**DSA Notes for Arsh Goyal SDE List By Alankruthi**

**Arrays:**

1) **287. Find the Duplicate Number**

**Link:** <https://leetcode.com/problems/find-the-duplicate-number/>

**Code:**

class Solution {

public int findDuplicate(int[] nums) {

//Linked List cycle approach

int slow=nums[0];

int fast=nums[0];

do{

slow=nums[slow];

fast=nums[nums[fast]];

}while(slow!=fast);

fast=nums[0];

while(slow!=fast){

slow=nums[slow];

fast=nums[fast];

}//Duplicate number is when it collides at 2nd time

return slow;

}

}

T.C: O(N)

S.C: O(1)

**O/P:**

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

**Output:** 2

2) **75. Sort Colors**

**Link:** [**https://leetcode.com/problems/sort-colors/**](https://leetcode.com/problems/sort-colors/)

**Code:**class Solution {

public void sortColors(int[] nums) {

//Dutch national Flag Algorithm

int low=0,mid=0;

int high=nums.length-1;

while(mid<=high){

switch(nums[mid]){

case 0: int temp=nums[low];

nums[low]=nums[mid];

nums[mid]=temp;

low++;mid++;

break;

case 1: mid++;

break;

case 2: temp=nums[high];

nums[high]=nums[mid];

nums[mid]=temp;

high--;

break;

}

}

}

}

T.C: O(N)

S.C: O(1)

**Input:** nums = [2,0,2,1,1,0]

**Output:** [0,0,1,1,2,2]

3. **26. Remove Duplicates from Sorted Array**

**Link:** [**https://leetcode.com/problems/remove-duplicates-from-sorted-array/**](https://leetcode.com/problems/remove-duplicates-from-sorted-array/)

**Code:**

import java.util.\*;

class Solution {

public int removeDuplicates(int[] nums) {

//Two Pointer Approach

int i=0,j=1;

for(j=1;j<nums.length;j++){

if(nums[j]!=nums[i]){

i++;

nums[i]=nums[j];

}

}

return i+1;

}

}

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

**Output:** 2, nums = [1,2, \_]

T.C:O(N)

S.C:O(1)

4. **73. Set Matrix Zeroes**

**Link:** [**https://leetcode.com/problems/set-matrix-zeroes/**](https://leetcode.com/problems/set-matrix-zeroes/)

**Code:**

class Solution {

public void setZeroes(int[][] matrix) {

int col0=1,rowSize=matrix.length,colSize=matrix[0].length;

for (int i=0;i<rowSize;i++){

if(matrix[i][0]==0) col0=0;

for (int j=1;j<colSize;j++){

if(matrix[i][j]==0){

matrix[i][0]=0;

matrix[0][j]=0;

}

}

}

for(int i=rowSize-1;i>=0;i--){

for(int j=colSize-1;j>=1;j--){

if(matrix[i][0]==0 || matrix[0][j]==0){

matrix[i][j]=0;

}

}

if(col0==0){

matrix[i][0]=0;

}

}

}

}

T.C: O(2\*(N+M))

S.C:O(1)



**Input:** matrix = [[0,1,2,0],[3,4,5,2],[1,3,1,5]]

**Output:** [[0,0,0,0],[0,4,5,0],[0,3,1,0]]

5. **283. Move Zeroes**

**Link:** [**https://leetcode.com/problems/move-zeroes/**](https://leetcode.com/problems/move-zeroes/)

**Code:**

class Solution {

public void moveZeroes(int[] nums) {

int left=0,right=0;

while(right<nums.length){

if(nums[right]==0){

right++;

}

else{

int temp=nums[left];

nums[left]=nums[right];

nums[right]=temp;

left++;

right++;

}

}

}

}

T.C: O(n)

S.C:O(1)

Output:

**Input:** nums = [0,1,0,3,12]

**Output:** [1,3,12,0,0]

6. **121. Best Time to Buy and Sell Stock**

**Link:** [**https://leetcode.com/problems/best-time-to-buy-and-sell-stock/**](https://leetcode.com/problems/best-time-to-buy-and-sell-stock/)

