

מבני נתונים קיץ 2019

תרגיל בית 1

תאריך הנשחה: **25 באוגוסט 2019**

את העבודה יש להגיש במערכת ההצעות submission system
על העבודה להיות מוגשות בכתב יד ברור סרוק או הkulda במחשב

- הנקם נדרשים לנסח **חשיבות ברורות וקצרות**
- עליים לנסח תחילת את האלגוריתם במילים בכמה משפטים
- לאחר מכן בפסאדו קוד (לא בשפת תכנות אלא כמו שלמדו בכיתה, ללא פרטים טכניים)
- לאחר מכן להסביר את נכונותו (אין דרישת הסביר חיבר להיות מאוד מוכנע)
- ולבסוף לנתח את זמן הריצה

שאלות לגבי העבודה ניתן לשאול בפורום של העבודה הנמצא באתר של הקורס.

שאלה 1:

פתרו את נוסחאות הרקורסיה הבאות.

$$T(n) = 2T(n/2) + n^3 \quad \text{א.}$$

ב. $T(n) = T(n-1) + n$

ג. $T(n) = T(\sqrt{n}) + 1$

ד. $T(n) = 2T(n/2) + \frac{n}{\log n}$

ה. $T(n) = 2T(n-1) + O(1)$

ו. $T(n) = T(\sqrt{n}) + \theta(\log \log n)$

ז.

$$T(n) = T(n/2 + \sqrt[3]{n^2}) + 10$$

שאלה 2:

מצאו את החסם ההדוק ביותר המתאים לייצג היחס בין זוגות הפונקציות הבאות, נמקו את תשובהיכם

1. $10n^2 + 100n + \log n + 1000 = \underline{(n^2 + n \log n)}$
2. $(n + a)^b = \underline{(n^b)}$
3. $\sqrt{n} = \underline{((\sqrt{2})^{\log n})}$
4. $2^{n+5} = \underline{(2^n)}$
5. $2^n = \underline{(2^{2n})}$
6. $2^{3n} = \underline{(4^n)}$
7. $n2^n = \underline{(2^n)}$
8. $2^n = \underline{(3^n)}$
9. $\log_{16} n = \underline{(\log_2 n)}$
10. $n! = \underline{(1000^n)}$

שאלה 3:

הוכיחו או הפריכו באמצעות דוגמה נגדית את הטענות הבאות.
הנראה כי $f(n) \geq 1$ ו- $g(n) \geq 1$ עבור כל n .

. א. $f(n) = O((f(n))^2)$

. ב. If $f(n) = O(n)$ then $2^{f(n)} = O(2^n)$

. ג. $\max(f(n), g(n)) = \Theta(f(n) + g(n))$

. ד.

יהז $\log(f(n)) = \Theta(\log(g(n)))$ שתי פונקציות חיוביות. אם $f(n), g(n)$ איז:

. א. $f(n) = O(g(n))$

. ב. $f(n) = \Omega(g(n))$

. ג. $f(n) = \Theta(g(n))$

. ד. אף אחת מהטענות לא נכונה.

. ה. הוכיחו או הפריכו:

. א. קיימת פונקציה f כך ש- $f(n) = O(f(n))^2$

. ב. אם f ו- g פונקציות מונוטוניות עולה כך ש- $f(n) = \Omega(n)$, $f(g(n)) = O(n)$ וגם $g(n) = O(n)$.

. ג. $\log(f(n)) = \Theta(\log(g(n)))$

שאלה 4:

. א. בהינתן שתי מחרוזות, S_1 ו- S_2 , הצעו אלגוריתם לקבוע האם כל התווים של S_2 נמצאים גם כן ב- S_1 , העובד בזמן $O(n)$ במקורה המקורי. אתם רשאים להשתמש ב(1) O זכרון נסוף בלבד.

. ב. בהינתן מערך A בגודל $N+1$ המכיל מספרים שונים מ-1 עד N ותו הרוחה "_" יחיד, צריך למיין את המערך, ולשים את הרוחה בתא אחרון במערך. מותרת אך ורק פעולה החלפה ביןתו הרוחה לביןתו מסווג מספר במערך. הצעו אלגוריתם הייעיל ביותר.

שאלה 5:

נתון מערך A בגודל n , המחולק ל- $\lceil \frac{n}{k} \rceil$ קטעים באורך k כל אחד (ניתן להניח כי n מחלק ב- k ללא שארית). האיברים בכל קטע גדולים מכל האיברים בקטע משמאלו וקטנים מכל האיברים בקטע מימינו. הziיעו אלגוריתם יעיל ככל הניתן למציאת איבר x כלשהו במערך A. אם x נמצא במערך A, האלגוריתם מחזיר את האינדקס המתאים במערך, אחרת מציין -1.

