1] Maximum Index

Given an array arr[] of n non-negative integers. The task is to find the maximum of j - i (i<=j) subjected to the constraint of arr[i] <= arr[j].

Example 1:

Input:
n = 9
arr[] = {34, 8, 10, 3, 2, 80, 30, 33, 1}
Output:
6
Explanation:
In the given array arr[1] < arr[7] satisfying the required condition (arr[i] <= arr[j]) thus giving the maximum difference of j - i which is 6(7-1).

Example 2:

Input:
N = 2
arr[] = {18, 17}
Output:
0
Explanation:
We can either take i and j as 0 and 0
or we can take 1 and 1 both give the same result 0.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(N)

Constraints:
$1 \leq N \leq 10^6$
$0 \leq Arr[i] \leq 10^9$

2] Pythagorean Triplet

Given an array arr of N integers, write a function that returns true if there is a triplet (a, b, c) that satisfies a2 + b2 = c2, otherwise false.

Example 1:

Input:
N = 5
Arr[] = {3, 2, 4, 6, 5}
Output: Yes
Explanation: a=3, b=4, and c=5 forms a pythagorean triplet.

Example 2:

Input:
N = 3
Arr[] = {3, 8, 5}
Output: No
Explanation: No such triplet possible.

Expected Time Complexity: O(max(Arr[i])2)
Expected Auxiliary Space: O(max(Arr[i]))

Constraints:
1 <= N <= 10^7
1 <= Arr[i] <= 1000

3] Zero Sum Subarrays

You are given an array arr[] of size n. Find the total count of sub-arrays having their sum equal to 0.

Example 1:

Input:
n = 6
arr[] = {0,0,5,5,0,0}
Output: 6
Explanation: The 6 subarrays are
[0], [0], [0], [0], [0,0], and [0,0].

Example 2:

Input:
n = 10
arr[] = {6,-1,-3,4,-2,2,4,6,-12,-7}
Output: 4
Explanation: The 4 subarrays are [-1 -3 4]
[-2 2], [2 4 6 -12], and [-1 -3 4 -2 2]

Expected Time Complexity: O(n*log(n))
Expected Auxiliary Space: O(n)

Constraints:
1 <= n <= 10^6
-10^9 <= arr[i] <= 10^9

4] Kth smallest element

Given an array arr[] and an integer K where K is smaller than the size of the array, the task is to find the Kth smallest element in the given array. It is given that all array elements are distinct.

Note :- l and r denotes the starting and ending index of the array.

Example 1:

Input:
N = 6
arr[] = 7 10 4 3 20 15
K = 3
Output : 7
Explanation : 3rd smallest element in the given array is 7.

Example 2:

Input:
N = 5
arr[] = 7 10 4 20 15
K = 4
Output : 15
Explanation : 4th smallest element in the given array is 15.

Expected Time Complexity: O(n)
Expected Auxiliary Space: O(log(n))

Constraints:
1 <= N <= 105
1 <= arr[i] <= 105
1 <= K <= N

5] k largest elements

Given an array Arr of N positive integers and an integer K, find K largest elements from the array. The output elements should be printed in decreasing order.

Example 1:

> **Input:**
> **N** = 5, **K** = 2
> **Arr**[] = {12, 5, 787, 1, 23}
> **Output**: 787 23
> **Explanation**: 1st largest element in the array is 787 and second largest is 23.

Example 2:

> **Input**:
> **N** = 7, **K** = 3
> **Arr**[] = {1, 23, 12, 9, 30, 2, 50}
> **Output**: 50 30 23
> **Explanation**: 3 Largest element in the array are 50, 30 and 23.

Expected Time Complexity: O(K+(N-K)*logK)
Expected Auxiliary Space: O(K+(N-K)*logK)

Constraints:
$1 \leq K \leq N \leq 10^5$
$1 \leq Arr[i] \leq 10^6$

6] Largest Number formed from an Array

Given a list of non negative integers, arrange them in such a manner that they form the largest number possible.The result is going to be very large, hence return the result in the form of a string.

