Problem | Editorial | Submissions | Comments

Median of an Array

Difficulty: **Basic** Accuracy: **44.57%** Submissions: **151K+** Points: **1**

Given an array **arr[]** of integers, calculate the median.

Examples:

Input: arr[] = [90, 100, 78, 89, 67]
Output: 89
Explanation: After sorting the array middle element is the median

Input: arr[] = [56, 67, 30, 79]
Output: 61.5

Output Window

Compilation Results | Custom Input | Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed
1115 / 1115

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Java (21) | Start Timer

```java
class Solution {
    public double findMedian(int[] arr) {
        Arrays.sort(arr);
        int n = arr.length;

        if (n % 2 != 0) {
            return (double) arr[n / 2];
        } else {
            return (double) (arr[(n - 1) / 2] + arr[n / 2]) / 2.0;
        }
    }
}
```

Custom Input | Compile & Run | Submit

https://www.geeksforgeeks.org/problems/chocolate-distribution-problem3825/1

Search...

Get 90% Refund

Courses Tutorials Practice Jobs

Problem Editorial Submissions Comments

Difficulty: Easy Accuracy: 49.91% Submissions: 267K+ Points: 2 Average Time: 15m

Given an array **arr[]** of positive integers, where each value represents the number of chocolates in a packet. Each packet can have a variable number of chocolates. There are **m** students, the task is to distribute chocolate packets among **m** students such that -

i. Each student gets **exactly** one packet.

ii. The difference between maximum number of chocolates given to a student and minimum number of chocolates given to a student is minimum and return that minimum possible difference.

Examples:

Input: arr = [3, 4, 1, 9, 56, 7, 9, 12], m = 5

Output: 6

Explanation: The minimum difference between maximum chocolates and minimum chocolates is 9 − 3 = 6 by choosing following m packets :[3, 4, 9, 7, 9]

Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed

1112 / 1112

Attempts : Correct / Total

1 / 1

Accuracy : 100%

Java (21) Start Timer

```java
// User function Template for Java

class Solution {
    public int findMinDiff(ArrayList<Integer> arr, int m) {
        if (m == 0 || arr.size() == 0) {
            return 0;
        }

        Collections.sort(arr);

        int n = arr.size();
        if (n < m) {
            return -1;
        }

        int minDiff = Integer.MAX_VALUE;

        for (int i = 0; i + m - 1 < n; i++) {
            int currentDiff = arr.get(i + m - 1) - arr.get(i);
            if (currentDiff < minDiff) {
                minDiff = currentDiff;
            }
        }

        return minDiff;
    }
}
```

Custom Input Compile & Run Submit

https://www.geeksforgeeks.org/problems/smallest-subarray-with-sum-greater-than-x5651/1

Search...

Get 90% Refund

Courses Tutorials Practice Jobs

Problem Editorial Submissions Comments

Smallest subarray with sum greater than x

Difficulty: **Easy** Accuracy: **37.07%** Submissions: **154K+** Points: **2** Average Time: **20m**

Given a number **x** and an array of integers **arr**, find the smallest subarray with sum greater than the given value. If such a subarray do not exist return 0 in that case.

Examples:

Input: x = 51, arr[] = [1, 4, 45, 6, 0, 19]
Output: 3
Explanation: Minimum length subarray is [4, 45, 6]

Input: x = 100, arr[] = [1, 10, 5, 2, 7]

Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed
1112 / 1112

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Java (21) Start Timer

```java
class Solution {
    public static int smallestSubWithSum(int x, int[] arr) {
        int n = arr.length;
        int currentSum = 0;
        int minLen = n + 1;
        int start = 0;
        int end = 0;

        while (end < n) {
            while (currentSum <= x && end < n) {
                currentSum += arr[end++];
            }

            while (currentSum > x && start < n) {
                if (end - start < minLen) {
                    minLen = end - start;
                }
                currentSum -= arr[start++];
            }
        }

        return (minLen == n + 1) ? 0 : minLen;

    }
}
```

Custom Input Compile & Run Submit

https://www.geeksforgeeks.org/problems/three-way-partitioning/1

Search...

Get 90% Refund
Courses | Tutorials | Practice | Jobs

Problem | Editorial | Submissions | Comments

Three way partitioning

Difficulty: **Easy** Accuracy: **41.58%** Submissions: **187K+** Points: **2** Average Time: **20m**

Given an **array** and a range **a**, **b**. The task is to partition the array around the range such that the array is divided into three parts.
1) All elements smaller than **a** come first.
2) All elements in range **a** to **b** come next.
3) All elements greater than **b** appear in the end.
The individual elements of three sets can appear in any order. You are required to return the modified array.

Note: The generated output is true if you modify the given array successfully. Otherwise false.

Geeky Challenge: Solve this problem in O(n) time complexity.

Output Window

Compilation Results | Custom Input | Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed
1111 / 1111

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Java (21) Start Timer

```java
class Solution {
    public void threeWayPartition(int arr[], int a, int b) {
        // code here
        int n = arr.length;
        int low = 0;
        int high = n - 1;
        int i = 0;

        while (i <= high) {
            if (arr[i] < a) {
                int temp = arr[i];
                arr[i] = arr[low];
                arr[low] = temp;
                low++;
                i++;
            } else if (arr[i] > b) {
                int temp = arr[i];
                arr[i] = arr[high];
                arr[high] = temp;
                high--;
            } else {
                i++;
            }
        }
    }
}
```

Custom Input | Compile & Run | Submit

https://www.geeksforgeeks.org/problems/minimum-swaps-required-to-bring-all-elements-less-than-or-equal-to-k-together4847/1

Search...

Get 90% Refund

Courses | Tutorials | Practice | Jobs

Problem | Editorial | Submissions | Comments

Given an array **arr** and a number **k**. One can apply a swap operation on the array any number of times, i.e choose any two index i and j (i < j) and swap arr[i] , arr[j] . Find the **minimum** number of swaps required to bring all the numbers less than or equal to **k** together, i.e. make them a contiguous subarray.

Examples :

Input: arr[] = [2, 1, 5, 6, 3], k = 3
Output: 1
Explanation: To bring elements 2, 1, 3 together, swap index 2 with 4 (0-based indexing), i.e. element arr[2] = 5 with arr[4] = 3 such that final array will be- arr[] = [2, 1, 3, 6, 5]

Input: arr[] = [2, 7, 9, 5, 8, 7, 4], k = 6

Output Window

Compilation Results | Custom Input | Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed
1112 / 1112

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Java (21)

Start Timer

```java
class Solution {
    // Function for finding minimum swaps required
    int minSwap(int arr[], int k) {
        int n = arr.length;

        // 1. Count how many elements are less than or equal to k
        int count = 0;
        for (int i = 0; i < n; i++) {
            if (arr[i] <= k) {
                count++;
            }
        }

        // 2. Count how many "bad" elements (greater than k)
        // are in the first window of size 'count'
        int bad = 0;
        for (int i = 0; i < count; i++) {
            if (arr[i] > k) {
                bad++;
            }
        }

        // Initialize answer with 'bad' elements in the first window
        int ans = bad;

        // 3. Use a sliding window to check all other windows of size 'count'
        for (int i = 0, j = count; j < n; i++, j++) {

            // If the element leaving the window was bad, decrement bad count
            if (arr[i] > k) {
                bad--;
            }

            // If the element entering the window is bad, increment bad count
            if (arr[j] > k) {
                bad++;
            }
```

Custom Input | Compile & Run | Submit