מבני נתונים

תרגול 1 - מבוא

zvimints@gmail.com - צבי מינץ

שעות קבלה – בתיאום מראש

היום

- מבני נתונים
- אלגורתמי מיון
- זמני ריצה ואסימפטוטיקה
- אלגורתמי חיפוש
- עבודה עצמית

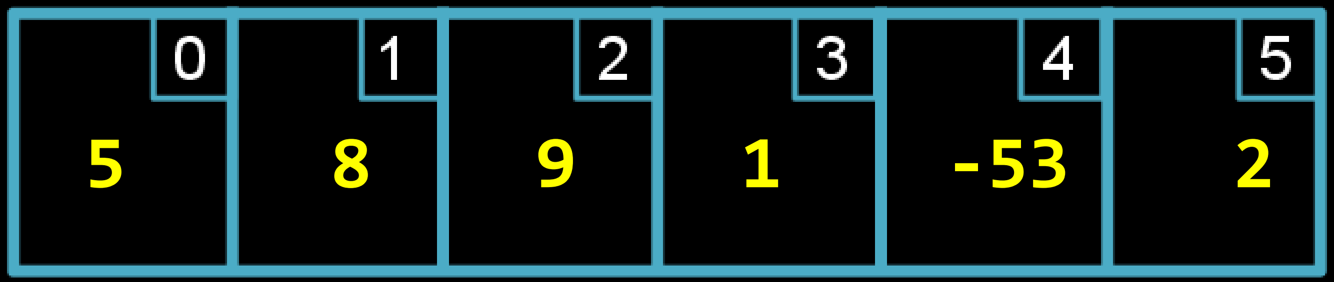
מבנה נתונים

דרך לאחסן נתונים (**מידע**) במחשב

צבי מינץ

מערך

מבנה שמורכב מאוסף של תאים **סדרתיים**



Arrays

מחלקה זו **מספקת** פונקציות סטטיות אשר פעולות על מערכים

```
import java.util.Arrays;
```

מערך ממויין

```
int[] arr = {5,8,9,1,-53,2};

Arrays.sort(arr);
```

[-53,1,2,5,8,9]

מערך לא ממויין

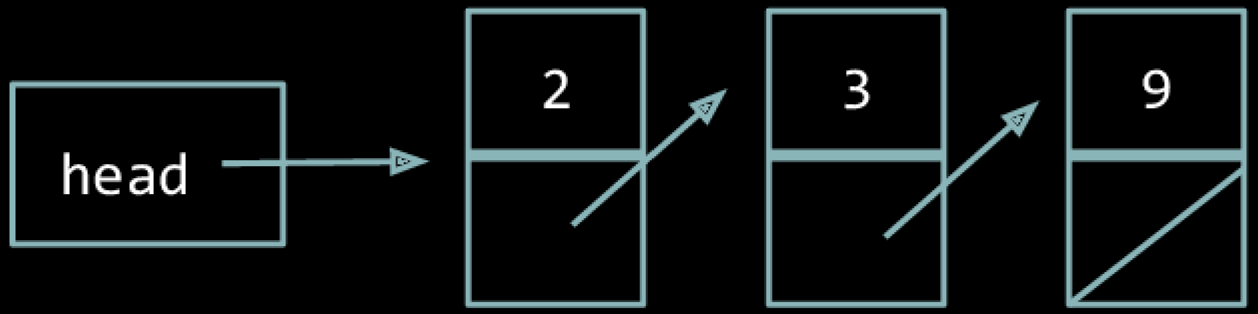
```
int[] arr = {5,8,9,1,-53,2};

int[] arr = new int[6];
arr[0] = 5;
arr[1] = 8;
arr[2] = 9;
arr[3] = 1;
ar
ar
intln(Arrays.toStr
```

[5,8,9,1,-53,2]

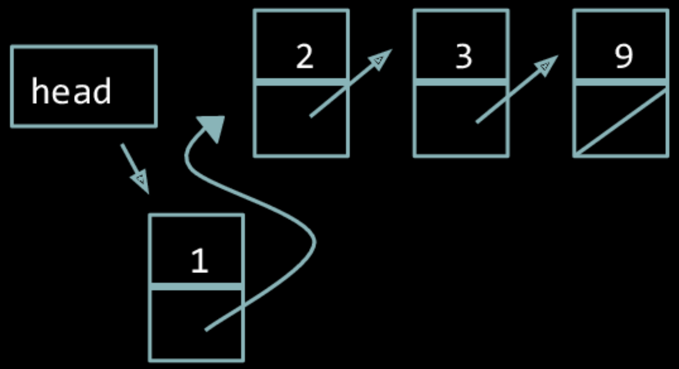
צבי מינץ

רשימה מקושרת

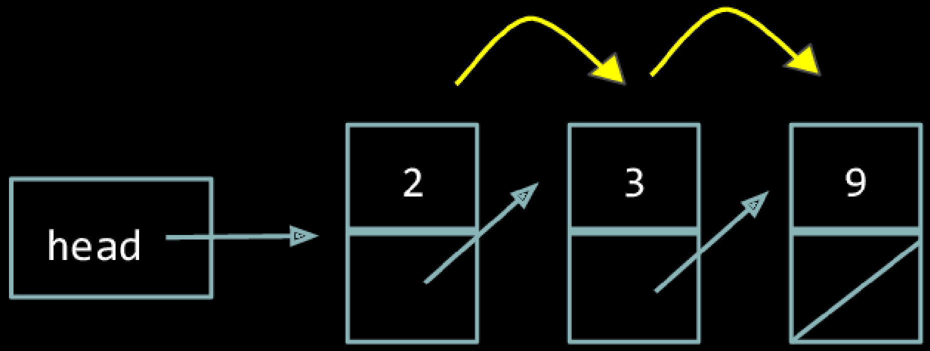
מבנה נתונים בו כל איבר **מצביע** על האיבר הבא אחריו



הוספה



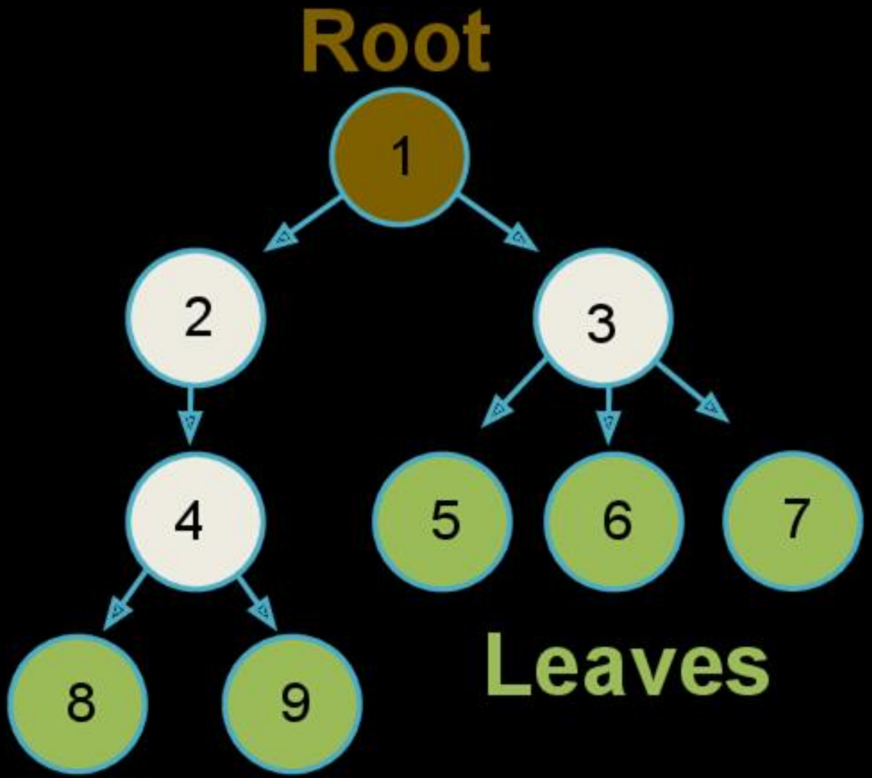
חיפוש



עץ

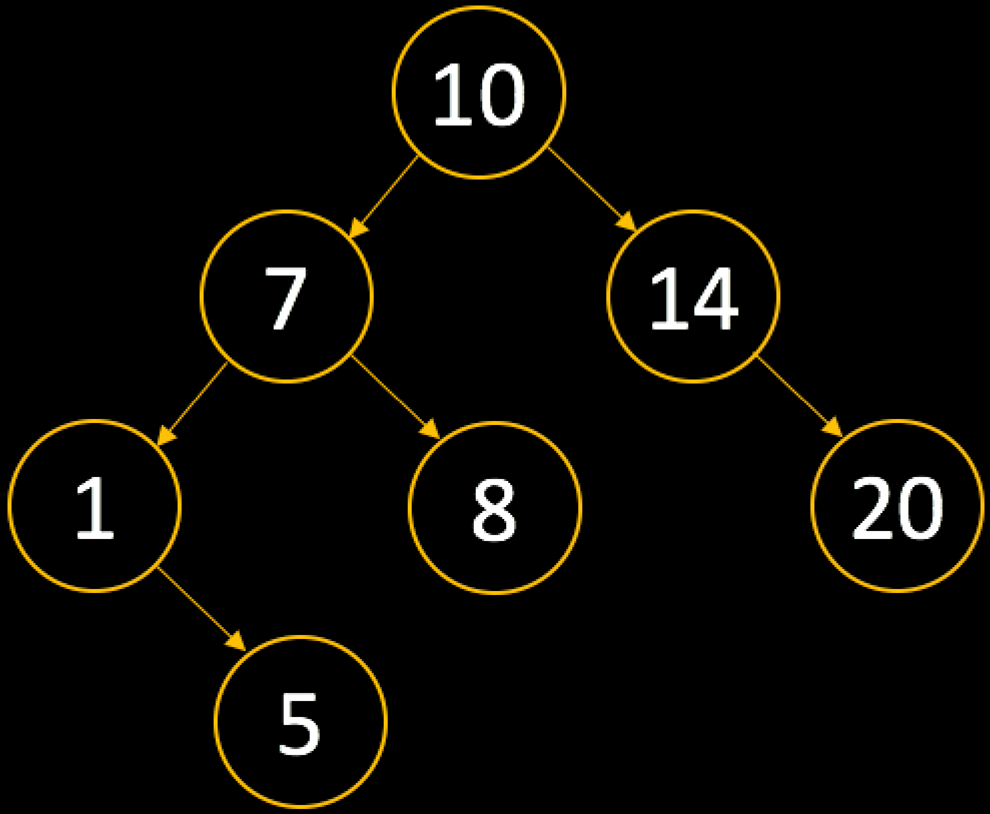
עץ הוא אוסף של קודקודים (Nodes) כאשר לכל קודקוד יש:

