

# Top 10 Algorithms for Coding Interview



194

Lastest Udate: 5/19/2015 ([PDF Version](#))

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](#). (Note: Similar problems are placed together even with the same number.)

## 1. String/Array

String is a class in Java. It contains a field of a character array and other fields and methods. 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
```

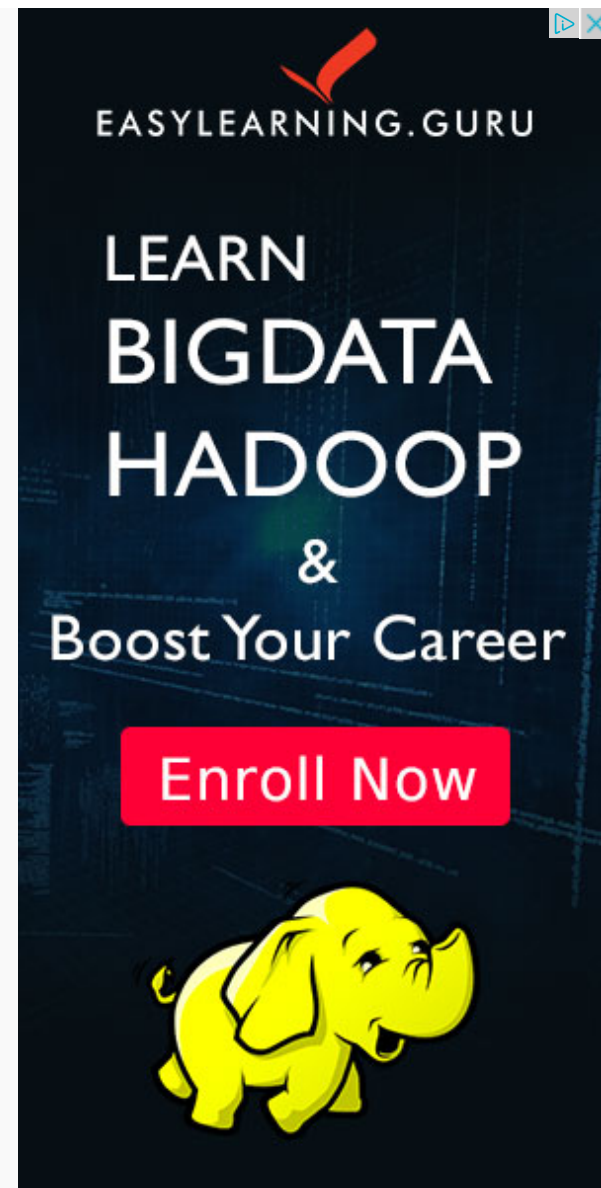
Search

```
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)
```

Strings/arrays are easy to understand, but the interview problems often require advanced algorithm to solve, such as dynamic programming, recursion, etc.

Classic problems:


- 1) Rotate Array
- 2) Evaluate Reverse Polish Notation (Stack)
- 3) Isomorphic Strings
- 4) Word Ladder (BFS)
- 4) Word Ladder II (BFS)
- 5) Median of Two Sorted Arrays
- 5) Kth Largest Element in an Array
- 6) Wildcard Matching
- 6) Regular Expression Matching
- 7) Merge Intervals
- 8) Insert Interval
- 9) Two Sum
- 9) Two Sum II – Input array is sorted
- 9) Two Sum III - Data structure design
- 9) 3Sum
- 9) 4Sum
- 10) 3Sum Closest
- 11) String to Integer
- 12) Merge Sorted Array
- 13) Valid Parentheses



EASYLEARNING.GURU

# LEARN BIGDATA HADOOP & Boost Your Career

**Enroll Now**



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  
24) Contains Duplicate II  
24) Contains Duplicate III  
25) Remove Duplicates from Sorted Array  
26) Remove Duplicates from Sorted Array II  
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  
30) Reverse Words in a String  
31) Find Minimum in Rotated Sorted Array  
31) Find Minimum in Rotated Sorted Array II  
31) Search in Rotated Sorted Array  
31) Search in Rotated Sorted Array II  
32) Find Peak Element  
33) Min Stack  
34) Majority Element  
34) Majority Element II  
35) Remove Element  
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
- 44) Pascal's Triangle 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
- 49) Anagrams
- 50) Shortest Palindrome
- 51) Rectangle Area
- 52) Summary Ranges

## 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
- 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)
- 9) Surrounded Regions (BFS)

10) Maximal Rectangle

10) Maximal Square

11) Word Search (DFS)

11) Word Search II

### 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:

- 1) Add Two Numbers
- 2) Reorder List
- 3) Linked List Cycle
- 4) Copy List with Random Pointer
- 5) Merge Two Sorted Lists
- 6) Merge k Sorted Lists \*
- 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
- 13) Reverse Linked List II
- 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

## 4. Tree & Heap

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.

