

# מיונים וחיפוש



קורן כליר

# ביחידה זו נלמד:

---

- מהו מיון
- מוטיבציה למיון
- מיון בועות (Bubble Sort)
  - מיון בסדר עולה, מיון בסדר יורד, מיון טקסט
- Selection Sort
- Insertion Sort
- חיפוש בינארי
  - מימוש איטרטיבי
  - מימוש רקורסיבי
- מיזוג מערכים ממוינים

# מהו מיון

□ מיון הוא סידור איברי קבוצה מסוימת בסדר עולה או בסדר יורד

□ הקבוצה היחידה שאנו מכירים עד כה היא מערך, לכן נראה כיצד ממיינים מערך

3	1	9	2
---	---	---	---

מעריך לא ממוין:

1	2	3	9
---	---	---	---

מעריך ממוין:

# מוטיבציה למיון – שיקולי יעילות בחיפוש ערך מסוים

□ כדי למצוא את האיבר המינימאלי במערך, צריך לסרוק את כל האיברים (חיפוש לינארי)

■ יעילות:  $O(n)$

■ אם המערך ממין צריך לבדוק מהו ערכו של האיבר הראשון בלבד

□ יעילות:  $O(1)$

□ כדי למצוא את האיבר המקסימאלי במערך, צריך לסרוק את כל האיברים (חיפוש לינארי)

■ יעילות:  $O(n)$

■ אם המערך ממין צריך לבדוק מהו ערכו של האיבר האחרון בלבד

□ יעילות:  $O(1)$

# מוטיבציה למיון – שיקולי יעילות בחיפוש ערך

## מסוים (2)

---

□ כדי למצוא איבר כלשהו, צריך לסרוק את כל האיברים

■ יעילות:  $O(n)$

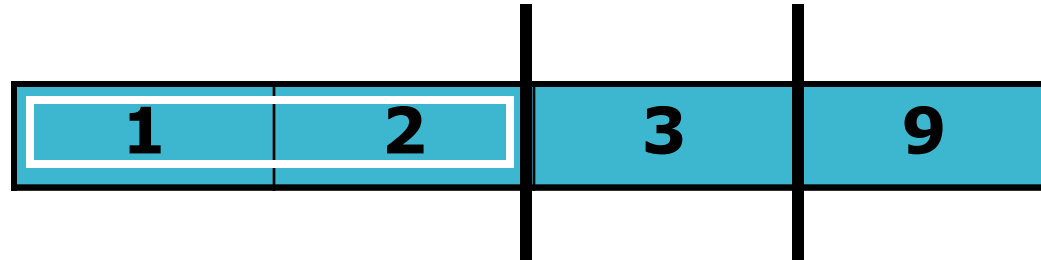
■ אם המערך ממין ניתן למצוא את האיבר ב-  $O(\log(n))$   
שזה שיפור בסדר גודל שלם!

□ כדי למצוא איבר במערך ממין בסדר גודל של  
 $O(\log(n))$  נשתמש בחיפוש בינארי

# מיון בועות

- הרעיון מאחורי השם הוא שמפעפעים את האיבר המתאים למעלה ע"י החלפות
- הרעיון מאחורי מיון בועות (עבור מיון מהערך הקטן לגדול):
  - באיטרציה הראשונה נפעפע את האיבר המקסימלי לסוף המערך ונמקם אותו במקום ה-  $n-1$  ע"י החלפות
  - באיטרציה השניה נפעפע את האיבר המקסימלי מבין האיברים 0 עד  $n-2$  ונמקם אותו במקום ה-  $n-2$  ע"י החלפות
  - וכו' עד סוף המערך
- המצב בסיום האיטרציה ה-  $k$  שיש לנו את  $k$  האיברים הגדולים ביותר במערך ממוינים ב-  $k$  האיברים האחרונים

# מיון בועות – דוגמת הרצה



אם האיבר השמאלי גדול מהימני, נחליף ביניהם

החלף  $3 > 1$  ?

$3 > 9$  ?

החלף  $9 > 2$  ?

