100. Same Tree

**Question**

Given two binary trees, write a function to check if they are equal or not.

Two binary trees are considered equal if they are structurally identical and the nodes have the same value.

**Solution1**

public boolean isSameTree(TreeNode p, TreeNode q) {

if(p == null && q == null) return true;

if(p == null || q == null) return false;

if(p.val == q.val)

return isSameTree(p.left, q.left) && isSameTree(p.right, q.right);

return false;

}

**Solution2 non-recursive preorder traverse**

public boolean isSameTree(TreeNode p, TreeNode q) {

Stack<TreeNode> stack\_p = new Stack <> ();

Stack<TreeNode> stack\_q = new Stack <> ();

if (p != null) stack\_p.push( p ) ;

if (q != null) stack\_q.push( q ) ;

while (!stack\_p.isEmpty() && !stack\_q.isEmpty()) {

TreeNode pn = stack\_p.pop() ;

TreeNode qn = stack\_q.pop() ;

if (pn.val != qn.val) return false ;

if (pn.right != null) stack\_p.push(pn.right) ;

if (qn.right != null) stack\_q.push(qn.right) ;

if (stack\_p.size() != stack\_q.size()) return false ;

if (pn.left != null) stack\_p.push(pn.left) ;

if (qn.left != null) stack\_q.push(qn.left) ;

if (stack\_p.size() != stack\_q.size()) return false ;

}

return stack\_p.size() == stack\_q.size() ;

}

**Complexity**

121. Best Time to Buy and Sell Stock

**Question**

Say you have an array for which the *i*th element is the price of a given stock on day *i*.

If you were only permitted to complete at most one transaction (ie, buy one and sell one share of the stock), design an algorithm to find the maximum profit.

Example 1:

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

Output: 5

max. difference = 6-1 = 5 (not 7-1 = 6, as selling price needs to be larger than buying price)

**Solution**

public class Solution {

public int maxProfit(int prices[]) {

int minprice = Integer.MAX\_VALUE;

int maxprofit = 0;

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

if (prices[i] < minprice)

minprice = prices[i];

else if (prices[i] - minprice > maxprofit)

maxprofit = prices[i] - minprice;

}

return maxprofit;

}

}

**Complexity**

Time complexity : O(n)*O*(*n*). Only a single pass is needed.

Space complexity : O(1)*O*(1). Only two variables are used.

217. Contains Duplicate

**Question**

Given an array of integers, find if the array contains any duplicates. Your function should return true if any value appears at least twice in the array, and it should return false if every element is distinct.

**Solution1 sorting**

public boolean containsDuplicate(int[] nums) {

Arrays.sort(nums);

for (int i = 0; i < nums.length - 1; ++i) {

if (nums[i] == nums[i + 1]) return true;

}

return false;

}

**Complexity**

Time complexity : O(nlogn). Sorting is O(nlogn) and the sweeping is O(n). The entire algorithm is dominated by the sorting step, which is O(nlogn).

Space complexity : O(1). Space depends on the sorting implementation which, usually, costs O(1) auxiliary space if heapsort is used.

**Solution2 HashTable**

public boolean containsDuplicate(int[] nums) {

Set<Integer> set = new HashSet<>(nums.length);

for (int x: nums) {

if (set.contains(x)) return true;

set.add(x);

}

return false;

}

**Complexity Analysis**

Time complexity : O(n). We do search() and insert() for nn times and each operation takes constant time.

Space complexity : O(n). The space used by a hash table is linear with the number of elements in it.

237. Delete Node in a Linked List

**Question**

Write a function to delete a node (except the tail) in a singly linked list, given only access to that node.

Supposed the linked list is 1 -> 2 -> 3 -> 4 and you are given the third node with value 3, the linked list should become 1 -> 2 -> 4 after calling your function.

**Solution**

public void deleteNode(ListNode node) {

node.val = node.next.val;

node.next = node.next.next;

}

**Complexity**

Time and space complexity are both *O*(1).

292. Nim Game

**Question**

You are playing the following Nim Game with your friend: There is a heap of stones on the table, each time one of you take turns to remove 1 to 3 stones. The one who removes the last stone will be the winner. You will take the first turn to remove the stones.

Both of you are very clever and have optimal strategies for the game. Write a function to determine whether you can win the game given the number of stones in the heap.

For example, if there are 4 stones in the heap, then you will never win the game: no matter 1, 2, or 3 stones you remove, the last stone will always be removed by your friend.