נתחו את זמן הריצה של האלגוריתם שהצאתם כפונקציה של n ו- k .

שאלה 6:

- א. הziיעו אלגוריתם יועל ככל הניתן שבהנתן מערך **מופיע** A המכיל n מספרים שלמים (עם **חרזרות**) ומספר נסוף x , מחייב כמה איברים הגדולים מ x יש במערך. x לא בהכרח מופיע במערך. מצאו חסם הדוק לאלגוריתם שהציגתם.
- ב. נתון מערך בן n מספרים זרים (שונים זה מהזה). הziיעו אלגוריתם הטוב ביותר המוצא האם יש במערך שלושה מספרים z , x ו- y כך ש: $z = y + x$. במידה ולא קיימת שלשה כזו' בלבד. על האלגוריתם מחייב "לא". צינו ונמקו את זמן הריצה של האלגוריתם.

שאלה 7:

נתונים שני מערכים A ו B, המכילים מספרים שלמים עם חרזרות. כל מערך בגודל n . המספרים בטוויה [ח-ח]. הziיעו אלגוריתם אשר סופר כמה מספרים משותפים יש במערכות A ו B. מספר משותף - אותו ערך מספרי מופיע בשני המערכיים. אם מספר מופיע יותר מפעם אחת, סופר אותו פעמי' אחת בלבד. על האלגוריתם לעבוד בזמן ריצה (**O(n)** במקרה הגרוע).

לדוגמא:
 $A = \{1, 2, 14, 14, 5\}$ ו $B = \{2, 2, 14, 3\}$. התשובה הינה "שני מספרים משותפים", בדוגמא שלנו זה 2 ו 14.

שאלה 8:

נתון מערך בגודל n . הziיעו אלגוריתם אשר מתחזק קבוצה של n מספרים חיוביים, ותומך בפעולות הבאות. **אתם רשאים להשתמש ב (1) O** **וברו נסוף בלבד.**

	<u>Operation</u>	<u>Time</u>
Init()	Initialize empty set	$O(1)$
Insert(element)	Add element to set	$O(1)$
Delete()	Delete last inserted element from set	$O(1)$
GetLastElement()	Return the value of last inserted element (without removing it from set)	$O(1)$

Min()	Return the value of the minimal element (without removing it from set)	O(1)
-------	--	------

:אנוון

```

init ()
insert (6)
min ()→6
insert (10)
insert (5)
insert (7)
min ()→5
GetLastElement () → 7
delete ()
GetLastElement () → 5
min ()→5
delete ()
GetLastElement () → 10
min ()→6

```

שאלה 9:

נתונים שני מערכים A ו B המאחסנים מספרים, כאשר כל מערך ממוקן בפני עצמו ואין תלות בין הערכים של שני המריכים. נגיד רצינו של המערך המומן להיות האיבר האמצעי של המערך. במידה ומערך מכיל מספר זוגי של איברים, אז נשימוש בחציו ימנית שמודרך להיות $[n/2]$.

A. הצעו אלגוריתם למציאת החציו של המערך המאוחד המומן שבוני מהאיברים של שני המערכים A ו B.

שימוש לבן: המערך המאוחד אינו נתון ! השאלה היא : אילו היה לנו מערך ממוקן שבוני מכל האיברים של מערכים A ו B יחד, איזה איבר היה החציו של המערך זהה ?

על האלגוריתם לרוץ בזמן $(n \log m + \log m)$, כאשר n זה מספר האיברים ב A, ו m זה מספר האיברים ב B.

B. הצעו אלגוריתם למציאת אינדקס של איבר X בערך A.
על האלגוריתם לרוץ בזמן $(d \log \theta)$, כאשר d הוא מספר האיברים מופיעים לפני איבר X בערך A במידה ו X (היה) קיים בערך זהה.

הסבירו לכל אלגוריתם את הנכונות, ונתחו זמני הריצה.

דוגמה לסעיף A: יהיו נתונים שני מערכים A ו B כדלקמן:

index	1	2	3	4	5	6	7	8	9	10
A:	1	5	10	15	20	25	30	35	40	45

index	1	2	3	4	5	6	7	8	9	10
B:	2	3	4	6	7	8	9	90	100	145

(median)

אילו היה לנו מערך מאוחד ממוקן המורכב מאיברים של A ו B, היה בו 20 איברים, והערך זהה היה נראה כך:

index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
AUB:	1	2	3	4	5	6	7	8	9	10	15	20	25	30	35	40	45	90	100	145

לכן, האיבר שופיע באמצע הוא **10**, וזה החציו של המערך המאוחד. لكن, האלגוריתם שלכם צריך להחזיר תשובה "10".