המצב כעת הוא שהאיבר המקסימלי אכן נמצא בסוף, ונתייחס רק למערך שמשמאל לקו

$1 > 3$  ?

החלף  $3 > 2$  ?

המצב כעת הוא ששני האיברים המקסימאליים אכן נמצאים ממוינים בסוף, ונתייחס רק למערך שמשמאל לקו

$1 > 2$  ?

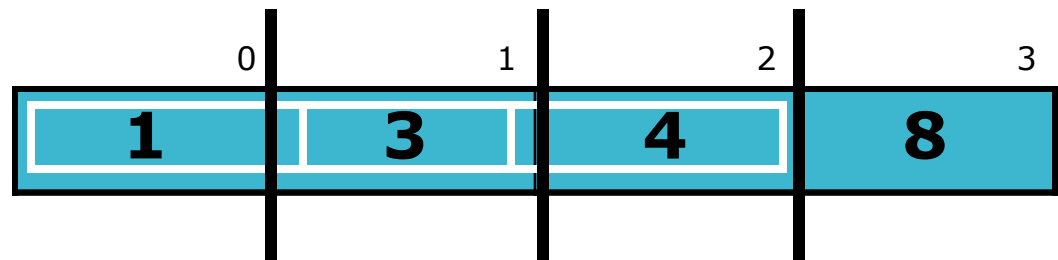
```
C:\WINDOWS\system32\cmd.exe
1 3 4 8
Press any key to continue . . . _
```

## מיון בועות - הקוד

```
void main()
{
    int arr[] = {3,8,1,4};
    int size = sizeof(arr)/sizeof(arr[0]);
    int i, j, temp;
    for ( i=size-1 ; i > 0 ; i-- )
    {
        for ( j=0 ; j < i ; j++ )
        {
            if (arr[j] > arr[j+1])
            {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }

    for (i=0 ; i < size ; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
```

int[]: arr	<b>1</b>	<b>1000</b>
	<b>3</b>	<b>1004</b>
	<b>4</b>	<b>1008</b>
	<b>8</b>	<b>1012</b>
int: size	<b>4</b>	<b>1016</b>
int: i	<b>0</b>	<b>1020</b>
int: j	<b>1</b>	<b>1024</b>
int: temp	<b>3</b>	<b>1028</b>



ניתוח זמן ריצה:  $O(\text{size}^2)$



# מיון בועות – קצת סדר ופונקציות (1)

```
#include <stdio.h>
```

```
void swap(int* a, int* b);
```

```
void main()
```

```
{
```

```
    int arr[] = {3,8,1,4};
```

```
    int i, j, size = sizeof(arr)/sizeof(arr[0]);
```

```
    for ( i=size-1 ; i > 0 ; i-- )
```

```
    {
```

```
        for ( j=0 ; j < i ; j++ )
```

```
        {
```

```
            if (arr[j] > arr[j+1])
```

```
                swap(&arr[j], &arr[j+1]);
```

```
        }
```

```
    }
```

```
    for (i=0 ; i < size ; i++)
```

```
        printf("%d ", arr[i]);
```

```
    printf("\n");
```

```
}
```

```
void swap(int* a, int* b)
```

```
{
```

```
    int temp = *a;
```

```
    *a = *b;
```

```
    *b = temp;
```

```
}
```

## מיון בועות – קצת סדר ופונקציות (2)

```
#include <stdio.h>
```

```
void sortAscending(int arr[], int size);  
void swap(int* a, int* b);
```

```
void main()  
{  
    int arr[] = {3,8,1,4};  
    int i, size = sizeof(arr)/sizeof(arr[0]);  
    sortAscending(arr, size);  
  
    for (i=0 ; i < size ; i++)  
        printf("%d ", arr[i]);  
    printf("\n");  
}
```

```
void swap(int* a, int* b)  
{  
    int temp = *a;  
    *a = *b;  
    *b = temp;  
}
```

```
void sortAscending(int arr[], int size)  
{  
    int i, j;  
    for ( i=size-1 ; i > 0 ; i-- )  
    {  
        for ( j=0 ; j < i ; j++ )  
        {  
            if (arr[j] > arr[j+1])  
                swap(&arr[j], &arr[j+1]);  
        }  
    }  
}
```

