אלגוריתמים – ממ"ן 13

שם : איתי אירמאי

תז : 204078224

תאריך הגשה : 04/01/2020

מייל : I.tai307@gmail.com

(הערה כללית : אני מניח בכל השאלות על FFT שדרגת הפולינום הינה חזקה של 2. יישום האלגוריתם העיד בבירור על הבעייתיות בפירוק דרגות ראשוניות.)

.1

.א

יהא $p(x) = x^3 + 2x^2 -3x - 1 = (-1,-3,2,1)$.

נחשב את ייצוג הפולינום ב-4 שורשי היחידה, $FFT(p(x),w_4)$, כאשר $w_i = cis(2\pi i / 4)$,

לפי הנוסחא הרקורסיבית $p(x) = p\_even(x^2) + x * p\_odd(x^2)$ על חצי מהשורשים.

pe(x) = (-1,2) // Split to even poly and run FFT with 2 roots.

pee(x) = (-1) // Split + FFT with 1 root.

peo(x) = (2)

pee(cis(0)) = -1, peo(cis(0)) = 2 // Both are of degree = 0, so return the constant value for all roots.

// Go back one call and calculate pe.

$pe(cis(0)) = pee(cis^2(0)) + cis(0) * peo(cis^2(0)) = -1 + 1 * 2 = 1$

$pe(cis(\pi)) = pee(cis^2(\pi)) + cis(\pi) * peo(cis^2(\pi)) = pee(cis(2\pi)) - cis(0) * peo(cis(2\pi)) =$

$= pee(cis(0)) - cis(0) * peo(cis(0)) = -1 - 1 * 2 = -3$

// 1st call's even poly is complete, apply FFT to odd with 2 roots.

po(x) = (-3,1)

poe(x) = (-3)

poo(x) = (1)

poe(cis(0)) = -3, poo(cis(0) = 1

$po(cis(0)) = poe(cis^2(0)) + cis(0) * poo(cis^2(0)) = -3 + 1 * 1 = -2$

$po(cis(\pi)) = poe(cis^2(\pi)) + cis(\pi) * poo(cis^2(\pi)) = -3 - 1 * 1 = -4$

// Back to main call.

$p(cis(0)) = pe(cis^2(0)) + cis(0) * po(cis^2(0)) = 1 + 1 * -2 = \mathbf{-1}$

$p(cis(\pi/2)) = pe(cis^2(\pi/2)) + cis(\pi/2) * po(cis^2(\pi/2)) = -3 + I * -4 = \mathbf{-3 - 4i}$

$p(cis(\pi)) = pe(cis^2(\pi)) + cis(\pi) * po(cis^2(\pi)) = 1 - 1 * -2 = \mathbf{3}$

$p(cis(3\pi/2)) = pe(cis^2(3\pi/2)) + cis(3\pi/2) * po(cis^2(3\pi/2)) = -3 - I * -4 = \mathbf{-3 + 4i}$

$\Rightarrow$ **$FFT(p(x),w_4) = (-1,-3-4i,3,-3+4i)$**

ב.

נחשב את מקדמי הפולינום p(x) המקורי באמצעות הרצת FFT הופכי :

$INV\_FFT(p(x),w_4^{-1}) = (FFT(p\_fft(x),w_4) / r) \circ (1)(4\ 3\ 2)$

$p2(x) = (-1,-3-4i,3,-3+4i)$

$p2e(x) = (-1,3)$

$p2ee(x) = (-1)$

$p2eo(x) = (3)$

$p2ee(cis(0)) = -1$, $peo(cis(0)) = 3$

$p2e(cis(0)) = p2ee(cis^2(0)) + cis(0) * p2eo(cis^2(0)) = -1 + 1 * 3 = 2$

$p2e(cis(\pi)) = p2ee(cis^2(\pi)) + cis(\pi) * p2eo(cis^2(\pi)) = -1 - 1 * 3 = -4$

$p2o(x) = (-3-4i,-3+4i)$

$p2oe(x) = (-3-4i)$

$p2oo(x) = (-3+4i)$

$p2oe(cis(0)) = -3-4i$, $poo(cis(0) = -3+4i$

$p2o(cis(0)) = p2oe(cis^2(0)) + cis(0) * p2oo(cis^2(0)) = -3-4i + 1 * (-3+4i) = -6$

$p2o(cis(\pi)) = p2oe(cis^2(\pi)) + cis(\pi) * p2oo(cis^2(\pi)) = -3-4i - 1 * (-3+4i) = -8i$

