Approaches Used for Solving Problems

|  |
| --- |
| * Brute Force * Backtracking * Top-Down Approach (Recursive) * Bottom-Up Approach (Iterative) * Recursion * Kadane’s Algorithm * Prefix sum and Suffix sum * Prefix max and Suffix max * Monotonic order * Boyer Moore Method * Two Pointer Approach * Divide and Conquer * Greedy * Hashing * Searching * Sorting * Randomized * Memorization * Tabulation * Space Optimization |

07.10.2024

Algo tutor Problems

Question 1 :

|  |
| --- |
| Two Sum  Leetcode – https://leetcode.com/problems/two-sum/description/  Difficulty: **Easy** Max Score: **25**  **Problem Description**  You are provided with a non-decreasing array A consisting of N integers. Additionally, you receive an integer target.  Your task is to identify two distinct indices in the array such that the values at these indices add up to the target.  Return these indices as a sorted integer array of length 2.  It is assured that there is exactly one solution for each test case.  The array is 1-indexed.  Note: Complete the given function as specified. The input and output will be managed by the driver code.  Your solution must utilize only constant extra space.  Input Format:   * The first line contains a single integer N. * The second line contains N space-separated integers representing array A. * The third line contains a single integer target.   Output Format:   * Print the result in a new line.   **Example**  **Example 1**  **Input**  5 2 7 11 15 17 9  **Output**  1 2  **Example 2**  **Input**  4 2 3 4 10 6  **Output**  1 3  **Problem Constraints**  1 <= N <= 10000 -100000 <= A[i] <= 100000 -200000 <= target <= 200000 |
|  |

Question 2:

|  |
| --- |
| Maximum Subarray  Leetcode –  Difficulty: **Medium** Max Score: **25**  **Problem Description**  Given an array of integers, write a function to find the maximum sum of a contiguous subarray. This means you need to identify the subarray within the given array that has the largest sum.  **Input Format**   * The first line contains an integer n, representing the number of elements in the array. * The second line contains n space-separated integers, which represent the elements of the array.   **Output Format**   * Print the maximum sum of the subarray.   **Example**  **Example 1**  **Input** 7 -1 2 -3 4 -1 2 1  **Output** 6  **Explanation** The subarray [4, -1, 2, 1] has the maximum sum of 6. |
| Kadane’s Algorithm  class Solution {  public:  int maxSubArray(vector<int>& nums) {  int maxsum=INT\_MIN;  int sum=0;  // kadanes' algorithm  int n=nums.size();  for(int i=0;i<n;i++){  if(sum>=0){  sum+=nums[i];  }  else{  sum=nums[i];  }  maxsum=max(maxsum,sum);  }  return maxsum;  }  }; |

Question 3:

|  |
| --- |
| Subarray sum divisible by k  Difficulty: **Easy** Max Score: **25**  **Problem Description**  Given an integer array nums and an integer k, write a function to find and print the number of non-empty subarrays that have a sum divisible by k.  A subarray is a contiguous part of an array.  **Input Format:**   * The first line contains two integers, n (the size of the array) and k (the divisor). * The second line contains n space-separated integers representing the array nums.   **Output Format:**   * Print the count of subarrays whose sum is divisible by k.   **Example**  **Example 1**  **Input** 5 3 3 1 2 7 4  **Output** 4  **Explanation** There are 4 subarrays with a sum divisible by k = 3: [3] [3, 1, 2] [1, 2] [7, 4]  **Example 2**  Input 6 5  4 5 0 -2 -3 1  Output: 7  **Explanation** There are 7 subarrays with a sum divisible by k = 5:  [4, 5, 0, -2, -3, 1] [5] [5, 0] [5, 0, -2, -3] [0] [0, -2, -3] [-2, -3] |
| Kadane’s Algorithm - Kadane's Algorithm is typically used to find the maximum sum of a contiguous subarray and **Kadane’s Algorithm alone can’t solve the problem directly** because it doesn’t consider cumulative sums (To calculate a cumulative sum, you can start with an initial value of zero and then add each subsequent value to the running total.). |