**Code:**

class Solution {

public int maxProfit(int[] prices) {

int min=Integer.MAX\_VALUE,profit=0;

for(int i=0;i<prices.length;i++){

if(prices[i]<min){

min=prices[i];

}

else if(profit<(prices[i]-min)){

profit=prices[i]-min;

}

}

return profit;

}

}

T.C: O(N)

S.C: O(1)

**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.

**7. Chocolate Distribution Problem**

**Link:** [**https://www.geeksforgeeks.org/chocolate-distribution-problem/**](https://www.geeksforgeeks.org/chocolate-distribution-problem/)

**Code:**

class Solution

{

public long findMinDiff (ArrayList<Long> a, long n, long m)

{

// your code here

long minDiff=Integer.MAX\_VALUE;

Collections.sort(a);

for(int i=0;i<=a.size()-m;i++){

long v=a.get(i+(int)m-1)-a.get(i);

if(v<minDiff){

minDiff=v;

}

}

return minDiff;

}

}

T.C:O(N)

S.C:O(1)

**Input:**

N = 8, M = 5

A = {3, 4, 1, 9, 56, 7, 9, 12}

**Output:** 6

**Explanation:** The minimum difference between

maximum chocolates and minimum chocolates

is 9 - 3 = 6 by choosing following M packets :

{3, 4, 9, 7, 9}.

8. **1. Two Sum**

**Link:** [**https://leetcode.com/problems/two-sum/**](https://leetcode.com/problems/two-sum/)

**Code:**

class Solution {

public int[] twoSum(int[] nums, int target) {

//Using hashmap

int[] res=new int[2];

HashMap<Integer,Integer> hm=new HashMap<>();

for(int i=0;i<nums.length;i++){

if(hm.containsKey(target-nums[i])){

res[1]=i;

res[0]=hm.get(target-nums[i]);

return res;

}

hm.put(nums[i],i);

}

return res;

}

}

T.C: O(N)

S.C:O(N)

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

**Output:** [0,1]

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

9. **122. Best Time to Buy and Sell Stock II**

**Link:** [**https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/**](https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/)

**Code:**

class Solution {

public int maxProfit(int[] prices) {

int maxP=0;

for(int i=1;i<prices.length;i++){

if(prices[i]>prices[i-1]){

maxP+=prices[i]-prices[i-1];

}

}

return maxP;

}

}

T.C:O(N)

S.C:O(1)

10) **974. Subarray Sums Divisible by K**

**Link:** [**https://leetcode.com/problems/subarray-sums-divisible-by-k/**](https://leetcode.com/problems/subarray-sums-divisible-by-k/)

**Code:**

class Solution {

public int subarraysDivByK(int[] nums, int k) {

HashMap<Integer,Integer> hm = new HashMap<>();

hm.put(0,1);

int sum=0;

int rem=0;

int count=0;

for(int i=0;i<nums.length;i++){

sum+=nums[i];

rem=sum%k;

if(rem<0){

rem+=k;

}

if(hm.containsKey(rem)){

count+=hm.get(rem);

hm.put(rem,hm.get(rem)+1);

}

else{

hm.put(rem,1);

}

}

return count;

}

}

T.C: O(N)

S.C:O(N)

**Input:** nums = [4,5,0,-2,-3,1], k = 5

**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]

11) **442. Find All Duplicates in an Array**

**Link:** [**https://leetcode.com/problems/find-all-duplicates-in-an-array/**](https://leetcode.com/problems/find-all-duplicates-in-an-array/)

**Code:**

class Solution {

public List<Integer> findDuplicates(int[] nums) {

ArrayList<Integer> res=new ArrayList<>();

for(int i=0;i<nums.length;i++){

int ind=Math.abs(nums[i])-1;

if(nums[ind]<0){

res.add(Math.abs(nums[i]));

}

else{

nums[ind]\*=-1;

}

}

return res;

}

}

T.C:O(N)  
S.C:O(1)

