

התמרת פורייה מהירה (FFT) וכפל מהיר של פולינומים

1. חזרה על פולינומים: פולינום מדרגה קטנה מ n מעל שדה F (במקרה שלנו: המספרים הממשיים או המרוכבים):

$$A(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}$$

לכל ערך $x_0 \in F$ הפולינום A מתאים ערך $y_0 = A(x_0)$. ניתן לחשב את $A(x_0)$ בזמן ליניארי

$$A(x_0) = a_0 + x_0(a_1 + x_0(a_2 + \dots + x_0(a_{n-2} + x_0 a_{n-1}) \dots)) \quad \text{Horner} : \text{כלל}$$

2. ייצוגים של פולינומים:

$$A(x) = \sum_{j=0}^{n-1} a_j x^j \quad \text{2.1. ייצוג סטנדרטי ע"י וקטור המקדמים:}$$

$$a = (a_0, a_1, a_2, \dots, a_{n-1})$$

2.2. ייצוג ע"י ערכי הפולינום בנקודות שונות.

משפט יחידות האינטרפולציה הפולינומית: לכל n זוגות של ערכים

$$(x_0, y_0), (x_1, y_1), (x_2, y_2), \dots, (x_{n-1}, y_{n-1})$$

כך ש- $x_i \neq x_j$ לכל $i \neq j$, קיים פולינום יחיד

$$A(x) \text{ , } n \text{ כך ש: } A(x_i) = y_i \text{ לכל } i = 0, 1, \dots, n-1.$$

יחידות: נובעת מכך שלפולינום מדרגה קטנה מ n יש פחות מ n שרשים.

קיום: נוסחת לגרנז' להלן מחשבת את פולינום האינטרפולציה (שדרגתו קטנה מ n) בסבוכיות

$$: \Theta(n^2)$$

$$A(x) = \sum_{k=0}^{n-1} y_k \frac{\prod_{j \neq k} (x - x_j)}{\prod_{j \neq k} (x_k - x_j)}$$

הערה: כשהשדה F אין סופי יש אינסוף אפשרויות לבחור n זוגות ערכים לייצוג פולינום נתון.

יתרון של ייצוג זה: אם נתונים שני פולינומים $A(x), B(x)$, ע"י ערכיהם באותן נקודות, אז

ייצוג ע"י ערכים של פולינום המכפלה $C(x) = A(x) \cdot B(x)$ באותן נקודות ניתן לחישוב בזמן

$$C(x_i) = A(x_i) \cdot B(x_i) \text{ המכפלות } n \text{ ביצוע } n \text{ הנקודות ע"י}$$

3. מכפלת פולינומים והגדרת הבעיה:

נתונים 2 פולינומים מדרגה קטנה מ n מעל שדה F , מיוצגים על ידי :

$$A(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}$$

$$B(x) = b_0 + b_1x + b_2x^2 + \dots + b_{n-1}x^{n-1}$$

פולינום המכפלה $C(x) = A(x) \cdot B(x)$ הוא פולינום מדרגה קטנה מ $2n-1$ המוגדר על ידי:

$$c_j = \sum_{k=0}^j a_k b_{j-k} \quad \text{כאשר } C(x) = c_0 + c_1x + c_2x^2 + \dots + c_{2n-2}x^{2n-2}$$

דוגמה: כאשר $A(x) = 1 - 2x + x^4$, $B(x) = 2 - x + x^2$ אז למשל:

$$c_4 = a_4b_0 + a_3b_1 + a_2b_2 + a_1b_3 + a_0b_4 = 2 + 0 + 0 + 0 + 0 = 2$$

הוקטור $c = (c_0, \dots, c_{2n-2})$ הוא הקונבולוציה של $a = (a_0, \dots, a_{n-1})$ ו $b = (b_0, \dots, b_{n-1})$. מסמנים

$$c = a \otimes b$$

זמן חישוב פולינום המכפלה (או הקונבולוציה) באופן ישיר: $\Theta(n^2)$.