Question 4:

|  |
| --- |
| Product of Array Except Self  Difficulty: **Medium** Max Score: **25**  **Problem Description**  Create a function that takes an array of integers as input, where the length of the array is greater than 1. The function should return an output array, where each element in the output is the product of all the elements in the input array except the one at the corresponding index.  **For example, if** the input array is [2, 3, 4], the output array should be [12, 8, 6], as each element in the output is the product of all the other elements in the input.  The expected time complexity for the solution should be O(n).  **Input Format**  line 1: contains an integer n denoting number of elements in array.  line 2: contains n separated integers denoting elements of array. All integers are greater than 0.  **Output Format**  Print an array output such that output[i] is equal to the product of all the elements of nums except nums[i].  **Example**  **Example 1**  **Input**  1 2 3 4 5  **Output**  120 60 40 30 24  **Example 2**  **Input**  2 4 6  **Output**  24 12 8  **Problem Constraints**  2 <= n <= 10^5  -30 <= nums[i] <= 30  **Topics** |
|  |

Question 5:

|  |
| --- |
| Subarray Problem 1  Difficulty: **Easy** Max Score: **25**  **Problem Description**  Given an array arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} of size 10, write a function to determine whether there exists a subarray of length n that sums up to a given value s. Both n and s are provided by the user.  Print "YES" if such a subarray exists, otherwise print "NO".  **Input Format:**   * The first and only line of input contains two integers, n (length of the subarray) and s (desired sum of the subarray).   **Output Format:**   * Print "YES" if a subarray of length n with sum s exists in the array, otherwise print "NO".   **Example**  **Example 1**  **Input**: 3 6  **Output:** YES  **Explanation:** The subarray [1, 2, 3] has a sum of 6.  **Example 2**  **Input:** 4 10  **Output:** YES  **Explanation:** The subarray `[1, 2, 3, 4]` has a sum of 10. |
|  |

Question 6:

|  |
| --- |
| Print the pattern  Difficulty: **Easy** Max Score: **25**  **Problem Description**  Using recursion, print the given pattern.  **Input Format**  First line contains integer n, representing rows of the given pattern  **Output Format**  Display the given pattern.  **Example**  **Example 1**  **Input** 5  **Output** \*  \* \*  \* \* \*  \* \* \* \*  \* \* \* \* \*  **Explanation** The number of rows given is 5. |
|  |

Question 7:

|  |
| --- |
| Switch Case Problem 1  Difficulty: **Easy** Max Score: **10**  **Problem Description**  You are given a positive integer M. Write a program using a switch case to print the following output based on the value of M:   * If the value is 42, print "Education is crucial". * For any other value, print "Undefined input".   **Input Format:**   * The first line contains an integer M.   **Output Format:**   * Print the corresponding message based on the integer M as described.   **Example**  **Example 1**  **Input** 42  **Output** Education is crucial |
|  |

Question 8:

|  |
| --- |
| Switch Case Problem 2  Difficulty: **Easy** Max Score: **25**  **Problem Description**  You are given a positive integer X. Write a program using a switch case such that:   * If the user enters 18, print "I am an even number". * If the user enters 19, print "I am a prime number". * If the user enters 33, print "I am an odd number". * For any other input, print "I am just another number".   **Input Format:**   * The first line contains an integer X.   **Output Format:**   * Print the sentence related to the integer as per the description.   **Example**  **Example 1**  **Input** 33  **Output** I am an odd number |
|  |

Question 10:

|  |
| --- |
| Sum of Natural Numbers  Difficulty: **Easy** Max Score: **25**  **Problem Description**  You are given a **whole number 𝑁**. Your task is to find and print the sum of all **natural numbers**up to 𝑁.  **Natural numbers**are positive numbers starting from 1 and going up forever.  **Input Format:**  The first line has a whole number 𝑁.  **Output Format:**  Print the sum of the first 𝑁natural numbers.  **Example**  **Example 1**  **Input**  5  **Output**  15  **Explanation**  Here, n = 5, so 1 + 2 + 3 + 4 + 5 = 15 |
|  |

Question 11:

|  |
| --- |
| Check Positive or Not  Difficulty: **Easy** Max Score: **25**  **Problem Description**  Write a program to a number and check if it is positive or negative.  **Input Format**  The first line contains a number n.  **Output Format**  Print 1 for positive and 0 for negative  **Example**  **Example 1**  **Input**  200  **Output**  1 |
|  |

