מבוא למדעי המחשב
Introduction to Computer Science

Lecture 8a

Complexity

הרצאה 8א

סיבוכיות

כמה זמן לוקח לחשב סכום של רשימה?

איך מודדים ביצועים של אלגוריתם?

זמן?

measure_sum.py ×

```python
import random
import timeit

def generate_random_ints(n):
    return [random.randint(0, n) for _ in range(n)]

lst = generate_random_ints(20)
time = timeit.timeit(
    lambda: sum(lst),
    number=10000
)
print(time) # 0.0009647
```

זה טוב? לא טוב?

הביצועים על קלט יחיד לא נותנים מספיק מידע

יותר מעניין: איך הביצועים משתנים כאשר כמות המידע גדלה?

איך מודדים ביצועים של אלגוריתם?

זמן?

measure_sum.py ×

```python
import random
import timeit

def generate_random_ints(n):
    return [random.randint(0, n) for _ in range(n)]

time = [0]*100
for i in range(1, 100):
    lst = generate_random_ints(i)
    time_i = timeit.timeit(
        lambda: sum(lst),
        number=10000
    )
    time[i]=time_i

print(time)
```

```
[0,
0.0007682000286877155,
0.0005425999406725168,
0.0005778000922873616,
0.0006636000471189618,
0.0005756000755354762,
0.0006232999730855227,
0.0006259001092985272,
0.0006594000151380897,
0.0006598000181883887,
0.0006896000122651458,
0.0008490999462082982,
0.0007486999966204166,
0.0007317999843508005,
0.0008289999095723033,
0.0008039000676944852,
0.0008288000244647264,
0.0008541999850422144,
0.0008666999638080597,
0.0008917000377550721,
0.0009617999894544482,
0.0009389000479131937,
0.0010174000635743141,
0.0009621999925002456,
0.0011297999881207943,
0.0011477000080049038,
0.0019256999948993325,
0.0012887999182567,
0.0012743000406771898,
0.0021883000154048204,
0.0013224999942034483,
0.0013067000545561314,
0.0013318000128492713,
0.0018001999706029892,
0.0019916000310331583,
0.0016953999875113368,
0.0013944000238552269,
0.0015122999902814627,
0.0014749999390915036,
0.0014534998917952218,
0.0014346999814733863,
0.0015022000297904015,
0.0019739000126719475,
0.0018144499963074922 6,
0.0018523000180721283,
0.0018553000409156084,
0.0016097000334411186,
0.0015662000514566898,
0.0016343999886885285,
0.0016314000822603703,
0.0016632999759167433,
0.0017179999267682433,
0.0017256999853998423,
0.0022037999005988824,
0.0017568999901413918,
0.0018667000113055 11,
0.0017844999674707651,
0.0018088000360876322,
0.0018180999990487 0987,
0.0021365999709814787,
0.0019164999248459935,
0.0019429000094532967,
0.0019257999956607819,
0.0019351999508216977,
0.0019372000824660063,
0.0019335000542923808,
0.0020509000169113 28,
0.0025731000350788236,
0.0020831000292673707,
0.0021877000108361244,
0.0021283000241965055,
0.0021175999427214265,
0.0021268000127747655,
0.0021573000121861696,
0.0021190999541431665,
0.0024518000427633524,
0.0023882999908488989,
0.0025714000022134185,
0.0022995000472292304,
0.0021740000022932887,
0.0022360999137163162,
0.0022974999915584922,
0.0024196000304073095,
0.0029476999977084 9943,
0.0023555200005695224,
0.0025241000112146 14,
0.0022711999481543 9,
0.0023153999354690313,
0.0024624000070616603,
0.0029134000651538 37,
0.0025584000395610 93,
0.0023277000291273,
0.0025741999270394444,
0.0025171000743284 82,
0.0030142000177875 16,
0.0027485999974385 48,
0.0027701000217348337,
0.0026307000080123544,
0.0027271000435575843]
```

```
0.0008039000676944852,
0.0008288000244647264,
0.0008541999850422144,
0.0008666999638080597,
0.0008917000377550721,
0.0009617999894544482,
0.0009389000479131937,
0.0010174000635743141,
0.0009621999925002456,
0.0011297999881207943,
0.0011477000080049038,
0.0019256999948993325,
```

