הוכן ע"י אמיר רובינשטיין

מבני נתונים
ומבא לאלגוריתמים

נושא 3

רקורסיה וטכניקת הפרד-משול-צרף
Recursion and
divide-conquer-join technique

בתוכנית

פרק 4 בספר הלימוד

- ניזכר בגישת הפרד-משול-צרף לתכנון אלגוריתמים.
- נראה כיצד לבטא זמן ריצה של פונקציה רקורסיבית באמצעות נוסחת נסיגה
- נלמד 4 שיטות לתת פתרון אסימפטוטי (סדר גודל) לנוסחאות נסיגה:
  1. שיטת האיטרציות
  2. שיטת עץ הרקורסיה
  3. משפט האב (Master Theorem)
  4. שיטת ההצבה

(+ טכניקת החלפת משתנים)

טכניקת הפרד-משול-צרף

בהרצאה זו נפגוש את אחת הטכניקות החשובות לתכנון אלגוריתמים:
טכניקת הפרד-משול-צרף (divide-conquer-join).

בטכניקה זו:

- מפרידים את הבעיה המקורית לתת-בעיות קטנות יותר ← הפרד
- פותרים כל תת-בעיה בנפרד ← משול
- מצרפים את תת-הפתרונות לפתרון לבעיה המקורית ← צרף

למשל, במיון מיזוג:

- מחלקים את המערך לשני חצאים
- ממיינים כל חצי בנפרד
- ממזגים את שני החצאים

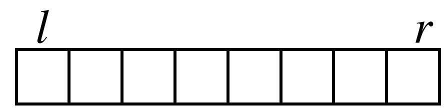
את זמן הריצה של אלגוריתם רקורסיבי ניתן לתאר ע"י נוסחת נסיגה.
נראה כעת כמה שיטות לפתרון נוסחאות נסיגה.

שיטה 1 - שיטת האיטרציות

<u>דוגמה 1</u>

ננתח את סיבוכיות הזמן של חיפוש בינארי רקורסיבי <u>במקרה הגרוע</u>

```
Binary-Search (A, l, r, key)
    if l > r
        return NIL
    mid ← ⌊(l+r)/2⌋
    if key < A[mid]
        return Binary-Search(A,l,mid-1,key)
    else if key > A[mid]
        return Binary-Search(A,mid+1,r,key)
    else return mid
```



הקריאה הראשית (n הוא גודל המערך):

Binary-Search(A, 1, n, key)

$$\boxed{t(n) = c + t(\lfloor n/2 \rfloor)} = 2c + t(\lfloor n/4 \rfloor) = 3c + t(\lfloor n/8 \rfloor) = \ldots = ic + t(\lfloor n/2^i \rfloor)$$

מתי $n/2^i < 1$?

אחרי $\Theta(\log n)$ איטרציות (או במדויק: $i = \lfloor \log n \rfloor + 1$ איטרציות).

$$= c\,(\lfloor \log n \rfloor + 1) + t(0) \quad = \Theta(\log n)$$

שיטה 1 - שיטת האיטרציות

<u>שני פרטים טכניים</u>

- גודל של בעיה הוא לרוב מספר שלם. יחד עם זאת, בד"כ אפשר להתעלם מערכי רצפה ותקרה, כאשר מנתחים סדר גודל אסימפטוטי.
  שיטת ההצבה (בהמשך) מאפשרת להוכיח שאכן מדובר בהבדל זניח.

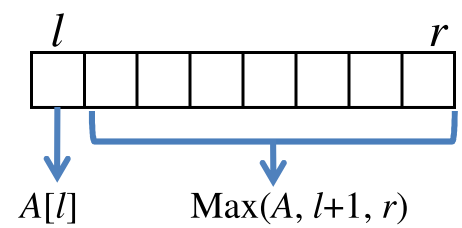
- אם לא צוין אחרת, אנו מניחים כי עבור n-ים קטנים מתקיים $t(n)=\Theta(1)$.

