THE TECHNICAL INTERVIEW

ADVICE, INFORMATION, AND TYPICAL QUESTIONS

Dinan & Associates
www.dinanrecruiting.com

(650) 328-2790

THE TECHNICAL INTERVIEW

TABLE OF CONTENTS

These are interview questions pulled from on-line sources, primarily Amazon.com and Google interview questions. Most of my clients have similar processes, and you should be prepared for a challenging technical interview.

All of these questions can be found on line if you search for "Software interview questions" or "programming questions Google interview." Be prepared for a tough round of interviews, and challenging technical questions.

-Mark Dinan
Dinan & Associates

13 Great Pieces of Advice

http://steve-yegge.blogspot.com/2008/03/get-that-job-at-google.html

1. Study a data-structures and algorithms book.

Why? Because it is the most likely to help you beef up on problem identification. Many interviewers are happy when you understand the broad class of question they're asking without explanation. For instance, if they ask you about coloring U.S. states in different colors, you get major bonus points if you recognize it as a graph-coloring problem, even if you don't actually remember exactly how graph-coloring works.

And if you do remember how it works, then you can probably whip through the answer pretty quickly. So your best bet, interview-prep wise, is to practice the art of recognizing that certain problem classes are best solved with certain algorithms and data structures.

My absolute favorite for this kind of interview preparation is Steven Skiena's **The Algorithm Design Manual**. More than any other book it helped me understand just how astonishingly commonplace (and important) graph problems are – they should be part of every working programmer's toolkit. The book also covers basic data structures and sorting algorithms, which is a nice bonus. But the gold mine is the second half of the book, which is a sort of encyclopedia of 1-pagers on zillions of useful problems and various ways to solve them, without too much detail. Almost every 1-pager has a simple picture, making it easy to remember. This is a great way to learn how to identify hundreds of problem types.

2. Have a friend interview you.

The friend should ask you a random interview question, and you should go write it on the board. You should keep going until it is complete, no matter how tired or lazy you feel. Do this as much as you can possibly tolerate.

I didn't do these two types of preparation before my first Google interview, and I was absolutely shocked at how bad at whiteboard coding I had become since I had last interviewed seven years prior. It's hard! And I also had forgotten a bunch of algorithms and data structures that I used to know, or at least had heard of.

Going through these exercises for a week prepped me mightily for my second round of Google interviews, and I did way, way better. It made all the difference.

As for short-term preparation, all you can really do is make sure you are as alert and warmed up as possible. Don't go in cold. Solve a few problems and read through your study books. Drink some coffee: it actually helps you think faster, believe it or not. Make sure you spend at ***least*** an hour practicing immediately before you walk into the interview. Treat it like a sports game or a music recital, or heck, an exam: if you go in warmed up you'll give your best performance.

3. Algorithm Complexity

You need to know Big-O. It's a must. If you struggle with basic big-O complexity analysis, then you are almost guaranteed not to get hired. It's, like, one chapter in the beginning of one theory of computation book, so just go read it. You can do it.

4. Sorting

Know how to sort. Don't do bubble-sort. You should know the details of at least one n*log(n) sorting algorithm, preferably two (say, quicksort and merge sort). Merge sort can be highly useful in situations where quicksort is impractical, so take a look at it.

For God's sake, don't try sorting a linked list during the interview.

5. Hashtables

Hashtables are arguably the single most important data structure known to mankind. You ***absolutely have to know how they work***. Again, it's like one chapter in one data structures book, so just go read about them. You should be able to implement one using only arrays in your favorite language, in about the space of one interview.

6. Trees

You should know about trees. I'm tellin' ya: this is basic stuff, and it's embarrassing to bring it up, but some of you out there don't know basic tree construction, traversal and manipulation algorithms. You should be familiar with binary trees, n-ary trees, and trie-trees at the very ***very*** least. Trees are probably the best source of practice problems for your long-term warmup exercises.

You should be familiar with at least one flavor of balanced binary tree, whether it's a red/black tree, a splay tree or an AVL tree. You should actually know how it's implemented.

You should know about tree traversal algorithms: BFS and DFS, and know the difference between inorder, postorder and preorder.

You might not use trees much day-to-day, but if so, it's because you're avoiding tree problems. You won't need to do that anymore once you know how they work. Study up!

7. Graphs

Graphs are, like, really ***really*** important. More than you think. Even if you already think they're important, it's probably more than you think.

There are three basic ways to represent a graph in memory (objects and pointers, matrix, and adjacency list), and you should familiarize yourself with each representation and its pros and cons.

You should know the basic graph traversal algorithms: breadth-first search and depth-first search. You should know their computational complexity, their tradeoffs, and how to implement them in real code.

You should try to study up on fancier algorithms, such as Dijkstra and A*, if you get a chance. They're really great for just about anything, from game programming to distributed computing to you name it. You should know them.