מטרתנו לבצע את המכפלה בסיבוכיות $\Theta(n \log n)$. לצורך זה נשתמש בעובדה שכאשר

מייצגים את הפולינומים A ו B באמצעות ערכיהם ב N - נקודות, ניתן להגיע ל"יצוג באמצעות

ערכים" של C בזמן $O(N)$ ע"י $C(x_i) = A(x_i) \cdot B(x_i)$ עבור $i = 0, \dots, N-1$ (שימו לב שצריך

להתקיים $N \geq 2n-1$ כדי להבטיח יצוג יחיד של C). מה שדרוש הוא מעבר מ"יצוג על ידי

מקדמים" ל"יצוג ע"י ערכים" ובחזרה. לצורך זה משתמשים בהתמרת פורייה המהירה

(FFT), המשתמשת ביצוג הפולינומים באמצעות ערכיהם על שרשי היחידה.

4. תזכורת מספרים מרוכבים ותכונות של שרשי היחידה:

מספר מרוכב ניתן ליצוג $z = |z|(\cos \theta + i \sin \theta)$, כאשר $i = \sqrt{-1}$.

$$\text{De Moivre: } (\cos \theta + i \sin \theta)^n = \cos n\theta + i \sin n\theta$$

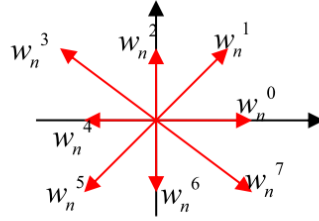
מספר מרוכב w הוא שורש יחידה מסדר n אם $w^n = 1$. מנוסחת De Moivre נובע שיש

בדיוק n שורשי יחידה מרוכבים מסדר n , והם נתונים ע"י $e^{i \frac{2\pi k}{n}}$, $k = 0, 1, 2, \dots, n-1$.

אם בנוסף $w^k \neq 1$ עבור $0 < k < n$, הוא שורש יחידה פרימיטיבי מסדר n . נסמן את

שורש היחידה הפרימיטיבי $e^{i \frac{2\pi}{n}}$ ב w_n . אז שורשי היחידה הם: $w_n^0, w_n^1, w_n^2, \dots, w_n^{n-1}$.

הציור להלן מתאר את שרשי היחידה עבור $n=8$.



5. הרעיון הכללי של האלגוריתם – שימוש בהתמרת פורייה המהירה FFT:

5.1. נייצג את הפולינומים A ו B ע"י ערכיהם ב $2n$ שורשי היחידה מסדר $2n$. המעבר למיצוג ע"י מקדמים ליצוג זה נקרא התמרת פוריה הבדידה (DFT או DFT_{2n}). נראה איך לממשו ע"י אלגוריתם FFT בזמן $\Theta(n \log n)$. כפי שנראה המימוש היעיל של FFT דורש ש n יהיה חזקה של 2.

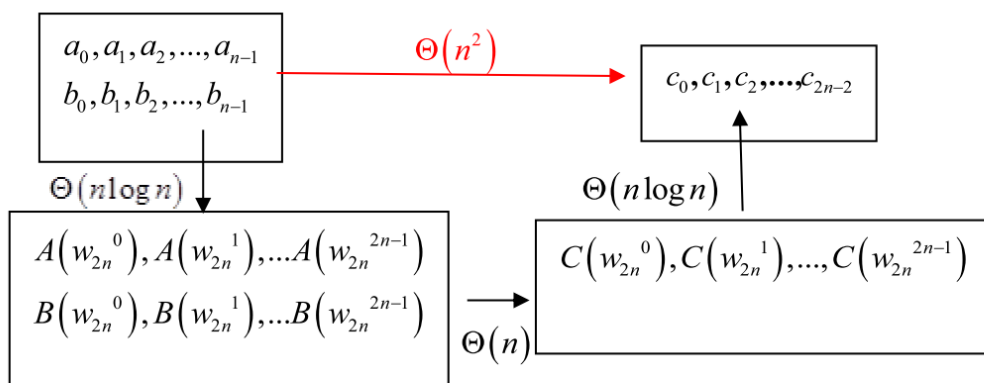
5.2. נכפיל את שני הפולינומים A ו B בייצוג הנ"ל ונקבל ייצוג של המכפלה C ע"י ערכים ב $2n$ שורשי היחידה מסדר $2n$. סבוכיות $\Theta(n)$.

