

- [Program Creek](#)
 - [R](#)
 - [Research](#)
 - [Machine Learning](#)
 - [Contact](#)
- [Simple Java](#)
- [Java 8](#)
- [Coding Interview](#)
- [Java](#)
 - [Java Basics](#)
 - [Java Object Oriented Concepts](#)
 - [Java Collections & Generics](#)
 - [Java File I/O](#)
 - [Java Database](#)
 - [Java Multi-Threading](#)
 - [Java XML Parsing](#)
 - [Advanced Topics](#)
- [DP Stories](#)
- [Java Examples](#)
- [Frameworks](#)

- [Python Examples](#)

Top 10 Algorithms for Coding Interview

PDF: [Update History](#), [Latest version \(8/1/2016\)](#)

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 \(DFS\)](#)

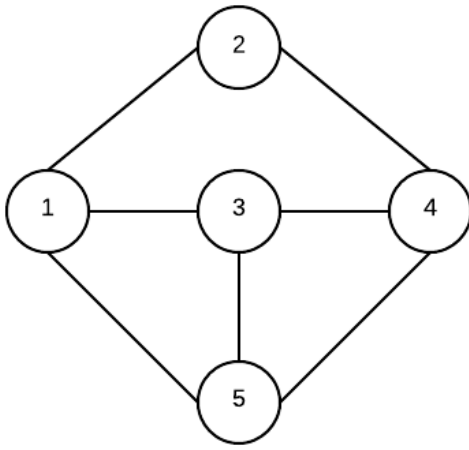
4.4 Segment Tree

- [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\) \$\text{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](#), [9Ugly 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](#)






Fitbit - Flex 2 Activity Tracker - Black

\$99⁹⁵



Shop Now

©2016 Best Buy

You May Also Like ...

1. [How to answer coding questions for your interview?](#)
2. [面试10大算法汇总+常见题目解答](#)
3. [LeetCode – Word Ladder II \(Java\)](#)
4. [LeetCode – LRU Cache \(Java\)](#)