שיטה 1 - שיטת האיטרציות

דוגמה 2

לפנינו פונקציה רקורסיבית המחשבת מקסימום:

```
Max (A, l, r)
1.    if  l = r
2.          return A[l]
3.    return  maximum{ A[l]  , Max(A, l+1, r) }
```



הקריאה הראשית (n הוא גודל המערך):
$\mathrm{Max}(A, 1, n)$

מה סיבוכיות הזמן של Max כתלות ב- n?

$$\boxed{t(n) = c + t(n-1)} = 2c+t(n-2) = . \quad . \quad . = i\cdot c+t(n-i) =$$

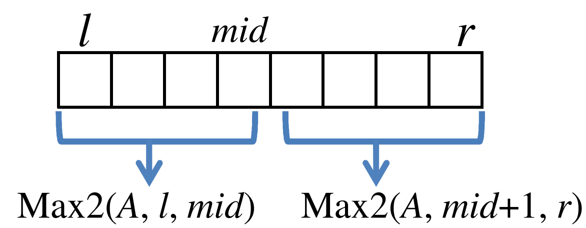
$$= (n-1)\cdot c + t(1) \quad = \Theta(n)$$

שיטה 1 - שיטת האיטרציות

המשך דוגמה 2

הנה גרסה רקורסיבית אחרת למציאת מקסימום:

```
Max2(A, l, r)
    if  l = r
        return  A[l]
    mid ← ⌊(l+r)/2⌋
    lmax ← Max2(A, l, mid)
    rmax ← Max2(A, mid+1, r)
    return  maximum{ lmax, rmax }
```



$$\boxed{t(n) = c + 2t(n/2)} \quad = c + 2c + 4t(n/4) \quad = c + 2c + 4c + 8t(n/8) = \ .\ \ .\ \ .$$

$$= c(1+2+\ \ldots\ +2^{i-1}) + 2^{i}\cdot t(n/2^{i})$$

$$= c(\underbrace{1+2+4+8+\ \ldots + 2^{\log n-1}}_{\text{סדרה הנדסית}}) + n\cdot t(1) \quad = c\cdot(2^{\log n}-1) + \Theta(n)$$

$$= \Theta(n)$$

שיטה 1 - שיטת האיטרציות

דוגמה 3

מיון מיזוג

```
Merge-Sort(A, l, r)
    if  l < r
        mid ← ⌊(l+r)/2⌋
        Merge-Sort (A, l, mid)
        Merge-Sort (A, mid+1, r)
        Merge  A[l..mid]   with   A[mid+1..r]
```

