Part 1

Data Structures

Chapter 1 | Arrays and Strings

Hash Tables

While not all problems can be solved with hash tables, a shocking number of interview problems can be. Before your interview, make sure to practice both using and implementing hash tables.

```
public HashMap<Integer, Student> buildMap(Student[] students) {
    HashMap<Integer, Student> map = new HashMap<Integer, Student>();
    for (Student s : students) map.put(s.getId(), s);
    return map;
}
```

ArrayList (Dynamically Resizing Array):

An ArrayList, or a dynamically resizing array, is an array that resizes itself as needed while still providing O(1) access. A typical implementation is that when a vector is full, the array doubles in size. Each doubling takes O(n) time, but happens so rarely that its amortized time is still O(1).

```
public ArrayList<String> merge(String[] words, String[] more) {
    ArrayList<String> sentence = new ArrayList<String>();
    for (String w : words) sentence.add(w);
    for (String w : more) sentence.add(w);
    return sentence;
}
```

StringBuffer / StringBuilder

Question: What is the running time of this code?

```
public String makeSentence(String[] words) {
    StringBuffer sentence = new StringBuffer();
    for (String w : words) sentence.append(w);
    return sentence.toString();
}
```

Answer: O(n^2), where n is the number of letters in sentence. Here's why: each time you append a string to sentence, you create a copy of sentence and run through all the letters in sentence to copy them over. If you have to iterate through up to n characters each time in the loop, and you're looping at least n times, that gives you an O(n^2) run time. Ouch!

With StringBuffer (or StringBuilder) can help you avoid this problem.

```
public String makeSentence(String[] words) {
    StringBuffer sentence = new StringBuffer();
    for (String w : words) sentence.append(w);
    return sentence.toString();
}
```

Chapter 1 | Arrays and Strings

1.1 Implement an algorithm to determine if a string has all unique characters. What if you can not use additional data structures?

pg 95

1.2 Write code to reverse a C-Style String. (C-String means that "abcd" is represented as five characters, including the null character.)

pg 96

1.3 Design an algorithm and write code to remove the duplicate characters in a string without using any additional buffer. NOTE: One or two additional variables are fine. An extra copy of the array is not.

FOLLOW UP

Write the test cases for this method.

pg 97

1.4 Write a method to decide if two strings are anagrams or not.

pg 99

1.5 Write a method to replace all spaces in a string with '%20'.

pg 100

1.6 Given an image represented by an NxN matrix, where each pixel in the image is 4 bytes, write a method to rotate the image by 90 degrees. Can you do this in place?

pg 101

1.7 Write an algorithm such that if an element in an MxN matrix is 0, its entire row and column is set to 0.

pg 102

1.8 Assume you have a method isSubstring which checks if one word is a substring of another. Given two strings, s1 and s2, write code to check if s2 is a rotation of s1 using only one call to isSubstring (i.e., "waterbottle" is a rotation of "erbottlewat").

pg 103

Chapter 2 | Linked Lists

How to Approach:

Linked list questions are extremely common. These can range from simple (delete a node in a linked list) to much more challenging. Either way, we advise you to be extremely comfortable with the easiest questions. Being able to easily manipulate a linked list in the simplest ways will make the tougher linked list questions much less tricky. With that said, we present some "must know" code about linked list manipulation. You should be able to easily write this code yourself prior to your interview.

Creating a Linked List:

NOTE: When you're discussing a linked list in an interview, make sure to understand whether it is a single linked list or a doubly linked list.

```
class Node {
    Node next = null;
    int data;
    public Node(int d) { data = d; }
    void appendToTail(int d) {
        Node end = new Node(d);
        Node n = this;
        while (n.next != null) { n = n.next; }
        n.next = end;
    }
}
```

Deleting a Node from a Singly Linked List

```
Node deleteNode(Node head, int d) {
    Node n = head;
    if (n.data == d) {
        return head.next; /* moved head */
    }
    while (n.next != null) {
        if (n.next.data == d) {
            n.next = n.next.next;
            return head; /* head didn't change */
        }
        n = n.next;
    }
}
```

Chapter 2 | Linked Lists

2.1 Write code to remove duplicates from an unsorted linked list.

FOLLOW UP

How would you solve this problem if a temporary buffer is not allowed?

pg 105

2.2 Implement an algorithm to find the nth to last element of a singly linked list.

pg 106

2.3 Implement an algorithm to delete a node in the middle of a single linked list, given only access to that node.

EXAMPLE

Input: the node 'c' from the linked list a->b->c->d->e

Result: nothing is returned, but the new linked list looks like a->b->d->e

pg 107

2.4 You have two numbers represented by a linked list, where each node contains a single digit. The digits are stored in reverse order, such that the 1's digit is at the head of the list. Write a function that adds the two numbers and returns the sum as a linked list.

EXAMPLE

Input: (3 -> 1 -> 5) + (5 -> 9 -> 2)

Output: 8 -> 0 -> 8

pg 108

2.5 Given a circular linked list, implement an algorithm which returns node at the beginning of the loop.

DEFINITION

Circular linked list: A (corrupt) linked list in which a node's next pointer points to an earlier node, so as to make a loop in the linked list.

EXAMPLE

input: A -> B -> C -> D -> E -> C [the same C as earlier]

output: C

pg 109

Chapter 3 | Stacks and Queues

How to Approach:

Whether you are asked to implement a simple stack / queue, or you are asked to implement a modified version of one, you will have a big leg up on other candidates if you can flawlessly work with stacks and queues. Practice makes perfect! Here is some skeleton code for a Stack and Queue class.

Implementing a Stack

```
class Stack {
    Node top;
    Node pop() {
        if (top != null) {
            Object item = top.data;
            top = top.next;
            return item;
        }
        return null;
    }
    void push(Object item) {
        Node t = new Node(item);
        t.next = top;
        top = t;
    }
}
```

Implementing a Queue

```
class Queue {
    Node first, last;
    void enqueue(Object item) {
        if (!first) {
            back = new Node(item);
            first = back;
        } else {
            back.next = new Node(item);
            back = back.next;
        }
    }
    Node dequeue(Node n) {
        if (front != null) {
            Object item = front.data;
            front = front.next;
            return item;
        }
        return null;
    }
}
```

Chapter 3 | Stacks and Queues

3.1 Describe how you could use a single array to implement three stacks.

pg 111

3.2 How would you design a stack which, in addition to push and pop, also has a function min which returns the minimum element? Push, pop and min should all operate in O(1) time.

pg 113

3.3 Imagine a (literal) stack of plates. If the stack gets too high, it might topple. Therefore, in real life, we would likely start a new stack when the previous stack exceeds some threshold. Implement a data structure SetOfStacks that mimics this. SetOfStacks should be composed of several stacks, and should create a new stack once the previous one exceeds capacity. SetOfStacks.push() and SetOfStacks.pop() should behave identically to a single stack (that is, pop() should return the same values as it would if there were just a single stack).