Classic problems:

- 1) [Binary Tree Preorder Traversal](#)
- 2) [Binary Tree Inorder Traversal](#) [Palantir]
- 3) [Binary Tree Postorder Traversal](#)
- 4) [Binary Tree Level Order Traversal](#)
- 4) [Binary Tree Level Order Traversal II](#)
- 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](#) \*

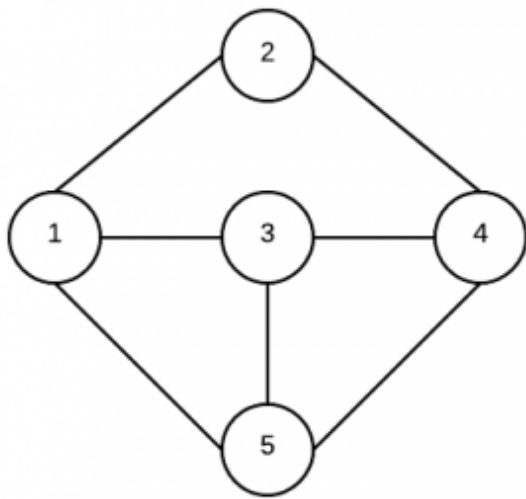


- 13) Balanced Binary Tree
- 14) Symmetric Tree
- 15) Binary Search Tree Iterator
- 16) Binary Tree Right Side View
- 17) Implement Trie (Prefix Tree)
- 18) Add and Search Word - Data structure design (DFS)
- 19) Merge k sorted arrays[Google]
- 20) Populating Next Right Pointers in Each Node
- 21) Populating 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) Invert Binary Tree
- 25) Kth Smallest Element in a BST
- 26) Lowest Common Ancestor of a Binary Search Tree
- 26) Lowest Common Ancestor of a Binary Tree

## 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 (BFS/DFS)
- 2) Course Schedule II (BFS)

## 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) First Missing Positive (Bucket Sort)
- 6) 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
- 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

## 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) Power of Two

## 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)

5) Combination Sum II (DFS)

5) Combination Sum III (DFS)

6) Combinations (DFS)

7) Letter Combinations of a Phone Number (DFS)

8) Restore IP Addresses

## 10. Math

Solving math problems usually require us to get some observations and form rules:



- 1) Reverse Integer
- 2) Palindrome Number
- 3) Pow(x,n)
- 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

## Additional Resources

1. Share your code to Github/BitBucket



## You May Also Like ...

1. [How to answer coding questions for your interview?](#)
2. [LeetCode – Search a 2D Matrix \(Java\)](#)
3. [LeetCode – LRU Cache \(Java\)](#)
4. [LeetCode – Binary Search Tree Iterator \(Java\)](#)

Category >> Algorithms >> Interview

If you want to post code and let me or someone else review it, please format your code in eclipse and put the code inside `<pre>` and `</pre>` tags.

**53 Comments**   **programcreek**

♥ Recommend 8    Share



Join the discussion...



**Joe Pepersack** · 2 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 reinventing wheels. You want your interview questions to test a candidate's ability to solve problems and design decisions. Reinventing the wheel is almost never a smart design decision.

89 ^ | v · Reply · Share ›



**ryanlr** Mod  Joe Pepersack · 2 years ago

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

25 ^ | v · Reply · Share ›



**Joe Pepersack**  ryanlr · 2 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 care less if you know how to implement a breadth-first search from memory, because that's a trivial problem. Even if you don't have a suitable library for your platform, the expectation for a solution is to use a library.