$$\boxed{t(n) = 2t(n/2) + n} \quad = 2(2t(n/4) + n/2) + n \ = 4t(n/4) + 2n$$

$$= 2^i \cdot t(n/2^i) \ + i \cdot n$$

$$= n \cdot t(1) \ + n\log n \quad = \Theta(n\log n)$$

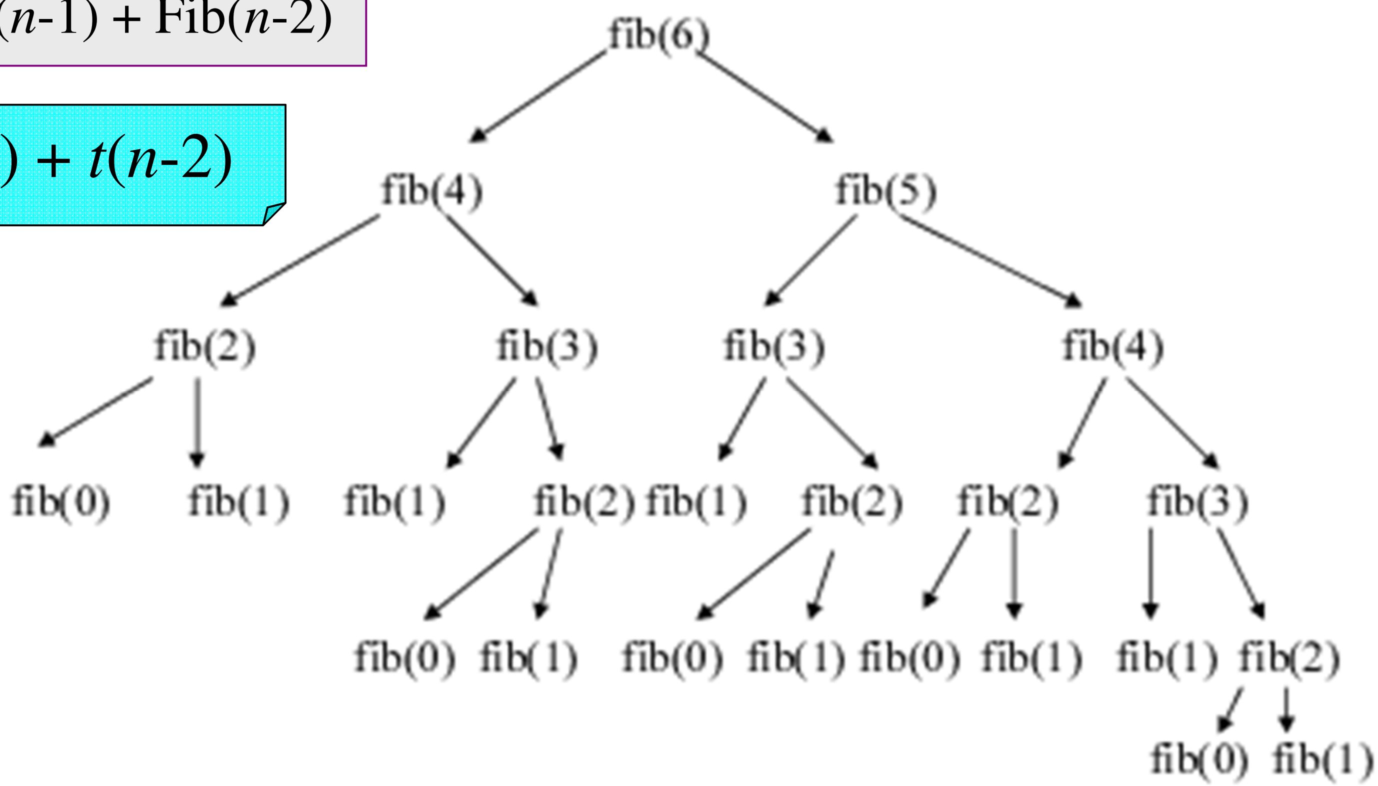
שיטה 1 - שיטת האיטרציות

דוגמה 4

חישוב האיבר ה- n בסדרת פיבונאצ'י

```
Fib(n)
    if  n = 0  or  n = 1
        return  n
    return  Fib(n-1) + Fib(n-2)
```

$$t(n) = c + t(n-1) + t(n-2)$$



שיטה 1 - שיטת האיטרציות

$$t(n) = c + t(n-1) + t(n-2)$$

המשך דוגמה 4

נמצא חסם עליון ותחתון

$$t(n) \leq c + 2t(n-1) \quad = c + 2c + 4t(n-2) \quad = c + 2c + 4c + 8t(n-3) = . \;\; . \;\; .$$

$$= c\,(1+2+ \ldots +2^{i-1}) + 2^{i} \cdot t(n-i)$$

$$= c\,(1+2+ \ldots +2^{n-1}) + 2^{n} \cdot t(0) \quad = c \cdot (2^{n}-1) + 2^{n} \quad = O(2^{n})$$

$$t(n) \geq c + 2t(n-2) \quad = c + 2c + 4t(n-4) \quad = c + 2c + 4c + 8t(n-6) = . \;\; . \;\; .$$

$$= c\,(1+2+ \ldots +2^{i-1}) + 2^{i} \cdot t(n-2i) \quad = \Omega(\sqrt{2^{n}})$$