**Solution**

public boolean canWinNim(int n) {

return (n % 4 != 0);

}

**Complexity Analysis**

Time complexity is *O*(1), since only one check is performed. No additional space is used, so space complexity is also *O*(1).

344. Reverse String

**Question**

Write a function that takes a string as input and returns the string reversed.

Example:  
Given s = "hello", return "olleh".

**Solution1**

public class Solution {

public String reverseString(String s) {

char[] word = s.toCharArray();

int i = 0;

int j = s.length() - 1;

while (i < j) {

char temp = word[i];

word[i] = word[j];

word[j] = temp;

i++;

j--;

}

return new String(word);

}

}

**Solution2**

public class Solution {

public String reverseString(String s) {

return new StringBuilder(s).reverse().toString();

}

}

**Complexity**

Time Complexity: `O(n)` (Average Case) and `O(n)` (Worst Case) where `n` is the total number character in the input string. The algorithm need to reverse the whole string.

Auxiliary Space: `O(n)` space is used where `n` is the total number character in the input string. Space is needed to transform string to character array.

374. Guess Number Higher or Lower

**Question**

We are playing the Guess Game. The game is as follows:

I pick a number from **1** to ***n***. You have to guess which number I picked.

Every time you guess wrong, I'll tell you whether the number is higher or lower.

You call a pre-defined API guess(int num) which returns 3 possible results (-1, 1, or 0):

-1 : My number is lower

1 : My number is higher

0 : Congrats! You got it!

**Solution1**

*/\* The guess API is defined in the parent class GuessGame.*

*@param num, your guess*

*@return -1 if my number is lower, 1 if my number is higher, otherwise return 0*

*int guess(int num); \*/*

public class Solution extends GuessGame {

public int guessNumber(int n) {

int low = 1;

int high = n;

while (low <= high) {

int mid = low + (high - low) / 2;

int res = guess(mid);

if (res == 0)

return mid;

else if (res < 0)

high = mid - 1;

else

low = mid + 1;

}

return -1;

}

}

**Complexity**

Time complexity : O(log​2​​ n). Binary Search is used.

Space complexity : O(1). No extra space is used.

**Solution2**

public class Solution extends GuessGame {

public int guessNumber(int n) {

int low = 1;

int high = n;

while (low <= high) {

int mid1 = low + (high - low) / 3;

int mid2 = high - (high - low) / 3;

int res1 = guess(mid1);

int res2 = guess(mid2);

if (res1 == 0)

return mid1;

if (res2 == 0)

return mid2;

else if (res1 < 0)

high = mid1 - 1;

else if (res2 > 0)

low = mid2 + 1;

else {

low = mid1 + 1;

high = mid2 - 1;

}

}

return -1;

**}**

**}**

**Complexity**

Time complexity : O(log​3​​ n). Ternary Search is used.

Space complexity : O(1). No extra space is used.

412. Fizz Buzz

**Question**

Write a program that outputs the string representation of numbers from 1 to *n*.

But for multiples of three it should output “Fizz” instead of the number and for the multiples of five output “Buzz”. For numbers which are multiples of both three and five output “FizzBuzz”.

**Solution1**

public class Solution {

public List<String> fizzBuzz(int n) {

List<String> ret = new ArrayList<String>(n);

for(int i=1,fizz=0,buzz=0;i<=n ;i++){

fizz++;

buzz++;

if(fizz==3 && buzz==5){

ret.add("FizzBuzz");

fizz=0;

buzz=0;

}else if(fizz==3){

ret.add("Fizz");

fizz=0;

}else if(buzz==5){

ret.add("Buzz");

buzz=0;

}else{

ret.add(String.valueOf(i));

}

}

return ret;

}

}

**Solution2**

public class Solution {

public List<String> fizzBuzz(int n) {

List<String> list = new ArrayList<>();

for (int i = 1; i <= n; i++) {

if (i % 3 == 0 && i % 5 == 0) {

list.add("FizzBuzz");

} else if (i % 3 == 0) {

list.add("Fizz");

} else if (i % 5 == 0) {

list.add("Buzz");

} else {

list.add(String.valueOf(i));

}

}

return list;

}

}

**Complexity**

O(n)

104. Maximum Depth of Binary Tree

**Question**

Given a binary tree, find its maximum depth.

The maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.

**Solution**

**Complexity**

344. Reverse String

**Question**

**Solution**

**Complexity**

344. Reverse String

**Question**

**Solution**

**Complexity**

344. Reverse String

**Question**

**Solution**

**Complexity**

344. Reverse String

**Question**

**Solution**

**Complexity**