Whenever someone gives you a problem, ***think graphs***. They are the most fundamental and flexible way of representing any kind of a relationship, so it's about a 50-50 shot that any interesting design problem has a graph involved in it. Make absolutely sure you can't think of a way to solve it using graphs before moving on to other solution types. This tip is important!

8. Other data structures

You should study up on as many other data structures and algorithms as you can fit in that big noggin of yours. You should especially know about the most famous classes of NP-complete problems, such as traveling salesman and the knapsack problem, and be able to recognize them when an interviewer asks you them in disguise.

You should find out what NP-complete means.

Basically, hit that data structures book hard, and try to retain as much of it as you can, and you can't go wrong.

9. Math

Some interviewers ask basic discrete math questions. This is more prevalent at Google than at other places I've been, and I consider it a Good Thing, even though I'm not particularly good at discrete math. We're surrounded by counting problems, probability problems, and other Discrete Math 101 situations, and those innumerate among us blithely hack around them without knowing what we're doing.

Don't get mad if the interviewer asks math questions. Do your best. Your best will be a heck of a lot better if you spend some time before the interview refreshing your memory on (or teaching yourself) the essentials of combinatorics and probability. You should be familiar with n-choose-k problems and their ilk – the more the better.

I know, I know, you're short on time. But this tip can really help make the difference between a "we're not sure" and a "let's hire her". And it's actually not all that bad – discrete math doesn't use much of the high-school math you studied and forgot. It starts back with elementary-school math and builds up from there, so you can probably pick up what you need for interviews in a couple of days of intense study.

Sadly, I don't have a good recommendation for a Discrete Math book, so if you do, please mention it in the comments. Thanks.

10. Operating Systems

This is just a plug, from me, for you to know about processes, threads and concurrency issues. A lot of interviewers ask about that stuff, and it's pretty fundamental, so you should know it. Know about locks and mutexes and semaphores and monitors and how they work. Know about deadlock and livelock and how to avoid them. Know what resources a processes needs, and a thread needs, and how context switching works, and how it's initiated by the operating system and underlying hardware. Know a little about scheduling. The world is rapidly moving towards multi-core, and you'll be a dinosaur in a real hurry if you don't understand the fundamentals of "modern" (which is to say, "kinda broken") concurrency constructs.

The best, most practical book I've ever personally read on the subject is Doug Lea's **Concurrent Programming in Java**. It got me the most bang per page. There are obviously lots of other books on concurrency. I'd avoid the academic ones and focus on the practical stuff, since it's most likely to get asked in interviews.

12. Coding

You should know at least one programming language really well, and it should ***preferably*** be C++ or Java. C# is OK too, since it's pretty similar to Java. You will be expected to write some code in at least some of your interviews. You will be expected to know a fair amount of detail about your favorite programming language.

13. Other Stuff

Because of the rules I outlined above, it's still possible that you'll get Interviewer A, and none of the stuff you've studied from these tips will be directly useful (except being warmed up.) If so, just do your best. Worst case, you can always come back in 6-12 months, right? Might seem like a long time, but I assure you it will go by in a flash.

The stuff I've covered is actually mostly red-flags: stuff that really worries people if you don't know it. The discrete math is potentially optional, but somewhat risky if you don't know the first thing about it.

General Interview Questions

Personality Fit, Warm up and General Background

Tell us a little bit about yourself.

Why are you leaving your current job?

What would you like to do?

What do you like about your current/former job? What don't you like?

How would you rate your current/former management?

What motivates you?

Would you like to be the team leader or team member?

Tell me about a conflict at a previous job and how you resolved it.

(Summary personality item: Think to yourself, "Would I want to spend four hours driving in a car with this person I am intering?")

Software Engineering Skills

What is Object Oriented Design? What are the benefits and drawbacks?

What is the Agile software philosophy?
What is the Lean software philosophy?

Have you looked at "Domain Driven Design"?

What are the benefits of Dependency Injection?

What books have you read on software engineering that you thought were good?
What are the really important aspects of software development?

Tell me about your philosophy of database design. Database tools?

What are important aspects of GUI design?

What Object Relational Mapping tools have you used?

Tell me about the Model--Controller pattern and why it's important?

What is Test Driven Development and Design? Why is it important?

Describe some of the software patterns you have used?

How do you design scalable applications?

What is continuous integration?

What is the REST architecture pattern?

How would you design a solution to the following problem....

Testing Technical Questions

What software have you used for bug tracking and version control?

What do you use for unit testing? GUI testing?

Describe an interesting class you have designed.

Have you been doing code res? What was the format?

When and how do you optimize code?

Describe your favorite build environment.

Technical Questions for Web Developers

What is the difference between GET and POST in web forms? How do you decide which to use?

Tell me about HTTP?

What is Service Oriented Architecture? Advantages and Disadvantages?

What JavaScript libraries have you used?

What is the advantage of using CSS? What are some of the irritating limitations of CSS?