Question 12:

|  |
| --- |
| Operations in Array  Difficulty: **Easy** Max Score: **25**  **Problem Description**  Given an array arr of size n, Write a program to find the below 3 values.  Find Sum of all the elements in the array.  Find Average of all the elements in the array (floor value is  given).  Find Maximum element in the array.  **Input Format:**  First line contains n representing the number of elements.  Next line contains representing elements of the array.  **Output Format:**  In one line, output three space-separated integers representing the sum, average, and maximum values among all the elements in the array. |
|  |

Question 13:

|  |
| --- |
| Algo Tutor Conditional Problem 2  Difficulty: **Easy** Max Score: **25**  **Problem Description**  Given two integers a and b. Your task is to print Yes if one of these is 10 or their sum is 10, else print No.  **Input Format**  The First line contains two integers a and b.  **Output Format**  Print Yes if the given condition satisfies else print No.  **Example**  **Example 1**  **Input**  6 4  **Output**  Yes  **Example 2**  **Input**  8 1  **Output**  No |
|  |

NOTES :

ZOOM LINK - <https://shorturl.at/uD0a2>

WEBSITE LINK - <https://academy.algotutor.io/>

Leetcode 151 – Reverse words in a string https://leetcode.com/problems/reverse-words-in-a-string/description/

|  |
| --- |
| class Solution {  public String reverseWords(String s) {  String[] str = s.trim().split("\\s+");  String ans="";  for(int i=str.length-1;i>=0;i--){  if(i!=0)  ans+=str[i]+" ";  else  ans+=str[i];  }  return ans;  }  } |
| class Solution {      public String reverseWords(String s) {          StringBuilder ans = new StringBuilder(s);          StringBuilder finalstr = ans.reverse();          return finalstr.toString();      }  } |

<https://www.hackerrank.com/challenges/30-running-time-and-complexity/problem>

|  |
| --- |
| #include <cmath>  #include <cstdio>  #include <vector>  #include <iostream>  #include <algorithm>  using namespace std;  int main() {      /\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/      int testcases;      cin>>testcases;      while(testcases>0){          int num;          cin>>num;          // if(num<=1){          //     cout<<"Not prime"<<endl;          // }          // else{          //     bool isnotprime = false;          //     for(int i=2;i<=sqrt(num);i++){          //         if(num%i==0){          //             isnotprime = true;          //             break;          //         }          //     }          //     if(isnotprime){          //         cout<<"Not prime"<<endl;          //     }          //     else{          //         cout<<"Prime"<<endl;          //     }          // }            int count = 0;          for(int i=1;i\*i<=num;i++){              if(num%i == 0){                  if(i==num/i) count = count+1;                  else count = count + 2;              }          }          if(count == 2){              cout<<"Prime"<<endl;          }          else{              cout<<"Not prime"<<endl;          }            testcases--;      }      return 0;  } |

DOES EVERY NUMBERS HAVE DIVISORS THAT ARE LESSER THAN THE SQRT ROOT OF THE GIVEN NUMBER

Solution - In conclusion, every composite number has at least one divisor less than or equal to its square root. For prime numbers, however, we only have 1 and the number itself as divisors, so no divisors less than the square root exist other than 1.

