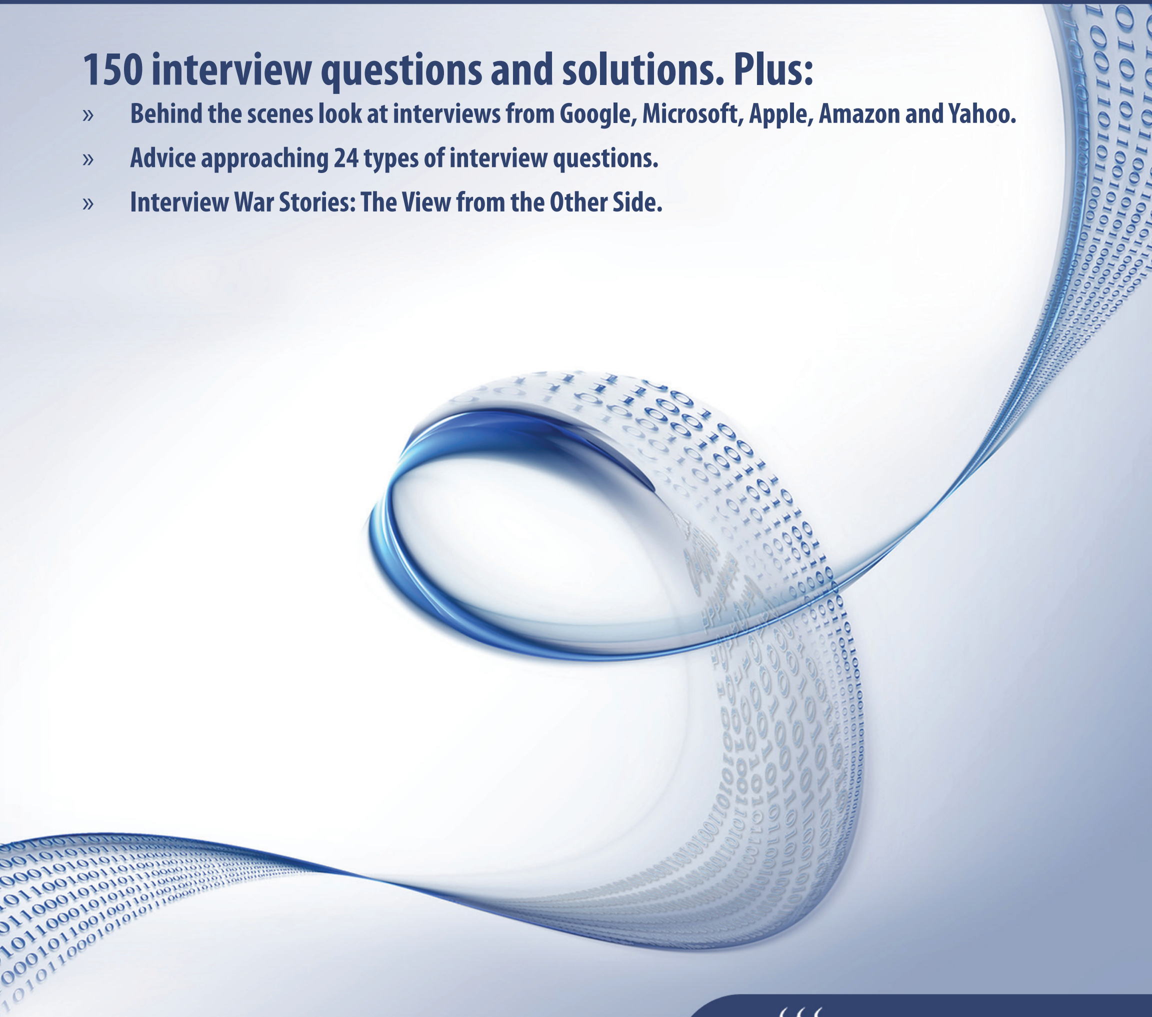
CRACKING THE TECHNICAL INTERVIEW

150 interview questions and solutions. Plus:

- » **Behind the scenes look at interviews from Google, Microsoft, Apple, Amazon and Yahoo.**
- » **Advice approaching 24 types of interview questions.**
- » **Interview War Stories: The View from the Other Side.**



CAREERCUP

Copyright Information

Cracking the Technical Interview will give you the interview preparation you need to get a job with a top technology company. This book provides:

» **150 Technical Interview Questions and Solutions**
questions are grouped by category and sorted by difficulty, allowing you to get the practice you need.

» **Behind the Scenes at the Microsoft, Google, Yahoo and Amazon interview**
Let us take you behind the scenes at four top tech companies and show you what your interviewers are up to.

» **Interview War Stories: The View from the Other Side**
These humorous but instructive stories from our interviewers show you how some candidates really flopped on the most important question - and how you can avoid doing the same.

» **Advice on Approaching 24 Types of Questions**
We break down our interview questions into 24 types, and provide advice on how to approach each type of question.

Gayle Laakmann founded CareerCup.com in 2005 to provide technical interview coaching for software engineers. She has worked as a Software Engineer for Microsoft, Google and Apple, interviewed over one hundred and fifty applicants and recruited many more.

She holds Bachelors and Masters degrees in Computer Science from the University of Pennsylvania.

Table of Contents

Introduction

Practice Interviews 6

The Microsoft Interview 7

The Amazon Interview 8

The Google Interview 9

The Apple Interview 10

The Yahoo Interview 11

Interview War Stories 12

Interview Questions 19

Chapter 1 Applied Mathematics 20

Chapter 2 Arrays & Hash Tables 22

Chapter 3 Bit Manipulation 24

Chapter 4 Brain Teasers 26

Chapter 5 C / C++ 28

Chapter 6 Counting and Combinatorics 30

Chapter 7 Database 32

Chapter 8 Debugging Existing Code 34

Chapter 9 Games 36

Chapter 10 Java 38

Chapter 11 Large Scale & Memory Limits 40

Chapter 12 Linked Lists 42

Chapter 13 Low Level 44

Table of Contents

Chapter 14 Matrix 46

Chapter 15 Networking 48

Chapter 16 Object Oriented Design 50

Chapter 17 Probability and Randomness 52

Chapter 18 Software and System Design 55

Chapter 19 Sorting and Searching 56

Chapter 20 Stacks and Queues 58

Chapter 21 Strings 60

Chapter 22 Testing 63

Chapter 23 Threads And Locks 65

Chapter 24 Trees and Graphs 67

Solutions 70

Practice Interviews

Studying helps, but nothing can prepare you like the real thing. Each CareerCup interviewer has given over a hundred interviews at Google, Microsoft, or Amazon. To nail your interview, sit down with a trained interviewer and get their experienced feedback.

See www.careercup.com/interview for more details.

One Hour Interview with Real Interviewers

Our interviewers will give you a real interview, just like you'd get at Google, Microsoft or Amazon. We'll test you on the same types of questions that they do. We'll grade you the same way they do. How can we do this? We've done over 100 interviews each for these companies. We've screened resumes. We've been part of their hiring committees. We know what they want.

We'll Also Give You...

» An .mp3 recording of your interview.

» Feedback on where you shined and where you struggled.

» Specific suggestions on how to improve.

» Instructions on how to approach tough problems

» Lessons on what interviewers look for in your code.

Schedule Your Interview Today!

See http://www.careercup.com/interview for pricing and details. Check out our special student rates!

A Typical Interview

A typical interview includes a brief discussion of your resume and one or more technical questions. Each interview will do coding via a shared document.

When the interview is completed, we'll give you immediate feedback on how you did while it's still fresh in your mind. Later that day, you'll receive an mp3 of the interview to refresh your memory.

The Microsoft Interview

Microsoft wants smart people. Geeks. People who are passionate about technology. You probably won't be tested on the ins and outs of C++ APIs, but you will be expected to write code on the board.

In a typical interview, you'll show up at Microsoft at some time in the morning and fill out initial paper work. You'll have a short interview with a recruiter where he or she will give you a sample question. Your recruiter is usually there to prep you, and not to grill you on technical questions. Be nice to your recruiter. Your recruiter can be your biggest advocate, even pushing to re-interview you if you stumbled on your first interview. They can fight for you to be hired - or not!

During the day, you'll do four or five interviews, often with two different teams. Unlike many companies, where you meet your interviewers in a conference room, you'll meet with your Microsoft interviewers in their office. This is a great time to look around and get a feel for the team culture.

Interviewers are not allowed to share their feedback on you with other interviewers, due to concerns of bias. Thus, a bad performance in one interview won't hurt you in the next. That being said, many sources indicate that some feedback is shared.

When you complete your interviews with a team, you might speak with a hiring manager. If so, that's a great sign! It likely means that you passed the interviews with a particular team. It's now down to the hiring manager's decision.

You might get a decision that day, or it might be a week. After one week of no word from HR, send them a friendly email asking for a status update.

Definitely Prepare:

"Why do you want to work for Microsoft?"

In this question, Microsoft wants to see that you're passionate about technology. A great answer might be, "I've been using Microsoft software as long as I can remember, and I'm really impressed at how Microsoft manages to create a product that is universally excellent. For example, I've been using Visual Studio recently to learn game programming, and it's APIs are excellent." Note how this shows a passion for technology!

What's Unique:

You'll only reach the hiring manager if you've done well, but if you do, that's a great sign!

The Amazon Interview

Amazon's recruiting process usually begins with one or two phone screens in which you interview with a specific team. The engineer who interviews you will usually ask you to write simple code and read it aloud on the phone. They will ask a broad set of questions to explore what areas of technology you're familiar with.

Next, you fly to Seattle for four or five interviews with one or two teams which have selected you based on your resume and phone interviews. You will have to code on a whiteboard, and some interviewers will stress other skills. Interviewers are each assigned a specific area to probe and may seem very different from each other. They can not see other feedback until they have submitted their own and they are discouraged from discussing it until the hiring meeting.

Amazon's "bar raiser" interviewer is charged with keeping the interview bar high. They attend special training and will interview candidates outside their group in order to balance out the group itself. If one interview seems significantly harder and different, that's most likely the bar raiser. This person has both significant experience with interviews and veto power in the hiring decision. You will meet with your recruiter at the end of the day.

Once your interviewers have entered their feedback, they will meet to discuss it. They will be the people making the hiring decision.

While Amazon's recruiters are excellent at following up with candidates, occasionally there are delays. If you haven't heard from Amazon within a week, we recommend a friendly email.

Definitely Prepare:

Amazon is a web-based company, and that means they care about scale. Make sure you prepare for questions in "Large Scale." You don't need a background in distributed systems to answer these questions. Just answer the question for one system and then think, how does your solution change with multiple computers?

Additionally, Amazon tends to ask a lot of questions that are based in mathematics and randomness.

What's Unique:

The Bar Raiser, who is brought in from a different team to keep the bar high.

The Google Interview

There are many scary stories floating around about Google interviews, but it's mostly just that: stories. The interview is not terribly different from Microsoft's or Amazon's. However, because Google HR can be a little disorganized, we recommend being proactive in communication.

A Google engineer performs the first phone screen, so expect tough technical questions. On your onsite interview, you'll interview with four to six people, one of whom will be a lunch interviewer. Interviewer feedback is kept confidential from the other interviewers, so you can be assured that you enter each interview with blank slate. Your lunch interviewer doesn't submit feedback, so this is a great opportunity to ask honest questions.

Written feedback is submitted to a hiring committee of engineers to make a hire/no-hire recommendation. Feedback is typically broken down into four categories (Analytical Ability, Coding, Experience and Communication) and you are given a score from 1.0 to 4.0 overall.

The hiring committee understands that you can't be expected to excel in every interview, but if multiple people raise the same red flag (arrogance, poor coding skills, etc), that can disqualify you. A hiring committee typically wants to see one interviewer who is an "enthusiastic endorser." In other words, a packet with scores of 3.6, 3.1, 3.1 and 2.6 is better than all 3.1's. Your phone screen is usually not a factor in the final decision.

The Google hiring process can be slow. If you don't hear back within one week, politely ask your recruiter for an update. A lack of response says nothing about your performance.

Definitely Prepare:

As a web-based company, Google cares about how to design a scalable system. So, make sure you prepare for questions from "Large Scale." Additionally, many Google interviewers will ask questions involving bit shifting and Bit Manipulation, so please brush up on these questions.

What's Different:

Your interviewers do not make the hiring decision. Rather, they enter feedback which is passed to a hiring committee. The hiring committee recommends a decision which can be—though rarely is—rejected by Google executives.

Apple Interview

Much like the company itself, Apple's interview process has minimal beaucracy. The interviewers will be looking for excellent technical skills, but a passion for the position and company is also very important. While it's not a prerequisite to be a Mac user, you should at least be familiar with the system.

The interview process typically begins with a recruiter phone screen to get a basic sense of your skills, followed up by a series of technical phone screens with team members.

Once you're invited on campus, you'll typically be greeted by the recruiter who provides an overview of the process. You will then have 6-8 interviews with members of the team for which you're interviewing, as well as key people with whom your team works.

You can expect a mix of 1-on-1 and 2-on-1 interviews. Be ready to code on a whiteboard and make sure all of your thoughts are clearly communicated. Lunch is with your potential future manager and appears more casual, but is still an interview. Each interviewer is usually focused on a different area and is discouraged from sharing feedback unless there's something they want subsequent interviewers to drill into.

Towards the end of the day, your interviewers will compare notes and if everyone still feels you're a viable candidate, you'll interview with the director and then VP of the organization you're applying to. While this decision is rather informal, it's a very good sign if you make it. This decision also happens behind the scenes and if you don't pass, you'll simply be escorted out of the building without ever having been the wiser (until now).

If you made it to the director and VP interviews, all of your interviewers will gather in a conference room to give an official thumbs up or thumbs down. The VP typically won't be present, but can still veto the hire if they weren't impressed. Your recruiter will usually follow up a few days later, but feel free to ping your recruiter for updates.

Definitely Prepare:

If you know what team you're interviewing with, make sure you read up on that product. What do you like about it? What would you improve?

What's Unique:

Apple does 2-on-1 interviews often, but don't get stressed out about them - it's the same as a 1-on-1 interview!

Also, Apple employees are huge Apple fans. You should show this same passion in your interview.

The Yahoo Interview

Resume Selection & Screening: While Yahoo tends to only recruit at the top 10 – 20 schools, other candidates can still get interviewed through Yahoo's job board (or – better yet – if they can get an internal referral). If you're one of the lucky ones selected, your interview process will start off with a phone screen. Your phone screen will be with a senior employee (tech lead, manager, etc).

Onsite Interview: You will typically interview with 6 – 7 people on the same team for 45 minutes each. Each interviewer will have an area of focus. For example, one interviewer might focus on databases, while another interviewer might focus on your understanding of computer architecture. Interviews will often be composed as follows:

5 minutes: General conversation. Tell me about yourself, your projects, etc.

20 minutes: Coding question. For example, implement merge sort.

20 minutes: System design. For example, design a large distributed cache. These questions will often focus on an area from your past experience or on something your interviewer is currently working on.

Decision: At the end of the day, you will likely meet with a Program Manager or someone else for a general conversation (product demos, concerns about the company, your competing offers, etc). Meanwhile, your interviewers will discuss your performance and attempt to come to a decision. The hiring manager has the ultimate say and will weigh the positive feedback against the negative.

If you have done well, you will often get a decision that day, but this is not always the case. There can be many reasons that you might not be told for several days – for example, the team may feel it needs to interview several other people.

Definitely Prepare:

Yahoo, almost as a rule, asks questions about system design, so make sure you prepare for that. They want to know that you can not only write code, but that you can design software. Don't worry if you don't have a background in this - you can still reason your way through it!

What's Unique:

Your phone interview will likely be performed by someone with more influence, such as a hiring manager.

Yahoo is also unusual in that it often gives a decision (if you're hired) on the same day. Your interviewers will discuss your performance while you meet with a final interviewer.

Interview War Stories

The View from the Other Side of the Front, by Peter Bailey

For the eager candidate getting ready for a big job interview, *Cracking the Technical Interview* is an invaluable reference, containing excellent coaching and practice material that gives you an inside edge on the interview process. However, as you go over your old data structures textbook and drill yourself with homemade discrete math flash cards, don't make the mistake of thinking of the interview as a kind of high-pressure game show – that if you just give all the right answers to the tech questions, you too can win a shiny new career (this week, on *Who Wants to be a Software Engineer?*)

While the technical questions on computer science obviously *are* very important, the *most* important interview question is not covered in this guidebook. In fact, it's often the single most important question in your interviewers' minds as they grill you in that little room. Despite the questions on polymorphism and heaps and virtual machines, the question they really want an answer to is ...

Would I have a beer with this guy?

Don't look at me like that, I'm serious! Well, I may be embellishing a little, but hear me out. The point I'm trying to make is that interviewers, especially those that you might work with, are probably just as anxious as you are. *Nonsense*, you say, as a nervous young professional, checking your pants for lint while you bite your fingernails, waiting for the interview team to show up in the front lobby. After all, this is the big leagues, and these guys are just waiting for you to slip up so they can rip you apart, laugh at your shriveled corpse, and grind your career dreams to dust beneath the heels of their boots.

Right? Just like pledge week, back in freshman year? Right? Hmmm?

Nothing could be further from the truth. The team of developers and managers interviewing you have their own tasks and projects waiting for them, back at their own desks. Believe me, they're hoping that every interview is going to be the last one. They'd rather be doing *anything* else. There might be a batch of upcoming projects looming on their calendar, and they need more manpower if they're going to even have a *prayer* of making their deadline. But the last guy the agency sent over was a complete flake who railed about Microsoft's evil for half an hour. And the one before that couldn't code his way out of a wet paper bag without using copy-and-paste. Sheesh, they think, where is HR *getting* these guys? How hard can it be to hire one lousy person?

While they may not literally be asking themselves "Would I have a beer with this guy (or gal)", they *are* looking to see how well you would fit in with the team, and how you would affect team chemistry. If they hire you, you're all going to be spending a lot of time together for the next few months or years, and they want to know that they can rely on you – and maybe even come to consider you a friend and colleague. They want to know that they can *depend* on you. And as tempting as it might be to them to just settle and hire the next person who comes along, they know better.

In many companies, particularly large U.S. companies, it's harder to fire somebody than it is to hire somebody. (Welcome to the US: Land of Lawsuits!) If they hire a dud, they're stuck with them. That person might be unproductive or, even worse, a drain on the team's productivity. So they keep interviewing, until they find the right person. They know that it's better to reject a good candidate than hire a bad one.

Some of those interviews are real doozies. Once you've interviewed long enough, you build up a repertoire of horror stories. War stories, of candidates who looked promising on paper until the interviews went terribly, terribly wrong. These war stories are not only humorous – they're also instructive.

Names have been changed to protect the innocent – or downright ridiculous.

Interview War Stories

Pop Divas Need Not Apply

Leonard was a very promising C++ coder, three years out of college, with a solid work history and an impressive skill set. He proved on the phone screen that he was above-average technically, and so he was invited in for an interview. We needed a savvy C++ person to work on a piece of middleware that interfaced with our database, and Leonard seemed like a sure fit.

However, once we started talking to him, things went south in a hurry. He spent most of the interview criticizing every tool and platform that we questioned him on. We used SQL Server as our database? Puhleease. We were planning to switch to Oracle soon, right? What's that? Our team used Tool A to do all our coding in? Unacceptable. He used Tool B, and only Tool B, and after he was hired, we'd all have to switch to Tool B. And we'd have to switch to Java, because he really wanted to work with Java, despite the fact that 75 percent of the codebase would have to be rewritten. We'd thank him later. And oh, by the way, he wouldn't be making any meetings before ten o'clock.