Example 1:

> **Input**:
> **N** = 5
> **Arr**[] = {3, 30, 34, 5, 9}
> **Output**: 9534330
> **Explanation**: Given numbers are {3, 30, 34, 5, 9}, the arrangement 9534330 gives the largest value.

Example 2:

> **Input:**
> **N** = 4
> **Arr**[] = {54, 546, 548, 60}
> **Output**: 6054854654
> **Explanation**: Given numbers are {54, 546, 548, 60}, the arrangement 6054854654 gives the largest value.

Expected Time Complexity: O(NlogN)
Expected Auxiliary Space: O(N)

Constraints:
$1 \leq N \leq 105$
$0 \leq Arr[i] \leq 1018$

Sum of all the elements of the array is greater than 0.

7] Maximum Subarray

Find out the maximum sub-array of non negative numbers from an array.
The sub-array should be contiguous i.e., a sub-array created by choosing the second and fourth element and skipping the third element is invalid.
Maximum sub-array is defined in terms of the sum of the elements in the sub-array.
Sub-array A is greater than sub-array B if sum(A) > sum(B).

Example:
a = [1, 2, 5, -7, 2, 3]
The two sub-arrays are [1, 2, 5] [2, 3].
The answer is [1, 2, 5] as its sum is larger than [2, 3]

NOTE: If there is a tie, then compare the segment's length and the return segment which has the maximum length.
If there is still a tie, then return the segment with the minimum starting index.
If no such subarray is present return "-1"

Example 1:

Input:
n = 3
a[] = {1, 2, 3}
Output: 1 2 3
Explanation: In the given array every element is non-negative.

Example 2:

Input:
n = 2
a[] = {-1, 2}
Output: 2
Explanation: The only subarray [2] is the answer.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

Constraints:
$1 \leq N \leq 10^5$
$-105 \leq A[i] \leq 10^5$

8] Find missing in second array

Given two arrays A and B containing integers of size N and M, the task is to find numbers which are present in the first array, but not present in the second array.

Example 1:

> **Input**: N = 6, M = 5
> **A**[] = {1, 2, 3, 4, 5, 10}
> **B**[] = {2, 3, 1, 0, 5}
> **Output**: 4 10
> **Explanation**: 4 and 10 are present in first array, but not in second array.

Example 2:

> **Input**: N = 5, M = 5
> **A**[] = {4, 3, 5, 9, 11}
> **B**[] = {4, 9, 3, 11, 10}
> **Output**: 5
> **Explanation**: Second array does not contain element 5.

Expected Time Complexity: O(N+M).
Expected Auxiliary Space: O(M).

Constraints:
$1 \leq N, M \leq 10^6$
$-10^6 \leq A[i], B[i] \leq 10^6$

9] K-th element of two Arrays

Given two arrays arr1 and arr2 of size N and M respectively and an element K. The task is to find the element that would be at the kth position of the final sorted array.

Example 1:

Input:
arr1[] = {2, 3, 6, 7, 9}
arr2[] = {1, 4, 8, 10}
k = 5
Output: 6
Explanation:
The final sorted array would be - 1, 2, 3, 4, 6, 7, 8, 9, 10
The 5th element of this array is 6.

Example 2:

Input:
arr1[] = {100, 112, 256, 349, 770}
arr2[] = {72, 86, 113, 119, 265, 445}
k = 7
Output: 256
Explanation:
Final sorted array is - 72, 86, 100, 112, 113, 119, 256, 265, 349, 445, 770
7th element of this array is 256.

Expected Time Complexity: O(Log(N) + Log(M))
Expected Auxiliary Space: O(Log (N))

Constraints:
1 <= N, M <= 106
0 <= arr1i, arr2i < INT_MAX
1 <= K <= N+M

10] Count the number of possible triangles

Given an unsorted array arr[] of n positive integers. Find the number of triangles that can be formed with three different array elements as lengths of three sides of triangles.