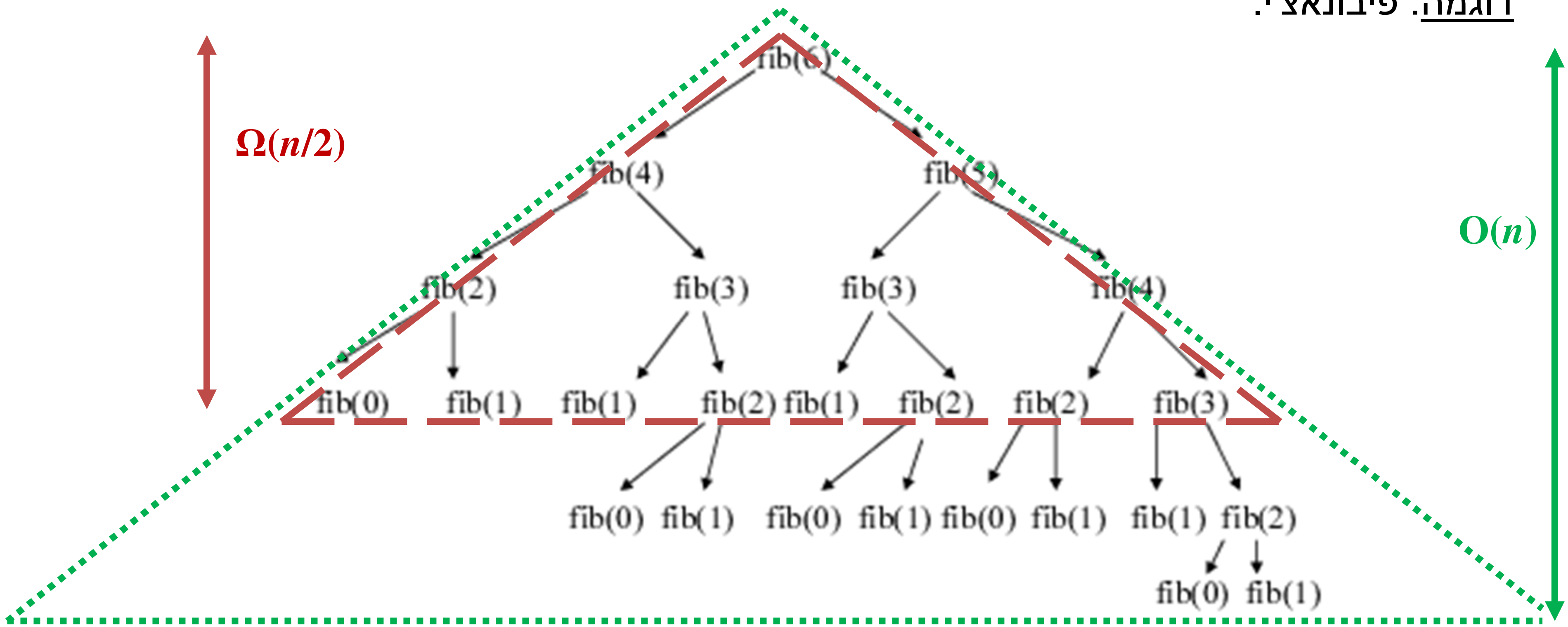
בשיטות אחרות, שנלמדות בקורסים אחרים, ניתן גם למצוא חסם הדוק:

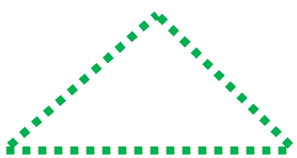
$$t(n) = \Theta(\Phi^{n}) \qquad \Phi = \frac{1+\sqrt{5}}{2}$$

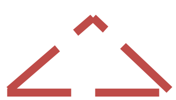
שיטה 2 - שיטת עץ הרקורסיה

מעין גרסה ויזואלית של שיטת האיטרציות. פורשים את עץ הקריאות הרקורסיביות, ומסכמים את כמות העבודה בכל הצמתים.

<u>דוגמה</u>: פיבונאצ'י.



 $= c(1+2+...+2^{n-1}) = c(2^n-1) = O(2^n)$

 $= c(1+2+...+2^{(n/2)}) = c(2^{n/2+1}-1) = \Omega(2^{n/2}) = \Omega(\sqrt{2}^n)$

שיטה 2 - שיטת עץ הרקורסיה

$t(n) = 2t(n/2) + n^2$

שיטה 2 - שיטת עץ הרקורסיה

$t(n)= t(n/3) + t(2n/3) + n$

<u>הכללה</u>: $t(n) = t(\alpha n) + t(\beta n) + cn$, כאשר $0<\alpha,\beta<1$, $\alpha+\beta=1$ ו- $c>0$ קבוע.

<u>שאלה</u>: מה משתנה כאשר $\alpha+\beta<1$?

שיטה 3: משפט האב (Master theorem)

- $a \geq 1$ ו $b > 1$ קבועים
- $f(n)$ פונקציה
- $T(n)$ מוגדרת על שלמים אי שליליים ע"י הרקורסיה: $T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$

ניתן לחסום את $T(n)$ אסימפטוטית באופן הבא:

1. **אם** $f(n) = O\left(n^{\log_b a - \varepsilon}\right)$ כאשר $\varepsilon > 0$ קבוע כלשהו,
   אזי $T(n) = \Theta\left(n^{\log_b a}\right)$

2. **אם** $f(n) = \Theta\left(n^{\log_b a}\right)$
   אזי $T(n) = \Theta\left(n^{\log_b a} \cdot \log n\right)$

3. **אם** $f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right)$ כאשר $\varepsilon > 0$ קבוע כלשהו,
   וגם $a \cdot f\left(\frac{n}{b}\right) \leq c \cdot f(n)$ עבור קבוע $c < 1$ והחל מ n מספיק גדול,
   אזי $T(n) = \Theta\left(f(n)\right)$.