Passion for Software Development

What are some influential software books you've read lately?

What are your favorite technical web sites and blogs?

Are you doing any interesting personal web projects?

What do you like about software?

A History of Getting Jobs Done

Tell me about your last project.

Tell me about your part in the release of some important software.

Questions to ask prospective employers

What is your software process?

Tell me about your version control.

What is your build process?

Do you do continuous integration?

Tell me about your bug tracking system.

How is unit testing done? How is system testing done?

Tell me about your Quality Assurance.

How many hours per week do people usually work?

What motivates your employees?

Why is there an opening?

What are your distinct advantages in the marketplace?

How much training per year do you offer?

Do you have a Wiki to share developer knowledge?

Interview Experiences

“She was very calm and cool. Asked me about my favorite subject. Any project I did on that. A brief intro reqd on that. Then she went to technical part. Asked about hashtables. Asked to explain how would you implement one. Data structure and how and what hash function will do. Cross questioned a lot with logic proposed. Gave clues where implementation or way I think is wrong, gave a chance to re-think and correct myself (many times during whole inter on different questions). Problems with hashtables, how would I solve such a problem (gave me an example scenario,collision). Asked me to design and code an algorithm (actual code NOT pseudo code). Asked to optimize, calculate Big O and how and what each steps takes (again gave a clue when she thought i was wrong, in optimization. i corrected it quickly). Asked about some design patterns and in the end gave me a chance to ask some questions i had.”

“The first inter was the toughest. Coding questions involved max subset sum, linear time sorting (I did counting sort and coded both in java), phone no. extraction from a big log (I used bash and perl). Finally a problem was asked about querying on data streams and I walked through a probable answer. Importance was given to algorithms and complexity analysis but I was confident in that.. so it went well.”

The next inter was easy. The person was testing basic OOPS skills (polymorphism, method overriding, templates), class design, basic databases and programming. Coding questions were quick sort and binary search. At the end the interviewers threw in buzz questions like what size of int, its range of values and what is 2^8. Both interviewers were very supportive and seemed excited to talk about their projects there.”

The next day I got an email for scheduling a third phone inter. That was of moderate difficulty but with a senior person. Got quizzed on my projects, long-term goals and also Amazon ec2! Coding questions were max length palindrome and weighted BT traversal. The person tried to negate my complexity analysis... but it was a test! Overall, it went well... and I got an offer in about a week. The entire process was completed in less than three weeks.

“Be prepared for CS basics like difference between low level and high level languages, hash tables, binary search”

“Very to-the-point and very technical. Almost no fluffy questions. Algorithms, data structures, but fairly low level ones. Hash tables and BSTs are basically the limit. Nothing related to Dynamic Programming, Divide and Conquer or any algorithms that require even an inch of thought.”

“The first one was more friendly, he asked me about Design Patterns, Garbage Collection algorithms, Priority Queue data structure (which DS would you use if you had to implement pop and push methods).”

Technical Questions

In an array find pairs of numbers that add to a particular value

In a Fibonacci series give the sum of all even numbers.

Reverse a list in C++

How will you implement garbage collector in c++

Explain Depth First Search and Breadth First Search. Write a routine to traverse the nodes of a binary tree using BFS

How do you find out if a graph has a circular reference?

What is a Minimum Spanning tree?

Explain OO design by taking an example of designing a card game

Given an array of positive integers, print out all the numbers which are repeated an even number of times? Can you do this without using additional storage?

How to implement a hash-table?

What is factory?

Having an infinite supply of water and two containers, one for 3 liters and one for 5 liters, how would you measure 4 liters?

Write a function to search for a string within another string. Analyze its complexity, and propose optimizations.

Write a function to obtain a string with the binary representation of an integer

Write a prime number service in language of your choice. how you scale it.

Design an airline company in object-oriented way.

Compilation diffs b/w c++ and java/c#? Advantages of IL and JIT?

Advantage of factory pattern? Singleton class?

Mutable vs immutable types? Advantage of immutable types?

Details about Hashtable? Good hash function? HashTable time complexities? Binary search complexity?

Find the maximum subset sum in an array of numbers. Discuss complexity.

Search on a large incoming number stream. Discuss complexity.

Given a weighted binary tree, traverse and find the max weight efficiently.

Find the max ;length palindrome in an input string.

Extract phone numbers from a large number of text logs in a hierarchy of directories.

How would you implement a binary search algorithm for a tree-type data structure?

Given two linked lists, find out IF they intersect.

Consider a simple array. What is the time complexity to insert, search, delete an element?

Design a Parking Garage using Object oriented concepts.

In Linux, a folder consists of 10000 files and some files contain US phone numbers. What would you do to display the names of files containing US phone numbers?

How the browser acquires IP address of the website that you entered in the address bar?

Explain in details. Then after that how does the browser process the IP address to get the contents of the URL?

The client has complained about your website that it runs very slowly. How would you increase the speed of the website?

