209. Minimum Size Subarray Sum

Medium 8.8K 245

Companies

Given an array of positive integers `nums` and a positive integer `target`, return *the* ***minimal length*** *of a subarray whose sum is greater than or equal to* `target`. If there is no such subarray, return `0` instead.

Example 1:

```
Input: target = 7, nums = [2,3,1,2,4,3]
Output: 2
Explanation: The subarray [4,3] has the minimal length under the problem constraint.
```

Example 2:

```cpp
class Solution {
public:
    int minSubArrayLen(int target, vector<int>& a) {
        int ans = INT_MAX;
        int sum = 0;
        int start = 0, end = 0;
        int n = a.size();
        while(end < n){
            sum += a[end];
            while(start < a.size() && sum >= target){
                int temp = end - start + 1;
                ans = min(ans,temp);
                sum -= a[start++];
            }
            end++;
        }
        if(ans == INT_MAX)
            return 0;
        return ans;
    }
};
```

Description | Discussion (36) | Solutions (3.5K) | Submissions

C++ | Auto

152. Maximum Product Subarray

Medium | 15.2K | 458

Companies

Given an integer array `nums`, find a subarray that has the largest product, and return *the product*.

The test cases are generated so that the answer will fit in a **32-bit** integer.

Example 1:

```
Input: nums = [2,3,-2,4]
Output: 6
Explanation: [2,3] has the largest product 6.
```

Example 2:

```
Input: nums = [-2,0,-1]
Output: 0
Explanation: The result cannot be 2, because [-2,-1] is not a subarray.
```

```cpp
class Solution {
public:
    int maxProduct(vector<int>& a) {
        int res = a[0];
        int maxi = res;
        int mini = res;

        int n = a.size();
        for(int i = 1 ; i<n ; i++){
            if(a[i] < 0){
                swap(maxi,mini);
            }
            maxi = max(a[i] , maxi*a[i]);
            mini = min(a[i] , mini*a[i]);
            res = max(res , maxi);
        }
        return res;
    }
};
```

Console | Run | Submit

Triplet Sum in Array

Medium Accuracy: 35.0% Submissions: 205K+ Points: 4

Don't Miss Out on the Exciting World of Data Science and Unlock Your Full Potential Today. Try this course!

Given an array arr of size n and an integer X. Find if there's a triplet in the array which sums up to the given integer X.

Example 1:

Input:
n = 6, X = 13
arr[] = [1 4 45 6 10 8]
Output:
1
Explanation:
The triplet {1, 4, 8} in
the array sums up to 13.

Example 2:

C++ (g++ 5.4) Average Time: 15m Start Timer

```
// } Driver Code Ends
class Solution{
    public:
    //Function to find if there exists a triplet in the
    //array A[] which sums up to X.
    bool find3Numbers(int A[], int n, int X)
    {
        //Your Code Here
        sort(A,A+n);
        for(int i = 0 ; i<n ; i++){
            int y = X- A[i];
            int low = i+1 , high = n-1;

            while(low < high){
                if(A[low] + A[high] == y){
                    return 1;
                }else if (A[low ] + A[high] > y){
                    high--;
                }else if (A[low] + A[high] < y){
                    low++;
                }
            }
        }
        return 0;
    }

};
// } Driver Code Ends
```

Custom Input Compile & Run Submit