הערה: המשמעות של $\frac{n}{b}$ יכולה להיות $\left\lfloor\frac{n}{b}\right\rfloor$ או $\left\lceil\frac{n}{b}\right\rceil$.

שיטה 3: משפט האב (Master theorem)

1. $T(n) = 9T\left(\frac{n}{3}\right) + n$

2. $T(n) = T\left(\frac{n}{2}\right) + 1$

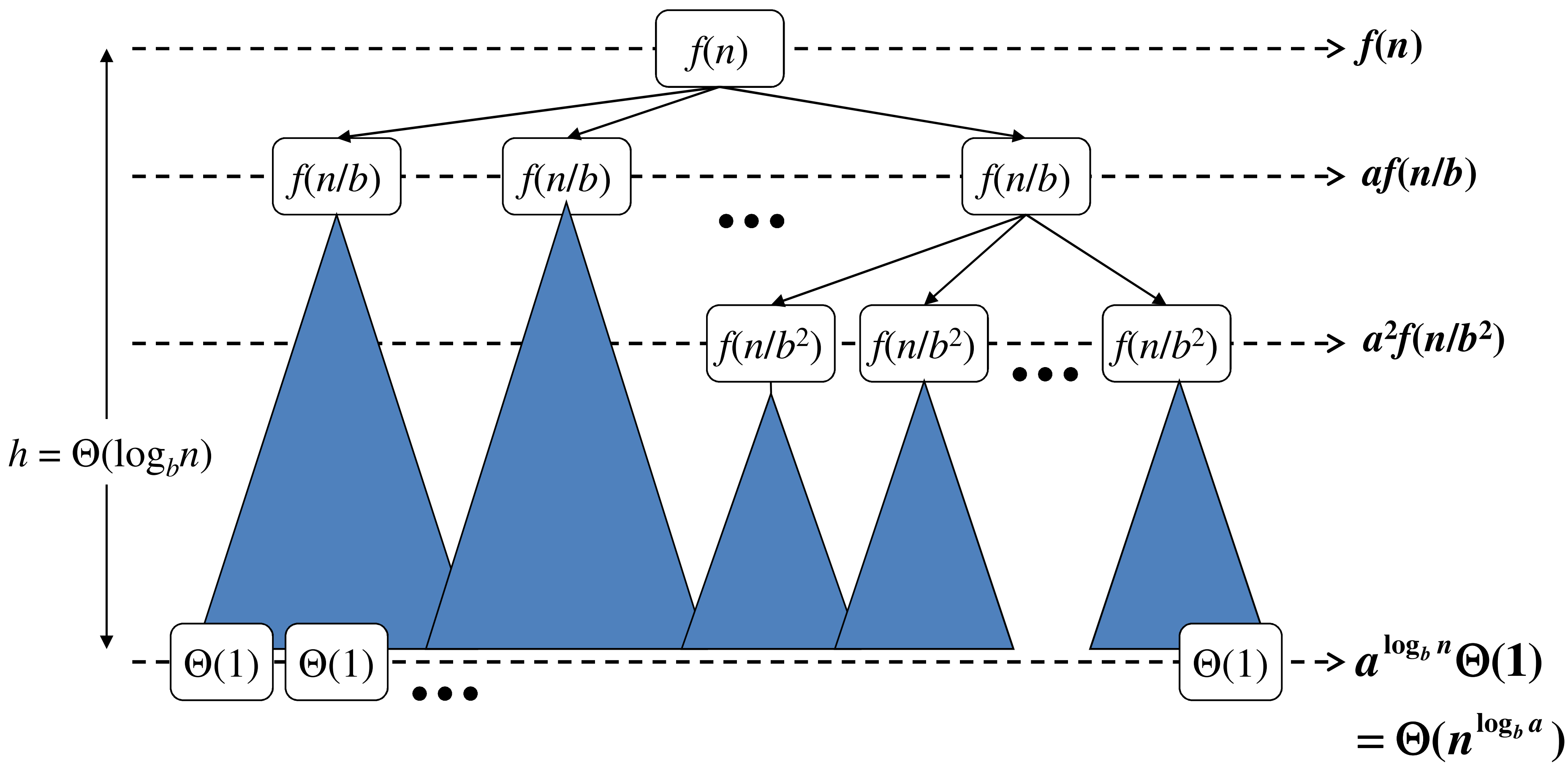
שיטה 3: משפט האב (Master theorem)

