

- [Home](#)
- [Simple Java](#)
- [Coding Interview](#)
- [Machine Learning](#)
- [Java Examples](#)
- [Python Examples](#)
- [Scala Examples](#)
- [Contact](#)

Top 10 Algorithms for Coding Interview

[Latest PDF](#), [Latest Problem Classifications](#)

The following are the common subjects in coding interviews. As understanding those concepts requires much more effort, this tutorial only serves as an introduction. The subjects that are covered include: 1) *String/Array/Matrix*, 2) *Linked List*, 3) *Tree*, 4) *Heap*, 5) *Graph*, 6) *Sorting*, 7) *Dynamic Programming*, 8) *Bit Manipulation*, 9) *Combinations and Permutations*, and 10) *Math Problems*. I highly recommend you to read "[Simple Java](#)" first, if you need a brief review of Java basics. If you want to see code examples that show how to use a popular API, you can use [JavaSED.com](#).

1. String/Array

An algorithm problem's input is often a string or array. Without auto-completion of any IDE, the following methods should be remembered.

```
toCharArray() //get char array of a String
charAt(int x) //get a char at the specific index
length() //string length
length //array size
substring(int beginIndex)
substring(int beginIndex, int endIndex)
Integer.valueOf()//string to integer
String.valueOf()//integer to string
Arrays.sort() //sort an array
Arrays.toString(char[] a) //convert to string
Arrays.copyOf(T[] original, int newLength)
System.arraycopy(Object src, int srcPos, Object dest, int destPos, int length)
```

Classic problems:

--Two Pointers--

[1\) Rotate Array](#), [Reverse Words in a String](#)
[2\) Evaluate Reverse Polish Notation \(Stack\)](#)
[3\) Isomorphic Strings](#)
[4\) Word Ladder \(BFS\)](#), [Word Ladder II \(BFS\)](#)
[5\) Median of Two Sorted Arrays](#)
[5\) Kth Largest Element in an Array](#)
[6\) Wildcard Matching](#), [Regular Expression Matching](#)
[7\) Merge Intervals](#), [Insert Interval](#)
[9\) Two Sum](#), [Two Sum II](#), [Two Sum III](#), [3Sum](#), [4Sum](#)
[10\) 3Sum Closest](#)
[11\) String to Integer](#)
[12\) Merge Sorted Array](#)
[13\) Valid Parentheses](#)
[13\) Longest Valid Parentheses](#)
[14\) Implement strStr\(\)](#)
[15\) Minimum Size Subarray Sum](#)
[16\) Search Insert Position](#)
[17\) Longest Consecutive Sequence](#)
[18\) Valid Palindrome](#)
[19\) ZigZag Conversion](#)
[20\) Add Binary](#)
[21\) Length of Last Word](#)
[22\) Triangle](#)
24) Contains Duplicate: [I](#), [II](#), [III](#)
25) Remove Duplicates from Sorted Array: [I](#), [II](#), [Remove Element](#), [Move Zeroes](#)
[27\) Longest Substring Without Repeating Characters](#)
[28\) Longest Substring that contains 2 unique characters](#) [Google]
[28\) Substring with Concatenation of All Words](#)
[29\) Minimum Window Substring](#)
31) Find Minimum in Rotated Sorted Array: [I](#), [II](#)
32) Search in Rotated Array: [I](#), [II](#)
[33\) Min Stack](#)
34) Majority Element: [I](#), [II](#)
[35\) Bulls and Cows](#)
[36\) Largest Rectangle in Histogram](#)
[37\) Longest Common Prefix](#) [Google]
[38\) Largest Number](#)
[39\) Simplify Path](#)
[40\) Compare Version Numbers](#)
[41\) Gas Station](#)
44) Pascal's Triangle: [I](#), [II](#)

[45\) Container With Most Water](#)
[45\) Candy](#) [Google]
[45\) Trapping Rain Water](#)
[46\) Count and Say](#)
[47\) Search for a Range](#)
[48\) Basic Calculator](#), [Basic Calculator II](#)
[49\) Group Anagrams](#)
[50\) Shortest Palindrome](#)
[51\) Rectangle Area](#)
[52\) Summary Ranges](#)
[53\) Increasing Triplet Subsequence](#)
[54\) Get Target Using Number List And Arithmetic Operations](#)
[55\) Reverse Vowels of a String](#)
[56\) Flip Game](#), [Flip Game II](#)
[57\) Missing Number](#), [Find the duplicate number](#), [First Missing Positive](#)
[58\) Valid Anagram](#), [Group Shifted Strings](#)
[59\) Top K Frequent Elements](#)
[60\) Find Peak Element](#)
[61\) Word Pattern](#), [Word Pattern II](#)
[62\) H-Index](#) , [H-Index II](#)
[63\) Palindrome Pairs](#)
[64\) One Edit Distance](#)
[65\) Scramble String](#)
[66\) First Bad Version](#)
[67\) Integer to English Words](#)
[68\) Text Justification](#)
[69\) Remove Invalid Parentheses](#)
[70\) Intersection of Two Arrays](#), [Intersection of Two Arrays II](#)
[71\) Sliding Window Maximum](#), [Moving Average from Data Stream](#)
[72\) Guess Number Higher or Lower](#)

2. Matrix

Common methods to solve matrix related problem include DFS, BFS, dynamic programming, etc.

