קורס 20417 סמסטר 2017ב

אודי ליוטשי 036745438

ממ"ן 13

p(x) = x^3 + 2x^2 -3x -1   (1

a. call FFT((-1,-3,2,1), w=i)
    call FFT((-1,2), w^2=-1)
        call FFT ((-1),w^4=1) -> return -1 (תנאי עצירה מתקיים)
        call FFT ((2),w^4=1) -> return 2 (תנאי עצירה מתקיים)

        return (-1+w^0*2, -1-w^0*2) = (1,-3)

    call FFT((-3,1), w^2=-1)
        call FFT ((-3),w^4=1) -> return -3 (תנאי עצירה מתקיים)
        call FFT ((1),w^4=1) -> return 1 (תנאי עצירה מתקיים)

        return (-3+w^0*1, -3-w^0*1) = (-2,-4)

   for loop iteration 1:

   v1=1+w^0*-2 = -1

   v3=1-w^0*-2 = 3

   for loop iteration 2:

   v2=-3+w^1*-4 = -3-4i

   v4=-3-w^1*-4 = -3+4i

   return (-1, -3-4i, 3, -3+4i)

b. call FFT^{-1} ((-1, -3-4i, 3, -3+4i), w^-1 = -i)

    call FFT^{-1} ((-1,3), w^-2=-1)

        call FFT^{-1} (-1, w^-4=1) -> return -1 (תנאי עצירה מתקיים)

        call FFT^{-1} (3, w^-4=1) -> return 3 (תנאי עצירה מתקיים)

        return (-1+1*3, -1-1*3) = (2,-4)

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

        call FFT^{-1} (-3-4i, w^-4=1) -> return -3-4i (תנאי עצירה מתקיים)

        call FFT^{-1} (-3+4i, w^-4=1) -> return -3+4i (תנאי עצירה מתקיים)

        return (-3-4i +1*(-3+4i), -3-4i -1*(-3+4i)) = (-6, -8i)

   for loop iteration 1:

   v1=(2+1*-6) = -4

v3=(2-1*-6) = 8

for loop iteration 2:

v2=(-4+-i*-8i) = -12

v4=(-4-(-i)(-8i)) = 4

return (-4,-12,8,4) [= (-1,-3,2,1)*4]

(2

האלגוריתם:

1. נחלק את הקלט לn/k בלוקים בגודל k באופן הבא:

$x = P1(2) = \sum_{i=0}^{\frac{n}{k}-1} x_i \qquad y = P2(2) = \sum_{i=0}^{\frac{n}{k}-1} y_i$

x,y מהווים ייצוג פולינומי (וקטורים) של הקלט בבסיס 2 (ניתן להשתמש בכל בסיס אחר) ו- xi,yi מייצגים את הבלוק ה-i בגודל k ביטים ב-x,y בהתאמה.

2. נריץ FFT על x,y ונקבל את ערכיהם ב 2n/k נקודות שונות
3. נכפול את תוצאת שלב 2 על מנת לקבל את ערכי וקטור המכפלה V באופן הבא:

vi = P1(wi)*P2(wi), iϵ[2n/k נקודות שונות]

4. נריץ INVERSE-FFT על תוצאת שלב 3 על מנת לחלץ את מקדמי וקטור המכפלה V
5. מכפלת שני המספרים שקיבלנו מתקבלת ע"י פעולת סכימה :

result = $\sum_{i=0}^{\frac{2n}{k}-1} v_i * 2^{ik}$

ניתוח סיבוכיות (ממוספר לפי שלבי האלגוריתם):

1. חלוקת הקלט: O(n) (למעשה החסם קטן יותר אך לא רלוונטי לחישוב הכללי)
2. 2 הרצות של FFT:

input size = 2n bits
assuming Ɵ(k^2) for each recursion call
let k=log n, then:
T(2n) = 2T(n) + n/k Ɵ (k^2) = 2T(n) + Ɵ (n logn) = Ɵ (n log^2(n))

3. הכפלה: O(2n/k * k^2) = O(nk)
4. הרצת INVERSE-FFT: Ɵ (n log^2(n)) בדומה לשלב 2
5. סכימה: O(n/k) לפי הספר

סה"כ: Ɵ (n log^2(n))

3) נתון פולינום f מסדר n ונקודה x_0.
נגדיר 2 וקטורים:

$$A = (n!\,a_n, (n-1)!\,a_{n-1}, \dots, 0!\,a_0)$$
$$B = \left(\frac{x_0^0}{0!}, \frac{x_0^1}{1!}, \dots, \frac{x_0^n}{n!}\right)$$

A וקטור של כל המקדמים של בכל הנגזרות של f.
B וקטור של חזקות של x_0 מחולקות בעצרת בגודל החזקה.

נכפול את הוקטורים באמצעות שימוש ב-FFT ונקבל וקטור המכיל את הערכים המבוקשים ב-n+1 איבריו הראשונים בסדר הפוך (ערך הנגזרת הגבוהה יותר יופיע קודם).
בדיקה:
נניח ואנו מעוניינים בערך הנגזרת ה-k של f בנקודה x_0

$$\sum_{0\le i,j\le k \; and \; i+j=k} A_i B_j = A_0 B_k \dots A_k B_0 = n!\, a_n \frac{x_0^k}{k!} + \cdots + (n-k)!\, a_{n-k} \frac{x_0^0}{0!} =$$

$$= \frac{n!}{k!} a_n x_0^k + \dots + (n-k)!\, a_{n-k} = f^{(k)}(x_0)$$

ניתן לראות שקיבלנו את הערך המבוקש.

סיבוכיות זמן ריצה:

בניית הוקטורים A,B נעשית בלולאה שרצה n+1 פעמים : 2*O(n+1) = O(n)

כפל פולינומים באמצעות FFT (2 הרצות FFT + הרצת INVERSE-FFT): 3*Θ(nlogn) = Θ (nlogn)

סה"כ: Θ (nlogn)

4) האלגוריתם הינו רקורסיבי, בכל שלב האלגוריתם מפרק את הבעיה ל-7 תת בעיות כפל מטריצות בגודל מחצית מהשלב הקודם (n/2).
בכל שלב מתבצעות גם פעולות חיבור/חיסור של מטריצות שלוקחות סדר גודל של:
O(n/2*n/2) = O(n^2)

נקבל אם כך את נוסחת הנסיגה הבאה:

$$T(n) = 7T\left(\frac{n}{2}\right) + O(n^2)$$

מכאן שלפי למת המאסטר למקרה שבו $\log_2 7 > 1 = c$ נקבל:

$$T(n) = \Theta(n^{\log_2 7})$$