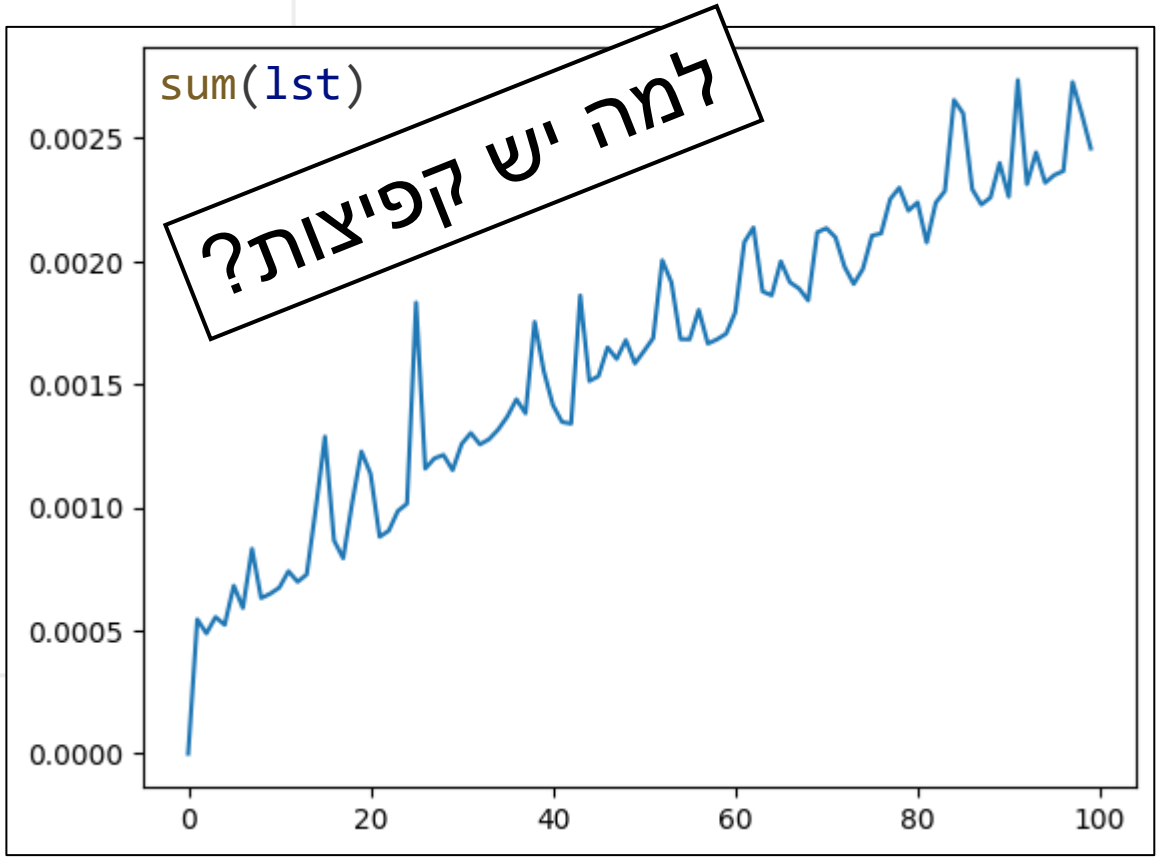
איך מודדים ביצועים של אלגוריתם?

זמן?

measure_sum.py

```python
import random
import timeit

def generate_random_ints(n):
    return [random.randint(0, n) for _ in range(n)]

time = [0]*100
for i in range(1, 100):
    lst = generate_random_ints(i)
    time_i = timeit.timeit(
        lambda: sum(lst),
        number=10000
    )
    time[i]=time_i

import matplotlib.pyplot as plt
plt.plot(time)
```

נראה שיש קשר לינארי
בין הזמן לגודל הרשימה



למה יש קפיצות?

איך מודדים ביצועים של אלגוריתם?

זמן?