1. ערך
2. רשימה (יכולה להיות ריקה) של מצביעים לבנים



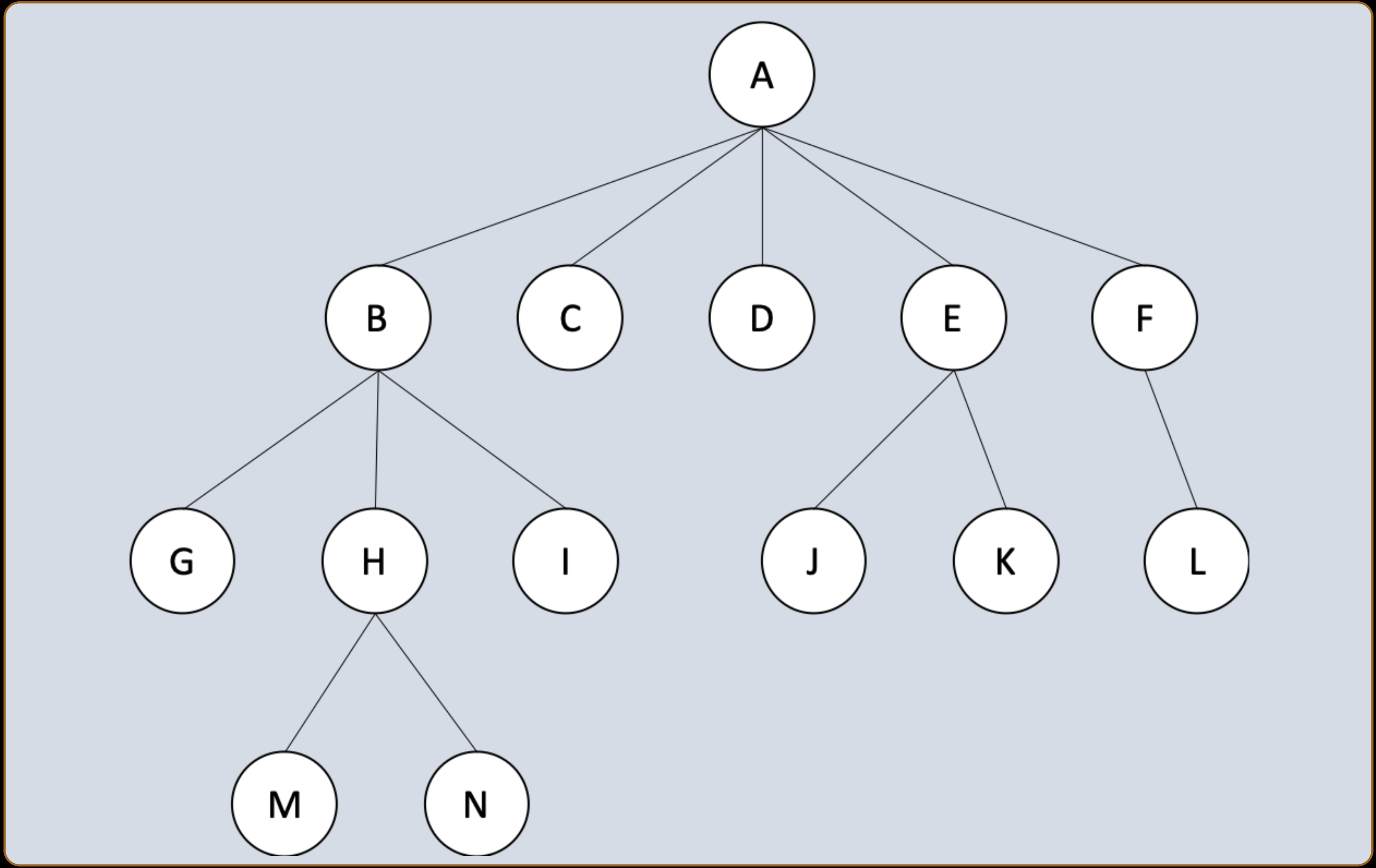
עץ בינארי

עץ שלכל קודקוד פנימי יש לכל היותר 2 בנים



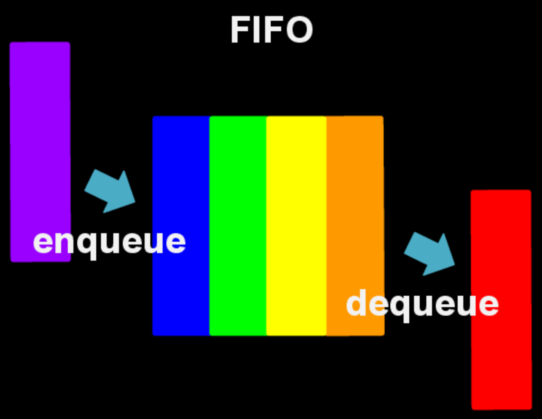
עץ טרינרי

עץ שלכל קודקוד פנימי יש לכל היותר 3 בנים

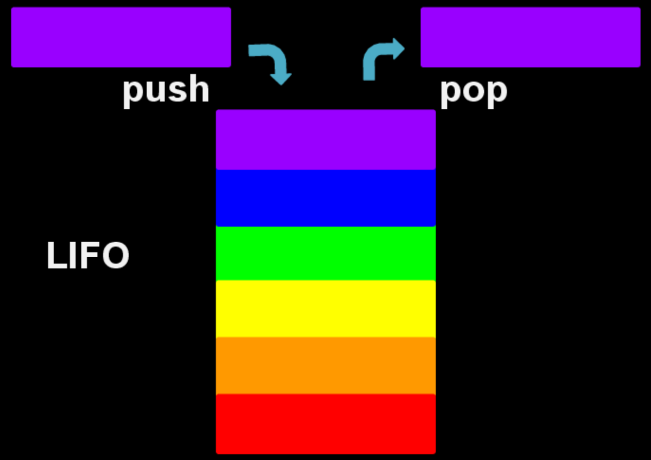


עוד מבני נתונים

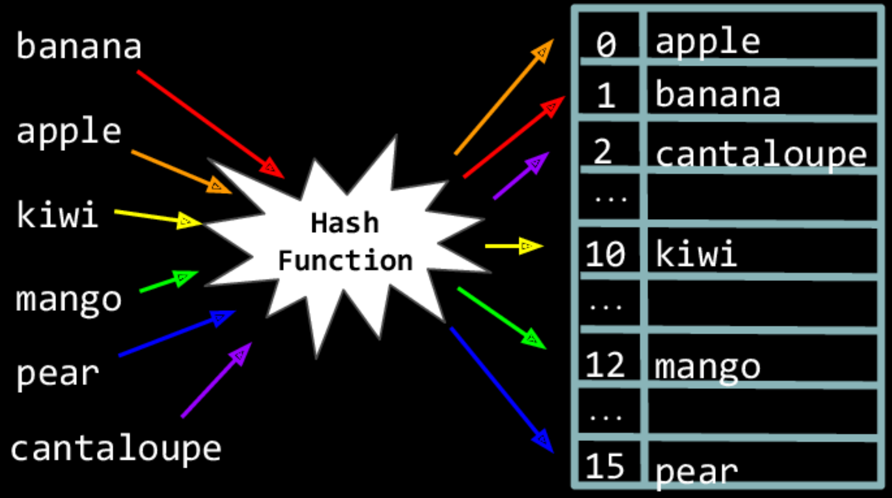
תור Queue (FIFO)



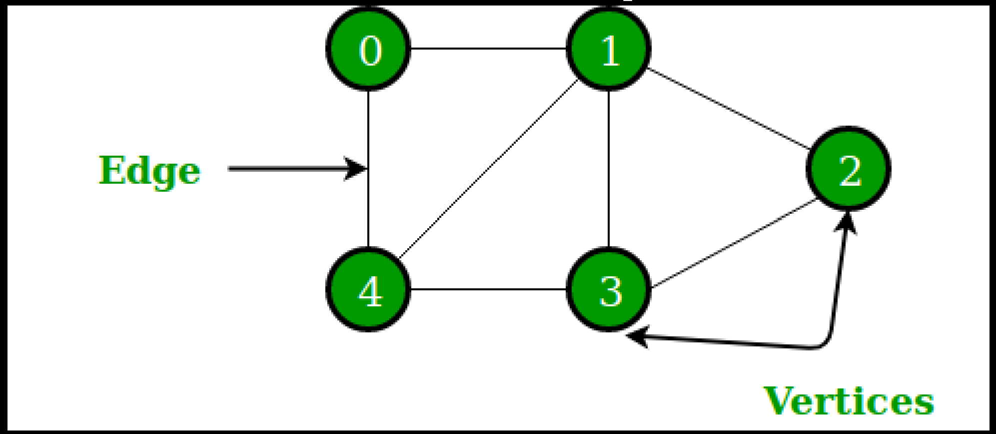
מחסנית Stack (LIFO)



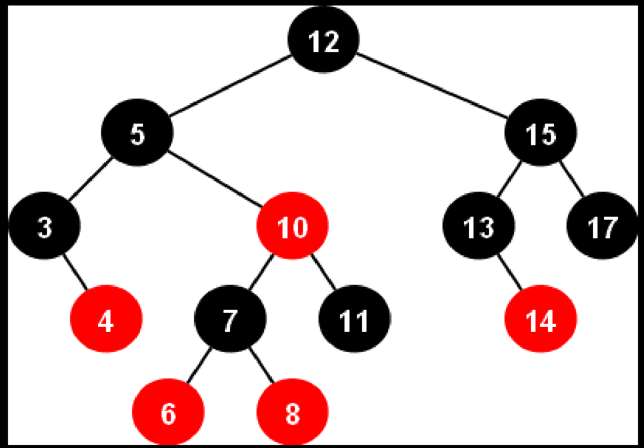
טבלת גיבוב



גרף



עצים מאוזנים



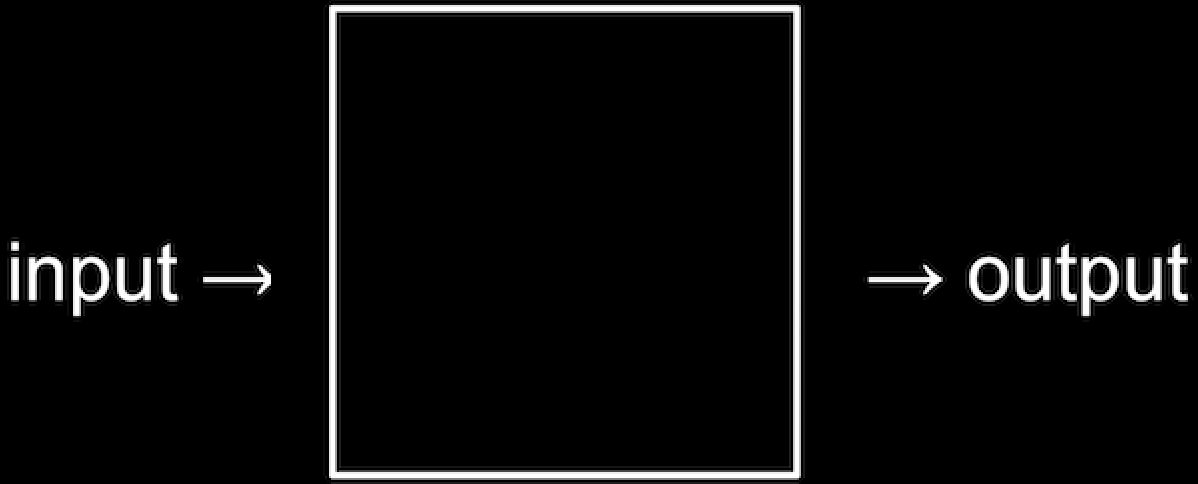
ועוד...

http://bigocheatsheet.com/

צבי מינץ

אלגוריתם

דרך שיטתית וחד-משמעית לביצוע של משימה מסוימת, במספר <u>סופי</u> של צעדים.



