**Sorting:**

import java.util.Arrays;

public class Example {

public static void main(String[] args) {

// TODO Auto-generated method stub

int arr[]= {1,8,7,4,6,3};

int n=arr.length;

Arrays.sort(arr);

System.out.println(arr[n-1]);

}

}

time complexity---o(nlogn)

space complexity-o(1)

**Maximum element:**

import java.util.Arrays;

public class Example {

public static void main(String[] args) {

// TODO Auto-generated method stub

int arr[]= {1,8,7,4,6,3};

int n=arr.length;

int max=Integer.MIN\_VALUE;

for(int i=0;i<n;i++)

{

if(arr[i]>max)

{

max=arr[i];

}

}

System.out.println(max);

}

}

o(n)

------------------------

**Superior Elements or leader Elements in Array**

Problem statement

There is an integer array ‘a’ of size ‘n’.

An element is called a Superior Element if it is greater than all the elements present to its right.

You must return an array all Superior Elements in the array ‘a’.

Note:

The last element of the array is always a Superior Element.

Example

Input: a = [1, 2, 3, 2], n = 4

Output: 2 3

Explanation:

a[ 2 ] = 3 is greater than a[ 3 ]. Hence it is a Superior Element.

a[ 3 ] = 2 is the last element. Hence it is a Superior Element.

The final answer is in sorted order.

Detailed explanation ( Input/output format, Notes, Images )

Sample Input 1:

4

1 2 2 1

Sample Output 1:

1 2

Explanation of Sample Input 1:

Element present at the last index is '1' and is a superior element since there are no integers to the right of it.

Element present at index 2 (0-indexed) is '2' and is greater than all the elements to the right of it.

There are no other superior elements present in the array.

Hence the final answer is [1,2].

Sample Input 2:

3

5 4 3

Sample Output 2:

3 4 5

Expected Time Complexity:

Try to solve this in O(n).

Constraints:

1 <= n <=10^5

1 <= a[i] <= 10^9

Time Limit: 1 sec

import java.util.Arrays;

public class Example {

public static void main(String[] args) {

// TODO Auto-generated method stub

int arr[]= {10,22,12,3,0,6};

int n=arr.length;

boolean leader=true;

for(int i=0;i<n;i++)

{

leader=true;

for(int j=i+1;j<n;j++)

{

if(arr[j]>arr[i])

{

leader=false;

break;

}

}

if(leader)

System.out.println(arr[i]);

}

}

}

o(n2)

------------------------

o(n)

import java.util.Arrays;

public class Example {

public static void main(String[] args) {

// TODO Auto-generated method stub

int arr[]= {10,22,12,3,0,6};

int n=arr.length;

int max=arr[n-1];

System.out.println(max);

for(int i=n-2;i>=0;i--)

{

if(arr[i]>max)

{

max=arr[i];

System.out.println(arr[i]);

}

}

}

}

----------------------

**Reverse an array;**

public class Example {

public static void main(String[] args) {

// TODO Auto-generated method stub

int arr[]= {1,2,3,4,5,6};

int n=arr.length;

int l=0;

int r=n-1;

while(l<=r)

{

int temp=arr[l];

arr[l]=arr[r];

arr[r]=temp;

l++;

r--;

}

for(int i=0;i<n;i++)

{

System.out.print(arRr[i]+" ");

}

}

}

o(n)

sc-o(1)

--------------

**Reverse an array in a sub groups given k(k is multiple of n)**

public class Example {

public static void main(String[] args) {

// TODO Auto-generated method stub

int arr[]= {1,2,3,4,5,6,7,8,9,10,11,12};

int n=arr.length;

int k=3;

int l;

int r;

for(int i=0;i<n;i=i+k)

{

l=i;

r=i+k-1;

while(l<=r)

{

int temp=arr[l];

arr[l]=arr[r];

arr[r]=temp;

l++;

r--;

}

}

for(int i=0;i<n;i++)

{

System.out.print(arr[i]+" ");