**Input:** nums = [4,3,2,7,8,2,3,1]

**Output:** [2,3]

12) **11. Container With Most Water**

**Link:** [**https://leetcode.com/problems/container-with-most-water/**](https://leetcode.com/problems/container-with-most-water/)

**Code:**

class Solution {

public int maxArea(int[] height) {

int left=0,right=height.length-1,maxWater=0;

while(left<right){

int width=right-left;

int heigh=Math.min(height[left],height[right]);

int area=width\*heigh;

if(area>maxWater){

maxWater=area;

}

if(height[left]<=height[right]){

left++;

}

else{

right--;

}

}

return maxWater;

}

}

T.C: O(N)

S.C:O(1)

**Input:** height = [1,8,6,2,5,4,8,3,7]

**Output:** 49

**Explanation:** The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.

13) **15. 3Sum**

**Link:** [**https://leetcode.com/problems/3sum/**](https://leetcode.com/problems/3sum/)

**Code:**

class Solution {

public List<List<Integer>> threeSum(int[] nums) {

Arrays.sort(nums);

List<List<Integer>> res=new ArrayList<>();

for(int i=0;i<nums.length-2;i++){

if(i==0 || i>0 && nums[i]!=nums[i-1]){

int low=i+1,high=nums.length-1,tar=0-nums[i];

while(low<high){

if((nums[low]+nums[high])==tar){

res.add(Arrays.asList(nums[i],nums[low],nums[high]));

while(low<high && nums[low]==nums[low+1]) low++;

while(low<high && nums[high]==nums[high-1]) high--;

low++;high--;

}

else if((nums[low]+nums[high])<tar){

low++;

}

else{

high--;

}

}

}

}

return res;

}

}

T.C: O(N^2)

S.C:O(M)

**Input:** nums = [-1,0,1,2,-1,-4]

**Output:** [[-1,-1,2],[-1,0,1]]

14. **18. 4Sum**

**Link:** [**https://leetcode.com/problems/4sum/**](https://leetcode.com/problems/4sum/)

**Code:**

class Solution {

public List<List<Integer>> fourSum(int[] nums, int target) {

List<List<Integer>> res=new ArrayList<>();

Arrays.sort(nums);

for(int i=0;i<nums.length;i++){

for(int j=i+1;j<nums.length;j++){

int low=j+1,high=nums.length-1,tar=target-(nums[i]+nums[j]);

while(low<high){

if((nums[low]+nums[high])<tar){

low++;

}

else if((nums[low]+nums[high])>tar){

high--;

}

else{

res.add(Arrays.asList(nums[i],nums[j],nums[low],nums[high]));

while(low<high && nums[low]==nums[low+1]) low++;

while(low<high && nums[high]==nums[high-1]) high--;

low++;high--;

}

}

while(j+1<nums.length && nums[j+1]==nums[j]) j++;

}

while(i+1<nums.length && nums[i+1]==nums[i]) i++;

}

return res;

}

}

T.C:O(N^3)

S.C:O(1)

**Input:** nums = [1,0,-1,0,-2,2], target = 0

**Output:** [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]

15. **1423. Maximum Points You Can Obtain from Cards**

**Link:** [**https://leetcode.com/problems/maximum-points-you-can-obtain-from-cards/**](https://leetcode.com/problems/maximum-points-you-can-obtain-from-cards/)

**Code:**

class Solution {

public int maxScore(int[] cardPoints, int k) {

//Sliding Window Technique

int maxSum=0,currSum=0,len=cardPoints.length,i=0;

for(int i=0;i<k;i++){

currSum+=cardPoints[i];

}

maxSum=currSum;

while(i<k){

currSum=currSum-cardPoints[k-i-1]+cardPoints[len-i-1];

if(maxSum<currSum) maxSum=currSum;

i++;

}

return maxSum;

}

}

T.C:O(K)  
S.C:O(1)

**Input:** cardPoints = [1,2,3,4,5,6,1], k = 3

**Output:** 12

**Explanation:** After the first step, your score will always be 1. However, choosing the rightmost card first will maximize your total score. The optimal strategy is to take the three cards on the right, giving a final score of 1 + 6 + 5 = 12.