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>
```

80 Comments Program Creek

Login ▾

Recommend 44 Share

Sort by Best ▾



Join the discussion...



Malleus Veritas • 3 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.

107 ^ | v • Reply • Share ›



ryanlr Mod → Malleus Veritas • 3 years ago

Large companies (e.g., Google, Facebook, etc) test developers' knowledge of algorithm and data structure. That's the essence of coding interview.

28 ^ | v • Reply • Share ›



Malleus Veritas → ryanlr • 3 years ago

I've interviewed with industry-leading companies.

I've worked for industry-leading companies.

I've interviewed candidates for industry-leading companies.

The top companies want candidates who can SOLVE PROBLEMS EFFICIENTLY. They could care less if you know how to implement a breadth-first search from memory, because that's a solved problem. Even if you don't have a suitable library for your platform, the expectation for a senior developer is that you can pick up a reference and implement it.

What they DO care about is if you know how to select the correct data structures and algorithms to solve a problem, and do so efficiently and professionally - which means means writing legible, maintainable code that follows best practices like code reuse.

Re-implementing basic library functions would be a GUARANTEED way to fail an interview for a top company. If I give you an interview problem that requires you to count the number of distinct values in an input set, I'm testing you on your ability to recognize that you need to use a hash table, your ability to justify that design decision, and your ability to tell me how it's going to scale... not on your ability to come up with an ad-hoc implementation of a hash table. Rolling your own hash table would be just as much of a fail as using the wrong data structure.

Reinventing the wheel is bad engineering. If you hire engineers based on their ability to reinvent wheels, you are by definition hiring bad engineers.

49 ^ | v • Reply • Share ›



ryanlr Mod → Malleus Veritas • 3 years ago

Those questions are indeed asked during interviews.

10 ^ | v • Reply • Share ›



Malleus Veritas → ryanlr • 3 years ago

Yes, the posted questions are VERY typical of the questions that get asked by clueless interviewers at second-rate organizations.

They ask questions like this and then they wonder why they hire programmers who write slow, buggy, and unmaintainable code. They scratch their heads why their products are delivered late, are unreliable, and won't scale. They wonder why the top talent turns down their offers.

This article is a shining example of how NOT to interview programmers and what questions NOT to ask. In all honesty, I'd walk out of an interview that asked these kind of questions, because it's obvious to me that they have no clue what they're doing. It speaks volumes about their corporate culture, and none of it is positive.

25 ^ | v • Reply • Share ›



Wei Qiu → Malleus Veritas • 3 years ago

For me, knowing about how to reinvent stuff is the only way to really understand how stuff works.

These are the basics. Knowledge of them doesn't make people awful programmers. Lacking knowledge of engineering does.

33 ^ | v • Reply • Share ›



Vinz → Malleus Veritas • 2 years ago

Hi Joe,

I am a Grad Student and preparing for the interviews..

I completely agree with you, however in my past experience when i told the interviewer the similar answer just just posted ! he glared at me and then i got rejected :(

11 ^ | v • Reply • Share ›



abossard → Malleus Veritas • 2 years ago

I basically agree, it doesn't make sense in most companies, to reinvent the square wheel all the time. I had an interview with a startup that asked my how to implement a Hash-table in the end they didn't know anything about my problem solving skills and ended up explaining a Hash-table to me. Still they were interested in me taking the job.

But for companies which are driven by algorithms, like Google or Facebook, it does make sense, to check whether the applicant know these things. Not to implement it by heart, but because the questions they asked are related to such algorithms. They don't ask to implement a linked list, but they may asked how to detect a cycle in a linked list, or the start of the cycle. They want to see whether you can use the basic knowledge about algorithms to solve problems.

To know the algorithms by heart is one part, but to apply that knowledge to solve trick-questions, that's what they want to see.

In the end, I know I'm not a good or bad programmer just by this knowledge, but I want that Job at Google :-)

6 ^ | v • Reply • Share ›



Kenneth ➔ abossard • 2 years ago

Yes Google and Facebook do ask these questions, but not for the reason you believed.

I have friends in G and FB. They told me that they spent <1% of their time solving problems like "detecting cycle in linkedlist". It doesn't seem to be a wise thing to ask 90% of such questions interviews.