measure_sum.py ×

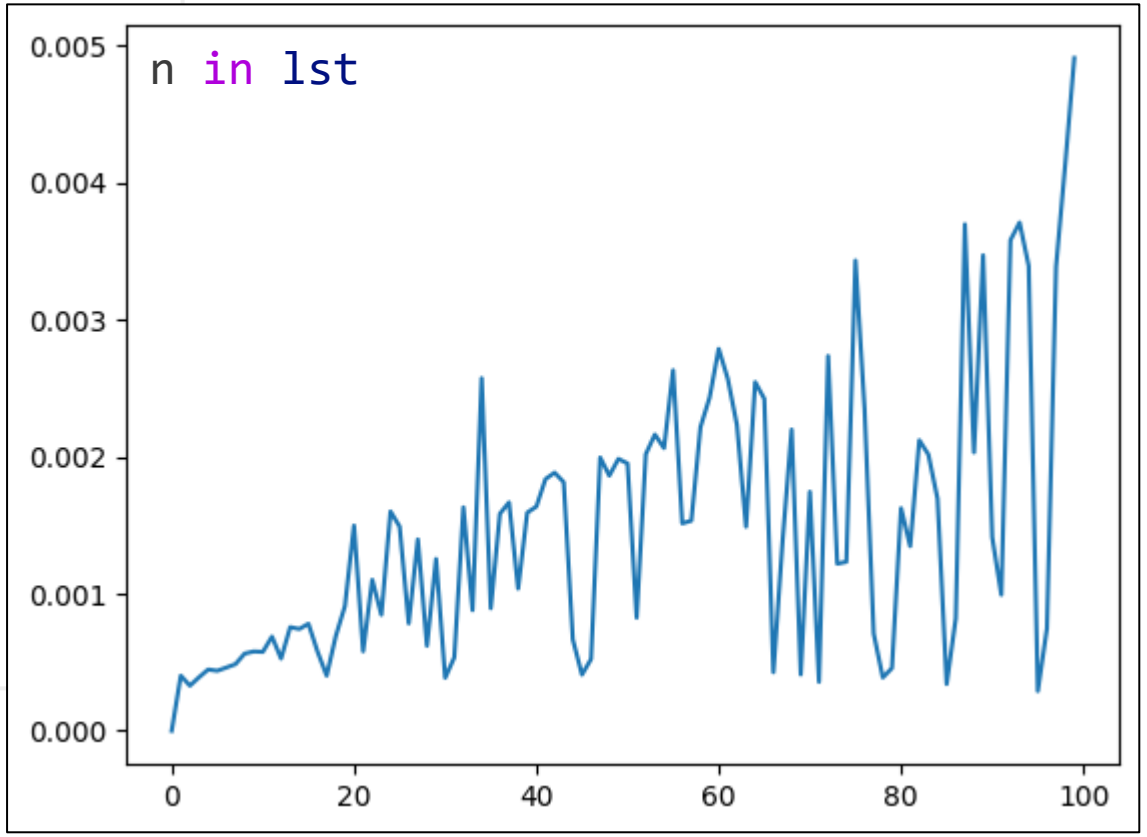
```python
import random
import timeit

def generate_random_ints(n):
    return [random.randint(0, n) for _ in range(n)]

time = [0]*100
for i in range(1, 100):
    lst = generate_random_ints(i)
    time_i = timeit.timeit(
        lambda: sum(lst),
        number=10000
    )
    time[i]=time_i

import matplotlib.pyplot as plt
plt.plot(time)
```

איך מודדים ביצועים של אלגוריתם?

זמן?

