Arrays Problems:

Question1: Longest Subarray with give sum k(positives)

Solutions:

Question2: Majority Of element n/2 **Moore’s voting**

**Solutions:**

We need to find the element which appears more than n/2

Arr[] = {2,2,3,4,2,2,1,1,2};

Ans = 2;

Brute force :  
  
Search the entire with 2 loops and scan the entire array and chech

If count > n/2 then break

And return the value!!

TC = O(n^2)

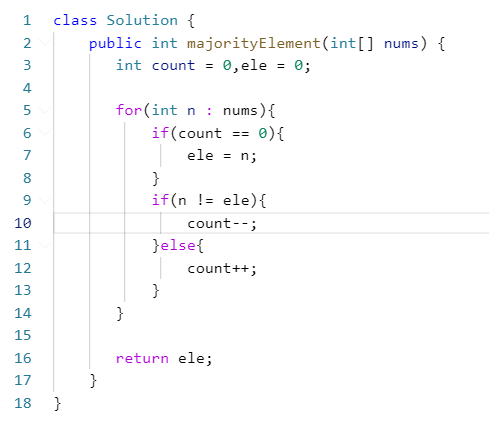
Optimal Solutions2

Sort the array and return the value of nums[length/2]

It will take O(nlogn);

Optimal Solution 2:  
  
Take one hashtable

**Moore’s Algorithm:**

****

**Kaden’s Algorithm:**

**Maximum Subarray sum in an array**

**Input:** arr = [-2,1,-3,4,-1,2,1,-5,4]

**Output:** 6

**BruteForce:**

**Public static int maxSubarraySum(int arr[] , int n){**

**Int maxI = Integer.MIN\_VALUE;**

**For(int I = 0; I < n.length; i++){**

**Int sum = 0;**

**For(int j = i+1; I < n; i++){**

Sum = sum + arr[j];

maxI = Math.max(sum, maxI);

}

# Optimal Approach :

Int maxI = Integer.MIN\_VALUE;

For(int I = 0; I < n; i++)

{

Sum = sum + arr[i];

If(sum > MaxI)

{  
maxI = sum;

}

If(sum <0)

Sum = 0;

Return maxI;

# Second Largest Element In An Array:

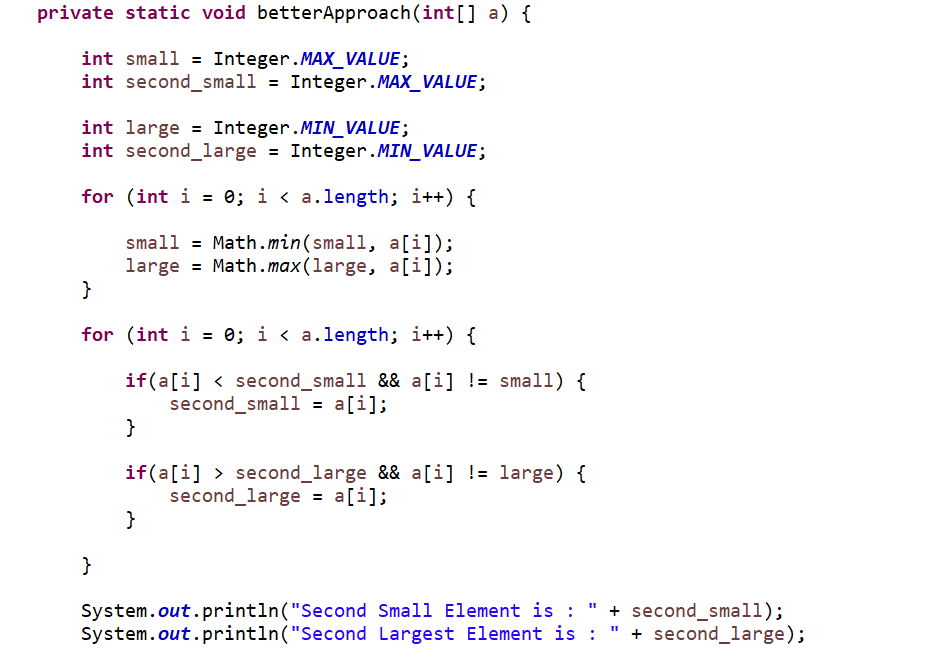
Brute Force Algorithm:

1. Sort the array in ascending order
2. Take the a[n-2] value if array have unique values then it will work

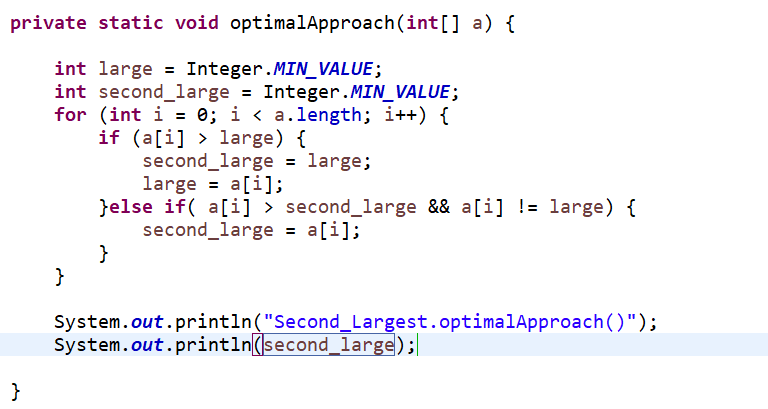
Better Algorithm:

1. What we will do we will find the Largest element & smallest element from the array
2. Then we will iterate the array find the element which is greater the small one and small one from the largest one !!

