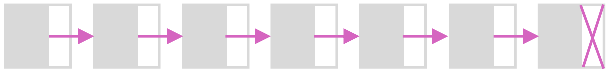
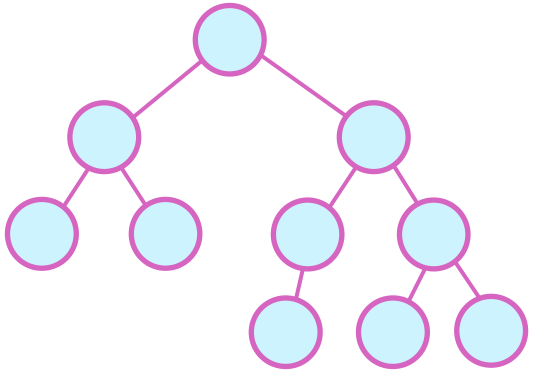
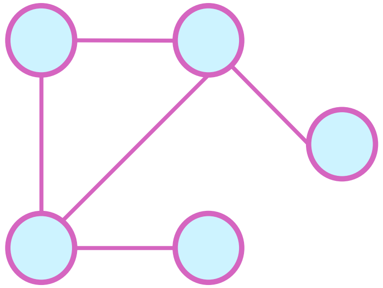
ניתוח אסימפטוטי של זמני ריצה של האלגוריתמים

מבנה נתונים

- הגדרה מבנה נתונים הוא דרך לארגון נתונים במחשב כדי להקל על גישה ושינוים







אלגוריתם

הגדרה אלגוריתם הוא תהליך חישובי מוגדר היטב, המקבל ערך (או קבוצת ערכים) כקלט, ומפיק ערך (או קבוצת ערכים) כפלט

דוגמאות

1. **מתכון לעוגה**

קלט מצרכים (ביצים, סוכר, קמח, ...)
פלט עוגה

2. **מיון מספרים**

קלט סדרה של n מספרים
פלט סדרה של n מספרים הקלט ממוינים בסדר לא יורד

41 5 6 18 9

5 6 9 18 41

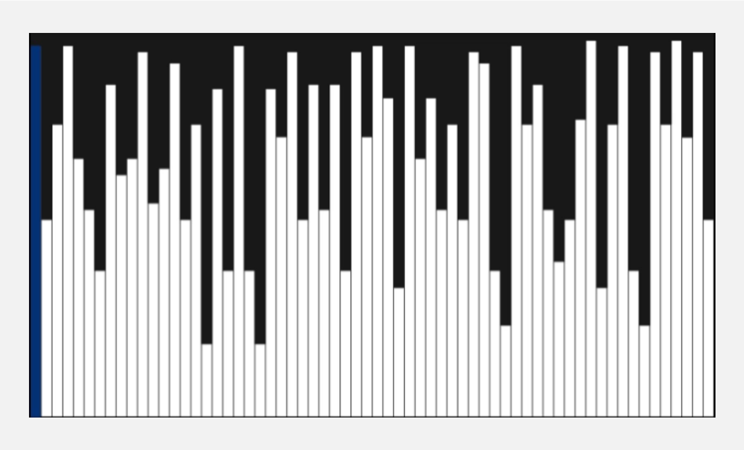
ניתוח אלגוריתמים

- השאלה המעניינת:
  עד כמה האלגוריתם מורכב? כלומר, נרצה לחשב ולהעריך את כמות המשאבים שהאלגוריתם דורש
  - זמן ריצה
  - גודל זכרון
  - רוחב פס התקשורת
  - ועוד
- ניתוח מספר אלגוריתמים אפשריים לפתרון בעיה מסוימת מאפשר את מציאת האלגוריתם היעיל ביותר מביניהם
- בחירה של מבנה נתונים הוא שלב חשוב בפיתוח אלגוריתם

זמן ריצה של האלגוריתם

- זמן ריצה גדל כשקלט גדל
  - מיון מערך בגודל 1,000,000 לוקח יותר זמן ממיון מערך בגודל 100
  - אנחנו מעוניינים לדעת איך משתנה זמן ריצה ביחס לגודל הקלט

100



1,000,000

זמן ריצה של האלגוריתם

- **גודל הקלט**
  - בבעיות רבות מספר הפריטים, n, בקלט (מיון לדוגמה)
  - בהתאם לבעיה יכולים להשתמש במדד אחר
- **זמן ריצה**
  - זמן ריצה של אלגוריתם הוא פונקציה של גודל הקלט, n
  - סימון $T(n)$

שאלה

האם זמן ריצה מושפע מבחירת שפת תכנות?