|  |
| --- |
| Sum of all elements in an array  // Online Java Compiler  // Use this editor to write, compile and run your Java code online  import java.util.\*;  class HelloWorld {  public static void main(String[] args) {  Scanner in=new Scanner(System.in);  int n = in.nextInt();  int a[] = new int [n];  for(int i=0;i<n;i++){  a[i]=in.nextInt();  }  int sum=0;  for(int i=0;i<n;i++){  sum+=a[i];  }  System.out.print(sum);  }  } |
| Second largest element in an array - https://www.geeksforgeeks.org/problems/second-largest3735/1?itm\_source=geeksforgeeks&itm\_medium=article&itm\_campaign=practice\_card  // Online Java Compiler  // Use this editor to write, compile and run your Java code online  import java.util.\*;  class HelloWorld {  public static void main(String[] args) {  Scanner in=new Scanner(System.in);  int n = in.nextInt();  int a[] = new int [n];  for(int i=0;i<n;i++){  a[i]=in.nextInt();  }  int largest = a[0];  int secondlargest = -1;  for(int i=0;i<n;i++){  if(largest<a[i]){  secondlargest = largest;  largest = a[i];  }  else if(secondlargest<a[i] && a[i] < largest);  {  secondlargest=a[i];  }  }  System.out.print(secondlargest);  }  }  // to avoid duplicates  // example a.size = 5  // elements 14 2 5 13 5  // output 13  // old code output 5 |
| Count all the numbers which has atleast 1 greater number of that element in the array  import java.util.\*;  class CountSmallerElements {  public static void main(String[] args) {  Scanner in = new Scanner(System.in);  int n = in.nextInt(); // Read the size of the array  int a[] = new int[n]; // Array to store the elements  // Reading the elements of the array  for (int i = 0; i < n; i++) {  a[i] = in.nextInt();  }  // Step 1: Find the maximum element in the array  int maxElement = a[0];  for (int i = 1; i < n; i++) {  if (a[i] > maxElement) {  maxElement = a[i];  }  }  // Step 2: Count elements that are smaller than the maximum element  int count = 0;  for (int i = 0; i < n; i++) {  if (a[i] < maxElement) {  count++;  }  }  // Output the result  System.out.println(count);  }  } |
| Approach 2  import java.util.\*;  class HelloWorld {  public static void main(String[] args) {  Scanner in = new Scanner(System.in);  int n = in.nextInt();  int a[] = new int[n];    // Read the array elements  for (int i = 0; i < n; i++) {  a[i] = in.nextInt();  }    // Sort the array  Arrays.sort(a);    // Find the count of the largest element at the end of the sorted array  int maxCount = 0;  int maxElement = a[n - 1];    for (int i = n - 1; i >= 0; i--) {  if (a[i] == maxElement) {  maxCount++;  } else {  break;  }  }    // Result is the total number of elements minus the number of largest elements  System.out.print(n - maxCount);  }  } |
| Reverse the array – modify the array with no extra space  // Online Java Compiler  // Use this editor to write, compile and run your Java code online  import java.util.\*;  class HelloWorld {  public static void main(String[] args) {  Scanner in=new Scanner(System.in);  int n = in.nextInt();  int a[] = new int [n];  for(int i=0;i<n;i++){  a[i]=in.nextInt();  }  int start = 0;  int end = n-1;  while(start<end){  int temp = a[start];  a[start]=a[end];  a[end] = temp;    start++;  end--;  }  for(int i=0;i<n;i++){  System.out.print(a[i]+" ");  }  }  } |
| Print a pattern  \*  \* \* \*  \* \* \* \* \*  \* \* \*  \*  #include <iostream>  using namespace std;  int main() {  int n = 5; // Number of rows  int mid = n / 2; // Midpoint for pattern adjustment  for (int i = 0; i < n; i++) {  int nsp = 0; // Initialize number of spaces  int nst = 0; // Initialize number of stars  // Calculate number of spaces and stars for the current row  if (i <= mid) { // Upper part (including middle row)  nsp = mid - i; // Spaces decrease  nst = 2 \* i + 1; // Stars increase  } else { // Lower part  nsp = i - mid; // Spaces increase  nst = 2 \* (n - i - 1) + 1; // Stars decrease  }  // Print spaces  for (int j = 0; j < nsp; j++) {  cout << " ";  }  // Print stars  for (int j = 0; j < nst; j++) {  cout << "\*";  }  cout << endl; // Move to the next line  }  return 0; } |
| 4 5  1 2 3 4 5  6 7 8 9 10  11 12 13 14 15  16 17 18 19 20  1 2 3 4 5 10 9 8 7 6 11 12 13 14 15 20 19 18 17 16  // Online C++ compiler to run C++ program online  #include <iostream>  using namespace std;  int main() {    int n,m;  cin>>n>>m;  int a[n][m];  for(int i=0;i<n;i++){  for(int j=0;j<m;j++){  cin>>a[i][j];  }  }  for(int i=0;i<n;i++){  if(i%2==0){  for(int j=0;j<m;j++){  cout<<a[i][j]<<" ";    }  }  else{  for(int j=m-1;j>=0;j--){  cout<<a[i][j]<<" ";  }  }  }  return 0;  } |
| Transpose of a matrix  3 3  1 2 3  4 5 6  7 8 9  1 4 7 2 5 8 3 6 9  // Online C++ compiler to run C++ program online  #include <iostream>  using namespace std;  int main() {    int n,m;  cin>>n>>m;  int a[n][m];  for(int i=0;i<n;i++){  for(int j=0;j<m;j++){  cin>>a[i][j];  }  }  for(int i=0;i<m;i++){  for(int j=0;j<n;j++){  cout<<a[j][i]<<" ";  }  }  return 0;  } |
| CODING ASSESMENT – 07.10.2024  Previous greater element |
| ASSESMENT UNSOLVED QUESTIONS |