אלגוריתם

דרך שיטתית וחד-משמעית לביצוע של משימה מסוימת, במספר <u>סופי</u> של צעדים.

input →

{100,19,17,2,7,3,36,25,1}

אלגוריתם מיון

→ output

{1,2,3,7,17,19,25,36,100}

Pseudocode

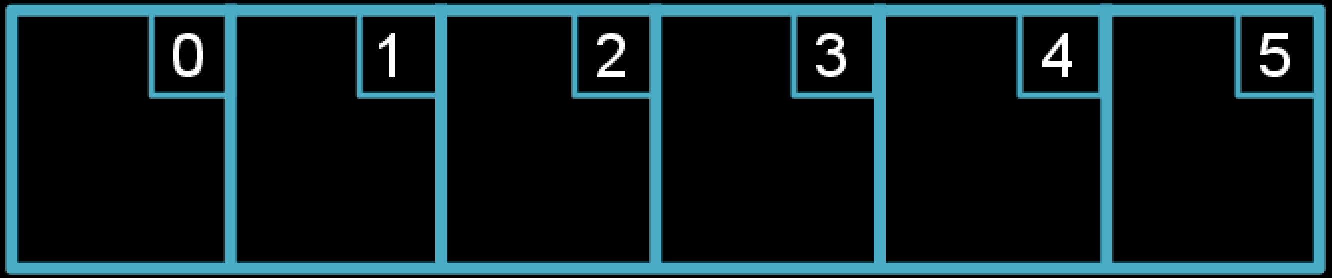
פסאדו קוד - תיאור מצומצם ולא רשמי לאלגוריתם של תוכנית מחשב

- שימו לב כי למבחן יש לדעת מימוש מלא של כל קוד בשפת JAVA ולא פסאדו!

Bubble Sort

מיון בועות

צבי מינץ



n=6

```
repeat until no swaps
    for i from 0 to n-2
        if i'th and i+1'th elements out of order
            swap them
```

-3	4	88	1	3

-3	4	88	1	3

```
public class MyProgram {
    // Method to test above
    public static void main(String[] args) {
        int[] arr = {8,7,6,5,4,3,2,1};
        BubbleSort(arr);
        System.out.println(Arrays.toString(arr));
    }

    // A function to implement bubble sort
    public static void BubbleSort(int[] arr) {
        for(int i=0,n=arr.length; i < n-1 ; i++)
            for (int j=0; j<n-i-1; j++)
                if(arr[j] > arr[j+1])

    p            void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
}
```

[1,2,3,4,5,6,7,8]

*after each **iteration** of the loop largest element of the array is always placed at right most position.*

Therefore, the loop invariant condition is that at the end of i iteration right most i elements are sorted and in place.

https://www.geeksforgeeks.org/loop-invariant-condition-examples-sorting-algorithms/

Selection Sort

מיון בחירה

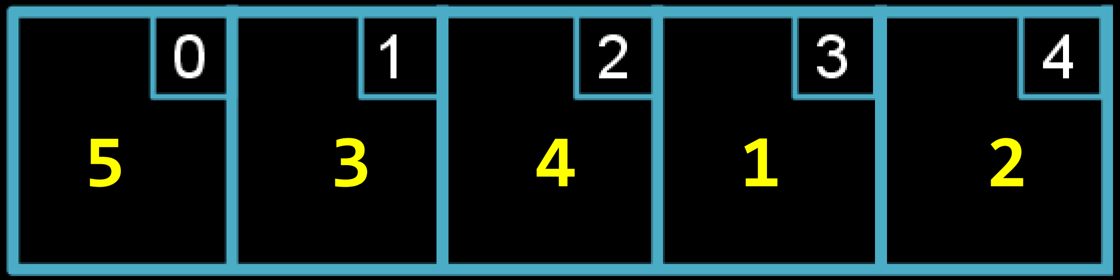
מציאת מינימום במערך

```java
// Find Minimum value in input array
// Input: Array
// Output: Index of the smallest value
public static int FindMin(int[] arr) {
if(arr.length == 0) return -1;
int min_index = 0;
for(int i=1, n = arr.length; i<n; i++)
    if(arr[i] < arr[min_index])
          min_index = i;
return min_index;
}
```

```java
System.out.println(FindMin(new int[0])); // Output: -1
System.out.println(FindMin(new int[]{1,-2,3})); // Output: 1
```

smallest ← first element in the array

```
for i from 0 to n-1
    find smallest element between i'th and n-1'th
    swap smallest with i'th element
```


0
5
1
3
2
4
3
1
4
2

5 3 4 1 2

Selection Sort

```java
 // Method to test above
public static void main(String[] args) {
    int[] arr = {8,7,6,5,4,3,2,1};
    SelectionSort(arr);
    System.out.println(Arrays.toString(arr));
}
```

[1,2,3,4,5,6,7,8]

שאלה: איך לממש את פונקציית Swap ללא משתנה עזר? (3 שורות)

```java
// A function to implement selection sort
public static void SelectionSort(int[] arr) {
// One by one move boundary of unsorted subarray
for(int i=0,n=arr.length; i < n-1 ; i++) {
    // Find the minimum element in unsorted array
    int min_index = i;
        for (int j=i+1; j<n; j++)
            if(arr[j] < arr[min_index])
                min_index = j;

    // Swap the minimum ele with the first element
    swap(arr,min_index,i);
    }
}

// Swap Functions
public  static void swap(int[] arr, int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

*loop **invariant** condition is that the subarray A[0 to i-1] is always sorted.*

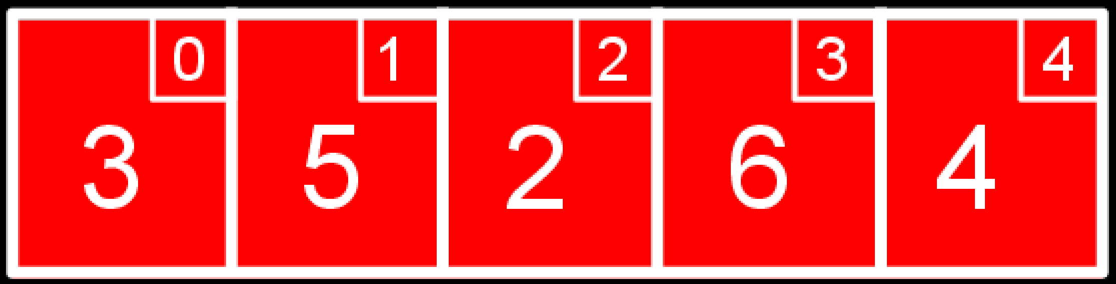
https://www.geeksforgeeks.org/loop-invariant-condition-examples-sorting-algorithms/

Insertion Sort

מיון הכנסה

Sorted

Unsorted



For each unsorted element n:

1. Determine where in sorted portion of the list to insert n

2. Shift sorted elements rightwards as necessary to make room for n

3. Insert n into sorted portion of the list

6 5 3 1 8 7 2 4

```java
 // Method to test above
public static void main(String[] args) {
    int[] arr = {8,7,6,5,4,3,2,1};
    InsertionSort(arr);
    System.out.println(Arrays.toString(arr));
}

// A function to implement Insertion sort
public static void InsertionSort(int[] arr) {
    int n = arr.length;
    for(int i=1; i < n ; i++) {
        int key = arr[i];
        int j = i - 1;
        // Move elements from arr[0...i-1]
        // that are greater then key to one position
        // ahead of their current position
        while( j >= 0 && arr[j] > key ) {
            arr[j+1] = arr[j];
            j = j-1;
        }
        arr[j+1] = key;
    }
}
```

6 5 3 1 8 7 2 4

זמן ריצה

זמן ריצה

ז... מסוים היא מספר פעולות היסוד המבוצעות

- ○
- ○ נרצ...
  כגון

שאלה:

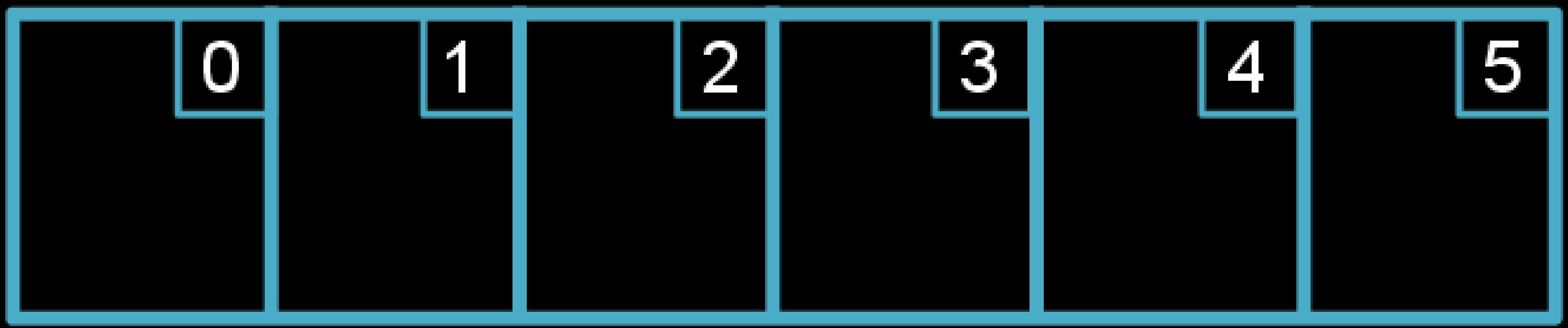
האם מודדים זמן ריצה ביחידות של זמן (שניות, דקות וכו') ?

לא!

משך הזמן לביצוע פעולות **משתנה מסביבת ריצה אחת לשנייה**.

בוחנים זמן ריצה ביחידות מדידה מוחלטות כגון,
מספר צעדים או מספר פעולות

מיונים – חסם עליון



מיונים – חסם עליון

$$(n-1)+(n-2)+\ldots+1$$

$$=\frac{(n-1)\cdot(n-1+1)}{2}=\frac{n\cdot(n-1)}{2}$$

$$=\frac{n^2}{2}-\frac{n}{2}=O(n^2)$$

$$\frac{(1,000,000)^2}{2}-\frac{1,000,000}{2}$$

= 500,000,000,000 – 500,000

=499999500000

Merge Sort