Code Snippet



BruteForce Algorithm:



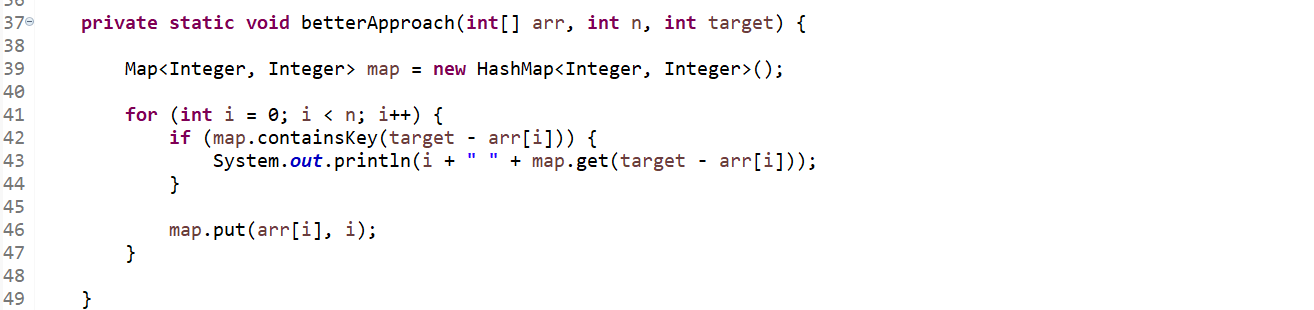
# Two Sum Problem

Brute Force :

1st Iterate the loop and nested loop and check for each element weather the a[i]+a[j] == k

If yes then return the index of both the element

Return new int []{I,j};

Better Apparoch :  


