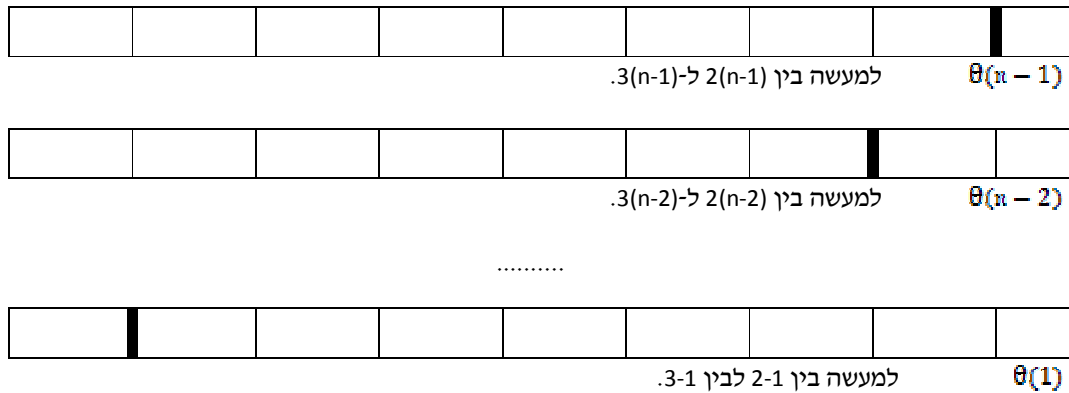


**חזרה והמשך על Selection Sort**

- נבדוק את הסיבוכיות של הריצה לכל שלב ב-Selection Sort (כאשר ממקמים את המספר המקסימלי מימין לקו, ועוברים על האיברים משמאלו על-מנת למצוא את המספר המקסימלי ביניהם (כלומר החלק מימין לקו תמיד ממויין כבר).
- קיימים לכל מעבר 3 או 2 פעולות כי בודקים באמצעות 2 פעולות את היחס בין המספרים, ובמידה ויש צורך בהחלפת מספר, אנחנו מוסיפים פעולה שלישית.



**ניתוח יעילות**

- נסמן ב- $T_i(n)$  את מספר הפעולות שמתבצעות בשלב ה- $i$  של Selection Sort כאשר ממיינים מערך בין  $n$  איברים:

$$\theta(n - i) T_i(n) = \Leftarrow 2(n-i) \leq T_i(n) \leq 3(n-i)$$

- לפיכך, מספר הפעולות הכולל שמתבצע במהלך Selection Sort עם מערך בין  $n$  איברים הוא:

$$\leq 3((n - 1) + (n - 2) + \dots + 1) T_1(n) + T_2(n) + \dots + T_{n-1}(n) \leq 2((n - 1) + (n - 2) + \dots + 1)$$

או ... (סכום סדרה חשבונית):

$$\leq 3(n - 1)n * \frac{1}{2} \quad 2(n - 1)n * \frac{1}{2} \leq$$

- ולפיכך מספר הפעולות הכולל הוא:

$$\theta(n^2) \quad \theta\left((n - 1)n * \frac{1}{2}\right)$$

- באופן כללי יותר, מתקיים שאם:

$$g_1 = \theta(f_1), \quad g_2 = \theta(f_2) \quad g_3 = \theta(f_3)$$

אז:

$$g_1 + g_2 + g_3 = \theta(f_1 + f_2 + f_3)$$

▪ **הסבר:** קיימים קבועים חיוביים:  $\overline{c_1}, \overline{c_2}, \overline{c_3}, c_1, c_2, c_3$ , כך שלכל  $n$ :

$$\min\{\overline{c_1}, \overline{c_2}, \overline{c_3}\} * (f_1(n) + f_2(n) + f_3(n)) \leq g_1(n) + g_2(n) + g_3(n) \\ \leq \max\{\overline{c_1}, \overline{c_2}, \overline{c_3}\} * (f_1(n) + f_2(n) + f_3(n))$$

**Insertion Sort** •

○ **תיאור מילולי:**

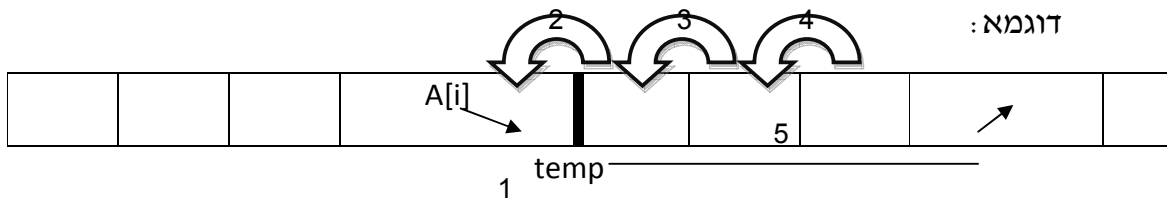
בכל שלב מזיזים את האיבר הימני ביותר בחלק הלא ממויין ימינה עד למקומו הנכון (זמנית) בחלק הממויין.

