיים $n_1,\dots,n_k$ **הכפולה המשותפת המינימלית** שלהם, המסומנת $\operatorname{lcm}(n_1,\dots,n_k)$ היא המספר המינימלי M כך ש־n_i מחלק את M לכל i.

למה 10.4. יהי m מספר אי־זוגי. אז $\operatorname{lcm}(1,2,\dots,m)\geq 2^{m-1}$.

הוכחה: נסמן $m=2n+1$. נביט באינטגרל

$$I:=\int_0^1 x^n(1-x)^n\,\mathrm{d}x.$$

כיוון ש־$x(1-x)\leq\frac{1}{4}$ בקטע $[0,1]$, $I\leq 2^{-2n}$.
נפתח את $(1-x)^n$ לפי נוסחת הבינום ונקבל

$$I=\int_0^1\sum_{k=0}^{n}(-1)^k\binom{n}{k}x^{n+k}\,\mathrm{d}x=\sum_{k=0}^{n}(-1)^k\binom{n}{k}\frac{1}{n+k+1} \tag{12}$$

נביע את צד ימין של המשוואה, שהוא מספר רציונלי, כשבר מצומצם M/L.
כיוון שכל המכנים במשוואה (12) קטנים מ־$2n+1$, $L\leq\operatorname{lcm}(1,\dots,2n+1)$.
בנוסף $M\geq 1$ כיוון שהאינטגרל גדול מאפס. לכן נקבל $L\geq 2^{2n}$. □

הוכחת למה 10.3: אם $\operatorname{ord}_r(N)=k$ אז r מחלק את N^k-1.
נסמן $T=100\log^5 N$, ויהי s המספר המינימלי שאינו מחלק את

$$M:=N^{\log T}\prod_{k=1}^{5\log^2 N}(N^k-1)<N^{50\log^4 N}=2^{50\log^5 N}.$$

מלמה 10.4, $\operatorname{lcm}(1,\dots,T)\geq 2^{T-1}>M$ כלומר קיים $1\leq r\leq T$ שאינו מחלק את M.
אם $\gcd(r,N)=1$ אז $\operatorname{ord}_r(N)>5\log^2 N$ וסיימנו. אחרת, נבחר את $s=r/\gcd(r,N^{\log T})$.
נשים לב ש־s במקרה זה קטן מ־T, גדול מ־1 (כיוון ש־r מחלק את $N^{\log T}$) וחייב להיות זר ל־N: כל חזקת ראשוני q^e שמופיעה בפירוק של r חייבת לקיים $e\leq\log T$ (כיוון ש־$r\leq T$) ולכן אם q מחלק את N, q^e מחלק את $N^{\log T}$ ולכן כל הגורמים המשותפים מתבטלים בחלוקה ב־$\gcd(r,N^{\log T})$. □

11 אלגוריתם AKS לבדיקת ראשוניות

כעת ניתן להציג את אלגוריתם AKS לבדיקת ראשוניות.
על קלט N, האלגוריתם יפעל באופן הבא:

1. בדוק האם $N=a^b$ עבור $a,b\geq 2$. אם כן, החזר "פריק".
2. מצא את ה־r הקטן ביותר כך ש־$\operatorname{ord}_r(N)\geq 5\log^2 N$.
3. אם $\gcd(a,N)\neq 1$ עבור $1\leq a\leq r$, החזר "פריק".

4. לכל $0 \leq a \leq r$, בדוק האם $(X+a)^N = x^N + a \mod x^r - 1$ מעל $\mathbb{Z}_N$. אם אחת מהבדיקות נכשלת, החזר "פריק".

5. אחרת, החזר "ראשוני".