Example 1:

Input:
n = 3
arr[] = {3, 5, 4}
Output:
1
Explanation:
A triangle is possible with all the elements 5, 3 and 4.

Example 2:

Input:
n = 5
arr[] = {6, 4, 9, 7, 8}
Output: 10
Explanation:
There are 10 triangles possible with the given elements like (6,4,9), (6,7,8),...

Expected Time Complexity: O(n2).
Expected Space Complexity: O(1).

Constraints:
3 <= n <= 10^3
1 <= arr[i] <= 10^3

11] Maximum of all subarrays of size k

Given an array arr[] of size N and an integer K. Find the maximum for each and every contiguous subarray of size K.

Example 1:

Input:
N = 9, **K** = 3
arr[] = 1 2 3 1 4 5 2 3 6
Output: 3 3 4 5 5 5 6
Explanation:
1st contiguous subarray = {1 2 3} **Max = 3**
2nd contiguous subarray = {2 3 1} **Max = 3**
3rd contiguous subarray = {3 1 4} **Max = 4**
4th contiguous subarray = {1 4 5} **Max = 5**
5th contiguous subarray = {4 5 2} **Max = 5**
6th contiguous subarray = {5 2 3} **Max = 5**
7th contiguous subarray = {2 3 6} **Max = 6**

Example 2:

Input:
N = 10, **K** = 4
arr[] = 8 5 10 7 9 4 15 12 90 13
Output: 10 10 10 15 15 90 90
Explanation:
1st contiguous subarray = {8 5 10 7}, **Max = 10**
2nd contiguous subarray = {5 10 7 9}, **Max = 10**
3rd contiguous subarray = {10 7 9 4}, **Max = 10**
4th contiguous subarray = {7 9 4 15}, **Max = 15**
5th contiguous subarray = {9 4 15 12}, **Max = 15**
6th contiguous subarray = {4 15 12 90}, **Max = 90**
7th contiguous subarray = {15 12 90 13}, **Max = 90**

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(k)

Constraints:
$1 \leq N \leq 105$
$1 \leq K \leq N$
$0 \leq arr[i] \leq 107$

12] Row with max 1s

Given a boolean 2D array of n x m dimensions where each row is sorted. Find the 0-based index of the first row that has the maximum number of 1's.

Example 1:

Input:
N = 4 , **M** = 4
Arr[][] = {{0, 1, 1, 1},
{0, 0, 1, 1},
{1, 1, 1, 1},
{0, 0, 0, 0}}
Output: 2
Explanation: Row 2 contains 4 1's (0-based indexing).

Example 2:

Input:
N = 2, **M** = 2
Arr[][] = {{0, 0}, {1, 1}}
Output: 1
Explanation: Row 1 contains 2 1's (0-based indexing).

Expected Time Complexity: O(N+M)
Expected Auxiliary Space: O(1)

Constraints:
$1 \leq N, M \leq 103$
$0 \leq Arr[i][j] \leq 1$

13] Find triplets with zero sum

Given an array arr[] of n integers. Check whether it contains a triplet that sums up to zero.

Note: Return 1, if there is at least one triplet following the condition else return 0.

Example 1:

Input: n = 5, arr[] = {0, -1, 2, -3, 1}
Output: 1
Explanation: 0, -1 and 1 forms a triplet with sum equal to 0.

Example 2:

Input: n = 3, arr[] = {1, 2, 3}
Output: 0
Explanation: No triplet with zero sum exists.

Expected Time Complexity: O(n2)
Expected Auxiliary Space: O(1)

Constraints:
1 <= n <= 10^4
-10^6 <= Ai <= 10^6

14] Sum of two elements with sum nearest to zero

Given an integer array of N elements. You need to find the maximum sum of two elements such that the sum is closest to zero.