```
On input of n elements:
    If n < 2
        Return.
    Else
        Sort left half of elements.
        Sort right half of elements.
        Merge sorted halves.
```

6 5 3 1 8 7 2 4

```java
public static void main(String[] args) {
    int[] arr = {8,7,6,5,4,3,2,1};
    MergeSort(arr);
    System.out.println(Arrays.toString(arr));
}
```

```java
private static void MergeSort(int[] arr) {
    MergeSort(arr,0,arr.length-1);
}
```

```java
private static void MergeSort(int[] arr, int start, int end)
{
    if(start < end) {
        int middle = (start+end)/2;
        MergeSort(arr,start,middle);
        MergeSort(arr,middle+1,end);

        Merge(arr,start,middle,end);
    }
}
```

```java
private static void Merge(int[] arr, int start, int middle, int end) {
    int[] temp = new int[end - start + 1];
    int i = start;
    int j = middle + 1;
    int k = 0; // The Running Pointer
    while( i <= middle && j <= end)
    {
        if(arr[i] < arr[j]) temp[k++] = arr[i++];
        else temp[k++] = arr[j++];
    }
    while( i <= middle)
        temp[k++] = arr[i++];

    while( j <= end)
        temp[k++] = arr[j++];

    // Copy The array
     i = start;
     k = 0;
     while( k < temp.length && i <= end)
        arr[i++] = temp[k++];
}
```

מיזוג

$$T(n) = T(n/2) + T(n/2) + O(n)$$

מיון התת מערך השמאלי   מיון התת מערך הימני

סיבוכיות קוד

```
// basic_step
// if condition
// X steps when X is a number
```

$$O(1)$$

```
for(int i=0; i<n; i++) {
    // basic_step1
    // basic_step2
}
```

$$T(n) = 2 \cdot n = O(n)$$

```
for(int i=0; i<20; i++) {
    // basic_step1
    // basic_step2
}
```

$$T(n) = 2 \cdot 20 = 40 = O(1)$$

לולאה חיצונית

לולאה פנימית

```
for(int i=0; i<50; i++) {
    for(int j=0; j<100; j++) {
        ...
    }
}
```

```java
for(int i=n; i>0; --i) {
    for(int j=0; j<n; j++) {
        // basic_step
    }
}
```

$$T(n) = \sum_{i=1}^{n}\sum_{j=0}^{n} 1 = \sum_{i=1}^{n} n = n^2 = O(n^2)$$

```java
for(int i=1; i<=n; i*=2) {
    if(j == 3) System.out.println("X");
    else if(j==4) System.out.println("Y");
    else {
        j++;
        System.out.println("Z");
    }
```

בכל איטרציה מבצעים 4 **פעולות**, כמה איטרציות יש?

נעקוב אחרי i, הערכים שהוא מקבל הינם: $1,2,4,8 \dots, n$

נניח כי $n = 2^k$ כאשר k זה כמות האיטרציות, ולכן $k = \log_2(n)$

מכיוון שבכל איטרציה מבצעים 4 פעולות, נקבל כי הסיבוכיות היא $O(\log_2(n))$

במקרה זה לא נוכל לכפול את הלולאה החיצונית בפנימית!

נתבונן קודם כל בניתוח **לא** מדוייק:

הלולאה **החיצונית** מתבצעת $\log_2 n$ פעמיים והלולאה **הפנימית**

מתבצעת במקרה הגרוע ביותר n פעמיים ולכן נקבל סה"כ $O(n \cdot logn)$

כדי לקבל חסם הדוק, נחשב לפי סיגמאות