3. $T(n) = 3T\left(\frac{n}{4}\right) + n\log n$

4. $T(n) = 2T\left(\frac{n}{2}\right) + n\log n$

שיטה 3: משפט האב (Master theorem)

אינטואיציה למשפט (הוכחה מלאה מופיעה בספר הלימוד):



$$T(n) = \sum_{j=0}^{h-1} a^j f\left(\frac{n}{b^j}\right) + \Theta(n^{\log_b a})$$

שיטה 4: שיטת ההצבה

בשיטה זו אנו מנחשים פתרון ומוכיחים אותו באינדוקציה.

חסרון: השיטה לא מספקת פיתרון

חשיבות: מאפשר להוכיח פתרון לנוסחאות "קשות", שקל לנחש עבורן פתרון

דוגמאות לנוסחאות "קשות" שקל לנחש עבורן פתרון:

$$t(n) = t(\lceil n/2 \rceil) + 1 \qquad = \Theta(\log n)$$

$$t(n) = t(\lfloor n/2 \rfloor) + t(\lceil n/2 \rceil) + n \qquad = \Theta(n \log n)$$

$$t(n) = 2t(n/2 + 17) + 1 \qquad = \Theta(n)$$

שיטה 4: שיטת ההצבה

נדגים את השיטה על הנוסחה הבאה:

$$t(n) = 2t(\lfloor n/2 \rfloor) + n$$

נחש חסם עליון: $t(n) = O(n\log n)$

נוכיח באינדוקציה על n: קיים קבוע c כך שלכל n מספיק גדול $t(n) \le cn\log n$

הנחת האינדוקציה: $t(\lfloor n/2 \rfloor) \le c\lfloor n/2 \rfloor \log(\lfloor n/2 \rfloor)$

$$t(n) = 2t(\lfloor n/2 \rfloor) + n \le 2(c\lfloor n/2 \rfloor \log(\lfloor n/2 \rfloor)) + n$$

$$\le cn\log(n/2)) + n \le cn \; \ldots \; n + n \le cn\log n$$

עבור $c \ge 1$

מה לגבי הוכחת בסיס האינדוקציה?
באופן דומה מוכיחים חסם תחתון (תרגיל 4.1-2 בספר הלימוד).

החלפת משתנים

טכניקה שמאפשרת לפעמים להפוך נוסחה שנראית "קשה" לנוסחה "קלה".

$$t(n) = 2t(\sqrt{n}) + \log n$$

נסמן:

$$m = \log n$$

$$t(2^m) = 2t(2^{m/2}) + m$$

נסמן:

$$s(m) = t(2^m)$$

$$s(m) = 2s(m/2) + m$$

קל לפתור בכל אחת מהשיטות שראינו

$$s(m) = \Theta(m\log m)$$

$$t(n) = t(2^m) = s(m) = \Theta(m\log m) = \Theta(\log n \cdot \log\log n)$$

החלפת משתנים

$t(n) = 4\sqrt{n}\, t(\sqrt{n}) + n \cdot \log n \cdot \log\log n$

נחלק ב- n:

$$\frac{t(n)}{n} = 4\frac{t(\sqrt{n})}{\sqrt{n}} + \log n \cdot \log\log n$$

נסמן: $m = \log n$

$$\frac{t(2^m)}{2^m} = 4\frac{t(2^{m/2})}{2^{m/2}} + m \cdot \log m$$

נסמן: $s(m) = \dfrac{t(2^m)}{2^m}$

קל לפתור

$$s(m) = 4s(m/2) + m \cdot \log m = \Theta(m^2)$$

$$\frac{t(n)}{n} = s(m) = \Theta(m^2) = \Theta(\log^2 n)$$

$$t(n) = \Theta(n \log^2 n)$$

דוגמה - כפל מטריצות

- דרוש אלגוריתם לכפל מטריצות $C = A \cdot B$
  - A, B, C מטריצות ריבועיות בגודל n x n
  - תזכורת: האיבר C_{ij} של מטריצת המכפלה $C = A \cdot B$ מחושב ע"י המכפלה הפנימית $C_{ij} = \sum_{k=1}^{n} A_{ik}B_{kj}$
  - אלגוריתם פשוט:

Simple-Matrix-Multiply(A, B, C, n)

1. **for** $i \leftarrow 1$ **to** n
2.   **for** $j \leftarrow 1$ **to** n
3. $C[i, j] \leftarrow 0$
4.   **for** $k \leftarrow 1$ **to** n
5. $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$
6. **return** C

  - זמן ריצה $\Theta(n^3)$, זיכרון $\Theta(1)$

כפל מטריצות - המשך

- מה לגבי אלגוריתם רקורסיבי?
  - הרעיון: הפרד ומשול (נניח $n = 2^m$)

$$A \times B = \begin{bmatrix} A1 & A2 \\ A3 & A4 \end{bmatrix} \times \begin{bmatrix} B1 & B2 \\ B3 & B4 \end{bmatrix} = \begin{bmatrix} C1 & C2 \\ C3 & C4 \end{bmatrix} = C$$

  - נחשב רקורסיבית את 8 מכפלות הרבעים, ונחבר

$$C1 = A1 \cdot B1 + A2 \cdot B3 \qquad C3 = A3 \cdot B1 + A4 \cdot B3$$

$$C2 = A1 \cdot B2 + A2 \cdot B4 \qquad C4 = A3 \cdot B2 + A4 \cdot B4$$

  - נוסחת הנסיגה המתקבלת: $T(n) = 8T(n/2) + 4(n/2)^2$
  - פיתרון (מקרה 1 של שיטת האב): $T(n) = \Theta(n^3)$
  - אותו זמן ריצה אסימפטוטי, אך צריכת הזיכרון היא $\Theta(\lg n)$
  - המצב רק מחמיר!...

כפל מטריצות - המשך

- האלגוריתם של שטראסן (Strassen, 1969)
  – מספיקות רק 7 מכפלות רבעים!

$P1 = A1\cdot(B2 - B4)$

$P2 = (A1 + A2)\cdot B4$

$P3 = (A3 + A4)\cdot B1$

$P4 = A4\cdot(B3 - B1)$

$P5 = (A1 + A4)\cdot(B1 + B4)$

$P6 = (A2 - A4)\cdot(B3 + B4)$

$P7 = (A1 - A3)\cdot(B1 + B2)$

$C1 = P4 + P5 + P6 - P2$

$C2 = P1 + P2$

$C3 = P3 + P4$

$C4 = P1 + P5 - P3 - P7$

  – נוסחת הנסיגה המתקבלת: $T(n) = 7T(n/2) + 18(n/2)^2$
  – פיתרון (מקרה 1 של שיטת האב): $T(n) = \Theta(n^{\log_2 7}) = \Theta(n^{2.81})$
  – מעשית, עבור $n < 45$, האלגוריתם הפשוט עדיף
  – האלגוריתם הטוב ביותר כיום (משנת 1990, בעל עניין תיאורטי בעיקר): $\Theta(n^{2.376})$

שאלות חזרה

1. במיון מיזוג, מה עושים בכל אחד מהשלבים הפרד-משול-צרף ?

2. בשיטת עץ הרקורסיה, מדוע אנו מעוניינים למצוא את סך ערכי הצמתים בעץ?

3. מדוע לא ניתן להשתמש במשפט האב לפתרון נוסחת הנסיגה: $T(n)=2T(n/2) + n/\log n$?

4. נתונה נוסחת הנסיגה $T(n)=2T(n/2) + f(n)$.
   תנו דוגמה לפונקציה $f(n)$ שעבורה מתקיים המקרה השלישי של משפט האב, אבל לא מתקיים תנאי הרגולריות של מקרה זה.
   רמז: בפונקציה תהיה הפרדה למשל בין n זוגי ואי-זוגי.

תשובות לשאלות חזרה

1. הפרד: חציית המערך לשניים. משול: מיון רקורסיבי של כל חצי. צרף: מיזוג שני החצאים.