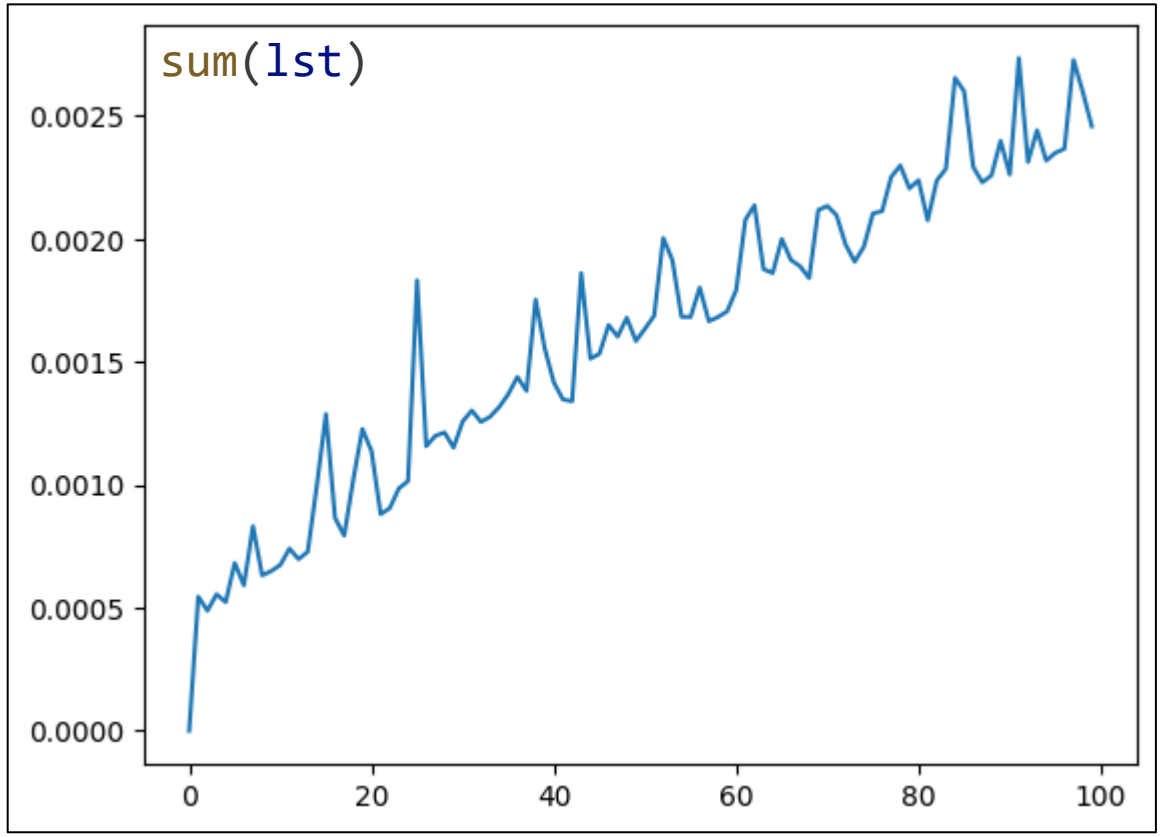
measure_sum.py ×

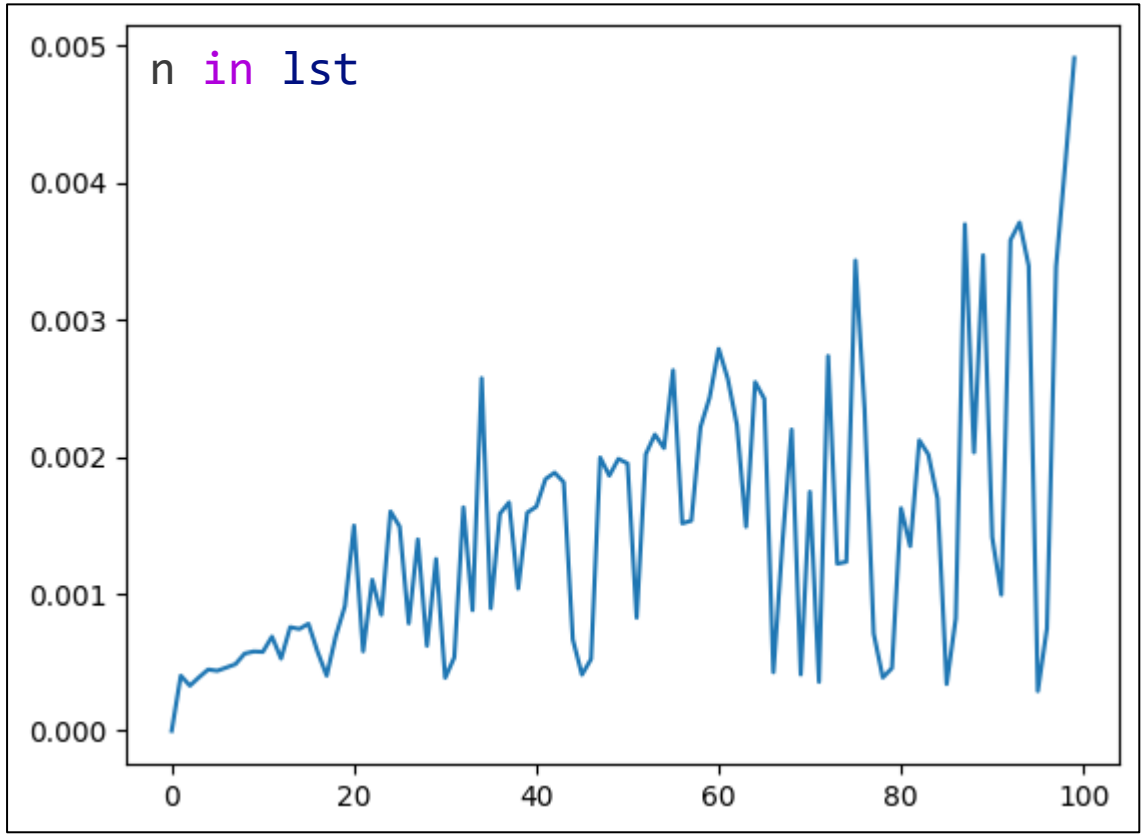
```python
import random
import timeit

def generate_random_ints(n):
    return [random.randint(0, n) for _ in range(n)]

time = [0]*100
for i in range(1, 100):
    lst = generate_random_ints(i)
    time_i = timeit.timeit(
        lambda: n in lst,
        number=10000
    )
    time[i]=time_i

import matplotlib.pyplot as plt
plt.plot(time)
```



איך מודדים ביצועים של אלגוריתם?
זמן?