}

}

}

----------------------

**which is not a multiple of k**

public class Example {

public static void main(String[] args) {

// TODO Auto-generated method stub

int arr[]= {1,2,3,4,5,6,7,8,9,10,11,12};

int n=arr.length;

int k=10;

int l;

int r;

for(int i=0;i<n;i=i+k)

{

l=i;

if (i+k-1<n-1)

r=i+k-1;

else

r=n-1;

while(l<=r)

{

int temp=arr[l];

arr[l]=arr[r];

arr[r]=temp;

l++;

r--;

}

}

for(int i=0;i<n;i++)

{

System.out.print(arr[i]+" ");

}

}

}

-----------------------

**Problem Description: Sort an Array of 0s and 1s (Binary Sort)**

Given an array containing only 0s and 1s, you are tasked with sorting the array so that all the 0s come before all the 1s with minimal swaps.

Problem Constraints:

• The array contains only two types of elements: 0s and 1s.

• The array length n satisfies:

1 <= n <= 10^5 (The size of the array can be up to 100,000 elements).

• The elements of the array are binary: 0 or 1.

Examples:

Example 1:

Input:

arr = [0, 1, 0, 1, 1, 0]

Output:

0 0 0 1 1 1

Explanation:

The 0s come first followed by all the 1s.

Example 2:

Input:

arr = [1, 0, 1, 0, 1]

Output:

0 0 1 1 1

Explanation:

The 0s come first, followed by the 1s, after sorting the array.

Example 3:

Input:

arr = [0, 0, 0, 0, 0]

Output:

0 0 0 0 0

Explanation:

The array is already sorted with all 0s.

Example 4:

Input:

arr = [1, 1, 1, 1, 0, 0]

Output:

0 0 1 1 1 1

Explanation:

The array contains all the 0s before the 1s after sorting

public class Example {

public static void main(String[] args) {

// TODO Auto-generated method stub

int arr[]= {0,1,1,1,0,1,0};

int n=arr.length;

Arrays.sort(arr);

for(int i=0;i<n;i++)

{

System.out.print(arr[i]+" ");

}

}

}

o(nlogn)

sc-o(1)

---------------------------

**time o(n) with out sorting**

public class Example {

public static void main(String[] args) {

// TODO Auto-generated method stub

int arr[]= {0,1,1,1,0,1,0};

int n=arr.length;

int l=0;

int r=n-1;

while(l<r)

{

while(l<r && arr[l]==0)

l++;

while(l<r && arr[r]==1)

r--;

if(l<r)

{

int temp=arr[l];

arr[l]=arr[r];

arr[r]=temp;

l++;

r--;

}

}

for(int i=0;i<n;i++)

{

System.out.print(arr[i]+" ");

}

}

}-------------------------------------

**Problem Description: Sort Colors (Dutch National Flag Problem)**

The Dutch National Flag Problem involves sorting an array of 0s, 1s, and 2s in a way that all 0s come first, followed by all 1s, and all 2s come last. The problem can be efficiently solved without using extra space, i.e., in-place sorting, by utilizing a three-pointer technique.

Problem Breakdown:

Given an array consisting of only the values 0, 1, and 2, sort the array so that all 0s are placed at the beginning, followed by all 1s, and finally, all 2s are placed at the end.

Time Complexity:

• Time Complexity: O(N) — Each element is processed once, and no nested loops are required.

• Space Complexity: O(1) — The sorting is done in-place.

Constraints:

• The array only contains 0s, 1s, and 2s.

• 1 <= N <= 10^6 — the size of the array.