FOLLOW UP

Implement a function popAt(int index) which performs a pop operation on a specific sub-stack.

pg 115

3.4 In the classic problem of the Towers of Hanoi, you have 3 rods and N disks of different sizes which can slide onto any tower. The puzzle starts with disks sorted in ascending order of size from top to bottom (e.g., each disk sits on top of an even larger one). You have the following constraints:

(A) Only one disk can be moved at a time.

(B) A disk is slid off the top of one rod onto the next rod.

(C) A disk can only be placed on top of a larger disk.

Write a program to move the disks from the first rod to the last using Stacks.

pg 118

3.5 Implement a MyQueue class which implements a queue using two stacks.

pg 120

3.6 Write a program to sort a stack in ascending order. You should not make any assumptions about how the stack is implemented. The following are the only functions that should be used to write this program: push | pop | peek | isEmpty.

pg 121

Chapter 4 | Trees and Graphs

How to Approach:

Trees and graphs questions typically come in one of two forms:

1. Implement a tree / find a node / delete a node / other well known algorithm.
2. Implement a modification of a known algorithm.

Either way, it is *strongly* recommended to understand the important tree algorithms prior to your interview. If you're fluent in these, it'll make the tougher questions that much easier! We'll list some of the most important.

WARNING: Not all binary trees are binary search trees

When given a binary tree question, many candidates assume that the interviewer means "binary *search* tree", when the interviewer might only mean "binary tree." So, listen carefully for that word "search." If you don't hear it, the interviewer may just mean a binary tree with no particular ordering on the nodes. If you aren't sure, ask.

Binary Trees—"Must Know" Algorithms

You should be able to easily implement the following algorithms prior to your interview:

» **In-Order:** Traverse left node, current node, then right [usually used for binary search trees]

» **Pre-Order:** Traverse current node, then left node, then right node.

» **Post-Order:** Traverse left node, then right node, then current node.

» **Insert Node:** On a binary search tree, we insert a value v, by comparing it to the root. If v > root, we go right, and else we go left. We do this until we hit an empty spot in the tree.

Note: balancing and deletion of binary search trees are rarely asked, but you might want to have some idea how they work. It can set you apart from other candidates.

Graph Traversal—"Must Know" Algorithms

You should be able to easily implement the following algorithms prior to your interview:

» **Depth First Search:** DFS involves searching a node and all its children before proceeding to its siblings.

» **Breadth First Search:** BFS involves searching a node and its siblings before going on to any children.

Chapter 4 | Trees and Graphs

4.1 Implement a function to check if a tree is balanced. For the purposes of this question, a balanced tree is defined to be a tree such that no two leaf nodes differ in distance from the root by more than one.

pg 123

4.2 Given a directed graph, design an algorithm to find out whether there is a route between two nodes.

pg 124

4.3 Given a sorted (increasing order) array, write an algorithm to create a binary tree with minimal height.

pg 125

4.4 Given a binary search tree, design an algorithm which creates a linked list of all the nodes at each depth (i.e., if you have a tree with depth D, you'll have D linked lists).

pg 126

4.5 Write an algorithm to find the 'next' node (i.e., in-order successor) of a given node in a binary search tree where each node has a link to its parent.

pg 127

4.6 Design an algorithm and write code to find the first common ancestor of two nodes in a binary tree. Avoid storing additional nodes in a data structure. NOTE: This is not necessarily a binary search tree.

pg 128

4.7 You have two very large binary trees: T1, with millions of nodes, and T2, with hundreds of nodes. Create an algorithm to decide if T2 is a subtree of T1.

pg 130

4.8 You are given a binary tree in which each node contains a value. Design an algorithm to print all paths which sum up to that value. Note that it can be any path in the tree - it does not have to start at the root.

pg 131

Part 2
Concepts and Algorithms

Chapter 5 | Bit Manipulation

How to Approach:

Bit manipulation can be a scary thing to many candidates, but it doesn't need to be! If you're shaky on bit manipulation, we recommend doing a couple of arithmetic-like problems to boost your skills. Compute the following by hand:

`1010 - 0001`	`1010 + 0110`	`1100^1010`
`1010 << 1`	`1001^1001`	`1001 & 1100`
`1010 >> 1`	`0xFF - 1`	`0xAB + 0x11`

If you're still uncomfortable, examine very carefully what happens when you do subtraction, addition, etc in base 10. Can you repeat that work in base 2?

NOTE: The Windows Calculator knows how to do lots of operations in binary, including ADD, SUBTRACT, AND and OR. Go to View > Programmer to get into binary mode while you practice.

Things to Watch Out For:

» It's really easy to make mistakes on these problems, so be careful! When you're writing code, stop and think about what you're writing every couple of lines - or, better yet, test your code mid-way through! When you're done, check through your entire code.

» If you're bit shifting, what happens when the digits get shifted off the end? Make sure to think about this case to ensure that you're handling it correctly.

And (&):	`0 & 0 = 0`	`1 & 0 = 0`	`0 & 1 = 0`	`1 & 1 = 1`
Or (|):	`0 | 0 = 0`	`1 | 0 = 1`	`0 | 1 = 1`	`1 | 1 = 1`
Xor (^):	`0 ^ 0 = 0`	`1 ^ 0 = 1`	`0 ^ 1 = 1`	`1 ^ 1 = 0`

Left Shift:

x << y means x shifted y bits to the left. If you start shifting and you run out of space, the bits just "drop off". For example:

```
00011001 << 2 = 01100100
00011001 << 4 = 10010000
```

Right Shift:

x >> y means x shifted y bits to the right. If you start shifting and you run out of space, the bits just "drop off" the end. Example:

```
00011001 >> 2 = 00000110
00011001 >> 4 = 00000001
```

Chapter 5 | Bit Manipulation

5.1 You are given two 32-bit numbers, N and M, and two bit positions, i and j. Write a method to set all bits between i and j in N equal to M (e.g., M becomes a substring of N located at i and starting at j).