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 at a company. If I give you an interview problem that requires you to count the number of distinct elements in an input set, I'm testing you on your ability to recognize that you need to use a hash table, to justify that design decision, and your ability to tell me how it's going to scale... not on you coming up with an ad-hoc implementation of a hash table. Rolling your own hash table would be 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.

38 ^ | v • Reply • Share ›



**ryanlr** Mod → Joe Pepersack • 2 years ago

Those questions are indeed asked during interviews.

9 ^ | v • Reply • Share ›



**Joe Pepersack** → ryanlr • 2 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 best 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 kinds of questions, because it's obvious to me that they have no clue what they're doing. The article speaks volumes about their corporate culture, and none of it is positive.

17 ^ | v • Reply • Share ›



**Wei Qiu** → Joe Pepersack • a year 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.

27 ^ | v • Reply • Share ›



**Vinz** → Joe Pepersack • a year 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 :(

10 ^ | v • Reply • Share ›



**abossard** → Joe Pepersack • a year 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 me 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

But for companies which are driven by algorithms, like Google or Facebook, it does make sense, to check whether the applicant knows these things. Not to implement them by heart, but because the questions they asked are related to such algorithms. They ask to implement a linked list, but they may ask 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 the 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 got that job at Google :-)

4 ^ | v • Reply • Share ›



**Kenneth** → abossard • a year ago

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

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

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.

4 ^ | v • Reply • Share ›



**WJ** ➔ Joe Pepersack • a year ago

You are right. Top Tech companies won't ask these questions from you (definitely, YOU will apply there, a person will have 25 years of experience). If they do ask these questions, yeah!! You should probably walk out.....Actually, these questions are asked by TOP Tech companies for Software Engineer roles (Fresh/mid/senior) under 10 years of experience. Because they really want people who know how to invent wheels. 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 ...yes!!! you won't be asked such questions. Instead, companies will emphasize on design, your past experience plus your achievements, blogs, patents, your personal references, your links and your presence in IT industry. That's what companies like Google look for. I hope, you get it now!!

2 ^ | v • Reply • Share ›



**hardsoft** ➔ Joe Pepersack • a year 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 candidate's knowledge. If you walk out, I would (probably rightly) assume you are a prima donna 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 ›



**humfff humfff gnarl gnarl** . . ➔ Joe Pepersack • 7 days ago

Joe for president. He knows everything. I will vote you Dude. You seem so relaxed

cool

^ | v • Reply • Share ›



**Walt Corey** → Joe Pepersack • a year ago

Joe, you touched on the answer. These questions are asked by people who, other don't know how to interview a candidate. I'd go so far as to say it is a gotcha. That be a tad unfair, it depends on if the answer is relevant to the job or if it is from the r 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 t politely with your hands folded I was also interviewed at a startup and one questior what is the second argument to a hashmap constructor. I answered I was pretty s was capacity but as I use a smart IDE I let the IDE prompt me for the parameters. 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. akin to a MCS being way more important if you are 25 with 0 years of experience t 45 with 22 years of experience.

^ | v • Reply • Share ›



**chen gao** → Joe Pepersack • a year ago

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

^ | v • Reply • Share ›



**serge** → Joe Pepersack • a year ago

Design patterns are also solved problems. Do you want to hire someone who doe about them?

3 ^ | v • Reply • Share ›



**DC** → Joe Pepersack • 8 months ago

I was asked a couple of these questions on an Amazon interview about a month a interview went fairly well, although I didn't get the job. My guess is they ask the que a feel of how you analyze a problem, and your way of coming up with solutions. Th

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

^ | v • Reply • Share ›



**Oscar M** → Joe Pepersack • 9 months ago

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

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

Yes I agree that better questions could be asked, for example something like thes programming tests, they are focused on examining the candidate's problem solvir

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

^ | v • Reply • Share ›



**Walt Corey** → ryanlr • a year ago

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

^ | v • Reply • Share ›



**Виктор Маслов** → Joe Pepersack • a year ago

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

13 ^ | v • Reply • Share ›



**Paul** → Joe Pepersack · a year ago

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

3 ^ | v · Reply · Share ›



**Guest** → Paul · a year ago

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

7 ^ | v · Reply · Share ›



**Guest** → Joe Pepersack · a year ago

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

2 ^ | v · Reply · Share ›



**minus Seven** → Joe Pepersack · a year ago

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

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

1 ^ | v · Reply · Share ›



**Tom Bombadil** → Joe Pepersack · 5 days ago

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

^ | v · Reply · Share ›



**Joe Pepersack** → Tom Bombadil · 5 days ago

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

^ | v · Reply · Share ›



**Guest** → Joe Pepersack · a year ago

Well said.