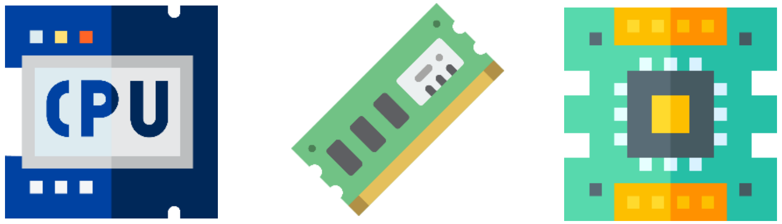
כן

לא

לא ניתן לדעת

זמן ריצה של האלגוריתם

- לא נוכל לנתח מורכבות של אלגוריתמים באמצעות הרצתם בפועל
  - על איזו חומרה נריץ?
  - איזו תוכנה נבחר?



- לא נוכל למדוד זמן ריצה על כל קלט אפשרי
  - לא פשוט להסיק מסקנות מהרצות בודדות
- כדי להשוות בין שני אלגוריתמים, נדרש יהיה להשוותם בתנאים זהים

הפתרון

- שיטה כללית לניתוח זמן ריצה של האלגוריתם
  - לקחת בחשבון כל קלט אפשרי
  - ניתוח שאינו תלוי בחומרה ותוכנה
  - ניתוח לפי תיאור האלגוריתם "ברמה גבוהה" בלי צורך לממש אותו בשפת התכנות
    - פסאודו קוד (Pseudo-Code) הוא תיאור מופשט לאלגוריתם, שמיועד לקריאה על ידי בני אדם

מיון הכנסה (Insertion-Sort)

- **נתון:** מערך A לא ממוין

	1	2	3	4	5	6	7	8	9	10
A	10	4	61	3	1	41	5	6	18	9

- **מטרה:** למיין את A

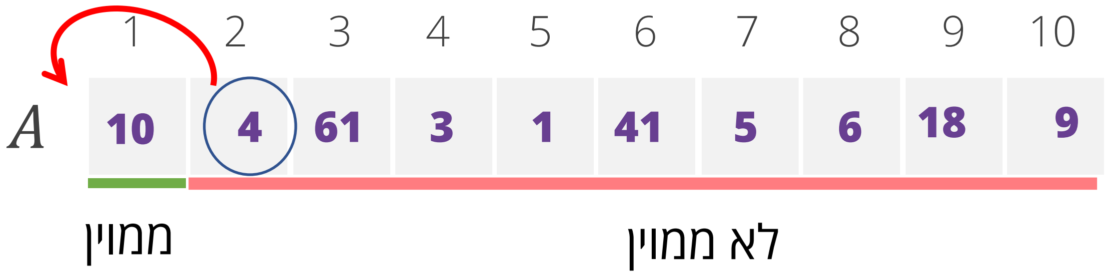
מיון הכנסה (Insertion-Sort)