5.3. נעבור מייצוג של C על ידי ערכיו ליצוג ע"י וקטור מקדמיו. המעבר נקרא התמרת פוריה ההפוכה (DFT^{-1}) וממומש ע"י FFT (עם שינויים קלים) בזמן $\Theta(n \log n)$ (במקום בזמן ריבועי ע"י נוסחת לגרנז').

שלושת השלבים האלו הם "משפט הקונבולוציה": עבור 2 וקטורים a, b באורך n מתקיים:

$$a \otimes b = DFT_{2n}^{-1} (DFT_{2n}(a) \cdot DFT_{2n}(b))$$

ניתן לתארם בסכימה הבאה:



6. FFT (Fast Fourier Transform) (התמרת פוריה המהירה):

נתון פולינום $A(x)$ שדרגתו קטנה מ n וצריך לחשב את ערכיו ב n שורשי היחידה מסדר n :

כלומר צריך לחשב את הערכים $y_k = A(w_n^k) = \sum_{j=0}^{n-1} a_j \cdot (w_n^k)^j$ עבור $k = 0, 1, 2, \dots, n-1$

הוקטור $y = (y_0, y_1, \dots, y_{n-1})$ הוא התמרת פוריה הבדידה (Discrete Fourier Transform) של $a = (a_0, a_1, \dots, a_{n-1})$ המסומנת: $y = DFT(a)$.

נניח ש n חזקה של 2 (אם יש צורך נוסיף אפסים מובילים מימין). נחלק את $A(x)$ לחזקות זוגיות ואי זוגיות:

$$A(x) = (a_0 + a_2x^2 + a_4x^4 + \dots + a_{n-2}x^{n-2}) + x(a_1 + a_3x^2 + a_5x^4 + \dots + a_{n-1}x^{n-2})$$

באמצעות חלוקה זו נגדיר שני פולינומים מדרגה קטנה מ $\frac{n}{2}$:

$$A_{\text{even}}(z) = a_0 + a_2z + a_4z^2 + a_6z^3 + \dots + a_{n-2}z^{\frac{n}{2}-1}$$

$$A_{\text{odd}}(z) = a_1 + a_3z + a_5z^2 + a_7z^3 + \dots + a_{n-1}z^{\frac{n}{2}-1}$$

נציב $z = x^2$ ונקבל: $A(x) = A_{\text{even}}(x^2) + x \cdot A_{\text{odd}}(x^2)$. נסמן שוויון זה ב (*).

כדי לחשב את $A(w_n^k)$ עבור $k=0, \dots, n-1$, ניתן להשתמש ב (*) ובערכי A_{even} ו A_{odd} בנקודות $\{w_n^{2k} \mid k = 0, 1, \dots, \frac{n}{2}-1\}$. מאחר ו n זוגי, הרי קבוצה זו היא בדיוק קבוצת שרשי היחידה מסדר $\frac{n}{2}$: $\{w_n^{2k} \mid k = 0, 1, \dots, \frac{n}{2}-1\}$. לכן ניתן לחשב את ערכי הפולינום A ב $\frac{n}{2}$ שרשי היחידה מסדר n באמצעות הערכים של כל אחד מהפולינומים A_{even} ו A_{odd} ב $\frac{n}{2}$ שרשי היחידה מסדר $\frac{n}{2}$, שאותם ניתן למצוא על ידי פתרון שתי בעיות FFT , כל אחת מהן בגודל $\frac{n}{2}$. זה מכתוב אלגוריתם הפותר את הבעיה עבור קלט בגודל n על ידי פיתרון רקורסיבי של 2 בעיות בגודל $\frac{n}{2}$. להלן האלגוריתם:

אלגוריתם RECURSIVE_FFT

קלט: $a = (a_0, a_1, \dots, a_{n-1})$ - סדרת מקדמי הפולינום $A(x)$, n חזקה של 2. §2.