# מיון בועות – השינוי עבור מערך ממזין בסדר יורד


```
#include <stdio.h>
```

```
void sortDescending(int arr[], int size);
void swap(int* a, int* b);
```

```
void main()
{
    int arr[] = {3,8,1,4};
    int i, size = sizeof(arr)/sizeof(arr[0]);
    sortDescending(arr, size);

    for (i=0 ; i < size ; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
```

```
void swap(int* a, int* b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

```
void sortDescending(int arr[], int size)
{
    int i, j;
    for ( i=size-1 ; i > 0 ; i-- )
    {
        for ( j=0 ; j < i ; j++ )
        {
            if (arr[j]  arr[j+1])
                swap(&arr[j], &arr[j+1]);
        }
    }
}
```

כדי למיין את המערך בסדר יורד,  
נרצה לפעפע את האיבר הקטן ביותר לסוף.  
לכן השינוי היחידי הוא בתנאי לפני  
ההחלפה  
(וכמוכך ששם הפונקציה)

```
C:\WINDOWS\system32\cmd.exe
8 4 3 1
Press any key to continue .
```

# מיון בועות – מיון טקסט (1)

```
#define LINES 5
#define LEN 30

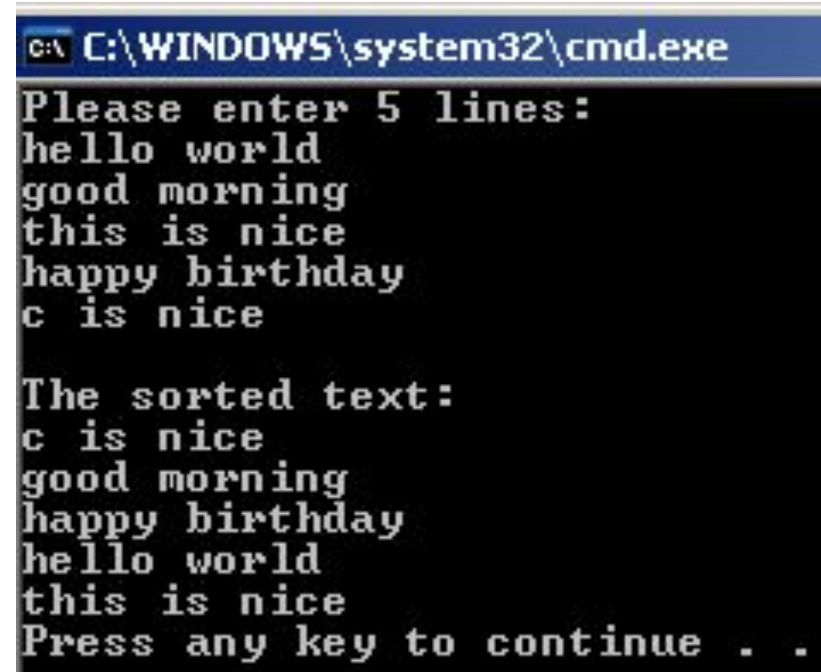
void swap(char* a, char* b);
void sortAscending(char text[][LEN], int lines);

void main()
{
    char text[LINES][LEN];
    int i;

    printf("Please enter %d lines:\n", LINES);
    for (i=0 ; i < LINES ; i++)
        gets(text[i]);

    sortAscending(text, LINES);

    printf("\nThe sorted text:\n");
    for (i=0 ; i < LINES ; i++)
        printf("%s\n", text[i]);
}
```



```
C:\WINDOWS\system32\cmd.exe
Please enter 5 lines:
hello world
good morning
this is nice
happy birthday
c is nice

The sorted text:
c is nice
good morning
happy birthday
hello world
this is nice
Press any key to continue . .
```

## מיון בועות – מיון טקסט (2)

---

```
void swap(char* a, char* b)
{
    char temp[LEN];
    strcpy(temp, a);
    strcpy(a, b);
    strcpy(b, temp);
}
```