חלק ממוין	חלק לא ממוין

איתחול

	1	2	3	4	5	6	7	8	9	10
A	**10**	**4**	**61**	**3**	**1**	**41**	**5**	**6**	**18**	**9**

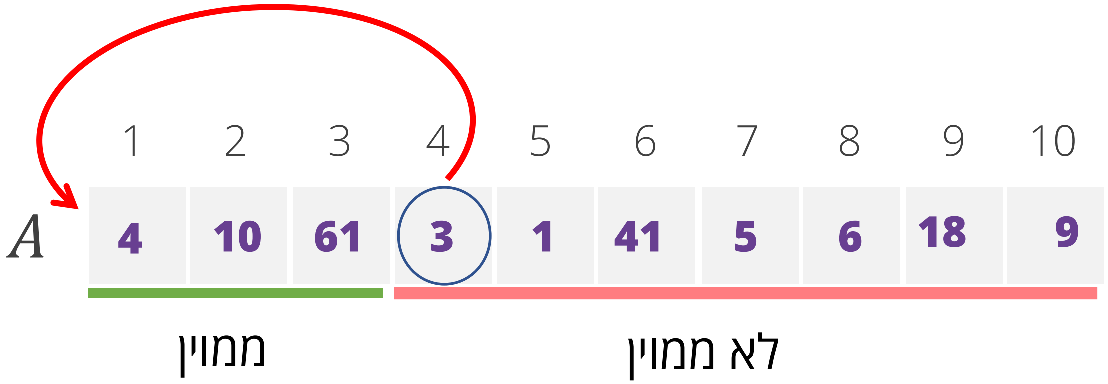
ממוין | לא ממוין

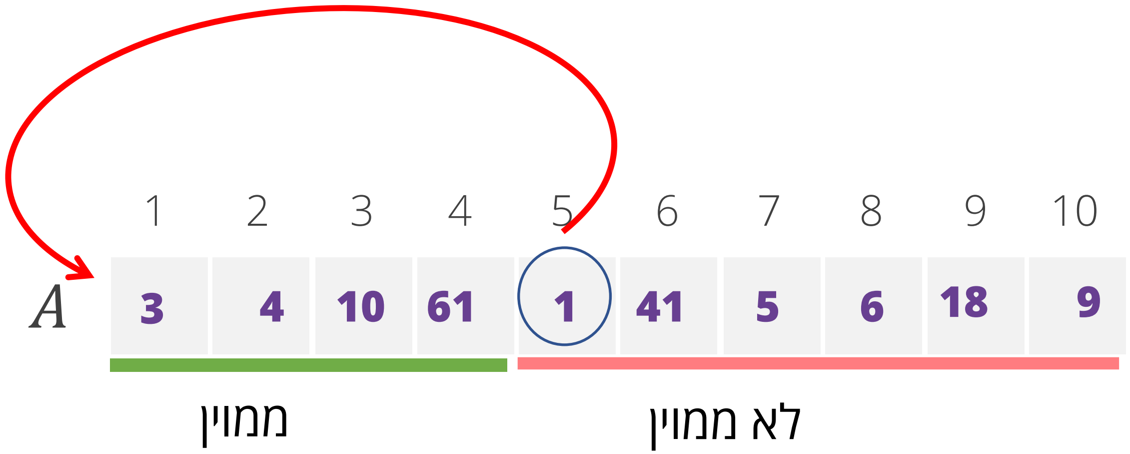

1 2 3 4 5 6 7 8 9 10
A
10 4 61 3 1 41 5 6 18 9
ממוין
לא ממוין

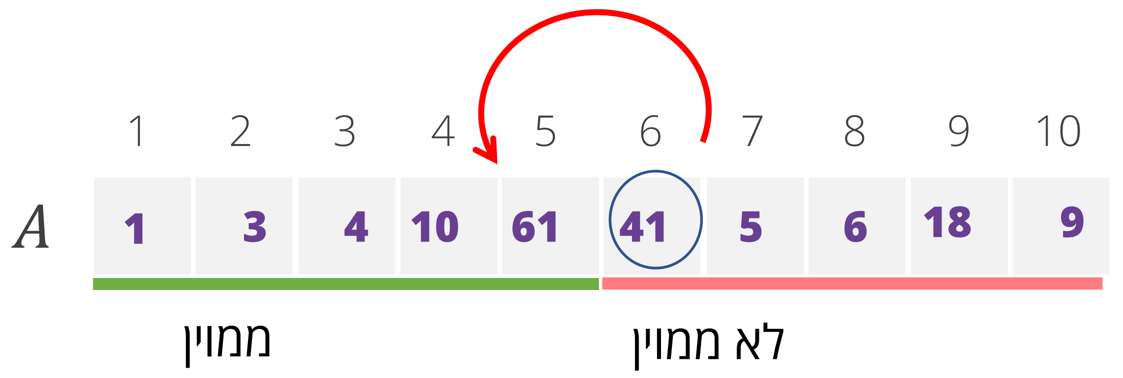
	1	2	3	4	5	6	7	8	9	10
A	**4**	**10**	**61**	**3**	**1**	**41**	**5**	**6**	**18**	**9**