$p2(cis(0)) = p2e(cis^2(0)) + cis(0) * p2o(cis^2(0)) = 2 + 1 * -6 = \mathbf{-4}$

$p2(cis(\pi/2)) = p2e(cis^2(\pi/2)) + cis(\pi/2) * p2o(cis^2(\pi/2)) = -4 + I * (-8i) = \mathbf{4}$

$p2(cis(\pi)) = p2e(cis^2(\pi)) + cis(\pi) * p2o(cis^2(\pi)) = 2 - 1 * -6 = \mathbf{8}$

$p2(cis(3\pi/2)) = p2e(cis^2(3\pi/2)) + cis(3\pi/2) * p2o(cis^2(3\pi/2)) = -4 - I * (-8i) = \mathbf{-12}$

$\Rightarrow \mathbf{INV\_FFT(p(x),w_4^{-1})} = ((-4,4,8,-12) / 4) \circ (1)(4\ 3\ 2) = (-1,1,2,-3) \circ (1)(4\ 3\ 2) =$

$= \mathbf{(-1,-3,2,1) = p(x)}$ ▌

.2

<u>בעיה</u>: בהינתן שני מספרים שלמים X, Y המיוצגים ע״י N ביטים, אנו מעוניינים לחשב את X*Y בצורה יעילה.

<u>תיאור</u>: נבצע רדוקציה להכפלת שני פולינומים ב-FFT והצבת ערך.

ממיר קלט: נפרק את X,Y ל (n/k) קבוצות ביטים (בלוקים) עבור K כלשהו בצורה הבאה:

$$X = (x_0 + x_1 * 2^k + x_2 * 2^{2k} + x_3 * 2^{3k} + \ldots + x_{n/k-1} * 2^{(n/k-1)*k})$$

$$Y = (y_0 + y_1 * 2^k + y_2 * 2^{2k} + y_3 * 2^{3k} + \ldots + y_{n/k-1} * 2^{(n/k-1)*k})$$

כאשר N, K הינם חזקות של 2 (ראה הערה כללית); המקדמים הינם מספרים טבעיים בעלי ערך מירבי $x_i,y_i < 2^k$. נתבונן בפולינומים הבאים:

$$P_x(x) = (x_0 + x_1*x + x_2*x^2 + x_3*x^3 + \ldots + x_{n/k-1}*x^{(n/k-1)}) = (x_0,x_1,\ldots,x_{n/k-1})$$

$$P_y(x) = (y_0 + y_1*x + y_2*x^2 + y_3*x^3 + \ldots + y_{n/k-1}*x^{(n/k-1)}) = (y_0,y_1,\ldots,y_{n/k-1})$$

קופסא שחורה: נריץ FFT + INV_FFT על מנת לקבל את P_{xy} של הפולינומים הנ״ל:

$$P_{xy}(x) = (xy_0 + xy_1*x + xy_2*x^2 + xy_3*x^3 + \ldots + xy_{2(n/k-1)}*x^{2(n/k-1)}) = (xy_0,xy_1,\ldots,xy_{2n/k-2})$$

נצטרך לנתח מחדש את סיבוכיות האלגוריתם עבור פעולות ביטים מטה.

ממיר פלט: ① נציב את הנקודה 2^k בפולינום P_{xy}.

$$P_{xy}(2^k) = (xy_0 + xy_1* 2^k + xy_2*2^{2k} + xy_3*2^{3k} + \ldots + xy_{2n/k-2}*2^{2n/k-2})$$

② <u>ערך זה שווה לערך X*Y</u>, אך אינו בהכרח מהווה את הייצוג בבלוקים שלו, מכיוון שהמקדמים xy_i עלולים לחרוג ממגבלת 2^k שקבענו בייצוג לגודל בלוק; צריך להרכיב מחדש את המספר ע״י הזחת ביטים שמאלה של המקדמים וחיבור האזורים החופפים, כפי שיפורט בהמשך, ואת <u>התוצאה נחזיר כ-X*Y</u>.

<u>פירוט</u>: **בממיר הקלט**, אין צורך לחשב את המקדמים – הם כבר קיימים כחלק מהייצוג; לדוגמא, כאשר K=1, כל אחד מהמקדמים הינו 0 / 1 בהתאם לאם הביט במקום I דולק או לאו.