Write a method take a String object as parameter and return the reversed String

What is the difference between an object and a class

What is the difference between an interface and a virtual class

What is the Java collection framework

What is a List, what is an ArrayList?

A lot of typical questions about collection framework

How would you reverse a String.

Given an array of integers, all but one of which appears an even number of times, find the one integer which appears an odd number of times.

Write a program to reverse the words in a string in place. for eg;

Two stacks are given, one is full of numbers and other in empty, one integer variable is given, fill the 2nd stack with numbers in ascending order with space and time constraints.

Sorting algorithms as well as about dynamic programming.

What is the fast sorting algorithm and why?

How would you design a deck of card in an OO language?

Given array A of size n, generate array B of size n, such that: A[i] = B[i]/(Sum(A[0] ... A[n-1])

Write a program to reverse words in a string

Explain deadlock scenario in Oracle

Explain garbage collection algorithms

Explain the data structure you would use to implement pop() and push(Object, int) for a Priority Queue. **(*)**

Mention one design pattern used in your project. Give me something good and bad about it.

Specific Software Engineering Topics

Requirements

1. Can you name a number of non-functional (or quality) requirements?
2. What is your advice when a customer wants high performance, high usability and high security?
3. Can you name a number of different techniques for specifying requirements? What works best in which case?
4. What is requirements tracing? What is backward tracing vs. forward tracing?
5. Which tools do you like to use for keeping track of requirements?
6. How do you treat changing requirements? Are they good or bad? Why?
7. How do you search and find requirements? What are possible sources?
8. How do you prioritize requirements? Do you know different techniques?
9. Can you name the responsibilities of the user, the customer and the developer in the requirements process?
10. What do you do with requirements that are incomplete or incomprehensible?

Functional Design

1. What are metaphors used for in functional design? Can you name some successful examples?
2. How can you reduce the user's perception of waiting when some functions take a lot of time?
3. Which controls would you use when a user must select multiple items from a big list, in a minimal amount of space?
4. Can you name different measures to guarantee correctness of data entry?
5. Can you name different techniques for prototyping an application?
6. Can you name examples of how an application can anticipate user behavior?
7. Can you name different ways of designing access to a large and complex list of features?
8. How would you design editing twenty fields for a list of 10 items? And editing 3 fields for a list of 1000 items?
9. What is the problem of using different colors when highlighting pieces of a text?
10. Can you name some limitations of a web environment vs. a Windows environment?

Technical Design

1. What do low coupling and high cohesion mean? What does the principle of encapsulation mean?
2. How do you manage conflicts in a web application when different people are editing the same data?
3. Do you know about design patterns? Which design patterns have you used, and in what situations?
4. Do you know what a stateless business layer is? Where do long-running transactions fit into that picture?
5. What kinds of diagrams have you used in designing parts of an architecture, or a technical design?
6. Can you name the different tiers and responsibilities in an N-tier architecture?
7. Can you name different measures to guarantee correctness and robustness of data in an architecture?
8. Can you name any differences between object-oriented design and component-based design?
9. How would you model user authorization, user profiles and permissions in a database?
10. How would you model the animal kingdom (with species and their behavior) as a class system?

Construction

1. How do you make sure that your code can handle different kinds of error situations?
2. Can you explain what Test-Driven Development is? Can you name some principles of Extreme Programming?
3. What do you care about most when reing somebody else's code?
4. When do you use an abstract class and when do you use an interface?
5. Apart from the IDE, which other favorite tools do you use that you think are essential to you?
6. How do you make sure that your code is both safe and fast?
7. When do you use polymorphism and when do you use delegates?
8. When would you use a class with static members and when would you use a Singleton class?
9. Can you name examples of anticipating changing requirements in your code?
10. Can you describe the process you use for writing a piece of code, from requirements to delivery?

Algorithms

1. How do you find out if a number is a power of 2? And how do you know if it is an odd number?
2. How do you find the middle item in a linked list?
3. How would you change the format of all the phone numbers in 10,000 static html web pages?

4. Can you name an example of a recursive solution that you created?
5. Which is faster: finding an item in a hashtable or in a sorted list?
6. What is the last thing you learned about algorithms from a book, magazine or web site?
7. How would you write a function to reverse a string? And can you do that without a temporary string?
8. What type of language do you prefer for writing complex algorithms?
9. In an array with integers between 1 and 1,000,000 one value is in the array twice. How do you determine which one?
10. Do you know about the Traveling Salesman Problem?

Data Structures

1. How would you implement the structure of the London underground in a computer's memory?
2. How would you store the value of a color in a database, as efficiently as possible?
3. What is the difference between a queue and a stack?
4. What is the difference between storing data on the heap vs. on the stack?
5. How would you store a vector in N dimensions in a datatable?
6. What type of language do you prefer for writing complex data structures?
7. What is the number 21 in binary format? And in hex?
8. What is the last thing you learned about data structures from a book, magazine or web site?
9. How would you store the results of a soccer/football competition (with teams and scores) in an XML document?
10. Can you name some different text file formats for storing unicode characters?