ממוין

לא ממוין


1 2 3 4 5 6 7 8 9 10
A
4 10 61 3 1 41 5 6 18 9
ממוין
לא ממוין


1 2 3 4 5 6 7 8 9 10
A
3 4 10 61 1 41 5 6 18 9
ממוין
לא ממוין


1 2 3 4 5 6 7 8 9 10
A
1 3 4 10 61 41 5 6 18 9
ממוין
לא ממוין

	1	2	3	4	5	6	7	8	9	10
A	**1**	**3**	**4**	**5**	**6**	**9**	**10**	**41**	**61**	**81**

ממוין

מיון הכנסה (Insertion Sort)

INSERTION_SORT($A[1..n]$)

Input: an array A of size n

Output: array A sorted in non-decreasing order

```
for j ← 2 to n
        key ← A[j]
        // Insert key into the sorted part A[1..j − 1]
        i ← j − 1
        while i > 0 and A[i] > key
                A[i + 1] ← A[i]
                i ← i − 1
        A[i + 1] ← key
```

	i	j								
	1	2	3	4	5	6	7	8	9	10
A	10	4	61	3	1	41	5	6	18	9

שאלת POP UP

נתבונן במערך הבא בגודל 5:

7	4	6	9	3

איך ייראה המערך לאחר כל איטרציה של לולאת for (משמאל לימין) ?

1	3 7 6 9 7	3 4 6 9 7	3 4 6 9 7	3 4 6 7 9
2	4 7 6 9 3	4 6 7 9 3	4 6 7 9 3	3 4 6 7 9
3	4 7 6 3 9	4 6 3 7 9	4 3 6 7 3	3 4 6 7 9
4	4 7 6 3 9	6 4 3 7 9	4 3 6 7 9	3 4 6 7 9

זמן ריצה כפונקציה של מספר הפעולות

- פעולת אלמנטרית/יסוד/אטום/בסיסית
  - השמה
  - פעולה מתמטיות (למשל +, -)
  - השוואה (למשל >, <)
  - פעולה לוגית (למשל AND, NOT)
  - חזרה מפונקציה
  - גישה לאיבר בתוך המערך

- זמן ריצה של האלגוריתם הוא פונקציה של מספר פעולות שהאלגוריתם מבצע שתלויה בגודל הקלט, n

זמן ריצה של מיון הכנסה

INSERTION_SORT($A[1..n]$)	**עלות**	**מס הפעמים**
1. **for** $j \leftarrow 2$ **to** n		
2 $key \leftarrow A[j]$		
3 //Insert $A[j]$ into the sorted part $A[1..j-1]$		
4 $i \leftarrow j - 1$		
5 **while** $i > 0$ **and** $A[i] > key$		
6 $A[i+1] \leftarrow A[i]$		
7 $i \leftarrow i - 1$		
8 $A[i+1] \leftarrow key$		

t_j מספר הפעמים שלולאת while מתבצעת עבור j נתון $(j = 2,\ldots,n)$

זמן ריצה של מיון הכנסה

INSERTION_SORT($A[1..n]$)	עלות	מס הפעמים
1. **for** $j \leftarrow 2$ **to** n	c_1	n
2 $key \leftarrow A[j]$	c_2	$n-1$
3 //Insert $A[j]$ into the sorted part $A[1..j-1]$	0	
4 $i \leftarrow j-1$	c_4	$n-1$
5 **while** $i > 0$ **and** $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] \leftarrow A[i]$	c_6	$\sum_{j=2}^{n}(t_j-1)$
7 $i \leftarrow i-1$	c_7	$\sum_{j=2}^{n}(t_j-1)$
8 $A[i+1] \leftarrow key$	c_8	$n-1$

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5\sum_{j=2}^{n} t_j + c_6\sum_{j=2}^{n}(t_j-1) + c_7\sum_{j=2}^{n}(t_j-1) + c_8(n-1)$$

זמן ריצה של מיון הכנסה

- מה יכול להיות $T(n)$?
  - מקרה **הטוב ביותר (best case)** – הלולאה הפנימית לעולם לא מתבצעת
  - מקרה **הגרוע ביותר (worst case)** – הלולאה הפנימית מתבצעת מקסימום פעמים