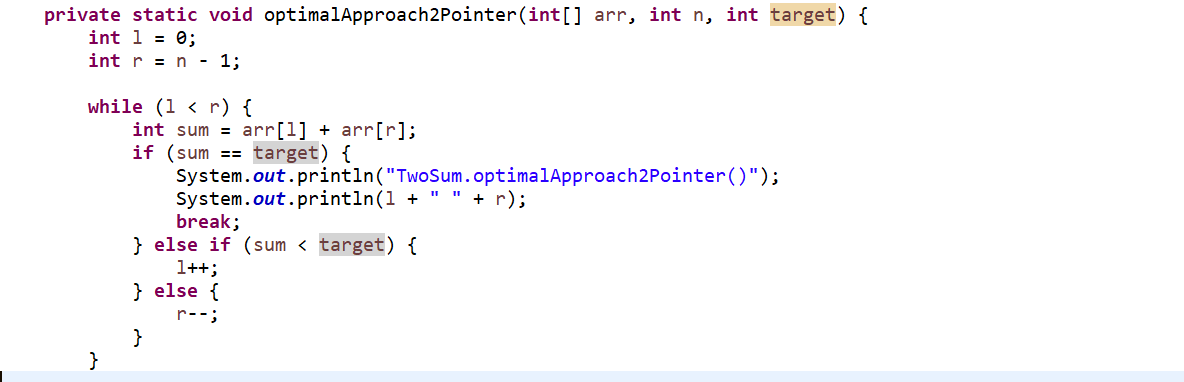
Optimal Approach Using 2 pointer is

If the sum of 2 ptr is equal to target the return the value of 2 ptr

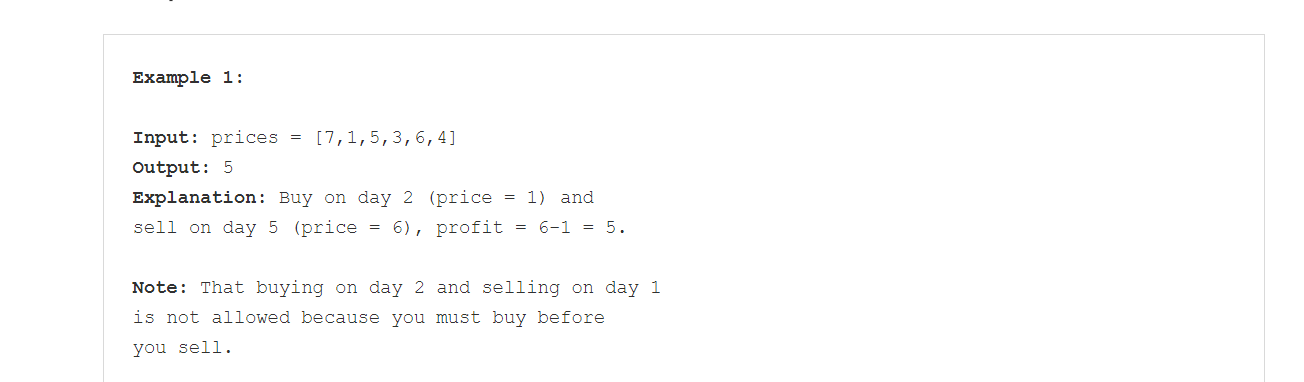
If 2ptr < target then l++

If 2ptr > target r--;

Single loop without space overhead



# Best Time To buy & sell Stock



Brute force Algorithm:

We can use 2 loop to track the value of the stock maintain one variable maxPro contain the max value among all the transaction

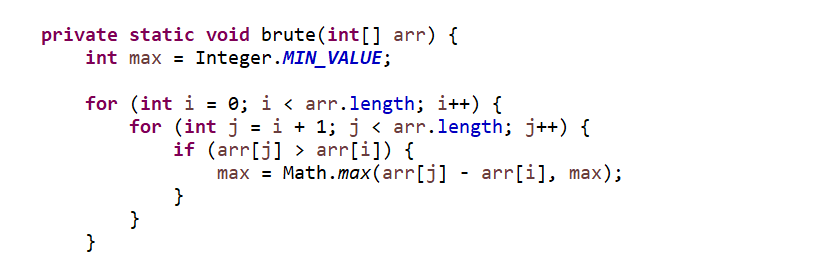
What we can do in this inside the loop we will check

If(arr[j] > arr[i]){

Max = Math.max(max, arr[j]-arr[i]);

}

Here using 2 loops we will get the max profit that we need



Optimal code :

The approach for this algo is

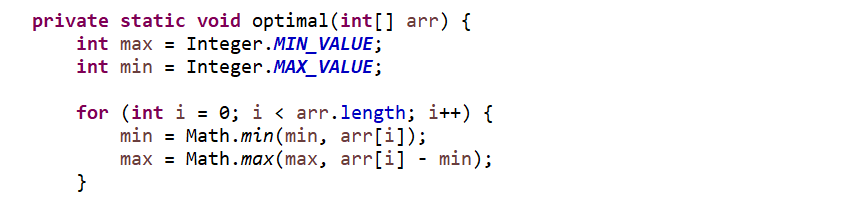
First take 2 variable min max

Iterate the loop from o to n and parallel find the min value from the index and

Max value from the same index

If we get the min value from the array the from the (current idx – min , max) we will find the max value where we need to sell the stocks.

Algorithm of the Code:



# Remove Duplicate Element from In place Array

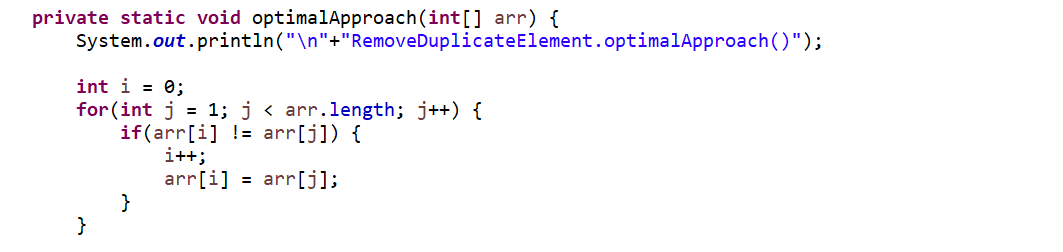
# 

Bruteforce approach says that   
if we have an array of element then what we can do is iterate over the element and store unique value into the HASHSET as it contains only distinct element.

Optimal Approach:

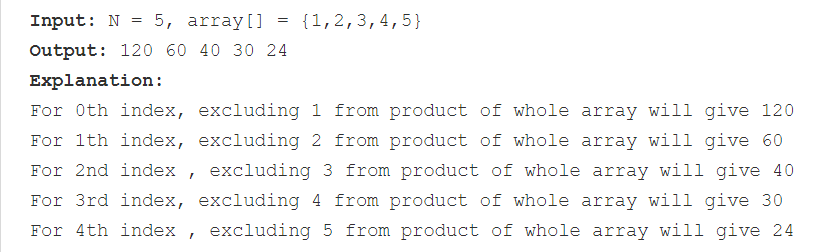
We will user 2 pointer approach to solve this problem and we will check for the value

Weather if a[i] != a[j] then increment the pointer otherwise we will swap



Here we will compare the value if we get the same value then without increasing we will change the value;

# Product Of Array Except itself



Appraoch 1 :   
 we will maintain 2 arrays One for prefix & another one for Suffix :