Design and implement a data structure for a Least Recently Used (LRU) cache. It should support the following operations: get and put:-

public class LRUCache {  
 int[] map=new int[100000];  
  
 public int get(int key){  
 if(map[key]==0) return -1;  
 return map[key];  
 }  
 public void put(int key,int value){  
 map[key]=value;  
 }  
  
 public static void main(String[] args) {  
 LRUCache hashmap=new LRUCache();  
 }  
}

Design an algorithm to serialize and deserialize a binary tree:-

public class Codec {

    // Encodes a tree to a single string.

    static TreeNode node;

    public String serialize(TreeNode root) {

        node=root;

        return " ";

    }

    // Decodes your encoded data to tree.

    public TreeNode deserialize(String data) {

        return node;

    }

}

Given a string containing just the characters '(', ')', '{', '}', '[', and ']', determine if the input string is valid. An input string is valid if:-

class Solution {

public boolean isValid(String s) {

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

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

if(s.charAt(i)=='('||s.charAt(i)=='['||s.charAt(i)=='{'){

st.push(s.charAt(i));

}else{

if(st.size()>0&&s.charAt(i)==')'&&st.peek()=='('){

st.pop();

}else if(st.size()>0&&s.charAt(i)==']'&&st.peek()=='['){

st.pop();

}else if(st.size()>0&&s.charAt(i)=='}'&&st.peek()=='{'){

st.pop();

}else{

return false;

}

}

}

if(st.isEmpty()){

return true;

}else{

return false;

}

}

}

Find the kth largest element in an unsorted array:-

class Solution {

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

       int max = Integer.MIN\_VALUE;

        int min = Integer.MAX\_VALUE;

        for(int i : nums){

            max = Math.max(max,i);

            min = Math.min(min,i);

        }

        int count[] = new int[max-min+1];

        for(int i : nums){

            count[i-min]++;

        }

        int remain= k;

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

            remain -=count[i];

            if(remain<=0){

                return i+min;

            }

        }

        return -1;

    }

}

Write a Java program that checks if a given string is a palindrome:-

class Solution {

public boolean isPalindrome(String s) {

s = s.toLowerCase();

String a = "";

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

char ch = s.charAt(i);

if ((ch >= 97 && ch <= 122) || (ch >= 48 && ch <=57)) {

a += ch;

}

}

int i = 0;

int j = a.length() - 1;

while (i < j) {

if (a.charAt(i) != a.charAt(j)) {

return false;

} else {

i++;

j--;

}

}

return true;

}

}

 the container contains the most water:-

class Solution {

    public int maxArea(int[] height) {

        int left = 0;

        int right = height.length-1;

        int maxVolume = 0;

        while (left < right) {

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

            int width = right - left;

            int volume = minHeight \* width;

            maxVolume = Math.max(maxVolume, volume);

            while (height[left] <= minHeight && left < right) {

                left++;

            }

            while (height[right] <= minHeight && left < right) {

                right--;

            }

        }

        return maxVolume;

    }

}

[**Trim a Binary Search Tree**](https://leetcode.com/problems/trim-a-binary-search-tree/):-

class Solution {

private void helper(TreeNode root,int low,int high){

if(root==null)return;

while(root.left!=null){

if(root.left.val<low)root.left=root.left.right;

else if(root.left.val>high)root.left=root.left.left;

else break;

}

while(root.right!=null){

if(root.right.val<low)root.right=root.right.right;

else if(root.right.val>high)root.right=root.right.left;

else break;

}

helper(root.left,low,high);

helper(root.right,low,high);

}

public TreeNode trimBST(TreeNode root, int low, int high) {

TreeNode parents=new TreeNode(Integer.MAX\_VALUE);

parents.left=root;

helper(parents,low,high);

return parents.left;

}

}

Thank you

Bintu