שאלה

INSERTION_SORT($A[1..n]$)

Input: an array A of size n

Output: A sorted in non-decreasing order

1. **for** $j \leftarrow 2$ **to** n

2. $key \leftarrow A[j]$

3. // Insert key into the sorted sequence $A[1..j-1]$

4. $i \leftarrow j - 1$

5. **while** $i > 0$ **and** $A[i] > key$

6. $A[i+1] \leftarrow A[i]$

7. $i \leftarrow i - 1$

8. $A[i+1] \leftarrow key$

במקרה הטוב ביותר גוף של לולאת while לעולם לא מתבצע. איך נראה מערך הקלט במקרה זה?

1. מערך הקלט ממוין בסדר יורד
2. מערך הקלט ממוין בסדר עולה
3. לא ניתן לדעת בוודאות

זמן ריצה של מיון הכנסה **המקרה הטוב ביותר**

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8\,(n-1)$$

- במקרה זה $t_j = 1$ לכל $j = 2,..,n$
- נציב $t_j = 1$ לנוסחה:

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1)$$

$$= an + b$$

כאשר a,b הם קבועים שתלויים ב-$c_1,\dots,\ c_8$

שאלה

INSERTION_SORT($A[1..n]$)

Input: an array A of size n

Output: A sorted in non-decreasing order

for $j \leftarrow 2$ **to** n

$\quad key \leftarrow A[j]$

$\quad$ // Insert key into the sorted sequence $A[1..j-1]$

$\quad i \leftarrow j - 1$

$\quad$ **while** $i > 0$ **and** $A[i] > key$

$\quad\quad A[i+1] \leftarrow A[i]$

$\quad\quad i \leftarrow i - 1$

$\quad A[i+1] \leftarrow key$

במקרה <u>הגרוע ביותר</u> גוף של לולאת while מתבצע מספר מקסימאלי של פעמים. איך נראה מערך הקלט במקרה זה?

1. מערך הקלט ממוין בסדר יורד
2. מערך הקלט ממוין בסדר עולה
3. לא ניתן לדעת בוודאות

זמן ריצה של מיון הכנסה **המקרה הגרוע ביותר**

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8\,(n-1)$$

- במקרה זה $t_j = j$ לכל $j = 2,..,n$
- נציב $t_j = j$ לנוסחה:

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \frac{(n+2)(n-1)}{2} + c_6 \frac{n(n-1)}{2} + c_7 \frac{n(n-1)}{2} + c_8(n-1)$$

$$= a'n^2 + b'n + c'$$

כאשר a',b',c' הם קבועים שתלויים ב-$c_1,\ldots,c_8$

שאלה

במקרה <u>הממוצע</u> מחצית מן האיברים ב- $A[1..j-1]$ קטנים מ- A[j], ומחצית גדולים ממנו, כלומר $t_j = \frac{j}{2}$. אם נפתח את הנוסחה לחישוב זמן הריצה במקרה הממוצע, איזה פונקציה תתקבל ?

1 פונקציה ליניארית

2 פונקציה ריבועית

3 פונקציה מערכית

$$\boxed{T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5\sum_{j=2}^{n} t_j + c_6\sum_{j=2}^{n}(t_j - 1) + c_7\sum_{j=2}^{n}(t_j - 1) + c_8\,(n-1)}$$

שאלה

מהן הסיבות לחישוב זמן ריצה במקרה הגרוע ביותר? סמנו את כל התשובות הנכונות.

1. זמן ריצה של האלגוריתם במקרה הגרוע ביותר מהווה חסם עליון על זמן הריצה עבור כל קלט אפשר.
2. בחלק מהאלגוריתמים המקרה הגרוע הינו שכיח.
3. לעתים זמן ריצה במקרה הממוצע זהה לזמן ריצה במקרה הגרוע.