```
void sortAscending(char text[][LEN], int lines)
{
    int i, j;
    for (i=lines-1 ; i > 0 ; i--)
    {
        for ( j=0 ; j < i ; j++ )
        {
            if (strcmp(text[j], text[j+1]) > 0)
                swap(text[j], text[j+1]);
        }
    }
}
```

# מיון בועות: אופטימיזציה

```
void sortAscending(int arr[], int size)
{
    int i, j;
    int hasChanged=1;
    for ( i=size-1 ; i > 0 && hasChanged ; i-- )
    {
        hasChanged = 0;
        for ( j=0 ; j < i ; j++ )
        {
            if (arr[j] > arr[j+1])
            {
                swap(&arr[j], &arr[j+1]);
                hasChanged = 1;
            }
        }
    }
}
```

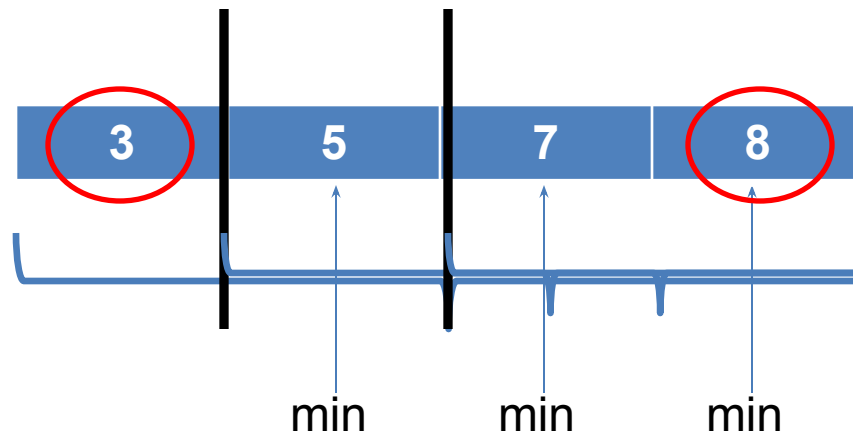
התוספת: לאחר איטרציה ללא כל  
החלפה, ברור כי המערך כבר ממוין ולכן

# Selection Sort

במיון זה מחפשים באיטרציה הראשונה את הערך המינימלי ושמים אותו באינדקס ה-0, ובמקומו שמים את הערך שהיה באינדקס ה-0

באיטרציה השניה מחפשים את הערך הנמוך ביותר החל מאינדקס 1, ומחליפים אותו עם הערך באינדקס 1

וכו' □



# Selection Sort

- הקוד

```
void swap(int* a, int* b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

void selectionSort(int arr[], int size)
{
    int i,j, minIndex;
    for (i=0 ; i < size ; i++)
    {
        minIndex = i;
        for (j=i+1 ; j < size ; j++)
        {
            if (arr[j] < arr[minIndex])
                minIndex = j;
        }
        swap(&arr[i], &arr[minIndex]);
    }
}