Classic Problems:

[1\) Set Matrix Zeroes](#)
[2\) Spiral Matrix](#)
[2\) Spiral Matrix II](#)
[3\) Search a 2D Matrix](#)
[3\) Search a 2D Matrix II](#)

[4\) Rotate Image](#) [Palantir]
[5\) Valid Sudoku](#)
[6\) Minimum Path Sum \(DP\)](#) [Google]
[7\) Unique Paths \(DP\)](#) [Google]
[7\) Unique Paths II \(DP\)](#)
[8\) Number of Islands \(DFS/BFS\)](#), [Number of Islands II](#) (Disjoint Set), [Number of Connected Components in an Undirected Graph](#)
[9\) Surrounded Regions \(BFS\)](#)
[10\) Maximal Rectangle](#)
[10\) Maximal Square](#)
[11\) Word Search \(DFS\)](#)
[11\) Word Search II](#)
[13\) Range Sum Query 2D – Immutable](#)
[14\) Longest Increasing Path in a Matrix \(DFS\)](#)
[15\) Shortest Distance from All Buildings](#)
[16\) Game of Life](#)
[17\) Paint House](#), [Paint House II](#)
[18\) Sudoku Solver \(DFS\)](#)
[19\) Walls and Gates \(DFS/BFS\)](#)
[20\) Tic-Tac-Toe](#)
[21\) Best Meeting Point](#)

3. Linked List

The implementation of a linked list is pretty simple in Java. Each node has a value and a link to next node.

```
class Node {  
    int val;  
    Node next;  
  
    Node(int x) {  
        val = x;  
        next = null;  
    }  
}
```

Two popular applications of linked list are stack and queue.

Stack

```
class Stack{
```

```

Node top;

public Node peek(){
    if(top != null){
        return top;
    }

    return null;
}

public Node pop(){
    if(top == null){
        return null;
    }else{
        Node temp = new Node(top.val);
        top = top.next;
        return temp;
    }
}

public void push(Node n){
    if(n != null){
        n.next = top;
        top = n;
    }
}
}

```

Queue

```

class Queue{
    Node first, last;

    public void enqueue(Node n){
        if(first == null){
            first = n;
            last = first;
        }else{
            last.next = n;
            last = n;
        }
    }

    public Node dequeue(){

```

```

        if(first == null){
            return null;
        }else{
            Node temp = new Node(first.val);
            first = first.next;
            return temp;
        }
    }
}

```

The Java standard library contains a class called "[Stack](#)". Another class from Java SDK is [LinkedList](#), which can be used as a Queue (add() and remove()). (LinkedList implements the Queue interface.) If a stack or queue is required to solve problems during your interview, they are ready to be used.

Classic Problems:

[0\) Implement a Stack Using an Array](#)

[1\) Add Two Numbers](#)

[2\) Reorder List](#)

[3\) Linked List Cycle](#)

[4\) Copy List with Random Pointer](#)

[5\) Merge Two Sorted Lists](#)

[6\) Odd Even Linked List](#)

[7\) Remove Duplicates from Sorted List](#)

[7\) Remove Duplicates from Sorted List II](#)

[8\) Partition List](#)

[9\) LRU Cache](#)

[10\) Intersection of Two Linked Lists](#)

[11\) Remove Linked List Elements](#)

[12\) Swap Nodes in Pairs](#)

[13\) Reverse Linked List](#), [Reverse Linked List II](#), Print Linked List in Reversed Order

[14\) Remove Nth Node From End of List \(Fast-Slow Pointers\)](#)

[15\) Implement Stack using Queues](#)

[15\) Implement Queue using Stacks](#)

[16\) Palindrome Linked List](#)

[17\) Implement a Queue using an Array](#)

[18\) Delete Node in a Linked List](#)

[19\) Reverse Nodes in k-Group](#)

4. Tree, Heap and Trie

A tree normally refers to a binary tree. Each node contains a left node and right node like the following:

```
class TreeNode{
    int value;
    TreeNode left;
    TreeNode right;
}
```

Here are some concepts related with trees:

1. *Binary Search Tree*: for all nodes, left children \leq current node \leq right children
2. *Balanced vs. Unbalanced*: In a balanced tree, the depth of the left and right subtrees of every node differ by 1 or less.
3. *Full Binary Tree*: every node other than the leaves has two children.
4. *Perfect Binary Tree*: a full binary tree in which all leaves are at the same depth or same level, and in which every parent has two children.
5. *Complete Binary Tree*: a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible

[Heap](#) is a specialized tree-based data structure that satisfies the heap property. The time complexity of its operations are important (e.g., find-min, delete-min, insert, etc). In Java, [PriorityQueue](#) is important to know.