○ **Psuedo-Code:**

```
Insertion Sort ( A[ 0 ... n-1 ] )
  For i <--- n-2 downto 0
    Do j <--- i+1
      While A[j] < A[j-1] and j <= n-1
        Do exchange A[j] <---> A[j-1]
        j <--- j+1
```

○ **וריאציות:**

1. לבצע את החיפוש של המיקום הנכון הזמני בחלק הממויין של המערך במצעות חיפוש בינארי.
2. במקום למקם את האיבר החדש במקומו הנכון ע"י סדרת החלפות של איברים סמוכים (פעפוע), ראשית יש למצוא את המיקום החדש, ואז לבצע את כל ההזזות (עבור האיברים משמאל למיקום החדש). וריאציה זו חוסכת את הצורך לבדוק את היחס בין כל איבר ואיבר, וכך בעצם במקום 3 פעולות לכל העברה, אנחנו מבצעים רק אחת (ההזזה עצמה).



○ **Psuedo-Code של וריאציה 2:**

```
Insertion Sort II ( A [ 0 ... n-1 ] )
  For i <--- n-2 downto 0
    Do temp <--- A[i]
```

```

j <--- i + 1
while A[j] < temp and j <= n-1
  do A[j-1] <--- A[j]
  j <--- j+i
A[j-1] <--- temp

```

### ○ ניתוח יעילות

- באיטרציה בודדת של לולאת ה-For (איטרציה i) תתבצענה בין 0 ל-n-i איטרציות של לולאת ה-while, ובכל אחת מהן  $\theta(1)$  פעולות.
- לכן, איטרציה i תתבצענה  $O(n-i)$  פעולות, ובסה"כ  $O(n^2)$  פעולות.
- כלומר ב-"worst case", אין שיפור בהשוואה ל-Selection Sort – מספר הפעולות ב-Selection **תמיד**  $\theta(n^2)$  ומספר הפעולות ב-Insertion Sort תמיד  $O(n^2)$  (ועבור קלטים מסויימים ממש פחות).

▪ כמה פעולות השוואה בלבד מתבצעות?

- ב-Selection בדיק  $\theta\left(n \cdot \frac{1}{2} (n-1)\right)$  כלומר  $\theta(n^2)$ .
- ב-Insertion בגרסאות שניתחנו (המקורית ו-וריאציה 2), במקרה הגרוע ביותר  $\theta\left(n \cdot \frac{1}{2} (n-1)\right)$  ובטוב ביותר n, אבל בוריאציה 1 (חיפוש בינארי), מספר ההשוואות יהיה:

$$\begin{aligned}
 & \lceil \log_2 1 \rceil + \lceil \log_2 2 \rceil + \lceil \log_2 3 \rceil + \dots + \lceil \log_2 n - 1 \rceil \\
 & < (n-1) * \lceil \log_2 (n-1) \rceil
 \end{aligned}$$

(למרות שמספר הפעולות הכולל עשוי להגיע לסדר גודל של  $n^2$ ).

### ○ טענה:

בכל אלגוריתם מיון, לצורך מיון מערך בן n איברים נחוצות  $\Omega(n * \log n)$  פעולות השוואה (בהסתייגות מסוימת שנסדר בהמשך) במקרה הגרוע ביותר.

### ○ הוכחה:

נתבונן באלגוריתם מיון כלשהו שהתחייבו שהוא מבצע לכל היותר K פעולות השוואה, כאשר הוא מטפל במיון מערך בגודל n. מספר "תסריטי" הריצה של האלגוריתם הוא לכל היותר  $2^k$ . כדי שהאלגוריתם אכן יצליח למיין כל מערך בגודל n, הכרחי שיהיו לו

לפחות  $n!$  תסריטי ריצה שונים (כדי להתמודד עם  $n!$  הדרכים השונות לבלגן  $n$  מספרים שונים). אם כן, אז בהכרח:  $n! \leq 2^k \leftarrow$

$$\begin{aligned} k \geq \log_2 n! &= \log_2 1 + \log_2 2 + \dots + \log_2 n \\ &\geq \log_2 \left(\frac{n}{2}\right) + \log_2 \left(\frac{n}{2} + 1\right) + \dots + \log_2 n \geq \left(\frac{n}{2}\right) * \log_2 \left(\frac{n}{2}\right) \\ &= \left(\frac{n}{2}\right) * \log_2 (n - 1) \geq \left(\frac{n}{2}\right) * \left(\frac{1}{2}\right) * \log_2 (n) = \left(\frac{1}{4}\right) * n * \log_2 n \end{aligned}$$

$n \geq 4$  עבור

### ○ הערות:

1. כמסקנה מהטענה נקבל שמסר הפעולות הכולל (לא רק ההשוואות) במקרה הגרוע של כל אלגוריתם מיון הוא  $\theta(n * \log n)$ .
2. אכן יש אלגוריתמי מיון (שנראה בשיעור הבא) שמספר הפעולות הכולל שלהם במקרה הגרוע הוא  $\theta(n * \log n)$  (לכאורה בניגוד לטענה, אך לא בניגוד להסתייגות שנדבר עליה בהמשך).
3. אם האלגוריתם לא אמור לטפל במערך כללי של  $n$  מספרים ממשיים אלא רק במערכים של  $n$  מספרים שידוע עבורם מידע נוסף (למשל, שלמים וחסומים ע"י מספר מסויים), אז ניתן לקבל ביצועים טובים יותר ו-"לשבור" את מחסום ה- $\theta(n * \log n)$ .