למה יש הבדל בזמנים?





איך מודדים ביצועים של אלגוריתם?

בלי למדוד זמנים, איזה פונקציה לוקחת פחות זמן?

כמה פעולות מבצעים?

compare.py ×

```python
def print_stars_const(n):
    for i in range(100000):
        print("*")

def print_stars_linear(n):
    for i in range(n):
        print("*")

def print_stars_quadratic(n):
    for i in range(n):
        for j in range(n):
            print("*")
```

מספר פעולות לא תלוי ב-n

מספר פעולות לינארי ב-n

מספר פעולות ריבועי ב-n

סיבוכיות Complexity

מדד אבסטרקטי לביצועים של אלגוריתם

סיבוכיות זמן: מספר הפעולות הבסיסיות שמבצע אלגוריתם כתלות בגודל הקלט

סיבוכיות מקום: כמות הזיכרון שאלגוריתם דורש כתלות בגודל הקלט

איך נגדיר סיבוכיות?

Complexity

סיבוכיות

מדד אבסטרקטי לביצועים של אלגוריתם

איך נגדיר סיבוכיות?

מה חשוב לנו?

1. המקרה הגרוע ביותר
2. אי תלות במחשב
3. קצב גדילה עבור קלטים גדולים

סיבוכיות Complexity

מדד אבסטרקטי לביצועים של אלגוריתם

איך נגדיר סיבוכיות?

מה חשוב לנו?

1. המקרה הגרוע ביותר
2. אי תלות במחשב
3. קצב גדילה ~~עבור קלטים גדולים~~ אסימפטוטי

סימון O גדול Big O Notation

סימון O גדול

אינטואיציה

כמה פעולות מבצעים?

compare.py ×

```python
def print_stars_const(n):
    for i in range(10000):
        print("*")

def print_stars_linear(n):
    for i in range(n):
        print("*")

def print_stars_quadratic(n):
    for i in range(n):
        for j in range(n):
            print("*")
```

מספר פעולות לא תלוי ב-n

מספר פעולות לינארי ב-n

מספר פעולות ריבועי ב-n

סימון O גדול

אינטואיציה

כמה פעולות מבצעים?

compare.py ×

```python
def print_stars_const(n):
    for i in range(10000):
        print("*")

def print_stars_linear(n):
    for i in range(n):
        print("*")

def print_stars_quadratic(n):
    for i in range(n):
        for j in range(n):
            print("*")
```

$O(1)$

מספר פעולות לינארי ב-n

מספר פעולות ריבועי ב-n

סימון O גדול

אינטואיציה

כמה פעולות מבצעים?

compare.py ×

```python
def print_stars_const(n):
    for i in range(10000):
        print("*")

def print_stars_linear(n):
    for i in range(n):
        print("*")

def print_stars_quadratic(n):
    for i in range(n):
        for j in range(n):
            print("*")
```

$O(1)$

$O(n)$

מספר פעולות ריבועי ב-n

סימון O גדול

אינטואיציה

כמה פעולות מבצעים?

compare.py

```python
def print_stars_const(n):
    for i in range(10000):
        print("*")

def print_stars_linear(n):
    for i in range(n):
        print("*")

def print_stars_quadratic(n):
    for i in range(n):
        for j in range(n):
            print("*")
```

$O(1)$

$O(n)$

$O(n^2)$

סימון O גדול

אינטואיציה

כמה פעולות מבצעים?

compare.py ×

```python
def print_stars_const(n):
    for i in range(10000):
        print("*")

def print_stars_linear(n):
    for i in range(n):
        print("*")

def print_stars_quadratic(n):
    for i in range(n):
        for j in range(n):
            print("*")
```

$O(n)$

$O(n^2)$

compare.py ×

```python
def print_stars_const(n):
    for i in range(10000):
        print("*")
        print("*")
```

$O(1)$

$$O(1) = O(2) = O(c)$$

בסימון O גדול
מתעלמים מקבועים

סימון O גדול

אינטואיציה

כמה פעולות מבצעים?

compare.py

```python
def print_stars_const(n):
    for i in range(10000):
        print("*")

def print_stars_linear(n):
    for i in range(n):
        print("*")

def print_stars_quadratic(n):
    for i in range(n):
        for j in range(n):
            print("*")
```

$O(1)$

$O(n^2)$

compare3.py

```python
def print_stars_linear(n):
    for i in range(n+100):
        print("*")
```

$O(n+100) = O(n)$

מתעלמים מקבועים!