כבר עכשיו ניתן לנתח את זמן הריצה ולהראות שהאלגוריתם רץ בזמן $O(\log^C N)$.
עבור שלב 1, נשים לב שאם $N = a^b$ אז בפרט $b \leq \log N$. לכל $2 \leq b \leq \log N$ ניתן בעזרת חיפוש בינארי לבדוק האם יש $2 \leq a \leq \sqrt{N}$ כך ש־$a^b = N$.
בשלב 2, למה 10.3 מבטיחה קיום של $r \leq 100\log^5 N$ כך ש־$\text{ord}_r(N)$ גדול דיו, וחסם תחתון על הסדר הכפלי ניתן לוודא ע"י חישוב חזקות של N. למה 10.3 נכונה עבור N גדול דיו כך שפורמלית ניתן להוסיף שלב 0 לאלגוריתם שבודק האם N קטן מקבוע אוניברסלי כלשהו, ואם כן, בודק ראשוניות בצורה נאיבית כלשהי. שלב כזה לא ישפיע על זמן הריצה של האלגוריתם.
שלב 3 דורש מספר חישובי gcd שניתן לבצע בזמן $O(\log^C N)$.
השלב העיקרי באלגוריתם הוא שלב 4. עם זאת, כיוון שראינו שניתן להניח כי $r \leq 100\log^5 N$, כל פעולות החשבונות על פולינומים והשוואות הפולינומים שמתבצעות בו הן עבור פולינומים ממעלה לכל היותר $100\log^5 N$ ולכן גם את השלב הזה ניתן לבצע בזמן הדרוש.
בנוסף, מלמה 9.3 נובע שעבור כל מספר ראשוני האלגוריתם יחזיר "ראשוני". נותר לוודא שלכל מספר פריק N האלגוריתם יחזיק "פריק". בהינתן שלב 1, ניתן אף להניח ש־N אינו חזקה שלמה של מספר טבעי אחר. להוכחה זו נקדיש את שארית הפרק, אך נצטרך לתת עוד רקע מתמטי הדרוש להוכחת הנכונות.

11.1 פירוק לגורמים של פולינומים

כפי שניתן לפרק מספרים לגורמים ראשוניים, כך ניתן לפרק פולינומים למכפלה של גורמים **אי־פריקים**.

הגדרה 11.1. יהי $\mathbb{F}$ שדה. פולינום $f \in \mathbb{F}[x]$ נקרא **אי־פריק** אם לא ניתן לכתוב את f כמכפלה $g \cdot h$ כך ש־$\deg(g), \deg(h) \geq 1$.
אם כן ניתן לכתוב את f כך, הוא נקרא **פריק**.

עובדה 11.2. יהי $\mathbb{F}$ שדה. כל פולינום $f \in \mathbb{F}[x]$ ניתן לכתיבה כמכפלה של גורמים אי־פריקים $f = g_1^{e_1} \cdot g_2^{e_2} \cdots g_k^{e_k}$. הפירוק הוא יחיד עד כדי סדר הגורמים ועד כדי מכפלות באיברי $\mathbb{F}$.

מתברר כי בניגוד לפירוק לגורמים של מספרים טבעיים, קיימים אלגוריתמים יעילים אשר בהינתן פולינום $f(x)$ מחשבים את הפירוק של $f(x)$ לגורמים אי־פריקים.
חשיבות נוספת של פולינומים אי־פריקים נובעת מהעובדה הבאה. בפרק 10.1 הצגנו את החוג $\mathbb{F}[x]/(h(x))$.

עובדה 11.3. יהי $h(x)$ פולינום אי־פריק. אזי $\mathbb{F}[x]/(h(x))$ הוא **שדה**.

שימו לב עד כמה עובדה זו אנלוגית למשפט 2.3 שהוכחנו בתחילת הקורס.
מסקנה נוספת של עובדה 11.3 היא קיומם של שדות ראשוניים מגודל של חזקות של מספרים ראשוניים. מסתבר כי לכל k קיים פולינום אי־פריק $h(x)$ ממעלה k מעל $\mathbb{Z}_p$ ואז $\mathbb{Z}_p/(h(x))$ הוא שדה סופי מגודל p^k. עוד מתברר כי אלו השדות הסופיים היחידים – כל שדה סופי הוא מגודל p^k עבור ראשוני כלשהו p ומספר טבעי k, וניתן לבנות אותו באופן הזה.