שאלה 10:

חישבו את זמן הריצה של קטעי קוד הבאים בМОונחים של θ והסבירו.

.א.

```
S=0;  
for (i=1; i<n; i++)  
    for (j=0; j<n; j+=i)  
        S++;
```

.ב.

```
for (i=1; i<=n; i*=2)  
    for (j=1; j<=i; j++)  
        k++;
```

.ג.

```
k=0;  
for (i=0; i<n/4; i++)  
    for (j=n/2; j<n; j++)  
        for (k=1; k<n; k++)  
            k+=2;
```

.ד.

```
for (i = 1; i <= n; i++)  
    for (j = 1; j <= 10; j++)  
        for (k = n; k <= n+5; k++)  
            x = n;  
            while (x > 1)  
                x = x / 2;
```

.ה.

```
for i = 1 to n  
    for j = 1 to i  
        k=n  
        while k>2  
            k = ⌊k1/3⌋
```

.ו. foo הינה פונקציה שמקבלת ערך שלם ח

```
foo (n)  
    if n ≤ 1  
        return 1
```

```
if n is odd
    for i=1 to n
        print i
    return foo(n-1) + 1
if n is even
    i=n
    while i>2
        j=1
        while j<i
            j=j*2
            i=└i/3┘
    return 2 * foo(n-1)
```

בצלחה !

מבנה נתונים – עבודה 1

מג'ישים: שגב נגר ואלמוג למא'

שאלות 1, 3 ו-10 מצורפות בסוף בכתב יד.

שאלה 2

סעיף 1

נפתח את אגד שמאל:

$$10n^2 + 100n + \log n + 1000 \leq 10n^2 + 100n^2 + n^2 + 1000n^2 = 1111n^2 = \Theta(n^2)$$

נפתח את אגד ימין:

$$n^2 + n \log n \leq n^2 + n^2 = 2n^2 = \Theta(n^2)$$

$$\text{Therefore: } 10n^2 + 100n + \log n + 1000 = \underline{\Theta}(n^2 + n \log n)$$

סעיף 2

$$(n+a)^b \leq (n+n)^b = (2n)^b = \underline{\Theta}(n^b)$$

סעיף 3

$$(\sqrt{2})^{\log n} = 2^{1/2 * \log n} = (2^{\log n})^{1/2} = n^{1/2} = \sqrt{n}$$

$$\text{Therefore: } (\sqrt{2})^{\log n} = \underline{\Theta}(\sqrt{n})$$

סעיף 4

$$2^{n+5} = 2^n * 2^5 = 32 * 2^n = \underline{\Theta}(2^n)$$

סעיף 5

$$2^{2n} = (2^n)^2 = 2^n * 2^n = \underline{\Omega}(2^n)$$

הכפלנו את 2^n בגורם התלוי ב- n , ומכאן האומגה.

סעיף 6

$$2^{3n} = (2^3)^n = 8^n = (2 * 4)^n = 2^n * 4^n = \underline{\Omega}(4^n)$$

סעיף 7

$$n * 2^n = \underline{\Omega}(2^n)$$

סעיף 9

$$\log_{16} n = \log n / \log 16 = c * \log n = \underline{\Theta}(\log n)$$

שאלה 4

סעיף א'

פירוט האלגוריתם

- ניצור מערך A בגודל 177 המציג את כל התווים האפשריים בטבלת ASCII מאותחל לאפסים.
- נעבור על איברי המחרוזת S1, ונעלם ב-1 את ערך ה-ASCII הרלוונטי של כל איבר שלא בערך A.
- נעבור על איברי S2, ולכל איבר נבדוק האם אינדקס ה-ASCII המתאים שלו ב- A שווה לאפס.

פואודו קוד:

isS2inS1

```

A = new array with 177 elements
for i=0 to s1.length
    A[s1.charAt[i]]++
for j = 0 to s2.length
    if[A[s2.charAt[j]] = 0
        return false
return true

```

ניתוח זמן ריצה:

Initializing the array costs $O(1)$, each loop costs $O(n)$
 $O(1) + O(n) + O(n) = O(n)$

סעיף ב'

פירוט האלגוריתם:

- נמקם באופן חד פעמי את הרוחה במיקום האחרון בערך
 - נעבור בלולאה על כל איברי המערך, יהי a המספר בלולאה הנוכחיות:
 - נחליף את a ברוחה בסוף המערך
 - נחליף את הרוחה במספר הנמצא באינדקס השווה ל-a
 - נחליף את הרוחה ב-a הנמצא כרגע בסוף המערך
- לאחר 3 פעולות אלו המס' a יהיה במיקום המתאים שלו בערך המקורי. מדובר בפעולות של החלפת רוחה במספר בלבד כמובן.