סימון O גדול

אינטואיציה

כמה פעולות מבצעים?

compare.py

```python
def print_stars_const(n):
    for i in range(10000):
        print("*")

def print_stars_linear(n):
    for i in range(n):
        print("*")

def print_stars_quadratic(n):
    for i in range(n):
        for j in range(n):
            print("*")
```

$O(n^2)$

compare4.py

```python
def print_stars_quadratic(n):
    for i in range(n):
        print("*")
    for i in range(n):
        for j in range(n):
            print("*")
```

$$O(n^2 + n) = O(n^2)$$

משאירים רק את
הגורם הכי גדול

סימון O גדול

אינטואיציה

כמה פעולות מבצעים?

תרגיל

compare5.py

```
def print_stars_4(n):
    m = 1
    for i in range(n):
        m *= 2
        for j in range(m):
            print("*")
```

$O(2^n)$

מה יהיה m באיטרציה האחרונה של i?

סימון O גדול

אינטואיציה

כמה פעולות מבצעים?

תרגיל

compare5.py

```python
def print_stars_4(n):
    m = 1
    for i in range(n):
        m *= 2
        for j in range(m):
            print("*")
```

$O(2^n)$

compare6.py

```python
def print_stars5(n):
    while n > 1:
        print("*")
        n = n // 2
```

n=2 -> *
n=3 -> *
n=4 -> **
n=8 -> ***

n=16 -> ****
n=32 -> *****

$O(\log_2 n)$

סימון O גדול

הגדרה פורמלית

בהינתן שתי פונקציות (חיוביות ומונוטוניות), $f(n),g(n)$

נאמר ש-

$f(n) = O\big(g(n)\big), n \to \infty$ או במילים f היא O של $g(n)$

אם החל מ-N מסוים, קיים c כך ש-

$$f(n) \leq c \cdot g(n), \qquad n > N$$

סימון O גדול

הגדרה פורמלית

בהינתן שתי פונקציות (חיוביות ומונוטוניות), $f(n),g(n)$

נאמר ש-

$f(n) = O\big(g(n)\big), n \to \infty$ — f היא O של $g(n)$

או במילים

אם החל מ-N מסוים, קיים c כך ש-

$$f(n) \le c \cdot g(n), \qquad n > N$$

$f(n)$ חסום אסימפטוטית מלמעלה ע"י $c \cdot g(n)$

סימון O גדול

הגדרה פורמלית

$$f(n) = O\big(g(n)\big) \quad \text{if} \quad f(n) \le c \cdot g(n), \qquad n > N$$

דוגמאות

$n = O(n)$ because $n \le c \cdot n, c = 1, n \ge 0$

$20n = O(n)$ because $n \le c \cdot n, c = 20, n \ge 0$

$n = O(n^2)$ because $n \le c \cdot n^2, c = 1, n \ge 0$

$n^2 \ne O(n)$ because

for every N and every c there is an $n > N$ such that

סימון O גדול

הגדרה פורמלית

$$f(n) = O\big(g(n)\big) \quad \text{if} \quad f(n) \le c \cdot g(n), \qquad n > N$$

דוגמאות

$n = O(n)$ because $n \le c \cdot n, c = 1, n \ge 0$

$20n = O(n)$ because $n \le c \cdot n, c = 20, n \ge 0$

$n = O(n^2)$ because $n \le c \cdot n^2, c = 1, n \ge 0$

$n^2 \ne O(n)$ because

for every N and every c there is an $n > N$ such that

$$n^2 > c \cdot n$$

סימון O גדול

הגדרה פורמלית

$$f(n) = O\big(g(n)\big) \quad \text{if} \quad f(n) \le c \cdot g(n), \qquad n > N$$

דוגמאות

for every N and every c there is an $n > N$ such that

$$n^2 > c \cdot n$$

הוכחה:

נבחר N ו-c כלשהם. אז $n = \max\{N+1, c+1\}$ מקיים ש-$n > N$, ו-

$$n^2 = n \cdot n > c \cdot n$$

סימון O גדול

טענה: לכל k קבוע, מתקיים ש

$$f(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0 = O\left(n^k\right)$$

הוכחה: נגדיר

$$a_{\max} = \max\{a_0, \dots, a_k\}$$

$$\Rightarrow f(n) \leq a_{\max} n^k + a_{\max} n^{k-1} + \cdots + a_{\max} n + a_{\max}$$

$$= a_{\max}\left(\boxed{n^k + n^{k-1} + \cdots + n + 1}\right)$$

$$= a_{\max}\left(\frac{n^{k+1} - 1}{n - 1}\right)$$

סכום של טור הנדסי

סימון O גדול

טענה: לכל k קבוע, מתקיים ש

$$f(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0 = O\left(n^k\right)$$

הוכחה:

$$= a_{\max}\left(\frac{n^{k+1} - 1}{n - 1}\right)$$

$$= a_{\max}\left(n^k \boxed{\frac{n}{n-1}} \boxed{-\frac{1}{n-1}}\right) \leq a_{\max}\left(n^k 2 - 0\right)$$

$$= 2a_{\max} n^k$$

$\boxed{n > 2 \text{ אם}}$ $\boxed{\leq 2}$ $\boxed{\text{שלילי}}$

סימון O גדול

טענה: לכל k קבוע, מתקיים ש

$$f(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0 = O\left(n^k\right)$$

הוכחה:

$$= a_{\max}\left(\frac{n^{k+1} - 1}{n - 1}\right)$$

$$= a_{\max}\left(n^k \boxed{\frac{n}{n-1}} \boxed{-\frac{1}{n-1}}\right) \leq 2a_{\max} n^k$$

$\boxed{\leq 2}$ $\boxed{\text{שלילי}}$

כלומר...

סימון O גדול

טענה: לכל k קבוע, מתקיים ש

$$f(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0 = O\left(n^k\right)$$

הוכחה:

$$f(n) \leq 2a_{\max} n^k$$

או

$$f(n) \leq c \cdot n^k$$

עבור $c = 2a_{\max}$ וגם $n > 2$

סימון O גדול

הערה

הסימון $f(n) = O\big(g(n)\big)$ עם סימן = הוא מטעה!

$O\big(g(n)\big)$ היא למעשה קבוצה

קבוצת כל הפונקציות $f(n)$ כך ש...

לכן, הסימון היותר נכון הוא

$$f(n) \in O\big(g(n)\big)$$

אבל כבר התרגלנו לסימן = ...

סימון O גדול

סדרי גודל

סיבוכיות פולינומיאלית

Polynomial complexity

$O(1)$ $O(n)$ $O(n^2)$ $O(n^3)$ $O(n^k)$

Lower complexity Higher complexity

סימון O גדול

סדרי גודל

סיבוכיות אקפוננציאלית

Exponential complexity

$O(1)$ $O(n^k)$ $O(2^n)$ $O(3^n)$ $O(n!)$

Lower complexity → Higher complexity

סימון O גדול

סדרי גודל

סיבוכיות פולינומיאלית

Polynomial complexity

$O(1)$ $O(\log n)$ $O(n)$ $O(n^2)$ $O(n^3)$

Lower complexity → Higher complexity

סימון O גדול

סדרי גודל

סיבוכיות לוגריתמית
Logarithmic complexity

$O(1)$ $O(\log n)$ $O(n)$ $O(n^2)$ $O(n^3)$

Lower complexity → Higher complexity

סיבוכיות תת לינארית
Sublinear complexity

סימון O גדול

סדרי גודל

$O(1)$ $O(\log n)$ $O(n)$ $O(n \log n)$ $O(n^2)$ $O(n^2 \log n)$

Lower complexity → Higher complexity

סימון O גדול

סדרי גודל

$O(1)$ $O(\log \log n)$ $O(\log n)$ $O(n)$

Lower complexity → Higher complexity

סימון O גדול

סדרי גודל

$O(1)$ $O(\log^* n)$ $O(\log \log n)$ $O(\log n)$

Lower complexity → Higher complexity

סימונים אחרים

;

סימון O גדול

$$f(n) = O\big(g(n)\big) \quad \text{if} \quad f(n) \le c \cdot g(n), \qquad n > N$$

סימון Ω גדול

$$f(n) = \Omega\big(g(n)\big) \quad \text{if} \quad f(n) \ge c \cdot g(n), \qquad n > N$$