Needless to say, we encouraged Leonard to seek opportunities elsewhere. It wasn't that his ideas were bad – in fact, he was "technically" right about many things, and his (strong) opinions were all backed with solid fact and sound reason (except for the ten o'clock thing – we think he may have just been making a "power play".) But it was obvious that, if hired, Leonard wasn't going to play well with others – he would have been toxic kryptonite for team chemistry. He actually managed to offend two of the team members during the forty-five minutes of his interview. Leonard also made the mistake of assuming that Code Purity and Algorithm Beauty were always more important than a business deadline.

In the real world, there are always compromises to be made, and knowing how to work with the business analysts is just as important as knowing how to refactor a blob of code. If Leonard would not have gotten along with other IT people, he definitely wouldn't have gotten along with the business folks. Maybe you can get away with hiring a Leonard if he's one of the best ten coders in the world (he wasn't). But he was the classic failure example for the "Would you have a beer with this guy?" test.

Interview War Stories

What We Have Here is Failure to Communicate

Trisha was a mid-level Java developer with a solid history of middleware and JSP work on her resume. Since she was local, we invited her in for an interview without a phone screen. When we started asking her questions, it quickly became obvious that Trisha was a woman of few words. Her answers were short and often composed of "yes/no" responses, even to questions that were meant to start a dialog. Once she did start opening up, I still wasn't sure she was *actually* talking. I saw her lips moving, and heard mumbling sounds coming out, but it wasn't anything that sounded like English.

I'm not sure if Trisha was nervous or just shy, but either way, I had to ask her numerous times to repeat herself. Now I was the one getting nervous! I didn't want to be the guy who "ruined" the interview, so I pulled back on my questions. The other folks in the room and I exchanged uneasy glances. We felt like we were on a *Seinfeld* episode. It was almost impossible to understand Trisha, and when she did speak up, her halting, uncertain, confused speech patterns made us feel more like code breakers than interviewers. I am not exaggerating to say that I did not understand a single answer she gave during the interview.

Knowing, alone, isn't good enough. You're going to be talking with other technical people, and you're going to be talking to customers, and sales reps, and Betty from Marketing. You will write something eventually, whether it's documentation, or a project plan, or a requirements document. The word processor might correct your spelling, but it won't correct your lousy writing. The ability to communicate thoughts and ideas, in a clear, concise manner, is an absolutely invaluable skill that employers seek.

The same goes for verbal communication. I used to work with a co-worker who doubled the length of every meeting he was in, because he could not answer a question in less than ten minutes. "Hey, Dennis, what time is it?" "Well, that's kind of interesting, because I just happened to be reading an article on cesium clocks and leap seconds and the history of the Gregorian Calendar and ..."

I'll spare you the rest.

Interview War Stories

You Can Count on Me, Just Not Until Early Afternoon

Ahhh, 1999. The crest of the dot-com bubble, and the tightest labor market in history. Our company was racing to expand its development team, and we would have hired a German Shepherd if it knew HTML. Instead, we wound up hiring Ian. We should've hired the dog.

Ian was a cheerful, friendly guy who had a gift of natural charisma. He got along fantastically with all of the interviewers, and seemed very intelligent. Skillwise, he was adequate. He hadn't written a single line of computer code outside of his college courses, and didn't even have his own e-mail address. When we gave Ian the chance to ask us questions at the end of the interview, he asked about flexible work hours, and how soon he could take vacation time. Instead of showing an interest in the career opportunities, or in company's growth prospects, he asked whether he could take the all-you-could-drink break room soda home with him. The questions grew more bizarre from there.

Ian was very interested in our Legal Assistance benefit. He wanted to know if it covered the cost of filing lawsuits, if it covered him if he got sued himself, if it applied to any lawsuits he currently was involved in, and if he could "theoretically" use it to *sue the company itself.* He also asked us if he could use it to help him "fix" some unpaid speeding tickets.

In any other year, that should have been it for Ian right there. But, in 1999, we were hiring anybody who was even remotely competent. Ian collected paychecks from us for eighteen months, and he was about as productive as a traffic cone. He usually sauntered into the office around ten-thirty with some sort of lame excuse (by my count, he had to wait for the cable guy sixteen times in a six-month period). He usually killed the morning by answering e-mail and playing ping-pong, before breaking for a two-hour lunch. After lunch, it was more ping-pong, and maybe an hour of writing bad code, before bolting the office sometime around three. He was the dictionary definition of unreliable.

Remember, your potential future team members need to know that they can rely on you. And they need to know that you won't need constant supervision and hand-holding. They need to know that you're able to figure things out on your own. One of the most important messages that you, as a candidate, can convey in your interview is *hiring me will make your lives easier*. In fact, this is a large part of the reason for the famously difficult interview questions at places like Amazon and Google; if you can handle that kind of unpredictable pressure in an interview, then you stand a good chance of being useful to them on real projects.

To cite a more subtle example, once I was on a four person team that was desperately trying to recruit new members to help work on an old pile of software. It was a real mess; we'd inherited a nasty ball of spaghetti, and we needed people who could jump in, figure things out, and be part of the solution.

There was one very smart fellow, Terry, who would have been a great asset for our team –

but we didn't hire him, despite his excellent technical and personal skills. It was because he insisted on meticulous written instructions for every step of the coding process. He wasn't going to make a suggestion or take any initiative – or blow his nose, for that matter – without a mile-long audit trail and a dozen signatures. While he insisted that he worked that way for reasons of quality (a defensible point), we got the impression that it had more to do with butt-covering, and we simply didn't have the time for that kind of bureaucracy. Terry would have been an excellent fit in a government or aerospace IT department, something that required ISO 9000 procedures. But he would have never fit into our team; he would have been a burden, not an asset.

Interview War Stories

My Spider Senses are Tingling

I can think of lots of interviews that just fell into the general category of *weird and uncomfortable:*

» The Java coder who apparently considered hygiene optional, and had the interview room smelling like week-old blue cheese within ten minutes (my eyes were watering).

» The young fresh-out-of-college graduate with a tongue piercing that kept tick-tick-ticking against his teeth as he talked (after half an hour, it was like Chinese water torture).

» The girl who wore an iPod through her interview, with the volume turned loud enough that she actually had to ask the interviewers to repeat themselves a few times.

» The poor, hyper-nervous fellow who was sweating like a marathon runner for half an hour.

» The girl who wore a T-shirt with an obscene political slogan to her interview.

» The guy who asked (seriously) at the end of his interview, "So, are there any hot chicks in our department?"

Those are the interviews where we politely thank the people for their time, shake their hand (except for the sweaty guy), then turn to each other after the door closes and ask – *did that really just happen?*

Nobody is saying that you have to be a bland, boring robot in a Brooks Brothers suit and tie. Remember, the interview team wants you to be "the one", but they're also very worried about the possibility that you're going to be more of a distraction than an asset. Don't talk or behave in a way that will set off their early warning radar. Whether or not somebody bothers to behave professionally during an interview is often a very good indicator of what kind of teammate they're going to be.

Rudimentary social skills are part of the answer to "Would I have a beer with this guy?", or at least, "Will I mind working next to this guy for six months?" From the interviewer's point of view, they're picking a neighbor that they're going to live and work with 200 hours per week for foreseeable future. Would you really want a neighbor that smelled like a hog rendering plant?

Interview Questions

Study hard, practice and good luck!

How this Book is Organized

The interview questions in this book are grouped into categories, with a page preceding each category offering advice and other information. Within each category, the questions are sorted by approximate level of difficulty. Solutions for all questions are at the back.

How to Use this Book

An effective interview is not about memorizing interview questions, but rather, about applying an understanding of concepts and demonstrating your problem solving ability. Use these questions to find the gaps in your knowledge and to learn problem solving techniques that you can apply to new questions.

Advice for Devs

Your interview will be most likely not be conducted on a computer. Thus, when you practice the problems in this book, we recommend writing them down on paper first. Then, type your solution into the computer exactly as you wrote it and see how you did.

Suggestions and Corrections

While we do our best to ensure that all the solutions are correct, mistakes will be made. Moreover, sometimes there is no "right" answer. If you'd like to offer a suggestion or correction, please submit it at http://xrl.us/ccbook

Special Advice for SDETs

Not only do SDETs have to be great testers, but they also have to be great coders. Thus, we recommend that you complete the coding problems in this book with an eye for testing them. Even when the question doesn't specifically ask it, you should ask yourself, "how would I test this?" Remember: any problem can be an SDET problem.

Applied Mathematics CHAPTER-1

How to Approach:

Many of the so-called "Applied Mathematics" 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
```

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.

Prime Numbers

1. Every number can be written as a product of primes.
   Example: 504 = 2^3 * 3^2 * 7
2. f x is divisible by y, then every prime factor in y must be found in x.
   Example: If 504 is divisible by y, then y could be 168 (2^3 * 3 * 7), or 21 (3 * 7), or 4 (2^2), or many other things. Y could not, however, be equal to 10 (5 * 2), since 5 is not found in x

Applied Mathematics CHAPTER-1

1.1 Write a method to generate the nth Fibonacci number

pg 70

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

```
EXAMPLE
input: 35
output: 14 [list of 2's: 2, 12, 20, 21, 22, 23, 24, 25, 26, 27,
28, 29, 32]
```

pg 71

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

pg 75

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

pg 76

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

```
EXAMPLE
input: 11
output: 2 (11! = 39916800)
```

pg 77

1. 6 Write a function that adds two numbers. You should not use + or any arithmetic operators.

pg 78

1. 7 Write a method to implement *, - , / operations. You should use only the + operator.

pg 80

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

pg 82

1.9 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 bot-
tom: (56,90) (60,95) (65,100) (68,110) (70,150) (75,190)
```

pg 85

1.10 Given a two dimensional graph with 6000 points on it, find a line which passes the most number of points.

pg 87

Arrays & Hash Tables CHAPTER-2

How to Approach

While not all problem can be solved with a hash table, a shocking number of interview problems can be. Keeping track of which items you've already seen? Hash table. Needing a way to efficiently look up data? Hash table. The list goes on and on.

You should become very extremely comfortable with hash tables: how to implement them and how to use them.

Hash Tables

A hash table is a data structure that associates keys with values for O(1) lookup. Hash tables are frequently, though not always, implemented with an array. A simple implementation of a hash table that hashes a string to a Person is as follows:

```
class HashTable {
    Person[] data = new Person[MAX_HASH_KEY];
    int getId(string s) { /* return a key for this string */ };
    bool contains(string key) {
        int id = getId(key);
        if (data[id]) return true;
        return false;
    }
    void insert(string s, Person p) {
        data[getId(s)];
}
```

Note: This implementation does not handle collision. Collisions can be handled by "chaining" (eg, using a linked list), or a variety of other ways.

Vector (Dynamically Resizing Array):

A vector, 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 a long time (O(n)), but happens so rarely that its asymptotic time is still O(1).

Arrays CHAPTER-2

2.1 Suppose we have an array a1, a2, ..., an, b1, b2, ..., bn. Implement an algorithm to change this array to a1, b1, a2, b2, ..., an, bn.

pg 88

2.2 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 89

2.3 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 [ eg, {3, -2, 4} ]
```

pg 91

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

pg 92

2.5 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 94

Bit Manipulation CHAPTER-3

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?

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. 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".

```
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
```

Bit Manipulation CHAPTER-3

3.1 Write a function int BitSwapReqd(int A, int B) to determine the number of bits required to convert integer A to integer B.

EXAMPLE:

```
Input: 31, 14
Output: 2
```

pg 95

3.2 If you were to write a program to swap odd and even bits in integer, what is the minimum number of instructions required? (eg, bit 0 and bit 1 are swapped, bit 2 and bit 3 are swapped, etc).

EXAMPLE:

```
Input: 10001010
Output: 01000101
```

pg 96

3.3 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 97

3.4 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 98

3.5 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 (eg, 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 100

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

pg 101

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

pg 102

Brain Teasers CHAPTER-4

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.

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 15 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 - eg, half the rope length-wise might take only 2 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 min 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 min, start cooking the egg and light rope #2 at the other end. When Rope #2 burns up, your egg is done!

Brain Teasers CHAPTER-4

4.1 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 104

4.2 You have a 5 quart jug and a 3 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 105

4.3 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 106

4.4 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 (e.g., at least one). The hat is magical: it can be seen by other people, but not by the wearer of the hat himself. To remove the hat, you must dunk yourself 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 107

4.5 There are 100 closed lockers in a hallway. A man begins by opening all the 100 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 (eg, he toggles every third locker). After his 100th pass in the hallway, in which he toggles only locker number 100, how many lockers are open?

pg 109

C / C++ CHAPTER-5

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 either! 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.

Thought: 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?

C / C++ CHAPTER-5

5.1 What is the difference between a struct and a class? Where would you use each?

pg 110

5.2 Write a method to print the last ten lines of an input file using C.

pg 111

5.3 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 112

5.4 How do virtual functions work in C++?

pg 113

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

pg 114

5.6 In a class, the 'new' operator is used for allocating memory for new objects. Can this be done using malloc? If yes, how? If no, why not? Are there any restrictions associated with the use of malloc in place of new?

pg 115

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

pg 116

5.8 What is name hiding in C++?

pg 117

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

pg 118

5.10 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.

```
For example, the method signature could look like so:
Node* Copy(Node* root);
```

Note: Do not make any assumptions about the data structure – it could be a tree, linked list, graph etc.

Feel free to choose the language you are most comfortable with (C# or C++ are preferred) In addition to the function code, write a complete suite of unit tests for this problem.

pg 120

5.11 Write a smart pointer (smart_ptr) class.

Counting and Combinatorics CHAPTER-6

How to Approach:

While some problems in this chapter are simply coding problems, understanding the mathematical solution will help you "sanity check" your solution. For example, if you know how many subsets there are of a set, you can check to make sure that your algorithm to print all subsets will give you the correct number of subsets. When computing the number of ways of doing something, think about doing it step by step. The following two examples will illustrate this technique.

How many ways can you pick k elements from n elements, if order matters and elements are not replaced?

Eg - if we pick 5 different letters, "abcde" is considered to be different from "edcba".

The Approach:

We have n choices for the first draw. For the second draw, we only have n-1 since one is removed. Then n-2, …. When we draw k times, we get down to n-k+1 choices on the last draw. So, n * (n-1) * (n-2)*… * (n-k+1)

The Solution:

n! / (n-k)!

How many ways can you pick k elements from n elements, if order does not matter and elements are not replaced?

We're now just throwing letters into a bucket. Picking "a, b" is the same thing as "b, a".

The Approach:

If you compare this problem to the previous one, we've essentially double (or triple, quadruple, etc) counted items. That is, "abc", "acb", "bac", "bca", "cab", "cba" were all considered unique in the previous solution but now they're considered the same.

Just how many times has "abc" been included (in its other forms)? 3! times, since there are 3! ways of rearranging "abc". In fact, every item has been included 3! times, when it should have been included just once!

So, we divide our previous answer by 3! (or, more generally, k!).

We now get n! / (k! * (n-k)!).

This solution is often written as n-choose-k or nCk.

Counting and Combinatorics CHAPTER-6

6.1 In how many different ways can a cube be painted by using three different colors of paint?

pg 122

6.2 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 print all possible paths for the robot.

pg 123

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

pg 124

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

```
EXAMPLE:
input: 3 (eg, 3 pairs of parentheses)
output: ()()(), ()(()), (())(), ((()))
```

pg 125

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

pg 126

Database CHAPTER-7

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 school's registrar: course information, departments, course enrollment, teachers, etc.

What are the key objects?

Student. Professor. Course. Department.

How do they relate to each other?

***NOTE: I'm going to make some assumptions here for the purposes of writing up this explanation. In your interview, don't make assumptions! Ask your interviewer instead.**

Many-to-Many:

» A course can belong to multiple departments, and each department can have multiple courses. So, create a separate table DepartmentCourse that acts as a "pairing" of the two. DepartmentCourse has just a department_id field and a course_id field.

» A student can be in multiple courses, a course can have multiple students. So, do the same as above and create a StudentCourse table.

One-to-Many:

» A course only has one professor. A professor can teach multiple courses. So, add a field professor_id to the Courses table.

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.

Database CHAPTER-7

7.1 Write a method to find the number of employees in each department when we have the following tables:

```
Employees containing: Emp_ID, Emp_Name and Dept_ID (Primary key)
Departments containing: Dept_Name and Dept_ID (foreign key)
```

pg 127

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

pg 128

7.3 What is normalization? Explain the pros and cons.

pg 129

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

pg 130

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

pg 131

Debugging Existing Code CHAPTER-8

How to Approach

Sometimes, in an interview, an interviewer asks you to look at a piece of code and identify the mistakes. We recommend the following approach:

1. Examine the code and understand what it's expected to do. Ask your interviewer what types of data it's expected to handle, where it'll be used, etc.
2. Look for syntax errors: does everything type check? is the class declaration correct?
3. Look for "hot spots":
   » If you see float and doubles, check for precision errors
   » If you see division, check for rounding errors
   » If you see memory allocation, check for memory leaks
   » If you see unsigned ints, check to see if the int might ever be negative
   » If you see bit manipulation, check for correctness
4. Run through the code with a few examples:
   » The "normal" case
   » The boundary cases (null, 0, 1, MAX, etc)
5. Does it do everything it's expected to? For example, if the code is supposed to return everyone in a database under 21, does it only look for students? Maybe it should be looking for teachers as well?
6. Does it handle unexpected cases? What if it takes in a list and it has a loop?

Further Advice

1. Look at the space and time complexity—can you improve it?
2. Correct the mistakes in the code. But do so carefully! Don't simply make changes until it works. Deeply understand the issues and then correct them.

Debugging Existing Code CHAPTER-8

8.1 Explain what the following code does:

```
((n & (n-1)) == 0)
```

pg 132

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

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

pg 133

8.3 What problems do you see in this piece of code (explain without compilation):

```
template struct Foo : public Custom { };
template struct Foo {
    template struct rebind {
        typedef Foo Other;
    };
};
template struct Derived : public Base::template rebind >::Other {
    int foo() { printf(“here we are\n”); };
};
main() {
    typedef Foo typedef Derived Derived_inst;
    Derived_inst ii;
    ii.foo();
}
```

pg 134

Games CHAPTER-9

How To Approach:

Sometimes interviewers ask these problems simply because they're "fun," but often, it's also because "game" problems are heavy on object oriented design. Gaming problems tend to be more "free form" and thus they give you the chance to demonstrate how you really code.

Define Data Structures

When implementing something so free form, ask yourself, "Where can I define a class or a struct?" When in doubt, define a new class or struct. It tells the interviewer that you care about the maintainability of your code.

Validate Your Assumptions

Be careful about making assumptions. Suppose you're asked to implement the word game Scrabble. Don't assume that the dictionary will be in English—or even in that character set. It could be in any language! Ask your interviewer lots and lots of questions so that you know what to implement.

Are you solving this problem once, or many times?

Sometimes the solution will change depending on whether or not your code will be called multiple times. For example, suppose you're asked to find all anagrams of a word. If you're calling this code just once, it may be fastest just to rearrange the letters and check if they're in the dictionary. But, if you're going to call the code multiple times, it's now fastest to pre-compute data by iterating through all the words in the dictionary.

Can you generalize your code, or part of it?

If you really want to go above and beyond, try writing your code as though it were a more generalized case. For example, if you're trying to figure out if a tic-tac-toe board has a winner, you might suggest to your interviewer solving it for the more general NxN case. If you go down this path though, be warned—sometimes problems are trickier than they appear. Also, make sure you discuss this with your interviewer so he/she knows what you're doing.

Games CHAPTER-9

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

pg 136

9.2 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 (eg, Slot #1 is red, Slots #2 and #3 are green, #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 138

9.3 There is an 8x8 chess board in which two diagonally opposite corners have been cut off. You are given 31 dominos in which 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 139

9.4 Find a way to arrange 8 queens on a chess board so that none of them share the same row, column or diagonal.

pg 140

9.5 Othello is played as follows: Each Othello piece is white on one side and black on the other. On your turn, you place a piece on the board so that your color is facing up. You must pick a spot such that your opponent's pieces are either on the left and the right, or on the top and the bottom. All of your opponent's pieces on the line between two of yours are then turned over, to become yours. Your goal is to own the most pieces.

Design the game Othello. Write a method to check whether someone has won the game.

pg 142

Java CHAPTER-10

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 many questions like this at the larger software companies (Microsoft, Google, Amazon, etc), which tend to avoid 'trivia' based questions, but these questions are very common at many smaller 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 will likely 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 Fido {
    void abc();
}
class Foo extends Bar implements Fido { … }
```

final:

- » Class: Can not be subclassed
- » 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.