• The values in the array are limited to 0, 1, and 2.

public class Example {

public static void main(String[] args) {

// TODO Auto-generated method stub

int arr[]= {0,1,1,1,0,1,0,2,1,2};

int n=arr.length;

Arrays.sort(arr);

for(int i=0;i<n;i++)

{

System.out.print(arr[i]+" ");

}

}

}

o(nlogn)

----------------------

public class Example {

public static void main(String[] args) {

// TODO Auto-generated method stub

int arr[]= {0,1,1,1,0,1,0,2,1,2};

int n=arr.length;

System.out.println("Original array"+ Arrays.toString(arr));

sortColors(arr);

System.out.println("After sorted array"+ Arrays.toString(arr));

}

private static void sortColors(int[] arr) {

// TODO Auto-generated method stub

int n=arr.length;

int low=0,mid=0,high=n-1;

while(mid<=high)

{

switch(arr[mid])

{

case 0:swap(arr,mid,low);

low++;

mid++;

break;

case 1:mid++;

break;

case 2:swap(arr,mid,high);

high--;

break;

}

}

}

private static void swap(int arr[],int i, int j) {

// TODO Auto-generated method stub

int temp=arr[j];

arr[j]=arr[i];

arr[i]=temp;

}

}

[**Majority Element**](https://leetcode.com/problems/majority-element/)

Easy

Topics

Companies

Given an array nums of size n, return *the majority element*.

The majority element is the element that appears more than ⌊n / 2⌋ times. You may assume that the majority element always exists in the array.

**Example 1:**

**Input:** nums = [3,2,3]

**Output:** 3

**Example 2:**

**Input:** nums = [2,2,1,1,1,2,2]

**Output:** 2

**Constraints:**

* n == nums.length
* 1 <= n <= 5 \* 104
* -109 <= nums[i] <= 109

**Follow-up:** Could you solve the problem in linear time and in O(1) space?

**Moore voting algorithm.**

|  |  |
| --- | --- |
| **public** **class** Example {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  **int** arr[]= {7,7,5,7,5,1,5,7,5,5,7,7,5,5,5,5,1,1,1,1,1};  **int** n=arr.length;  System.***out***.println(n);    **int** result=*findMajority*(arr);  **if**(result!=-1)  System.***out***.println(result);  **else**  System.***out***.println("No majority element");    }  **private** **static** **int** findMajority(**int**[] arr) {  // **TODO** Auto-generated method stub      **int** candidate=*findCandidate*(arr);  **return** *isMajority*(arr,candidate)?candidate:-1;      }  **private** **static** **boolean** isMajority(**int**[] arr, **int** candidate) {  // **TODO** Auto-generated method stub  **int** count=0;  **for**(**int** i=0;i<arr.length;i++)  {  **if**(candidate==arr[i])  count++;  }    **return** (count>(arr.length/2));  }  **private** **static** **int** findCandidate(**int**[] arr) {  // **TODO** Auto-generated method stub  **int** count=0,candidate=0;  **for**(**int** i=0;i<arr.length;i++)  {  **if**(count==0)  candidate=arr[i];  **if**(arr[i]==candidate)  count++;  **else**  count--;  }      **return** candidate;  }    } |  |

**Balance gates design:**

|  |  |
| --- | --- |
| **public** **class** Example {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  String s=")))(((";  **int** c=0;  **for**(**int** i=0;i<s.length();i++)  {  **if**(s.charAt(i)=='(')  c++;  **else**  c--;  **if**(c==-1)  **break**;  }  **if**(c==0)  System.***out***.println("Balance gates");  **else**  System.***out***.println("Not Balance gates");      }    }  Time o(n)  Space 0(1) | import java.util.Stack;  public class BalancedParentheses {  public static void main(String[] args) {  String s = ")))((("; // Example input  if (isBalanced(s)) {  System.out.println("Balanced gates");  } else {  System.out.println("Not Balanced gates");  }  }  public static boolean isBalanced(String s) {  Stack<Character> stack = new Stack<>();  for (char ch : s.toCharArray()) {  if (ch == '(') {  stack.push(ch); // Push opening bracket  } else if (ch == ')') {  if (stack.isEmpty()) {  return false; // More ')' than '('  }  stack.pop(); // Pop matching '('  }  }  return stack.isEmpty(); // If empty, it is balanced  }  }  Time o(n)  Space o(n) |