16. **560. Subarray Sum Equals K**

**Link:** [**https://leetcode.com/problems/subarray-sum-equals-k/**](https://leetcode.com/problems/subarray-sum-equals-k/)

**Code:**

class Solution {

public int subarraySum(int[] nums, int k) {

//Using HashMap

int currSum=0,count=0;

HashMap<Integer,Integer> hm=new HashMap<>();

for(int i=0;i<nums.length;i++){

currSum+=nums[i];

if(currSum==k){

count+=1;

}

if(hm.containsKey(currSum-k)){

count+=hm.get(currSum-k);

}

if(hm.containsKey(currSum)){

hm.put(currSum,hm.get(currSum)+1);

}

else{

hm.put(currSum,1);

}

}

return count;

}

}

T.C: O(N)

S.C:O(N)

**Input:** nums = [1,1,1], k = 2

**Output:** 2

17. **54. Spiral Matrix**

**Link:** [**https://leetcode.com/problems/spiral-matrix/**](https://leetcode.com/problems/spiral-matrix/)

**Code:**

class Solution {

public List<Integer> spiralOrder(int[][] matrix) {

int top=0,down=matrix.length-1,left=0,right=matrix[0].length-1,dir=0;

List<Integer> result=new ArrayList<>();

while(top<=down && left<=right){

if(dir==0){

//Right

for(int i=left;i<=right;i++){

result.add(matrix[top][i]);

}

top++;

}

else if(dir==1){

//Down

for(int i=top;i<=down;i++){

result.add(matrix[i][right]);

}

right--;

}

else if(dir==2){

//Left

for(int i=right;i>=left;i--){

result.add(matrix[down][i]);

}

down--;

}

else if(dir==3){

//Top

for(int i=down;i>=top;i--){

result.add(matrix[i][left]);

}

left++;

}

dir=(dir+1)%4;

}

return result;

}

}

T.C:O(N)

S.C:O(1)



**Input:** matrix = [[1,2,3],[4,5,6],[7,8,9]]

**Output:** [1,2,3,6,9,8,7,4,5]

18. **79. Word Search**

**Link:** [**https://leetcode.com/problems/word-search/**](https://leetcode.com/problems/word-search/)

**Code:**

class Solution {

public boolean search(char[][] board,int i,int j,int pos, String word){

if(pos==word.length()-1) return true;

board[i][j]-=65;

//top

if(i>0 && board[i-1][j]==word.charAt(pos+1) && search(board,i-1,j,pos+1,word)) return true;

//left

if(j>0 && board[i][j-1]==word.charAt(pos+1) && search(board,i,j-1,pos+1,word)) return true;

//bottom

if(i<board.length-1 && board[i+1][j]==word.charAt(pos+1) && search(board,i+1,j,pos+1,word)) return true;

//right

if(j<board[0].length-1 && board[i][j+1]==word.charAt(pos+1) && search(board,i,j+1,pos+1,word)) return true;

board[i][j]+=65;

return false;

}

public boolean exist(char[][] board, String word) {

for(int i=0;i<board.length;i++){

for(int j=0;j<board[0].length;j++){

if(board[i][j]==word.charAt(0) && search(board,i,j,0,word)) return true;

}

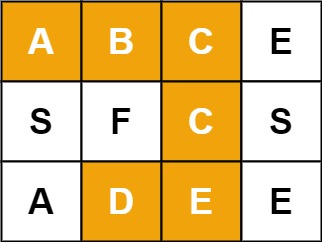
}

return false;

}

}

T.C:O(N)  
S.C:O(1)



**Input:** board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCCED"

**Output:** true

19. **55. Jump Game**

**Link:** [**https://leetcode.com/problems/jump-game/**](https://leetcode.com/problems/jump-game/)

**Code:**

class Solution {

public boolean canJump(int[] nums) {

int reachable=0;

for(int i=0;i<nums.length;i++){

if(reachable<i){

return false;

}

reachable=Math.max(reachable,nums[i]+i);

}

return true;

}

}

T.C:O(N)  
S.C:O(1)