^ | v · Reply · Share ›



**Johan Stén** · a year ago

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

I took part in a programming competition where the qualifying round was something akin to this. The final 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.

18 ^ | v · Reply · Share ›



**Mambo** · 2 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 enough people coming from university trying to solve everything with recursion - resulting in bad to read, 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 ›



**Wei Qiu** → Mambo · a year ago

Recursion is not hard to read. Recursion frequently leads to elegant and compact solutions. Check Sedgwick'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 optimization step when you get the solution right.

1 ^ | v · Reply · Share ›



**Walt Corey** → Wei Qiu · a year ago

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

1 ^ | v · Reply · Share ›



**Le Trung Kien** · a year ago

I don't know why some people keep complaining about how big companies conduct interviews, as if they 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 · a year 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 fit into an airplane? How many gas stations in Manhattan? A complete waste of time. They don't p anything. They serve primarily to make the interviewer feel smart."

<http://techcrunch.com/2013/06/...>

2 ^ | v · Reply · Share ›



**Walt Corey** → Le Trung Kien · a year ago

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

1 ^ | v · Reply · Share ›



**Drew** · a year ago

I find it funny that there is so much bashing about how these problems don't really tell the interviewer any than being able to jump through some hoops. I completely disagree with those sentiments. Having problem these while trivial for some, can quickly weed out those who don't have fundamental understanding of basic structure concepts or classic algorithms. Sure, there are libraries for such things to hide away the gory code you blindly just use them without knowing when they apply, you get into bad habits. These classical problem 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 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 see why a site advertises those type of questions because they can be much more subjective and open for great discussion.

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, it's a much harder job.

be knowledgeable enough to use classic algorithms and shape it to meet the necessary solution.

1 ^ | v · Reply · Share ›



**Walt Corey** → Drew · a year ago

Knowing how they apply is a vastly different problem than how to implement them. As a previous 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 ›



**not-just-yeti** · 2 years ago

Rather a nit-pick, but fwiw: what you call dynamic programming is really just "memoization".

DP is memoization plus an add'l optimization: don't memoize the entire table of other-inputs; instead try and only keep some portion of that table, and calculate in a way that you only need refer to that portion. (I Fibonacci, that the DP version only keeps the last two elements of the array and computes from 0 upwa #6). In other problems, rather than keep an entire 2-D array you might keep just the last column of that at the last three columns, if that's as far as the recursion needs to go.)

1 ^ | v · Reply · Share ›



**Fidel** → not-just-yeti · 2 years ago

"Bin Sort" is lacking in the sorting algorithms, as well as "Radix sort"!! And "Heap Sort" as well.

Besides, I would have put "Count Sort" after all the other methods, and after "Bin Sort" and "Radix Sort" because the three of them (BinSort, Radix Sort and CountSort) use a different set of assumption rest, and so they are not "general" sorting methods.

1 ^ | v · Reply · Share ›

This comment was deleted.



**ryanlr** Mod → Guest · 2 years ago

Thousands of people have visited this page, you are the only one who points this out! Thank you.

^ | v · Reply · Share ›



**Walt Corey** → ryanlr · a year ago

that's because the other 1,999 people didn't care.

3 ^ | v · Reply · Share ›



**GEEK** · 2 years ago

I think the worst-case running time of counting sort is  $O(N+K)$ , according to the book <introduction to algorithms>.

1 ^ | v · Reply · Share ›



**Runiko** · a month ago

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

^ | v · Reply · Share ›



**scvblwxq** · 3 months ago

What about SQL and GUI programming?