EXAMPLE:

Input: N = 10000000000, M = 10101, i = 2, j = 6

Output: N = 10001010100

pg 133

5.2 Given a (decimal - e.g. 3.72) number that is passed in as a string, print the binary representation. If the number can not be represented accurately in binary, print "ERROR"

pg 134

5.3 Given an integer, print the next smallest and next largest number that have the same number of 1 bits in their binary representation.

pg 135

5.4 Explain what the following code does: ((n & (n-1)) == 0).

pg 138

5.5 Write a function to determine the number of bits required to convert integer A to integer B.

Input: 31, 14

Output: 2

pg 139

5.6 Write a program to swap odd and even bits in an integer with as few instructions as possible (e.g., bit 0 and bit 1 are swapped, bit 2 and bit 3 are swapped, etc).

pg 140

5.7 An array A[1...n] contains all the integers from 0 to n except for one number which is missing. In this problem, we cannot access an entire integer in A with a single operation. The elements of A are represented in binary, and the only operation we can use to access them is "fetch the jth bit of A[i]", which takes constant time. Write code to find the missing integer. Can you do it in O(n) time?

pg 141

Chapter 6 | Brain Teasers

Do companies really ask brain teasers?

While many companies, including Google and Microsoft, have policies banning brain teasers, interviewers still sometimes ask these tricky questions. This is especially true since people have different definitions of brain teasers.

Advice on Approaching Brain Teasers

Don't panic when you get a brain teaser. Interviewers want to see how you tackle a problem; they don't expect you to immediately know the answer. Start talking, and show the interviewer how you approach a problem.

In many cases, you will also find that the brain teasers have some connection back to fundamental laws or theories of computer science.

If you're stuck, we recommend simplifying the problem. Solve it for a small number of items or a special case, and then see if you can generalize it.

Example

You are trying to cook an egg for exactly fifteen minutes, but instead of a timer, you are given two ropes which burn for exactly 1 hour each. The ropes, however, are of uneven densities - i.e., half the rope length-wise might take only two minutes to burn.

The Approach

1. What is important? Numbers usually have a meaning behind them. The fifteen minutes and two ropes were picked for a reason.
2. Simplify! You can easily time one hour (burn just one rope).
3. Now, can you time 30 minutes? That's half the time it takes to burn one rope. Can you burn the rope twice as fast? Yes! (Light the rope at both ends.)
4. You've now learned: (1) You can time 30 minutes. (2) You can burn a rope that takes X minutes in just X/2 minutes by lighting both ends.
5. Work backwards: if you had a rope of burn-length 30 minutes, that would let you time 15 minutes. Can you remove 30 minutes of burn-time from a rope?
6. You can remove 30 minutes of burn-time from Rope #2 by lighting Rope #1 at both ends and Rope #2 at one end.
7. Now that you have Rope #2 at burn-length 30 minutes, start cooking the egg and light Rope #2 at the other end. When Rope #2 burns up, your egg is done!

Chapter 6 | Brain Teasers

6.1 Add arithmetic operators (plus, minus, times, divide) to make the following expression true: 3 1 3 6 = 8. You can use any parentheses you'd like.

pg 143

6.2 There is an 8x8 chess board in which two diagonally opposite corners have been cut off. You are given 31 dominos, and a single domino can cover exactly two squares. Can you use the 31 dominos to cover the entire board? Prove your answer (by providing an example, or showing why it's impossible).

pg 144

6.3 You have a five quart jug and a three quart jug, and an unlimited supply of water (but no measuring cups). How would you come up with exactly four quarts of water?

NOTE: The jugs are oddly shaped, such that filling up exactly 'half' of the jug would be impossible.

pg 145

6.4 A bunch of men are on an island. A genie comes down and gathers everyone together and places a magical hat on some people's heads (i.e., at least one person has a hat). The hat is magical: it can be seen by other people, but not by the wearer of the hat himself. To remove the hat, those (and only those who have a hat) must dunk themselves underwater at exactly midnight. If there are n people and c hats, how long does it take the men to remove the hats? The men cannot tell each other (in any way) that they have a hat.

FOLLOW UP

Prove that your solution is correct.

pg 146

6.5 There is a building of 100 floors. If an egg drops from the Nth floor or above it will break. If it's dropped from any floor below, it will not break. You're given 2 eggs. Find N, while minimizing the number of drops for the worst case.

pg 148

6.6 There are one hundred closed lockers in a hallway. A man begins by opening all one hundred lockers. Next, he closes every second locker. Then he goes to every third locker and closes it if it is open or opens it if it is closed (e.g., he toggles every third locker). After his one hundredth pass in the hallway, in which he toggles only locker number one hundred, how many lockers are open?

pg 149

Chapter 7 | Object Oriented Design

How to Approach

Object oriented design questions are very important, as they demonstrate the quality of a candidate's code. A poor performance on this type of question raises serious red flags.

Handling Ambiguity in an Interview

OOD questions are often intentionally vague to test if you'll make assumptions, or if you'll ask clarifying questions. How do you design a class if the constraints are vague? Ask questions to eliminate ambiguity, then design the classes to handle any remaining ambiguity.

Object Oriented Design for Software

Imagine we're designing the objects for a deck of cards. Consider the following approach:

1. What are you trying to do with the deck of cards? Ask your interviewer. Let's assume we want a general purpose deck of cards to implement many different types of card games.
2. What are the core objects—and what "sub types" are there? For example, the core items might be: Card, Deck, Number, Suit, PointValue
3. Have you missed anything? Think about how you'll use that deck of cards to implement different types of games, changing the class design as necessary.
4. Now, get a little deeper: how will the methods work? If you have a method like Card Deck:.getCard(Suit s, Number n), think about how it will retrieve the card.

Object Oriented Design for Real World Object

Real world objects are handled very similarly to software object oriented design. Suppose you are designing an object oriented design for a parking lot:

1. What are your goals? For example: figure out if a parking spot is taken, figure out how many cars of each type are in the parking lot, look up handicapped spots, etc.
2. Now, think about the core objects (Car, ParkingSpot, ParkingLot, ParkingMeter, etc—Car has different subclasses, and ParkingSpot is also subclassed for handicapped spot).
3. Have we missed anything? How will we represent parking restrictions based on time or payment? Perhaps, we'll add a class called Permission which handles different payment systems. Permission will be sub-classed into classes PaidPermission (fee to park) and FreeParking (open parking). ParkingLot will have a method called GetPermission which will return the current Permission object based on the time.
4. How will we know whether or not a car is in a spot? Think about how to represent the data so that the methods are most efficient.