פואודו קוד:

SpaceSort(A)

```

find space and switch A[N] and Space
for i =0 to N
    switch A[i] with A[N]
    switch A[i] with A[A[N]]           \\\ A[A[N]] = A[a]
    switch A[A[N]] with A[N]

```

ניתוח זמן ריצה

- Finding space is done with linear array scan = $O(N)$
- each switch action in the loop costs = $O(1)$
- a total of: $O(N) + N*3*O(1) = O(N)$

שאלה 5

פירוט האלגוריתם

- נבצע חיפוש לינארי על האיברים הראשונים של כל קטע ב-A.
- בתום החיפוש נבודד לכל היותר 2 קטעים צמודים ש-x עשוי לעמוד בהם - כאשר x גדול מהאיבר הראשון בקטע השמאלי, וקטן מהאיבר הראשון בקטע הימני.
- לאחר מכן נבצע חיפוש איבר-איבר בשני קטעים אלו למציאת x, אם לא נמצא נחזיר -1.

פואודו קוד

isXInA(A, x)

Perform binary search on A[a*k], 0 <= a <= n/k

Let A[m] <= x <= A[m+k]

for i=m to m+2k

 if A[i] = x

 return i

return -1;

ניתוח זמן ריצה:

- There are n/k parts, so binary search on the first elements in each parts costs $O(\log(n/k))$.
- for loop costs $O(2k)$
- Total: $O(\log(n/k)) + O(2k) = O(\log(n/k) + k)$

שאלה 6

סעיף א'

פירוט האלגוריתם

- המערך ממויין, לכן ניתן להשתמש בחיפוש בינארי למציאת x
- אם x במערך: נחזיר אותו כרגע במסגרת החיפוש הבינארי
- אם x אינו במערך: נבדוק מתי בחישוב הבינארי הגיענו לקטע במערך באורך 1, נבדוק האם האיבר האחרון גדול או קטן מ-x, ונחזיר את כמות האיברים המתאימה.

פואודו קוד

HowManyBiggerThanX(A, x)

implement binary search to look for x

\\\ high & low are the right & left binary search indexes:

if high - low = 1

 if A[low] > x

 return n - low + 1

 else

 return n - low

חישוב זמן ריצה

- The addition to the code is performed in $O(1)$, so the running time is equal to the running time of binary Search: $O(\log n)$

סעיף ב'

פירוט האלגוריתם

- נמיין את המערך באמצעות מיוון השוואה כלשהו
- נרצה בלולאה על כל איבר במערך, כך שבסכל לולאה נבדוק האם קיימים שני איברים שסכום
- שווה לאיבר זה.
- במסגרת הבדיקה בכל לולאה, נשתמש בשני פוינטרים להתחלה ולסוף המערך, ונסכם את שני האיברים במקומות של הפוינטרים ונשווה ל-z. אם הסכום קטן/גדול מ-z נקדם בהתאם את אחד הפוינטרים כדי לקרב אותו ל-z.

פואודו קוד

FindZSum(A)

```

Sort the array using merge sort
for k = 0 to k = n
    i = 0, j = n
    while (i < j)
        if A[i] + A[j] = A[k]
            return true
        else if A[i] + A[j] < A[k]
            i++
        else
            j--
return false

```

ניתוח זמן ריצה

- Merge sort = $O(n \log n)$
- for occurs n times in the worst case, while occurs n times in the worst case.
- Total of: $O(n \log n) + O(n^2) + O(n^2)$

שאלה 7

פירוט האלגוריתם

- ניצור מערך עזר C בגודל $2n$.
- עבור כל איברி מערך A. לכל איבר a במערך A נעדכן את האינדקס $a+n$ במערך העזר.
- מוסיפים ח' כדי שנמנעו מניסיון עדכון של אינדקסים שליליים שאינם קיימים.
- עבור כל איברி מערך B. לכל איבר b במערך B נבדוק האם $[a+b]C$ גדול מ一封ס מה שמצויב על מספר זהה במערכות A ו-B. במקרה זה נעלם את הקאונטר שלנו ונאפס את $[a+b]C$ כדי שנמנעו מדופלי-קציות בספירה.