08.09.2024

|  |
| --- |
| Printing matrix in spiral order  Given Matrix  1 2 3 4 5  6 7 8 9 10  11 12 13 14 15  16 17 18 19 20  21 22 23 24 25    Required Output  1 2 3 4 5 10 15 20 25 24 23 22 21 16 17 18 19 14 13 12 11 6 7 8 9  1 2 3 4 5  6 7 8 9 10  11 12 13 14 15  16 17 18 19 20  21 22 23 24 25  // Online C++ compiler to run C++ program online  #include <iostream>  using namespace std;  int main() {  int n,m;  cin>>n>>m;  int a[n][m];  for(int i=0;i<n;i++){  for(int j=0;j<m;j++){  cin>>a[i][j];  }  }  // int n=5;  // int m=5;  // int a[n][m]={{1,2,3,4,5},  // {6,7,8,9,10},  // {11,12,13,14,15},  // {16,17,18,19,20},  // {21,22,23,24,25}  // };    int top=0;  int bottom=n-1;  int right=m-1;  int left=0;  int rightin=m-2;  int upin=n-2;  for(int i=0;i<m;i++){  cout<<a[top][i]<<" ";  }  for(int i=1;i<n;i++){  cout<<a[i][right]<<" ";  }  for(int i=m-2;i>=0;i--){  cout<<a[bottom][i]<<" ";  }  if(n%2==0){  for(int i=n-2;i>0;i--)  {  if(i%2==0)  {  for(int j=0;j<=m-2;j++)  {  cout<< a[i][j]<<" ";  }  }  else  {  for(int j=m-2;j>=0;j--){  cout<<a[i][j]<<" ";  }  }  }  }  else{  for(int i=n-2;i>0;i--)  {  if(i%2==0)  {  for(int j=m-2;j>=0;j--){  cout<<a[i][j]<<" ";  }  }  else  {  for(int j=0;j<=m-2;j++)  {  cout<< a[i][j]<<" ";  }  }  }  }      return 0;  } |
| Best time to buy and sell stock  Leetcode - <https://leetcode.com/problems/best-time-to-buy-and-sell-stock/description/>  class Solution {      public int maxProfit(int[] prices) {          int maxprofit = 0;          int minbuy = prices[0];          int currprofit=0;          for(int i=1;i<prices.length;i++){              currprofit = prices[i]-minbuy;              maxprofit = Math.max(currprofit,maxprofit);              minbuy = Math.min(prices[i],minbuy);          }          return maxprofit;      }  } |
| Running Sum of 1D Array  Leetcode – <https://leetcode.com/problems/running-sum-of-1d-array/description/>  class Solution {      public int[] runningSum(int[] nums) {          int arr[]=new int[nums.length];          for(int i=1;i<nums.length;i++){              nums[i]=nums[i-1]+nums[i];          }          return nums;      }  } |
| Range Sum Query  Leetcode – <https://leetcode.com/problems/range-sum-query-immutable/>  class NumArray {      private int[] prefixsum;      public NumArray(int[] nums) {          this.prefixsum = new int[nums.length];          int sum=0;          for(int i=0;i<prefixsum.length;i++){              sum+=nums[i];              this.prefixsum[i]=sum;          }      }        public int sumRange(int left, int right) {          if(left == 0){              return prefixsum[right];          }          return prefixsum[right]-prefixsum[left-1];      }  } |
| Product of array except self  Leetcode – https://leetcode.com/problems/product-of-array-except-self/  class Solution {      public int[] productExceptSelf(int[] nums) {          int n=nums.length;          int arr[]=new int[n];          int l=1;          int r=1;          for(int i=0;i<n;i++){              arr[i]=l;              l\*=nums[i];          }          for(int i=n-1;i>=0;i--){              arr[i]\*=r;              r\*=nums[i];          }          return arr;      }  } |
| Trapping rain water  Leetcode - <https://leetcode.com/problems/trapping-rain-water/>  class Solution {      public int trap(int[] height) {          int n=height.length;          int left[]=new int[n];          int right[]=new int[n];          left[0]=height[0];          right[n-1]=height[n-1];          for(int i=1;i<n;i++){              left[i]=Math.max(left[i-1],height[i]);          }          for(int i=n-2;i>=0;i--){              right[i]=Math.max(right[i+1],height[i]);          }          int minheight=0;          int solution=0;          for(int i=0;i<n;i++){              minheight=Math.min(left[i],right[i]);              solution+=minheight-height[i];          }          return solution;      }  } |