Chapter 7 | Object Oriented Design

7.1 Design the data structures for a generic deck of cards. Explain how you would subclass it to implement particular card games.

pg 151

7.2 Imagine you have a call center with three levels of employees: fresher, technical lead (TL), product manager (PM). There can be multiple employees, but only one TL or PM. An incoming telephone call must be allocated to a fresher who is free. If a fresher can't handle the call, he or she must escalate the call to technical lead. If the TL is not free or not able to handle it, then the call should be escalated to PM. Design the classes and data structures for this problem. Implement a method getCallHandler().

pg 152

7.3 Design a musical juke box using object oriented principles.

pg 154

7.4 Design a chess game using object oriented principles.

pg 156

7.5 Design the data structures for an online book reader system.

pg 157

7.6 Implement a jigsaw puzzle. Design the data structures and explain an algorithm to solve the puzzle.

pg 159

7.7 Explain how you would design a chat server. In particular, provide details about the various backend components, classes, and methods. What would be the hardest problems to solve?

pg 161

7.8 Othello is played as follows: Each Othello piece is white on one side and black on the other. When a piece is surrounded by its opponents on both the left and right sides, or both the top and bottom, it is said to be captured and its color is flipped. On your turn, you must capture at least one of your opponent's pieces. The game ends when either user has no more valid moves, and the win is assigned to the person with the most pieces. Implement the object oriented design for Othello.

pg 163

7.9 Explain the data structures and algorithms that you would use to design an in-memory file system. Illustrate with an example in code where possible.

pg 166

7.10 Describe the data structures and algorithms that you would use to implement a garbage collector in C++.

pg 167

Chapter 8 | Recursion

How to Recognize

While there is a wide variety of recursive problems, many recursive problems follow similar patterns. A good hint that problem is recursive is that it appears to be built off sub-problems.

When you hear a problem beginning with the following, it's often (though not always) a good candidate for recursion: "Design an algorithm to compute the nth ..."; "Write code to list the first n..."; "Implement a method to compute all..."; etc.

Again, practice makes perfect! The more problems you do, the easier it will be to recognize recursive problems.

How to Approach

Recursive solutions, by definition, are built off solutions to sub-problems. Many times, this will mean simply to compute f(n) by adding something, removing something, or otherwise changing the solution for f(n-1). In other cases, you might have to do something more complicated. Regardless, we recommend the following approach:

1. Think about what the sub-problem is. How many sub-problems does f(n) depend on? That is, in a recursive binary tree problem, each part will likely depend on two problems. In a linked list problem, it'll probably be just one.
2. Solve for a "base case." That is, if you need to compute f(n), first compute it for f(0) or f(1). This is usually just a hard-coded value.
3. Solve for f(2).
4. Understand how to solve for f(3) using f(2) (or previous solutions). That is, understand the exact process of translating the solutions for sub-problems into the real solution.
5. Generalize for f(n).

This "bottom-up recursion" is often the most straight-forward. Sometimes, though, it can be useful to approach problems "top down", where you essentially jump directly into breaking f(n) into its sub-problems.

Things to Watch Out For

1. All problems that can be solved recursively can also be solved iteratively (though the code may be much more complicated). Before diving into a recursive code, ask yourself how hard it would be to implement this algorithm iteratively. Discuss the trade-offs with your interviewer.
2. Recursive algorithms can be very space inefficient. Each recursive call adds a new layer to the stack, which means that if your algorithm has O(n) recursive calls then it uses O(n) memory. Ouch! This is one reason why an iterative algorithm may be better.

Chapter 8 | Recursion

8.1 Write a method to generate the nth Fibonacci number.

pg 169

8.2 Imagine a robot sitting on the upper left hand corner of an NxN grid. The robot can only move in two directions: right and down. How many possible paths are there for the robot?

FOLLOW UP

Imagine certain squares are "off limits", such that the robot can not step on them. Design an algorithm to get all possible paths for the robot.

pg 170

8.3 Write a method that returns all subsets of a set.

pg 171

8.4 Write a method to compute all permutations of a string.

pg 173

8.5 Implement an algorithm to print all valid (e.g., properly opened and closed) combinations of n-pairs of parentheses.

EXAMPLE:

input: 3 (e.g., 3 pairs of parentheses)

output: ()()(), ()(()), (())(), ((()))

pg 174

8.6 Implement the "paint fill" function that one might see on many image editing programs. That is, given a screen (represented by a 2 dimensional array of Colors), a point, and a new color, fill in the surrounding area until you hit a border of that color.'

pg 175

8.7 Given an infinite number of quarters (25 cents), dimes (10 cents), nickels (5 cents) and pennies (1 cent), write code to calculate the number of ways of representing n cents.

pg 176

8.8 Write an algorithm to print all ways of arranging eight queens on a chess board so that none of them share the same row, column or diagonal.

pg 177

Chapter 9 | Sorting and Searching

How to Approach:

Understanding the common sorting algorithms is incredibly valuable, as many sorting or searching solutions require tweaks of known sorting algorithms. A good approach when you are given a question like this is to run through the different sorting algorithms and see if one applies particularly well.

Example: You have a very large array of 'Person' objects. Sort the people in increasing order of age.

We're given two interesting bits of knowledge here: (1) It's a large array, so efficiency is very important. (2) We are sorting based on ages, so we know the values are in a small range. By scanning through the various sorting algorithms, we might notice that bucket sort would be a perfect candidate for this algorithm. In fact, we can make the buckets small (just 1 year each) and get O(n) running time.

Bubble Sort:

Start at the beginning of an array and swap the first two elements if the first is bigger than the second. Go to the next pair, etc, continuously making sweeps of the array until sorted. O(n^2).

Selection Sort:

Find the smallest element using a linear scan and move it to the front. Then, find the second smallest and move it, again doing a linear scan. Continue doing this until all the elements are in place. O(n^2).

Merge Sort:

Sort each pair of elements. Then, sort every four elements by merging every two pairs. Then, sort every 8 elements, etc. O(n log n) expected and worst case.

Quick Sort:

Pick a random element and partition the array, such that all numbers that are less than it come before all elements that are greater than it. Then do that for each half, then each quarter, etc. O(n log n) expected, O(n^2) worst case.

Bucket Sort:

Partition the array into a finite number of buckets, and then sort each bucket individually. This gives a time of O(n + m), where n is the number of items and m is the number of distinct items.

Chapter 9 | Sorting and Searching

9.1 You are given two sorted arrays, A and B, and A has a large enough buffer at the end to hold B. Write a method to merge B into A in sorted order.

pg 179

9.2 Write a method to sort an array of strings so that all the anagrams are next to each other.

pg 180

9.3 Given a sorted array of n integers that has been rotated an unknown number of times, give an O(log n) algorithm that finds an element in the array. You may assume that the array was originally sorted in increasing order.

EXAMPLE:

Input: find 5 in array (15 16 19 20 25 1 3 4 5 7 10 14)

Output: 8 (the index of 5 in the array)

pg 181

9.4 If you have a 2 GB file with one string per line, which sorting algorithm would you use to sort the file and why?

pg 182

9.5 Given a sorted array of strings which is interspersed with empty strings, write a method to find the location of a given string.