The reason why they asked these questions was that they did know a better way to objectively evaluate candidate's real design and coding skills. So they fell back to the questions they were asked at school.

5 ^ | v • Reply • Share ›



WJ ➔ Malleus Veritas • 2 years ago

You are right. Top Tech companies won't ask these question from you (definitely, if YOU will apply there, a person will 25 years of experience). If they do ask these question, yeah!! You should probably walk out.....Actually, these questions are asked by TOP Tech companies for Software Engineer roles (Fresh/mid/senior) under 10-15 year experience. Because they really want people who knows how to invent wheel. And most of the time developers will have to invent a wheel in such companies. A person with 25 years of experience shouldn't apply for SDE roles. If you do....I would wonder....why you would. I think..... you better go for lead/architect positions. And then ...yes!!! you won't be asked such questions. Instead, companies will emphasis on design, your past experience plus your achievements, blogs, patents, your personal references, your links and your presence in IT industry. That's what company like Google look for. I hope, you get it now!!

2 ^ | v • Reply • Share ›



humfff humfff gnarl gnarl . . ➔ Malleus Veritas • a year ago

Joe for president. He knows everything. i will vote you Dude. You seem so relaxed and cool

1 ^ | v • Reply • Share ›



hardsoft ➔ Malleus Veritas • 2 years ago

When interviewing individuals for an embedded position, I might ask them to write a swap function because it is a simple and quick way to gauge their understanding of pointer operations and can lead into good conversation.

An embedded programmer needs to have a rock solid foundation in C or will have a steep learning curve.

So technical questions can be useful to judge a candidates knowledge. If you walked out, I

would (probably rightly) assume you are a primadonna who would end up writing slow, buggy, and unmaintainable code because you would not have the fundamentals to perform your job.

1 ^ | v • Reply • Share ›



Walt Corey → Malleus Veritas • 2 years ago

Joe, you touched on the answer. These questions are asked by people who, otherwise don't know how to interview a candidate. I'd go so far as to say it is a gotcha. That may be a tad unfair, it depends on if the answer is relevant to the job or if it is from the mind of someone who doesn't know how to interview. If a prospective coworker has 30 minutes with a candidate, give them a test and spend the next 25 minutes sitting there politely with your hands folded I was also interviewed at a startup and one question was what is the second argument to a hashmap constructor. I answered I was pretty sure it was capacity but as I use a smart IDE I let the IDE prompt me for the parameters. I focus on solving the business problem not memorizing parameter sequences.

I think for a young or otherwise junior level position it may be important they know. It's akin to a MCS being way more important if you are 25 with 0 years of experience than 45 with 22 years of experience.

^ | v • Reply • Share ›



chen gao → Malleus Veritas • 3 years ago

I totally agree with you, but the problem is that if you only offer some questions about your company's products or software, it's really hard for them who have never done an internship in this company or never used these software. Maybe they only use these algorithm problems only to test whether this guy is qualified after a short training in the company.

^ | v • Reply • Share ›



serge → Malleus Veritas • 2 years ago

Design patterns are also solved problems. Do you want to hire someone who doesn't know about them?

4 ^ | v • Reply • Share ›



Mike → serge • 8 months ago

Good point!

^ | v • Reply • Share ›



Anshul Goel → Malleus Veritas • 4 months ago

Perfect

^ | v • Reply • Share ›



DC → Malleus Veritas • 2 years ago

I was asked a couple of these questions on an Amazon interview about a month ago. The interview went fairly well, although I didn't get the job. My guess is they ask the questions to get a feel of how you analyze a problem, and your way of coming up with solutions. This also gives a company an idea of your underlying knowledge of a language, and time complexity.

I'm not saying its right or wrong, but I guess it works for the companies who use this method of interviewing.

^ | v • Reply • Share ›



Oscar M → Malleus Veritas • 2 years ago

Although I do agree reinventing the wheel will get you nowhere and actually no one expects you to do it on the job, regardless I believe this is still a valid interview question to me. Why? Because even you said you want a candidate which has great problem solving skills and makes smart design decisions.

And this is nothing more than a problem which needs to be resolved, and by asking this you are looking for someone that can invent a wheel when needed.