4.1 Tree

- 1) Binary Tree Traversal: [Preorder](#), [Inorder](#), [Postorder](#), [Level Order](#), [Level Order II](#), [Vertical Order](#)
- 2) [Invert Binary Tree](#)
- 3) [Kth Smallest Element in a BST](#)
- 4) [Binary Tree Longest Consecutive Sequence](#)
- 5) [Validate Binary Search Tree](#)
- 6) [Flatten Binary Tree to Linked List](#)
- 7) [Path Sum \(DFS or BFS\)](#)
- 7) [Path Sum II \(DFS\)](#)
- 8) [Construct Binary Tree from Inorder and Postorder Traversal](#)
- 8) [Construct Binary Tree from Preorder and Inorder Traversal](#)
- 9) [Convert Sorted Array to Binary Search Tree](#) [Google]
- 10) [Convert Sorted List to Binary Search Tree](#) [Google]
- 11) [Minimum Depth of Binary Tree](#)
- 12) [Binary Tree Maximum Path Sum *](#)
- 13) [Balanced Binary Tree](#)
- 14) [Symmetric Tree](#)
- 15) [Binary Search Tree Iterator](#)
- 16) [Binary Tree Right Side View](#)
- 17) [Lowest Common Ancestor of a Binary Search Tree](#)
- 18) [Lowest Common Ancestor of a Binary Tree](#)
- 19) [Verify Preorder Serialization of a Binary Tree](#)
- 20) [Populating Next Right Pointers in Each Node](#)

[21\) Populating Next Right Pointers in Each Node II](#)
[21\) Unique Binary Search Trees \(DP\)](#)
[21\) Unique Binary Search Trees II \(DFS\)](#)
[22\) Sum Root to Leaf Numbers \(DFS\)](#)
[23\) Count Complete Tree Nodes](#)
[24\) Closest Binary Search Tree Value](#)
[25\) Binary Tree Paths](#)
[26\) Maximum Depth of Binary Tree](#)
[27 Recover Binary Search Tree](#)
[28\) Same Tree](#)
[29\) Serialize and Deserialize Binary Tree](#)
[30\) Inorder Successor in BST](#)
[31\) Find Leaves of Binary Tree](#)
[32\) Largest BST Subtree](#)

4.2 Heap

[1\) Merge k sorted arrays \[Google\]](#)
[2\) Merge k Sorted Lists *](#)
[3\) Find Median from Data Stream](#)
[4\) Meeting Rooms II, Meeting Rooms](#)
[5\) Range Addition](#)

4.3 Trie

[1\) Implement Trie \(Prefix Tree\)](#)
[2\) Add and Search Word - Data structure design \(DFS\)](#)

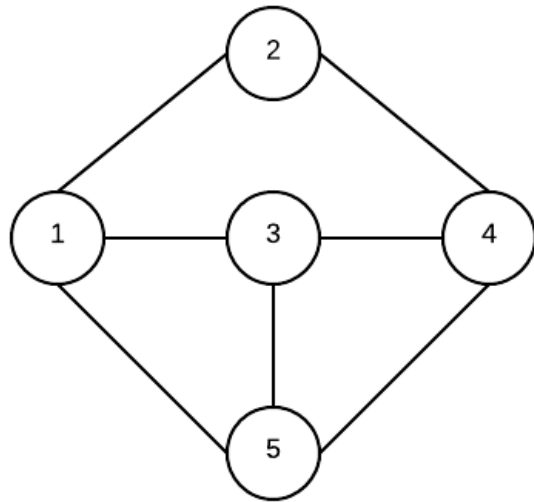
4.4 Segment Tree

[1\) Range Sum Query - Mutable](#)
[2\) The Skyline Problem](#)

5. Graph

Graph related questions mainly focus on depth first search and breath first search. Depth first search is straightforward, you can just loop through neighbors starting from the root node.

Below is a simple implementation of a graph and breath first search. The key is using a queue to store nodes.