Example: find "ball" in ["at", "", "", "", "ball", "", "", "car", "", "", "dad", "", ""] will return 4
Example: find "ballcar" in ["at", "", "", "", "", "ball", "car", "", "", "dad", "", ""] will return -1

pg 183

9.6 Given a matrix in which each row and each column is sorted, write a method to find an element in it.

pg 184

9.7 A circus is designing a tower routine consisting of people standing atop one another's shoulders. For practical and aesthetic reasons, each person must be both shorter and lighter than the person below him or her. Given the heights and weights of each person in the circus, write a method to compute the largest possible number of people in such a tower.

EXAMPLE:

Input (ht, wt): (65, 100) (70, 150) (56, 90) (75, 190) (60, 95) (68, 110)

Output: The longest tower is length 6 and includes from top to bottom: (56, 90) (60,95) (65,100) (68,110) (70,150) (75,190)

pg 185

Chapter 10 | Mathematical

How to Approach:

Many of these problems read as brain teasers at first, but can be worked through in a logical way. Just remember to rely on the rules of mathematics to develop an approach, and then to carefully translate that idea into code.

Example: Given two numbers m and n, write a method to return the first number r that is divisible by both (e.g., the least common multiple).

The Approach: What does it mean for r to be divisible by m and n? It means that all the primes in m must go into r, and all primes in n must be in r. What if m and n have primes in common? For example, if m is divisible by 3^5 and n is divisible by 3^7, what does this mean about r? It means r must be divisible by 3^7.

> *The Rule:* For each prime p such that p^a \ m (e.g., m is divisible by p^a) and p^b \ n, r must be divisible by p^max(a, b).

The Algorithm:

```
Define q to be 1.
for each prime number p less than m and n:
    find the largest a and b such that p^a \ m and p^b \ n
    let q = q * p^max(a, b)
return q
```

NOTE: An alternate solution involves recognizing that gcd(a, b) * lcm(a, b) = ab. One could then compute the gcd(a, b) using the Euclidean algorithm. Of course, unless you already know this fact, it's unlikely that this rule would occur to you during an interview.

Things to Watch Out For:

1. Be careful with the difference in precision between floats vs. doubles.
2. Don't assume that a value (such as the slope of a line) is an int unless you've been told so.

Bayes' Rule and Probability

1. If A and B are independent, then P(A and B) = P(A) * P(B).
2. Else (in general), P(A and B) = P(A given B) * P(B)
3. If A and B are mutually exclusive (e.g., if one happens, the other one can't), P(A or B) = P(A) + P(B).
4. Else (in general), P(A or B) = P(A) + P(B) - P(A and B).

Chapter 10 | Mathematical

10.1 You have a basketball hoop and someone says that you can play 1 of 2 games.

Game #1: You get one shot to make the hoop.

Game #2: You get three shots and you have to make 2 of 3 shots.

If p is the probability of making a particular shot, for which values of p should you pick one game or the other?

pg 187

10.2 There are three ants on different vertices of a triangle. What is the probability of collision (between any two or all of them) if they start walking on the sides of the triangle?

Similarly find the probability of collision with 'n' ants on an 'n' vertex polygon.

pg 188

10.3 Given two lines on a Cartesian plane, determine whether the two lines would intersect.

pg 189

10.4 Write a method to implement *, - , / operations. You should use only the + operator.

pg 190

10.5 Given two squares on a two dimensional plane, find a line that would cut these two squares in half.

pg 192

10.6 Given a two dimensional graph with points on it, find a line which passes the most number of points.

pg 193

10.7 Design an algorithm to find the kth number such that the only prime factors are 3, 5, and 7.

pg 195

Chapter 11 | Testing

Testing Problems: Not Just for Testers!

Although testers are obviously asked more testing problems, developers will often be asked testing problems as well. Why? Because a good developer knows how to test their code!

Types of Testing Problems:

Testing problems generally fall into one of three categories:

1. Explain how you would test this real world object (pen, paperclip, etc).
2. Explain how you would test this computer software (e.g., a web browser).
3. Write test cases / test code to test this specific method.

We'll discuss type #1, since it's usually the most daunting. Remember that all three types require you to not make assumptions that the input or the user will play nice. Expect abuse and plan for it.

How to Test A Real World Object

Let's imagine that you were asked to test a paperclip. The first thing to understand is: what is it expected to be used for and who are the expected users. Ask your interviewer—the answer may not be what you think! The answer could be "by teachers, to hold papers together" or it could be "by artists, to bend into new shapes." These two use-cases will have very different answers. Once you understand the intended use, think about:

» What are the specific use cases for the intended purpose? For example, holding 2 sheets of paper together, and up to 30 sheets. If it fails, does it fail gracefully? (see below)

» What does it mean for it to fail? Answer: "*Failing gracefully*" means for the paperclip to not hold paper together. If it snaps easily, that's (probably) not failing gracefully.

» Ask your interviewer—what are the expectations of it being used outside of the intended use case? Should we ensure that it has a minimum of usefulness for the other cases?

» What "stress" conditions might your paperclip be used in? *Answer:* hot weather, cold weather, frequent re-use, etc.

Chapter 11 | Testing

11.1 Find the mistake(s) in the following code:

```
unsigned int i;
for (i = 100; i <= 0; --i)
    printf("%d\n", i);
```

pg 209

11.2 You are given the source to an application which crashes when it is run. After running it ten times in a debugger, you find it never crashes in the same place. The application is single threaded, and uses only the C standard library. What programming errors could be causing this crash? How would you test each one?

pg 210

11.3 We have the following method used in a chess game: boolean canMoveTo(int x, int y) x and y are the coordinates of the chess board and it returns whether or not the piece can move to that position. Explain how you would test this method.

pg 211

11.4 How would you load test a webpage without using any test tools?

pg 212

11.5 How would you test a pen?

pg 213

11.6 How would you test an ATM in a distributed banking system?

pg 214

Chapter 12 | System Design and Memory Limits

How to Approach:

Don't be scared by these types of questions. Unless you claim to know how to design large systems, your interviewer probably won't expect you to know this stuff automatically. They just want to see how you tackle these problems.

General Approach

The general approach is as follows: Imagine we're designing a hypothetical system X for millions of items (users, files, megabytes, etc):

1. How would you solve it for a small number of items? Develop an algorithm for this case, which is often pretty straight-forward.
2. What happens when you try to implement that algorithm with millions of items? It's likely that you have run out of space on the computer. So, divide up the files across many computers.
   » How do you divide up data across many machines? That is, do the first 100 items appear on the same computer? Or all items with the same hash value mod 100?
   » About how many computers will you need? To estimate this, ask how big each item is and take a guess at how much space a typical computer has.
3. Now, fix the problems that occur when you are using many computers. Make sure to answer the following questions:
   » How does one machine know which machine it should access to look up data?
   » Can data get out of sync across computers? How do you handle that?
   » How can you minimize expensive reads across computers?