Testing

1. Do you know what a regression test is? How do you verify that new changes have not broken existing features?
2. How can you implement unit testing when there are dependencies between a business layer and a data layer?
3. Which tools are essential to you for testing the quality of your code?
4. What types of problems have you encountered most often in your products after deployment?
5. Do you know what code coverage is? What types of code coverage are there?
6. Do you know the difference between functional testing and exploratory testing? How would you test a web site?
7. What is the difference between a test suite, a test case and a test plan? How would you organize testing?
8. What kind of tests would you include for a smoke test of an ecommerce web site?
9. What can you do reduce the chance that a customer finds things that he doesn't like during acceptance testing?
10. Can you tell me something that you have learned about testing and quality assurance in the last year?

Maintenance

1. What kinds of tools are important to you for monitoring a product during maintenance?
2. What is important when updating a product that is in production and is being used?
3. How do you find an error in a large file with code that you cannot step through?
4. How can you make sure that changes in code will not affect any other parts of the product?
5. How do you create technical documentation for your products?
6. What measures have you taken to make your software products more easily maintainable?
7. How can you debug a system in a production environment, while it is being used?
8. Do you know what load balancing is? Can you name different types of load balancing?
9. Can you name reasons why maintenance of software is the biggest/most expensive part of an application's life cycle?
10. What is the difference between re-engineering and reverse engineering?

Configuration Management

1. Do you know what a baseline is in configuration management? How do you freeze an important moment in a project?
2. Which items do you normally place under version control?
3. How can you make sure that team members know who changed what in a software project?
4. Do you know the differences between tags and branches? When do you use which?
5. How would you manage changes to technical documentation, like the architecture of a product?
6. Which tools do you need to manage the state of all digital information in a project? Which tools do you like best?
7. How do you deal with changes that a customer wants in a released product?
8. Are there differences in managing versions and releases?
9. What is the difference between managing changes in text files vs. managing changes in binary files?
10. How would you treat simultaneous development of multiple RfC's or increments and maintenance issues?

Project Management

1. How many of the three variables scope, time and cost can be fixed by the customer?
2. Who should make estimates for the effort of a project? Who is allowed to set the deadline?
3. Do you prefer minimization of the number of releases or minimization of the amount of work-in-progress?
4. Which kind of diagrams do you use to track progress in a project?
5. What is the difference between an iteration and an increment?
6. Can you explain the practice of risk management? How should risks be managed?

7. Do you prefer a work breakdown structure or a rolling wave planning?
8. What do you need to be able to determine if a project is on time and within budget?
9. Can you name some differences between DSDM, Prince2 and Scrum?
10. How do you agree on scope and time with the customer, when the customer wants too much?
11. Why are manhole covers round?
12. What is the difference between a mutex and a semaphore? Which one would you use to protect access to an increment operation?
13. A man pushed his car to a hotel and lost his fortune. What happened?
14. Explain the significance of "dead beef".
15. Write a C program which measures the the speed of a context switch on a UNIX/Linux system.
16. Given a function which produces a random integer in the range 1 to 5, write a function which produces a random integer in the range 1 to 7.
17. Describe the algorithm for a depth-first graph traversal.
18. Design a class library for writing card games.
19. You need to check that your friend, Bob, has your correct phone number, but you cannot ask him directly. You must write the question on a card which and give it to Eve who will take the card to Bob and return the answer to you. What must you write on the card, besides the question, to ensure Bob can encode the message so that Eve cannot read your phone number?
20. How are cookies passed in the HTTP protocol?
21. Design the SQL database tables for a car rental database.
22. Write a regular expression which matches a email address.
23. Write a function f(a, b) which takes two character string arguments and returns a string containing only the characters found in both strings in the order of a. Write a version which is order N-squared and one which is order N.
24. You are given the source to a application which is crashing when run. After running it 10 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?
25. Explain how congestion control works in the TCP protocol.
26. In Java, what is the difference between final, finally, and finalize?
27. What is multithreaded programming? What is a deadlock?
28. Write a function (with helper functions if needed) called to Excel that takes an excel column value (A,B,C,D…AA,AB,AC,… AAA..) and returns a corresponding integer value (A=1,B=2,… AA=26..).
29. You have a stream of infinite queries (ie: real time Google search queries that people are entering). Describe how you would go about finding a good estimate of 1000 samples from this never ending set of data and then write code for it.
30. Tree search algorithms. Write BFS and DFS code, explain run time and space requirements. Modify the code to handle trees with weighted edges and loops with BFS and DFS, make the code print out path to goal state.