11.2 פולינומים ציקלוטומיים ושורשי יחידה פרימיטיביים

בפרק זה נשלים את הרקע האלגברי הדרוש להוכחת נכונות האלגוריתם. בפרק 7.3 התוודענו לשורשי היחידה המרוכבים. בפרק זה נחקור עוד קצת את המבנה שלהם ואת הפולינומים שהם מאפסים. תחילה, נישאר בעולם המספרים המרוכבים (בהמשך נעסוק בשורשי יחידה מעל שדות סופיים).

הגדרה 11.4. איבר $z \in \mathbb{C}$ ייקרא **שורש יחידה פרימיטיבי מסדר** n אם $z^n = 1$ ולכל $0 < i < n$, $z^n \neq 1$.

לדוגמה, $\omega = e^{2\pi i/n}$ הוא שורש יחידה פרימיטיבי.
עבור $n = 4$, שורשי היחידה מסדר n הם $\pm 1, \pm i$. מתוכם, ± 1 אינם פרימיטיביים.

עובדה 11.5. אם a מחלק את b אז $x^a - 1$ מחלק את $x^b - 1$.

הוכחה:

$$x^b - 1 = (x^a - 1)\cdot(1 + x^a + x^{2a} \cdots + x^{b-a}).$$ □

בפרט נובע שאם a מחלק את b, אז שורש יחידה z מסדר a הוא גם שורש יחידה מסדר b. בכיוון השני, נטען שכל שורש יחידה מסדר b הוא שורש פרימיטיבי מסדר a כך ש־a מחלק את b.
כדי לראות זאת, יהי z כך ש־$z^b = 1$ ונסמן ב־a את המספר החיובי המינימלי כך ש־$z^a = 1$. נכתוב $b = qa + m$ כך ש־$m < a$. אזי

$$1 = z^b = z^{qk+m} = (z^a)^q \cdot z^m = z^m.$$

אם $m > 0$ אז קיבלנו סתירה לכך ש־a החיובי המינימלי כך ש־$z^a = 1$. מצד שני אם $m = 0$ אז a מחלק את b.

הגדרה 11.6. יהי $n \in \mathbb{N}$, ותהי P קבוצת שורשי היחידה הפרימיטיביים מסדר n. הפולינום

$$\Phi_n(x) := \prod_{\alpha \in P}(x - \alpha)$$

נקרא **הפולינום הציקלוטומי מסדר** n.

מעלתו של הפולינום Φ_n היא מספר המספרים $d < n$ שזרים ל־n. נהוג לסמן מספר זה $\varphi(n)$.

טענה 11.7. לכל n,

$$x^n - 1 = \prod_{d|n} \Phi_d$$

הוכחה: כל שורש יחידה מסדר n הוא שורש יחידה מסדר d כלשהו עבור d שמחלק את n, ולהיפך, לכן שורשי הפולינומים בשני הצדדים הם שווים, והפולינומים שווים עד כדי קבוע. בנוסף, שניהם פולינומים מתוקנים. □

נסיים בעובדות שאת חלקן נוכיח בשיעורי הבית.

עובדה 11.8. לכל n, הפולינום Φ_n הוא פולינום במקדמים שלמים ואי־פריק מעל $\mathbb{Q}$.

מעל שדה סופי $\mathbb{Z}_p$ הדברים מעט שונים.
ניתן באופן אנלוגי להגדיר שורשי יחידה ושורשי יחידה פרימיטיביים. השדה עצמו עשוי לא להכיל n שורשי יחידה שונים מסדר n אך מתברר כי ניתן להרחיב אותו על מנת לקבל שדה שמכיל n שורשי יחידה שונים.
בנוסף, כיוון ש־Φ_n פולינום במקדמים שלמים, ניתן להגדיר לכל p את הפולינום $\Phi_n \mod p$, והנוסחה מטענה 11.7 עדיין מתקיימת.
עם זאת, מעל $\mathbb{Z}_p$ הפולינום Φ_n אינו בהכרח אי־פריק. ניתן להוכיח שהפולינום Φ_d מתפרק מעל $\mathbb{Z}_p$ למכפלה של $\varphi(d)/\mathrm{ord}_d(p)$ גורמים אי־פריקים, כל אחד ממעלה $\mathrm{ord}_d(p)$.

