מציאת מקסימום ומקסימום במערך

תיאור הבעיה

נתון מערך A בעל n איברים, יש למצוא את 2 הערכים הכי מקסימלים במערך.

לצורך הנחה כללית לנושא זה: $max1 > max2$.

5	7	8	4	9	14	3	52	2	16
							max1		**max2**

פתרון הבעיה

עבור אלגוריתם זה ישנם 4 אפשרויות, כל אפשרות מציגה סיבוכיות שונה,

המטרה היא להשתמש **בכמות השוואות קטנה** ככל הניתן כדי לייעל את סיבוכיות האלגוריתם.

- אפשרות א':
  - נפעיל את האלגוריתם max המוכר שהצגנו למציאת מקסימום.
  - נמחק את הערך שיצא מהמערך.
  - נפעיל שוב את האלגוריתם max למציאת מקסימום נוסף.
  - נחזיר את התוצאות.

```
/**
* Comparisons : 2n - 3
* @param arr a numeric array.
* @return the two maximum numbers in the array.
*/
public static int[] maxMax(int[] arr){
   int maxIndex = maximum(arr); // n-1 comparisons
   int max1 = arr[maxIndex];
   arr[maxIndex] = Integer.MIN_VALUE;
   maxIndex = maximum(arr); // n-2 comparisons
   int max2 = arr[maxIndex];
   return new int[] {max1, max2};
}

/**
* a simple method to return the maximum value of the given array.
* @param arr a numeric array.
* @return the maximum value in this array.
*/
public static int maximum(int[] arr) {
   int max =  arr[0];
```

```
    int index = 0;
    for(int i = 1 ; i < arr.length ; i++) {
        if(max < arr[i]) {
            max = arr[i];
            index = i;
        }
    }
    return index;
}

public static void main(String[] args) {
    int[] arr = {84,31,3,1,567,4,2,93202,32,3};
    System.out.println(Arrays.toString(maxMax(arr)));
}
```

↜ **סיבוכיות השוואה:** סה"כ נקבל $2n-3$ השוואות.

❖ אפשרות ב':

⇚ נצמצם עוד השוואה בעזרת קביעת max1,max2 בהתחלה.

```
/**
* Not so different from the first solution...
* Comparisons : 2n - 3
* @param arr a numeric array.
* @return the two maximum numbers in the array.
*/
public static int[] maxMax(int[] arr) {
    int max1 = arr[0];
    int max2 = arr[1];

    if(max2 > max1) { // 1 comparison
        max1 = arr[1];
        max2 = arr[0];
    }

    for(int i = 2; i < arr.length; i++) {

        if(max1 < arr[i]) { // n-2 comparisons
            max2 = max1;
            max1 = arr[i];
        }
        else if(max2 < arr[i]) { // n-2 comparisons
                max2 = arr[i];
        }

    } // end for
```

```
    return new int[] { max1, max2 };
}

public static void main(String[] args) {
    int[] arr = {84,31,3,1,567,4,2,93202,32,3};
    System.out.println(Arrays.toString(maxMax(arr)));
}
```

סיבוכיות השוואה: סה"כ נקבל $2n-3$ השוואות.

- אפשרות ג' - נעבור על המערך בזוגות:
  - נבדוק מי יותר גדול $a[i], a[i+1]$.
  - נבדוק מה הערך החדש של max1 (או שהוא לא משתנה).
  - יש לנו 2 מועמדים ל-max2: שהם 2 אלו שיותר קטנים מ-max1 החדש.
  - נמצא את max2.
  - נמשיך הלאה לזוג הבא..

```
public static int[] maxMax(int[] arr) {
    int max1 = arr[0];
    int max2 = arr[1];

    if(max2 > max1) {
        max1 = arr[1];
        max2 = arr[0];
    }

    for(int i = 2 ; i < arr.length -1 ; i += 2) {

        if(arr[i] > arr[i+1]) {

            if(arr[i] > max1) {
                if(arr[i+1] > max1) {
                    max2 = arr[i+1];
                } else {
                    max2 = max1;
                }

                max1 = arr[i];
            }
            else if(arr[i] > max2) {
                max2 = arr[i];
            }
        }
        else { // arr[i] < arr[i+1]
```

```
            if(arr[i+1] > max1) {
                if(arr[i] > max1) {
                    max2 = arr[i];
                } else {
                    max2 = max1;
                }

                max1 = arr[i+1];
            }
            else if(arr[i+1] > max2) {
                max2 = arr[i+1];
            }
        }
    }

    if(arr.length%2 != 0) {
        if(arr[arr.length-1] > max1) {
            max2 = max1;
            max1 = arr[arr.length-1];
        } else if(arr[arr.length-1] > max2) {
            max2 = arr[arr.length-1];
        }
    }
    return new int[] {max1, max2};
}

public static void main(String[] args) {
    int[] arr = {84, 31, 3, 1, 567, 4, 2, 93202, 32, 3};
    System.out.println(Arrays.toString(maxMax(arr)));
}
```

↜ **סיבוכיות השוואה:** סה"כ נקבל $3\frac{n-2}{2}+1+2=\frac{3n}{2}-3+3=\frac{3n}{2}$ השוואות.

❖ אפשרות ד' -אלגוריתם אופטימלי:

⇚ מעבר על המערך בזוגות והצמדת מחסנית לכל איבר.

ניצור מחלקה פנימית בשם Node.

```
static class Node {
    int number;
    Stack<Integer> stack;
    public Node(int num) {
        number = num;
        stack = new Stack<>();
    }
}
```

ולמחלקת הפתרון הראשית נממש את הפונקציות הבאות (בשיטה רקורסיבית):

```
public static int[] maxMax(int[] arr) {

   Node[] nodes = new Node[arr.length];

   // init - O(n)
   for(int i = 0 ; i < nodes.length ; i++) {
       nodes[i] = new Node(arr[i]);
   }

   int index = maxMaxRec(nodes, 0 , nodes.length - 1);

   Node biggest = nodes[index];
   int max1 = biggest.number;
   int max2 = biggest.stack.pop();

   while(!biggest.stack.isEmpty()) {
       int temp_max2 = biggest.stack.pop();
       if(temp_max2 > max2)
           max2 = temp_max2;
   }

   return new int[] {max1, max2};
}


private static int maxMaxRec(Node[] nodes, int low, int high) {

   if(low < high) {

       int middle = (high + low)/2;
       int i = maxMaxRec(nodes,low, middle);
       int j = maxMaxRec(nodes, middle + 1 , high);

       int index;
       if(nodes[i].number > nodes[j].number) {
           nodes[i].stack.push(nodes[j].number);
           index = i;
       }
       else{ // nodes[i].number <= nodes[j].number

           nodes[j].stack.push(nodes[i].number);
           index = j;
       }

       return index;
   }
```

```
    else {
        return  low;
    }
}

public static void main(String[] args) {
    int[] arr = {84, 31, 3, 1, 567, 4};
    System.out.println(Arrays.toString(maxMax(arr)));
}
```

↜ **סיבוכיות השוואה:** סה"כ נקבל $n - 1 + log(n) - 1 = n + log(n) - 2$ השוואות.

מצורף טסט השוואות זמנים עבור 4 שיטות הפתרון לבעיה בגיטהאב שלי.