Example 1:

> **Input**:
> **N** = 3
> **arr**[] = {-8 -66 -60}
> **Output**: -68
> **Explanation:** Sum of two elements closest to zero is -68 using numbers -60 and -8.

Example 2:

> **Input:**
> **N** = 6
> **arr**[] = {-21 -67 -37 -18 4 -65}
> **Output:** -14
> **Explanation**: Sum of two elements closest to zero is -14 using numbers -18 and 4.

Note : In Case if we have two or more ways to form the sum of two elements closest to zero return the maximum sum.

Expected Time Complexity: O(N*logN).
Expected Auxiliary Space: O(1).

Constraints:
$2 \leq N \leq 5 * 10^5$
$-10^6 \leq arr[i] \leq 10^6$

15] IPL 2021 - Match Day 2

Due to the rise of covid-19 cases in India, this year BCCI decided to organize knock-out matches in IPL rather than a league.

Today is matchday 2 and it is between the most loved team Chennai Super Kings and the most underrated team - Punjab Kings. Stephen Fleming, the head coach of CSK, analyzing the batting stats of Punjab. He has stats of runs scored by all N players in the previous season and he wants to find the maximum score for each and every contiguous sub-list of size K to strategize for the game.

Example 1:

Input:
N = 9, **K** = 3
arr[] = 1 2 3 1 4 5 2 3 6
Output:
3 3 4 5 5 5 6
Explanation:
1st contiguous subarray = {1 2 3} Max = 3
2nd contiguous subarray = {2 3 1} Max = 3
3rd contiguous subarray = {3 1 4} Max = 4
4th contiguous subarray = {1 4 5} Max = 5
5th contiguous subarray = {4 5 2} Max = 5
6th contiguous subarray = {5 2 3} Max = 5
7th contiguous subarray = {2 3 6} Max = 6

Example 2:

Input:
N = 10, **K** = 4
arr[] = 8 5 10 7 9 4 15 12 90 13
Output:
10 10 10 15 15 90 90

Explanation:

1st contiguous subarray = {8 5 10 7}, Max = 10
2nd contiguous subarray = {5 10 7 9}, Max = 10
3rd contiguous subarray = {10 7 9 4}, Max = 10
4th contiguous subarray = {7 9 4 15}, Max = 15
5th contiguous subarray = {9 4 15 12}, Max = 15
6th contiguous subarray = {4 15 12 90}, Max = 90
7th contiguous subarray = {15 12 90 13}, Max = 90

Constraints:

$1 \leq N \leq 106$
$1 \leq K \leq N$
$0 \leq arr[i] \leq 106$

16] Nuts and Bolts Problem

Given a set of N nuts of different sizes and N bolts of different sizes. There is a one-one mapping between nuts and bolts. Match nuts and bolts efficiently. Comparison of a nut to another nut or a bolt to another bolt is not allowed. It means nut can only be compared with bolt and bolt can only be compared with nut to see which one is bigger/smaller.
The elements should follow the following order ! # $ % & * @ ^ ~ .

Example 1:

Input:
N = 5
nuts[] = {@, %, $, #, ^}
bolts[] = {%, @, #, $ ^}
Output:
$ % @ ^
$ % @ ^

Example 2:

Input:
N = 9
nuts[] = {^, &, %, @, #, *, $, ~, !}
bolts[] = {~, #, @, %, &, *, $,^, !}
Output:
! # $ % & * @ ^ ~
! # $ % & * @ ^ ~

Expected Time Complexity: O(NlogN)
Expected Auxiliary Space: O(1)
Constraints:
1 <= N <= 9
Array of Nuts/Bolts can only consist of the following elements: {'@', '#', '$', '%', '^', '&', '~', '*', '!'}.

17] Maximize Number of 1's

Given a binary array arr of size N and an integer M. Find the maximum number of consecutive 1's produced by flipping at most M 0's.