פתרונות קיד

SumCommon(A, B)

```

C = new array with 2n cells
sum = 0
for i=0 to m
    C[A[i] +n] ++
for l = 0 to m
    if[C[B[i] + n] > 0
        sum ++
        C[B[i] + n] = 0
return sum

```

ניתוח זמן ריצה

- Creating empty array = $O(1)$
 - each loop = $O(m)$
- total of $O(1) + O(m) + O(m) = O(m)$

שאלה 8

פירוט האלגוריתם

- נישם את האלגוריתם באמצעות מחסנית ומשתנה גלובלי `min`, שישמר את הערך המינימלי במחסנית ויתעדכן בכל הכנסה והוצאה.
- הבעיה: בפעולה מחקה של איבר מינימלי, כיצד נשחרר את האיבר המינימלי הקודם שהוכנס?
- להתמודדות עם בעיה זו, בכל פעם שנכנס איבר מינימלי למחסנית, נעדכן את המשתנה `min` כרגע אך נכניס למחסנית את ערך ה-`min` הישן עם אינדיקטור כלשהו - למשל עם סימן מינוס.
- כתע, בכל פעם שנוציא ערך שלילי מהמחסנית נדע כי מדובר בערך המינימלי הקודם שאנו צריכים לשחרר - לכן נעדכן את ה-`min` להיות ערך זה (חויבי כמובן).
- סגירת הקצה האחרון: בפעולה `GetLastElement`, במקרה שהאיבר בראש המחסנית שלילי, הרו שהערך האמתי שלו שמור בשדה `min`, לכן נחזיר את `min` במקומם.

פואודו קוד

```
Init()
    create empty stack s
    int min = null

Insert(element)
    if(s.isEmpty())
        s.push(element);
        min = element;
    else
        if(element>min)
            s.push(element);
        else
            s.push(-1*min);
            min=element;

Delete()
    if(!s.isEmpty())
        if(s.peek()>0)
            s.pop();
        else
            min=-1*s.pop();
    if(s.isEmpty())
        min=null;

GetLastElement
    if(s.peek()>0)
        return s.peek();
    return min;
```

```
Min()
    return min;
```

ניתוח זמן ריצה:

Initializing the stack and the min variable costs O(1).
Stack pop, push and peek methods costs O(1) each.

שאלה 9

סעיף א

פירוט האלגוריתם

- מוצאים את החזיונים של המרכיבים A ו-B, נסמן a ו-b בהתאם.
- אם $a = b$: משמע ש-a ו-b הם גם החזיונים במערך המאוחד, לכן נחזיר אותם.
- אם $a > b$:

משמע שכלי האיברים ממשאל ל-a במערך A קטנים ממחצית המרכיבים ב-A וגם קטןים מ לפחות חצי המרכיבים ב-B, על כן לא יכולים להיות חזיונים של המערך המאוחד. באופן דומה, כל האיברים מימין ל-B גדולים מחצי מהאיברים ב-B ולפחות מחצי מהאיברים ב-A, לכן אין מועמדים להיות חזיון.

לכן, נשלח רקורסיבית את מערך A עם כל האיברים מ-a ואילך, ואת B עם כל עד b.

אם $a < b$: באופן סימטרי לסעיף הקודם.

תנאי העצירה של הרקורסיה: כאשר אחת מהמערכות בגודל קטן או שווה לאחד. נשווה את האיבר לחזיוון של המערכת השני (נראה אם הוא לפני או אחרי), ונחזיר את החזיוון המתאים.

פואודו קוד

```
FindMedian(A, B)
    a = A[A.length\2]
    b = B[B.length\2]
    if      a = b
        return a
    else if a.length <=1 or B.length <=1
        compare the single element with the median of the second array, and
        return the relevant median
    else if a < b
        return FindMedian(second half of A, first half of B)
    else
        return FindMedian(first half of A, second half of B)
```

זמן ריצה

הrekorsiah כל הזמן מתבצעת על חזי מערך A ועל חזי מערך B, על כן:
 $O(\log m + \log n)$

סעיף ב

פירוט האלגוריתם

- נסורך את איברי המערך A בקצבירות של חזקות של 2 ($0, 2, 4, 8$) עד אשר נגיע לאיבר גדול מ- x .
 - תהא i האיטרציה בה הגיענו למספר הראשון שגדול מ- x . לכן:
$$A[2i-1] < x < A[2i]$$
 - נבצע חיפוש ביןארי בתחום זה מספרים עד אשר נמצא את x .

פואודו קוד

```
FindIndex(A, x)
    i = 0
    while A[2i] < x
        i++
    if A[2i] = x
        return x
    else
        implement binary search on A[2i-1] to A[2i] to find x and return its index
```

ניתוח זמן ריצה

- Scanning A elements in the while loop costs $O(\log d)$ (less than $2d$ elements before $A[2i]$)
- In the worst case, there are d elements within $A[2i-1]$ to $A[2i]$, so the binary search costs $O(\log d)$
- total of: $O(\log d) + O(\log d) = O(\log d)$

1 2210X - P'UJUJ UZN

- IONS ZINGO

322529316 281 282

1 2810e

(16) $T(n) = 2T\left(\frac{n}{2}\right) + n^3$ $\sim \text{CONSTANT} \times \log \text{base}$