[**game-of-thrones**](https://www.hackerrank.com/challenges/game-of-thrones/)

|  |
| --- |
| **import** java.util.Arrays;  **import** java.util.HashMap;  **public** **class** Example {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  String s="aabbccddef";  String res=*canFormPalindrome*(s);  System.***out***.println(res);  }  **private** **static** String canFormPalindrome(String s) {  // **TODO** Auto-generated method stub    HashMap<Character, Integer> charcount= **new** HashMap<Character, Integer>();    **for**(**char** c:s.toCharArray())  {  **int** f=charcount.getOrDefault(c, 0)+1;  charcount.put(c,f);  }    System.***out***.println(charcount);    **int** oddc=0;  **for**(**int** count:charcount.values())  {  **if**(count%2!=0)  oddc++;  **if**(oddc>1)  **return** "No";  }    **return** "Yes";  }    } |

Window sliding technique:

[**Longest Substring Without Repeating Characters**](https://leetcode.com/problems/longest-substring-without-repeating-characters/)

Given a string s, find the length of the **longest** **substring** without duplicate characters.

**Example 1:**

**Input:** s = "abcabcbb"

**Output:** 3

**Explanation:** The answer is "abc", with the length of 3.

**Example 2:**

**Input:** s = "bbbbb"

**Output:** 1

**Explanation:** The answer is "b", with the length of 1.

**Example 3:**

**Input:** s = "pwwkew"

**Output:** 3

**Explanation:** The answer is "wke", with the length of 3.

Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.

**Constraints:**

* 0 <= s.length <= 5 \* 104
* s consists of English letters, digits, symbols and spaces.

|  |
| --- |
| **import** java.util.HashSet;  **public** **class** Example {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  String s="bbbbb";  **int** len=*LongestSub*(s);  System.***out***.println(len);  }  **private** **static** **int** LongestSub(String s) {  // **TODO** Auto-generated method stub  HashSet<Character> hs= **new** HashSet<Character>();  **int** i=0,res=0;  **for**(**int** j=0;j<s.length();j++)  {  **while**(hs.contains(s.charAt(j)))  {  hs.remove(s.charAt((i)));  i++;  }  hs.add(s.charAt(j));  res=Math.*max*(res, j-i+1);  }  **return** res;  }    } |

|  |
| --- |
| import java.util.HashSet;  public class LongestSubstringBruteForce {  public static int lengthOfLongestSubstring(String s) {  int maxLength = 0;    // Iterate over all possible substrings  for (int i = 0; i < s.length(); i++) {  HashSet<Character> seen = new HashSet<>();  for (int j = i; j < s.length(); j++) {  if (seen.contains(s.charAt(j))) {  break; // Stop if duplicate is found  }  seen.add(s.charAt(j));  maxLength = Math.max(maxLength, j - i + 1);  }  }    return maxLength;  }  public static void main(String[] args) {  String s1 = "abcabcbb";  String s2 = "bbbbb";  String s3 = "pwwkew";  System.out.println(lengthOfLongestSubstring(s1)); // Output: 3  System.out.println(lengthOfLongestSubstring(s2)); // Output: 1  System.out.println(lengthOfLongestSubstring(s3)); // Output: 3  }  }--- o(n2) |

**Pair sum of elements or two sum**

|  |  |
| --- | --- |
| **import** java.util.ArrayList;  **import** java.util.HashSet;  **public** **class** Example {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  **int** arr[]= {1,4,9,8,25,5,12};  **int** n=13;  ArrayList<Integer> l=**new** ArrayList<Integer>();  **for**(**int** i=0;i<arr.length;i++)  {  **int** target=n-arr[i];  **if**(l.contains(target))  {  System.***out***.println(arr[i]+ " "+target);  }  l.add(arr[i]);  }  }  } | import java.util.HashSet;  public class Example {  public static void main(String[] args) {  int arr[] = {1, 4, 9, 8, 25, 5, 12};  int n = 13;  HashSet<Integer> set = new HashSet<>();  for (int i = 0; i < arr.length; i++) {  int target = n - arr[i];  if (set.contains(target)) {  System.out.println(arr[i] + " " + target);  }  set.add(arr[i]);  }  }  } |