Java CHAPTER-10

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

pg 144

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

pg 145

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

pg 146

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

pg 147

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

pg 148

10.6 Explain the different ways to pass parameters to a function (by value, by reference, by pointer) for the following cases:

6.1 Basic data type (int, char etc)

6.2 Array of integers

6.3 An object of a struct

6.4 An object of a class

pg 149

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

pg 150

10.8 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 152

Large Scale & Memory Limits CHAPTER-11

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 generally 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 have occurred when you're on many computers. Make sure to answer the following questions:

» How does one machine know which machine it should access to look up other 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, go back to the issues of 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.

Large Scale & Memory Limits CHAPTER-11

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

pg 153

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

pg 154

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

pg 155

11.4 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 156

11.5 You have two very large binary trees: T1, with millions of nodes, and T2, with hundreds of nodes. The trees store character data and duplicates are allowed. Create an algorithm to decide if T2 is a subtree of T1.

pg 158

11.6 Find the largest 1 million numbers in 1 billion numbers. Assume that the computer memory can hold all one billion numbers.

pg 159

11.7 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 find out if a particular number exists in the array?

pg 160

11.8 Given a dictionary of millions of words, write a program 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 161

Linked Lists

CHAPTER-12

How to Approach:

Linked list questions are extremely common in an interview. These can range from the simplest (delete a node in a linked list) to much more challenging. Either way, we advise you to be extremely comfortable with the easiest question. 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 double linked list.

```
class Node {
    Node next = null;
    int data;
    public Node(int d) { d = data; }
    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

```
/* delete node with data d and return new head */
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 */
        }
    }
}
```

Linked Lists CHAPTER-12

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

pg 163

12.2 Write code to remove duplicates from an unsorted linked list.
FOLLOW UPS & COMPLICATIONS
How would you solve this problem if a temporary buffer is not allowed?

pg 165

12.3 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 167

12.4 Imagine you have an unbalanced binary search tree. Design an algorithm which creates a linked list of all the nodes at each depth (eg, if you have a tree with depth D, you'll have D linked lists).

pg 169

12.5 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 170

12.6 You have two numbers represented by a linked list, where each node contains a single digit. 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: 9 -> 0 -> 7
```

pg 171

Low Level CHAPTER-13

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, these go 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—eg, 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.

Low Level CHAPTER-13

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

pg 172

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

pg 173

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

pg 174

13.4 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 175

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

pg 176

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

pg 177

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

pg 179

13.8 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 180

13.9 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 183

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

pg 184

Matrix CHAPTER-14

How to Approach:

A strange thing happens when a candidate walks into an interview room. They inexplicably forget how to declare a double array. We provide the syntax below so that you can study it. When working through these problems, don't forget about how the size of the matrix affects the speed of an algorithm. That is, if you have an unsorted NxN matrix and you need to search it for an element, the speed will be O(N^2). If it is sorted, the speed might be O(log N) [why? Because O(log(N^2)) = O(2 log N) = O(log N)]. And, as always, be careful. It's very easy in these questions to introduce small mistakes by mixing up the row and column. So, instead of using m and n to describe variable names, use row and col, or at least r and c.

Java Syntax

```
int[] x = new int[4];
int[][] x = new int[2][3];
```

C / C++ Syntax

```
int x[2][3];
int x[2][3] = {{1, 2. 3}, {4, 5, 6}};
```

CAUTION: Are you designing a matrix during an interview?

If you find yourself designing a matrix (without the interviewer telling you to do so) during an interview, stop and think: is this really the best data structure to hold this data? Many times, it is better to design a custom data structure.

For example, suppose you are implementing code to handle a simple polynomial (eg, 7x^5 + 13x^2 + 9x + 17), you could do this using a matrix. In this case, the first spot in the array would hold the coefficient and the second spot would hold the exponent. That is, A[1][0] = 13 and A[1][1] = 2.

However, it would be far cleaner to design a struct to hold each term, and then to represent the polynomial as an array of structs.

```
struct PolyTerm {
    double coefficient;
    double exponent;
}
```

Matrix CHAPTER-14

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

pg 185

14.2 Given an image represented by a 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 186

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

pg 187

14.4 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 188

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

pg 189

Networking CHAPTER-15

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 concepts that you are unfamiliar with, click on them 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:

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

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

Networking CHAPTER-15

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

pg 191

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

pg 192

15.3 Compare and contrast the IPv4 and IPv6 protocols.

pg 194

15.4 What is network/subnet mask? Explain how a host A sends a message/packet to host B when:

a) both are on same network

b) both are on different networks

Explain which layer takes routing decision and how.

pg 196

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

pg 198

Object Oriented Design CHAPTER-16

How to Approach

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

CAUTION: 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 will 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 OOD 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 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 OOD. Suppose you are designing an OOD for a parking lot:

1. What are your goals? eg, 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 in a class called Permission which handles different payment systems. Permission will be subclassed 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.

Object Oriented Design CHAPTER-16

16.1 Imagine you have a call center as follows:

1. Call center has 3 levels of employees: fresher, technical lead (TL), product manager (PM). There can be multiple employees, but only one TL or PM.
2. An incoming telephone call must be allocated to a Fresher who is free.
3. If a fresher can not handle the call, he or she must escalate the call to technical lead.
4. If TL is not free or not able to handle, then the call should be escalated to PM.

   Design the classes and data structures for this problem. Implement a method getCallHandler().

pg 200

16.2 Design a musical juke box using object oriented principles.

pg 202

16.3 Design a chess game using object oriented principles.

pg 204

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

pg 206

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

pg 207

16.6 Implement a jigsaw puzzle in C++. Design the data structures and explain an algorithm to solve the puzzle.

pg 209

Probability And Randomness CHAPTER-17

How to Approach:

Questions dealing with randomness are relatively common in interviews (particularly at Amazon), so it's important to understand probability.

CAUTION: Make sure random is *uniformly random*

When an interviewer asks you to design an algorithm to, for example, generate a random subset from a set, they mean *uniformly* random. That is, each subset should be equally likely. Make sure that your algorithm ensured uniform randomness.

Independent Events:

Independent: A and B are independent if A happening tells you nothing about likelihood of the B happening. Example: A = {I wear a blue shirt on Monday} and B = {my cell phone breaks}. P(A and B) = P(A) * P(B)

How do you use this in an interview?

Example: You have a stream of random numbers throw numbers at you. At any point, the stream might stop and you will need to return a random number from the stream. Without holding the entire stream in memory, how would you do this?

1. Each number must be equally likely. So, if we see n numbers in the stream, the probability of returning any specific number must be 1/n.
2. If breaking on Item #1: Easy. Return item #1 with 100% (1/1) probability.
3. If breaking on Item #2: Still pretty easy. Let's track item #1 in a variable called res, and then flip a coin to set the value of res. If we have to return, we return whatever is in res.
4. If breaking on Item #3: This is where is gets tricky. We know that item #3 should be returned with 1/3 probability. We also know that, from the previous step, P(res = item2) = 1/2 and P(res = item1) = 1/2. Note that 1/2 * 2/3 = 1/3. This means that if we return whatever is in res with 2/3 probability and return item 3 with 1/3 probability, item1, item2 and item3 will be equally likely.
5. More generally: on element n, swap res with item[n] with probability 1/n, and keep res as-is with probability (n-1)/n. When asked to break, return res.

How do we know this works? Because the probability that res = item[i] at the nth swap is:

$$[1 / i] * [i / (i+1)] * [(i+1) / (i+2)] * [(i+2) / (i+3)] + ... [(n-1) / n]$$

If you cancel out terms, we get 1 / n. Thus, each element has a probability of 1 / n.

Probability And Randomness CHAPTER-17

17.1 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 211

17.2 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 212

17.3 Numbers are randomly generated and stored in an array. Write a program to find and maintain the median value as new values are generated.

pg 213

17.4 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 214

17.5 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 216

17.6 Write a method to generate a random number between 1 and 7, given a method that generates a random number between 1 and 5.

pg 217

Software and System Design CHAPTER-18

How to Approach:

Like object oriented design questions, Software and System Design questions aim to discover if a candidate can tackle a vague problem. They require a candidate to understand the real world application of a problem in order to create the requisite algorithms or data structures.

Step 1: Resolve Ambiguity

Interviewers are often specifically testing to see if you make assumptions. So, make sure when you get a question, make sure to ask about how exactly the software will be used. Perhaps, for example, the elevator that you're designing isn't just any elevator—it's an elevator for livestock. That changes things.

Step 2: Understand How It Will Be Used

Who are the users? What exactly will they be doing? If they are sending you data to process, how much data? And how often? Of what type? Is it possible to have malicious users? These are all questions that you need to know before designing your approach. Where applicable, try to develop some general guidelines. Is approximation OK, or must the answer be exact? Is it more important for the algorithm to be fast, or will there be significant memory restrictions as well? What are the sort of real world problems you might hit? Draw from your own experience!

Step 3: Consider the core entities and how they interact

Who are the key entities and what are their restrictions? For example, suppose you are designing a database for banks to hold customer data. Entities might include customers, bank accounts, bank employees, ATMs, employers (who deposit money), a website with online banking, etc. Each of these people has certain restrictions. A customer will care about ease of use through online banking. Online banking needs API access to the bank account. The bank employee does as well, but he might have different security restrictions.

Step 4: Design!

At this point, the approach we take depends a lot on what's asked. If you're asked for an algorithm, you need to consider time and space complexity . If you're asked for the object oriented design, you might consider the future flexibility of the system.

Software and System Design CHAPTER-18

18.1 Picture a computer screen with multiple windows and a mouse. Each window can be represented as a rectangle, and the mouse is represented by a (x, y) coordinate. If you click on the screen, the top most window should become active. Describe an algorithm to return the top most window when the mouse is clicked.

pg 218

18.2 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 219

18.3 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 221

18.4 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 223

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

pg 226

Sorting and Searching CHAPTER-19

How to Approach:

Understanding the common sorting algorithms is incredibly valuable, as many sorting or searching solutions are 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 increase 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're 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).

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).

Bucket Sort:

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

Sorting and Searching CHAPTER-19

19.1 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 229

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

pg 230

19.3 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.

```
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 231

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

pg 232

19.5 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 233

Stacks and Queues CHAPTER-20

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.

Implementing a Stack

```
class Stack {
    Node top;
    int data;
    Node pop() {
        if (!top) {
            return null;
        }
        Node t = top;
        top = top->next;
        return t;
    }
    void push(Node n) {
    if (n) {
        n->next = top;
    }
    top = n;
    }
}
```

Implementing a Queue

```
class Queue {
    Node first;
    Node last;
    Node pop() {
        if (!top) {
            return null;
        }
        Node t = top;
        top = top->next;
        return t;
    }
    void push(Node n) {
        if (last) {
            last->next = n;
        }
        last = n;
    }
}
```

Stacks and Queues CHAPTER-20

20.1 Write an algorithm to implement a queue using two stacks.

pg 234

20.2 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 235

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

pg 237

20.4 Write a C 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 | Top | IsEmpty | IsFull

pg 238

20.5 The Towers of Hanoi is a classical mathematical puzzle in which you have N rods and K disks of different sizes which can slide onto any rod. The puzzle starts with disks sorted in ascending order of size from top to bottom (eg, each disk sits on top of an even larger one). You have the following constraints:

1. Only one disk can be moved at a time.

2. A disk is slid off the top of one rod onto the next rod.

3. A disk can only be placed on top of a larger disk.

In programming terms, this can be simulated with stacks and the operations removeblock, putblock, findblock, isempty, isfull. Write a program to sort the disks in ascending order, given 10 disks and 5 rods.

pg 239

Strings CHAPTER-21

How to Approach:

String questions are some of the most common interview questions. Here are some of the gotchas:

C++ String Methods

strstr: Returns a pointer to the first occurrence of a string within a substring

strcmp: Compares two strings to each other.

strlen: Returns the length of the string, as determined by the location of the terminating null character.

strcpy: Copies a string from a source to a destination, including the null character.

C++ String Gotcha!

Question: What is the run time of this code?

```
    for (i = 1; i < strlen(s); i++) { n += i; }
```

Answer: O(n^2). Strlen is an O(n) function, which means that for each of the n cycles, you're doing O(n) work. n cycles, with O(n) work each time, means O(n^2)!

String Buffers

Question: What is the running time of this code?

```
    string[] words = {"foo", . . . };
    string sentence = "";
    foreach (string w in words) {
       sentence = sentence + word;
    }
```

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're actually creating a copy of sentence and running through all the letters in sentence to copy them over. If you have to run through up to n characters each time in the loop, and you're looping at least n times, that gives you an O(n) run time. Ouch! How do you avoid this problem? String buffers!

Strings CHAPTER-21

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

pg 240

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

pg 241

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

pg 243

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

pg 244

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

Example: find "bcd" in ["a", "", "", "", "", "b", "c", "", "", "", "d", ""]

pg 245

21.6 Code: Reverse C-Style String. (C-String means that "abcd\n" is actually represented as six characters)

pg 246

21.7 Given two strings, s1 and s2, write code to check if s2 is a rotation of s1 using only one call to strstr (eg, "waterbottle" is a rotation of "erbottlewat").

pg 247

21.8 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 encode xml element (as char *) as Byte *

FOLLOW UP

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

pg 248

Strings CHAPTER-21

21.9 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 250

21.10 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?

EXAMPLE:

input:

file: as was is the as the yahoo you me was the and

words: was, as

output: 2

pg 252

21.11 Write a program to find the longest word made of other words.

EXAMPLE:

input: test, tester, testertest, testing, testingtester

output (longest word): testingtester .

pg 253

Testing CHAPTER-22

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 (eg, 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.

Testing CHAPTER-22

22.1 How would you test a pen?

pg 255

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

pg 256

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

pg 257

22.4 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 258

22.5 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 259

Threads And Locks CHAPTER-23

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();
```

Threads And Locks CHAPTER-23

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

pg 260

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

pg 261

23.3 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 262

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

pg 264

23.5 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(.....); /* If B is called, a new thread will be created and the
corresponding function will be executed.
    C(.....); /* If C is called, a new thread will be created and the
corresponding function will be executed. */
}
Foo f;
f.A(.....);
f.B(.....);
f.C(.....);
```

PART A

i) Can you explain multithread synchronization mechanism?

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

PART B

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(.....);
```

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

pg 265

Trees and Graphs CHAPTER-24

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.

CAUTION: Not all binary trees are binary search trees

When given a binary tree question, many candidates assume that the interviewer means "binary *search* tree." So, listen carefully for that word "search." If you don't hear it, the interviewer just means 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 right node, then left 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.

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.

Trees and Graphs CHAPTER-24

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

pg 268

24.2 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 269

24.3 Design an algorithm and write code to find the first common ancestor of two nodes in a binary search tree. Avoid storing additional nodes in a data structure.

pg 270

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

pg 272

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

pg 273

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

pg 274

24.7 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 276

24.8 Given a directed graph, find a minimal set of vertices which touch all edges within a graph.

A vertex is said to 'touch' an edge if the edge either originates or terminates with that vertex.

pg 278

Solutions

The solutions provided are just one way of solving the problem. Many interview questions have multiple solutions which are optimal in memory, run time, flexibility or clarity.

Got A Better Solution? Or just a different solution?

We'd love to include it! Please contact gayle@careercup.com. We will attribute the solution to you (unless you prefer otherwise).

Can you provide clarity?

If you think the solution is confusing and you'd like to provide a better explanation, great! Contact gayle@careercup.com with your thoughts.

Corrections? Feedback? Other thoughts?

Please contact ***gayle@careercup.com***.

1.1 Write a method to generate the nth Fibonacci number pg 21

SOLUTION

There are three potential algorithms: (1) recursive approach (2) iterative approach (3) approach using a matrix. We have described the recursive and iterative approach below, as those are the most realistic for an interview. However, for those who are interest, you may read about the matrix approach at http://pages.cs.wisc.edu/~mhock/SSL/fibcalc.pdf.

```
Fibonacci: f(0) = 0, f(1) = 1, f(n) = f(n - 1) + f(n - 2)
int fibo(int n) { // Recursive solution
  if (n == 0) {
     return 0; // f(0) = 0
  } else if (n == 1) {
     return 1; // f(1) = 1
  } else if (n > 1) {
     return fibo(n-1) + fibo(n-2); // f(n) = f(n–1) + f(n-2)
  } else {
     return –1; // Error condition
  }
}

int fibo(int n)  { // Iterative solution
  if (n == 0) {
    return 0;
  }
  int a = 1
  int b = 1;
  for (int i = 3; i <= n; i++) {
     int c = a + b;
     a = b;
     b = c;
  }
  return b;
}
```

SUGGESTIONS AND OBSERVATIONS

» Did you solve this iteratively or recursively? Why? Explain to your interviewer.

» This is an easy, but very common problem. Interviewers often use this to verify that you have a basic knowledge of computer science before proceeding to more challenging questions.

» Don’t forget to validate your input!

1.2 Write a method to count the number of 2's between 0 and n. pg 21

SOLUTION

Let's find a pattern:

0-10	(10^1)	has 1	2s.
0-100	(10^2)	has 20	2s.
0-1000	(10^3)	has 300	2s

Given a digit x, 1 out of every 10 numbers will have a 2 in digit x. Therefore, between 1 and 1000, you will see 100 2's in digit 0, 100 2's in digit 1, and 100 2's in digit 2. So, 300 2's total. To generalize this into a rule: between 0 and 10^n, there are n*(10^n-1) 2s.

Let's call this function f(n) = n*(10^(n-1)).