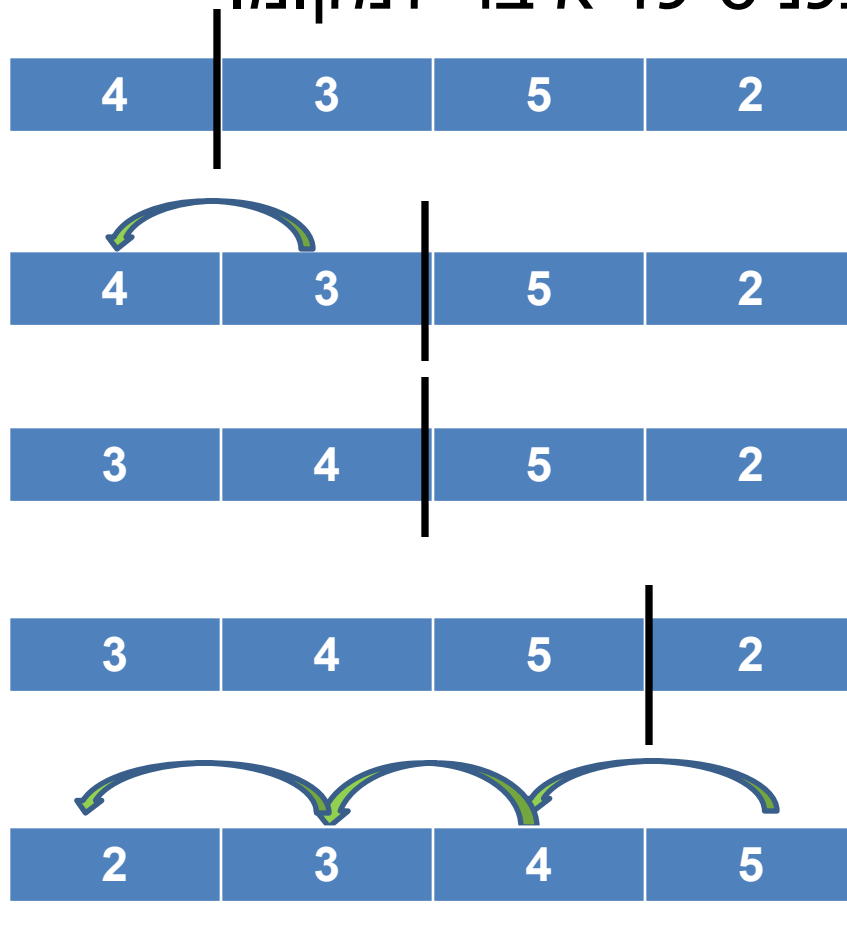
void main()
{
    int arr[] = {5,2,8,1,3};
    selectionSort(arr, sizeof(arr)/sizeof(arr[0]));
}
```

יעילות: סכום של סדרה חשבונית ולכן  
 $O(size^2)$



# Insertion Sort

הרעיון: נכניס כל איבר למקומו



# הקוד - Insertion Sort

```
void insertionSort(int arr[], int size)
{
    int i,j;
    for (i=1 ; i < size ; i++)
    {
        for (j=i ; j > 0 && arr[j-1] > arr[j] ; j--)
        {
            int temp = arr[j];
            arr[j] = arr[j-1];
            arr[j-1] = temp;
        }
    }
}
```

יעילות: סכום של סדרה חשבונית ולכן  
 $O(\text{size}^2)$

```
void main()
{
    int arr[] = {7,1,8,2,3,6,4,5};
    insertionSort(arr, sizeof(arr)/sizeof(arr[0]));
}
```

# חיפוש בינארי

- מטרתו של חיפוש בינארי היא למצוא את מיקומו של איבר במערך ממוין
- לא ניתן להפעיל חיפוש בינארי על מערך לא ממוין!
- הגדרתו:

```
int binarySearch(int arr[], int size, int value)
```

■ הפונקציה מקבלת:

□ מערך, גודלו וערך לחיפוש

■ הפונקציה מחזירה:

□ את האינדקס של האיבר, או -1 אם לא נמצא

# חיפוש בינארי - הרעיון

□ כל עוד יש במערך איברים:

■ נבדוק האם האיבר שאנחנו מחפשים הוא האמצעי

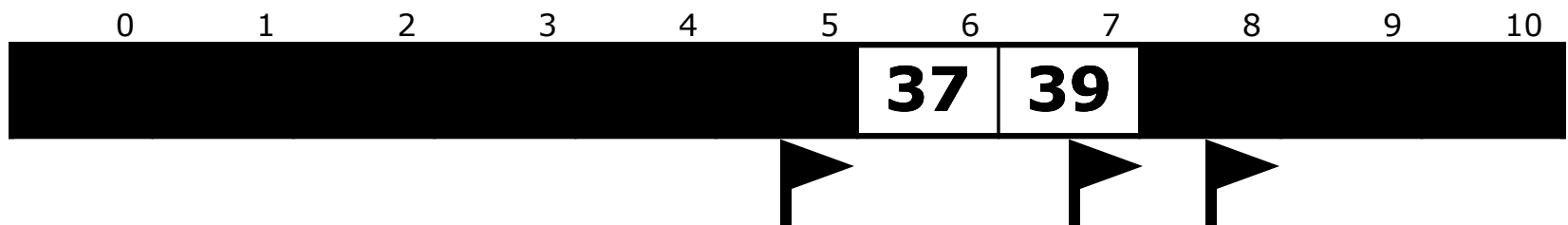
□ אם כן, נחזיר את האינדקס שלו

□ אחרת אם הוא קטן מהאיבר האמצעי, נחפש אותו בחצי המערך השמאלי

□ אחרת, נחפש אותו בחצי המערך הימני

□ נחזיר 1- (לא מצאנו..)

□ דוגמא, נחפש את הערך 39 במערך:



# חיפוש בינארי - ניתוח זמן ריצה

---

- בכל איטרציה מספר הפעולות קבוע
- בכל איטרציה גודל הקלט שלנו קטן פי 2, לכן יהיו לנו  $\log(n)$  איטרציות
- בסה"כ זמן החישוב במקרה הגרוע ביותר הוא  $O(\log(n))$ , לעומת  $O(n)$  בחיפוש לינארי
- שמו של חיפוש בינארי בא מכך שאנו מטפלים בכל איטרציה ב-  $1/2$  מהקלט (בינארי=2)