Example 1:

Input:
N = 3
arr[] = {1, 0, 1}
M = 1
Output: 3
Explanation:
Maximum subarray is of size 3 which can be made subarray of all 1 after flipping one zero to 1.

Example 2:

Input:
N = 11
arr[] = {1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1}
M = 2
Output: 8
Explanation:
Maximum subarray is of size 8 which can be made subarray of all 1 after flipping two zeros to 1.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

Constraints:
1 <= N <= 107
0 <= M <= N
0 <= arri <= 1

18] Container With Most Water

Given N non-negative integers a1,a2,....and where each represents a point at coordinate (i, ai). N vertical lines are drawn such that the two endpoints of line i are at (i, ai) and (i,0). Find two lines, which together with the x-axis forms a container, such that it contains the most water.
Note : In Case of a single vertical line it will not be able to hold water.

Example 1:

Input:
N = 4
a[] = {1,5,4,3}
Output: 6
Explanation: 5 and 3 are distance 2 apart.
So the size of the base = 2. Height of
container = min(5, 3) = 3. So total area= 3 * 2 = 6.

Example 2:

Input:
N = 5
a[] = {3,1,2,4,5}
Output: 12

Explanation: 5 and 3 are distance 4 apart.
So the size of the base = 4. Height of
container = min(5, 3) = 3. So total area = 4 * 3 = 12.

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(1).
Constraints:
1<=N<=105
1<=A[i]<=105

19] Rotate a 2D array without using extra space

Given a N x N 2D matrix Arr representing an image. Rotate the image by 90 degrees (anti-clockwise). You need to do this in place. Note that if you end up using an additional array, you will only receive a partial score.

Example 1:

Input:
N = 3
Arr[][] = {{1, 2, 3}
{4, 5, 6}
{7, 8, 9}}
Output:
3 6 9
2 5 8
1 4 7
Explanation: The given matrix is rotated
by 90 degrees in an anti-clockwise direction.

Example 2:

Input:
N = 4
Arr[][] = {{1, 2, 3, 4}
{5, 6, 7, 8}
{9, 10, 11, 12}
{13, 14, 15, 16}}
Output:
4 8 12 16
3 7 11 15
2 6 10 14
1 5 9 13
Explanation: The given matrix is rotated
by 90 degrees in an anti-clockwise direction.

Expected Time Complexity: O(N*N)
Expected Auxiliary Space: O(1)

Constraints:
$1 \leq N \leq 1000$
$1 \leq Arr[i][j] \leq 1000$

20] Majority Element

Given an array A of N elements. Find the majority element in the array. A majority element in an array A of size N is an element that appears more than or equal to N/2 times in the array.

Example 1:

Input:
N = 3
A[] = {1,2,3}
Output: -1
Explanation:
Since, each element in
{1,2,3} appears only once so there
is no majority element.

Example 2:

Input:
N = 5
A[] = {3,1,3,3,2}
Output: 3
Explanation:
Since, 3 is present more
than N/2 times, so it is
the majority element.

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(1).

Constraints:
$1 \leq N \leq 10^7$
$0 \leq Ai \leq 10^6$

21] Kadane's Algorithm

Given an array Arr[] of N integers. Find the contiguous sub-array(containing at least one number) which has the maximum sum and return its sum.

Example 1:

Input:
N = 5
Arr[] = {1,2,3,-2,5}
Output:
9
Explanation:
Max subarray sum is 9 of elements (1, 2, 3, -2, 5) which is a contiguous subarray.

Example 2:

Input:
N = 4
Arr[] = {-1,-2,-3,-4}
Output:
-1
Explanation:
Max subarray sum is -1 of element (-1)

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

Constraints:
$1 \leq N \leq 10^6$
$-10^7 \leq A[i] \leq 10^7$

22] Jump Game

Given a positive integer N and a list of N integers A[]. Each element in the array denotes the maximum length of jump you can cover. Find out if you can make it to the last index if you start at the first index of the list.