ניתוח המקרה הגרוע

- זמן ריצה מהווה חסם עליון על זמן הריצה עבור כל קלט אפשר. אם נדע אותו, נדע שהאלגוריתם לעולם לא ירוץ זמן ארוך יותר.
- בחלק מהאלגוריתמים המקרה הגרוע הינו שכיח
- לעתים זמן ריצה במקרה הממוצע זהה לזמן ריצה במקרה הגרוע.
- במקרים מיוחדים נתעניין בזמן הריצה **במקרה הממוצע** או בזמן הריצה **הצפוי**

מיון הכנסה – ניתוח זמן ריצה

- זמן ריצה של מיון הכנסה במקרה הגרוע הוא $a'n^2 + b'n + c'$

מיון הכנסה – ניתוח זמן ריצה

- זמן ריצה של מיון הכנסה במקרה הגרוע הוא $a'n^2 + b'n + c'$
- שיעור הגידול (order of grows)
- נאמר של זמן ריצה של מיון הכנסה במקרה הגרוע גדל בקצב n^2.
- לניתוח כזה קוראים **ניתוח אסימפטוטי (asymptotic)** של זמן ריצה

סימונים אסימפטוטיים

סימון אסימפטוטי O

קצב הגדילה

- הגדרה של **קצב הגדילה** למעשה מפשטת את האופן בו אנחנו מסתכלים על אלגוריתמים ותוכניות

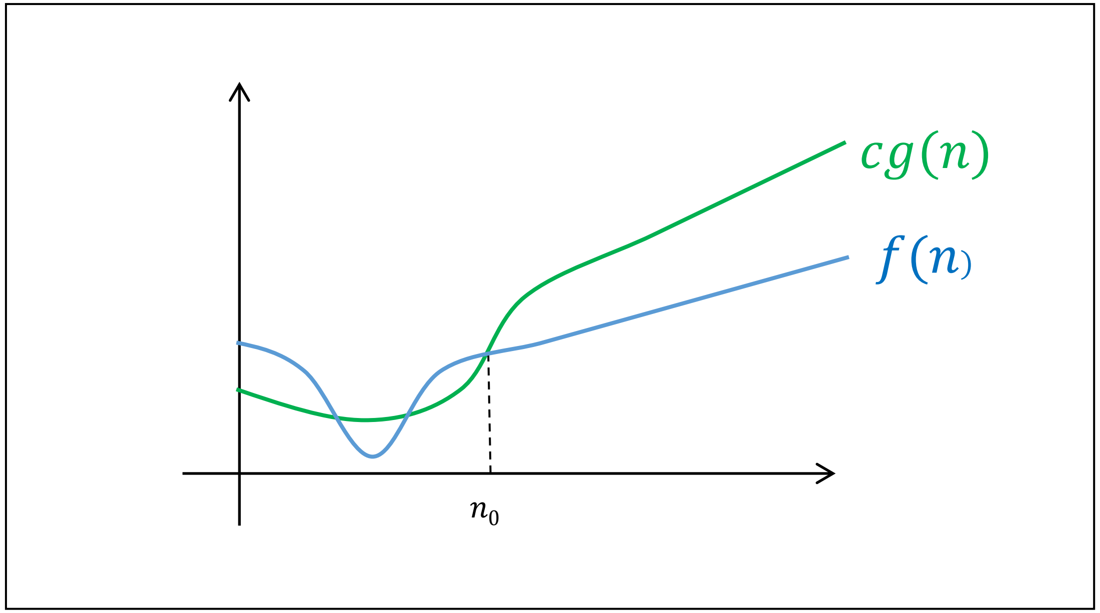
n	$\log n$	$\sqrt{n}$	n^2	2^n	4^n	$n!$	n^n
1	0	1	1	2	4	1	1
2	1	1.4	4	4	16	2	4
4	2	2	16	16	256	24	256
8	3	2.8	64	256	65,536	40,320	16,777,216
16	4	4	256	65,536	1,024	$\approx 3.09 \times 10^{13}$	$\approx 1.8 \times 10^{19}$
32	5	5.7	1,024	4,294,967,296	4,294,967,296	$\approx 2.63 \times 10^{35}$	$\approx 1.46 \times 10^{48}$
1,024	10	32	1,048,576	$\approx 1.79 \times 10^{38}$	$\approx 3.23 \times 10^{616}$	$\approx 5.41 \times 10^{2639}$	$\approx 3.52 \times 10^{3082}$