1000 -> 300

2000 -> 300*2

3000 -> 300*3 + 1000

If we want to know the number of 2's between 1 and x * 10^n, then the solution is:

```
if x > 2 —> x * f(n) + 10^(n-1)
if x = 2 —> x * f(n) + 1
if x < 2 —> f(n-1)
```

Now, let's take a number like 2145

2's between 1 and 1999: 2*300 (300 2's between 1 and 999, and 300 between 1000 and 1999)

2's between 2000 and 2099: 1*20 + 100

20 2's between 2001 and 2099 (just like between 1 and 99)

Plus, each number starts with a 2

2's between 2100 and 2139: 4*1 + 10 + 40

4 2's in form 21x2

Plus 10 in form 212x

2's between 2140 and 2145: 1 + 6

SOLUTION (Con't)

```
int Count2sEfficiently(int num) {
    int i=0, countof2s = 0, digit = 0;
    int j = num, seendigits=0, position=0;

    while (j) {
        digit = j % 10;
        /* Digit < 2 implies there are no 2s contributed by this
         * digit */
        if (digit < 2) {
            countof2s = countof2s + digit * Numof2s(position-1);
        }
        /* Digit == 2 implies there are 2*numof2s contributed by the
         * previous position + num of 2s contributed by the
         * presence of this 2 */
        else if (digit == 2) {
            countof2s = countof2s + (digit * Numof2s(position-1)) +
                         seendigits + 1;
        }
        /* Digit > 2 implies there are digit * num of 2s by the prev.
         * position + 10^position */
        else if(digit > 2) {
            countof2s = countof2s + (digit) * Numof2s(position-1) +
                         power(10, position);
        }
        seendigits = seendigits + power(10, position)*digit;
        position++;
        j = j / 10;
    }
    return(countof2s);
}
// Returns the number of 2s between 0 and 10^n.
int Numof2s(int exponent) {
    int i=0;
    int power = 1;
    for(i=0; i< exponent; i++) {
        power = power * 10;
    }
    power = (exponent+1)*power;
    if (exponent >= 0) {
        return(power);
    }
    return(0);
}
```

SOLUTION #2 (Optimized)

```
#include <math.h>
#define NUM_OF_2s(n) (n * pow10_posMinus1)
uint Count2sEfficiently(uint num) {
    uint i=0, countof2s = 0, digit = 0;
    uint j = num, seendigits=0, position=0, pow10_pos = 1,
    /* maintaining this value instead of calling pow() is an 6x perf
     * gain (48s -> 8s) pow10_posMinus1. maintaining this value ‘
     * instead of calling Numof2s is an 2x perf gain (8s -> 4s).
     * overall > 10x speedup */
    while (j) {
        digit = j % 10;
        pow10_posMinus1 = pow10_pos / 10;
        countof2s += digit * NUM_OF_2s(position);
        /* we do this if digit <, >, or = 2
         * Digit < 2 implies there are no 2s contributed by this
         * digit.
         * Digit == 2 implies there are 2 * numof2s contributed by
         * the previous position + num of 2s contributed by the
         * presence of this 2 */
        if (digit == 2) {
            countof2s += seendigits + 1;
        }
        /* Digit > 2 implies there are digit * num of 2s by the prev.
         * position + 10^position */
        else if(digit > 2) {
            countof2s += pow10_pos;
        }
        seendigits = seendigits + pow10_pos * digit;
        pow10_pos *= 10;
        position++;
        j = j / 10;
    }
    return(countof2s);
}
```

Credit to Eric van Tassell

SOLUTION (Con't)

SUGGESTIONS AND OBSERVATIONS

This is a tricky problem. You can approach this problem as follows:

1. Select small simple numbers (10, 100, etc) to look for a pattern
2. Once you have a basic idea of the pattern, look at more complex numbers (152, 223, etc).
3. Develop a set of "rules" that govern how many 2's there are.
4. Test your rules! It's much easier to test a set of rules than it is to test code.
5. Translate your rules into code. Be careful. It's very easy to make mistakes.
6. Pause every couple of lines to make sure what you've written is correct.
7. Done? Go through and check your work.

The interviewer is looking for your ability to write bug-free code, so be very careful.

1.3 Given two lines on a Cartesian plane, determine whether the two lines would intersect. pg 21

SOLUTION

There are a lot of unknowns in this problem (what format are the lines in? What if they are the same line?), but let's assume:

» If two lines are the same (same line = same slope and y-intercept), they are considered to intersect.

» We get to decide the data structure.

```
struct Line {
    double slope;
    double yintercept;
}
bool intersect(Line line1, Line line2) {
    const double epsilon = 0.000001;
    return abs(line1.slope - line2.slope) > epsilon ||
        abs(line1.yintercept - line2.yintercept) < epsilon;
}
```

OBSERVATIONS AND SUGGESTIONS:

» Ask questions. This question has a lot of unknowns—ask questions to clarify them. Many interviewers intentionally ask vague questions to see if you'll clarify your assumptions.

» When possible, design and use data structures. It shows that you understand and care about object oriented design.

» Think through which data structures you design to represent a line. There are a lot of options, with lots of trade offs. Pick one and explain your choice.

» Don't assume that the slope and y-intercept are integers.

» Understand limitations of floating point representations. Never check for equality with ==.

1.4 Given two squares on a two dimensional plane, find a line that would cut these two squares in half. pg 21

SOLUTION

Any line that goes through the center of a rectangle must cut it in half. Therefore, if you drew a line connecting the centers of the two squares, it would cut both in half.

```
struct Line { Point start; Point end; }
Point middle(Square s) {
    return new Point((s.left + s.right)/2, (s.top + s.bottom)/2);
}
Line cut(Square s, Square t) {
    Point middle_s = middle(s);
    Point middle_t = middle(t);
    Line line = new Line();
    if (middle_s == middle_t) {
        line.start = Point(s.left, s.top);
        line.end = Point(s.right, s.bottom);
    } else {
        line.start = middle_s;
        line.end = middle_t;
    }
}
```

SUGGESTIONS AND OBSERVATIONS

The main point of this problem is to see how careful you are about coding. It's easy to glance over the special cases (eg, the two squares having the same middle). Make a list of these special cases *before* you start the problem and make sure to handle them appropriately.

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

pg 21

SOLUTION

Trailing zeros are contributed by pairs of 5 and 2, because 5*2 = 10. To count the number of pairs, we just have to count the number of multiples of 5. Note that while 5 contributes to one multiple of 10, 25 contributes two (because 25 = 5*5).

```
int NumOfTrailingZeros(int num) {
int count = 0;
    if (num < 0) {
    printf("Factorial is not defined for negative numbers\n");
    return 0;
}
for (int i = 5; num / i > 0; i *= 5) {
    count += num / i;
}
    return count;
}
```

Let's walk through an example to see how this works: Suppose num = 26. In the first loop, we count how many multiples of five there are by doing 26 / 5 = 5 (these multiples are 5, 10, 15, 20, and 25). In the next loop, we count how many multiples of 25 there are: 26 / 25 = 1 (this multiple is 25). Thus, we see that we get one zero from 5, 10, 15 and 20, and two zeros from 25 (note how it was counted twice in the loops). Therefore, 26! has six zeros.

OBSERVATIONS AND SUGGESTIONS:

» This is a bit of a brain teaser, but it can be approached logically (as shown above). By thinking through what exactly will contribute a zero, and what doesn't matter, you can come up with a solution. Again, be very clear in your rules up front so that you can implement this correctly.

1.6 Write a function that adds two numbers. You should not use + or any arithmetic operators.
pg 21

SOLUTION

To investigate this problem, let's start off by gaining a deeper understanding of how we add numbers. We'll work in Base 10 so that it's easier to see. To add 759 + 674, I would usually add digit[0] from each number, carry the one, add digit[1] from each number, carry the one, etc. You could take the same approach in binary: add each digit, and carry the one as necessary.

Can we make this a little easier? Yes! Imagine I decided to split apart the "addition" and "carry" steps. That is, I do the following:

1. Add 759 + 674, but "forget" to carry. I then get 323.
2. Add 759 + 674 but only do the carrying, rather than the addition of each digit. I then get 1110.
3. Add the result of the first two operations (recursively, using the same process described in step 1 and 2): 1110 + 323 = 1433.

Now, how would we do this in binary?

1. If I add two binary numbers together but forget to carry, bit[i] will be 0 if bit[i] in a and b are both 0 or both 1. This is an xor.
2. If I add two numbers together but only carry, I will have a 1 in bit[i] if bit[i-1] in a and b are both 1's. This is an and, shifted.
3. Now, recurse until there's nothing to carry.

```
int add_no_arithm(int a, int b) {
    if (b == 0) return a;
    int sum = a ^ b; // add without carrying
    int carry = (a & b) << 1; // carry, but don't add
    return add_no_arithm(sum, carry); // recurse
}
```

OBSERVATIONS AND SUGGESTIONS:

The Approach: There are a couple of suggestions for figuring out this problem:

1. Our first instinct in problems like these should be that we're going to have to work with bits. Why? Because when you take away the + sign, what other choice do we have? Plus, that's how computers do it.
2. Our next thought in problems like these should be to really, really understand how you add. Walk through an addition problem to see if you can understand something new—some pattern—and then see if you can replicate that on a computer.

Your interviewer is looking for two things in this problem:

1. Can you break down a problem and solve it?
2. Do you understand how to work with bits?

1.7 Write a method to implement *, - , / operations. You should use only the + operator.
pg 21

SOLUTION

With an understanding of what each operation (minus, times, divide) does, this problem can be approached logically.

» Subtraction should be relatively straightforward, as we all know that a - b is the same thing as a + (-1)*b.

» Multiplication: we have to go back to what we learned in grade school: 21 * 3 = 21 + 21 + 21. It's slow, but it works.

» Division is the trickiest, because we usually think of 21 / 3 as something like "if you divide a 21 foot board into 3 pieces, how big is each piece?" If we think about it the other way around (eg, the reciprocal), it's a little easier: "I divided a 21 foot board in x pieces and got pieces of 3 feet each, how many pieces were there?" From here, we can see that if we continuously subtract 3 feet from 21 feet, we'll know how many pieces there are. That is, we continuously subtract b from a and count how many times we can do that.

```
/* Flip a positive sign to negative, or a negative sign to pos */
int FnNegate(int a) {
    int neg = 0;
    int d = a < 0 ? 1 : -1;
    while (a != 0) {
        neg += d;
        a += d;
    }
    return neg;
}
/* Subtract two numbers by negating b and adding them
int FnMinus(int a, int b) {
    return a + FnNegate(b);
}
/* Check if a and b are different signs */
boolean DifferentSigns(int a, int b) {
    return ((a < 0 && b > 0) || (a > 0 && b < 0)) ? true : false;
}
/* Return absolute value */
int abs(int a) {
    if (a < 0) return FnNegate(a);
    else return a;
}
```

```
/* Multiply a by b by adding a to itself b times */
int FnTimes(int a, int b) {
    if (a < b) return FnTimes(b, a); // algo is faster if b < a
    int sum = 0;
    for (int iter = abs(b); iter > 0; --iter) sum += a;
    if (b < 0) sum = FnNegate(sum);
    return sum;
}
// returns 1, if a/b >= 0.5, and 0 otherwise
int DefineAndRoundFraction(int a, int b) {
    if(FnTimes(abs(a), 2) >= abs(b)) return 1;
    else return 0;
}
/* Divide a by b by literally counting how many times does b go into
* a. That is, count how many times you can subtract b from a until
* you hit 0. */
int FnDivide(int a, int b) {
    int quotient = 0;
    int divisor = FnNegate(abs(b));
    int divend; /* dividend */
    for (divend = abs(a); divend>=abs(divisor); divend += divisor) {
        ++quotient;
    }
    if(divend > 0) quotient += DefineAndRoundFraction(dividend, b);
    if (DifferentSigns(a, b)) quotient = FnNegate(quotient);
    return quotient;
}
```

OBSERVATIONS AND SUGGESTIONS

» A logical approach of going back to what exactly multiplication and division do comes in handy. Remember that. All (good) interview problems can be approached in a logical, methodical way!

» The interviewer is looking for this sort of logical work-your-way-through-it approach.

» This is a great problem to demonstrate your ability to write clean code—specifically, to show your ability to re-use code. For example, if you were writing this solution and didn't put FnNegate in its own method, you should move it out once you see that you'll use it multiple times.

» Be careful about making assumptions while coding. Don't assume that the numbers are all positive, or that a is bigger than b.

1.8 Design an algorithm to find the kth number such that the only prime factors are 3, 5, and 7. pg 21

SOLUTION

Note: Any such number will be in the form (3^i)*(5^j)*(7^k).

Let's start with an example list of the first 13 numbers.

1	-	3^0 * 5^0 * 7 ^ 0
3	3	3^1 * 5^0 * 7 ^ 0
5	5	3^0 * 5^1 * 7 ^ 0
7	7	3^0 * 5^0 * 7 ^ 1
9	3*3	3^2 * 5^0 * 7 ^ 0
15	3*5	3^1 * 5^1 * 7 ^ 0
21	3*7	3^1 * 5^0 * 7 ^ 1
25	5*5	3^0 * 5^2 * 7 ^ 0
27	3*9	3^3 * 5^0 * 7 ^ 0
35	5*7	3^0 * 5^1 * 7 ^1
45	5*9	3^2 * 5^1 * 7 ^0
49	7*7	3^0 * 5^0 * 7 ^7
63	3*21	3^2 * 5^0 * 7 ^1

» 3 * (previous number in list)

» 5 * (previous number in list)

» 7 * (previous number in list)

How would we find the next number in the list? Well, we could multiply 3, 5 and 7 times each number in the list and find the smallest element that has not yet been added to our list. This solution is O(n^2). Not bad, but I think we can do better.

In our current algorithm, we're doing 3*1, 3*3, 3*5, 3*7, 3*9, 3*15, 3*21, 3*25 …, and the same for 5 and 7. We've already done almost all this work before—why are we doing it again?

We can fix this by multiplying each number we add to our list by 3, 5, 7 and putting the results in one of the three first-in-first-out queues. To look for the next "magic" number, we pick the smallest element in the three queues. For a step-by-step example, see http://spreadsheets.google.com/pub?key=pZ-dftwPAN8FWyzA4xxlWSQ .

Algorithm:

```
1. Initialize array magic and queues Q3, Q5 and Q7
2. Insert 1 into magic.
3. Insert 1*3, 1*5 and 1*7 into Q3, Q5 and Q7 respectively.
4. Let x be the minimum element in Q3, Q5 and Q7.
5. Append x to magic
6. If x was found in:
    Q3 -> append x*3, x*5 and x*7 to Q3, Q5 and Q7. Remove x from Q3.
    Q5 -> append x*5 and x*7 to Q5 and Q7. Remove x from Q5.
    Q7 -> only append x*7 to Q7. Remove x from Q7.
Note: we do not need to append x*3 and x*5 to all lists because they
will already be found in another list.
7. Repeat steps 4 - 6 until we’ve found k elements, then return the
kth element.
```

Pseudo-code:

```
getKthMagicNumber(int k) {
    if k <= 0 then return 0
    val = 1
    Create FIFO queues Q3, Q5, Q7
    Q3.append(3)
    Q5.append(5)
    Q7.append(7)
    for (--k; k > 0; --k) { // we’ve done one iter already
        val = min(Q3.head, Q5.head, Q7.head)
        if val == Q7.head {
            Q7.pop
        } else {
            if val == Q5.head {
                Q5.pop
            } else { // must be from Q3
                Q3.pop
                Q3.append(val * 3)
            }
            Q5.append(val * 5)
        }
        Q7.append(val * 7)
    }
    return val
}
```

OBSERVATIONS AND SUGGESTIONS:

This is a hard problem. Don't feel bad if you weren't able to solve this. Does that mean it's too hard to be solved in an interview? Not exactly. After all, it is here because someone else was asked this question!

When you get this question, do your best to solve it—even though it's really difficult. Explain a brute force approach (not as tricky) and then start thinking about how you can optimize it. Or, try to find a pattern in the numbers.

Chances are, your interviewer will help you along when you get stuck. Whatever you do, don't give up! Think out loud, wonder aloud, explain your thought process. Your interviewer will probably jump in to guide you.

1.9 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. pg 21

```
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 bot-
tom: (56,90) (60,95) (65,100) (68,110) (70,150) (75,190)
```

SOLUTION

1. Sort all items by height at first, and by weight at second. So, if heights are different, sorting is being done by height, if heights are equal, sorting is being done by weight.

   For example, if we have: (60, 100) (70, 150) (56, 90) (75, 190) (60, 95) (68,110).
   After sorting we will have: (56, 90), (60, 95), (60,100), (68, 110), (70,150), (75,190).

Find the longest sequence which contains increasing heights and increasing weights

For this purpose we:

a) Start at the beginning of the sequence. There is the "max sequence" (empty at the beginning) – the sequence which has the max quantity of items.

b) If for the next item the height and the weight is not greater than those for the ent item, we mark this item as "unfit" .

(60,95)	(65,100)	(75,80)	(80, 100)
		(unfit item)	

c) If the sequence found has more items, then the "max sequence", it becomes "max sequence".

d) After that the search is repeated from the "unfit item", until we reach the end of the original sequence.

CONTINUED ON NEXT PAGE

```
List<HtWt> items = collections.sort(...); // Started with sorted list
List<HtWt> lastFoundSeq;
List<HtWt> findMaxSeq() {
  int currentUnfit = 0;
  List<HtWt> maxSeq = new ArrayList<HtWt>();
  while (!endOfOriginalSeqIsReached(currentUnfitItem)) {
     List<HtWt> nextSeq = new ArrayList<HtWt>();
     int nextUnfit = fillNextSeq(currentUnfit, nextSeq);
     maxSeq = seqWithMaxLength(maxSeq, nextSeq);
     if (nextUnfit == currentUnfit) break;
     else currentUnfit = nextUnfit;
  }
  return maxSeq;
}
// returns longer sequence
List<HtWt> seqWithMaxLen(List<HtWt> seq1, List<HtWt> seq2) {...}
// fills next seq w decreased wts - returns index of 1st unfit item.
int fillNextSeq(int startFrom, List<HtWt> seq) {
  int firstUnfitItem = startFrom;
  if (!endOfOriginalSeqIsReached(startFrom)) {
     HtWt prev = null;
     for (int i = startFrom; i <= items.size()-1; i++) {
        HtWt curr = items.get(i);
        if (itemCanBePlacedInSeq(curr, prev)) seq.add(curr);
        else firstUnfitItem = i;
        prev = curr;
     }
  }
  return firstUnfitItem;
}
boolean itemCanBePlacedInSeq(HtWt item, HtWt prev) {
  if (prev == null) return true;
  else if(prev.getHt()<=item.getHt() && prev.getWt()<=item.getWt())
     return true;
  else return false;
}
private boolean endOfOriginalSeqIsReached(int pos) {
  return pos >= items.size() - 1;
}
}
```

1.10 Given a two dimensional graph with 6000 points on it, find a line which passes the most number of points. pg 21

SOLUTION

Basic Idea: We need two points to define a line. So, for all possible lines, check which has the max number of points passing through. Let's say there are N points, and P is the set of all points.

Algorithm:

```
Lmax = 0, Maxp=0
for all pairs of point (Pi,Pj) in P
    calculate number of points m passing through the line L(P1,Pj)
    if (m > Maxp) {
        Maxp = m;
        Lmax = L(Pi,Pj);
    }
}
return Lmax;
```

Time complexity is O(N^3) because calculating the number of points on the query line is yet another pass through the set.