Yes I agree that better questions could be asked, for example something like these Java programming tests, they are focused on examining the candidate's problem solving skills.

But nevertheless I see it like this, at some point this was a problem and then someone resolved it and made an API out of it. Linked list structure is commonly known so we could expect from the candidate that he knows what it is, and so what you are actually asking is to see how he designs some structure (even an existing structure in this case), which at some point he certainly will do on the job.

^ | v • Reply • Share ›



mary joe → ryanlr • 6 months ago

Thx OP for compiling a great list of interview questions! They are very commonly seen in those big 4 compaines.

^ | v • Reply • Share ›



Walt Corey → ryanlr • 2 years ago

Apparently Facebook also tests developers ability to program while getting intoxicated, see "Social Network". I agree 100% with Joe. I was once asked, either by Google or Amazon how'd I'd implement a stack in Java. I said I would instanciate an instance of the java Stack() class and then use it as a last in/first out container, I didn't get the job.

^ | v • Reply • Share ›



Róbert Papp → Walt Corey • 4 months ago

Hmm, I know it's not the point here, but I wouldn't accept Stack as an answer: it's synchronised and based on Vector. This may have been the answer way back more than 10 years ago, today a better answer may be: use ArrayDeque or LinkedList instance though the Deque interface, but one should be able to whip one up from scratch on request, because it's so simple.

^ | v • Reply • Share ›



Walt Corey → Róbert Papp • 4 months ago

Perhaps but that wasn't the question Bob. I completely disagree with your conclusion. Too many professional developers forget they are paid to write software, generally for sale, not rewrite the language in their own image.

This is a derivative of,

- 1) get it working.
- 2) get it working correctly.
- 3) If, and only if, there is a performance problem in that piece of code, optimize the implementation.

If they wanted to know how to implement a DeQueue that should/would have been the question.

The meta answer I gave was I do not reinvent the language unless I absolutely need to. And that was two years ago....c'mon dude. Actually, when that interview took place was in 2006, my comment was 2 years ago.

^ | v • Reply • Share ›



Victor Maslov aka Nakilon → Malleus Veritas • 2 years ago

Looks like you are making horrible bad code of totally boring and useless software for 25 years, because you didn't even realize, that these questions aren't about data structures implementation but about algorithms of work with them.

13 ^ | v • Reply • Share ›



Paul → Malleus Veritas • 2 years ago

Maybe that's why you don't work for a fortune 500 company.

3 ^ | v • Reply • Share ›



Guest → Paul • 2 years ago

No, they just hire me as a consultant at \$200/hr.

7 ^ | v • Reply • Share ›



Guest → Malleus Veritas • 2 years ago

In top software companies (google, fb, amazon, ms, etcetera) you will face this kind of questions because what they want to see it's not only if you can solve the problem but HOW did you solve it. That's exactly what they want to evaluate, they want to see what questions you asked to understand the problem, how did you analyze it, how you decomposed the problem, how you transformed your analysis and ideas into code, how good is your code (readable, maintainable, flexible). So the questions are not all about algorithms but they are also about your skills

2 ^ | v • Reply • Share ›



mary joe → Malleus Veritas • 6 months ago

You are a very ignorant yet highly arrogant person and I cant believe there are so many idiots give u plus ones i bet there are many losers out there who got rejected by google or facebook. I have checked your resume and the company that you worked for is nothing impressive. Yes it is a bad practice to reinvent the wheels in real life, but does it necessarily mean it a bad pratice to ask those questions in interview? Why are the universities all over the world teach calculus, linear algebra, differential equations? Will those subjects really be used in real life, by your logic by bother learning all of those, why dont the universtisy dump all the theoretical courses anyway. I am pretty sure that you are not more than an average programmer, just becauser you are not good at it and you cant work for google or mircosoft doesnt mean those questions are horrible. I just cant stand incompetent people bullshiting here and pretend to be pro.

1 ^ | v • Reply • Share ›



Malleus Veritas → mary joe • 6 months ago

The criticism of someone who cannot use proper grammar, punctuation, and capitalization is so stinging. Consider me rebuked by your rapier-keen insight.

The unskilled often mistake competence and confidence for arrogance because they lack the skill to tell the difference.