11.3 הוכחת נכונות האלגוריתם

כזכור מהדיון בסוף פרק 11, יש להוכיח את המשפט הבא.

משפט 11.9. יהי N פריק כך ש־N אינו חזקה שלמה של מספר טבעי. אז אלגוריתם AKS על הקלט N מחזיר "פריק".

בפרק זה נוכיח את המשפט. יהי N פריק ויהי p ראשוני שמחלק אותו. מההנחה על N, הוא אינו חזקה של p. בנוסף ניתן להניח ש־$p > r$ בשל הבדיקות בשלב 3, ומשלב 4 לכל $0 \le a \le r$ מתקיים

$$(x+a)^N = x^N + a \mod (x^r - 1)$$

מעל $\mathbb{Z}_N$. כיוון ש־p מחלק את N, השוויון הזה נכון גם נכון מעל $\mathbb{Z}_p$.
בנוסף, כיוון ש־p ראשוני, מלמה 9.3 מתקיים גם

$$(x+a)^p = x^p + a \mod (x^r - 1)$$

מעל $\mathbb{Z}_p$. כלומר, שני השוויונות האחרונים מתקיימים בחוג $R := \mathbb{Z}_p/(x^r-1)$.
השוויונות הללו נותן מוטיבציה להגדרה הבאה.

הגדרה 11.10. נאמר שפולינום $f(x)$ ומספר טבעי m **מתחלפים** אם $f(x^m) = f(x)^m$ בחוג $\mathbb{Z}_p/(x^r-1)$.

רעיון ההוכחה יהיה להראות שמצד אחד, מספר הפולינומים f שמתחלפים עם n הוא גדול. עובדה זו תנבע מכך שמההנחה, $(x+a)$ מתחלף עם N לכל $0 \le a \le r$, ומכמה תכונות סגירות של קבוצת הפולינומים המתחלפים שנוכיח.
מצד שני, נראה שאם n אינו חזקה של p, אז מספר הפולינומים המתחלפים הוא קטן. שילוב כל העובדות האלה יוביל לסתירה, כלומר לא ייתכן ש־N פריק שאינו חזקה של ראשוני יעבור בהצלחה את כל הבדיקות בשלב 4 של האלגוריתם.

למה 11.11. אם f מתחלף עם m ועם k אז f מתחלף עם mk.

הוכחה: מההנחה, $f(x)^m = f(x^m)$ בחוג $R := \mathbb{Z}_p/(x^r-1)$.
אם נציב בשוויון את x^k במקום x נקבל

$$f(x^k)^m = f(x^{km})$$

בחוג $\mathbb{Z}_p[x]/(x^{kr}-1)$. מעובדה 11.5, $x^r - 1$ מחלק את $x^{kr}-1$ ולכן השוויון האחרון נכון גם בחוג R. בנוסף, כיוון ש־f מתחלף עם k, $f(x^k) = f(x)^k$ ונקבל

$$f(x)^{km} = f(x^{km}).$$

בחוג R. □

תכונות סגירות נוספת היא התכונה הבאה.

למה 11.12. אם f, g מתחלפים עם m אז $f \cdot g$ מתחלף עם m.

הוכחה:

$$f \cdot g(x^m) = f(x^m) \cdot g(x^m) = (f(x))^m \cdot (g(x))^m = (f(x) \cdot g(x))^m = ((f \cdot g)(x))^m.$$ □

כזכור, הפולינומים $(x+a)$ עבור $0 \le a \le r$ מתחלפים עם N ו־p.
למות 11.11 ו־11.12 גוררות שכל פולינום בקבוצה $P = \left\{\prod_{a=1}^{r}(x+a)^{e_a} : e_a \ge 0\right\}$ מתחלף עם כל מספר בקבוצה $I = \left\{N^i p^j : i, j \ge 0\right\}$.
כעת, נגדיר את הקבוצה $I_r = \{a \mod r : a \in I\}$.

אבחנה 11.13. נסמן $t = |I_r|$. אז $t \ge 5\log^2 N$.