מה שצריך לעשות זה להעתיק אותם החוצה בלולאה: למשל, וגם (&) עם מסכת 1ים באורך K, והזחת K ביטים שמאלה של המסכה (לא יעיל), לעומת העתקת K ביטים וקפיצה למיקום K+. סיבוכיות פעולה זו תלויה ברמת הגישה של היישום לזיכרון – **O(n)** פעולות ביטים אם היא ישירה, $O(n^2/k)$ עבור הפעלת המסכה על כל המספר בכל איטרציה. מן הסתם נעדיף ליישם ברמה נמוכה ככל האפשר.

נשים לב כי בפולינומים שהוגדרו לעיל, $P_x(2^k) = X$, $P_y(2^k) = Y$. כפי שצוין בהרצאה,

$P_x(p) * P_y(p) = P_{xy}(p)$, כאשר p הינה נקודה מסוימת ו-P_{xy} הינו הפולינום המתקבל מהכפלת שני פולינומים (כלשהם); מכיוון שבנוסף $X * Y = XY$ על פי הגדרה, נובע כי $XY = P_{xy}(2^k)$. כלומר, <u>חישוב ערך הפולינום P_{xy} בנקודה 2^k יחזיר את הערך הרצוי</u> (בצורת ביניים).

לגבי **הקופסא השחורה**, דרישת היעילות בהכפלת שלמים משנה את הגדרת הסיבוכיות ואופן החישוב של FFT בשתי דרכים:

① לא ניתן לחשב מראש שום ערך CIS השונה מ 0 או π (קרי 1 או -1 בהתאמה), מכיוון שכל שאר שורשי היחידה הינם מרוכבים וערכיהם לא שלמים (פרט ל cis(π/2), עדיין נותר נפרד) – גם אם נחשב את כולם מראש בטבלה כלשהי, נניח, אין אפשרות לבצע הכפלה של לא שלמים, שהינה בלתי נמנעת ב-FFT.

למרות זאת, אנו יודעים שאם $x,y \in N$, אזי $x * y \in N$ (ובפרט כל הבלוקים שלהם, שגם הם מייצגים מספרים טבעיים); מנכונות INV_FFT נקבל שכל האיברים <u>המרוכבים בהכרח יתקזזו</u> בסוף ריצת האלגוריתם, ונוכל להשתמש רק בזהויות טריגונומטריות בשלבי הביניים.

ספציפית, **cis(x + π) = -cis(x) וכן cis(x)*cis(y) = cis(x + y)**. כלומר, <u>אין שום הכפלת מרוכבים</u> – לכל מקדם שלם באלגוריתם תוצמד זווית cis (מאותחלת ב-0) <u>והכפלה ב-cis אחר תומר לחיבור הזווית או שינוי סימן כנדרש</u> – <u>**O(logK)**</u> זניח, מכיוון שכפל השורשים מהווה חבורה מעל K.

② הכפלה בין שני **מקדמים שלמים** נשארת הכפלה 'אמיתית', בסיבוכיות **$O(K1*K2)$** בהתאם למספר הביטים בכל מקדם. החילוק היחיד שיש באלגוריתם הוא בקבוע n (אחרי INV_FFT), וניתן להמיר אותו להזחה שמאלה עקב ההנחה על גודל n.

③ הוספת / חיסור שני מקדמים בעלות סיבוכיות **$O(K1+K2)$**.

נביט בשלבי הרקורסיה עצמם, תחילה של FFT:

ברמת העומק המקסימאלית, $pe(cis(0)) = pe(cis(\pi)) = d1$, ומצד שני $po(cis(0)) = po(cis(\pi)) = d2$.

ברמה אחת מעליה, $p(cis(r)) = pe(cis^2(r)) + cis(r) * po(cis^2(r)) = d1 * cis(0) + d2 * cis(r)$.

אין זהות טריגונומטרית המאחדת חיבור של שתי זוויות שונות (פרט למקרה מיוחד של $X+Y = \pi$ ומקדמים זהים). לכן, עבור כל נקודה של פולינום באורך D (ברמה הנוכחית) נצטרך לשמור לכל היותר D מקדמים + זוויות ⇐ לבצע $O(D)$ הכפלות זווית וחיבורים בין מקדמים, בכל רמה.

הכפלת נקודות של P_x,P_y: קיימות $2n/k$ נקודות, ועל פי הנחת התרגיל ההכפלה דורשת $O(k^2)$ פעולות על ביטים. [לא בטוח שזה נכון – כל נקודה מכילה בתיאוריה בין 1 עד n/k זוויות ומקדמים שצריך להכפיל ולחבר, אבל כנראה פחות בממוצע. קשה להוכיח.] ⇐ בסה"כ $O(n*k)$ פעולות, חד פעמי.