| Max chunks to make sorted  Leetcode - <https://leetcode.com/problems/max-chunks-to-make-sorted/>  class Solution {      public int maxChunksToSorted(int[] arr) {          int y=-1;          int j=0;          int h=arr.length;          for(int i=0;i<h;i++){              y=Math.max(y,arr[i]);              if(y==i) j++;          }          return j;      }  } |
| Max chunks to make sorted II  Leetcode - <https://leetcode.com/problems/max-chunks-to-make-sorted-ii/>  class Solution {      public int maxChunksToSorted(int[] arr) {         // same as partitioning use for trap water problem         int n=arr.length;         int left=0;         int right=n-1;         int leftarr[]=new int[n];         int rightarr[]=new int[n];         leftarr[0]=arr[0];         rightarr[n-1]=arr[n-1];         for(int i=1;i<n;i++){              leftarr[i]=Math.max(leftarr[i-1],arr[i]);         }         for(int i=n-2;i>=0;i--){          rightarr[i]=Math.min(rightarr[i+1],arr[i]);         }         int chunks=1; // as entire array can be partitioned as chunks         for(int i=0;i<n-1;i++){          if(leftarr[i]<=rightarr[i+1]){              chunks++;          }         }         return chunks;      }  } |
| Toeplitz matrix  Leetcode – https://leetcode.com/problems/toeplitz-matrix/  class Solution {      public boolean isToeplitzMatrix(int[][] a) {          int n=a.length;          int m=a[0].length;          for(int i=1;i<n;i++){              for(int j=0;j<m-1;j++){                  if(a[i-1][j]!=a[i][j+1]){                      return false;                  }              }          }          return true;      }  } |
| Majority element I  Leetcode – https://leetcode.com/problems/majority-element/  class Solution {      public int majorityElement(int[] nums) {         int n=nums.length;         int count=0;         int majority=-1;         for(int i=0;i<n;i++){          if(count==0){              majority = nums[i];              count=1;          }          else{              if(nums[i] == majority){                  count++;              }              else{                  count--;              }          }         }         count=0;         for(int i=0;i<n;i++){          if(nums[i]==majority){              count++;          }         }          if(count>(nums.length/2)){              return majority;          }          return -1;      }  } |
| Majority Element II  Leetcode – https://leetcode.com/problems/majority-element-ii/description/  class Solution {      public List<Integer> majorityElement(int[] nums) {          int n=nums.length;          int count1=0;          int count2=0;          int m1=-1;          int m2=-1;          for(int i=0;i<n;i++){              if(count1==0 && m2!=nums[i]){                  count1=1;                  m1=nums[i];              }              else if(count2==0 && m1!=nums[i]){                  count2=1;                  m2=nums[i];              }              else if(nums[i]==m1){                  count1++;              }              else if(nums[i]==m2){                  count2++;              }              else{                  count1--;                  count2--;              }          }          List<Integer> ans=new ArrayList<>();          count1=0;          count2=0;          for(int i=0;i<n;i++){              if(nums[i]==m1) count1++;              if(nums[i]==m2) count2++;          }          int mini = (int)(n/3)+1;          if(m1!=m2){          if(count1>=mini) ans.add(m1);          if(count2>=mini) ans.add(m2);          }          else{              if(count1>=mini) ans.add(m1);          }          return ans;      }  } |