הוכחה: נובע מכך שהסדר של N מודולו r הוא לפחות $5\log^2 N$. בפרט החזקות N^k ל־$k < 5\log^2 n$ כולן שונות מודלו r. □

בנוסף, יהי Φ_r הפולינום הציקלוטומי מסדר r מעל $\mathbb{Z}_p$. כזכור, Φ_r מחלק את $x^r - 1$ ומתפרק למכפלות של גורמים אי־פריקים ממעלה $\text{ord}_r(p)$. כיוון ש־$\text{ord}_r(N) > 1$, ניתן להניח גם ש־$\text{ord}_r(p) > 1$: אם $\text{ord}_r(p)$ היה שווה ל־1 לכל הגורמים הראשוניים של N, זה היה נכון גם ל־N, ולכן ניתן לבחור את p כך שזה יתקיים.
יהי $h(x)$ אחד מהגורמים האי־פריקים של Φ_r, ונביט ב**שדה** $\mathbb{Z}_p/(h(x))$ (היזכרו בעובדה 11.3). נסמן את השדה הזה $\mathbb{F}$ ונגדיר

$$P_h = \left\{f \bmod h(x) : f \in P\right\} \subseteq \mathbb{F}.$$

הקבוצה P_h תהיה הקבוצה שננסה להעריך את הגודל שלה. נתחיל בחסם תחתון על הגודל של P_h. האתגר כאן הוא שאמנם P מכילה פולינומים רבים אבל ב־P_h אנחנו מסתכלים על הפולינומים הללו מודולו h ולא ברור מראש כמה "התנגשויות" הפעולה הזאת יוצרת.
איברי $\mathbb{F}$ הם פולינומים ממעלה לכל היותר

$$\deg(h(x)) - 1 = \text{ord}_r(p) - 1 > 0.$$

נוכיח תחילה את האבחנה הבאה.

אבחנה 11.14. בשדה $\mathbb{F} = \mathbb{Z}_p/(h(x))$, האיבר x הוא שורש יחידה פרימיטיבי מסדר r.

הוכחה: כיוון ש־$h(x)$ הוא גורם אי־פריק של Φ_r שמחלק את $x^r - 1$, מהעובדה ש־$x^r = 1 \mod (x^r - 1)$ נובע ש־$x^r = 1 \mod h(x)$.
בנוסף, לכל $0 < k < r$, אם $x^k = 1 \mod h(x)$ אז הפולינום $x^k - 1$ מחלק את $h(x)$ ולכן מחלק את Φ_r. זו סתירה לכך ש־Φ_r הוא הפולינום הציקלוטומי מסדר r. □

כעת נוכל להוכיח את החסם התחתון הבא על $|P_h|$.

למה 11.15. $|P_h| \ge \binom{t+r}{t-1}$.

הוכחה: תחילה נשים לב ש־$p > r$, מההנחה על p שנבע משלב 3 של האלגוריתם.
לכן, הפולינומים $x, x+1, \ldots, x+r$ כולם שונים ב־$\mathbb{Z}_p$ ולכן גם ב־$\mathbb{F}$.
כעת נראה שכל שני פולינומים שונים ב־P ממעלה קטנה מ־t גם יהיו שונים ב־P_h.
יהיו $f, g \in P$ ממעלה קטנה מ־t. נניח כי $f = g$ מעל $\mathbb{F}$ ונראה כי $f = g$ כפולינומים ב־$\mathbb{Z}_p[x]$.
יהי $m \in I$. מתקיים גם $f(x)^m = g(x)^m$ מעל $\mathbb{F}$, ולכן, כיוון ש־f, g מתחלפים עם m, ו־$h(x)$ מחלק את $x^r - 1$, $f(x^m) = g(x^m)$ בשדה $\mathbb{F}$.
כעת, אם נגדיר את הפולינום $Q(Y) = f(Y) - g(Y) \in \mathbb{Z}_p[Y]$, נקבל שלכל $m \in I$ מתקיים $Q(x^m) = 0$ בשדה $\mathbb{F}$.