סימונים אסימפטוטיים

- **סימונים אסימפטוטיים** - הסימונים שאנו משתמשים בהם לתיאור זמן הריצה האסימפטוטי של אלגוריתם
- מוגדרים עבור פונקציות שתחום ההגדרה שלהן הוא קבוצת המפרים הטבעיים $\mathbb{N}$
  - גודל הקלט – מספר טבעי

סימון - O

$$O(g(n)) = \left\{ f(n): \begin{array}{r} \text{קיימים קבועים חיוביים } c \text{ ו-} n_0 \text{ כך ש-} \\ 0 \le f(n) \le cg(n) \text{ לכל } n \ge n_0 \end{array} \right\}$$



משמעות

"$f(n) \le g(n)$"

$$f(n) = O(g(n))$$

דוגמה 1

- הוכיחו ש- $4n + 3 = O(n^2)$

דוגמה 2

- הוכיחו ש- $an + b = O(n)$
- a,b קבועים, $a > 0$

דוגמה 3

- הוכיחו ש- $n\log n \neq O(n)$

דוגמה 4

- הוכיחו ש- $2^{10} = O(1)$

שאלה 1

סמנו את כל התשובות הנכונות

1. $3n + 8 = O(n)$
2. $3n + 8 = O(\log n)$
3. $3n + 8 = O(n^2)$
4. $3n + 8 = O(n \log n)$
5. $3n + 8 = O(\sqrt{n})$

שאלה 2

מהו זמן ריצה (במקרה הגרוע) של האלגוריתם חיפוש בינארי (Binary Search)?

1. $O(n)$
2. $O(\log n)$
3. $O(n^2)$
4. $O(n \log n)$
5. $O(\sqrt{n})$

שאלה 3

סדרו את הפונקציות הבאות מהגדולה לקטנה לפי סדרי גודל.

רשמו את תשובתכם כמחרוזת של אותיות במקום המיועד. לדוגמה, abdec.

$a.$ 2020

$b.$ $n^{5.2}$

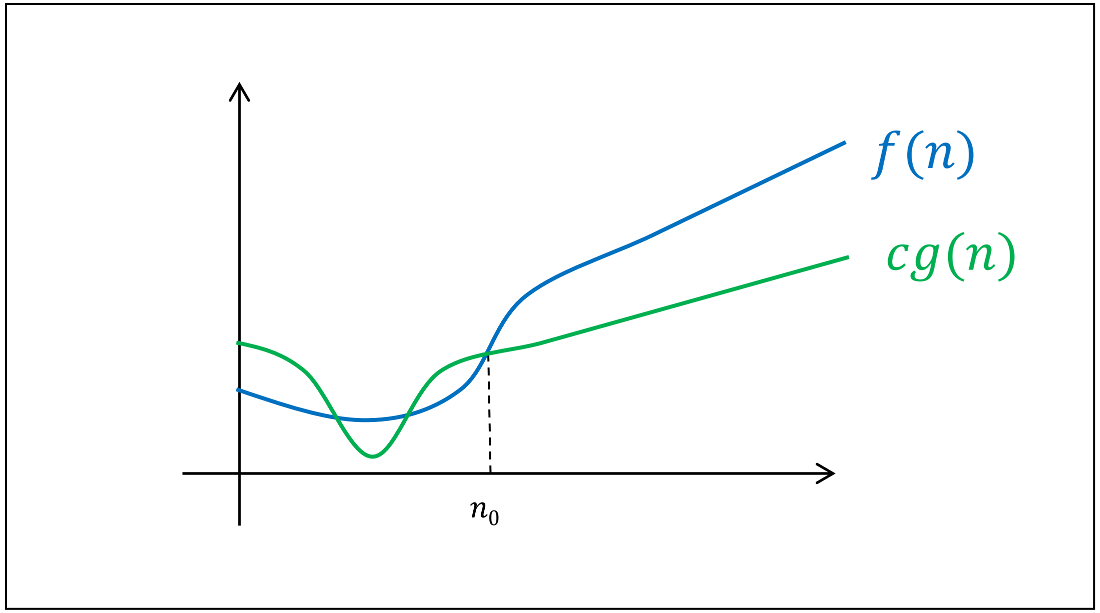
$c.$ 2^n

$d.$ 2^{2^n}

$e.$ $n^2 \log n$

סימון Ω-

$\Omega(g(n)$ $\{ f(n):$ קיימים קבועים חיוביים c ו-n_0 כך ש- $0 \le cg(n) \le f(n)$ לכל $n \ge n_0$ $\}$