```
#define NOT_FOUND -1
```

```
int binarySearch(int a[], int size, int val)
{
    int low=0, high=size-1, middle;
    while (low <= high)
    {
        middle = (low + high)/2;
        if (val == a[middle])
            return middle;
        else if (val < a[middle])
            high = middle - 1;
        else
            low = middle + 1;
    }
    return NOT_FOUND;
}
```

```
void main()
{
    int arr[] = {1, 4, 7, 9, 12, 16}, index, value;
    int size = sizeof(arr)/sizeof(arr[0]);

    printf("Please enter the value to search: ");
    scanf("%d", &value);
    index =
    if (index = binarySearch(arr, size, value);
        printf("The value %d doesn't exist in the array", value);
    else
        printf("The value %d exists in index %d\n", value, index);
}
```

# חיפוש בינארי –

## הקוד (חיפוש 4)

int*: a	1000	2000
int: size	6	2004
int: val	4	2008

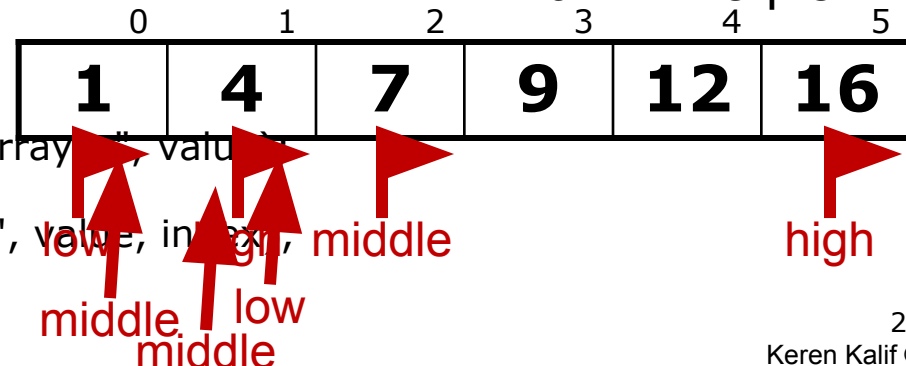
int[]: arr	1	1000
	4	1004
	7	1008

```
int: C:\WINDOWS\system32\cmd.exe
Please enter the value to search: 4
The value 4 exists in index 1
Press any key to continue . . .
```

הזיכרון של binarySearch

int: size	6	1024
int: index	1	1028
int: value	4	1032

הזיכרון של ה-main



```
#define NOT_FOUND -1
```

```
int binarySearch(int a[], int size, int val)
{
    int low=0, high=size-1, middle;
    while (low <= high)
    {
        middle = (low + high)/2;
        if (val == a[middle])
            return middle;
        else if (val < a[middle])
            high = middle - 1;
        else
            low = middle + 1;
    }
    return NOT_FOUND;
}
```