```java
for(int i=1; i<=n; i*=2) {
    for(int j=1; j<=i; j++) {
        // basic_step
    }
}
```

נשים לב כי i מקבל ערכים $1,2,4, \dots, n$, לפיכך, נגדיר משתנה k שירוץ מ-0 עד $log\ n$, נגדיר את i להיות 2^k ונקבל:

$$T(n) = \sum_{k=0}^{logn}\sum_{j=1}^{i} 1 = \sum_{k=0}^{logn}\sum_{j=1}^{2^k} 1 = \sum_{k=0}^{logn} 2^k = 2^{\log n+1} - 1 = 2 \cdot 2^{logn} - 1 = 2n - 1 = \Theta(n)$$

צבי מינץ

אסימפטוטיקה

אסימפטוטיקה הינה <u>הערכה</u> של קצב גידול של פונקציה.

עבור שתי פונקציות נתונות $f(n), g(n)$ נרצה לומר מה היחס בינהן

נתובנן בפונקציות $f(n), g(n)$ חיוביות

חסם אסימפטוטי עליון

$$O$$

Big-O

חסם אסימפטוטי עליון O

הגדרה 1:

נאמר ש - $f(n) \in O(g(n))$

$$\Updownarrow$$

קיימים שני קבועים $c \geq 0$, $n_0 \geq 0$

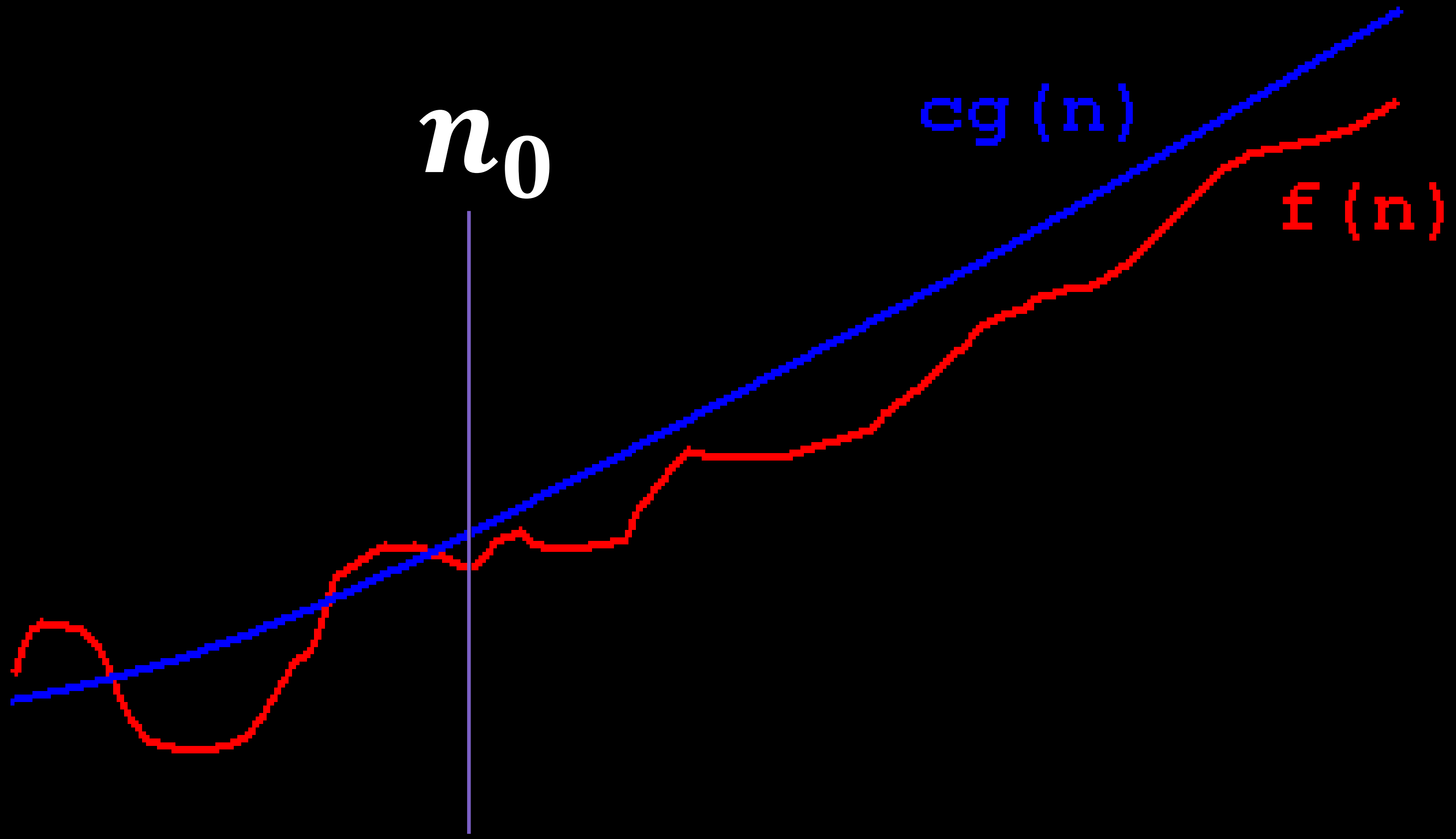
כך **שלכל** $n \geq n_0$

מתקיים:

$$|f(n)| \leq c \cdot |g(n)|$$

$$\lim_{n\to\infty} \left| \frac{f(n)}{g(n)} \right|$$

חסם אסימפטוטי עליון O



צבי מינץ

חסם אסימפטוטי עליון O

הגדרה 2:

נאמר ש- $f(n) \in O(g(n))$

$$\Updownarrow$$

$$0 \leq \lim_{n \to \infty} \left| \frac{f(n)}{g(n)} \right| < \infty$$

חסם אסימפטוטי עליון O

- **אם אנו אומרים ש-$f(n) \in O(g(n))$ אין זה אומר שתמיד $f(n) \le g(n)$**
- כאשר אנו אומרים שזמן הריצה הוא $O(n^2)$ הכוונה היא:
  שזמן הריצה במקרה **הגרוע ביותר**,
  לקלט **הגרוע ביותר**, חסום ע"י $c \cdot n^2$
- **לפעמים רושמים ביטויים מהצורה $n^2 + 5n + 2$ כ- $n^2 + O(n)$**
- $O\big(f(n) + g(n)\big) = O(max(f(n),g(n)))$

חסם אסימפטוטי עליון O

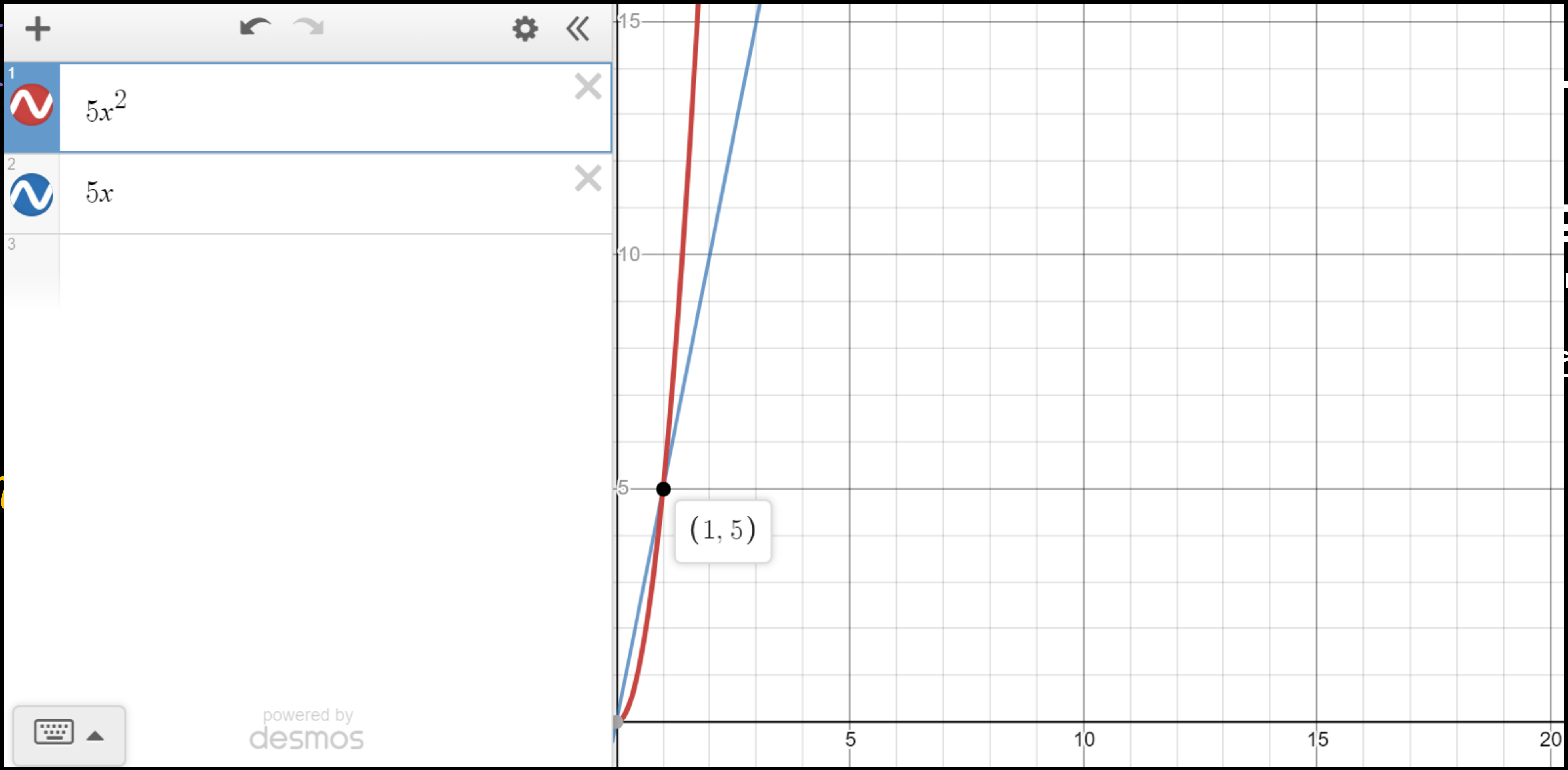
$T($ דו

לפי הגדרה 1:

נראה כי קיימי

כך שלכל $n_0 \ge$

$|T(n$ נשים לב כי:



ולכן עבור $c = 15 \ge 0$, $n_0 = 1 \ge 0$ נקבל כי $T(n) = O(n^2)$

חסם אסימפטוטי עליון O

סימון Notation	שם
$O(c^n)$	Exponential אקספוננציאלי
$O(n^c)$	Polynomial פולינומי
$O(n^2)$	Quadratic ריבועי
$O(n)$	Linear לינארי
$O(log(n))$	Logarithmic לוגרתמי
$O(1)$	Constant קבוע

$O(nlogn)$

$O(loglogn)$

$c \in \mathbb{R}$

צבי מינץ

חסם אסימפטוטי עליון O

דוגמא: נניח שזמן ריצה של תוכנית A הוא $T(n) = 10n^2 + 5n$

נוכיח כי $T(n) \in O(n^2)$

לפי הגדרה 2:

$$\lim_{n\to\infty} \left|\frac{T(n)}{g(n)}\right| = \lim_{n\to\infty} \left|\frac{10n^2+5n}{n^2}\right| = 10$$

חסם אסימפטוטי עליון O

טענה: $n^2 \neq O(n)$

הוכחה בעזרת הגדרה 1:

נניח בשלילה שקיימים $\begin{matrix} c \geq 0 \\ n_0 \geq 0 \end{matrix}$ כך שלכל $n \geq n_0$ מתקיים:

$$n^2 \leq c \cdot n$$

בגלל שהטענה נכונה לכל $n \geq n_0 = 0$ נוכל לבחור $n \neq 0$ ולצמצם

$$\boldsymbol{n \leq c}$$

בסתירה לכך שהנוסחא מתקיימת עבור כל $n > n_0$

חסם אסימפטוטי עליון O

טענה: $n^2 \neq O(n)$

<u>הוכחה בעזרת הגדרה 2:</u>

$$\lim_{n\to\infty}\left|\frac{f(n)}{g(n)}\right| = \lim_{n\to\infty}\left|\frac{n^2}{n}\right| = \lim_{n\to\infty}|n| \longrightarrow \infty$$

בסתירה להגדרה

$$x! \qquad\qquad x^x$$

בעזרת כלל המנה נחשב את $\lim_{n\to\infty} \frac{x!}{x^x}$

עבור $a_n = \frac{x!}{x^x}$ נחשב את $\lim_{n\to\infty} \left(\frac{a_{n+1}}{a_n}\right) = q$

וכן אם $q < 1$ אזי $\lim_{n\to\infty} a_n = 0$ ואם $q > 1$ אז $\lim_{n\to\infty} a_n = \infty$

$$\lim_{n\to\infty} \left(\frac{\mathbf{a_{n+1}}}{\mathbf{a_n}}\right) = \lim_{n\to\infty} \left(\frac{(x+1)!}{(x+1)^{x+1}} \cdot \frac{x^x}{x!}\right) = \lim_{n\to\infty} \left(\frac{x!\,(x+1)}{(x+1)} \frac{x^x}{(x+1)^x}\right) = \lim_{n\to\infty} \frac{x^x}{(x+1)^x}$$