Example 1:

Input:
N = 6
A[] = {1, 2, 0, 3, 0, 0}
Output:
1
Explanation:
Jump 1 step from first index to
second index. Then jump 2 steps to reach
4th index, and now jump 2 steps to reach
the end.

Example 2:

Input:
N = 3
A[] = {1, 0, 2}
Output:
0
Explanation:
You can't reach the end of the array.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

Constraints:
1 <= N <= 10^5
0 <= A[i] <= 10^5

23] Top k numbers in a stream

Given N numbers in an array. Your task is to keep at-most K numbers at the top (According to their frequency). We basically need to print top k numbers when the input stream has k distinct elements, else we need to print all distinct elements sorted by frequency.

Example 1:

Input:
N=5, **K**=4
arr[] = {5, 2, 1, 3, 2}
Output: 5 2 5 1 2 5 1 2 3 5 2 1 3 5
Explanation:
Firstly there were 5 whose frequency is max till now. so print 5.
Then 2 , which is smaller than 5 but their frequency is the same so print 2 5.
Then 1, which is the smallest among all the numbers, arrived, so print 1 2 5.
Then 3 , so print 1 2 3 5
Then again 2, which has the highest frequency among all numbers so 2 1 3 5.

Example 2:

Input:
N=5, **K**=4
arr[] = {5, 2, 1, 3, 4}
Output: 5 2 5 1 2 5 1 2 3 5 1 2 3 4
Explanation:
Firstly there were 5 whose frequency is max till now. so print 5.
Then 2 , which is smaller than 5 but their frequency is the same so print 2 5.
Then 1, Which is the smallest among all the numbers arrived, so print 1 2 5.
Then 3 , so print 1 2 3 5.
Then 4, so 1 2 3 4 as K is 4 so print at-most k elements.

Expected Time Complexity: O(N*K).
Expected Auxiliary Space: O(N).

Constraints:
$1 \leq N,K \leq 10^3$

24] Tic Tac Toe

A Tic-Tac-Toe board is given after some moves are played. Find out if the given board is valid, i.e., is it possible to reach this board position after some moves or not.
Note that every arbitrary filled grid of 9 spaces isn't valid e.g. a grid filled with 3 X and 6 O isn't a valid situation because each player needs to take alternate turns.
Note : The game starts with X

Example 1:

Input:
board[] = {'X', 'X', 'O',
'O', 'O', 'X',
'X', 'O', 'X'};
Output: Valid
Explanation: This is a valid board.

Example 2:

Input:
board[] = {'O', 'X', 'X',
'O', 'X', 'X',
'O', 'O', 'X'};
Output: Invalid
Explanation: Both X and O cannot win.

Expected Time Complexity: O(1)
Expected Auxiliary Space: O(1)

Constraints:
Every character on the board can either be 'X' or 'O'.

25] Longest Arithmetic Progression

Given an array called A[] of sorted integers having no duplicates, find the length of the Longest Arithmetic Progression (LLAP) in it.

Example 1:

Input:
N = 6
set[] = {1, 7, 10, 13, 14, 19}
Output: 4
Explanation: The longest arithmetic progression is {1, 7, 13, 19}.

Example 2:

Input:
N = 5
A[] = {2, 4, 6, 8, 10}
Output: 5
Explanation: The whole set is in AP.

Expected Time Complexity: O(N2)
Expected Auxiliary Space: O(N2)

Constraints:
$1 \leq N \leq 1000$
$1 \leq set[i] \leq 104$

26] Spirally traversing a matrix

Given a matrix of size r*c. Traverse the matrix in spiral form.