1) Define a GraphNode

```
class GraphNode{
    int val;
    GraphNode next;
    GraphNode[] neighbors;
    boolean visited;

    GraphNode(int x) {
        val = x;
    }

    GraphNode(int x, GraphNode[] n){
        val = x;
        neighbors = n;
    }

    public String toString(){
        return "value: "+ this.val;
    }
}
```

2) Define a Queue

```

class Queue{
    GraphNode first, last;

    public void enqueue(GraphNode n){
        if(first == null){
            first = n;
            last = first;
        }else{
            last.next = n;
            last = n;
        }
    }

    public GraphNode dequeue(){
        if(first == null){
            return null;
        }else{
            GraphNode temp = new GraphNode(first.val, first.neighbors);
            first = first.next;
            return temp;
        }
    }
}

```

3) Breath First Search uses a Queue

```

public class GraphTest {

    public static void main(String[] args) {
        GraphNode n1 = new GraphNode(1);
        GraphNode n2 = new GraphNode(2);
        GraphNode n3 = new GraphNode(3);
        GraphNode n4 = new GraphNode(4);
        GraphNode n5 = new GraphNode(5);

        n1.neighbors = new GraphNode[] {n2, n3, n5};
        n2.neighbors = new GraphNode[] {n1, n4};
        n3.neighbors = new GraphNode[] {n1, n4, n5};
        n4.neighbors = new GraphNode[] {n2, n3, n5};
        n5.neighbors = new GraphNode[] {n1, n3, n4};

        breathFirstSearch(n1, 5);
    }
}

```

```
public static void breathFirstSearch(GraphNode root, int x){
    if(root.val == x)
        System.out.println("find in root");

    Queue queue = new Queue();
    root.visited = true;
    queue.enqueue(root);

    while(queue.first != null){
        GraphNode c = (GraphNode) queue.dequeue();
        for(GraphNode n: c.neighbors){

            if(!n.visited){
                System.out.print(n + " ");
                n.visited = true;
                if(n.val == x)
                    System.out.println("Find "+n);
                queue.enqueue(n);
            }
        }
    }
}
```

Output:

value: 2 value: 3 value: 5 Find value: 5
value: 4

Classic Problems:

- 1) [Clone Graph](#)
- 2) [Course Schedule](#) , [Course Schedule II](#) , [Minimum Height Trees](#)
- 3) [Reconstruct Itinerary](#)
- 4) [Graph Valid Tree](#)

6. Sorting

Time complexity of different sorting algorithms. You can go to wiki to see basic idea of them.

Algorithm	Average Time	Worst Time	Space
Bubble sort	n^2	n^2	1
Selection sort	n^2	n^2	1

Insertion sort	n^2	n^2	
Quick sort	$n \log(n)$	n^2	
Merge sort	$n \log(n)$	$n \log(n)$	depends

* BinSort, Radix Sort and CountSort use different set of assumptions than the rest, and so they are not "general" sorting methods. (Thanks to Fidel for pointing this out)

Here are some implementations/demos, and in addition, you may want to check out how [Java developers sort in practice](#).

- 1) [Mergesort](#)
- 2) [Quicksort](#)
- 3) [InsertionSort](#).
- 4) [Maximum Gap \(Bucket Sort\)](#)
- 5) [Sort Colors \(Counting Sort\)](#)

7. Dynamic Programming

Dynamic programming is a technique for solving problems with the following properties:

- 1. An instance is solved using the solutions for smaller instances.
- 2. The solution for a smaller instance might be needed multiple times.
- 3. The solutions to smaller instances are stored in a table, so that each smaller instance is solved only once.
- 4. Additional space is used to save time.

The problem of climbing steps perfectly fit those 4 properties. Therefore, it can be solve by using dynamic programming.

```
public static int[] A = new int[100];

public static int f3(int n) {
    if (n <= 2)
        A[n]= n;

    if(A[n] > 0)
        return A[n];
    else
        A[n] = f3(n-1) + f3(n-2);//store results so only calculate once!
    return A[n];
}
```

Classic problems:

- 1) [Edit Distance](#)

- 1) [Distinct Subsequences Total](#)
- 2) [Longest Palindromic Substring](#)
- 3) [Word Break](#)
- 3) [Word Break II](#)
- 4) [Maximum Subarray](#)
- 4) [Maximum Product Subarray](#)
- 5) [Palindrome Partitioning](#)
- 5) [Palindrome Partitioning II](#)
- 6) [House Robber](#) [Google]
- 6) [House Robber II](#)
- 6) [House Robber III](#)
- 7) [Jump Game](#)
- 7) [Jump Game II](#)
- 8) [Best Time to Buy and Sell Stock](#)
- 8) [Best Time to Buy and Sell Stock II](#)
- 8) [Best Time to Buy and Sell Stock III](#)
- 8) [Best Time to Buy and Sell Stock IV](#)
- 9) [Dungeon Game](#)
- 10) [Minimum Path Sum](#)
- 11) [Unique Paths](#)
- 12) [Decode Ways](#)
- 13) [Longest Common Subsequence](#)
- 14) [Longest Common Substring](#)
- 15) [Longest Increasing Subsequence](#)
- 16) [Coin Change](#)
- 17) [Perfect Squares](#)