2.1 Suppose we have an array a1, a2, ..., an, b1, b2, ..., bn. Implement an algorithm to change this array to a1, b1, a2, b2, ..., an, bn. pg 23

SOLUTION

Shifting Array Solution - O(N^2)

```
    Shift the array as follows: a1, a2, a3, b1, b2, b3
        => a1, b1, a2, a3, b2, b3
        => a1, b1, a2, b2, a3, b3

    j=1;
    for(i=0,j=1; i<N; i++,j+=2) {
        RightRotate(j, N+i); //right rotate sub-array arr[j] to arr[N+i]
    }
```

O(N log N) Solution

Let's solve it by using the divide and conquer technique.

Rearrange(A,p,q)

```
1. if p is not equal to q do the following
2. r ← (p+q)/2
3. Exchange A[(p+r)/2..r] ←→ A[(p+q)/2 +1 ..(r+q)/2].
4. Rearrange(A,p,r)
5. Rearrange(A,r+1,q)
6. return
```

T(n) = 2T(n/2) + O(n) hence . T(n) < O(nlogn)

Call above procedure with p=1 and q=2n

Example:

Say we have i/p array : a1, a2, a3, a4, b1, b2, b3, b4

Carefully observe each step:

Step1: a1, a2, b1, b2, a3, a4, b3, b4

Now if you notice, we have two smaller problems to solve:

(a1, a2, b1, b2) and (a3, a4, b3, b4)

Now apply same logic again:

Step2: (a1, b1) (a2, b2) (a3, b3) (a4, b4)

2.2 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. pg 23

FOLLOW UP

Write the test cases for this method.

SOLUTION

First, ask yourself, what does the interviewer mean by an additional buffer? Can we use an additional array of constant size?

Algorithm—No (Large) Additional Memory:

1. For each character, check if it is a duplicate of already found characters.
2. Skip duplicate characters and update the non duplicate characters.

Time complexity is O(N^2).

```
void removeDuplicates(char *str) {
    if (!str)
        return;
    int len = strlen(str);
    if (len < 2)
        return;
    int tail = 1;

    for (int i = 1; i < len; ++i) {
        int j;
        for (j = 0; j < tail; ++j)
            if (str[i] == str[j])
                break;
        if (j == tail) {
            str[tail] = str[i];
            ++tail;
        }
    }
    str[tail] = 0;
}
```

Test Cases:

1. String does not contain any duplicates, e.g.: abcd

2. String contains all duplicates, e.g.: aaaa
3. Null string
4. String with all continuous duplicates, e.g.: aaabbb
5. String with non-contiguous duplicate, e.g.: abababa

Algorithm—With Additional Memory of Constant Size

```
void removeDuplicatesEff(char *str) {
    if (!str)
        return;
    int len = strlen(str);
    if (len < 2)
        return;

    bool hit[256];
    for(int i = 0; i < 256; ++i)
        hit[i] = false;
    hit[str[0]] = true;

    int tail = 1;
    for (int i = 1; i < len; ++i) {
        if (!hit[str[i]]) {
            str[tail] = str[i];
            ++tail;
            hit[str[i]] = true;
        }
    }
    str[tail] = 0;
}
```

Test Cases:

1. String does not contain any duplicates, e.g.: abcd
2. String contains all duplicates, e.g.: aaaa
3. Null string
4. Empty string
5. String with all continuous duplicates, e.g.: aaabbb
6. String with non-contiguous duplicates, e.g.: abababa

2.3 You are given an array of integers (both positive and negative). Find the continuous sequence with the largest sum. Return the sum. pg 23

```
EXAMPLE
input: {2, -8, 3, -2, 4, -10}
output: 5 [ eg, {3, -2, 4} ]
```

SOLUTION

You might have identified that dynamic programming is appropriate for this problem. Let's write down the recurrence:

sum(i,j) = max_k sum(i,k) + sum(k+1, j)

This will give an O(N^3) solution. A brute force summing of the sequences will give the same time.

A simple linear algorithm will work by keeping track of the current subsequence sum. If that sum ever drops below zero, that subsequence will not contribute to the subsequent maximal subsequence since it would reduce it by adding the negative sum.

```
int a[] = {6,-8, 3, -2, 4};
maxsum = 0;
sum = 0;
for (int i = 0; i < n; i++) {
    sum += a[i];
    if (maxsum < sum) {
        maxsum = sum;
    } else if (sum < 0) {
        sum = 0;
    }
}
```

2.4 Design an algorithm to find all pairs of integers within an array which sum to a specified value. pg 23

SOLUTION

One easy and (time) efficient solution involves a hash map from integers to integers. This algorithm works as follows:

1. Let V be the specified value, and let A be the array.
2. Create a hash map, M, which will map from array elements to occurrence counts.
3. For each element A[i] in the array:

    If V-A[i] is present in M, then print A[I] and V-A[I], M[V-A[i]] times.

    If A[i] is present in M, increment M[A[i]], otherwise let M[A[i]] = 1.

This algorithm conveys the special cases such as identical pairs (V/2, V/2) and repeated values (e.g. V=7 and A=[3,4,3,4]).

Alternate Solution

Definition of Complement: If we're trying to find a pair of numbers that sums to z, the complement of x will be z - x (that is, the number that can be added to x to make z). For example, if we're trying to find a pair of numbers that sum to 12, the complement of –5 would be 17.

The Algorithm: Imagine we have the following sorted array: {-2 -1 0 3 5 6 7 9 13 14 }. Let *first* point to the head of the array and *last* point to the end of the array. To find the complement of *first*, we just move *last* backwards until we find it. If *first* + *last* < *sum*, then there is no complement for *first*. We can therefore move *first* forward. We stop when *first* is greater than *last*.

Why must this find all complements for *first*? Because the array is sorted and we're trying progressively smaller numbers. When the sum of *first* and *last* is less than the sum, we know that trying even smaller numbers (as *last*) won't help us find a complement.

Why must this find all complements for *last*? Because all pairs must be made up of a *first* and a last. We've found all complements for *first*, therefore we've found all complements of *last*.

Implementation:

```
#include <iostream>
#include <conio.h>
#include <algorithm>
using namespace std;

void print_pairs(int * ptr, int num, int sum) {
    std::sort(ptr, ptr + num);
    int first = 0;
    int last = num - 1;
    while (first < last) {
        int s = ptr[first] + ptr[last];
        if (s == sum) {
            cout << ptr[first] << " " << ptr[last] << endl;
            ++first;
            --last;
        } else {
            if (s < sum) {
                ++first;
            } else {
                --last;
            }
        }
    }
}
int main() {
    int test[] = {9, 3, 6, 5, 7, -1, 13, 14, -2, 12, 0};
    print_pairs(test, sizeof(test) / sizeof(int), 12);
    _getch();
    return 0;
}
```

2.5 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 23

SOLUTION

Your first response to this question should be to ask about the O(N) requirement. The main constraint of the problem is that the mystery number must be determined with only bit inquiries.

Let’s assume that n is one less than a power of two (eg, n = 2^k - 1). If it’s not, it can be padded with no more than n/2 values to make it so.

Consider the four two-bit numbers 00, 01, 10, 11. If you add up the one’s bit, you get an even number. Likewise, if you add up the two’s bit, you get an even number. No matter how many bits there are in the number, if you add up a column, you get an even number.

Now if a number is missing, one of two things can happen to the column sums. Either it remains the same, indicating that the missing value didn’t contribute anything to the sum, or the sum is different because the missing value did contribute. Normally, the sums are even. So if they are different, they must be odd. If we keep a single bit count for each column, the result will be the missing number.

```
for j = 0 to k-1
    b[j] = 0
for i = 1 to n
    for j = 0 to k-1
        b[j] = (b[j] + fetch(a, i, j)) % 2
```

Note that the modulus operator may impact implementation performance. You might recommend that it be replaced with a bit-wise *and* with 1.

3.1 Write a function int BitSwapReqd(int A, int B) to determine the number of bits required to convert integer A to integer B. pg 25

```
input: 31, 14
output: 2
```

SOLUTION

```
int BitSwapReqd(int a, int b) {
    unsigned int count = 0;
    for (int c = a ^ b; c != 0; c = c >> 1) {
        count += c & 1;
    }
    return count
}
```

3.2 If you were to write a program to swap odd and even bits in integer, what is the minimum number of instructions required? (eg, bit 0 and bit 1 are swapped, bit 2 and bit 3 are swapped, etc). EXAMPLE: pg 25

SOLUTION

```
/* Mask all odd bits with 10101010 in binary (which is 0xAA), then
 * shift them left to put them in the even bits. Then, perform a
 * similar operation for even bits. This takes a total 5
 * instructions.
 */

#include <iostream>
#include <conio.h>
using namespace std;

int swapOddEvenBits(int x) {
    return ( ((x & 0xaaaaaaaa) >> 1) | ((x & 0x55555555) << 1) );
}
int main() {
    cout << swapOddEvenBits(0xa) << endl;
    cout << swapOddEvenBits(0x5) << endl;
    _getch();
    return 0;
}
```

3.3 Write a method which finds the maximum of two numbers. You should not use if-else or any other comparison operator. pg 25

```
Example
input: 5, 10
output: 10
```

SOLUTION

Let A = 1st number & B= 2nd number

Step1. Perform operation C = (A-B).

Step2. k = Most significant bit of C , i.e k =1 if B>A else K=0

Step 3. return A - k*C. This will return the maximum of A and B.

3.4 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 25

SOLUTION

Break this problem up into two parts. Printing the binary form of an integer and printing the binary form of the fractional part. The first problem is handled by repetitive division by two and storing the remainder.

```
void print_integer(string str) {
    string result;
    while (str[0] == '0') {
        str.erase(str.begin());
    }
    while (!str.empty()) {
        int remainder = 0;
        for(string::iterator i = str.begin(); i != str.end(); ++i){
            int x = *i - '0' + 10 * remainder;
            *i = '0' + x / 2;
            remainder = x % 2;
        }
        result.push_back('0' + remainder);
        if (str[0] == '0') {
            str.erase(str.begin());
        }
    }
    reverse(result.begin(),result.end());
    cout << result;
}
```

The second problem involves repetitive multiplying of the fractional part. Any time a carry pushes past the decimal, it's captured for the result. The function below incorporates this algorithm along with making calls to print_integer when necessary.

CONTINUED ON NEXT PAGE

```
void print_binary_repr(string str) {
    string::size_type decimal_pos = str.find(‘.’);
    string fraction;
    if (decimal_pos != str.npos) {
        string::iterator decimal = str.begin() + decimal_pos;
        set<string> seen;
        fraction.push_back(‘.’);
        while (true) {
            bool zeros = true;
            *decimal = ‘0’;
            string::iterator i = decimal;
            string::iterator j = i + 1;
            while (j != str.end()) {
                if (‘5’ <= *j) ++*i;
                *j = ‘0’ + (2*(*j - ‘0’) % 10);
                zeros = zeros && (*j == ‘0’);
                i = j;
                ++j;
            }
            fraction.push_back(*decimal);
            if (zeros)
                break; // all done!
            pair<set<string>::iterator, bool> res = seen.insert(str);
            if (!res.second) {
                cout << “ERROR”;
                return;
            }
        }
        str.erase(decimal,str.end());
    }
    print_integer(str);
    cout << fraction;
}
```

3.5 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 (eg, M becomes a substring of N located at i and starting at j). pg 25

```
EXAMPLE
input: N = 10000000000, M = 10101, i = 2, j = 6
output: N = 10001010100
```

SOLUTION

```
int foo(int n, int m, int i, int j) {
    int max = ~0; /* All 1's */
    int left = max - ((1 << j) - 1); /* 1's through position j, then
all 0's */
    int right = ((1 << i) - 1); /* 1's after position i */
    int mask = left | right; /* 1's, with 0s between i and j */
    return (n & mask) | (m << i); /* Clear i through j, then put m in
    there */
}
```

3.6 Write a function to swap a number in place without temporary variables.pg25

SOLUTION

```
swap(a,b) {
    if (a != b) {
        a = a^b;
        b = a^b;
        a = a^b;
    }
}
```

Note that if a and b share the same address, then the if condition will evaluate to true.

The *if* condition is important because:

1. If a and b have the same value, it's useless to swap them.
2. If a and b have the same reference, the XOR'ing logic would fail.

3.7 Given an integer, print the next smallest and next largest number that have the same number of 1 bits in their binary representation. pg 25

SOLUTION

The Brute Force Approach:

```
int GetNext(int i) {
    int num_ones = countOnes(i);
    i++;
    while (countOnes(i) != num_ones) {
        i++;
    }
}

int GetPrevious(int i) {
    int num_ones = countOnes(i);
    i--;
    while (countOnes(i) != num_ones) {
        i--;
    }
}
```

Boring, eh? Let's try this a better way.

Number Properties Approach for Next Number

Observations:

» If we "turn on" a 0, we need to "turn off" a 1

» If we turn on a 0 at bit i and turn off a 1 at bit j, the number changes by 2^i - 2^j.

» If we want to get a bigger number with the same number of 1s and 0s, i must be bigger than j.

Solution:

1. Traverse from right to left. Once we've passed a 1, turn on the next 0. We've now increased the number by 2^i. Yikes! Example: xxxxx011100 becomes xxxxx111100

2. Turn off the one that's just to the right side of that. We're now bigger by 2^i - 2^(i-1) Example: xxxxx111100 becomes xxxxx101100

3. Make the number as small as possible by rearranging all the 1s to be as far right as possible: Example: xxxxx101100 becomes xxxxx100011

To get the previous number, we do the reverse.

1. Traverse from right to left. Once we've passed a zero, turn off the next 1. Example: xxxxx100011 becomes xxxxx000011
2. Turn on the 0 that is directly to the right. Example: xxxxx000011 becomes xxxxx010011
3. Make the number as big as possible by shifting all the ones as far to the left as possible. Example: xxxxx010011 becomes xxxxx011100

Note: This step can also be worded as:

» Put one pointer, p1, at the least significant bit of the number

» Put another pointer, p2, at the most significant bit just after the found pattern '10'

» Now keep moving both of the pointers to each other and swap every occurrence of 1 for p1 and 0 for p2 until they meet each other.

4.1 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 27

SOLUTION

An interviewer is asking this problem to see how you think and approach problems—so don't just guess randomly.

Try approaching this the following way: What sorts of operations would get us to 8? I can think of a few:

4 * 2 = 8

16 / 2 = 8

4 + 4 = 8

Let's start with the first one. Is there any way to make 3 1 3 6 produce 4 * 2? We can easily notice that 3 + 1 = 4 (the first two numbers). We can also notice that 6 / 3 = 2. If we had "3 1 6 3", we'd be done, since (3 + 1)*(6 / 3) = 8. Although it seems a little unconventional to do this, we can, in fact, just flip the 6 and the 3 to get the solution:

((3 + 1)/ 3) * 6 = 8

OBSERVATIONS AND SUGGESTIONS:

» This problem is related to top down search in AI.

4.2 You have a 5 quart jug and a 3 quart jug, and an unlimited supply of water (but no measuring cups). How would you come up with exactly four quarts of water? pg 27

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

SOLUTION

5 Quart Contents	3 Quart Contents	Note
5	0	Filled 5 quart jug
2	3	Filled 3Q with 5Q's contents
0	2	Dumped 3Q
5	2	Filled 5Q
4	3	Fill remainder of 3Q with 5Q
4		Done! We have four quarts.

OBSERVATIONS AND SUGGESTIONS:

» Many brain teasers have a math / CS root to them—this is one of them! Note that as long as the two jug sizes are relatively prime (eg, have no common prime factors), you can find a pour sequence for any value between 1 and the sum of the jug sizes.

4.3 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 27

SOLUTION

Observation: Regardless of how we drop Egg1, Egg2 must do a linear search. Eg, if the Egg1 breaks between floor 10 and 15, we have to check every floor in between with the Egg2

The Approach:

A First Try: Suppose we drop an egg from the 10th floor, then the 20th, …

» If the first egg breaks on the first drop (Floor 10), then we have at most 10 drops total.

» If the first egg breaks on the last drop (Floor 100), then we have at most 19 drops total (floors 10, 20, ...,90, 100, then 91 through 99).

» That's pretty good, but all we've considered is the absolute worst case. We should do some "load balancing" to make those two cases more even.

Goal: Create a system for dropping Egg1 so that the most drops required is consistent, whether Egg1 breaks on the first drop or the last drop.

1. A perfectly load balanced system would be one in which Drops of Egg1 + Drops of Egg2 is always the same, regardless of where Egg1 broke.
2. For that to be the case, since each drop of Egg1 takes one more step, Egg2 is allowed one fewer step.
3. We must, therefore, reduce the number of steps potentially required by Egg2 by one drop each time. For example, if Egg1 is dropped on Floor 20 and then Floor 30, Egg2 is potentially required to take 9 steps. When we drop Egg1 again, we must reduce potential Egg2 steps to only 8. eg, we must drop Egg1 at floor 39.
4. We know, therefore, Egg1 must start at Floor X, then go up by X-1 floors, then X-2, …, until it gets to 100.
5. Solve for X+(X-1)+(X-2)+…+1 = 100. X(X+1)/2 = 100 -> X = 14

We go to Floor 14, then 27, then 39, … This takes 14 steps maximum.

4.4 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 (e.g., at least one). The hat is magical: it can be seen by other people, but not by the wearer of the hat himself. To remove the hat, you must dunk yourself 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. pg 27

FOLLOW UP:
Prove that your solution is correct.

SOLUTION

This problem seems hard, so let's simplify it by looking at specific cases.

Case c = 1: Exactly one man is wearing a hat.

Assuming all the men are intelligent, the man with the hat should look around and realize that no one else is wearing a hat. Since the genie said that at least one person is wearing a hat, he must conclude that he is wearing a hat. Therefore, he would be able to remove it that night.

Case c = 2: Exactly two men are wearing hats.

The two men with hats see one hat, and are unsure whether c = 1 or c = 2. They know, from the previous case, that if c = 1, the hats would be removed on Night #1. Therefore, if the other man still has a hat, he must deduce that c = 2, which means that he has a hat. Both men would then remove the hats on Night #2

Case General: If c = 3, then each man is unsure whether c = 2 or 3. If it were 2, the hats would be removed on Night #2. If they are not, they must deduce that c = 3, and therefore they have a hat. We can follow this logic for c = 4, 5, …

Proof by Induction

Using induction to prove a statement P(n)

If (1) ***P(1)*** = TRUE (eg, the statement is true when n = 1)

AND ***(2)*** if P(n) = TRUE -> P(n+1) = TRUE (eg, P(n+1) is true whenever P(2) is true).

THEN P(n) = TRUE for all n >= 1.

Explanation:

- Condition two sets up an infinite deduction chain: P(1) implies P(2) implies P(3) implies ... P(n) implies P(n+1) implies ....

- Condition one. (P(1) is true) ignites this chain, with truth cascading off into infinity.

Base Case: c = 1 (See previous page).

Assume true for c hats. Eg, if there are c hats, it will take c nights to remove all of them.

Prove true for c+1 hats.

Each man with a hat sees c hat, and can not be immediately sure whether there are c hats or c+1 hats. However, he knows that if there are c hats, it will take exactly c nights to remove them. Therefore, when c nights have passed and everyone still has a hat, he can only conclude that there are c+1 hats. He must know that he is wearing a hat. Each man makes the same conclusion and simultaneously removes the hats on night c+1.

Therefore, we have met our principles of induction. We have proven that it will take c nights to remove c hats.

4.5 There are 100 closed lockers in a hallway. A man begins by opening all the 100 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 (eg, he toggles every third locker). After his 100th pass in the hallway, in which he toggles only locker number 100, how many lockers are open? pg 27

SOLUTION

There are 10 lockers open, and here's why: Door n will be toggled (1 toggle = door opened or door closed) x times, where x is the number of factors of n. That is, door 20 will be toggled on round 1, 2, 4, 5, 10, and 20.