כמה שורשים שונים מתקבלים כך? כיוון ש־x שורש יחידה פרימיטיבי מסדר r, האיברים $\{x^m : m \in I_r\}$ כולם שונים, ולכן יש לכל הפחות $t = |I_r|$ שורשים ל־Q. מצד שני, Q הוא סכום של שני פולינומים ממעלה קטנה מ־t, ולכן מעלתו קטנה מ־t. המסקנה היא ש־Q פולינום האפס ו־$f = g$ כפולינומים ב־$\mathbb{Z}_p[x]$.
לפיכך, כל הפולינומים מהצורה

$$\{x^{e_0}(x+1)^{e_1}\cdots(x+r)^{e_r} : e_i \geq 0, e_0+e_1+\cdots e_r < t\}$$

הם שונים ב־P_h. מספר הפולינומים הללו הוא מספר הפתרונות במספרים טבעיים למשוואה $e_0+\cdots+e_r \leq t-1$, שהוא $\binom{t+r}{t-1}$. □

השלב האחרון בהוכחה יהיה לתת חסם עליון על הגודל של P_h.

למה 11.16. $|P_h| \leq N^{2\sqrt{t}}$.

הוכחה: נביט בתת הקבוצה הבאה של I:

$$\left\{N^i p^j : 0 \leq i, j \leq \sqrt{t}\right\}.$$

כיוון ש־N אינו חזקה של p, מספר האיברים בקבוצה הוא $(1+\lfloor\sqrt{t}\rfloor)^2$, שזה מספר גדול ממש מ־$t$. מצד שני, כיוון ש־$t = |I_r|$, נובע שיש $m_1 < m_2 \in I$ כך ש־$m_1 = m_2 \mod r$. כלומר,

$$x^{m_1} = x^{m_2} \mod (x^r - 1).$$

יהא $f(x) \in P$. אז כיוון ש־f מתחלף עם m_1, m_2, ב־$\mathbb{Z}_p$ מתקיים השוויון

$$(f(x))^{m_1} = f(x^{m_1}) = f(x^{m_2}) = (f(x)^{m_2}) \mod (x^r - 1).$$

בפרט מתקיים השוויון $f(x)^{m_1} = f(x)^{m_2}$ בשדה $\mathbb{F}$. כלומר, $f(x) \mod h(x)$, שהוא איבר ב־P_h, הוא שורש של הפולינום $Q_1(Y) = Y^{m_2} - Y^{m_1}$ מעל $\mathbb{F}$.
כיוון שזה נכון לכל איברי P_h, לפולינום Q_1 יש לפחות $|P_h|$ שורשים שונים ב־F. מצד שני, מעלתו של Q_1 היא $m_2 \leq N^{\sqrt{t}} \cdot p^{\sqrt{t}} \leq N^{2\sqrt{t}}$ ועל כן זה מספר השורשים המקסימלי שלו. □

כעת אנחנו מוכנים להוכחת משפט 11.9.

הוכחת משפט 11.9: נניח ש־N אינו חזקת ראשוני אך עובר את כל הבדיקות בשלב 4 של האלגוריתם. כל מה שנותר לעשות זה להראות שעבור הבחירה של t, החסמים התחתון של למה 11.15 והחסם העליון של למה 11.16 מובילים לסתירה.
מלמה 11.15 נובע ש־$|P_h| \geq \binom{r+t}{t-1}$.
ניזכר כי $t \geq 5\log^2 N$ ו־$t \leq r$, ולכן

$$\binom{r+t}{t-1} \geq \binom{2t}{t-1} \geq \frac{(2t)(2t-1)\cdots(t+2)}{(t-1)(t-2)\cdots 1} \geq 2^{t-1}$$

מצד שני, מלמה 11.16 $|P_h| \leq N^{2\sqrt{t}} = 2^{2\sqrt{t}\log N}$, כלומר $t-1 \leq \sqrt{t}\log N$ וזו סתירה לכך ש־$t \geq 5\log^2 N$. □

תודות

תודתי נתונה לסטודנטים בקורס "חישוב אלגברי ואלגוריתמים אלגבריים" שניתן באוניברסיטת רייכמן בסמסטר חורף 2021-2022, ובפרט לעידו ידלין ומתן שסקין, על תשומת לבם הרבה והתיקונים ששלחו לשגיאות שנפלו בגרסאות קודמות של סיכומי ההרצאות.