8. Bit Manipulation

Bit operators:

OR ()	AND (&)	XOR (^)	Left Shift (<<)	Right Shift (>>)	Not (~)
1 0=1	1&0=0	1^0=1	0010<<2=1000	1100>>2=0011	~1=0

Get bit i for a give number n. (i count from 0 and starts from right)

```
public static boolean getBit(int num, int i){
    int result = num & (1<<i);

    if(result == 0){
```

```
        return false;
    }else{
        return true;
    }
}
```

For example, get second bit of number 10.

i=1, n=10

1<<1= 10 1010&10=10 10 is not 0, so return true;

Classic Problems:

[1\) Single Number](#)

[1\) Single Number II](#)

[2\) Maximum Binary Gap](#)

[3\) Number of 1 Bits](#)

[4\) Reverse Bits](#)

[5\) Repeated DNA Sequences](#)

[6\) Bitwise AND of Numbers Range](#)

[7\) Sum of Two Integers](#)

[8\) Counting Bits](#)

[9\) Maximum Product of Word Lengths](#)

[10\) Gray Code](#)

9. Combinations and Permutations

The difference between combination and permutation is whether order matters.

Example 1:

Given 5 numbers - 1, 2, 3, 4 and 5, print out different sequence of the 5 numbers. 4 can not be the third one, 3 and 5 can not be adjacent. How many different combinations?

Example 2:

Given 5 banana, 4 pear, and 3 apple, assuming one kind of fruit are the same, how many different combinations?

Class Problems:

[1\) Permutations](#)

[2\) Permutations II](#)

[3\) Permutation Sequence](#)

[4\) Generate Parentheses](#)

- [5\) Combination Sum \(DFS\), II \(DFS\), III \(DFS\), IV \(DP\)](#)
- [6\) Combinations \(DFS\)](#)
- [7\) Letter Combinations of a Phone Number \(DFS\)](#)
- [8\) Restore IP Addresses](#)
- [9\) Factor Combinations \(DFS\)](#)

10. Math

Solving math problems usually require us to find regularities or repeated pattern from the observations. List the results for a small set of numbers first, if you do not have any ideas.

- [1\) Reverse Integer](#)
- [2\) Palindrome Number](#)
- [3\) Pow\(x,n\), Power of Two, Power of Three, Power of Four](#)
- [4\) Subsets](#)
- [5\) Subsets II](#)
- [6\) Fraction to Recurring Decimal \[Google\]](#)
- [7\) Excel Sheet Column Number](#)
- [8\) Excel Sheet Column Title](#)
- [9\) Factorial Trailing Zeroes](#)
- [10\) Happy Number](#)
- [11\) Count Primes](#)
- [12\) Plus One](#)
- [13\) Divide Two Integers](#)
- [14\) Multiply Strings](#)
- [15\) Max Points on a Line](#)
- [16\) Product of Array Except Self](#)
- [17\) Integer Break](#)
- [18\) Add Digits](#)
- [21\) Ugly Number, Ugly Number II, Super Ugly Number, Find K Pairs with Smallest Sums](#)

UPDATE: I decided to add more categories below.

11. HashMap

- [1\) Shortest Word Distance II](#)

Additional Problems:

- [1\) Self Crossing](#)
- [2\) Patching Array](#)
- [3\) Nim Game](#)
- [4\) Bulb Switcher](#)

- 5) [Pain Fence](#)
- 6) [Nested List Weight Sum](#)

Additional Resources

- 1. [Share your code to Github/BitBucket](#)

Related Posts:

- 1. [LeetCode – Remove Nth Node From End of List \(Java\)](#)
- 2. [Find a Path in a Matrix](#)
- 3. [LeetCode – LRU Cache \(Java\)](#)
- 4. [LeetCode – Binary Search Tree Iterator \(Java\)](#)

Category >> [Algorithms](#) >> [Interview](#)

If you want someone to read your code, please put the code inside `<pre><code>` and `</code></pre>` tags. For example:

```
<pre><code>
String foo = "bar";
</code></pre>
```


111 Comments

Program Creek

 Login ▾


 Recommend 9  Share

Sort by Best ▾







Join the discussion...

LOG IN WITH


OR SIGN UP WITH DISQUS 

Name

- **Balaram Maharana** • 23 days ago
Please update PDF version to 7/2018
Its been long since PDF updated.
 |  • Reply • Share ›
- **Catherine puspita** • 8 months ago
I already read all the content in your website. It is really very helpful. I also appreciate it.