Example: Design a Web Crawler

1. Forget about the fact that you're dealing with billions of pages. How would you design this system if it were just a small number of pages? You should have an understanding of how you would solve the simple, small case in order to understand how you would solve the bigger case.
2. Now, think about the issues that occur with billions of pages. Most likely you can't fit the data on one machine. How will you divide it up? How will you figure out which computer has a particular piece of data?
3. You now have different pieces of data on different machines. What problems might that create? Can you try to solve them?

And remember, don't get scared! This is just an ordinary problem solving question.

Chapter 12 | System Design and Memory Limits

12.1 If you were integrating a feed of end of day stock price information (open, high, low, and closing price) for 5,000 companies, how would you do it? You are responsible for the development, rollout and ongoing monitoring and maintenance of the feed. Describe the different methods you considered and why you would recommend your approach. The feed is delivered once per trading day in a comma-separated format via an FTP site. The feed will be used by 1000 daily users in a web application.

pg 197

12.2 How would you design the data structures for a very large social network (Facebook, LinkedIn, etc)? Describe how you would design an algorithm to show the connection, or path, between two people (e.g., Me -> Bob -> Susan -> Jason -> You).

pg 199

12.3 Given an input file with four billion integers, provide an algorithm to generate an integer which is not contained in the file. Assume you have 1 GB of memory.

FOLLOW UP

What if you have only 10 MB of memory?

pg 202

12.4 You have an array with all the numbers from 1 to N, where N is at most 32,000. The array may have duplicate entries and you do not know what N is. With only 4KB of memory available, how would you print all duplicate elements in the array?

pg 205

12.5 If you were designing a web crawler, how would you avoid getting into infinite loops?

pg 206

12.6 You have a billion urls, where each is a huge page. How do you detect the duplicate documents?

pg 207

12.7 You have to design a database that can store terabytes of data. It should support efficient range queries. How would you do it?

pg 208

Part 3

Knowledge Based

Chapter 13 | C++

How To Approach:

A good interviewer won't demand that you code in a language you don't profess to know. Hopefully, if you're asked to code in C++, it's listed on your resume. If you don't remember all the APIs, don't worry—your interviewer probably doesn't care that much. We do recommend, however, studying up on basic C++ syntax.

Pointer Syntax

```
int *p; // Defines pointer.
p = &q; // Sets p to address of q.
v = *p; // Set v to value of q.
Foo *f = new Foo(); // Initializes f.
int k = f->x; // Sets k equal to the value of f's member variable.
```

C++ Class Syntax

```
class MyClass {
    private:
      double var;
    public:
      MyClass(double v) {var = v; }
      ~MyClass() {};
      double Update(double v);
};
double Complex::Update(double v) {
      var = v; return v;
}
```

C++ vs Java

A very common question in an interview is "describe the differences between C++ and Java." If you aren't comfortable with any of these concepts, we recommend reading up on them.

1. Java runs in a virtual machine.
2. C++ natively supports unsigned arithmetic.
3. In Java, parameters are always passed by value (or, with objects, their references are passed by value). In C++, parameters can be passed by value, pointer, or by reference.
4. Java has built-in garbage collection.
5. C++ allows operator overloading.
6. C++ allows multiple inheritance of classes.

Question: Which of these might be considered strengths or weaknesses of C++ or Java? Why? In what cases might you choose one language over the other?

Chapter 13 | C++

13.1 Write a method to print the last K lines of an input file using C++.

pg 215

13.2 Compare and contrast a hash table vs. an STL map. How is a hash table implemented? If the number of inputs is small, what data structure options can be used instead of a hash table?

pg 216

13.3 How do virtual functions work in C++?

pg 217

13.4 What is the difference between deep copy and shallow copy? Explain how you would use each.

pg 218

13.5 What is the significance of the keyword "volatile" in C?

pg 219

13.6 What is name hiding in C++?

pg 220

13.7 Why does a destructor in base class need to be declared virtual?

pg 221

13.8 Write a method that takes a pointer to a Node structure as a parameter and returns a complete copy of the passed-in data structure. The Node structure contains two pointers to other Node structures.

pg 223

13.9 Write a smart pointer (smart_ptr) class.

pg 224

Chapter 14 | Java

How to Approach:

While Java related questions are found throughout this book, this chapter deals with questions about the language and syntax. You generally will not find too many questions like this at the larger software companies (though they are sometimes asked), but these questions are very common at other companies.

What do you do when you don't know the answer?

If you don't know the answer to a question about the Java language, try to figure it out by doing the following: (1) Think about what other languages do. (2) Create an example of the scenario. (3) Ask yourself how you would handle the scenario if you were designing the language.

Your interviewer may be equally—or more—impressed if you can derive the answer than if you automatically knew it. Don't try to bluff though. Tell the interviewer, "I'm not sure I can recall the answer, but let me see if I can figure it out. Suppose we have this code…"

Classes & Interfaces (Example)

```
public static void main(String args[]) { … }
interface Foo {
    void abc();
}
class Foo extends Bar implements Foo { … }
```

final:

» Class: Can not be sub-classed

» Method: Can not be overridden.

» Variable: Can not be changed.

static:

» Method: Class method. Called with Foo.DoIt() instead of f.DoIt()

» Variable: Class variable. Has only one copy and is accessed through the class name.

abstract:

» Class: Contains abstract methods. Can not be instantiated.

» Interface: All interfaces are implicitly abstract. This modifier is optional.

» Method: Method without a body. Class must also be abstract.

Chapter 14 | Java

14.1 In terms of inheritance, what is the effect of keeping a constructor private?

pg 225

14.2 In Java, does the finally block gets executed if we insert a return statement inside the try block of a try-catch-finally?

pg 226

14.3 What is the difference between final, finally, and finalize?

pg 227

14.4 Explain the difference between templates in C++ and generics in Java.

pg 228

14.5 Explain what object reflection is in Java and why it is useful.

pg 229

14.6 Suppose you are using a map in your program, how would you count the number of times the program calls the put() and get() functions?

pg 230

Chapter 15 | Databases

How to Approach:

You could be asked about databases in a variety of ways: write a SQL query, design a database to hold certain data, or design a large database. We'll go through the latter two types here.

Small Database Design

Imagine you are asked to design a system to represent a large, multi-location, apartment rental company.

What are the key objects?