```
void main()
{
    int arr[] = {1, 4, 7, 9, 12, 16}, index, value;
    int size = sizeof(arr)/sizeof(arr[0]);

    printf("Please enter the value to search: ");
    scanf("%d", &value);
    index =
    if (index = binarySearch(arr, size, value);
        printf("The value %d doesn't exist in the array", value);
    else
        printf("The value %d exists in index %d\n", value, index);
}
```

```
Please enter the value to search: 10
The value 10 doesn't exist in the array
```

# חיפוש בינארי –

## הקוד (חיפוש 10)

int*: a	1000	2000
int: size	6	2004
int: val	10	2008
int: low	4	2012
int: high	3	2016
int: middle	3	2020

הזיכרון של binarySearch

int[]: arr	1	1000
	4	1004
	7	1008
	9	1012
	12	1016
	16	1020
int: size	6	1024
int: index	1-	1028
int: value	10	1032

הזיכרון של ה-main

0	1	2	3	4	5
1	4	7	9	12	16



# חיפוש בינארי - מימוש רקורסיבי

□ כדי לממש את החיפוש הבינארי בקוד רקורסיבי נמצא את מרכיבי הרקורסיה בבעיה:

■ תנאי עצירה:

□ אם המערך בגודל 0, נחזיר NOT\_FOUND

■ קישור:

□ אם האיבר האמצעי שווה לערך שאנו מחפשים, נחזיר את האינדקס שלו

■ קריאה רקורסיבית:

□ אם האיבר שאנחנו מחפשים קטן מהאיבר האמצעי, נחפש בתת-המערך השמאלי

□ אחרת, נחפש בתת-המערך הימני



```
#define NOT_FOUND -1
```

חיפוש בינארי – `int binarySearch(int arr[], int low, int high, int value)`

```
{
    int middle = (low + high)/2;
    if (low > high) return NOT_FOUND;

    if (value == arr[middle])
        return middle;
    else if (value < arr[middle])
        return binarySearch(arr, low, middle-1, value);
    else
        return binarySearch(arr, middle+1, high, value);
}
```

הקוד הרקורסיבי

ניתוח זמן הריצה:  
 $O(\log(\text{size}))$

`binS (1000, 0, 5, 4)`

```
void main()
{
    int arr[] = {1, 4, 7, 9, 12, 16};
    int size = sizeof(arr)/sizeof(arr[0]), index, value;

    printf("Please enter the value to search: ");
    scanf("%d", &value);
    index = binarySearch(arr, 0, size-1, value);

    if (index == NOT_FOUND)
        printf("The value %d doesn't exist in the array\n", value);
    else
        printf("The value %d exists in index %d\n", value, index);
}
```

# מיזוג מערכים ממוינים

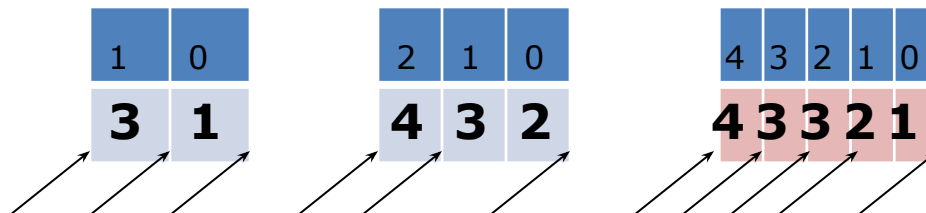
□ כל עוד יש איברים בשני המערכים:

■ נבדוק האם האיבר הנוכחי במערך הראשון קטן/שווה מהאיבר הנוכחי במערך השני:

□ אם כן, נעתיק איבר נוכחי מהמערך הראשון לנוכחי בשלישי ונקדם את מיקום האיבר הנוכחי

□ אחרת, נעתיק איבר נוכחי מהמערך השני לנוכחי בשלישי ונקדם את מיקום האיבר הנוכחי

רק אחד מהם יקרה... { נעתיק את כל מה שנותר מהמערך הראשון לשלישי  
נעתיק את כל מה שנותר מהמערך השני לשלישי



# מיזוג מערכים ממוינים - הקוד

```
void mergeArrays(const int arr1[], int size1,  
                const int arr2[], int size2, int arr3[])
```