Thank you.

^ | v • Reply • Share ›

 **FirstGerman** • 10 months ago

I see you don't monetize your website, don't waste your traffic, you can earn extra cash every month because you've got hi quality content. If you want to know how to make extra money, search for: Mrdalekjd methods for \$\$\$

^ | v • Reply • Share ›

 **Prem K Chettri** • a year ago

Message to those who are bashing these problems by telling they got multiple years of experiences. Here is what I gotto say to you folks..


I have 7 of those highly technical, algorithmic developer roles but guess what any coder can solve them with different amount of time but here is the real thing. Any of you can't solve these complicated mind churning problem even if you are given plenty of time and thats called skill my friends.

Doing what you are doing is just a way of saying I don't care but I am pretty sure each one of us, true developer never stops learning. These problems are so challenging ones, that even if you could solve it, doesn't mean, you did it right way. They basic understanding of proper usage of algorithms not only helps create scalable and efficient codes but also helps to segregate best candidates from average ones.

So by saying, these won't help doesnt make sense, and IMO, sometimes, I do needs these knowledge in my algorithmic design in my work place too. Thus, saying these skills are useless way to test or not required in real time environment is just like saying, you dunno if there were any such help you could have used, if you were aware of it.


So, stop crying and improve yourself.

^ | v • Reply • Share ›

 **vmasoft** • a year ago

I usually ask during coding interviews a question about recursion. Something like finding the maximum number in a jagged array: <http://www.codeavenger.com/...>

^ | v • Reply • Share ›

 **Chris** • a year ago

Take a look into fresh set of Java interview questions. It could be helpful both for programmers and team-leads as kind of preparing for interview.

^ | v • Reply • Share ›

 **Bhuwan Tripathi** • a year ago

I do throw your whole web site and i wanna appreciate your efforts. You have brilliantly done the job. hats off to you man. Awesome website and fabulous content. Very nice good



I go through your whole web site and I really appreciate your content. You have definitely done the job, take on to your main / welcome website and add more content. Very nice good job.

Thanks

^ | v • Reply • Share ›



Top Coder • a year ago

Another great list -
<http://www.techiedelight.co...>

^ | v • Reply • Share ›



Sharvin Shaji • a year ago

PLEASE ANSWER
interview question

One mismatch
Max. Marks: 100
Given a string
s
s and a set of
n
n strings
p
i
pi.

For each
i
i find the number of occurrences of
p
:

[see more](#)

^ | v • Reply • Share ›



Raj • 2 years ago

I think this is a great list of problems to practice for coding interviews.

It would be nice if the admin could rearrange the leetcode questions in the list into (Easy/Medium/Hard) based on the difficulty level given on leetcode.

Thanks!

^ | v • Reply • Share ›



Hogan Freeman • 2 years ago

The new Leetcode has more than 400 problems. When would this website get updated ? It would be really nice to get the synced version.

^ | v • Reply • Share ›



Róbert Papp ➔ Hogan Freeman • 2 years ago

You can find solutions for each of those problems in their corresponding discussion forums.

^ | v • Reply • Share ›



Prathiksha Gowda • 2 years ago

If you guys think this is not the best website for the interview questions. Can you please suggest a website where we have good questions? Since I am new to programming and desperately looking for a job in US suggest me where I can look for interview questions and prepare well.

^ | v • Reply • Share ›



Bhaskar Reddy ➔ Prathiksha Gowda • a month ago

This is a pretty good list, but from my experience, I would say Leetcode + Practice on Paper

^ | v • Reply • Share ›



Cherry Zhao • 2 years ago

This is a great post. I like that it includes almost all the common topics asked in coding interviews. I also like to share this blog about coding interview questions - <http://blog.gainlo.co/index...>

^ | v • Reply • Share ›



Anjali • 2 years ago

Programming Problem List

<http://www.binarycoder.org/...>

^ | v • Reply • Share ›



3D • 2 years ago

Could you please update the PDF version to the latest 2016. 03. 19. as well?



Thank you!

^ | v • Reply • Share ›



ryanlr → 3D • 2 years ago

PDF updated now. Thanks!

^ | v • Reply • Share ›



3D → ryanlr • 2 years ago

Thank you!

^ | v • Reply • Share ›



Maya • 2 years ago

Thanks for your solution. There are lots of new question in leetcode now, please update.

^ | v • Reply • Share ›



Sergey Muravyov • 2 years ago

THX a lot for this post, really useful, especially for closed issues from leet!