פלט: $y = (y_0, y_1, \dots, y_{n-1})$, כך ש $y_k = A(w_n^k)$, כלומר $y = DFT_n(a)$.

§ כאמור, כל אחד ממקדמי הפולינום יכול להיות 0.

אם $n=1$ החזר $y = (a_0)$, אחרת:

$$\text{קריאה לאלגוריתם עבור שני קלטים} \begin{cases} y^{[0]} = (y_0^{[0]}, \dots, y_{\frac{n}{2}-1}^{[0]}) \leftarrow \text{RECURSIVE_FFT}(a_0, a_2, \dots, a_{n-2}) \\ y^{[1]} = (y_0^{[1]}, \dots, y_{\frac{n}{2}-1}^{[1]}) \leftarrow \text{RECURSIVE_FFT}(a_1, a_3, \dots, a_{n-1}) \end{cases}$$

בגודל $\frac{n}{2}$.

$$\text{עבור } k=0 \text{ עד } k = \frac{n}{2} - 1 \text{ בצע: } y_k = y_k^{[0]} + w_n^k y_k^{[1]}; y_{\frac{n}{2}+k} = y_k^{[0]} - w_n^k y_k^{[1]}$$

[השתמשנו בעובדה ששורש יחידה פרימיטיבי מסדר n מקיים $w_n^{\frac{n}{2}+k} = -w_n^k$].

החזר $y = (y_0, y_1, \dots, y_{n-1})$

$$\begin{cases} T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) \\ T(1) = O(1) \end{cases} \Rightarrow T(n) = \Theta(n \log n) \text{ סיבוכיות}$$

דוגמת ריצה : ראו במצגת.

תרגיל 1

תרגיל

יהיו $A, B \subseteq \{1, 2, \dots, n\}$ נגדיר

$$A + B = \{a + b : a \in A, b \in B\}$$

הצע אלגוריתם המחשב את $A + B$ ב- $O(n \log n)$.

ראשית נבחין כי קל לחשב את $A+B$ בזמן $O(n^2)$ פשוט על ידי מעבר על כל הזוגות האפשריים. קיומו של אלגוריתם נאיבי הרץ בזמן שהוא פי $O(n/\log n)$ מהנדרש הוא סממן היכר לבעיות הנפתרות באמצעות FFT.

נגדיר את הפולינומים

$$a(x) = \sum_{i=1}^n a_i x^i$$

$$b(x) = \sum_{i=1}^n b_i x^i$$

כאשר

$$a_i = \begin{cases} 1 & i \in A \\ 0 & i \notin A \end{cases}$$

-1

$$b_i = \begin{cases} 1 & i \in B \\ 0 & i \notin B \end{cases}$$

עבור כל $1 \leq k \leq 2n$, המקדם של x^k בפולינום ab , שנסמנו ב- c_k הוא

$$c_k = \sum_{i=1}^{2n} a_i b_{k-i}$$

אם $c_k \geq 1$ אם"ם קיים i כך ש- $i \in A$ ו- $k-i \in B$. כלומר, $c_k \geq 1$ אם"ם $k \in A+B$. מכאן נובע האלגוריתם הבא לחישוב $A+B$. (יתרה מכך, c_k מחשב את מספר הזוגות $(a,b) \in A \times B$ כך ש- $a+b=k$).

אלגוריתם

קלט: קבוצות A, B ומספר n כך ש- $A, B \subseteq \{1, 2, \dots, n\}$.

פלט: הקבוצה $A+B$.