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

**Output:** true

**Explanation:** Jump 1 step from index 0 to 1, then 3 steps to the last index.

20. **88. Merge Sorted Array**

**Link:** [**https://leetcode.com/problems/merge-sorted-array/**](https://leetcode.com/problems/merge-sorted-array/)

**Code:**

class Solution {

public void merge(int[] nums1, int m, int[] nums2, int n) {

int i=m-1,j=n-1,k=m+n-1;

while(j>=0){

if(i>=0 && nums1[i]>nums2[j]){

nums1[k--]=nums1[i--];

}

else{

nums1[k--]=nums2[j--];

}

}

}

}

T.C:O(M+N)  
S.C:O(1)

**Input:** nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3

**Output:** [1,2,2,3,5,6]

**Explanation:** The arrays we are merging are [1,2,3] and [2,5,6].

The result of the merge is [1,2,2,3,5,6] with the underlined elements coming from nums1.

21 **169. Majority Element**

**Link:** [**https://leetcode.com/problems/majority-element/**](https://leetcode.com/problems/majority-element/)

**Code:**

class Solution {

public int majorityElement(int[] nums) {

int major=nums[0],count=1;

for(int i=1;i<nums.length;i++){

if(nums[i]!=major){

count-=1;

if(count==0) {

major=nums[i];

count=1;

}

}

else{

count++;

}

}

return major;

}

}

T.C:O(N)  
S.C:O(1)  
**Input:** nums = [3,2,3]

**Output:** 3

22 **493. Reverse Pairs**

**Link:** [**https://leetcode.com/problems/reverse-pairs/**](https://leetcode.com/problems/reverse-pairs/)

**Code:**

class Solution {

public int merge(int low,int mid,int high,int[] nums){

int j=mid+1,count=0;

for(int i=low;i<=mid;i++){

while(j<=high && nums[i]>(2\*(long)nums[j])){

j++;

}

count+=j-(mid+1);

}

ArrayList<Integer> temp=new ArrayList<>();

int x=low,y=mid+1;

while(x<=mid && y<=high){

if(nums[x]<=nums[y]){

temp.add(nums[x++]);

}

else{

temp.add(nums[y++]);

}

}

while(x<=mid){

temp.add(nums[x++]);

}

while(y<=high){

temp.add(nums[y++]);

}

for(int l=low;l<=high;l++){

nums[l]=temp.get(l-low);

}

return count;

}

public int mergeSort(int low,int high,int[] nums){

if(low==high){

return 0;

}

int mid=(low+high)/2;

int count=mergeSort(low,mid,nums);

count+=mergeSort(mid+1,high,nums);

count+=merge(low,mid,high,nums);

return count;

}

public int reversePairs(int[] nums) {

return mergeSort(0,nums.length-1,nums);

}

}

T.C: O(NlogN)

S.C:O(N)

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

**Output:** 2

23. **77. Combinations**

**Link:** [**https://leetcode.com/problems/combinations/**](https://leetcode.com/problems/combinations/)

**Code:**

class Solution {

public void combinations(int[] arr,int n, int k,int index,List<Integer> temp

,List<List<Integer>> res){

if(temp.size()==k){

res.add(new ArrayList(temp));

return;

}

for(int i=index;i<n;i++){

temp.add(arr[i]);

combinations(arr,n,k,i+1,temp,res);

temp.remove(temp.get(temp.size()-1));

}

}

public List<List<Integer>> combine(int n, int k) {

int[] arr=new int[n];

for(int i=0;i<n;i++){

arr[i]=i+1;

}

List<List<Integer>> res=new ArrayList<>();

combinations(arr,n,k,0,new ArrayList<>(),res);

return res;

}

}

T.C:O(N-R)

S.C:O(N)

**Input:** n = 4, k = 2

**Output:**

[

[2,4],

[3,4],

[2,3],

[1,2],

[1,3],

[1,4],

]