דוגמא

$$f(n) = n^4 + n + \log n$$

$$f(n) = O(n^4) \qquad f(n) = \Omega(\log n)$$

$$f(n) = O(n^5) \qquad f(n) = \Omega(n^4)$$

$$f(n) \ne O(n) \qquad f(n) \ne \Omega(n^5)$$

סימונים אחרים

סימון Θ גדול

$$f(n) = \Theta\big(g(n)\big) \quad \text{if} \quad f(n) = O\big(g(n)\big)$$

$$\text{and} \quad f(n) = \Omega\big(g(n)\big)$$

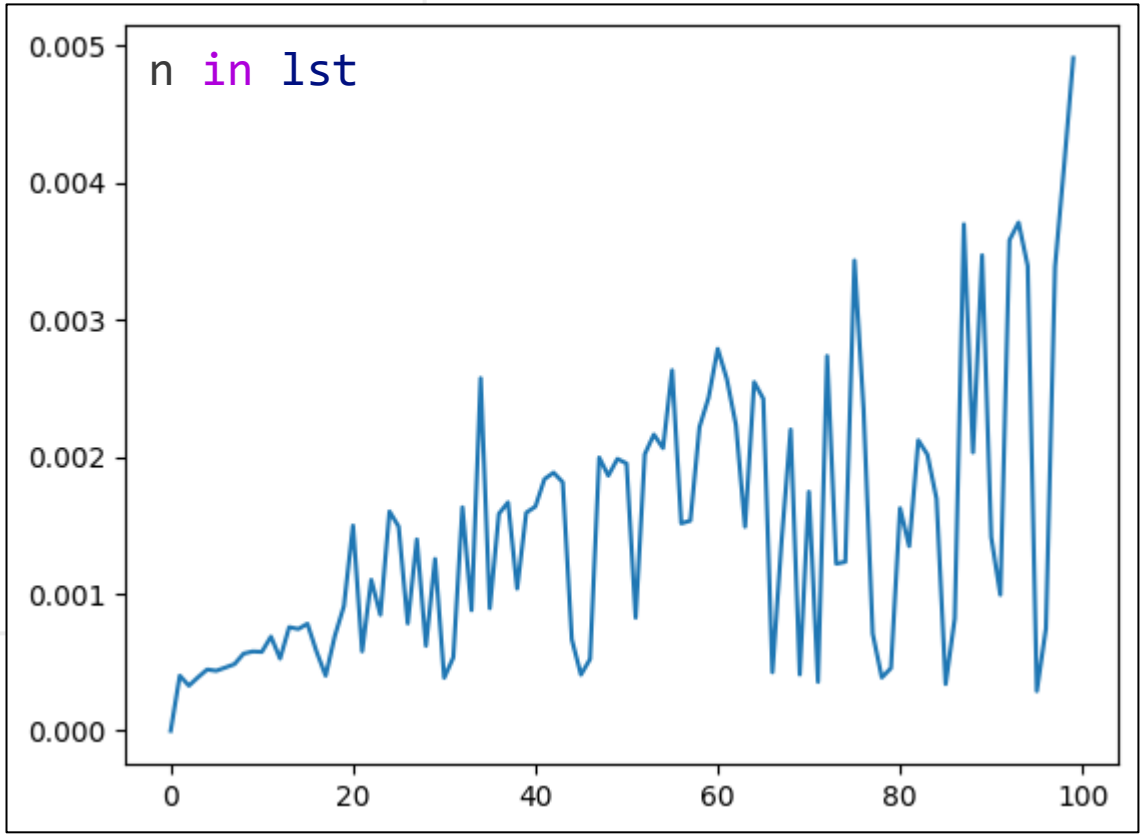
איך מודדים ביצועים של אלגוריתם?

מה הסיבוכיות O פה? $O(n)$

למרות שלפעמים מוצאים את n מוקדם, אנחנו בוחנים את המקרה הגרוע!

measure_sum.py ×

```python
import random
import timeit

def generate_random_ints(n):
    return [random.randint(0, n) for _ in range(n)]

time = [0]*100
for i in range(1, 100):
    lst = generate_random_ints(i)
    time_i = timeit.timeit(
        lambda: n in lst,
        number=10000
    )
    time[i]=time_i

import matplotlib.pyplot as plt
plt.plot(time)
```



תרגיל

רשימה של מספרים בגודל נקראת מושלמת אם היא מכילה כל מספר פעם אחת בדיוק

יש לממש תכנית המקבלת רשימה של מספרים ומחזירה True אם היא מושלמת ו-False אם לא

loop_in_loop.py ×

```python
def perfect_list(A):
    for i in range(len(A)):
        for j in range(i+1, len(A)):
            if A[i] == A[j]:
                return False
    return True
```

סיבוכיות זמן

$$1 + 2 + \cdots + n = O(n^2)$$

סיבוכיות מקום

$$O(1)$$

תרגיל

רשימה של מספרים בגודל נקראת מושלמת אם היא מכילה כל מספר פעם אחת בדיוק

יש לממש תכנית המקבלת רשימה של מספרים ומחזירה True אם היא מושלמת ו-False אם לא

memoization.py ×

```python
def all_unique(lst):
    memory = [False]*len(lst)
    for i in lst:
        if memory[i]:
            return False
        memory[i] = True
```

סיבוכיות זמן

$O(n)$

סיבוכיות מקום

$O(n)$