Bottom line is that if you want to hire the right people, you need to ask the right questions. Questions that test the candidate's ability to regurgitate facts they learned by rote are not the right questions. Questions that demonstrate the candidate's ability to apply that knowledge are.

^ | v • Reply • Share ›



minus Seven → Malleus Veritas • 2 years ago

This is true. But most companies don't want to spend so time on evaluating candidates perfectly. These are just ways to weed out as many candidates as possible so that they can reduce the number of choices of potential hires.

Still most companies insist on them and as an interview candidate it is important to know them enough to be able to solve them. Interviews are hardly perfect these days.

1 ^ | v • Reply • Share ›



Om Shankar → Malleus Veritas • 4 months ago

Can you give any example of your statement? How many horrible programmers you know who ended up being in Google, Netflix, etc. - they all take puzzle, algorithms, and such interviews. By now, Google, Facebook, Netflix, LinkedIn, Apple, etc. have the best programmers. Hence they are the best companies.

For your solution, "I'd develop in a language which supports lists natively" - you take for granted that the language exists, and it can exec a Linked List. Whoa!! Someone intelligent has to create that first, right? for it to exist, for you to then use it!

Referring your comment below, industry leading companies - might not be the companies who create things. For eg., Accenture/ServiceNow, etc. are industry leading companies for the "Service Industry". How many patents have they filed?

Compare them to patents filed by Google, Facebook, LinkedIn, etc.

Industry leading companies usually want to "pick up the best reference and implement it" - like you said. But who creates those as references?

Someone should create a masterpiece, to be chosen and used by others.

Companies like Google, Facebook, etc. "create" things. And that's why they are fair in taking such interview.

Experience of XYZ yrs, is an important thing, very important. But "raw intelligence" is also very important. Both go hand in hand. Companies that are shaping the technology around us, want to be sure that they are hiring intelligent people. So that, when some entirely new problem has to be solved (which can't be solved with previous experience), it can be done.

^ | v • Reply • Share ›



Java Developer → Malleus Veritas • 4 months ago

Unfortunately, these questions are being asked to Fresh graduates and Seniors alike. Last few months I attended 10+ interviews. All started with a phone interview on laptop with naive editor where they expect you to even remember all syntax. Interviewers are expecting you to solve these types of questions in 30 minutes with best possible algorithm. In real world, best algorithms evolve.

You can clear these interviews only if you have exclusively prepared for these companies and type of questions. Once interview is cleared, all these companies are same old crap.

Google, Facebook, linkedin, amazon, ebay - all of them do it same way.

Even worse - startups are giving coding project and round zero screening than do next rounds as coding. Waste almost 1 full day in interview and reject most people wasting everyone's time.

Here is a good read from Jeff Atwood - <http://blog.codinghorror.com/w...>

^ | v • Reply • Share ›



airjordan919 → Malleus Veritas • 5 months ago



As I see, a lot of these questions are actually not asking you to reinvent wheels but testing your ability to design efficient and simple algorithms as well as using appropriate data structures.

^ | v · Reply · Share ›



Raul Astudillo → Malleus Veritas · a year ago

I've recently arrived to the US and I'm having this huge struggle with interviews for the same thing. I had three interviews where recruiters liked my profile to be later rejected by indian technical reviewers because I can't implement over the phone a hashmap.

I'm really tired to be treated like a google search client over the phone for things that can be easily solved by a 10 second search on google.

I have 6 years working as a programmer and 4 of those years working at IBM Chile so I think I can do just fine without knowing letter by letter those algorithms (and I haven't even used any of those because frameworks already implement them).

^ | v · Reply · Share ›



Tom Bombadil → Malleus Veritas · a year ago

Right. And we should remove basic match from schools since we have calculators now.

^ | v · Reply · Share ›



Malleus Veritas → Tom Bombadil · a year ago

Since you can't even spell "math" correctly, I see no reason to even bother addressing the idiocy and irrelevance of your comment.

4 ^ | v · Reply · Share ›



a → Malleus Veritas · a year ago

irrelevance*

4 ^ | v · Reply · Share ›



Keith Moon → Malleus Veritas · 6 months ago

Irrelevance, you mean. I agree with your basic premise about the uselessness of implementing a linked list, but your arrogance is grating. If you are going to make the argument that you aren't going to listen to the arguments made by someone who cannot spell properly, or use proper punctuation, grammar, and capitalization, it is ironic that you can't spell either. Or maybe it was just a typo, like the "match" in question.