Property. Building. Apartment. Tenant. Manager.

How do they relate to each other?

Many-to-Many:

» A property could have multiple managers, and a manager could manage multiple properties.

One-to-Many:

» A building can only be part of one property.

» An apartment can only be part of one building.

What is the relationship between Tenant and Apartment? An apartment can obviously have multiple tenants. Can a tenant rent multiple apartments? It would be very unusual to, but this could actually happen (particularly if it's a national company). Talk to your interviewer about this. There is a trade-off between simplifying your database and designing it to be flexible. If you do assume that a Tenant can only rent one Apartment, what do you have to do if this situation occurs?

Large Database Design

When designing a large, scalable database, joins (which are required in the above examples), are generally very slow. Thus, you must *denormalize* your data. Think carefully about how data will be used—you'll probably need to duplicate it in multiple tables.

Chapter 15 | Databases

15.1 Write a method to find the number of employees in each department.

pg 231

15.2 What are the different types of joins? Please explain how they differ and why certain types are better in certain situations.

pg 232

15.3 What is denormalization? Explain the pros and cons.

pg 234

15.4 Draw an entity-relationship diagram for a database with companies, people, and professionals (people who work for companies).

pg 235

15.5 Imagine a simple database storing information for students' grades. Design what this database might look like, and provide a SQL query to return a list of the honor roll students (top 10%), sorted by their grade point average.

pg 236

Chapter 16 | Low Level

How to Approach:

Many candidates find low level problems to be some of the most challenging. Low level questions require a large amount of knowledge about the underlying architecture of a system. But just how much do you need to know? The answer to that depends, of course, on the company. At a typical large software company where you'd be working on desktop or web applications, you usually only need a minimum amount of knowledge. However, you should understand the concepts below very well, as many interview questions are based off this information.

Big vs Little Endian:

In big endian, the most significant byte is stored at the memory address location with the lowest address. This is akin to left-to-right reading order. Little endian is the reverse: the most significant byte is stored at the address with the highest address.

Stack (Memory)

When a function calls another function which calls another function, this memory goes onto the stack. An int (not a pointer to an int) that is created in a function is stored on the stack.

Heap (Memory)

When you allocate data with new() or malloc(), this data gets stored on the heap.

Malloc

Memory allocated using malloc is persistent—i.e., it will exist until either the programmer frees the memory or the program is terminated.

```
void *malloc(size_t sz)
```

Malloc takes as input sz bytes of memory and, if it is successful, returns a void pointer which indicates that it is a pointer to an unknown data type.

```
void free(void * p)
```

Free releases a block of memory previously allocated with malloc, calloc, or realloc.

Chapter 16 | Low Level

16.1 Explain the following terms: virtual memory, page fault, thrashing.

pg 237

16.2 What is a Branch Target buffer? Explain how it can be used in reducing bubble cycles in cases of branch misprediction.

pg 238

16.3 Describe direct memory access (DMA). Can a user level buffer / pointer be used by kernel or drivers?

pg 239

16.4 Write a step by step execution of things that happen after a user presses a key on the keyboard. Use as much detail as possible.

pg 237

16.5 Write a program to find whether a machine is big endian or little endian.

pg 241

16.6 Discuss how would you make sure that a process doesn't access an unauthorized part of the stack.

pg 242

16.7 What are the best practices to prevent reverse engineering of DLLs?

pg 244

16.8 A device boots with an empty FIFO queue. In the first 400 ns period after startup, and in each subsequent 400 ns period, a maximum of 80 words will be written to the queue. Each write takes 4 ns. A worker thread requires 3 ns to read a word, and 2 ns to process it before reading the next word. What is the shortest depth of the FIFO such that no data is lost?

pg 245

16.9 Write an aligned malloc & free function that takes number of bytes and aligned byte (which is always power of 2)

EXAMPLE

align_malloc (1000,128) will return a memory address that is a multiple of 128 and that points to memory of size 1000 bytes.

aligned_free() will free memory allocated by align_malloc.

pg 247

16.10 Write a function called my2DAlloc which allocates a two dimensional array. Minimize the number of calls to malloc and make sure that the memory is accessible by the notation arr[i][j].

pg 248

Chapter 17 | Networking

How to Approach

While the big software houses probably won't ask you many detailed networking questions in general, some interviewers will attempt to assess your understanding of networking as far as it relates to software and system design. Thus, you should have an understanding of http post and get requests, tcp, etc.

For a more networking based company (Qualcomm, CISCO, etc), we recommend a more thorough understanding. A good way to study is to read the material below, and delve further into it on Wikipedia. When Wikipedia discusses a concept that you are unfamiliar with, click on the concept to read more.

OSI 7 Layer Model

Networking architecture can be divided into seven layers. Each layer provides services to the layer above it and receives services from the layer below it. The seven layers, from top to bottom, are:

OSI 7 Layer Model	
Level 7	Application Layer
Level 6	Presentation Layer
Level 5	Session Layer
Level 4	Transport Layer
Level 3	Network Layer
Level 2	Data Link Layer
Level 1	Physical Layer

For a networking focused interview, we suggest reviewing and understanding these concepts and their implications in detail.

Chapter 17 | Networking

17.1 Explain what happens, step by step, after you type a URL into a browser. Use as much detail as possible.

pg 249

17.2 Explain any common routing protocol in detail. For example: BGP, OSPF, RIP.

pg 250

17.3 Compare and contrast the IPv4 and IPv6 protocols.

pg 252

17.4 What is a network / subnet mask? Explain how host A sends a message / packet to host B when: (a) both are on same network and (b) both are on different networks. Explain which layer makes the routing decision and how.

pg 254

17.5 What are the differences between TCP and UDP? Explain how TCP handles reliable delivery (explain ACK mechanism), flow control (explain TCP sender's / receiver's window) and congestion control.

pg 255

Chapter 18 | Threads and Locks

How to Approach:

In a Microsoft, Google or Amazon interview, it's not terribly common to be asked to implement an algorithm with threads (unless you're working in a team for which this is a particularly important skill). It is, however, relatively common for interviewers at any company to assess your general understanding of threads, particularly your understanding of deadlocks

Deadlock Conditions

In order for a deadlock to occur, you must have the following four conditions met:

1. Mutual Exclusion: Only one process can use a resource at a given time.
2. Hold and Wait: Processes already holding a resource can request new ones.
3. No Preemption: One process cannot forcibly remove another process' resource.
4. Circular Wait: Two or more processes form a circular chain where each process is waiting on another resource in the chain.