Question: When would a door be left open?

Answer: A door is left open if x is odd. You can think about this by pairing factors off as an open and a close. If there's one remaining, the door will be open.

Question: When would x be odd?

Answer: x is odd if n is a perfect square. Here's why: pair n's factors by their complements. For example, if n is 36, the factors are (1, 36), (2, 18), (3, 12), (4, 9), (6, 6). Note that (6, 6) only contributes 1 factor, thus giving n an odd numer of factors.

Question: How many perfect squares are there?

Answer: There are 10 perfect squares. You could count them (1, 4, 9, 16, 25, 36, 49, 64, 81, 100), or you could simply realize that you can take the numbers 1 through 10 and square them (1*1, 2*2, 3*3, ..., 10*10).

5.1 What is the difference between a struct and a class? Where would you use each?
pg 29

SOLUTION

The only difference is that structs have all members by default public, and classes private. However, because of this difference, structs are often used to hold small sets of data, whereas classes are more often used when there are methods.

Further reading:

http://carcino.gen.nz/tech/cpp/struct_vs_class.php

5.2 Write a method to print the last ten lines of an input file using C. pg 29

SOLUTION

One brute force way could be to count the number of lines (N) and then print from N-10 to Nth line. But, this requires two reads of the file – potentially very costly if the file is large.

We need a solution which allows us to read just once and be able to print the last K lines. We can create extra space for K lines and then store each set of K lines in the array. So, initially, our array has lines 0 through 9, then 1 through 10, then 2 through 11, etc (if K = 10). Each time that we read a new line, we purge the oldest line from the array. Instead of shifting the array each time (very inefficient), we will use a circular array. This will allow us to always find the oldest element in O(1) time.

EXAMPLE of Circular Array:

step 1: array = a b c d e f, p = 0

step 2: g b c d e f, p = 2

step 3: g h c d e f, p = 3

step 4: g h i d e f, p = 4

```
string L[K];

int index = 0;
while (!File.eof()) {
    L[index % K] = getLine(File); // read file line by line
    ++index;
}
// if less than ten lines were read, print them all
if (index < 10) {
    index = 0;
}
for(int i = 0; i < K; ++i) {
    printf("%s", L[(index + i) % K]);
}
```

OBSERVATIONS AND SUGGESTIONS:

» Note, if you printf(L[(index + i) % K]) bad things will happen if there are %'s in the strings.

5.3 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 29

SOLUTION

Compare and contrast Hash Table vs. STL map

In a hash table, a value is stored by applying hash function on a key. Thus, values are not stored in a hash table in sorted order. Additionally, since hash tables use the key to find the index that will store the value, an insert/lookup can be done in O(1) time. One must also handle potential collisions in a hash table.

In an STL map, insertion of key/value pair is in sorted order of key. It uses a tree to store values, which is why an O(log N) insert/lookup is required. There is also no need to handle collisions. An STL map works well for things like:

» find min element

» find max element

» print elements in sorted order

» find the exact element or, if the element is not found, find the next smallest number

How is a hash table implemented?

1. A good hash function is required, (e.g.: operation % prime number) to ensure that the hash values are uniformly distributed.
2. A collision resolving method is also needed: chaining (good for dense table entries), probing (good for sparse table entries), etc.
3. Implement methods to dynamically increase or decrease the hash table size on a given criterion. For example, when the [number of elements] by [table size] ratio is greater than the fixed threshold, increase the hash table size by creating a new hash table and transfer the entries from the old table to the new table by computing the index using new hash function.

What can be used instead of a hash table, if the number of inputs is small?

You can use an STL map. Although this takes O(log n) time, since the number of inputs is small, this time is negligible.

5.4 How do virtual functions work in C++? pg 29

SOLUTION

A virtual function depends on a "vtable" or "Virtual Table". If any function of a class is declared as virtual, a v-table is constructed which stores addresses of the virtual functions of this class. The compiler also adds a hidden vptr variable in all such classes which points to the vtable of that class. If a virtual function is not overridden in the derived class, the vtable of the derived class stores the address of the function in his parent class. The v-table is used to resolve the address of the function, for whenever the virtual function is called. Thus dynamic binding in c++ is performed through the vtable mechanism.

Thus, when we assign the derived class object to the base class pointer, the vptr points to the vtable of the derived class. This assignment ensures that the most derived virtual function gets called.

```
class Shape {
    int edge_length;
    virtual int Circumference () {
        cout << "Circumference of Base Class\n";
        return 0;
    }
}
class Triangle: public Shape {
    int Circumference () {
        cout<< "Circumference of Triangle Class\n";
        return 3 * edge_length;
    }
}
void main() {
    Shape * x = new Shape();
    x->circumference(); // prints "Circumference of Base Class"
    Shape *y = new Triangle();
    y->circumference(); // prints "Circumference of Triangle Class"
}
```

In the above example, circumference is a virtual function in shape class, so it becomes virtual in each of the derived classes (triangle, rectangle). C++ non-virtual function calls are resolved at compile time with static binding, while virtual function calls are resolved at run time with dynamic binding.

5.5 What is the difference between deep copy and shallow copy? Explain how you would use each. pg 29

SOLUTION

```
struct Test {
    char * ptr;
}
void shallow_copy(Test & src, Test & dest) {
    dest.ptr = src.ptr;
}
void deep_copy(Test & src, Test & dest) {
    dest.ptr = malloc(strlen(src.ptr) + 1);
    memcpy(dest.ptr, src.ptr);
}
```

Note that shallow_copy may cause a lot of programming run-time errors, especially with the creation and deletion of objects. Shallow copy should be used very carefully and only when a programmer really understands what he/she wants to do. In most cases shallow copy is used when there is a need to pass information about a complex structure without actual duplication of data (eg, call by reference). One must also be careful with destruction of shallow copy.

In real life, shallow copy is almost never used. There is an important programming concept called "smart pointer" that, in some sense, is an enhancement of the plain old shallow copy concept. See documentation on boost::shared_ptr and boost::weak_ptr for more details.

Deep copy should be used in most cases, especially when the size of the copied structure is small.

5.6 In a class, the ‘new’ operator is used for allocating memory for new objects. Can this be done using malloc? If yes, how? If no, why not? Are there any restrictions associated with the use of malloc in place of new? pg 29

SOLUTION

Yes and no. There are a few things which *new* does that *malloc* doesn’t:

1. *new* constructs the object by calling the constructor of that object

2. *new* doesn’t require typecasting of allocated memory.

3. It doesn’t require an amount of memory to be allocated, rather it requires a number of objects to be constructed.

So, if you use *malloc*, then you need to do the above things explicitly, which is not always practical. Additionally, *new* can be overloaded but *malloc* can’t be.

5.7 What is the significance of the keyword "volatile" in C? pg 29

SOLUTION

Volatile informs the compiler that the value of the variable can change from the outside, without any update done by the code.

Declaring a simple volatile variable:

```
    volatile int x;
    int volatile x;
```

Declaring a pointer variable for a volatile memory (only the pointer address is volatile):

```
    volatile int * x;
    int volatile * x;
```

Declaring a volatile pointer variable for a non-volatile memory (only memory contained is volatile):

```
    int * volatile x;
```

Declaring a volatile variable pointer for a volatile memory (both pointer address and memory contained are volatile):

```
    volatile int * volatile x;
    int volatile * volatile x;
```

Volatile variables are not optimized, but this can actually be useful. Imagine this function:

```
    void Fn(void) {
        Start:
            int opt = 1;
            if (opt== 1) goto start;
            else break;
    }
```

At first glance, our code appears to loop infinitely. The compiler will try to optimize it to:

```
    void Fn(void) {
        Start:
            int opt=1;
            if (true)
            goto start;
    }
```

This becomes an infinite loop. However, an external program might write '0' to the location of variable opt. Volatile variables are also useful when multithreaded programs have global variables and any thread can modify these shared variables. Of course, we don't want optimization on them.

5.8 What is name hiding in C++? pg 29

SOLUTION

Let us explain through an example. In C++, when you have a class with an overloaded method, and you then extend and override that method, you must override all of the overloaded methods.

For example:

```
class FirstClass {
public:
    virtual void MethodA (int);
    virtual void MethodA (int, int);
};
void FirstClass::MethodA (int i) {
    std::cout << “ONE!!\n”;
}
void FirstClass::MethodA (int i, int j) {
    std::cout << “TWO!!\n”;
}
```

This is a simple class with two methods (or one overloaded method). If you want to override the one-parameter version, you can do the following:

```
class SecondClass : public FirstClass {
 public:
    void MethodA (int);
};
void SecondClass::MethodA (int i) {
    std::cout << “THREE!!\n”;
}
int main () {
    SecondClass a;
    a.MethodA (1);
    a.MethodA (1, 1);
}
```

However, the second call won’t work, since the two-parameter MethodA is not visible. That is name hiding.

5.9 Why does a destructor in base class need to be declared virtual? pg 29

SOLUTION

Calling a method with an object pointer always invokes:

» the most derived class function, if a method is virtual.

» the function implementation corresponding to the object pointer type (used to call the method), if a method is non-virtual.

A virtual destructor works in the same way. A destructor gets called when an object goes out of scope or when we call delete on an object pointer.

When any derived class object goes out of scope, the destructor of that derived class gets called first. It then calls its parent class destructor so memory allocated to the object is properly released.

But, if we call delete on a base pointer which points to a derived class object, the base class destructor gets called first (for non-virtual function). For example:

```
#include <iostream.h>
class Base {
 public:
    Base() { cout << "Base Constructor " << endl; }
    ~Base() { cout << "Base Destructor " << endl; } /* see below */
};
class Derived: public Base {
 public:
    Derived() { cout << "Derived Constructor " << endl; }
    ~Derived() { cout << "Derived Destructor " << endl; }
};
void main() {
    Base *p = new Derived();
    delete p;
}
```

CONTINUED ON NEXT PAGE

Output:

```
Base Constructor
Derived Constructor
Base Destructor
```

But if we declare the base class destructor as virtual, this makes all the derived class destructors virtual as well.

If we replace the above destructor with:

```
virtual ~Base() {
    cout << “Base Destructor” << endl;
}
```

Then the output becomes:

```
Base Constructor
Derived Constructor
Derived Destructor
Base Destructor
```

So we should use virtual destructors if we call delete on a base class pointer which points to a derived class.

5.10 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 29

SOLUTION

The algorithm will maintain a mapping from a node address in the original structure to the corresponding node in the new structure. This mapping will allow us to discover previously copied nodes during a traditional depth first traversal of the structure. (Traversals often mark visited nodes--the mark can take many forms and does not necessarily need to be stored in the node.) Thus, we have a simple recursive algorithm:

```
typedef map<Node*,Node*> NodeMap;

Node * copy_recursive(Node * cur, NodeMap & nodeMap) {
    if(cur == NULL) {
        return NULL;
    }
    NodeMap::iterator i = nodeMap.find(cur);
    if (i != nodeMap.end()) {
        // we’ve been here before, return the copy
        return i->second;
    }
    Node * node = new Node;
    nodeMap[cur] = node; // map current node before traversing links
    node->ptr1 = copy_recursive(cur->ptr1, nodeMap);
    node->ptr2 = copy_recursive(cur->ptr2, nodeMap);
    return node;
}
Node * copy_structure(Node * root) {
    NodeMap nodeMap; // we will need an empty map
    return copy_recursive(root, nodeMap);
}
```

5.11 Write a smart pointer (smart_ptr) class. pg 29

SOLUTION

Smart_ptr is the same as a normal pointer, but it provides safety via automatic memory. It avoids dangling pointers, memory leaks, allocation failures etc. The smart pointer must maintain a single reference count for all instances.

```
template <class T>
class SmartPointer {
 public:
    SmartPointer(T * ptr) {
        ref = ptr;
        ref_count = malloc(sizeof(unsigned));
        *ref_count = 1;
    }
    SmartPointer(SmartPointer<T> & sptr) {
        ref = sptr.ref;
        ref_count = sptr.ref_count;
        ++*ref_count;
    }
    SmartPointer<T> & operator=(SmartPointer<T> & sptr) {
        if (this != &sptr) {
            ref = sptr.ref;
            ref_count = sptr.ref_count;
            ++*ref_count;
        }
        return *this;
    }
    ~SmartPointer() {
        --*ref_count;
        if (*ref_count == 0) {
            delete ref;
            free(ref_count);
            ref = ref_count = NULL;
        }
    }
    T* operator->() { return ref; }
    T& operator*() { return *ref; }
 protected:
    T * ref;
    unsigned * ref_count;
};
```

6.1 In how many different ways can a cube be painted by using three different colors of paint? pg 31

SOLUTION

Let X be the number of "unique" cubes (using the same definition as in the problem). Let Y be the number of ways you can "align" a given cube in space such that one face is pointed north, one is east, one is south, one is west, one is up, and one is down. Then the total number of possibilities is X * Y. Each of these possibilities "looks" different, because if you could take a cube painted one way, and align it a certain way to make it look the same as a differently painted cube aligned a certain way, then those would not really be different cubes. Also note that if you start with an aligned cube and paint it however you want, you will always arrive at one of those X * Y possibilities.

How many ways can you paint a cube that is already "aligned" (as defined above)?

You have three different colors: C1 C2 C3. Let N(Ci) be number of faces colored with Ci. Note: N(C1) + N(C2) + N(C3) = 6 and 1 <= N(Ci) <= 4.

Note the following:

If N(C1) = 4, N(C2) = 1, N(C3) = 1, then the answer is 6! / 4! = 30.

If N(C1) = 3, N(C2) = 2, N(C3) = 1, then the answer is 6! / (3! * 2!) = 60.

If N(C1) = 2, N(C2) = 2, N(C3) = 2, then the answer is 6! / (2! * 2! * 2!) = 90.

... repeat what is above, moving around the C1, C2, and C3 (eg, 3, 1, 2) ...

The first set of items has a sum of 30 + 60 + 90 = 180. How many more times would we have repeated that group? We would have repeated it 3! ways (since that's the number of permutations of a 3-character word). So, if we wrote it all out, what would the total sum be? It would be 180 * 6.

How many ways can you align a given cube?

Choose one face, and point it north; you have six options here. Now choose one to point east. There are only four sides that can point east, because the side opposite the one you chose to point north is already pointing south. There are no further options for alignment, so the total number of ways you can align the cube is 6 * 4.

Finally, (180 * 6) / (6 * 4) = 45.

The answer is 45.

6.2 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? pg 31

FOLLOW UP

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

SOLUTION

Part 1: (For clarity, we will solve this part assuming an X by Y grid)

Each path has X+Y steps. Imagine the following paths:

X X Y Y X (we move right on the first 2 steps, then down on the next 2, then right for the last step)

X Y X Y X (we move right, then down, then right, then down, then right)

…

Each path can be fully represented by the moves at which we move right. That is, if I were to ask you which path you took, you could simply say "I moved right on step 3 and 4."

Since you must always move right X times, and you have X + Y total steps, you have to pick X times to move right out of X+Y choices. Thus, there are C(X, X+Y) paths (eg, X+Y choose X):

(X+Y)! / (X! * Y!)

Part 2: Pseudo-code

```
bool is_free(int x, int y) { // return true if we can move here };
global current_path;
void print_paths(int x, int y) {
    current_path.add( pair(x, y) );
    if (0 == x && 0 == y) {
        print current_path();
        return;
    }
    if (x > 1 && is_free(x - 1, y)) {
        print_paths(x - 1, y);
    }
    if (y > 1 && is_free(y - 1, x)) {
        print_paths(y - 1, 1);
    }
    current_path.remove(path(x, y);
}
```

6.3 Write a method to compute all permutations of a string. pg 31

SOLUTION

Let’s assume a given string S represented by the letters A1, …, Ai, Ai+1, …, An

To permute set S, select an element from S and recursively permute the remaining set. In other words, we do this:

```
ArrayList getPermutes(string s) {
    ArrayList permutes = new ArrayList();
    foreach (char c in s) {
        string[] words = getPermutes(s.removeChar(c));
        foreach (word in words) {
            permutes.append(c + word);
        }
    }
    return permutes;
}
```

This solution takes O(n!) time, since there are n! permutations.

6.4 Implement an algorithm to print all valid (eg, properly opened and closed) combinations of n-pairs of parentheses. pg 31

```
EXAMPLE:
input: 3 (eg, 3 pairs of parentheses)
output: ()()(), ()(()), (())(), ((()))
```

SOLUTION

For input 4, call function below as

```
char str[9];
str[8]='\0';
printPar(4,4,4,str,0);
```

It is very clear that we will have 4 open and 4 close parentheses. The only logic you have to apply is that the right parentheses can appear in an output string only when there are already more left parentheses present in the output string. The rest of the comments are in-line with the code.

```
void printPar(int l, int r, char *str, int count) {
    if (l < 0 || r < l) return; // invalid state
    if (l == 0 && r == 0) {
        printf("%s\n", str); // found one, so print it
    } else {
        if (l > 0) { // try a left paren, if there are some available
            str[count] = '(';
            printPar(l - 1, r, str, count + 1);
        }
        if (r > l) { // try a right paren, if there's a matching left
            str[count] = ')';
            printPar(l, r - 1, str, count + 1);
        }
    }
}
```

6.5 Write a method that returns all subsets of a set. pg 31

SOLUTION

Let's start with approaching this from a combinatorics perspective.

» When we're generating a set, we have two choices for each element: (1) the element is in the set (the "yes" state) or (2) the element is not in the set (the "no" state). This means that each subset is a sequence of yesses / nos—eg, "yes, yes, no, no, yes, no"

» This gives us 2^n possible subsets.

» How can we iterate through all possible sequences of "yes" / "no" states for all elements? If each "yes" can be treated as a 1 and each "no" can be treated as a 0, then each subset can be represented as a binary string.

» Generating all subsets then really just comes down to generating all binary numbers (that is, all integers). Easy!

```
void AllSubsets(int * ary, int n) {
    unsigned long i;
    unsigned long c = 1 << n;
    assert( n < sizeof(unsigned long)*8);
    for (i = 0; i < c; ++i) {
        unsigned long tmp = i;
        int * elem = ary;
        printf("set:");
        while ( tmp > 0) {
            printf(" %d", *elem);
            tmp >>= 1;
            ++elem;
        }
        printf("\n");
    }
}
```

7.1 Write a method to find the number of employees in each department. pg 33

SOLUTION

```
select Dept_Name, Dept_ID, count(*) as ‘num_employees’
from Departments
left join Employees
on Employees.Dept_ID = Departments.Dept_ID
group by Dept_ID
```

7.2 What are the different types of joins? Please explain how they differ and why certain types are better in certain situations. pg 33

SOLUTION

All types of JOINs join tables, but there is a difference in results (R) of the operation:

INNER join - R contains only those data where the criteria match.

Example: Retrieving a pairing of IDs and student names. Since an ID without a student name or a student without an ID number would be useless, you can use an inner join.

OUTER join - R contains all the data regardless of the criteria match. When criteria don't match, R from left table or the right table would be NULLs.

Example: You have two tables of beverages: one which lists "regular" beverages, and one which lists calorie-free beverages. You want a list of all beverages, pairing the regular and calorie free alternative when possible. eg:

Beverage	Calorie-Free Beverage
Budweiser	[NULL]
Coca-Cola	Diet Coca-Cola
[NULL]	Fresca
Pepsi	Diet Pepsi

LEFT join - same as OUTER join except that if the criteria don't match, only the R from the left table will have values.