$$= \lim\left(\left(1 - \frac{1}{x}\right)^x\right) = \frac{1}{e} < 1$$

$$\Rightarrow \lim_{n\to\infty} a_n = 0 \Rightarrow x! \in O(x^x)$$

צבי מינץ

חסם אסימפטוטי תחתון

$$\Omega$$

Omega

חסם אסימפטוטי תחתון Ω

לעיתים נרצה לדבר על **חסם תחתון** לבעיה מסויימת.

לדוגמא

מיון של n מספרים דורש **לפחות** n פעולות

למה?

חסם אסימפטוטי תחתון $\boldsymbol{\Omega}$

הגדרה 1:

נאמר ש - $f(n) \in \Omega(g(n))$

$\Updownarrow$

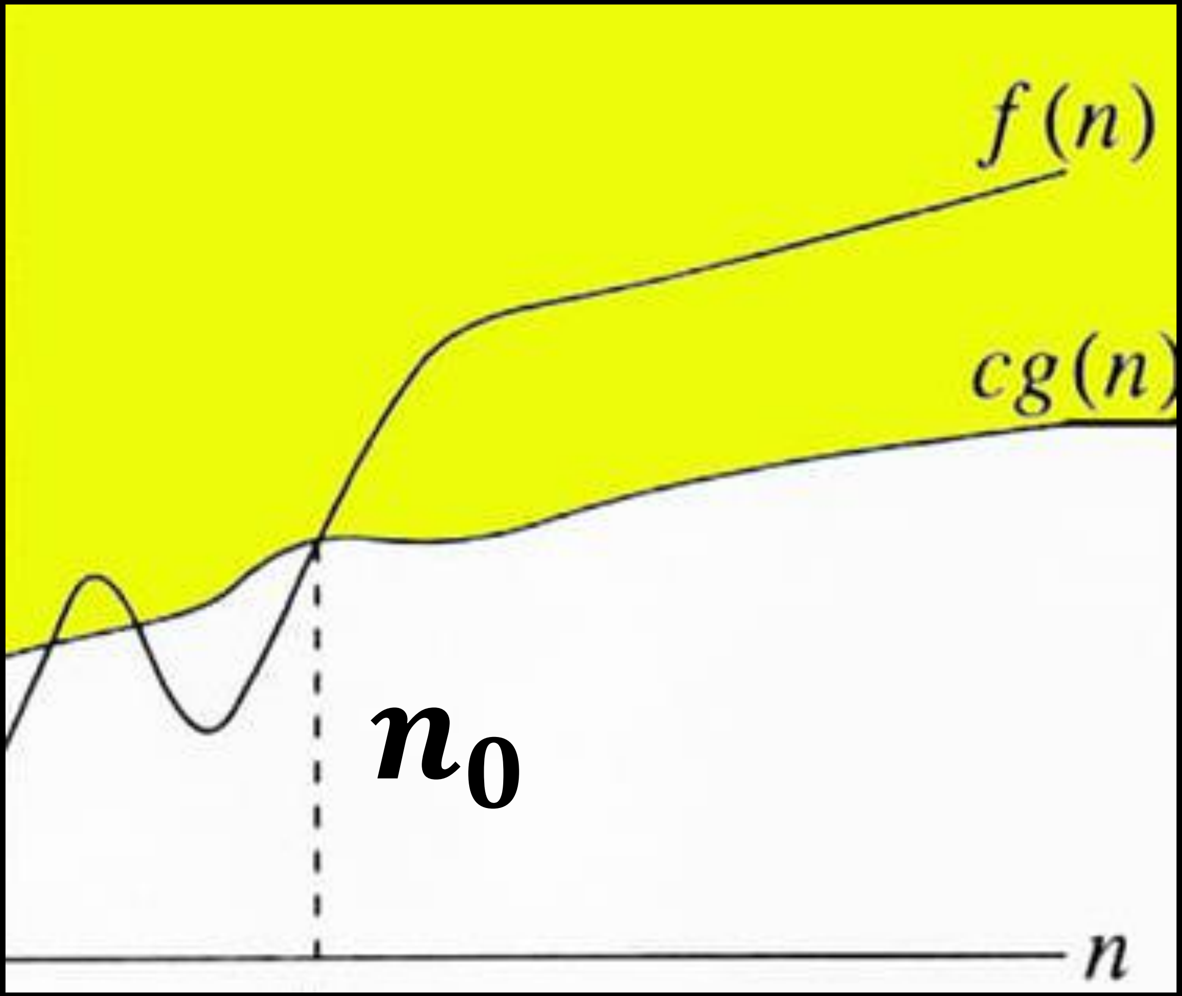
קיימים שני קבועים $c \geq 0$, $n_0 \geq 0$

כך **שלכל** $n \geq n_0$

מתקיים:

$0 \leq c \cdot |g(n)| \leq |f(n)|$

חסם אסימפטוטי תחתון Ω



חסם אסימפטוטי תחתון Ω

הגדרה 2:

נאמר ש- $f(n) \in \Omega(g(n))$

$$\Updownarrow$$

$$0 < \lim_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| \leq \infty$$

חסם אסימפטוטי תחתון Ω

<u>דוגמא:</u> נניח שזמן ריצה של תוכנית A הוא $T(n) = 3n^2 + 5n$

נוכיח כי $T(n) \in \Omega(n)$

<u>לפי הגדרה 1:</u>

נראה כי קיימים שני קבועים $c \geq 0$, $n_0 \geq 0$

כך <u>שלכל</u> $n \geq n_0$ מתקיים $0 \leq c \cdot |g(n)| \leq |f(n)|$

נשים לב כי:

$$|T(n)| = 3n^2 + 5n \geq 3n + 5n = 8n = 8 \cdot |g(n)| \geq 0$$

$3n^2 \geq 3n$
עבור כל $n \geq 1$

ולכן עבור $c = 8 \geq 0$, $n_0 = 1 \geq 0$ נקבל כי $T(n) = \Omega(n)$

צבי מינץ

חסם הדוק אסימפטוטית Θ

$$\Theta$$

Theta

חסם הדוק אסימפטוטית Θ

$$\Theta$$

נאמר ש- $f(n) = \Theta(g(n))$

אם מתקיים כי $f(n) = \Omega(g(n))$ <u>וגם</u> $f(n) = O(g(n))$

חסם הדוק אסימפטוטית Θ

הגדרה 1:

נאמר ש - $f(n) \in \Theta(g(n))$

$$\Updownarrow$$

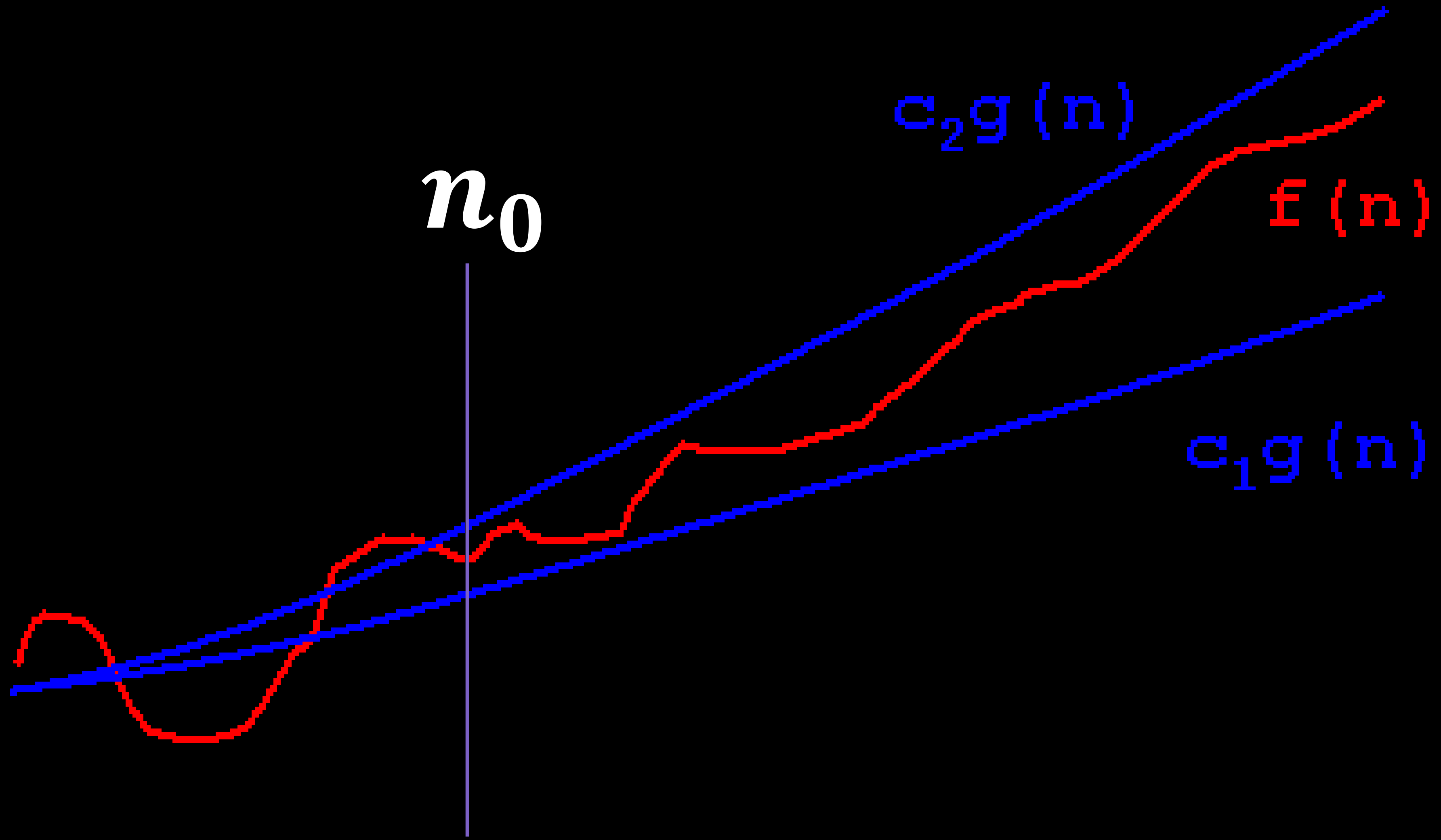
קיימים שלושה קבועים
$n_0 \geq 0$
$c_1 \geq 0$
$c_2 \geq 0$

כך **שלכל** $n \geq n_0$

מתקיים:

$$0 \leq c_1 \cdot |g(n)| \leq |f(n)| \leq c_2 \cdot |g(n)|$$

חסם הדוק אסימפטוטית Θ



חסם הדוק אסימפטוטית $\boldsymbol{\Theta}$

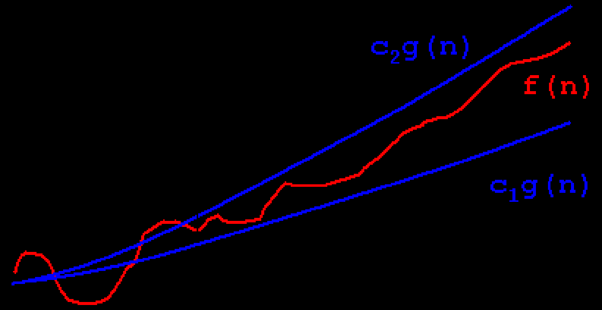
הגדרה 2:

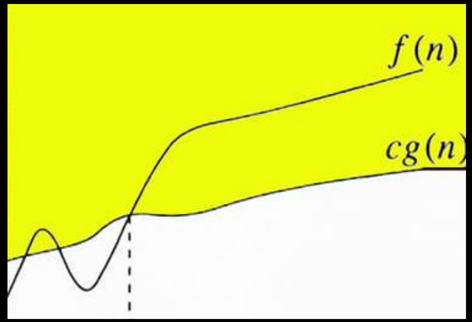
נאמר ש- $f(n) \in \Omega(g(n))$

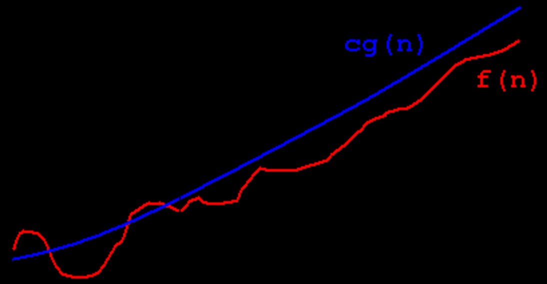
$$\Updownarrow$$

$$\mathbf{0} < \lim_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| < \infty$$

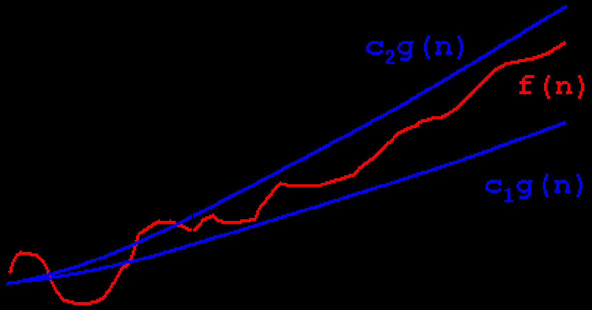
צבי מינץ

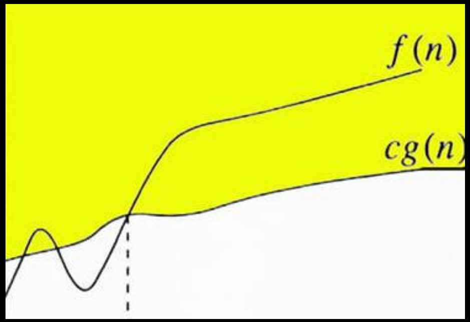


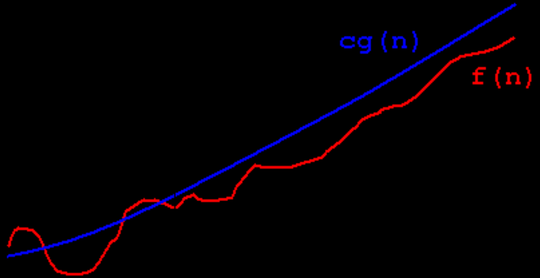




$\boldsymbol{f(n) \in \Theta(g(n))}$ חסם הדוק	$\boldsymbol{f(n) \in \Omega(g(n))}$ חסם תחתון	$\boldsymbol{f(n) \in O(g(n))}$ חסם עליון
$\begin{matrix} \exists c_1 > 0 \\ \exists c_2 > 0 \end{matrix}, n_0 \geq 0 : \ \forall n > n_0$ $c_2 \cdot g(n) \geq f(n) \geq c_1 \cdot g(n) \geq 0$	$\exists c > 0, n_0 \geq 0 : \ \forall n > n_0$ $f(n) \geq c \cdot g(n) \geq 0$	$\exists c > 0, n_0 \geq 0 : \ \forall n > n_0$ $f(n) \leq c \cdot g(n)$
$0 < \lim\limits_{n\to\infty} \frac{f(n)}{g(n)} < \infty$	$0 < \lim\limits_{n\to\infty} \frac{f(n)}{g(n)} \leq \infty$	$0 \leq \lim\limits_{n\to\infty} \frac{f(n)}{g(n)} < \infty$

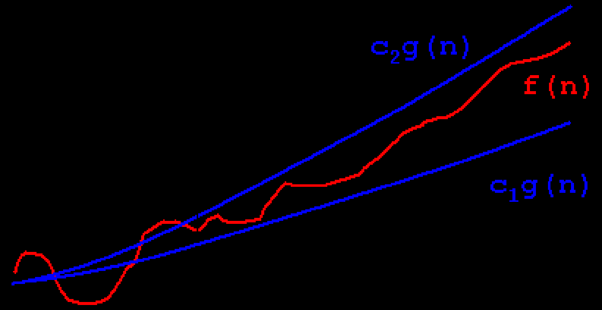


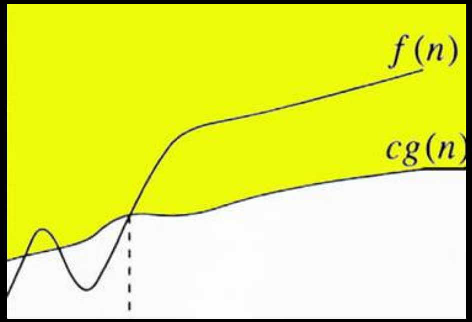


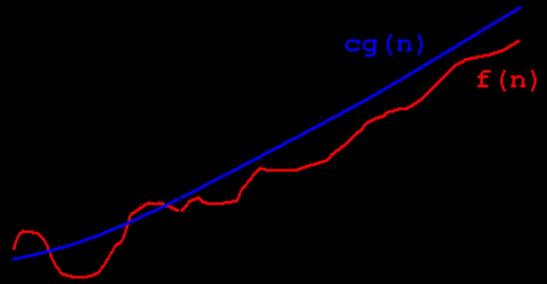


$f(n) \in \Theta(g(n))$ חסם הדוק	$f(n) \in \Omega(g(n))$ חסם תחתון	$f(n) \in O(g(n))$ חסם עליון
$\begin{matrix} \exists c_1 > 0 \\ \exists c_2 > 0 \end{matrix}, n_0 \geq 0 : \ \forall n > n_0$ $c_2 \cdot g(n) \geq f(n) \geq c_1 \cdot g(n) \geq 0$	$\exists c > 0, n_0 \geq 0 : \ \forall n > n_0$ $f(n) \geq c \cdot g(n) \geq 0$	$\exists c > 0, n_0 \geq 0 : \ \forall n > n_0$ $f(n) \leq c \cdot g(n)$