Deadlock Prevention

Deadlock prevention essentially entails removing one of the above conditions, but many of these conditions are difficult to satisfy. For instance, removing #1 is difficult because many resources can only be used by one process at a time (printers, etc). Most deadlock prevention algorithms focus on avoiding condition #4: circular wait.

If you aren't familiar with these concepts, please read http://en.wikipedia.org/wiki/Deadlock.

A Simple Java Thread

```
class Foo implements Runnable {
   public void run() {
      while (true) beep();
   }
}
Foo foo = new Foo ();
Thread myThread = new Thread(foo);
myThread.start();
```

Chapter 18 | Threads and Locks

18.1 What's the difference between a thread and a process?

pg 257

18.2 How can you measure the time spent in a context switch?

pg 258

18.3 Implement a singleton design pattern as a template such that, for any given class Foo, you can call Singleton::instance() and get a pointer to an instance of a singleton of type Foo. Assume the existence of a class Lock which has acquire() and release() methods. How could you make your implementation thread safe and exception safe?

pg 259

18.4 Design a class which provides a lock only if there are no possible deadlocks.

pg 261

18.5 Suppose we have the following code:

```
class Foo {
public:
    A(.....); /* If A is called, a new thread will be created and
               * the corresponding function will be executed. */
    B(.....); /* same as above */
    C(.....); /* same as above */
}
Foo f;
f.A(.....);
f.B(.....);
f.C(.....);
```

i) Can you design a mechanism to make sure that B is executed after A, and C is executed after B?

iii) Suppose we have the following code to use class Foo. We do not know how the threads will be scheduled in the OS.

```
Foo f;
f.A(.....); f.B(.....); f.C(.....);
f.A(.....); f.B(.....); f.C(.....);
```

Can you design a mechanism to make sure that all the methods will be executed in sequence?

pg 262

18.6 You are given a class with synchronized method A, and a normal method C. If you have two threads in one instance of a program, can they call A at the same time? Can they call A and C at the same time?

pg 264

Part 4
Additional Review Problems

Chapter 19 | Moderate

19.1 Write a function to swap a number in place without temporary variables.

pg 265

19.2 Design an algorithm to figure out if someone has won in a game of tic-tac-toe.

pg 266

19.3 Write an algorithm which computes the number of trailing zeros in n factorial.

pg 268

19.4 Write a method which finds the maximum of two numbers. You should not use if-else or any other comparison operator.

EXAMPLE

Input: 5, 10

Output: 10

pg 269

19.5 The Game of Master Mind is played as follows:

The computer has four slots containing balls that are red (R), yellow (Y), green (G) or blue (B). For example, the computer might have RGGB (e.g., Slot #1 is red, Slots #2 and #3 are green, Slot #4 is blue).

You, the user, are trying to guess the solution. You might, for example, guess YRGB.

When you guess the correct color for the correct slot, you get a "hit". If you guess a color that exists but is in the wrong slot, you get a "pseudo-hit". For example, the guess YRGB has 2 hits and one pseudo hit.

For each guess, you are told the number of hits and pseudo-hits.

Write a method that, given a guess and a solution, returns the number of hits and pseudo hits.

pg 270

19.6 Given an integer between 0 and 999,999, print an English phrase that describes the integer (eg, "One Thousand, Two Hundred and Thirty Four").

pg 271

19.7 You are given an array of integers (both positive and negative). Find the continuous sequence with the largest sum. Return the sum.

EXAMPLE

Input: {2, -8, 3, -2, 4, -10}

Output: 5 (i.e., {3, -2, 4})

pg 273

19.8 Design a method to find the frequency of occurrences of any given word in a book.

pg 273

19.9 Since XML is very verbose, you are given a way of encoding it where each tag gets mapped to a pre-defined integer value. The language/grammar is as follows:

```
Element --> Element Attr* END Element END [aka, encode the element
    tag, then its attributes, then tack on an END character, then
    encode its children, then another end tag]
Attr --> Tag Value [assume all values are strings]
END --> 01
Tag --> some predefined mapping to int
Value --> string value END
```

Write code to print the encoded version of an xml element (passed in as string).

FOLLOW UP

Is there anything else you could do to (in many cases) compress this even further?

pg 275

19.10 Write a method to generate a random number between 1 and 7, given a method that generates a random number between 1 and 5 (i.e., implement rand7() using rand5()).

pg 277

19.11 Design an algorithm to find all pairs of integers within an array which sum to a specified value.

pg 278

Chapter 20 | Hard

20.1 Write a function that adds two numbers. You should not use + or any arithmetic operators.

pg 279

20.2 Write a method to shuffle a deck of cards. It must be a perfect shuffle - in other words, each 52! permutations of the deck has to be equally likely. Assume that you are given a random number generator which is perfect.

pg 281

20.3 Write a method to randomly generate a set of m integers from an array of size n. Each element must have equal probability of being chosen.

pg 282

20.4 Write a method to count the number of 2s between 0 and n.

pg 283

20.5 You have a large text file containing words. Given any two words, find the shortest distance (in terms of number of words) between them in the file. Can you make the searching operation in O(1) time? What about the space complexity for your solution?

pg 285

20.6 Describe an algorithm to find the largest 1 million numbers in 1 billion numbers. Assume that the computer memory can hold all one billion numbers.

pg 286

20.7 Write a program to find the longest word made of other words in a list of words.

EXAMPLE

Input: test, tester, testertest, testing, testingtester

Output: testingtester

pg 287

20.8 Given a string s and an array of smaller strings T, design a method to search s for each small string in T.

pg 288

20.9 Numbers are randomly generated and passed to a method. Write a program to find and maintain the median value as new values are generated.

pg 290

20.10 Given two words of equal length that are in a dictionary, write a method to transform one word into another word by changing only one letter at a time. The new word you get in each step must be in the dictionary.

EXAMPLE

Input: DAMP, LIKE

Output: DAMP -> LAMP -> LIMP -> LIME -> LIKE

pg 291

20.11 Imagine you have a square matrix, where each cell is filled with either black or white. Design an algorithm to find the maximum subsquare such that all four borders are filled with black pixels.

pg 293

20.12 Given an NxN matrix of positive and negative integers, write code to find the sub-matrix with the largest possible sum.

pg 295

20.13 Given a dictionary of millions of words, give an algorithm to find the largest possible rectangle of letters such that every row forms a word (reading left to right) and every column forms a word (reading top to bottom).

pg 298

Each problem may have many 'optimal' solutions that differ in runtime, space, clarity, extensibility, etc. We have provided one (or more) optimal solutions. If you have additional solutions you would like to contribute, please contact us at http://www.xrl.us/ccbook or support@careercup.com.

We welcome all feedback and suggestions. Contact us at http://www.xrl.us/ccbook or support@careercup.com.