Example: Retrieving a list/pairing of students and their major in university. Some students don't have majors—that's ok. You still want to see those students.

RIGHT join - same as OUTER join except that if the criteria don't match, only the R from the right table will have values.

Example: Same as above, but reversing the left and right side.

7.3 What is normalization? Explain the pros and cons. pg 33

SOLUTION

Denormalization is the process of attempting to optimize the performance of a database by adding redundant data or by grouping data. In some cases, denormalization helps cover up the inefficiencies inherent in relational database software. A relational normalized database imposes a heavy access load over physical storage of data even if it is well tuned for high performance.

A normalized design will often store different but related pieces of information in separate logical tables (called relations). If these relations are stored physically as separate disk files, completing a database query that draws information from several relations (a join operation) can be slow. If many relations are joined, it may be prohibitively slow. There are two strategies for dealing with this. The preferred method is to keep the logical design normalized, but allow the database management system (DBMS) to store additional redundant information on disk to optimize query response. In this case it is the DBMS software's responsibility to ensure that any redundant copies are kept consistent. This method is often implemented in SQL as indexed views (Microsoft SQL Server) or materialized views (Oracle). A view represents information in a format convenient for querying, and the index ensures that queries against the view are optimized.

The more usual approach is to denormalize the logical data design. With care this can achieve a similar improvement in query response, but at a cost—it is now the database designer's responsibility to ensure that the denormalized database does not become inconsistent. This is done by creating rules in the database called constraints, that specify how the redundant copies of information must be kept synchronized. It is the increase in logical complexity of the database design and the added complexity of the additional constraints that make this approach hazardous. Moreover, constraints introduce a trade-off, speeding up reads (SELECT in SQL) while slowing down writes (INSERT, UPDATE, and DELETE). This means a denormalized database under heavy write load may actually offer worse performance than its functionally equivalent normalized counterpart.

A denormalized data model is not the same as a data model that has not been normalized, and denormalization should only take place after a satisfactory level of normalization has taken place and that any required constraints and/or rules have been created to deal with the inherent anomalies in the design. For example, all the relations are in third normal form and any relations with join and multivalued dependencies are handled appropriately.

From ***http://en.wikipedia.org/wiki/Denormalization***

7.4 Draw an entity-relationship diagram for a database with companies, people, and professionals (people who work for companies). pg 33

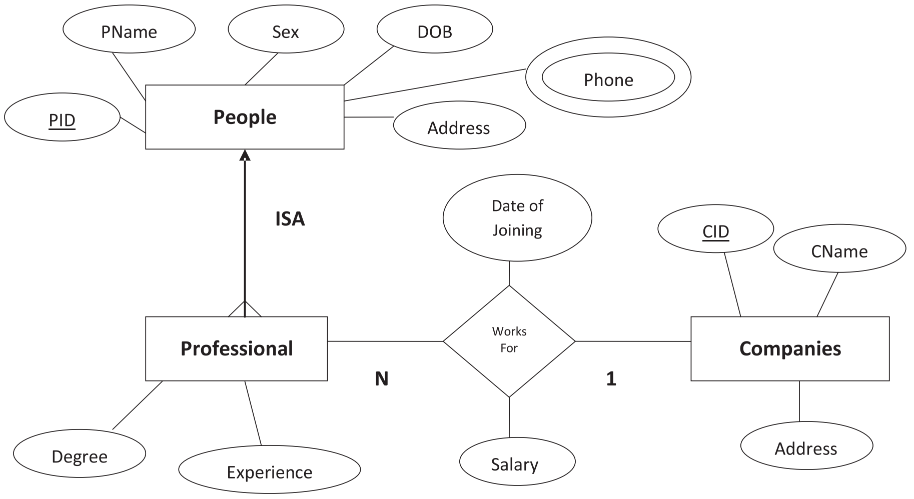
SOLUTION

People who work for companies are Professionals. So there is an ISA (is a) relationship between People and Professionals (or we could say that a Professional is derived from People).

Each Professional has additional information such as degree, work experiences, etc, in addition to the properties derived from People.

A Professional works for one company at a time, but Companies can hire many Professionals, so there is a Many to One relationship between Professionals and Companies. This "Works For" relationship can store attributes such as date of joining the company, salary, etc. These attributes are only defined when we relate a Professional with a Company.

A Person can have multiple phone numbers, that's why Phone is a multi-valued attribute.



7.5 You have to design a database that can store terabytes of data. It should support efficient range queries. How would you do it? pg 33

SOLUTION

Construct an index for each field that requires range queries. Use a B+ tree to implement the index. A B+ tree organizes sorted data for efficient insertion, retrieval and removal of records. Each record is identified by a key (for this problem, it is the field value). Since it is a dynamic, multilevel index, finding the beginning of the range depends only on the height of the tree, which is usually quite small. Record references are stored in the leaves, sorted by the key. Additional records can be found by following a next block reference. Records will be sequentially available until the key value reaches the maximum value specified in the query. Thus, runtimes will be dominated by the number of elements in a range.

Avoid using trees that store data at interior nodes, as traversing the tree will be expensive since it won't be resident in memory.

8.1 Explain what the following code does: pg 35

```
((n & (n-1)) == 0)
```

SOLUTION

It checks whether the value of "n" is a power of 2.

Example:

if n = 8, the bit representation is 1000

n & (n-1) = (1000) & (0111) = (0000)

So it returns zero only if its value is a power of 2.

The only exception to this is '0'.

0 & (0-1) = 0 but '0' is not a power of two.

Why does this make sense?

Imagine what happens when you subtract 1 from a string of bits. You read from left to right, turning each 0 to a 1 until you hit a 1, at which point that bit is flipped:

1000100100 -> (subtract 1) -> 1000100011

Thus, every bit, up through the first 1, is flipped. If there's exactly one 1 in the number, then every bit (other than the leading zeros) will be flipped. Thus, n & (n-1) == 0 if there's exactly one 1. If there's exactly one 1, then it must be a power of two.

8.2 Find the mistake(s) in the following code: pg 35

SOLUTION

The printf will never get executed, as "i" is initialized to 100, so condition check "i <= 0" will fail.

Suppose the code is changed to "i >= 0." Then, it will become an infinite loop, because "i" is an unsigned int which can't be negative.

The correct code to print all numbers from 100 to 1, is "i > 0".

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

One additional correction is to use %u in place of %d, as we are printing unsigned int.

8.3 What problems do you see in this piece of code (explain without compilation):
pg 35

```
template struct Foo : public Custom { };
template struct Foo {
    template struct rebind {
        typedef Foo Other;
    };
};
template struct Derived : public Base::template rebind >::Other {
    int foo() { printf("here we are\n"); };
};
main() {
    typedef Foo typedef Derived Derived_inst;
    Derived_inst ii;
    ii.foo();
}
```

SOLUTION

Issue #1

```
template <class Base>
struct Derived : public Base::template rebind <Derived<Base> >::Other {
```

Replace above line with:

```
template <class Base>
struct Derived : public Base::template rebind <Derived<Base> > {
```

That is, remove "::other" and it should work fine.

Issue #2

The above question does nothing but confuse you. I'll divide the code in three parts and figure out what is happening in each of them.

```
template <class T> struct Foo: public Custom{}
```

This defines a generic template, which has 'void' by default type, and then defines a struct foo.

By now we know a template version of struct Foo that has 'void' as a default type.

```
    template <>
        struct Foo<void> {
        template <class X>
        struct rebind { typedef Foo<X> Other;
        };
    };
```

Now we are specializing our template version for default 'void' type. We have one more struct 'rebind' defined in side Foo and that is also a generic version which has 'class X type'.

This is intended to confuse you. You have a struct inside a struct and they are both generic types.

The main concept is the definition of this line:

```
    typedef Foo<X> Other;
```

Here, you are doing typedef for Foo<X> as Other and it is only visible inside this struct. That is why the use of this unknown typedef in this line is not correct.

There are other cryptic lines inside main, which are, again, meant to confuse you. These were only syntactical problems but in real world applications simplicity is the key for coding. The above code just fails in that regard.

9.1 Design an algorithm to figure out if someone has won in a game of tic-tac-toe.
pg 37

SOLUTION

To make this more interesting, I've written this code for a general NxN tic tac toe board.

```
enum Piece { Empty, Red, Blue };
Piece hasWon(Piece[][] board) {
    int num_diagonal_matches1 = 0;
    int num_diagonal_matches2 = 0;
    for (int i = 0; i < N; i++) { // for each row and column
        int num_row_matches = 0;
        int num_col_matches = 0;
        if (board[i][i] == board[0][0]) {
            num_diagonal_matches1++; /* Check first diagonal. */
        }
        if (board[i][N-i-1] == board[0][N-1]) {
            num_diagonal_matches2++; /* Check second diagonal. */
        }
        for (int j = 0; j < N; j++) {
            if (board[i][j] == board[i][0] &&
                board[i][0] != Piece.Empty) {
                num_row_matches++; /* Check next cell in the row. */
            }
            if (board[j][i] == board[0][i]) {
                /* Check next cell in the column. */
                num_col_matches++;
            }
            /* Allow breaking early if you know no one can win this
             * row or col. */
            if (num_row_matches != j+1 && num_col_matches != j+1) {
                break;
            }
        }
        /* Check for a win. */
        if (num_row_matches == N) {
            return board[i][0]; /* There's a match on the ith row. */
        }
        /* CONTINUED ON NEXT PAGE */
```

```
            if (num_col_matches == N) {
                return board[0][i]; /* There’s a match on ith column. */
            }
        }
        if (num_diagonal_matches1 == N) {
            /* There’s a match on the downward diagonal */
            return board[0][0];
        }
        if (num_diagonal_matches2 == N) {
            /* There’s a match on the upward diagonal. */
            return board[N-1][0];
        }
        return Empty;
    }
```

SUGGESTIONS AND OBSERVATIONS:

» Note that the runtime could be reduced to O(N) with the addition of row and column count arrays (and two sums for the diagonals)

» A common follow up (or tweak) to this question is to write this code for an NxN board.

» A clever solution for the 3x3 case is to simply create a list of all winning tic tac toe boards. After all, there are only 2^9 = 512 boards (actually, less than that, since there can never be a board of all X’s). So, just create a list of all the boards in a bitmap, with 1 indicating that it’s a winning board. This will only take 512 bits—or 64 bytes. Not bad for an O(1) solution!

9.2 The Game of Master Mind is played as follows: pg 37

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 (eg, Slot #1 is red, Slots #2 and #3 are green, #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.

SOLUTION

```
// Use a bit mask for hash_mapping
struct result {
    int hits;
    int pseudoHits;
};
result estimate(const string& guess, const string& solution) {
    result res;
    res.hits = res.pseudohits = 0;
    int solution_mask = 0;
    for (int i = 0; i < 4; ++i) {
        solution_mask |= 1 << (1 + solution[i] - ‘A’)
    }
    for (int i = 0; i < 4; ++i) {
        if (guess[i] == solution[i]) {
            ++res.hits;
        } else if (solution_mask && ( 1 << (1 + guess[i] - ‘A’))) {
            ++res.pseudoHits;
        }
    }
    return res;
}
```

9.3 There is an 8x8 chess board in which two diagonally opposite corners have been cut off. You are given 31 dominos in which 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 37

SOLUTION

Impossible. Here's why: The chess board initially has 32 black and 32 white squares. By removing opposite corners (which must be the same color), we're left with 30 of one color and 32 of the other color. Let's say, for the sake of argument, that we have 30 black and 32 white.

When we lay down each domino, we're taking up one white and one black square. Therefore, 31 dominos will take up 31 white squares and 31 black squares exactly. On this board, however, we must 30 black squares and 32 white squares. Hence, it is impossible.

9.4 Find a way to arrange 8 queens on a chess board so that none of them share the same row, column or diagonal. pg 37

SOLUTION

We will use a backtracking algorithm. For each row, the column where we want to put the queen is based on checking that it does not violate the required condition.

1. For this, we need to store the column of the queen in each row as soon as we have finalized it. Let ColumnForRow[] be the array which stores the column number for each row.

2. The checks that are required for the three given conditions are:

»	On same Column :	ColumnForRow[i] == ColumnForRow[j]
»	On same Diagonal:	(ColumnForRow[i] - ColumnForRow[j]) == (i- j) or (ColumnForRow[j] - ColumnForRow[i]) == (i - j)

CONTINUED ON NEXT PAGE

Code to find one possible solution for "8 Queen Problem"

```
int ColumnForRow[9] = {0}; // We use row numbers 1 through 8

int Check(int row) {
    for (int cur = 1; cur < row; ++row) {
        int diff = abs(ColumnForRow[cur] - ColumnForRow[row]);
        if (diff == 0 || diff == row - cur ) {
            return false;
        }
    }
    return true;
}

void PlaceQueen(int row) {
    if (row == 9) {
        for (int i = 1; i <= 8; ++i) {
            printf("(%d,%d)", i, ColumnForRow[i]);
        }
        printf("\n");
        return; /* line (a) - see note below */
    }
    for (ColumnForRow[i] = 1; ColumnForRow[i] <= 8; ++ColumnForRow[i]) {
        if (Check(row)) {
            PlaceQueen(row + 1)
        }
    }
}
int main() {
    PlaceQueen(1);
    return(0);
}
```

NOTE: If you wanted to generate all queen arrangements, you would remove the return statement on line (a), and add an else statement at line (b).

9.5 Othello is played as follows: Each Othello piece is white on one side and black on the other. On your turn, you place a piece on the board so that your color is facing up. You must pick a spot such that your opponent's pieces are either on the left and the right, or on the top and the bottom. All of your opponent's pieces on the line between two of yours are then turned over, to become yours. Your goal is to own the most pieces.

Design the game Othello. Write a method to check whether someone has won the game. pg 37

SOLUTION

Othello has these major steps:

» Game () which would be the main function to manage all the activity in the game:

» Initialize the game which will be done by constructor

» Get first user input

» Validate the input

» Change board configuration

» Check if someone has won the game

» Get second user input

» Validate the input

» Change the board configuration

» Check if someone has won the game

See next page for pseudo-code.

```
class Othello {
 private: // data members
    int board[N][N];
    int size;
 public: // functions
    Othello(); // constructor to initialize the variables;
    /* It will alternately call the get moves for both players. After
     * every round, it would check if someone has won the game. */
    Game();
    getWhiteMove();
    getBlackMove();
    ValidateInput();
    chk_win();
}
Game (board[][]) {
    while (!chk_win(board)) {
        getWhiteMove();
        getBlackMove();
    }
}
getWhiteMove(board) / getBlackMove() {
    /* Ask user to input the index of the coordinate */
    if (validInput()) change the board configuration accordingly.
    else goto start of the function;
}
int chk_win(int board[], int size) {
    count=0;
    for (i=0; i<size; i++) {
        for (j=0; j<size; j++) {
            if (board[i][j]== WHITE) count++;
            else if(board[i][j] == BLACK) count --;
            else return EMPTY; /* Game not completed some squares are
still empty. */
        }
    }
    if (count > 0) return WHITE;
    if (count < 0) return BLACK;
    return DRAW;
}
```

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

pg 39

SOLUTION

If we want to make sure that no one outside the class can instantiate the class, we must declare the constructor of the class as private. Then the only possible way to give access is by providing a static public method within the class which creates an instance of the class.

This is usually used in Factory Method Pattern, by providing only one method (as factory method) which can instantiate the class.

So if you want to ensure that a class can't be inherited, declare the constructor of the class as private.

10.2 In Java, does the finally block gets executed if we insert a return statement inside the try block of a try-catch-finally? pg 39

SOLUTION

Yes, it will get executed.

The finally block gets executed when the try block exists. However, even when we attempt to exit within the try block (normal exit, return, continue, break or any exception), the finally block will still be executed.

Note: There are some cases in which the finally block will not get executed: if the virtual machine exits in between try/catch block execution, or the thread which is executing try/catch block gets killed.

10.3 What is the difference between final, finally, and finalize? pg 39

SOLUTIONS

Final

When applied to a variable (primitive): The value of the variable cannot change.

When applied to a variable (reference): The reference variable cannot point to any other object on the heap.

When applied to a method: The method cannot be overridden.

When applied to a class: The class cannot be subclassed.

Finally

There is an optional finally block after the try block or after the catch block. Statements in the finally block will always be executed (except if JVM exits from the try block). The finally block is used to write the clean up code.

Finalize

This is the method that the JVM runs before running the garbage collector.

10.4 Explain the difference between templates in C++ and generics in Java. pg 39

SOLUTION

C++ Templates	Java Generics
Classes and functions can be templated.	Classes and methods can be genericized.
Parameters can be any type or integral value.	Parameters can only be reference types (not primitive types).
Separate copies of the class or function are likely to be generated for each type parameter when compiled.	One version of the class or function is compiled, works for all type parameters.
Objects of a class with different type parameters are different types at run time.	Type parameters are erased when compiled; objects of a class with different type parameters are the same type at run time.
Implementation source code of the templated class or function must be included in order to use it (declaration insufficient).	Signature of the class or function from a compiled class file is sufficient to use it.
Templates can be specialized - a separate implementation could be provided for a particular template parameter.	Generics cannot be specialized.
Does not support wildcards. Instead, return types are often available as nested typedefs.	Supports wildcard as type parameter if it is only used once.
Does not directly support bounding of type parameters, but metaprogramming provides this.	Supports bounding of type parameters with "extends" and "super" for upper and lower bounds, respectively; allows enforcement of relationships between type parameters.
Allows instantiation of class of type parameter type.	Does not allow instantiation of class of type parameter type.
Type parameter of templated class can be used for static methods and variables.	Type parameter of templated class cannot be used for static methods and variables.
Static variables are not shared between classes of different type parameters.	Static variables are shared between instances of a classes of different type parameters.

*From **http://en.wikipedia.org/wiki/Comparison_of_Java_and_C%2B%2B#Templates_vs._Generics***

10.5 Explain what object reflection is in Java and why it is useful. pg 39

SOLUTION

Object Reflection is a feature in Java which provides a way to get reflective information about Java classes and objects, such as:

1. Getting information about methods and fields present inside the class at run time.
2. Creating a new instance of a class.
3. Getting and setting the object fields directly by getting field reference, regardless of what the access modifier is.

```
import java.lang.reflect.*;

public class Sample {
    public static void main(String args[]) {
        try {
            Class c = Class.forName("java.sql.Connection");
            Method m[] = c.getDeclaredMethods();
            for (int i = 0; i < 3; i++) {
                System.out.println(m[i].toString());
            } catch (Throwable e) {
                System.err.println(e);
            }
        }
    }
}
```

This code's output is the names of the first 3 methods inside the "java.sql.Connection" class (with fully qualified parameters).

Why it is Useful:

1. Helps in observing or manipulating the runtime behavior of applications
2. Useful while debugging and testing applications, as it allows direct access to methods, constructors, fields, etc.

10.6 Explain the different ways to pass parameters to a function (by value, by reference, by pointer) for the following cases: pg 39

SOLUTION

Passing by pointer and by reference are the same, so we will merge these in the table below. Classes and structures also handle passing by value vs by reference / pointer identically, so we will merge these as well.