24. **289. Game of Life**

**Link:** [**https://leetcode.com/problems/game-of-life/**](https://leetcode.com/problems/game-of-life/)

**Code:**

// Original ||| New ||| State

// 0 ||| 0 ||| 0

// 1 ||| 0 ||| 1

// 0 ||| 1 ||| 2

// 1 ||| 1 ||| 3

// 1 -> 2 or 3 live-> 1 else 0

// 0 -> 3 live -> 1 else 0

class Solution {

public int countNeighbours(int i,int j,int[][] board){

int nei=0;

for(int x=i-1;x<=i+1;x++){

for(int y=j-1;y<=j+1;y++){

if((x==i && y==j) || x<0 || y<0 || x==board.length || y==board[0].length){

continue;

}

if(board[x][y]==1 || board[x][y]==3){

nei++;

}

}

}

return nei;

}

public void gameOfLife(int[][] board) {

int rows=board.length,cols=board[0].length,neigh=0;

for(int i=0;i<rows;i++){

for(int j=0;j<cols;j++){

neigh=countNeighbours(i,j,board);

if(board[i][j]==1){

if(neigh==2 || neigh==3){

board[i][j]=3;

}

}

if(board[i][j]==0){

if(neigh==3){

board[i][j]=2;

}

}

}

}

for(int i=0;i<rows;i++){

for(int j=0;j<cols;j++){

if(board[i][j]==1){

board[i][j]=0;

}

else if(board[i][j]==2 || board[i][j]==3){

board[i][j]=1;

}

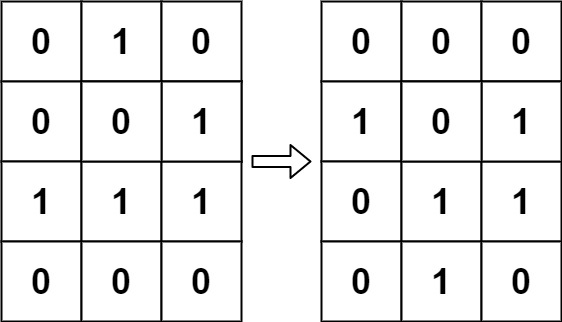
}

}

}

}

**Example 1:**



**Input:** board = [[0,1,0],[0,0,1],[1,1,1],[0,0,0]]

**Output:** [[0,0,0],[1,0,1],[0,1,1],[0,1,0]]

T.C:O(m\*n)

S.C:O(1)

25. **1499. Max Value of Equation**

**Link:** [**https://leetcode.com/problems/max-value-of-equation/**](https://leetcode.com/problems/max-value-of-equation/)

**Code:**

class Solution {

//curr value is i and queue value is j

public int findMaxValueOfEquation(int[][] points, int k) {

Deque<int[]> d=new ArrayDeque<>();

int res=Integer.MIN\_VALUE;

for(int i=0;i<points.length;i++){

//Checking the condition of |xi - xj| <= k i<j

while((!d.isEmpty()) && points[i][0]-d.peek()[0]>k){

d.pop();

}

int plus=points[i][1]+points[i][0];

int diff=points[i][1]-points[i][0];

//yi + yj + |xi - xj|==> yi+xi+(yj-xj)

if(!d.isEmpty()){

res=Math.max(res,plus+d.peek()[1]);

}

//Making it work like priority Queue like maximum value at peek

while((!d.isEmpty()) && diff>d.getLast()[1]){

d.removeLast();

}

d.add(new int[]{points[i][0],diff});

}

return res;

}

}

T.C:O(Nlogn)

S.C:O(N)

**Input:** points = [[1,3],[2,0],[5,10],[6,-10]], k = 1

**Output:** 4

**Explanation:** The first two points satisfy the condition |xi - xj| <= 1 and if we calculate the equation we get 3 + 0 + |1 - 2| = 4. Third and fourth points also satisfy the condition and give a value of 10 + -10 + |5 - 6| = 1.

No other pairs satisfy the condition, so we return the max of 4 and 1.