"I've interviewed with industry-leading companies.

I've worked for industry-leading companies.

I've interviewed candidates for industry-leading companies."

I am not sure if you are trolling or not, but in the last 6 months I have been asked multiple questions from this list by Google, Microsoft, and Amazon. To give them credit, they didn't care if I recalled the correct method names for STL or Collections classes and methods, but the questions were from this list. Regardless of how useful these problems are in real life (they are not that useful mostly), they are useful just for the purpose of interviews.

1 ^ | v · Reply · Share ›



Guest → Malleus Veritas · 2 years ago

Well said.

^ | v · Reply · Share ›

**Johan Stén** • 2 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" programming task, where you had to 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.

25 ^ | v • Reply • Share ›

**Drew** • 2 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 give 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.

4 ^ | v • Reply • Share ›

**Walt Corey** → Drew • 2 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 versus a 40 minute exercise.

^ | v • Reply • Share ›

**Mambo** • 3 years ago

So after asking all that stuff: how often in your professional career did you have to implement a linked list? Or graphs? Once? Twice? Nice things to ask, but not too practically relevant IMHO.

Recursion can be useful, but what you don't mention and don't ask: how expensive is recursion? I've seen more than enough people coming from university trying to solve everything with recursion - resulting in bad to read, slow and memory expensive code. Every recursion creates a copy of the recursive function in memory (in most languages). And recursion almost always is slower than the iterative solution.

4 ^ | v • Reply • Share ›

**Nathaniel Talcott** → Mambo • a year ago

"I've seen more than enough people coming from university trying to solve everything with recursion - resulting in bad to read, slow and memory expensive code."

Yeah, what do those University assholes know anyway?

"Every recursion creates a copy of the recursive function in memory(in most languages). And recursion almost always is slower than the iterative solution."

This statement is incorrect. When you do tail recursion there is no reason to create a new stack frame.

When you say that "most" languages can't do [tail recursion] you're wrong. C and C++ optimize for tail recursion and, in most non-contrived cases, will execute a properly implemented recursive function faster and with less memory than a comparable iterative function.

Java and C# do not have the optimization for tail recursion because they're being compiled just in time (JIT)... the JIT process doesn't allow time to perform the tail recursion optimization this feature was never implemented in these languages.

There is a reason it is a bad idea to use Java for real-time or embedded applications. Not only does the JIT process prevent optimization... but in every instance where the Java creators had to choose between performance and safety (by which I mean preventing the developer from making stupid mistakes) the Java creators always erred on the side of preventing stupid mistakes.

2 ^ | v • Reply • Share ›



Wei Qiu → Mambo • 3 years ago

Recursion is not hard to read. Recursion frequently leads to elegant and compact solutions. Check Sedgewick's implementation of Left Leaning RB tree as an example. For me recursion is the most natural way to tackle hard problems. Recursion is generally more powerful than iteration. Iteration is just an extra optimization step when you get the solution right.

2 ^ | v • Reply • Share ›



Walt Corey → Wei Qiu • 2 years ago

It absolutely can be but it can also blow a stack wide open. The goal is to have software that doesn't program check at the customer's site, not elegate to the eyes of the developer.

1 ^ | v • Reply • Share ›



Le Trung Kien • 3 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.

4 ^ | v • Reply • Share ›



Johan Stén → Le Trung Kien • 2 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/2013/06/...>

2 ^ | v • Reply • Share ›



Walt Corey → Le Trung Kien • 2 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.

1 ^ | v • Reply • Share ›




scvblwxq • a year ago

What about SQL and GUI programming?

1 ^ | v • Reply • Share ›


Load more comments

Subscribe Add Disqus to your site Add Disqus Add Privacy





Fitbit - Charge 2
Activity Tracker + Heart
Rate (Large) - Blk Slvr


\$149⁹⁵



Shop Now


 

©2016 Best Buy





Fitbit - Charge 2
Activity Tracker + Heart
Rate (Large) - Blk Slvr



\$149⁹⁵



Shop Now


 

©2016 Best Buy




Fitbit - Charge 2 Activity Tracker + Heart Rate (Small) - Plum Silver

\$149⁹⁵



Shop Now



©2016 Best Buy

Copyright © 2008 - 2016 Program Creek