$0 < \lim\limits_{n\to\infty} \frac{f(n)}{g(n)} < \infty$	$0 < \lim\limits_{n\to\infty} \frac{f(n)}{g(n)} \leq \infty$	$0 \leq \lim\limits_{n\to\infty} \frac{f(n)}{g(n)} < \infty$ $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} = 0$

צבי מינץ

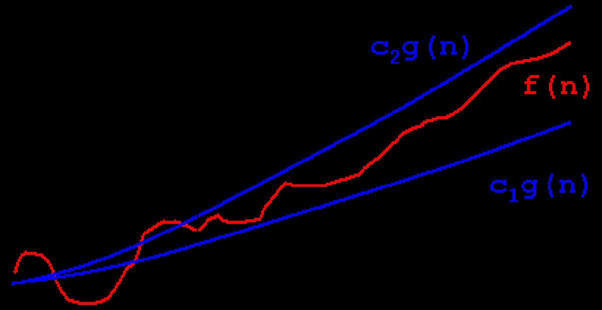


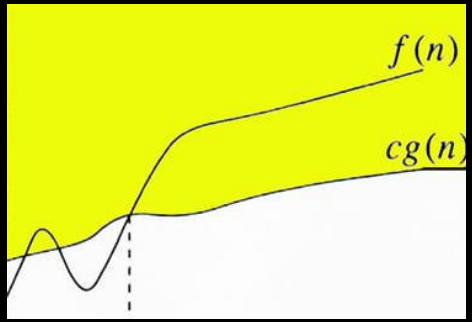


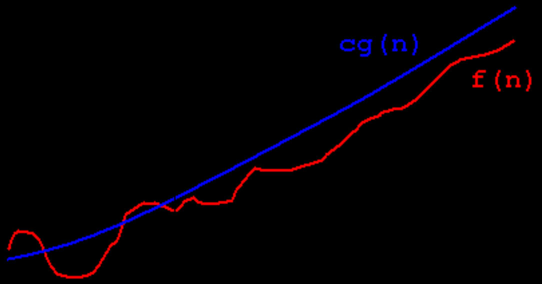


$\boldsymbol{f(n) \in O(g(n))}$ חסם עליון	$\boldsymbol{f(n) \in \Omega(g(n))}$ חסם תחתון	$\boldsymbol{f(n) \in \Theta(g(n))}$ חסם הדוק
$\exists c > 0, n_0 \geq 0 : \forall n > n_0$ $f(n) \leq c \cdot g(n)$	$\exists c > 0, n_0 \geq 0 : \forall n > n_0$ $f(n) \geq c \cdot g(n) \geq 0$	$\begin{matrix} \exists c_1 > 0 \\ \exists c_2 > 0 \end{matrix}, n_0 \geq 0 : \forall n > n_0$ $c_2 \cdot g(n) \geq f(n) \geq c_1 \cdot g(n) \geq 0$
$0 \leq \lim\limits_{n\to\infty} \frac{f(n)}{g(n)} < \infty$ $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} = 0$	$0 < \lim\limits_{n\to\infty} \frac{f(n)}{g(n)} \leq \infty$ $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} = \infty$	$0 < \lim\limits_{n\to\infty} \frac{f(n)}{g(n)} < \infty$

צבי מינץ







$f(n) \in \Theta(g(n))$ חסם הדוק	$f(n) \in \Omega(g(n))$ חסם תחתון	$f(n) \in O(g(n))$ חסם עליון
$\begin{matrix}\exists c_1 > 0 \\ \exists c_2 > 0\end{matrix}, n_0 \geq 0 : \ \forall n > n_0$ $c_2 \cdot g(n) \geq f(n) \geq c_1 \cdot g(n) \geq 0$	$\exists c > 0, n_0 \geq 0 : \ \forall n > n_0$ $f(n) \geq c \cdot g(n) \geq 0$	$\exists c > 0, n_0 \geq 0 : \ \forall n > n_0$ $f(n) \leq c \cdot g(n)$
$0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$ $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \mathrm{c} : 0 < c < \infty$	$0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} \leq \infty$ $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$	$0 \leq \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$ $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

צבי מינץ

אלגוריתמי חיפוש

חיפוש לינארי

$\boldsymbol{LinearSearch}(\boldsymbol{A}, \boldsymbol{n}, \boldsymbol{key})$

1. $for\ i \leftarrow 1\ to\ n:$
2. $\quad if\ A[i] = key:$
3. $\quad\quad return\ i$
4. $return\ NULL$

חיפוש בינארי Binary Search

אלגוריתם למציאת מקום של איבר במערך **ממויין**

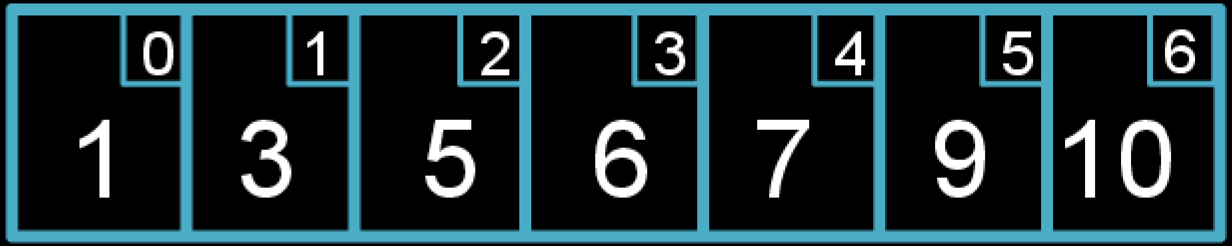
חיפוש_בינארי (תחילת_מערך, סוף_מערך, ערך_מבוקש)

1. אמצע -> 2 / (תחילת_מערך + סוף_מערך)
2. אם מערך[אמצע] = ערך_מבוקש
    1. החזר אמצע
3. אחרת,
    1. אם מערך[אמצע] < ערך_מבוקש
        1. חיפוש_בינארי (אמצע + 1, סוף_מערך, ערך_מבוקש)
    2. אחרת,
        1. חיפוש_בינארי (תחילת_מערך, אמצע - 1, ערך_מבוקש)

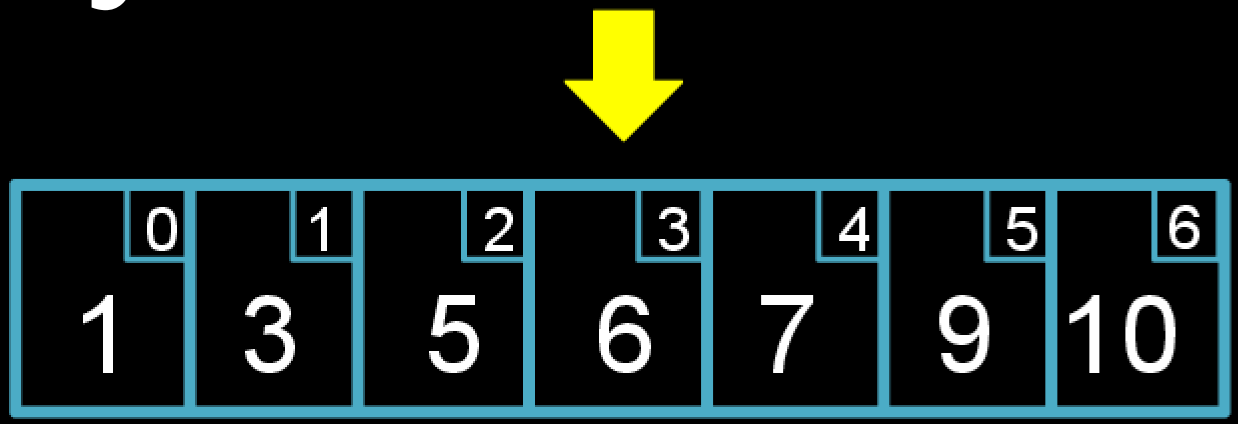
חיפוש בינארי Binary Search

אלגוריתם למציאת מקום של איבר במערך **ממויין**

Does the array contain 7?

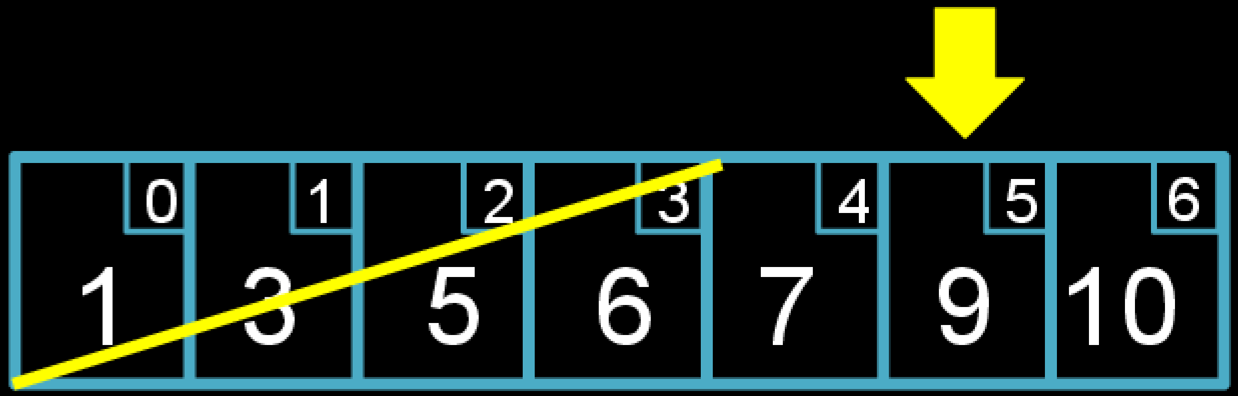


חיפוש בינארי Binary Search



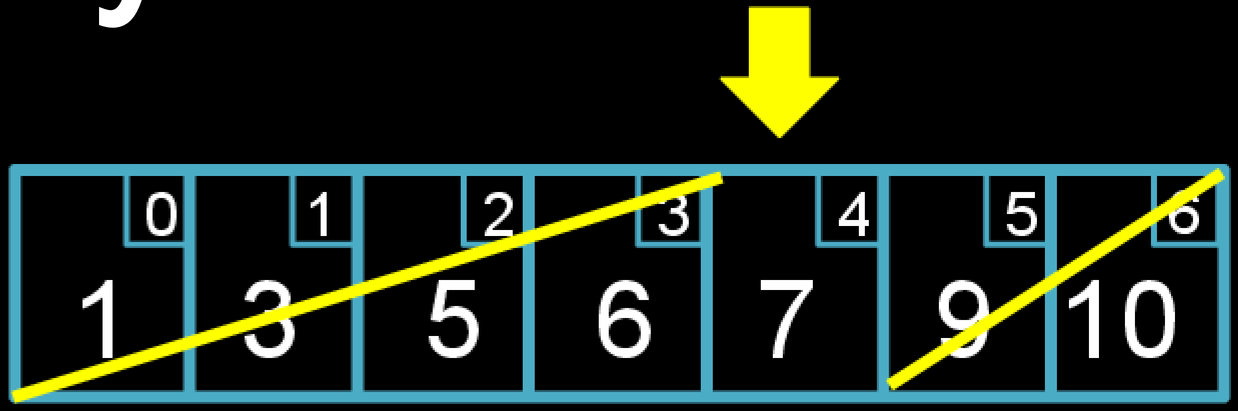
Is array[3] == 7?
Is array[3] < 7?
Is array[3] > 7?

Binary Search חיפוש בינארי



Is array[5] == 7?
Is array[5] < 7?
Is array[5] > 7?

Binary Search חיפוש בינארי



Is array[4] == 7?
Is array[4] < 7?
Is array[4] > 7?

מימוש Binary Search

מימוש רקורסיבי