^ | v · Reply · Share ›



**Youchen** · 7 months ago

Great to see this post, Thanks a lot!

^ | v · Reply · Share ›



**Henry** · 10 months ago

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

^ | v · Reply · Share ›



**RA the Guest** → Henry · 7 months ago

Did you get the job?

1 ^ | v · Reply · Share ›



**atlas** · 10 months 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 understand contents of article.

Hope to get your permit.

Have a good day!

^ | v · Reply · Share ›



**ryanlr** Mod → atlas · 10 months ago

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



**ISuckatProgramming** · a year ago

Having done more than 20 or more tech interviews during a job search recently, I've found the lists in this very useful. Most companies like Google or LinkedIn will first give you a phone screen and there you will get 1 or 3 questions of the variety found here. Once you survive that and get to the in person interview, then the REAL fun begins. You typically get much harder programming problems and have much less time to solve them. I guess most developers at Google or Facebook must be geniuses or something, I guess this is a weed out all the people of only average intelligence. :(

^ | v · Reply · Share ›



**Aarohi Shirke** · a year ago

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

^ | v · Reply · Share ›

Load more comments

#### ALSO ON PROGRAMCREEK

### LeetCode – Course Schedule II (Java)

1 comment · 2 months ago



**Yang L** — this is BFS

### LeetCode – Kth Smallest Element in a BST (Java)

2 comments · a month ago



**Megha Tiwari** — I think the program should return the result immediately after k equals 0, its unnecessary to keep on filling stack even ...

### LeetCode – Contains Duplicate III (Java)

2 comments · 2 months ago



**coder** — In that case, 'true' must be returned in the snippet.

### LeetCode – Divide Two Integers (Java)

1 comment · 3 months ago



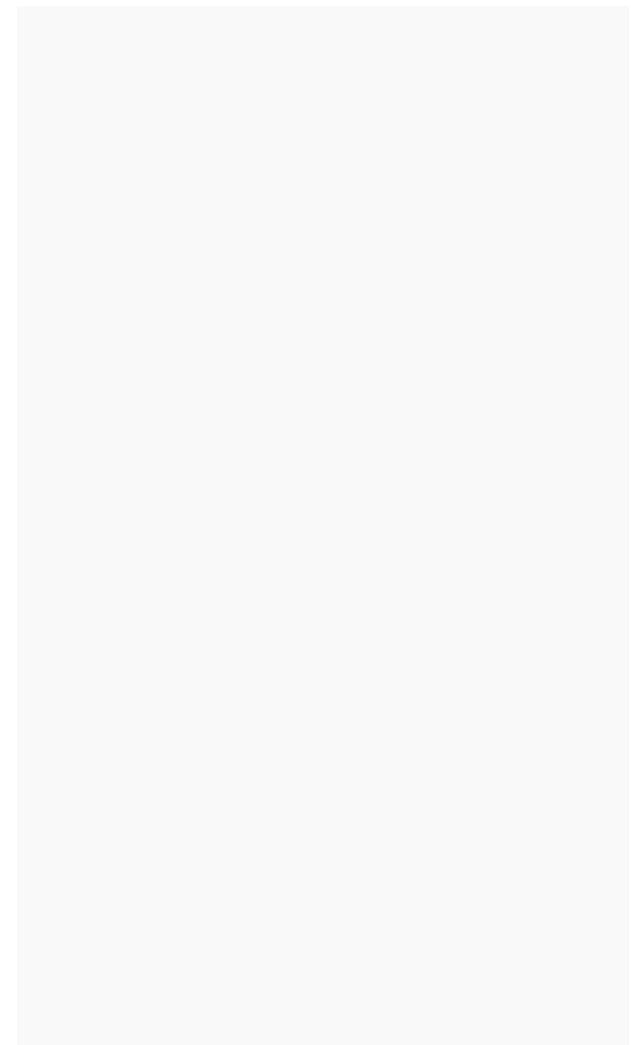
**Saurabh Rane** — Thank you for this code, this can be solved using sum as follows - public static int divide(int dividend, int divisor) { int ...

[Subscribe](#)

[Add Disqus to your site](#)

[Privacy](#)

|



---

Copyright © 2008 - 2015 programcreek.com