INV_FFT: כל מקדם בפולינום הראשוני כבר מכיל $O(n/k)$ זוויות; כל פעולת איחוד ברקורסיה מלווה ב $O(n/k)$ הכפלות זווית, ואחריהן $O(n/k)$ חיבורי מקדמים. בהנחה שהמקדמים מוגבלים בגודל K, סה"כ פעולות הביטים יהיה $O(n)$ לכל רמת רקורסיה.

לבסוף, **ממיר פלט** – פעולת הרכבה מחדש: בהינתן שהפולינומים X,Y בעלי מקדמים שלמים בטווח $x_i,y_i < 2^k$, מקדמי המכפלה P_{xy} צריכים להיות בטווח הבא:

$$xy_i < (2^{2k} - 1) * (n/k - | n/2k - 1 - i |)$$

הסבר הנוסחא – 2^{2k} מגיע מהכפלת שני מקדמים מקסימליים בX,Y; והחלק השני מגיע ממספר אופני ההכפלה של חזקות P_x,P_y לקבלת x^i. (לדוגמא, $i=0$ / n מגיע רק מהכפלת שתי החזקות הראשונות או האחרונות, $i=1$ מסכום מקדמי חזקה 0 של P_x ו-1 של P_y ולהיפך. במרכז יש בדיוק n/k שילובים.)

מספר החפיפות המקסימאלי בין בלוקים לכל בלוק הינו $\log(xy_i)/k$:

$$\log(xy_i) < \log((2^{2k} - 1) * (n/k - | n/2k - 1 - i |)) < 2k + 1 + \log(n/k) = 2k + 1 + \log(n) - \log(k)$$

או לחלופין, $O(1 + \log(n)/k)$ חפיפות לבלוק [בפועל משהו כמו 0 עד 3 בממוצע].

אלגוריתם ההרכבה עצמו כדלהלן: עבור בלולאה על כל המקדמים xy_i, הזח שמאלה כל מקדם ב $k*i$ ביטים, וחבר את כולם. יישום יעיל של ההרכבה בעל סיבוכיות $O(n * \log(n)/k)$, כלומר פעולת חיבור בהתאם למספר החפיפות הנקודתיות ואחרת העתקה.

סיבוכיות: נניח כי $k = \log(n)$, כמתבקש בתרגיל.

① המרת הקלט (חלוקה לבלוקים) דורשת $O(n)$ פעולות ביטים כמפורט.

② מבין FFT ו-INV_FFT, השני הוא המקרה הגרוע יותר. הראינו כי הוא מתכנס ל $O(n)$ פעולות בכל רמת רקורסיה, וקיימות n/k רמות ⇐ נוסחת הנסיגה הינה:

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

נציב $n' = n/k$:

$$T(n') = 2T(n'/2) + O(n'*k) = 2T(n'/2) + O(n'*\log(n'))$$

חישוב יחס הנסיגה מראה כי **$T(n) = O(n*\log^2(n))$**.

③ פעולת ההכפלה בין FFT ל-INV_FFT בסיבוכיות $O(n*k) = O(n*\log(n))$.

④ המרת הפלט בסיבוכיות $O(n*\log(n) / k) = O(n)$.

בסה"כ, סיבוכיות תהליך הרדוקציה לכפל שלמים ב-FFT עבור $k=\log(n)$ הינה **$O(n*\log^2(n))$**, כנדרש. ∎

.3

בהינתן פולינום $f(x) = a_0 + a_1 * x + a_2 * x^2 + \ldots + a_n * x^n$, אנו מעוניינים לחשב את כל ערכי נגזרותיו בנקודה x_0.

נשים לב כי ערך איבר i בפולינום בנגזרת k>0 הינה:

$$① \ \mathbf{f^{(k)}(x)[i]} = i * (i-1) * \ldots * (i-k+1) * a_i * x^{i-k} = \mathbf{I!/(i-k)! * a_i * x^{i-k}} \qquad ;\ 0 \text{ when } k > I.$$

נביט בשני הפולינומים הבאים:

$$P(x) = (x_0^{\,i} / i!)\ ,\ i = 0{:}\,n$$

$$T(x) = ((n-i)! * a_{n-i})\ ,\ i = 0{:}\,n$$

חישוב המקדמים הנ"ל בצורה יעילה ניתן לביצוע בלולאה בסיבוכיות $O(n)$, אם נתעלם מהעובדה שהכפלת מספרים ענקיים כמו עצרת היא עצמה בעלת סיבוכיות לא טריוויאלית, לפי הנוסחאות:

$$p_{i+1} = p_i * x_0 / (i+1)\ ;\ t_{i+1} = t_i * (n-i-1) / a_{n-i} * a_{n-i-1}$$

נמצא את מקדמי הפולינום $PT(x) = P(x) * T(x)$ באמצעות FFT + INV_FFT בסיבוכיות $O(n*\log(n))$.