31. You are given a list of numbers. When you reach the end of the list you will come back to the beginning of the list (a circular list). Write the most efficient algorithm to find the minimum # in this list. Find any given # in the list. The numbers in the list are always increasing but you don't know where the circular list begins, ie: 38, 40, 55, 89, 6, 13, 20, 23, 36.
32. Describe the data structure that is used to manage memory. (stack)
33. What's the difference between local and global variables?
34. If you have 1 million integers, how would you sort them efficiently? (modify a specific sorting algorithm to solve this)
35. In Java, what is the difference between static, final, and const. (if you don't know Java they will ask something similar for C or C++).
36. Talk about your class projects or work projects (pick something easy)… then describe how you could make them more efficient (in terms of algorithms).
37. Suppose you have an NxN matrix of positive and negative integers. Write some code that finds the sub-matrix with the maximum sum of its elements.
38. Write some code to reverse a string.
39. Implement division (without using the divide operator, obviously).
40. Write some code to find all permutations of the letters in a particular string.
41. What method would you use to look up a word in a dictionary?
42. Imagine you have a closet full of shirts. It's very hard to find a shirt. So what can you do to organize your shirts for easy retrieval?
43. You have eight balls all of the same size. 7 of them weigh the same, and one of them weighs slightly more. How can you fine the ball that is heavier by using a balance and only two weighings?
44. What is the C-language command for opening a connection with a foreign host over the internet?
45. Design and describe a system/application that will most efficiently produce a report of the top 1 million Google search requests. These are the particulars: 1) You are given 12 servers to work with. They are all dual-processor machines with 4Gb of RAM, 4x400GB hard drives and networked together.(Basically, nothing more than high-end PC's) 2) The log data has already been cleaned for you. It consists of 100 Billion log lines, broken down into 12 320 GB files of 40-byte search terms per line. 3) You can use only custom written applications or available free open-source software.
46. There is an array A[N] of N numbers. You have to compose an array Output[N] such that Output[i] will be equal to multiplication of all the elements of A[N] except A[i]. For example Output[0] will be multiplication of A[1] to A[N-1] and Output[1] will be multiplication of A[0] and from A[2] to A[N-1]. Solve it without division operator and in O(n).
47. There is a linked list of numbers of length N. N is very large and you don't know N. You have to write a function that will return k random numbers from the list. Numbers should be completely random. Hint: 1. Use random function rand() (returns a number between 0 and 1) and irand() (return either 0 or 1) 2. It should be done in O(n).

48. Find or determine non existence of a number in a sorted list of N numbers where the numbers range over M, M>> N and N large enough to span multiple disks. Algorithm to beat O(log n) bonus points for constant time algorithm.
49. You are given a game of Tic Tac Toe. You have to write a function in which you pass the whole game and name of a player. The function will return whether the player has won the game or not. First you to decide which data structure you will use for the game. You need to tell the algorithm first and then need to write the code. Note: Some position may be blank in the game So your data structure should consider this condition also.
50. You are given an array [a1 To an] and we have to construct another array [b1 To bn] where bi = a1*a2*...*an/ai. you are allowed to use only constant space and the time complexity is O(n). No divisions are allowed.
51. How do you put a Binary Search Tree in an array in a efficient manner. Hint :: If the node is stored at the ith position and its children are at 2i and 2i+1(I mean level order wise)Its not the most efficient way.
52. How do you find out the fifth maximum element in an Binary Search Tree in efficient manner. Note: You should not use use any extra space. i.e sorting Binary Search Tree and storing the results in an array and listing out the fifth element.
53. Given a Data Structure having first n integers and next n chars. A = i1 i2 i3 ... iN c1 c2 c3 ... cN.Write an in-place algorithm to rearrange the elements of the array ass A = i1 c1 i2 c2 ... in cn
54. Given two sequences of items, find the items whose absolute number increases or decreases the most when comparing one sequence with the other by reading the sequence only once.
55. Given That One of the strings is very very long , and the other one could be of various sizes. Windowing will result in O(N+M) solution but could it be better? May be NlogM or even better?
56. How many lines can be drawn in a 2D plane such that they are equidistant from 3 non-collinear points?
57. Let's say you have to construct Google maps from scratch and guide a person standing on Gateway of India (Mumbai) to India Gate(Delhi). How do you do the same?
58. Given that you have one string of length N and M small strings of length L. How do you efficiently find the occurrence of each small string in the larger one?
59. Given a binary tree, programmatically you need to prove it is a binary search tree.
60. You are given a small sorted list of numbers, and a very very long sorted list of numbers - so long that it had to be put on a disk in different blocks. How would you find those short list numbers in the bigger one?
61. Suppose you have given N companies, and we want to eventually merge them into one big company. How many ways are theres to merge?
62. Given a file of 4 billion 32-bit integers, how to find one that appears at least twice?
63. Write a program for displaying the ten most frequent words in a file such that your program should be efficient in all complexity measures.
64. Design a stack. We want to push, pop, and also, retrieve the minimum element in constant time.