$a=2, b=2, f(n) = n^3$

$n^{\log_b a} = n^{\log_2 2} = n$

3 گرگنی چکنی پس $f(n) = n^3 = \mathcal{R}(n)$

\therefore پیشنهاد ۲ پیشنهاد ۱ چکنی $\cdot \text{CONSTANT} \times \log \text{base}$ ε
 $\mathcal{E} = 1 \text{ چکنی (1)}$

$\mathcal{R}(n^{\log_b a + \varepsilon}) = \mathcal{R}(n^2) = n^3 = f(n)$ $c = \frac{1}{2} \text{ چکنی (2)}$

$a \cdot f\left(\frac{n}{b}\right) = 2 \cdot \left(\frac{n}{2}\right)^3 = 2 \cdot \frac{n^3}{8} = \frac{n^3}{4}$

$\leq c \cdot f(n) = \frac{n^3}{2}$ $n \geq 1 \text{ چکنی}$

$T(n) = \Theta(n^3)$

$\therefore \text{CONSTANT} \times \log \text{base} , \text{پس}$

$$\because \text{递归式} \quad T(n) = T(n-1) + n \quad \text{①}$$

$$T(n) = T(n-1) + n \xrightarrow{\text{递归式}} = T(n-2) + (n-1) + n$$

$$\xrightarrow{\text{递归式}} T(n-3) + (n-2) + (n-1) + n$$

\therefore 递归式 0103. 用 1 代替

$$T(n) = T(n-i) + \left(i \cdot n - \sum_{k=0}^{i-1} k \right)$$

\therefore 递归式 163NJ.

$$T(n-i) \Rightarrow n-i = 1, i = n-1$$

\therefore 递归式 163J

$$T(n) = T(n-(n-1)) + (n-1) \cdot n - \sum_{k=0}^{n-2} k$$

$$= T(1) + n^2 - n - \sum_{k=0}^{n-2} k$$

$\alpha_2 = 0, n = n-2, d = 1$ \therefore 163N00 0000

$$\alpha_n = \alpha_2 + d(n-2) = n-3$$

$$S_n = n \frac{(\alpha_2 + \alpha_n)}{2} = n \frac{(n-3)}{2} = \frac{n^2 - 3n}{2}$$

$$\dots = 1 + n^2 - n - \frac{n^2 - 3n}{2} = \frac{n^2}{2} + \frac{n}{2} + 1 = \underline{\underline{E}}$$

$$T(n) = T(\sqrt{n}) + 1$$

2

$$\begin{aligned} T(n) &= T(\sqrt{n}) + 1 \xrightarrow{\text{recursion}} = T(\sqrt{\sqrt{n}}) + 1 + 1 \xrightarrow{\text{recursion}} \\ &= T(\sqrt{\sqrt{\sqrt{n}}}) + 1 + 1 + 1 \end{aligned}$$

$$T(n) = T\left(n^{\frac{2}{21}}\right) + i$$

$$n^{\frac{1}{2^K}} = 2 \xrightarrow{j=K} n^{\frac{1}{K}} = 2 \Rightarrow n = 2^K$$

$$\cancel{\log_2} \rightarrow \log n = k \xrightarrow[\text{msh}]{\cancel{2^k}} 2^i = \log n$$

$$\underline{\underline{109_2}} \rightarrow i = \log(\log n)$$

$$T(n) = T(2) + \log(\log n) = \Theta(1) + \log(\log n)$$

$$= \underline{\underline{O(\log(\log n))}}$$

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

8

רְאֵת אֶת־בְּנֵי־עַמָּךְ כִּי־בְּנֵי־עַמָּךְ יְהוָה

ה כחוק נאטו ז כם היה נטף

• ፳፻፲፭ ፲፻፲፭

$$T(2^k) = 2T(2^{k-1}) + \frac{2^k}{\log 2^k} = 2T(2^{k-1}) + \frac{2^k}{k \log 2}$$

$$= 2T(2^{k-2}) + \frac{2^k}{k} \xrightarrow{\text{2^n3^nC10}} 2\left(2T(2^{k-2}) + \frac{2^{k-2}}{k-1}\right) + \frac{2^k}{k}$$

$$3 \xrightarrow{\text{using C10}} 2\left(2\left(2\left(2\left(T\left(2^{k-3}\right) + \frac{2^{k-2}}{K-2}\right) + \frac{2^{k-1}}{K-1}\right) + \frac{2^k}{K}\right)$$

$$T(2^k) = T(2) + \sum_{j=1}^k \frac{2^k}{j} = T(2) + 2^k \sum_{j=1}^k \frac{1}{j}$$

$$\xrightarrow{\text{ר'גנ'רָטִיבָּלִיטֶה}} = \Theta(2^k \cdot \log k)$$

וְנִזְמָן בְּבֵית יְהוָה

$$n = 2^k \Rightarrow k = \log n$$

$$T(n) = \Theta(n \cdot \log(\log n))$$

$$T(n) = 2T(n-1) + O(1)$$

: 3- $T(n-1)$ 2- $T(n-2)$...