מהם ערכי המקדמים?

$$② \ \mathbf{PT[i]} = \sum p_j * t_{i-j} = \qquad ;\ j = 0{:}\,i\ \ \&\&\ \ I \le n$$

$$= \sum (x_0^{\,j} / j!) * ((n-i+j)! * a_{n-i+j}) = \sum (n-I+j) * (n-I+j-1) * \ldots * (j+1) * a_{n-i+j} * x_0^{\,j} =$$

$$=① \ \sum f^{(n-i)}(x)[n-I+j] = \mathbf{f^{(n-i)}(x)}$$

[שימו לב להצבה בין נוסחאות:

$i1 = n2 - i2 + j2,\ i1 - k1 = j2 \Rightarrow \mathbf{i1 = n2 - i2 + j2}\ ;\ \mathbf{k1} = n2 - i2 + j2 - j2 = \mathbf{n2 - i2}$]

מסתבר שכל אחד מהמקדמים בפולינום PT עד דרגה n תואם לנגזרת $n - i$, עקב התאמה בין האיברים הבדידים. אם כך, נחזיר את <u>n המקדמים הראשונים של PT בסדר הופכי</u> ונקבל את n ערכי הנגזרות בנקודה x_0 כנדרש.

<u>סיבוכיות</u>: $\mathbf{O(n*\log(n))}$ עבור הרצת FFT, ועוד $O(n)$ על המרות קלט ופלט. ▌

4.

אלגוריתם Strassen כמתואר (חלקית) הינו רקורסיבי:

בהינתן שתי מטריצות A,B מגודל (n x n),

① יוצרים 8 מטריצות מגודל (n/2 x n/2) ע״י חלוקת A,B לרביעים.

② מתוכן מחשבים 7 מטריצות בגודל (n/2 x n/2) כתוצר 1-2 פעולות חיבור או חיסור מטריצות ופעולת כפל מטריצות רקורסיבית אחת לכל מטריצה.

③ מתוך 7 המטריצות מחשבים 4 מטריצות בגודל (n/2 x n/2) באמצעות 2-4 פעולות חיבור וחיסור מטריצות.

④ את 4 המטריצות האחרונות משרשרים חזרה כרביעים לתוצאה A*B.

בסה״כ, אם נסמן

T(n) = כפל מטריצות מגודל (n x n),

P(n) = חיבור / חיסור מטריצות מגודל (n x n),

S(n) = חלוקת מטריצה מגודל (n x n) לרביעים בגודל (n/2 x n/2),

M(n) = שרשור 4 מטריצות מגודל (n/2 x n/2) כרביעים למטריצה בגודל (n x n),

נקבל את נוסחת הנסיגה הבאה:

$$T(n) = 7 * T(n/2) + 2 * S(n) + O(P(n/2)) + M(n)$$

הסיבוכיות הנאיבית של הבעיות P, S, M הינה:

$$P(n) = O(n^2)\ ;\ S(n) = O(n^2)\ ;\ M(n) = 4 * O((n/2)^2) = O(n^2)$$

P = לולאה מקוננת על השורות והעמודות, סכימת או חיסור האיברים בשתי המטריצות למטריצה חדשה; S = לולאה מקוננת על השורות והעמודות, העתקת איברים למטריצות החדשות בהתאם למיקום; M = לולאה מקוננת על המטריצות, שורותיהן ועמודותיהן, העתקה למטריצה חדשה ברביע המתאים (row +n/2 OR col +n/2 OR either +0).

נחזור לנוסחת הנסיגה:

$$T(n) = 7 * T(n/2) + 2 * O(n^2) + O((n/2)^2) + O(n^2) = 7 * T(n/2) + O(n^2)$$

על פי נוסחת ה-Master theorem, האלגוריתם מקיים $a = 7,\ b = 2,\ f(n) = O(n^2) \Rightarrow c = 2$;

$$2.807 \sim \mathbf{log_b(a)} > \mathbf{c} = 2$$

⇐ לפי המקרה הראשון, $\mathbf{T(n)} = O(n^{\log(a)/\log(b)}) = \mathbf{O(n^{\log(7)/\log(2)})}$. ∎