1. חשב את הפולינומים $a(x), b(x)$ כפי שתוארו לעיל.

2. חשב את הפולינום $(ab)(x)$.

3. החזר $\{i : c_k \geq 1\}$.

נכונות

נכונות האלגוריתם נובעת מהדיון לעיל.

סיבוכיות

שלב 2 הוא השלב הכבד באלגוריתם וזמן הריצה שלו הוא $O(n \log n)$.

תרגיל 2

תרגיל

תהא $s = s_0 s_1 \cdots s_{n-1}$ מחרוזת בינארית. נאמר כי s היא בעלת מחזור t אם לכל $i \equiv_j$ מתקיים $s_i = s_j$. המחזור של s המסומן ב- $T(s)$ יוגדר להיות המספר המינימאלי t כך ש- s היא בעלת מחזור t . הצע אלגוריתם אשר בהינתן מחרוזת s מאורך n מחשב את $T(s)$ בזמן $O(n \log n)$.

פתרון

ראשית נבחין כי ישנו האלגוריתם הנאיבי הרץ בזמן $O(n^2)$: האלגוריתם עובר על כל n המחזורים האפשריים ($1 \leq T(s) \leq n$) מהקטן לגדול ובודק כל אחד מהם ב- $O(n)$.

נגדיר את הפולינומים

$$p(x) = (-1)^{s_0} + (-1)^{s_1} x + (-1)^{s_2} x^2 + \cdots + (-1)^{s_{n-1}} x^{n-1}$$
$$q(x) = (-1)^{s_{n-1}} + (-1)^{s_{n-2}} x + \cdots + (-1)^{s_0} x^{n-1}$$

עבור $1 \leq k \leq n-1$ המקדם של x^{n-k-1} בפולינום $pq(x)$, אותו נסמן ב- c_k הוא

$$\begin{aligned} \sum_{i=0}^{n-k-1} p_i q_{n-k-1-i} &= \sum_{i=0}^{n-k-1} (-1)^{s_i} (-1)^{s_{n-1-(n-k-1-i)}} \\ &= \sum_{i=0}^{n-k-1} (-1)^{s_i} (-1)^{s_{k+i}} \\ &= \sum_{i=0}^{n-k-1} (-1)^{s_i + s_{k+i}} \\ &= \left| \{i : s_i = s_{k+i} \wedge 0 \leq i \leq n-k-1\} \right| - \left| \{i : s_i \neq s_{k+i} \wedge 0 \leq i \leq n-k-1\} \right| \\ &= (n-k) - 2 \cdot \left| \{i : s_i \neq s_{k+i} \wedge 0 \leq i \leq n-k-1\} \right| \end{aligned}$$

אם כך $c_k = n-k$ אם"ם ל- s יש מחזור k (ואחרת $c_k < n-k$). אם כך, יש לנו האלגוריתם הבא לחישוב $T(s)$.

אלגוריתם

קלט: מחרוזת בינארית $s = s_0 s_1 \dots s_{n-1}$.

פלט: $T(s)$.

1. חשב את הפולינומים p, q כפי שהוגדרו לעיל.
2. חשב את הפולינום pq .
3. החזר את ה- k המינימאלי עבורו $c_k = n-k$, ואם לא קיים כזה החזר n .

נכונות

נכונות האלגוריתם נובעת מהדיון לעיל.

סיבוכיות

החלק הכבד באלגוריתם הוא שלב 2 שניתן לביצוע ב- $O(n \log n)$.

