

# מבני נתונים ויעילות אלגוריתמים

(18.1.2015)

## 1. טענה אודות סדר גודל

טענה: מתקיים  $\log_2(n!) = \Theta(n \log n)$ .

הוכחה:

ברור שמתקיים:

$$1 \cdot 2 \cdot 3 \cdot 4 \cdot \dots \cdot n \leq \underbrace{n \cdot n \cdot n \cdot \dots \cdot n}_n$$

במילים אחרות:

$$n! \leq n^n$$

נוציא לוגריתם משני האגפים:

$$\log(n!) \leq \log(n^n)$$

נפעיל את הכלל  $\log(a^b) = b \cdot \log(a)$  על אגף ימין של האי-שוויון, ונקבל:

$$\log(n!) \leq n \cdot \log(n)$$

מכאן נובע שקיים החסם העליון:  $\log_2(n!) = O(n \log n)$ . נותר להראות שקיים גם חסם תחתון  $\log_2(n!) = \Omega(n \log n)$ , ומזה ינבע החסם ההדוק שרצינו להראות.

כזכור, קירוב סטירלינג קובע ש:  $n! = \sqrt{2 \cdot \pi \cdot n} \cdot \left(\frac{n}{e}\right)^n \cdot \left(1 + \Theta\left(\frac{1}{n}\right)\right)$ . מכאן ש:

$$n! \geq \sqrt{2 \cdot \pi \cdot n} \cdot \left(\frac{n}{e}\right)^n$$

נוציא לוגריתם משני האגפים:

$$\log(n!) \geq \log\left(\sqrt{2 \cdot \pi \cdot n} \cdot \left(\frac{n}{e}\right)^n\right)$$

נפעיל על אגף ימין של אי-השוויון את הכלל  $\log(x \cdot y) = \log(x) + \log(y)$ , ונקבל:

$$\log(n!) \geq \log(\sqrt{2 \cdot \pi \cdot n}) + \log\left(\left(\frac{n}{e}\right)^n\right)$$

להוציא שורש ריבועי – שקול להעלאה בחזקת חצי :

$$\log(n!) \geq \log((2 \cdot \pi \cdot n)^{0.5}) + \log\left(\left(\frac{n}{e}\right)^n\right)$$

נשתמש פעמיים בכלל  $\log(a^b) = b \cdot \log(a)$  :

$$\log(n!) \geq \frac{1}{2} \cdot \log(2 \cdot \pi \cdot n) + n \cdot \log\left(\frac{n}{e}\right)$$

נשתמש בכלל  $\log\left(\frac{x}{y}\right) = \log(x) - \log(y)$  באגף ימין של האי-שוויון :

$$\log(n!) \geq \frac{1}{2} \cdot \log(2 \cdot \pi \cdot n) + n \cdot \log(n) - n \cdot \log(e)$$

באגף ימין של האי-שוויון יש שלושה מחוברים – השמאלי מביניהם הוא מסדר גודל לוגריתמי, והימני מביניהם הוא מסדר גודל לינארי. המחובר האמצעי הוא הדומיננטי מבין השלושה, וסדר הגודל שלו הוא  $\Theta(n \log n)$ . המשמעות של האי-שוויון הזה, אם כך, היא ש- $\log_2(n!) = \Omega(n \log n)$ , וסיימנו.

מ.ש.ל

### תרגיל

חשבו את סיבוכיות זמן הריצה של כל אחד מקטעי הקוד הבאים :

א.

```
for (i = 1; i <= n; i++)
    for (j = 1; j <= log(n); j++)
        S1;
```

ב.

```
for (i = 1; i <= n; i++)
    for (j = 1; j <= log10(n); j++)
        S1;
```

.ג

```
for (i = 1; i <= n; i++)
    for (j = 1; j <= log10(i); j++)          /* שימו לב */
        S1;
```

.ד

```
for (i = 1; i <= n; i++) {
    j = i;
    while (j > 0) {
        S1;
        j = j / 2;
    }
}
```

## 2. חסם תחתון על זמן ריצה של מיוני השוואה

כל אלגוריתמי המיון שראינו עד כה, הן הפחות יעילים מביניהם (מיון הכנסה, מיון בחירה ומיון בועות), והן היעילים יותר (מיון-מיזוג ומיון עץ) סיבוכיותם הייתה  $\Omega(n \log n)$ .

מסתבר, שאין זה צירוף מקרים שלא הצלחנו למצוא אלגוריתם שסיבוכיותו טובה יותר, שכן זהו חסם תחתון עבור כל אלגוריתם מיון המבוסס על השוואות. אלגוריתמי מיון כאלה מכונים מיוני השוואה (comparison sorts). כל אלגוריתמי המיון שראינו עד כה היו מיוני השוואה.

במיון השוואה משתמשים אך ורק בהשוואות בין איברים כדי לקבל מידע על סדר האיברים. כלומר, בהינתן שני איברים  $a_i$  ו- $a_j$ , כדי לקבוע מהו הסדר שלהם, אנחנו מבצעים אחת מהבדיקות  $a_i < a_j$ ,  $a_i \leq a_j$ ,  $a_i = a_j$ ,  $a_i > a_j$ , או  $a_i \geq a_j$ . איננו רשאים לבחון את ערכי האיברים או להשיג מידע על יחסי הסדר ביניהם בדרך אחרת.