26. **381. Insert Delete GetRandom O(1) - Duplicates allowed**

**Link:** [**https://leetcode.com/problems/insert-delete-getrandom-o1-duplicates-allowed/**](https://leetcode.com/problems/insert-delete-getrandom-o1-duplicates-allowed/)

**Code:**

class RandomizedCollection {

//val- corresponding indices

HashMap<Integer,HashSet<Integer>> hmap;

ArrayList<Integer> arr=new ArrayList<>();

Random r;

public RandomizedCollection() {

hmap=new HashMap<>();

arr=new ArrayList<>();

r=new Random();

}

public boolean insert(int val) {

boolean flag;

HashSet<Integer> set;

if(hmap.containsKey(val)==true){

set=hmap.get(val);

flag=false;

}

else{

set=new HashSet<>();

flag=true;

}

set.add(arr.size());

arr.add(val);

hmap.put(val,set);

return flag;

}

public boolean remove(int val) {

HashSet<Integer> set;

if(hmap.containsKey(val)==true){

set=hmap.get(val);

int remidx=-1;

for(int i:set){

remidx=i;

break;

}

set.remove(remidx);

hmap.put(val,set);

// if there is a val in hashmap with no size discard it

if(hmap.get(val).size()==0){

hmap.remove(val);

}

if(remidx==arr.size()-1){

arr.remove(remidx);

}

else{

arr.set(remidx,arr.get(arr.size()-1));

arr.remove(arr.size()-1);

int newval=arr.get(remidx);

HashSet<Integer> newset;

newset=hmap.get(newval);

newset.remove(arr.size());

newset.add(remidx);

hmap.put(newval,newset);

}

}

else{

return false;

}

return true;

}

public int getRandom() {

int randidx=r.nextInt(arr.size());

return arr.get(randidx);

}

}

/\*\*

\* Your RandomizedCollection object will be instantiated and called as such:

\* RandomizedCollection obj = new RandomizedCollection();

\* boolean param\_1 = obj.insert(val);

\* boolean param\_2 = obj.remove(val);

\* int param\_3 = obj.getRandom();

\*/

**Input**

["RandomizedCollection", "insert", "insert", "insert", "getRandom", "remove", "getRandom"]

[[], [1], [1], [2], [], [1], []]

**Output**

[null, true, false, true, 2, true, 1]

T.C:O(1)

27. **84. Largest Rectangle in Histogram**

**Link:** [**https://leetcode.com/problems/largest-rectangle-in-histogram/**](https://leetcode.com/problems/largest-rectangle-in-histogram/)

**Code:**

class Solution {

public int largestRectangleArea(int[] heights) {

int[] left=new int[heights.length];

int[] right=new int[heights.length];

Stack<Integer> st=new Stack<>();

int maxArea=0;

for(int i=0;i<heights.length;i++){

while(!st.isEmpty() && heights[st.peek()]>=heights[i]){

st.pop();

}

if(st.isEmpty()){

left[i]=0;

}

else{

left[i]=st.peek()+1;

}

st.push(i);

}

while(!st.isEmpty()){

st.pop();

}

for(int i=heights.length-1;i>=0;i--){

while(!st.isEmpty() && heights[st.peek()]>=heights[i]){

st.pop();

}

if(st.isEmpty()){

right[i]=heights.length-1;

}

else{

right[i]=st.peek()-1;

}

st.push(i);

}

System.out.println();

for(int i=0;i<heights.length;i++){

int area=(right[i]-left[i]+1)\*heights[i];

if(maxArea<area){

maxArea=area;

}

}

return maxArea;

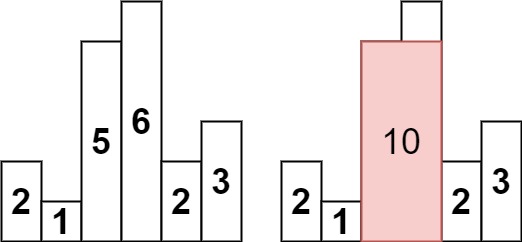
}

}

T.C:O(N)

S.C:O(3N)

**Example 1:**



**Input:** heights = [2,1,5,6,2,3]

**Output:** 10

**Explanation:** The above is a histogram where width of each bar is 1.

The largest rectangle is shown in the red area, which has an area = 10 units.