```
{
```

```
    int i=0, j=0;
```

```
    while ( i < size1 && j < size2 )
```

```
    {
```

```
        if (arr1[i] <= arr2[j])
```

```
            arr3[i+j] = arr1[i++];
```

```
        else
```

```
            arr3[i+j] = arr2[j++];
```

```
    {
```

```
    while ( i < size1) // copy rest of arr1
```

```
        arr3[i+j] = arr1[i++];
```

```
    while ( j < size2) // copy rest of arr2
```

```
        arr3[i+j] = arr2[j++];
```

```
}
```

**i = 0**

**j = 0**

arr1:

1	0
<b>3</b>	<b>1</b>

arr2:

2	1	0
<b>4</b>	<b>3</b>	<b>2</b>

arr3:

4	3	2	1	0
<b>4</b>	<b>3</b>	<b>3</b>	<b>2</b>	<b>1</b>

ניתוח זמן ריצה:  $O(\text{size1} + \text{size2})$

# מיזוג מערכי מחרוזות ממוינים

```
void mergeArrays(const char arr1[][LEN], int size1,  
                const char arr2[][LEN], int size2, char arr3[][LEN])  
{  
    int i=0, j=0;  
  
    while ( i < size1 && j < size2 )  
    {  
        if ( strcmp(arr1[i], arr2[j]) <= 0 )  
        {  
            strcpy(arr3[i+j], arr1[i]);  
            i++;  
        }  
        else  
        {  
            strcpy(arr3[i+j], arr2[j]);  
            j++;  
        }  
    }  
}
```

```
text 1:  
bbb  
ccc  
  
text 2:  
aaa  
ddd  
mmm  
  
text 3:  
aaa  
bbb  
ccc  
ddd  
mmm
```

```
while ( i < size1) // copy rest of arr1  
{  
    strcpy(arr3[i+j], arr1[i]);  
    i++;  
}  
  
while ( j < size2) // copy rest of arr2  
{  
    strcpy(arr3[i+j], arr2[j]);  
    j++;  
}  
}
```

# ביחידה זו למדנו:

---

- מהו מיון
- מוטיבציה למיון
- מיון בועות (Bubble Sort)
  - מיון בסדר עולה, מיון בסדר יורד, מיון טקסט
- Selection Sort
- Insertion Sort
- חיפוש בינארי
  - מימוש איטרטיבי
  - מימוש רקורסיבי
- מיזוג מערכים ממוינים

# תרגיל 1:

□ כתוב פונקציה המקבלת מטריצה ריבועית, אשר תמיין את ערכי האלכסון הראשי מהקטן לגדול

□ דוגמא:

עבור המטריצה

9	3	6	5
2	4	8	2
7	2	2	1
9	8	3	6

יש לעדכנה להיות

2	3	6	5
2	4	8	2
7	2	6	1
9	8	3	9

הפתרון יוצג בפתרון בועות, אתם יכולים למיין בכל אלגוריתם אחר

## תרגיל 2:

□ כתוב פונקציה המקבלת מחרוזת הכוללת אותיות קטנות וגדולות

□ הפונקציה תמיין את אותיות המחרוזת מהאות הקטנה לגדולה, אך לא תבדיל בין אותיות גדולות לקטנות

□ דוגמאות:

■ עבור המחרוזת zBa הפונקציה תמיין אותה להיות aBz

■ עבור הפונקציה zZaAC הפונקציה תמיין אותה להיות aACzZ

□ הנחה: כאשר יש אות גדולה וקטנה זהה, לא משנה מי תופיע קודם

## תרגיל 3:

□ כתוב פונקציה המקבלת מטריצה ריבועית שעמודותיה ממוינות וערך לחיפוש. הפונקציה תחזיר את השורה והעמודה בה נמצא הערך, הוא תשים בערכם 1- אם הערך אינו קיים.

2	1	5	1
3	6	6	2
8	7	7	3
9	8	9	4

□ למשל, המטריצה הבאה והערך 7:

□ הפונקציה תחזיר שורה 2 ועמודה 1 כי במיקום זה קיים הערך 7