65. Given a set of coin denominators, find the minimum number of coins to give a certain amount of change.
66. Given an array, i) find the longest continuous increasing subsequence. ii) find the longest increasing subsequence.
67. Suppose we have N companies, and we want to eventually merge them into one big company. How many ways are there to merge?
68. Write a function to find the middle node of a single link list.
69. Given two binary trees, write a compare function to check if they are equal or not. Being equal means that they have the same value and same structure.
70. Implement put/get methods of a fixed size cache with LRU replacement algorithm.
71. You are given with three sorted arrays (in ascending order), you are required to find a triplet (one element from each array) such that distance is minimum.
72. Distance is defined like this : If a[i], b[j] and c[k] are three elements then distance=max(abs(a[i]-b[j]),abs(a[i]-c[k]),abs(b[j]-c[k]))" Please give a solution in O(n) time complexity
73. How does C++ deal with constructors and deconstructors of a class and its child class?
74. Write a function that flips the bits inside a byte (either in C++ or Java). Write an algorithm that take a list of n words, and an integer m, and retrieves the mth most frequent word in that list.
75. What's 2 to the power of 64?
76. Given that you have one string of length N and M small strings of length L. How do you efficiently find the occurrence of each small string in the larger one?
77. How do you find out the fifth maximum element in an Binary Search Tree in efficient manner.
78. Suppose we have N companies, and we want to eventually merge them into one big company. How many ways are there to merge?
79. There is linked list of millions of node and you do not know the length of it. Write a function which will return a random number from the list.
80. You need to check that your friend, Bob, has your correct phone number, but you cannot ask him directly. You must write a the question on a card which and give it to Eve who will take the card to Bob and return the answer to you. What must you write on the card, besides the question, to ensure Bob can encode the message so that Eve cannot read your phone number?
81. How long it would take to sort 1 trillion numbers? Come up with a good estimate.
82. Order the functions in order of their asymptotic performance: 1) 2^n 2) n^100 3) n! 4) n^n
83. There are some data represented by(x,y,z). Now we want to find the Kth least data. We say (x1, y1, z1) > (x2, y2, z2) when value(x1, y1, z1) > value(x2, y2, z2) where value(x,y,z) = (2^x)*(3^y)*(5^z). Now we can not get it by calculating value(x,y,z) or through other indirect calculations as lg(value(x,y,z)). How to solve it?
84. How many degrees are there in the angle between the hour and minute hands of a clock when the time is a quarter past three?

85. Given an array whose elements are sorted, return the index of a the first occurrence of a specific integer. Do this in sub-linear time. I.e. do not just go through each element searching for that element.
86. Given two linked lists, return the intersection of the two lists: i.e. return a list containing only the elements that occur in both of the input lists.
87. What's the difference between a hashtable and a hashmap?
88. If a person dials a sequence of numbers on the telephone, what possible words/strings can be formed from the letters associated with those numbers?
89. How would you reverse the image on an n by n matrix where each pixel is represented by a bit?
90. Create a fast cached storage mechanism that, given a limitation on the amount of cache memory, will ensure that only the least recently used items are discarded when the cache memory is reached when inserting a new item. It supports 2 functions: String get(T t) and void put(String k, T t).
91. Create a cost model that allows Google to make purchasing decisions on to compare the cost of purchasing more RAM memory for their servers vs. buying more disk space.
92. Design an algorithm to play a game of Frogger and then code the solution. The object of the game is to direct a frog to avoid cars while crossing a busy road. You may represent a road lane via an array. Generalize the solution for an N-lane road.
93. What sort would you use if you had a large data set on disk and a small amount of ram to work with?
94. What sort would you use if you required tight max time bounds and wanted highly regular performance.
95. How would you store 1 million phone numbers?
96. Design a 2D dungeon crawling game. It must allow for various items in the maze - walls, objects, and computer-controlled characters. (The focus was on the class structures, and how to optimize the experience for the user as s/he travels through the dungeon.)
97. What is the size of the C structure below on a 32-bit system? On a 64-bit?

```
struct foo {
char a;
char* b;
};
```

Linked Lists

This is an extremely popular topic. I've had linked lists on every inter.

You must be able to produce simple clean linked list implementations quickly.