$$T(n) = 2T(n-2) + O(1) \xrightarrow{2-3-16}$$

$$2(2T(n-2) + O(1)) + O(1) \xrightarrow{3-3-16}$$

$$2(2(2T(n-3) + O(1)) + O(1)) + O(1).$$

$$T(n) = 2^i \cdot T(n-i) + \sum_{k=0}^{i-1} 2^k \cdot O(1)$$

: 3- $T(n-1)$ 2- $T(n-2)$... 1- $T(n-i)$

$$n-i = 1, i = n-1$$

$$T(n) = 2^{n-1} \cdot T(1) + \sum_{k=0}^{n-2} 2^k \cdot O(1)$$

$$= 2^{n-1} + \underline{\alpha \cdot O(1)} = \underline{\underline{\Theta(2^n)}}$$

per n : 3- $T(1)$ 2- $T(2)$... 1- $T(n)$

$$T(n) = T(\sqrt{n}) + \Theta(\log \log n)$$

①

$$m = \log \log n \Rightarrow n = 2^{2^m} \Rightarrow \sqrt{n} = 2^{2^{m-1}}$$

$$T(2^{2^m}) = T(2^{2^{m-1}}) + \Theta(m) \quad \because \text{rec}$$

מ \sqrt{n} נקבע $n = 2^{2^m}$ ו $m = \log \log n$

\therefore תרשים הינו יפה יפה

$$T(2^{2^{m-1}}) = T(2^2) = T(2)$$

$$T(2^{2^m}) = m \cdot \Theta(m) = \Theta(m^2) \quad \because \text{ps}$$

$$T(2^{2^{\log \log n}}) = \Theta((\log \log n)^2) \quad \because n \text{ נקבע}$$

$$T(2^{2^{\log \log n}}) = T(n) = \underline{\Theta((\log \log n)^2)}$$

$$T(n) = T\left(\frac{n}{2}\right) + \sqrt[3]{n^2} + 10$$

5

(100%) $T(n) = T\left(\frac{n}{2}\right) + \sqrt[3]{n^2}$, $\frac{n}{2} > n^{\frac{2}{3}}$ ת'ה גורם ל'ה סדרה
 $T(n) = T\left(\frac{n}{2}\right) + 10$ - מילוי 15 סדרה
merge sort של מילוי אמור ב $\Theta(\log n)$ סדרה

3 Ordre

(16)

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

לעומת: $f(n) \leq C \cdot f(n)^2$ $\forall n > n_0$ $\exists C, n_0 > 0$

לעת $\exists c, n_0$ $f(n) = \Theta(n^2) \quad \forall n > n_0$

$$\exists c, n_0 \quad f(n) \leq c \cdot f(n) \quad \therefore (f(n)) \geq c$$

$$n_0 = 1, c \geq 1 \quad \text{סבוכו}$$

(2)

$$\text{If } f(n) = O(n) \text{ then } 2^{f(n)} = O(2^n)$$

~~כגון שרטוט~~

$$f(n) \leq c_2 \cdot n \quad \because f(n) \text{ קדום}$$

~~לעומת שרטוט~~

$$2^{f(n)} \leq c_2 \cdot 2^n$$

לעת $f(n) = 2n$, $f(n) = 2n$ $\forall n > n_0, c_2 > 2$ $\exists c_2$

$$2^{2n} \leq c_2 \cdot 2^n \quad | : 2^n \quad \therefore \text{לא}$$

$$2^n \leq c_2 \Rightarrow \lim_{n \rightarrow \infty} 2^n = \infty$$

הוכחה

$$\max(f(n), g(n)) = \Theta(f(n) + g(n))$$

(2)

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

$\therefore n > n_0$ סופר $\Rightarrow n_0 > 0, C_1, C_2 > 0$ $\rho(N)$

$$0 \leq C_1 \cdot (f(n) + g(n)) \leq f(n) \leq C_2 \cdot (f(n) + g(n))$$

$$C_1 = \frac{1}{2} \text{ גורן}$$

$$f(n) + g(n) \leq 2f(n) \Leftarrow f(n) \geq g(n) - e \text{ גורן}$$

$$\underbrace{\frac{1}{2} \cdot (f(n) + g(n))}_{C_1} \leq \frac{1}{2} \cdot 2f(n) = f(n) \quad \therefore \text{סוד}$$

$$g(n) \geq 1 - e \text{ גורן}, C_2 = 1 \text{ גורן}$$

$e \approx 0.72$ $f(n) \leq f(n) + g(n)$ גורן

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

(2)