^ | v • Reply • Share ›



Shivam Singh • 3 years ago

Nice algorithms.... visit more Coding interview questions

^ | v • Reply • Share ›



Ashwin • 3 years ago

Appreciate your effort getting the algorithms , its fun to ponder over the questions and try solving it, weather of or not to be asked in interview , I like to go through the subjects u have shared.

^ | v • Reply • Share ›



Mamta Ajay Rai • 3 years ago

Good Article .Good point 'Joe Pepersack' agreed.

^ | v • Reply • Share ›



java_an • 3 years ago



visit string program for interview

^ | v • Reply • Share ›



Rajneesh Chaturvedi • 3 years ago

can any one tell me from following code that last inner loop should run infinitely but it is running fine why :

```
class Pyramid{
public static void main(String args[])
{
int i=0,j=0,k=0,l=0;
for(i=1;ii;j--)
{
System.out.print(" ");
}
for(k=i;k>=1;k--)
{
System.out.print(""+k);
}
for(l=2;l>1;l++) (it should run infinitely but it is running fine )
{

if(l<=i)
System.out.print(""+l);

}
System.out.println("");

}

}

}
```

^ | v • Reply • Share ›



Pierre Wang → Rajneesh Chaturvedi • 3 years ago

when l = Integer.MAX_VALUE, then l + 1 = 0

^ | v • Reply • Share ›

^ | v • Reply • Share ›



Runiko • 3 years ago

how can develop a software that will easily be convert the algorithm into java?

^ | v • Reply • Share ›



scvblwxq • 3 years ago

What about SQL and GUI programming?

^ | v • Reply • Share ›



Youchen • 4 years ago

Great to see this post, Thanks a lot!

^ | v • Reply • Share ›



Özge başak • 4 years ago

Thanks for the great article. i really appreciate it. I have been making research on the internet for a very long time and now come up with something useful. it will be a great guide for my thesis here:

<http://www.incehesap.com/ga...>

^ | v • Reply • Share ›



Henry • 4 years ago

GREAT work! Incredible useful for a quick review before an interview =)

^ | v • Reply • Share ›



RA the Guest ➔ Henry • 4 years ago

Did you get the job?

^ | v • Reply • Share ›



atlas • 4 years ago

I am a Java learner, and i like your articles very much. So,can i reprint your some wonderful articles to my blog(<http://zhangxiaolong.org/>)? Sometimes, i need to translate some articles to Chinese,and someone can understand contents os article.

Hope to get your permit.

Have a good day!

^ | v • Reply • Share ›



ryanlr → atlas • 4 years ago

Sure, you are welcome to do that, but remember to give me a back link:)

^ | v • Reply • Share ›



ISuckatProgramming • 4 years ago

Having done more than 20 or more tech interviews during a job search recently, I've found the lists in this post to be very useful. Most companies like Google or LinkedIn will first give you a phone screen and there you will encounter 2 or 3 questions of the variety found here. Once you survive that and get to the in person interview, then that's where the REAL fun begins. You typically get much harder programming problems and have much less time to finish them. I guess most developers at Google or Facebook must be geniuses or something, I guess this is a good way to weed out all the people of only average intelligence. :(

^ | v • Reply • Share ›



James • 4 years ago

http://www.amazon.com/b?_en...

^ | v • Reply • Share ›



Aarohi Shirke • 4 years ago

For more IT interview questions you can check with skillgun @ <http://skillgun.com> truly helpful for your preparation

^ | v • Reply • Share ›



Drew • 4 years ago

I find it funny that there is so much bashing about how these problems don't really tell the interviewer anything other than being able to jump through some hoops. I completely disagree with those sentiments. Having problems such as these while trivial for some, can quickly weed out those who don't have fundamental understanding of basic data structure concepts or classic algorithms. Sure, there are libraries for such things to hide away the gory details, but if you blindly just use them without knowing when they apply, you get into bad habits. These classical problems also gives a common baseline in which to judge applicants. It's not a perfect system, nothing is, but it's unrealistic for employers to devise more relevant questions because in any new job, there is going to be a learning curve no matter how much someone prepares.

Another thing to consider, these are computer science type of questions, applicable to a coder. Software Engineering is much more than just algorithms and encompasses the entire process of writing good software. I don't believe this site advertises those type of questions because they can be much more subjective and open for great debate.

One more thing. As someone who has programmed embedded systems, I'm not always afforded the luxury of using libraries for various business/legal reasons. Without the crutch of someone already doing the work for you, one must be knowledgeable enough to use classic algorithms and shape it to meet the necessary solution.

^ | v • Reply • Share ›



Walt Corev → Drew • 4 years ago



4 years ago

Knowing how they apply is a vastly different problem than how to implement them. As a previous poster wrote about never using a graph or never having to implement a RB tree that's different than having an understanding of why they are important. It's a 5 minute answer verses a 40 minute exercise.