09.09.2024

|  |
| --- |
| Remove duplicates from sorted array  Leetcode - <https://leetcode.com/problems/remove-duplicates-from-sorted-array/description/>  class Solution {      public int removeDuplicates(int[] nums) {          int n=nums.length;          int min=nums[0];          for(int i=1;i<n-1;i++){              if(nums[i]==min){                  nums[i]=nums[i+1];              }          }      }  } |
| Find pair given difference  Geeks for Geeks - <https://www.geeksforgeeks.org/problems/find-pair-given-difference1559/1>  class Solution {  public:  int findPair(int n, int x, vector<int> &arr) {  // code here  sort(arr.begin(),arr.end());  int left=0;  int right=1;  int diff=0;  while(right<n){  if(left!=right && arr[right]-arr[left]==x)  {  return 1;  }  else if(arr[right]-arr[left]<x){  right++;  }  else{  left++;  }  }  return -1;  }  }; |
| Container with most water  Leetcode – https://leetcode.com/problems/container-with-most-water/  class Solution {      public int maxArea(int[] height) {          int n=height.length;          int i=0;          int j=n-1;          int h=0,w=0,maxarea=0;          while(i<j){              h=Math.min(height[i],height[j]);              w=j-i;              maxarea= Math.max(maxarea,h\*w);              if(height[i]<=height[j]){                  i++;              }              else{                  j--;              }          }          return maxarea;      }  } |
| Range Sum Addition –  Lintcode – https://www.lintcode.com/problem/903/ |
| Increasing Triplet Subsequence  Leetcode – https://leetcode.com/problems/increasing-triplet-subsequence  class Solution {      public boolean increasingTriplet(int[] nums) {          int n=nums.length;          if(nums.length<3) return false;          int first=Integer.MAX\_VALUE;;          int second=Integer.MAX\_VALUE;          for(int i=0;i<n;i++){              if(nums[i]<=first){                  first=nums[i];              }              else if(nums[i]<=second){                  second=nums[i];              }              else{                  return true;              }          }          return false;      }  } |
| Car pooling  Leetcode – <https://leetcode.com/problems/car-pooling/description/>  class Solution {  public:      bool carPooling(vector<vector<int>>& trips, int capacity) {          int n=trips.size();          vector<vector<int>> arr;          for(int i=0;i<n;++i){              arr.push\_back({trips[i][1],1,trips[i][0]});              arr.push\_back({trips[i][2],0,trips[i][0]});          }          sort(arr.begin(),arr.end());          int current\_capacity = 0;          for(int i=0;i<arr.size();++i){              if(arr[i][1]==1) current\_capacity += arr[i][2];              else current\_capacity -= arr[i][2];              if(current\_capacity > capacity) return false;          }          return true;      }  }; |
| Valid triangle number  Leetcode – https://leetcode.com/problems/valid-triangle-number/description/  class Solution {      public int triangleNumber(int[] nums) {          // Logic          // sum of two sides must be greater than or equal to third side          //sort the array          Arrays.sort(nums);          int n=nums.length;          int result=0;          for(int i=0;i<n-2;i++){              for(int j=i+1;j<n-1;j++){                  for(int k=j+1;k<n;k++){                      if(nums[i]+nums[j]>nums[k]){                          result++;                      }                  }              }          }          return result;      }  } |
| 3 Sum  Leetcode – https://leetcode.com/problems/3sum/ |