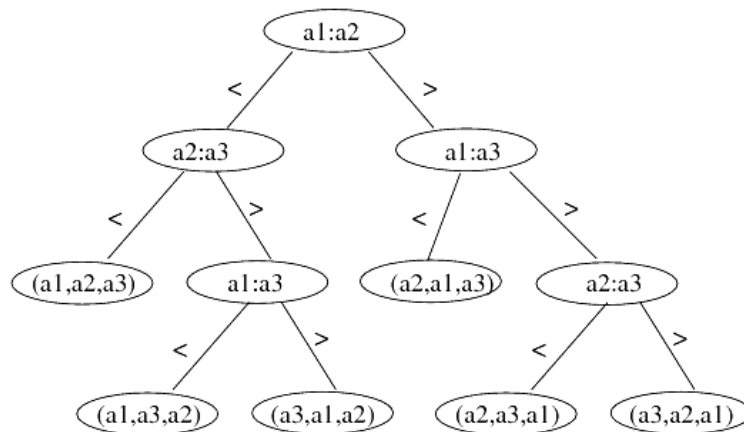
נניח, בלי הגבלת הכלליות, שכל האיברים במערך הקלט שונים זה מזה. בהינתן הנחה זו, לא נזדקק להשוואות מהצורה  $a_i = a_j$ .

כמו כן, תחת ההנחה שאין שני איברים שווים, לא יהיה הבדל בין השוואה מהצורה  $a_i < a_j$  לבין השוואה מהצורה  $a_i \leq a_j$ , או בין השוואה מהצורה  $a_i > a_j$  לבין השוואה מהצורה  $a_i \geq a_j$ . לפיכך, נוכל להניח כי לא מתבצעות השוואות מהסוג  $a_i \leq a_j$  או מהסוג  $a_i \geq a_j$ .

לבסוף, נשים לב שהשוואות  $a_i < a_j$  ו- $a_i > a_j$  שקולות זו לזו, במובן זה שהן מספקות מידע זהה על הסדר היחסי של  $a_i$  ו- $a_j$ . לפיכך, נניח שכל השוואות הן מהצורה  $a_i < a_j$ .

### עצי החלטה

ניתן לתאר מיוני השוואה באופן מופשט, באמצעות עצי החלטה (decision trees). עץ החלטה מייצג את ההשוואות שמבצע אלגוריתם מיון כשהוא פועל על קלט בגודל נתון. נביט בעץ ההחלטה הבא, המתאר את פעולתו של אלגוריתם מיון, המבוסס על השוואות, על קלט של שלושה איברים:



באופן כללי, בהינתן סדרת קלט  $(a_1, a_2, \dots, a_n)$ , כל צומת פנימי בעץ ההחלטה יסומן  $a_i : a_j$ , בעבור  $1 \leq i, j \leq n$ . כל עלה מייצג תמורה (permutation) של  $n$  האיברים. ביצוע של אלגוריתם המיון מתאים למעבר לאורך מסלול, משורש עץ ההחלטה לאחד העלים. בכל צומת פנימי נערכת השוואה  $a_i < a_j$  – התת-עץ השמאלי מכתוב את ההשוואות הבאות שיערכו אם  $a_i < a_j$ , ואילו התת-עץ הימני מכתוב את ההשוואות הבאות שיערכו אם  $a_i > a_j$ . כאשר מגיעים אל עלה, אלגוריתם המיון קבע את סדר המיון.

### גובהו של עץ החלטה

ראינו שעבור עץ החלטה עם שלושה איברים, היו  $3! = 6$  עלים. כמו כן, ברור שכל הצמתים הם מדרגה 0 או מדרגה 2. זהו המצב גם באופן כללי: בעץ החלטה עבור מערך המכיל  $n$  איברים, יש  $n!$  עלים, ובעץ כזה כל הצמתים הם מדרגה 0 או 2.

לפי המשפט העיקרי על עצים בינאריים, מספר העלים גדול ב-1 ממספר הצמתים בעלי דרגה 2. מכיוון שיש  $n!$  עלים, נסיק שיש  $n!-1$  צמתים מדרגה 2. כמות הצמתים הכוללת בעץ שווה לסכום של כמות העלים ושל כמות הצמתים מדרגה 2 (כי בעץ החלטה אין צמתים מדרגה 1). לכן:

$$\text{כמות הצמתים בעץ} = n! + n! - 1 = 2n! - 1$$

כעת, ניזכר במסקנה אותה הוכחנו, לפיה – בעץ בינארי בעל  $x$  צמתים, הגובה  $h$  מקיים:  $h \geq \log_2\left(\frac{x+1}{2}\right)$ . נציב  $x = 2n! - 1$ , ונקבל:

$$h \geq \log_2\left(\frac{x+1}{2}\right) = \log_2\left(\frac{2n!-1+1}{2}\right) = \log_2\left(\frac{2n!}{2}\right) = \log_2(n!)$$

מתקיים  $\log_2(n!) = \Theta(n \log n)$ , ולכן  $h = \Omega(n \log n)$ .

כשמבצעים מיון בעזרת השוואות, גובהו של עץ ההחלטה – דהיינו, אורכו של המסלול הארוך ביותר מהשורש של העץ לאחד מעליו – מייצג את מספר ההשוואות שאלגוריתם המיון עורך במקרה הגרוע ביותר (W.C – Worst Case).

מכיוון שהוכחנו כי הגובה  $h$  של כל עץ החלטה מקיים  $h = \Omega(n \log n)$ , נסיק שזהו חסם תחתון לסיבוכיות זמן הריצה, במקרה הגרוע ביותר (W.C) של מיון, במודל השוואות.

## תרגיל

האם קיים אלגוריתם המבוסס על השוואות, שזמן ריצתו הוא לינארי (W.C), אשר בהינתן מערך קלט, בונה ממנו עץ חיפוש בינארי? אם כן – כתבו את האלגוריתם, ואם לא – הוכיחו שאלגוריתם כזה לא יכול להתקיים.