```
public int runBinarySearchRecursively( int[] sortedArray, int key, int low, int high) {

    int middle = (low + high) / 2;

    if (high < low) { return -1; } // תנאי עצירה

    if (key == sortedArray[middle])  { return middle; } // תנאי עצירה

    else if (key < sortedArray[middle]) { // חיפוש בצד שמאל
        return runBinarySearchRecursively( sortedArray, key, low, middle - 1);
     }
    else { // חיפוש בצד ימין
        return runBinarySearchRecursively( sortedArray, key, middle + 1, high);
    }
}
```

מימוש Binary Search

בעזרת Arrays.BinarySearch()

```
int[] sortedArray = {1,2,3};
int key = 5; // Prints Not Found
int index = Arrays.binarySearch(sortedArray, key);
System.out.println((index > 0) ? index : "Not Found");
```

ניתוח זמן ריצה של חיפוש בינארי

- נסמן ב-$T(n)$ את זמן ריצה של אלגוריתם חיפוש בינארי

כמות תתי בעיות

ולכן:

$$T(n) = 1 \cdot T\left(\frac{n}{2}\right) + f(n)$$

$f(n)$ מציין את כמות העבודה בכל שלב ברקורסיה

ניתוח זמן ריצה של חיפוש בינארי

```
public int runBinarySearchRecursively( int[] sortedArray, int key, int low, int high) {

    int middle = (low + high) / 2;  O(1)

    if (high < low) { return -1; } // תנאי עצירה  O(1)

    if (key == sortedArray[middle])  { return middle; } // תנאי עצירה  O(1)

    else if (key < sortedArray[middle]) { // חיפוש בצד שמאל
        return runBinarySearchRecursively( sortedArray, key, low, middle - 1);  O(1)
    }
    else { // חיפוש בצד ימין
        return runBinarySearchRecursively( sortedArray, key, middle + 1, high);   O(1)
    }
}
```

ולכן:

$$T(n) = \begin{cases} T\left(\frac{n}{2}\right) + c, & n \geq 2 \\ c, & n = 1 \end{cases}$$

ניתוח זמן ריצה של חיפוש בינארי

$$T(n) = \begin{cases} T\left(\frac{n}{2}\right) + c, & n \geq 2 \\ c, & n = 1 \end{cases}$$

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

$$= \left[T\left(\frac{n}{4}\right) + c\right] + c = T\left(\frac{n}{4}\right) + 2c$$

$$= \left[T\left(\frac{n}{8}\right) + c\right] + 2c = T\left(\frac{n}{8}\right) + 3c$$

$$\cdots$$

$$= T\left(\frac{n}{2^k}\right) + k \cdot c$$

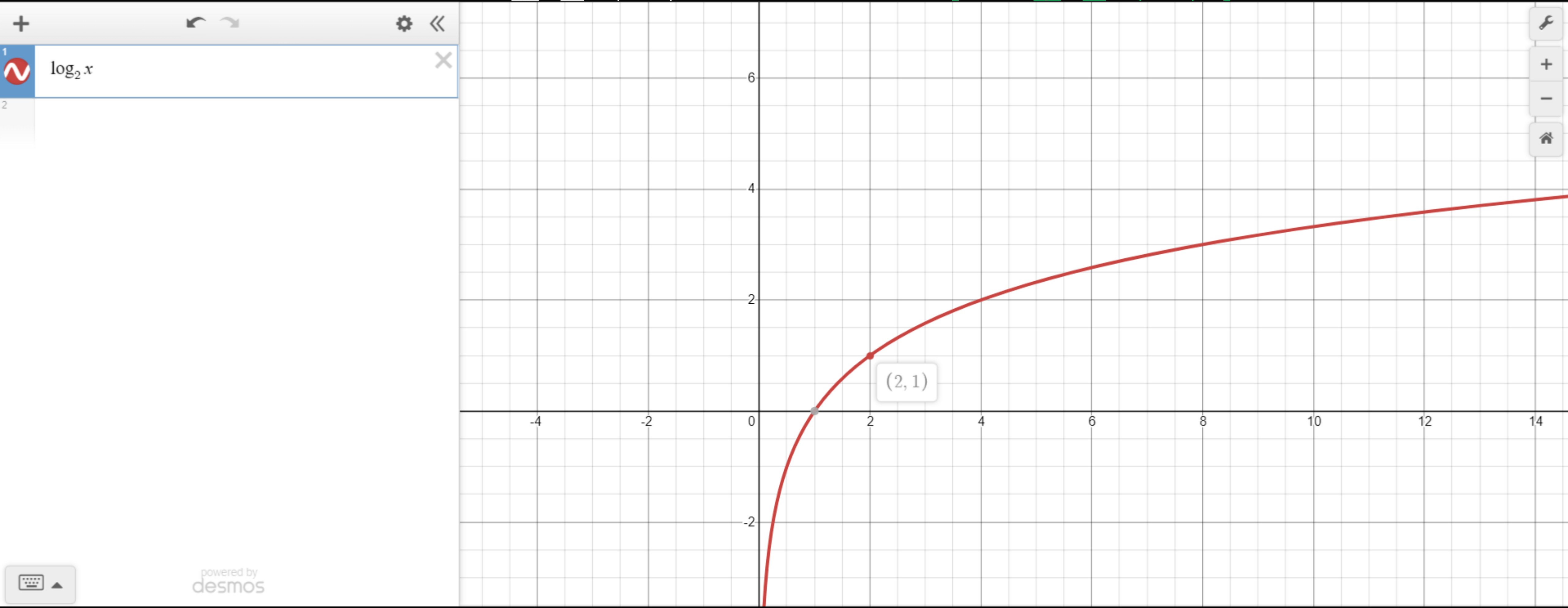
נמצא את k עבורו נגיע לתנאי עצירה $(\frac{n}{2^k} = 1)$

$$\frac{n}{2^k} = 1 \Rightarrow n = 2^k \Rightarrow k = log_2(n)$$

נציב $k = log_2(n)$ ונקבל:

$$T(n) = T(1) + log_2(n) \cdot c = c + log_2(n) \cdot c = O(?)$$

$$c + log_2(n) \cdot c = O(log_2(n))$$



חיפוש בינארי ללא אינדקסים

```
function binarySearch(A, x):
   // Input: A, a sorted array
   //          x, the item to find
   // Output: true if x is in A, else false

    if A.size == 0:          O(1)  (base case 1)
       return false          O(1)

    if A.size == 1:          O(1)  (base case 2)
       return A[0] == x      O(1)

    mid = A.size / 2 O(1)

    if A[mid] == x:   O(1)
       return true    O(1)
    if A[mid] < x:    O(1)
       return binarySearch(A[mid + 1...end], x)
    if A[mid] > x:    O(1)
       return binarySearch(A[0...mid - 1], x)
```

סיבוכיות:

$$T(n) = T\left(\frac{n}{2}\right) + c_1 n + c2$$

$$\Rightarrow O(n)\ !$$

חוקי חזקות ולוגרתמים שימושיים:

$$a^{-k} = \frac{1}{a^k}$$

$$(a^m)^n = a^{m \cdot n}$$

$$a^m \cdot a^n = a^{m+n}$$

$$log_c(a \cdot b) = log_c a + log_c b$$

$$log_c a^n = n log_c a$$

$$log_b a = \frac{log_c a}{log_c b}$$

$$log_b \left(\frac{1}{a}\right) = -log_b a$$

$$\boldsymbol{a^{log_b c} = c^{log_b a}}$$

$$a = b^{log_b a}$$

אוניברסיטת
אריאל
בשומרון

עבודה עצמית

צבי מינץ

Special credit to **CS50,** HarvardX

https://study.cs50.net/