משמעות

"$f(n) \geq g(n)$"

$$f(n) = \Omega(g(n))$$

דוגמה 1

- הוכיחו ש- $4n+3=\Omega(n)$

דוגמה 2

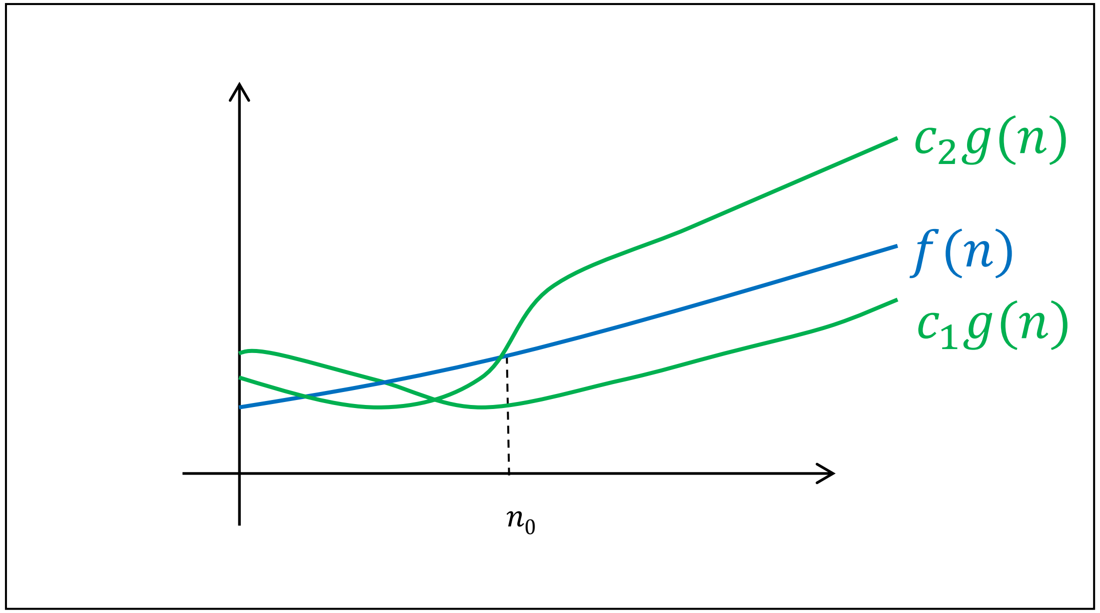
- הוכיחו ש- $(n-1)^2 = \Omega(n^2)$

דוגמה 4

- הוכיחו ש- $2^{10} = \Omega(1)$

סימון -Θ

$$\Theta(g(n)) = \left\{ f(n): \begin{array}{r} \text{קיימים קבועים חיוביים } c_1, c_2 \text{ ו-} n_0 \text{ כך ש-} \\ 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ לכל } n \ge n_0 \end{array} \right\}$$



משמעות

"$f(n) \approx g(n)$"

$$f(n) = \Theta(g(n))$$

דוגמה 1

- הוכיחו ש- $\frac{1}{2}n^2 - n = \Theta(n^2)$

משמעות – O, Ω, Θ

משמעות	סימון
$f(n) \le g(n)$	$f(n) = O(g(n))$
$f(n) \ge g(n)$	$f(n) = \Omega(g(n))$
$f(n) \approx g(n)$	$f(n) = \Theta(g(n))$

הערה: לא כל שתי פונקציות ניתנות להשוואה אסימפטוטית

דוגמה:

$$f(n) = n$$
$$g(n) = n^{2(n \bmod 2)}$$

משפט

לכל שתי פונקציות f(n) ו- g(n) מתקיים

$$f(n) = \Theta(g(n))$$

אם ורק אם

$f(n) = O(g(n))$ וגם $f(n) = \Omega(g(n))$

נובע ישירות מההגדרה של סימונים O, Ω, Θ