^ | v · Reply · Share ›



Jitendra Sangar · 4 years ago

A collection of problems asked in interviews

<http://algorithmsandme.blog...>

^ | v · Reply · Share ›



John · 4 years ago

In an embedded field is ok to ask about these topics, as reinventing the wheel in embedded C is pretty usual, but for higher level applications these kind of questions don't give any good qualifiers, apart from theoretical ones. E.g. would be a lot better if newbie C++ developers knew how to properly use containers and algorithms than knowing how to implement by hand a double linked list or a tree. A good example is this talk: <http://channel9.msdn.com/Ev...>

^ | v · Reply · Share ›



Johan Stén · 4 years ago

Decades of programming experience here as well, highly technical at that. Low-level, technical, algorithmical stuff. I've never had any use for 99% of all the stuff that comes up in so called "competitive programming". At best, this is masturbation. It's not programming, it's not what programming is about, it's not what programmers spend their time on. It's jerking off.

I took part in a programming competition where the qualifying round was something akin to this. The final round however was a "real life" programing task, where you had spend hours, to ship code that actually did something useful. None of the "competitive programmer" types ended up anywhere near the top.

Do I enjoy it? Highly. Does it have anything to do with what I get paid for? No.

^ | v · Reply · Share ›



kazoompa · 4 years ago

I am 100% with Joe. I've also done two decades of programming and never found most of the above questions too relevant. These days most people Google for the API's or algorithms, you can't know them all by heart or better still, some languages incorporate them in their grammar. Problem solving and the choosing the proper strategy is a lot more important.

^ | v · Reply · Share ›



Gabriel Ramírez · 4 years ago

great post! really

^ | v · Reply · Share ›

^ | v • Reply • Share ›



Le Trung Kien • 4 years ago

I don't know why some people keep complaining about how big companies conduct interviews, as if they know the optimal way. It is very similar to students who do not contend with the way professors give exams.

^ | v • Reply • Share ›



Walt Corey ➔ Le Trung Kien • 4 years ago

Not at all, those two scenarios are nothing alike. A student has absolutely no standing to question the professor. Professors generally have 90 semester hours in the subject the student maybe has 3 in. An interviewee with 20 years experience generally has 10 years or more experience than the person interviewing them.

^ | v • Reply • Share ›



Johan Stén ➔ Le Trung Kien • 4 years ago

"Q. Other insights from the studies you've already done?

A. On the hiring side, we found that brainteasers are a complete waste of time. How many golf balls can you fit into an airplane? How many gas stations in Manhattan? A complete waste of time. They don't predict anything. They serve primarily to make the interviewer feel smart."

<http://techcrunch.com/2013/...>

^ | v • Reply • Share ›



Joe Peppersack • 5 years ago

These are horrible - if typical - interview questions. Asking horrible programming questions will get you horrible programmers.

I've been developing software professionally for 25 years. If someone asked me how I would implement a linked list, my answer would be "I wouldn't. I'd a) develop in a language which supports lists natively or b) use a library.

If your interview tests people on their ability to reinvent wheels, you're going to get programmers who are good at reinventing wheels. You want your interview questions to test a candidate's ability to solve problems and make smart design decisions. Reinventing the wheel is almost never a smart design decision.

^ | v • Reply • Share ›



Stephen Boesch ➔ Joe Peppersack • a year ago


Do you really believe your own comment here? Oh.. apparently you do .. I could not agree much less. The areas in which I can build something from scratch OR from existing libraries are the areas counted as strengths. Knowing the libraries also means understanding what the limitations and optimal use cases are - which typically means referencing the source code - which is easier to do when you know the general outline of how to implement it yourself. Instead it is easy to stay "at a certain level" above the fray - and have no idea how to get into the guts if required. And yes I have worked for (and interviewed for..) Google, Apple, IBM Research, Nokia, ..

Load more comments

ALSO ON PROGRAM CREEK

LeetCode – Remove Duplicates from Sorted List II (Java)


4 comments • 6 months ago



Swati Mewara — public ListNode removeDuplicate(ListNode head){ var previous = head; var p = head.next; var newHead = null; while (p != null ...

LeetCode – Paint House II (Java)


1 comment • 6 months ago



Omender Sharma — Providing o(n^3) solution here but yea for understanding purpose i think it will be good.public class Solution { /** * @param costs n x k ...

FAQ about Web Services and Related Technologies


1 comment • 6 months ago



essay reviews live — We get now all types helps from online now. This is huge collection of solution and i know any kind of problems we can solution in a short time. ...

Java Design Pattern: Flyweight

4 comments • 6 months ago



Tristan Yan — Not only factory should be singleton, but also getCoffeeFlavor is not thread-safe