Example 1:

Input:
r = 4, c = 4
matrix[][] = {{1, 2, 3, 4},
{5, 6, 7, 8},
{9, 10, 11, 12},
{13, 14, 15,16}}
Output:
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
Explanation:

Example 2:

Input:
r = 3, c = 4
matrix[][] = {{1, 2, 3, 4},
{5, 6, 7, 8},
{9, 10, 11, 12}}
Output:
1 2 3 4 8 12 11 10 9 5 6 7
Explanation:
Applying the same technique as shown above, output for the 2nd test case will be 1 2 3 4 8 12 11 10 9 5 6 7.

Expected Time Complexity: O(r*c)
Expected Auxiliary Space: O(r*c), for returning the answer only.

Constraints:
1 <= r, c <= 100
0 <= matrixi <= 100

27] Smallest Positive missing number

You are given an array arr[] of N integers. The task is to find the smallest positive number missing from the array.

Note: Positive number starts from 1.

Example 1:

Input:
N = 5
arr[] = {1,2,3,4,5}
Output: 6
Explanation: Smallest positive missing number is 6.

Example 2:

Input:
N = 5
arr[] = {0,-10,1,3,-20}
Output: 2
Explanation: Smallest positive missing number is 2.

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(1).

Constraints:
1 <= N <= 10^6
-10^6 <= arr[i] <= 10^6

28] Find Missing And Repeating

Given an unsorted array Arr of size N of positive integers. One number 'A' from set {1, 2,....,N} is missing and one number 'B' occurs twice in array. Find these two numbers.

Example 1:

Input:
N = 2
Arr[] = {2, 2}
Output: 2 1
Explanation: Repeating number is 2 and the smallest positive missing number is 1.

Example 2:

Input:
N = 3
Arr[] = {1, 3, 3}
Output: 3 2
Explanation: Repeating number is 3 and smallest positive missing number is 2.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

Constraints:
$2 \leq N \leq 105$
$1 \leq Arr[i] \leq N$

29] Minimum Platforms

Given arrival and departure times of all trains that reach a railway station. Find the minimum number of platforms required for the railway station so that no train is kept waiting.
Consider that all the trains arrive on the same day and leave on the same day. Arrival and departure time can never be the same for a train but we can have the arrival time of one train equal to departure time of the other. At any given instance of time, the same platform can not be used for both departure of a train and arrival of another train. In such cases, we need different platforms.

Example 1:

Input: n = 6
arr[] = {0900, 0940, 0950, 1100, 1500, 1800}
dep[] = {0910, 1200, 1120, 1130, 1900, 2000}
Output: 3
Explanation: Minimum 3 platforms are required to safely arrive and depart all trains.

Example 2:

Input: n = 3
arr[] = {0900, 1100, 1235}
dep[] = {1000, 1200, 1240}
Output: 1
Explanation: Only 1 platform is required to safely manage the arrival and departure of all trains.

Note: Time intervals are in the 24-hour format(HHMM) , where the first two characters represent hours (between 00 to 23) and the last two characters represent minutes (this may be > 59).

Expected Time Complexity: O(nLogn)
Expected Auxiliary Space: O(n)

Constraints:
$1 \leq n \leq 50000$
$0000 \leq A[i] \leq D[i] \leq 2359$

30] Count the subarrays having product less than k

Given an array of positive numbers, the task is to find the number of possible contiguous subarrays having product less than a given number k.

Example 1:

Input :
n = 4, k = 10
a[] = {1, 2, 3, 4}
Output : 7
Explanation:
The contiguous subarrays are {1}, {2}, {3}, {4}
{1, 2}, {1, 2, 3} and {2, 3}, in all these subarrays
product of elements is less than 10, count of
such a subarray is 7.
{2,3,4} will not be a valid subarray, because
2*3*4=24 which is greater than 10.

Example 2:

Input:
n = 7 , k = 100
a[] = {1, 9, 2, 8, 6, 4, 3}
Output:16

Expected Time Complexity: O(n)
Expected Auxiliary Space: O(1)

Constraints:
1<=n<=10^6
1<=k<=10^15
1<=a[i]<=10^5