2. כל צומת מייצג זמן ריצה של שלב כלשהו בתהליך הרקורסיבי. סך ערכי הצמתים מייצג את סך זמן הריצה של כל התהליך הרקורסיבי.

3. כי לא מתקיים התנאי של אף אחד מהמקרים (גם לא מקרה 1 – מדוע?)

4. למשל: $f(n) = \begin{cases} n^2 & \text{if } n \text{ is even} \\ n^3 & \text{else} \end{cases}$

נניח בשלילה שמתקיים תנאי הרגולריות. כלומר קיים $c<1$ כך שלכל n גדול מספיק מתקיים:

$a\,f(n/b) \leq cf(n)$

אבל לכל n זוגי כזה ש- $n/2$ הוא אי-זוגי מתקיים:

$a\,f(n/b) = 2f(n/2) = 2(n/2)^3 = n^3/4 \leq cn^2$

וזה לא ייתכן כמובן.

תרגילים

<u>תרגילים נוספים</u>

1. מיצאו בשיטת האיטרציות חסם אסימפטוטי הדוק לנוסחת הנסיגה $T(n) = 3T(n/4) + n$.

2. תנו חסם הדוק אסימפטוטית לנוסחת הנסיגה הבאה: $t(n) = n^3 + t(n/3) + t(n/4)$.

3. שנו את אלגוריתם החיפוש הבינארי הרקורסיבי כך שבמקום לחלק את המערך לשני חלקים (כמעט) שווים, הוא יחלק אותו ביחס 1:2, כלומר: לחלק אחד בגודל שליש וחלק אחר בגודל שני שלישים. כיתבו את האלגוריתם החדש בפסאודו-קוד ונתחו את סיבוכיותו.

<u>פתרון 1</u>

$T(n) = 3T(n/4) + n$

$= 3\big(3T(n/4^2) + n/4\big) + n = 3^2T(n/4^2) + n(1+3/4)$

$= 3^2\big(3T(n/4^3)+n/4^2\big) + n(1+3/4) = 3^3T(n/4^3) + n(1+3/4+3^2/4^2) =$

$\ldots$

$= 3^iT(n/4^i) + n(1+3/4+3^2/4^2+\ldots+3^{i-1}/4^{i-1})$

הרקורסיה מגיעה לתנאי העצירה כאשר $n/4^i=1$, כלומר $i = \log_4 n$.

$$T(n) = 3^{\log_4 n} T(1) + n \cdot \sum_{k=0}^{\log_4 n - 1} \left(\frac{3}{4}\right)^k$$

$$\leq 3^{\log_4 n} \cdot \Theta(1) + n \cdot \sum_{k=0}^{\infty} \left(\frac{3}{4}\right)^k = \Theta(n^{\log_4 3}) + 4n = o(n) + 4n$$

ובכך מצאנו חסם עליון: $T(n) = O(n)$.

בנוסף גם $T(n) = 3T(n/4) + n > n$ ולכן $T(n) = \Omega(n)$.

סה"כ קיבלנו: $T(n) = \Theta(n)$.

פתרון 2

$$t(n)= n^3 + t(n/3) + \mathrm{t}(n/4)$$

חסם עליון: $t(n) \le n^3 + 2t(n/3)$

חסם תחתון: $t(n) \ge n^3 + 2t(n/4)$

בשני המקרים, ניתן לפתור לפי משפט האב (מקרה 3).

שני החסמים מתלכדים, ומקבלים $t(n) = \Theta(n^3)$.

פתרון 3

```
BINARY-SEARCH1-2(A, p, r, v)
if p > r
   then return NIL
q ← p + ⌊(r − p + 1)/3⌋
if v < A[q]
    then return BINARY-SEARCH1-2(A, p, q − 1, v)
    else if v > A[q]
        then return BINARY-SEARCH1-2(A, q + 1, r, v)
        else return q
```

ניתוח סיבוכיות

המקרה הגרוע הוא כאשר האיבר שמחפשים לא נמצא במערך, וגם בכל שלב ברקורסיה פונים לחלק הגדול יותר (בגודל שני שלישים בערך). במקרה זה:

$$t(n) = t(2n/3) + 1$$

ניתן לפתור לפי משפט האב (מקרה 2), או בשיטת האיטרציות, ומקבלים $t(n) = \Theta(\log n)$.