- Implement `Insert` and `Delete` for
  - singly-linked linked list
  - sorted linked list
  - circular linked list

```
int Insert(node** head, int data)
int Delete(node** head, int deleteMe)
```

- Split a linked list given a pivot value
  `void Split(node* head, int pivot, node** lt, node** gt)`
- Find if a linked list has a cycle in it. Now do it without marking nodes.
- Find the middle of a linked list. Now do it while only going through the list once. (same solution as finding cycles)

Strings

- Reverse words in a string (words are separated by one or more spaces). Now do it in-place. **By far the most popular string question!**
- Reverse a string
- Strip whitespace from a string in-place
  `void StripWhitespace(char* szStr)`
- Remove duplicate chars from a string ("AAA BBB" -> "A B")
  `int RemoveDups(char* szStr)`
- Find the first non-repeating character in a string:("ABCA" -> B)
  `int FindFirstUnique(char* szStr)`
- *More Advanced Topics:*
  - You may be asked about using Unicode strings. What the interer is usually looking for is:
    - each character will be two bytes (so, for example, char lookup table you may have allocated needs to be expanded from 256 to 256 * 256 = 65536 elements)
    - that you would need to use wide char types (wchar_t instead of char)
    - that you would need to use wide string functions (like wprintf instead of printf)
  - Guarding against being passed invalid string pointers or non nul-terminated strings (using walking through a string and catching memory exceptions

Binary Trees

- Implement the following functions for a binary tree:
  - `Insert`
  - `PrintInOrder`
  - `PrintPreOrder`
  - `PrintPostOrder`
- Implement a non-recursive PrintInOrder

Arrays

- You are given an array with integers between 1 and 1,000,000. One integer is in the array twice. How can you determine which one? Can you think of a way to do it using little extra memory?
- You are given an array with integers between 1 and 1,000,000. One integer is missing. How can you determine which one? Can you think of a way to do it while iterating through the array only once. Is overflow a problem in the solution? Why not?
- Returns the largest sum of contiguous integers in the array

  Example: if the input is (-10, 2, 3, -2, 0, 5, -15), the largest sum is 8
  `int GetLargestContiguousSum(int* anData, int len)`
- Implement Shuffle given an array containing a deck of cards and the number of cards. Now make it O(n).
- Return the sum two largest integers in an array
  `int SumTwoLargest(int* anData, int size)`
- Sum n largest integers in an array of integers where every integer is between 0 and 9
  `int SumNLargest(int* anData, int size, int n)`

Queues

- Implement a Queue class in C++ (which data structure to use internally? why? how to notify of errors?)

Other

- Count the number of set bits in a byte/int32 (7 different solutions)
- Difference between heap and stack? Write a function to figure out if stack grows up or down.
- SQL query to select some rows out of a table (only because I had SQL on the resume)
- Open a file as securely as possible (assume the user is hostile -- list all the nasty things that could happen and checks you would have to do to)
- Implement a function to return a ratio from a double (ie 0.25 -> 1/4). The function will also take a tolerance so if toleran ce is .01 then FindRatio(.24, .01) -> 1/4
  `int FindRatio(double val, double tolerance, int& numerator, int& denominator)`

Puzzles

You have 2 supposedly unbreakable light bulbs and a 100-floor building. Using fewest possible drops, determine how much of an impact this type of light bulb can withstand. (i.e. it can withstand a drop from 17th floor, but breaks from the 18th).
Note that the ever-popular binary search will give you a worst case of 50 drops. You should be able to do it with under 20.

- There are n gas stations positioned along a circular road. Each has a limited supply of gas. You can only drive clockwise around the road. You start with zero gas. Knowing how much gas you need to get from each gas station to the next and how much gas you can get at each station, design an algorithm to find the gas station you need to start at to get all the way around the circle.
- Out of 10 coins, one weighs less then the others. You have a scale.
  - How can you determine which one weighs less in 3 weighs?
  - Now how would you do it if you didn't know if the odd coin weighs less or more?
- What is the next line in the following sequence:
  ```
  1
  11
  21
  Answer: it's 1211 and the next is 111221
  ```

Design Questions

- How would you design a server that has to process a fair number of good number of requests a second. What if you didn't know how many requests you'd be getting? What if requests had different priorities? (I always think of the Apache design for this question)
- Design malloc and free. (give up? see how G++ malloc works or this page for more examples)
- Design an elevator control system. Don't forget buttons on every floor and supporting multiple elevators. (What objects/methods/properties/how components communicate)
- Design a chess game (what objects? what methods? which data where? how will it work?)
- Design a deck of cards class (object/methods/data)
- How would you design the infrastructure front half of a very large web ecommerce site? What if it was very personalized? (How sessions are handled? where and what you can cache? how to load-balance?)

Concurrency

- Difference between Mutexes and Critical Sections?
- What are Reentrant Locks? Implement a Reentrant Lock with Mutexes.
- Implement a thread-safe class that will read/write to/from a buffer
  ```
  TSBuffer::TSBuffer(int size)
  int TSBuffer::Read(char* buff, int max_size)
  int TSBuffer::Write(char* buff, int size)
  ```

Windows-specific Questions

- What is the IUnknown COM interface?
- Synchronization primitives available in Windows (see MSDN documentation)

- Basic structure of a Win32 program (WinMain, Message processing)

Networking

- Difference between TCP and UDP? When would you want to use one over the other?
- How would approach guaging performance of webpages/parts on a very large website?
- Questions you are unlikely to get unless you claim a lot of IP experience
    - How does traceroute work?
    - How does path MTU discovery work?
    - How can one poison a BGP peer?