העשרה:

7. נותר עדיין להראות איך לעבור מהיצוג של פולינום $A(x)$ מדרגה קטנה מ n על ידי וקטור y של ערכיו ב n שרשי היחידה, ליצוג שלו באמצעות וקטור a של מקדמיו. לצורך זה נציג את y באמצעות כפל מטריצה ון דר מונדה של שרשי היחידה V_n בוקטור עמודה a . כתיבה מפורשת של הכפל $V_n a = y$ מופיעה כאן:

$$\underbrace{\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w_n & w_n^2 & \dots & w_n^{n-1} \\ 1 & w_n^2 & (w_n^2)^2 & \dots & (w_n^2)^{n-1} \\ \vdots & & & & \\ 1 & w_n^{n-1} & (w_n^{n-1})^2 & \dots & (w_n^{n-1})^{n-1} \end{pmatrix}}_{V_n} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

המטריצה V_n אינה סינגולרית, לכן a מקיים את השויון $a = V_n^{-1} \cdot y$. כעת נראה ש V_n^{-1} היא מטריצה בעלת מבנה דומה לזה של V_n כאשר במקום w_n משתמשים ב w_n^{-1} , ולכן ניתן לחשבה באמצעות FFT. לצורך זה נשתמש בלמה הבאה:

למה (סכום שרשי היחידה): יהי $w \neq 1$ שורש יחידה מסדר n . אז $\sum_{k=0}^{n-1} w^k = 0$

$$\text{הוכחה: ע"פ נוסחת טור הנדסי } \sum_{k=0}^{n-1} w^k = \frac{w^n - 1}{w - 1} = \frac{1 - 1}{w - 1} = 0$$

טענה: $[V_n^{-1}]_{j,k} = \frac{w_n^{-jk}}{n}$ עבור $0 \leq j, k \leq n-1$, כלומר:

$$\frac{1}{n} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w_n^{-1} & (w_n^{-1})^2 & \dots & (w_n^{-1})^{n-1} \\ 1 & (w_n^{-1})^2 & ((w_n^{-1})^2)^2 & \dots & ((w_n^{-1})^2)^{n-1} \\ \vdots & & & & \\ 1 & (w_n^{-1})^{n-1} & ((w_n^{-1})^{n-1})^2 & \dots & ((w_n^{-1})^{n-1})^{n-1} \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

הוכחה: נראה ש: $V_n^{-1} \cdot V_n = I$. לצורך זה נחשב את ערך הכניסה ה j', j במטריצת המכפלה

$$(V_n^{-1} \cdot V_n)_{j',j} = \sum_{k=0}^{n-1} \frac{w_n^{-j'k}}{n} \cdot w_n^{jk} = \frac{1}{n} \sum_{k=0}^{n-1} w_n^{k(j-j')}$$

אם $j' = j$ אז התוצאה היא 1. אחרת $w_n^{j-j'}$ הוא שורש יחידה מסדר n שונה מ 1 (כי $0 < |j-j'| < n$), ולכן לפי למת סכום שרשי היחידה, התוצאה היא 0. מ.ש.ל.

מסקנה: $a = V_n^{-1} \cdot y$ כלומר המקדם a_k נתון ע"י:

$$a_k = \sum_{j=0}^{n-1} (V_n^{-1})_{kj} \cdot y_j = \frac{1}{n} \sum_{j=0}^{n-1} y_j \cdot w_n^{-jk}$$

$$y_k = \sum_{j=0}^{n-1} a_j \cdot (w_n^k)^j$$

כלומר המקדמים $a_k, k = 0, \dots, n-1$ הם ערכי הפולינום שמקדמיו $\frac{y_j}{n}$ ב n שורשי היחידה

$$: n \text{ מסדר } (w_n^{-1})^0, (w_n^{-1})^1, \dots, (w_n^{-1})^{n-1}$$

מכאן שניתן לחשב את מקדמי הפולינום $A(x)$ מוקטור y של ערכיו ב n שרשי היחידה על ידי הכנסת השינויים הבאים באלגוריתם ל FFT שניתן בתחילת ההרצאה:

1. להחליף תפקידים בין a ו y .
2. להחליף שורש היחידה הפרימיטיבי w_n בשורש הפרימיטיבי ההופכי w_n^{-1} .
3. לבצע FFT עם הקלט המוגדר בסעיפים 1 ו 2.
4. לחלק את אברי וקטור התוצאה ב n .