|  |
| --- |
| 14.10.2024  Subarray  No of subarrays from given array – (n\*(n+1))/2  Printing all subarrays from given array  // To print all subarrays of an array  import java.util.Scanner;  class subarray{  public static void main(String[] args){  Scanner in=new Scanner(System.in);  int n=in.nextInt();  int a[]=new int[n];  for(int i=0;i<n;i++){  a[i]=in.nextInt();  }    // Printing all subarrays    for(int i=0;i<n;i++){  for(int j=i;j<n;j++){  for(int k=i;k<=j;k++){  System.out.print(a[k]+" ");  }  System.out.println();  }  System.out.println();  }  in.close();  }  }  Output  5  1 2 3 4  5  1  1 2  1 2 3  1 2 3 4  1 2 3 4 5  2  2 3  2 3 4  2 3 4 5  3  3 4  3 4 5  4  4 5  5 |
| Printing sum of each subarray  import java.util.Scanner;  class subarray{  public static void main(String[] args){  Scanner in=new Scanner(System.in);  int n=in.nextInt();  int a[]=new int[n];  for(int i=0;i<n;i++){  a[i]=in.nextInt();  }    // Printing sum of each subarray  int sum=0;  for(int i=0;i<n;i++){  for(int j=i;j<n;j++){  sum=0;  for(int k=i;k<=j;k++){  System.out.print(a[k]+" ");  sum+=a[k];  }  System.out.println();  System.out.print("Sum ");  System.out.print(sum);  System.out.println();  }  System.out.println();  }  in.close();  }  }  Output  5  1 2 3 4 5  1  Sum 1  1 2  Sum 3  1 2 3  Sum 6  1 2 3 4  Sum 10  1 2 3 4 5  Sum 15  2  Sum 2  2 3  Sum 5  2 3 4  Sum 9  2 3 4 5  Sum 14  3  Sum 3  3 4  Sum 7  3 4 5  Sum 12  4  Sum 4  4 5  Sum 9  5  Sum 5  === Code Execution Successful ===  O(n^2) approach  import java.util.Scanner;  class subarray{  public static void main(String[] args){  Scanner in=new Scanner(System.in);  int n=in.nextInt();  int a[]=new int[n];  for(int i=0;i<n;i++){  a[i]=in.nextInt();  }    // Printing sum of each subarray  int sum=0;  for(int i=0;i<n;i++){  sum=0;  for(int j=i;j<n;j++){  sum+=a[j];  System.out.println(sum);  }  }  in.close();  }  }  Output  5  1 2 3 4 5  1  3  6  10  15  2  5  9  14  3  7  12  4  9  5  === Code Execution Successful === |
| Printing total sum of all subarrays  import java.util.Scanner;  class subarray{  public static void main(String[] args){  Scanner in=new Scanner(System.in);  int n=in.nextInt();  int a[]=new int[n];  for(int i=0;i<n;i++){  a[i]=in.nextInt();  }    // Printing sum of each subarray  int sum=0;  int totalsum=0;  for(int i=0;i<n;i++){  for(int j=i;j<n;j++){  sum=0;  for(int k=i;k<=j;k++){    sum+=a[k];  }  totalsum+=sum;    }    }  System.out.print(totalsum);  in.close();  }  }  Output  5  1 2 3 4 5  105  === Code Execution Successful ===  O(n^2) approach  import java.util.Scanner;  class subarray{  public static void main(String[] args){  Scanner in=new Scanner(System.in);  int n=in.nextInt();  int a[]=new int[n];  for(int i=0;i<n;i++){  a[i]=in.nextInt();  }    // Printing sum of each subarray  int sum=0;  int totalsum=0;  for(int i=0;i<n;i++){  sum=0;  for(int j=i;j<n;j++){  sum+=a[j];  totalsum+=sum;  }  }  System.out.print(totalsum);  in.close();  }  }  Output  5  1 2 3 4 5  105  === Code Execution Successful === |
| O(n) approach  Finding totalsum of all subarrays  // to find total sum by using O(n) approach  import java.util.Scanner;  class subarray{  public static void main(String[] args){  Scanner in=new Scanner(System.in);  int n=in.nextInt();  int a[]=new int[n];  for(int i=0;i<n;i++){  a[i]=in.nextInt();  }    // Printing sum of each subarray  int totalsum=0;  for(int i=0;i<n;i++){  totalsum+=a[i]\*(i+1)\*(n-i);  }  System.out.print(totalsum);  in.close();  }  }  Output  5  1 2 3 4 5  105  === Code Execution Successful === |

15.10.2024

|  |
| --- |
| Insertion Sort Algorithm  Leetcode - https://leetcode.com/problems/sort-colors/description/  // In Place Sorting Algorithms - Insertion sort and Quick Sort  // Insertion sort  #include<iostream>  using namespace std;  int main(){  int n;  cin>>n;  int a[n];  for(int i=0;i<n;i++){  cin>>a[i];  }    for(int i=0;i<n;i++){  int key=a[i];  int j=i-1;  for(;j>=0 && a[j]>key;j--){  a[j+1]=a[j];  }  a[j+1]=key;  }    for(int i=0;i<n;i++){  cout<<a[i]<<" ";  }  } |
| Find the duplicate number  Leetcode - <https://leetcode.com/problems/find-the-duplicate-number/description/>  O(n)  class Solution {  public int findDuplicate(int[] nums) {  boolean count[]=new boolean[nums.length];  for(int i=0;i<nums.length;i++){  if(count[nums[i]]){  return nums[i];  }  count[nums[i]]=true;  }  return 0;  }  }  O(n logn)  class Solution {  public int findDuplicate(int[] nums) {  Arrays.sort(nums);  for(int i=0;i<nums.length-1;i++){  if(nums[i]==nums[i+1]){  return nums[i];  }  }  return 0;  }  }  O(n^2)  class Solution {      public int findDuplicate(int[] nums) {          for(int i=0;i<nums.length-1;i++){             for(int j=i+1;j<nums.length;j++){                  if(nums[i] == nums[j]){                      // As given -> There is only one repeated number in nums.                      return nums[j];                  }             }          }          return 0;      }  } |