$\therefore n > n_0$ סופר $C_1, C_2 > 0$ $\rho(N)$

$$0 \leq C_1 \cdot \log(g(n)) \leq \log(f(n)) \leq C_2 \cdot \log(g(n))$$

10 תרגיל

לעומת הגדלת כפיפה כפיפה - נסמן 10

$$\begin{aligned} \text{ר'נ'ג } n &+ \frac{n}{2} + \frac{n}{3} + \dots + \frac{n}{n} = \\ \frac{n}{1} &+ \frac{n}{2} + \frac{n}{3} + \dots + \frac{n}{n} = \\ \text{ר'נ'ג} &+ \frac{n}{2} + \frac{n}{3} + \dots + \frac{n}{n} = \\ n + \frac{n}{2} + \frac{n}{3} + \dots + \frac{n}{n} &= \\ = \sum_{i=2}^n \frac{n}{i} &= n \cdot \underbrace{\sum_{i=2}^n \frac{1}{i}}_{\text{הרואית}} = n \cdot \Theta(\log n) = \underline{\underline{\Theta(n \log n)}} \end{aligned}$$

לעומת הגדלת כפיפה כפיפה - נסמן 2

$$\begin{aligned} \text{ר'נ'ג } n &+ 2 + 2^2 + 2^3 + \dots + n = \\ \text{ר'נ'ג } n &+ 2 + 2^2 + 2^3 + \dots + 2^{\log n} = \\ \text{ר'נ'ג } n &+ 2 + 2^2 + 2^3 + \dots + 2^{\log n} = \\ 1 + 2 + 2^2 + 2^3 + \dots + n &= 1 + 2 + 2^2 + 2^3 + \dots + 2^{\log n} \end{aligned}$$

$$= \sum_{k=0}^{\log n} 2^k \xrightarrow{\text{ר'נ'ג}} \alpha_2 = 1, \alpha_1 = 2$$

$$S_n = \frac{1 \cdot (2^{\log n} - 1)}{2 - 1} = 2^{\log n} - 1 = n - 1 = \underline{\underline{\Theta(n)}}$$

2. גורף מינימום: (i) $\frac{5}{4}$ סיבובים.

מיפויים נקראים (j) ו (k) מיפויים סימטריים (symmetric mappings) אם $\rho_{ij} = \rho_{ji}$ ו $\rho_{ij} = \rho_{ik}$ עבור כל i, j, k .

• **איך מגדירים נס** ? נס הוא מושג שפוך קיומו מוכיח קיומו.

$$\frac{n}{4} \cdot \frac{n}{2} \cdot \frac{n}{3} = \frac{n^3}{24} = \underline{\underline{\Theta(n^3)}}$$

$$n \cdot 10 \cdot 5 \cdot \log n = \Theta(n \log n)$$

• GCF, LCM of $24, 36, 144$ - $k > 2$, ') | L

$$k^{\left(\frac{1}{3}\right)^i} \leq 2 \Rightarrow \log k^{\left(\frac{1}{3}\right)^i} \leq \log 2 \Rightarrow \left(\frac{1}{3}\right)^i \log k \leq 1$$

$$\Rightarrow \log k \leq 3^i \Rightarrow \log_3 \log k \leq i$$

לעומת פונקציית $\log_3 \log n$ ש带回ה n פעמיים, פונקציית $\log n$ ש带回ה n פעם אחת, מוגדרת כפונקציית הלוגריתם הטבעי. פונקציית הלוגריתם היא פונקציה לא-ליניארית, כלומר לא ניתן לרשום אותה כפונקציית פולינום. פונקציית הלוגריתם היא פונקציה לא-ליניארית, כלומר לא ניתן לרשום אותה כפונקציית פולינום.

$$(1+2+3+\dots+n) \cdot \log_3 \log n = \Theta(n^2 \log_3 \log n)$$

לעומת $\Theta(n^2)$, מתי n גדול מ- \sqrt{n} ?

①

$O(n)$ פס $\log_3 n$ מיותר

לעומת

לעומת

בלעומת $\log_3 n$ מיותר?

בלעומת $\log_3 n$ מיותר?

בלעומת $\log_3 n$ מיותר?

בלעומת $\log_3 n$ מיותר?

$\log_3 n \cdot \log_3 n = O(\log_3 n \log_3 n)$?

לעומת $\log_3 n \cdot \log_3 n$ מיותר?

$$\frac{n}{2} \cdot n + \frac{n}{2} \cdot \log_3 n \log_3 n = \frac{n^2}{2} + \frac{n \log_3 n \log_3 n}{2} = \underline{\underline{\Theta(n^2)}}$$