BLIND 75 (cd /Users/neerajsharma/NeerajLearning/JAVA/Leetcode/Arrays)

1. **Two SUM**

Input: nums = [2,7,11,15], target = 9

Output: [0,1]

Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].

Brute Force: Nested Loop

|  |
| --- |
| **Intuition** |
| We need to find complement element from the array. for eg: 9 - 2=7 then complement is 7 |

|  |
| --- |
| **Learnings** |
| You do not need to start with j=0 you can start with j=i+1 because we are finding “complement” number |

A computer screen shot of a code

Description automatically generated

TC: O(N^2) – nested forloop

SC: O(1) -

Optimized approach Using HashMAP

|  |
| --- |
| **Learnings** |
| **containsKey()** to find they key Value  **put()** to add the element in the map  **get()** to find the value with the help of key. |

|  |
| --- |
| **Mistakes** |
| Initially I use HASHSET but later I realize I need in key value pair where key store the element and value will store it index , so I used HASHMAP. key= nums[i] , value = i |

A screen shot of a computer code

Description automatically generated

TC: O(N) - forloop

SC: O(N) -HASHMAP



Alternative Way:  
A computer screen shot of code

Description automatically generated



1. **Best Time to Buy and Sell Stock**

Input: prices = [7,1,5,3,6,4]

Output: 5

Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.

|  |
| --- |
| **Learnings** |
|  |

Brute Force: Using nested Loop But **TIME LIMIT EXCEEDED!! because of nested loop**  
A screen shot of a computer program

Description automatically generated

Time limit Exceed because of nested loop

**BruteForce :** In 1 for loop with the help of Math.max , Math.min

|  |
| --- |
| **Learnings** |
| Revise HashSet functions |

A computer screen shot of a program code

Description automatically generated

1. **Contains Duplicate**

Given an integer array nums, return true if any value appears at least twice in the array, and return false if every element is distinct.

Example 1:

Input: nums = [1,2,3,1]

Output: true

Example 2:

Input: nums = [1,2,3,4]

Output: false

|  |  |
| --- | --- |
| TC: O(N^2) | O(N^2) for nested for-loop |
| SC: O(1) | No extra space used |

Brute Force: Using Nested for-loop  
A screen shot of a computer program

Description automatically generated

Better Approach : Using Array.sort()

|  |  |
| --- | --- |
| TC: O(NlogN +N) | O(N log N) for Array. Sort() and O(N) for for-loop |
| SC: O(1) | No extra space used |

A screen shot of a computer code

Description automatically generated

Optimized Approach : Using HASHSET

|  |
| --- |
| **Mistakes** |
| * I always forgot to use HashSet as it doesnot contains duplicate. * If you found that current element which is also present in the SET then return true. |

A computer screen shot of a program code

Description automatically generated



1. **Product of Array except itself**

Given an integer array nums, return an array answer such that answer[i] is equal to the product of all the elements of nums except nums[i]. The product of any prefix or suffix of nums is guaranteed to fit in a 32-bit integer. You must write an algorithm that runs in O(n) time and without using the division operation.

Example 1:

Input: nums = [1,2,3,4]

Output: [24,12,8,6]

Example 2:

Input: nums = [-1,1,0,-3,3]

Output: [0,0,9,0,0]

**Better Approach: Using suffix and Prefix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0th | 1st | 2nd | 3rd |
| Nums | 1 | 2 | 3 | 4 |
| Prefix | **1** | 1 | 2 | 6 |
| Nums | 1 | 2 | 3 | 4 |
| Suffix | 24 | 12 | 4 | **1** |
| Result | 24 | 12 | 8 | 6 |

|  |
| --- |
| **Mistakes** |
| * I always forgot to use prefix and suffix * Forgot to for loop limits for all the loops * Forgot to prefix[0]=1 ; suffix[n-1]=1; |

A screen shot of a computer program

Description automatically generated



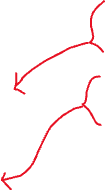
Optimize approach

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0th | 1st | 2nd | 3rd |
| Nums | 1 | 2 | 3 | 4 |
| curr | **1** | 1 | 2 | 6 |
| Result | 1 | 1 | 2 | 6 |
|  |  |  |  |  |
| Nums | 1 | 2 | 3 | 4 |
| curr | 24 | 12 | 4 | **1** |
| Result | 1 | 1 | 2 | 6 |
| Result | 24 | 12 | 8 | 6 |

|  |
| --- |
| **Mistakes** |
| * I always forgot to use prefix and suffix   int curr=1;  for(int i=0;i<n;i++)  {  result[i]\*=curr; // 1  curr\*=nums[i]; //1 1 2 6  }    curr=1;  for(int i=n-1;i>=0;i--)  {  result[i]\*=curr;  curr\*=nums[i];  } |

A screen shot of a computer code

Description automatically generated



1. **Maximum Subarray**

Given an integer array nums, find the subarray with the largest sum, and return its sum.

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

Output: 6

Explanation: The subarray [4,-1,2,1] has the largest sum 6.

**Brute Force :**  
A screen shot of a computer program

Description automatically generated

**Kadane Algorithm :**

|  |
| --- |
| **Mistakes** |
| * Always remember that Max=Sum will come first because if all element are (-)ve then if we donot put the statement before sum=0; then it by default sum will be always equals to 0.   sum+=nums[i];  if(sum>max) max=sum;  if(sum<0) sum=0; |

A computer screen shot of a program code

Description automatically generated



**Dynamic Programming:**

1. **Maximum Product Subarray**

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.

**Brute Force: Using Nested Forloop**  
A computer screen with colorful text

Description automatically generated

|  |
| --- |
| **Mistakes** |
| * Always forgot to use prefix ans suffix product   int prefix=1;  int suffix=1;  for(int i=0; i<nums.length;i++){  if(prefix==0) prefix=1;  if (suffix==0) suffix=1;  prefix\*=nums[i];  suffix\*=nums[nums.length-i-1];  max=Math.max(max,Math.max(prefix,suffix));  } |

**A computer screen shot of a program code

Description automatically generated**



1. **Find Minimum in Rotated Sorted Array**

|  |  |
| --- | --- |
| Example 1:  Input: nums = [3,4,5,1,2]  Output: 1 | Example 2:  Input: nums = [4,5,6,7,0,1,2]  Output: 0 |

**Brute Force: Using Integer.MAX\_VALUE and with a single for-loop.**

|  |  |
| --- | --- |
| TC: O(N) | O((N\*M)**2**\*(N+M)) |
| SC: O(1) | No extra space used |

**A computer code with colorful text

Description automatically generated**

**Better Approach: Using Modified Binary Search Algorithm**

|  |
| --- |
| **Mistakes** |
| * Always forgot how to use modified binary search algorithm over here. |

**`A computer screen shot of a program code

Description automatically generated**



1. **Search in Rotated Sorted Array**

**A computer screen shot of a program code

Description automatically generated**

**Optimized Approach: Using Binary Search   
A computer screen shot of a program code

Description automatically generated**

1. **3- Sum**

**A screen shot of a computer program

Description automatically generated**