	Value	Reference / Pointer
Basic Data Type	For small data types, it is more efficient to pass data by value because it may fit to hardware registers and/or because indirect addressing will slowdown the performance.	Generally less efficient.
Int Array	Generally less efficient.	Generally more efficient to pass by reference, since the cost of creating a copy is much less than the cost of indirect addressing.
Struct / Class	Depends on the size of the object. If it's small, it is generally more efficient to pass by value. Else, it is more efficient to pass by reference.	

One should always remember that passing by reference allows the called method to affect the original data.

10.7 You are given a class with synchronized methods A and B, and a normal method C. If you have two threads in one instance of a program, can these two threads call A at the same time? Can they call A and B at the same time? Can they call A and C at the same time? pg 39

SOLUTION

Java provides two ways to achieve synchronization: synchronized method and synchronized statement.

Synchronized method: Methods of a class which need to be synchronized are declared with "synchronized" keyword. If one thread is executing a synchronized method, all other threads which want to execute any of the synchronized methods on the same objects get blocked.

Syntax: method1 and method2 need to be synchronized

```
public class SynchronizedMethod {
    //variables declaration
    public synchronized returntype Method1() {
        //statements
    }
    public synchronized returntype method2() {
        //statements
    }
    //other methods
}
```

Synchronized statement: It provides the synchronization for a group of statements rather than a method as a whole. It needs to provide the object on which these synchronized statements will be applied, unlike in a synchronized method.

Syntax: synchronized statements on "this" object

```
synchronized(this) {
    /* statement1
    statement2
    .
    StatementN
    */
}
```

CONTINUED ON NEXT PAGE

Given a class with synchronized methods A and B, and a normal method C

Q1. If you have two threads in one instance of a program, can these two threads call A at the same time?

A1. Not possible, read the above paragraph.

Q2. Can they call A and B at the same time?

A2. Not possible, read the above paragraph.

Q3. Can they call A and C at the same time?

A3. Yes. Only the methods of the same object which are declared with the keyword *synchronized* can't be interleaved.

10.8 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 39

SOLUTION

One simple solution is to put count variables for get() and put() methods and, whenever they are called, increment the count. We can also achieve this by extending the existing library map and overriding the get() and put() functions.

At first glance, this seems to work. However, what if we created multiple instances of the map? How would you sum up the total count for each map object?

The simplest solution for this is to keep the count variables static. We know static variables have only one copy for all objects of the class so the total count would be reflected in count variables.

Solutions Large Scale and Memory Limits (CHAPTER-11)

11.1 If you were designing a web crawler, how would you avoid getting into infinite loops?
pg 41

SOLUTION

First, how does the crawler get into a loop? The answer is very simple: when we re-parse an already parsed page. This would mean that we revisit all the links found in that page, and this would continue in a circular fashion.

Be careful about what the interviewer considers the "same" page. Is it URL or content? One could easily get redirected to a previously crawled page.

So how do we stop visiting an already visited page? The web is a graph-based structure, and we commonly use DFS (depth first search) and BFS (breadth first search) for traversing graphs. We can mark already visited pages the same way that we would in a BFS/DFS.

We can easily prove that this algorithm will terminate in any case. We know that each step of the algorithm will parse only new pages, not already visited pages. So, if we assume that we have N number of unvisited pages, then at every step we are reducing N (N-1) by 1. That proves that our algorithm will continue until they are only N steps.

SUGGESTIONS AND OBSERVATIONS

» This question has a lot of ambiguity. Ask clarifying questions!

» Be prepared to answer questions about coverage.

» What kind of pages will you hit with a DFS versus a BFS?

» What will you do when your crawler runs into a honey pot that generates an infinite subgraph for you to wander about?

11.2 You have a billion urls, where each is a huge page. How do you detect the duplicate documents? pg 41

SOLUTION

Observations:

1. Pages are huge, so bringing all of them in memory is a costly affair. We need a shorter representation of pages in memory. A hash is an obvious choice for this.
2. Billions of urls exist so we don't want to compare every page with every other page (that would be O(n^2)).

Based on the above two observations we can derive an algorithm which is as follows:

1. Iterate through the pages and compute the hash table of each one.
2. Check if the hash value is in the hash table. If it is, throw out the url as a duplicate. If it is not, then keep the url and insert it in into the hash table.

This algorithm will provide us a list of unique urls. But wait, can this fit on one computer?

» How much space does each page take up in the hash table?
    » Each page hashes to a four byte value.
    » Each url is an average of 30 characters, so that's another 30 bytes at least.
    » Each url takes up roughly 34 bytes.
» 34 bytes * 1 billion = 31.6 gigabytes. We're going to have trouble holding that all in memory!

What do we do?

» We could split this up into files. We'll have to deal with the file loading / unloading—ugh.
» We could hash to disk. Size wouldn't be a problem, but access time might. A hash table on disk would require a random access read for each check and write to store a viewed url. This could take msecs waiting for seek and rotational latencies. Elevator algorithms could elimate random bouncing from track to track.
» Or, we could split this up across machines, and deal with network latency. Let's go with this solution, and assume we have n machines.
    » First, we hash the document to get a hash value v
    » v%n tells us which machine this document's hash table can be found on.
    » v / n is the value in the hash table that is located on its machine.

11.3 Design a method to find the frequency of occurrences of any given word in a book.
pg 41

SOLUTION

The first question – which you should ask your interviewer – is if you're just asking for a single word ("single query") or if you might, eventually, use the same method for many different words ("repetitive queries")? That is, are you simply asking for the frequency of "dog", or might you ask for "dog," and then "cat," "mouse," etc?

Solution: Single Query

In this case, we simply go through the book, word by word, and count the number of times that a word appears. This will take O(n) time. We know we can't do better than that, as we must look at every word in the book.

Solution: Repetitive Queries

In this case, we create a hash table which maps from a word to a frequency. Our pseudo code is then like this:

```
Hashtable setupDictionary(string[] book) {
    Hashtable table = new Hashtable();
    foreach (string word in book) {
        if (!table.contains(word)) {
            table.add(word, 0);
        }
        table[word] = table[word] + 1;
    }
}
int getFrequency(Hashtable table, string word) {
    if (table == null or word == null) {
        return -1;
    }
    if (table.contains(word)) {
        return table[word];
    }
    return 0;
}
```

Note: a problem like this is relatively easy. Thus, the interviewer is going to be looking heavily at how careful you are. Did you check for error conditions?

11.4 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 41

SOLUTION

There are a total of 2^32, or 4 billion, distinct integers possible. We have 1 GB of memory, or 8 billion bits.

Thus, with 8 billion bits, we can map all possible integers to a distinct bit with the available memory. The logic is as follows:

1. Create a bit vector (BV) of size 4 billion.
2. Initialize BV with all 0's
3. Scan all numbers (num) from the file and write BV[num] = 1;
4. Now scan again BV from 0th index
5. Return the first index which has 0 value.

Follow Up: What if we have only 10 MB memory?

It's possible to find a missing integer with just two passes of the data set. The first pass will count the integers that fall into blocks. There must be some block that contains fewer than the expected number of values in the block, otherwise there would be more values than we knew we had. The second pass uses a bit vector to track values within the target range. Now we just have to decide what the block size is.

CONTINUED ON NEXT PAGE

A quick answer is 2^20 values per block. We will need an array with 2^12 block counters and a bit vector in 2^17 bytes. Both of these can comfortably fit in 10*2^20 bytes.

What's the smallest footprint? When the array of block counters occupies the same memory as the bit vector. Let N = 2^32.

```
counters (bytes): blocks * 4
bit vector (bytes) : N / blocks / 8

blocks * 4 = N / blocks / 8
blocks^2 = N / 32

blocks = sqrt(N/2)/4
```

It's possible to find a missing integer with just under 65KB (or, more exactly, sqrt(2)*2^15 bytes).

```
int findMissing(source, blockSize)
    N = 2^32
    reset source
    initialize array count with ceil(N/blockSize) elements set to zero
    for each v in source do
        ++count[v/blockSize] // this can be bit shift if blockSize is 2^k

    for each index i in count do
        if count[i] < 2^32/blockSize
        base = blockSize * i
        break

free count array // not used anymore

reset source
initialize bit vector hit with blockSize elements set to zero
for each v in source do
    if base <= v < base + blockSize
        hit[(v-base)] = 1

for each index i in hit do
    if hit[i] == 0 then
        return i + base
```

11.5 You have two very large binary trees: T1, with millions of nodes, and T2, with hundreds of nodes. The trees store character data and duplicates are allowed. Create an algorithm to decide if T2 is a subtree of T1. pg 41

SOLUTION

Note that the problem here specifies that T1 has millions of nodes—this means that we should be careful of how much space we use. Let's say, for example, T1 has 10 million nodes—this means that the data alone is about 40 mb. We could create a string representing the inorder and preorder traversals. If T2's preorder traversal is a substring of T1's preorder traversal, and T2's inorder traversal is a substring of T1's inorder traversal, then T2 is a substring of T1. We can check this using a suffix tree. However, we may hit memory limitations because suffix trees are extremely memory intensive. If this become an issue, we can use an alternative approach.

Alternative Approach: The treeMatch procedure visits each node in the small tree at most once and is called no more than once per node of the large tree. Worst case runtime is at most O(n * m), where n and m are the sizes of trees T1 and T2, respectively. If k is the number of occurrences of T2's root in T1, the worst case runtime can be characterized as O(n + k * m).

```
treeMatch(r1, r2)
    if r2 == NULL
        return true /* nothing left in the subtree */
    if r1 == NULL
        return false /* nothing left in the big tree, but expecting
something */
    // match the tress
    if (r1.data != r2.data)
        return false /* data doesn't match */
    return treeMatch(r1.left, r2.left) and treeMatch(r1.right, r2.right)

subtree(r1, r2)
    // find a matching root
    if (r1.data == r2.data)
        if matchTree(r1,r2)
            return true
    return subtree(r1.left, r2) or subtree(r1.right, r2)
```

11.6 Find the largest 1 million numbers in 1 billion numbers. Assume that the computer memory can hold all one billion numbers. pg 41

SOLUTION

Approach 1: Sorting

Sort the elements and then take the first million numbers from that. Complexity is O(n log n).

Approach 2: Max Heap

1. Create a Min Heap with the first million numbers.
2. For each remaining number, insert it in the Min Heap and then delete the minimum value from the heap.
3. The heap now contains the largest million numbers.
4. This algorithm is O(n log m), where m is the number of values we are looking for.

Approach 3: Selection Rank Algorithm if you can modify the original array.

Selection Rank is a well known algorithm in computer science to find the ith smallest (or largest) element in an array in expected linear time. See http://users.encs.concordia.ca/~comp465_2/sl-w3-pdf.pdf for explanation, details and proof. The basic algorithm for finding the ith smallest elements goes like this:

» Pick a random element in the array and use it as a 'pivot'. Move all elements smaller than that element to one side of the array, and all elements larger to the other side.

» If there are exactly i elements on the right, then you just find the smallest element on that side.

» Otherwise, if the right side is bigger than i, repeat the algorithm on the right. If the right side is smaller than i, repeat the algorithm on the left for i – right.size()

Given this algorithm, you can either:

» Tweak it to use the existing partitions to find the largest i elements (where i = one million)

» Or, once you find the ith largest element, run through the array again to return all elements greater than or equal to it.

» This algorithm has expected O(n) time.

11.7 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 find out if a particular number exists in the array? pg 41

SOLUTION

We have 4KB of memory which means we can address up to 8 * 4 * (2^10) bits. Note that 32* (2^10) bits is greater than 32000.

The algorithm works as follows:

1. Create a bit vector (BV) of size 32000, with each element initially set to 0. We know we can create it since we have more memory available.
2. For each number 'num' in the array, set BV[num]=1.
3. Now for i=1 to N, check if(BV[i] == 0). If so, print ' Number i is Not Present'
4. END

11.8 Given a dictionary of millions of words, write a program 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 41

SOLUTION

Many problems involving a dictionary can be solved by doing some preprocessing. Where can we do preprocessing?

Well, if we're going to create a rectangle of words, we know that each row must be the same length and each column must have the same length. So, let's group the words of the dictionary based on their sizes. Let's call this grouping D, where D[i] provides a list of words of length i.

Next, observe that we're looking for the largest rectangle. What is the absolute largest rectangle that could be formed? It's (length of largest word) * (length of largest word).

```
int max_rectangle = longest_word * longest_word;
for z = max_rectangle to 1 {
    for each pair of numbers (i, j) where i*j = z {
        rectangle = make_rectangle(D[i], D[j])
        if (rectangle != null) {
            return rectangle.
        }
    }
}
```

By iterating in this order, we ensure that the first rectangle we find will be the largest.

Now, for the hard part: make_rectangle. Our approach is to rearrange words in list1 into rows and check if the columns are valid words in list2. However, instead of creating, say, a particular 10x20 rectangle, we check if the columns created after inserting the first two words are even valid pre-fixes. A trie becomes handy here.

CONTINUED ON NEXT PAGE

```
/* This function takes a list of words, one which forms possible
 * columns and the other that forms possible rows.  It then attempts
 * to find a subset of each list which can form a rectangle.  To do,
 * first assume that list1 has the shorter length.  */
void make_rectangle(list1, list2) {
    Trie t = createTrie(list2); // Create trie from words in list2
    return make_partial_rectangle(list1, list2, [], t)
}

/* This function recursively tries to form a rectangle with words
 * from list1 as the rows and list2 as the column.  To do so, we
 * start with an empty rectangle and add in a word from list1 as the
 * first row.  We then check the trie to see if each partial column is
 * a prefix of a word in list2.  If so, we branch recursively and
 * check the next word.
Rectangle make_partial_rectangle(list1, list2, rectangle, T) {
    // Check if we’ve formed a complete rectangle, and if each column
    // in the rectangle is a word.
    if (height of rectangle == length of each list2 word) {
        foreach (column w in rectangle) {
            if (w is in list2) {
                return null; // Invalid rectangle.
            }
        }
        return rectangle; // Valid Rectangle!
    }

    // Validate that each column is a substring of a word in list2.
    foreach (Column c in rectangle) {
        if (T.contains(w)) {
            return null // Invalid rectangle.
        }
    }
    foreach word in list1 {
        rect = make_partial_rectangle(list1, list2, rectangle,
                                      append(word), T)
        return rect;
    }
}
```

12.1 Implement an algorithm to find the nth to last element of a singly linked list.
pg 43

SOLUTION

Note: This problem screams recursion, but we'll do it a different way because it's trickier. In a question like this, expect follow up questions about the advantages of recursion vs iteration.

Assumption: minimum number of nodes in list is *n*.

Algorithm:

1. Create two pointers p1, p2 that point to the beginning of the node.
2. Increment p2 by n-1 positions, to make it point to the nth node from the beginning (to make the distance of n between p1 and p2).
3. Check for p2->next == null if yes return value of *p1*, otherwise increment *p1* and *p2*. If next of *p2* is null it means *p1* points to the nth node from the last as the distance between the two is *n*.
4. Repeat Step 3.

```
int* nthToLast(Node *head, int n) {
    if (head == NULL) {
        return NULL; // not found since there's no list
    }
    p1 = p2 = head;
    for (int j = 0; j < n - 1; ++j) { // skip n-1 steps ahead
        if (p2 == NULL) {
            return NULL; // not found since list size < n
        }
        p2 = p2->next;
    }
    while (p2->next != NULL) {
        p1 = p1->next;
        p2 = p2->next;
    }
    return &(p1->value);
}
```

CONTINUED ON NEXT PAGE

An alternate to returning the pointer is to provide a place to put the data once it's found and return a boolean to indicate success.

```
bool nthToLast(Node *head, int n, int *value) {
    if (head == NULL) {
        return false; // not found since there's no list
    }
    p1 = p2 = head;
    for (int j = 0; j < n - 1; ++j) { // skip n-1 steps ahead
        if (p2 == NULL) {
            return false; // not found since list size < n
        }
        p2 = p2->next;
    }
    while (p2->next != NULL) {
        p1 = p1->next;
        p2 = p2->next;
    }
    *value = p1->value;
    return true;
}
```

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

FOLLOW UPS & COMPLICATIONS

How would you solve this problem if a temporary buffer is not allowed?pg 43

SOLUTION

Start from the first node and hash the value of each node one by one. If a duplicate is found, delete the node.

```
void deleteDups(Node n) {
    Hashtable table = new Hashtable();
    Node root = n;
    while (n != null) {
        if (table.contains(n.value)) {
            Node next = removeElement(n);
            n = next;
        } else {
            table.add(n.value);
            n = n.next;
        }
    }
}
```

Note: There is no check for removing the root since the first element is always unique.

The above implementation takes O(n) space on average and O(n) time complexity.

CONTINUED ON NEXT PAGE

Follow Up: No buffer allowed.

Since no buffer is allowed, we can't sort the list. The best possible merge sort will take O(log n) stack space. So, compare every element with every other element one by one and remove the duplicate.

```
void DelDup1(struct node* head) {
    struct node * current = head;
    struct node * tail = head;
    if (current == NULL) return
    current = tail->next; // invariant
    while (current != NULL) {
        struct node * runner = head;
        while (runner != current) {
            if (runner->data == current->data) {
                struct node * tmp = current->next;
                free(current);
                tail->next = current = tmp;
                break;
            }
        }
        if (runner == current) {
            tail = current;
            current = current->next;
        }
    }
}
```

The runtime is O(NM) where N is the original list length and M is the resulting list length.

12.3 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. pg 43

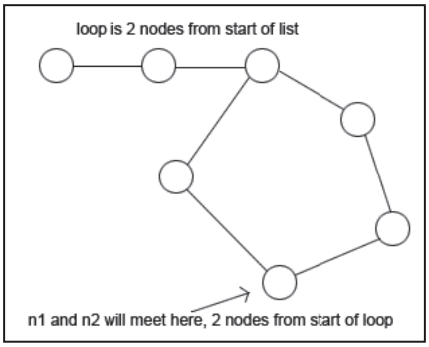
SOLUTION

General Idea: If we move two pointers, one with speed 1 and another with speed 2, they both end up meeting if the linked list has a loop.

We will break this process in two steps:

1. Find the meeting point
2. Find the first node in the loop.

```
/* (Step 1) Find the meeting point. This algorithm moves two pointers at
 * different speeds: one moves forward by 1 node, the other by 2. They
 * must meet (why? Think about two cars driving on a track—the faster car
 * will always pass the slower one!). If the loop starts k nodes after the
 * start of the list, they will meet at k nodes from the start of the
 * loop. */
n1 = n2 = head;
while (TRUE) {
    n1 = n1->next;
    n2 = n2->next->next;
    if (n1 == n2) {
        break;
    }
}
// Find the start of the loop.
n1 = head;
while (n1->next != n2->next) {
    n1 = n1->next;
    n2 = n2->next;
}
// Now n2 points to the start of the loop.
```


loop is 2 nodes from start of list
n1 and n2 will meet here, 2 nodes from start of loop

CONTINUED ON NEXT PAGE

So, why does this strategy work? Let's do a little bit of math. Suppose the "front" of the list is of length k and the loop is of length *m*.

(Note: if *a%m* = b, then for some integer *x, a - mx = b.*)

After t units of time:

» Faster Pointer: is at position k + (2t - k)%m (or, rewritten as k+(2t-k) - mx)

» Slower Pointer is at position k + (t - k)%m (or, rewritten as k+(t-k) - my)

So, when do they meet? Set them equal to each other

k+(2t-k)-mx = k+(t-k) - my

2t - k = t - k - m(x-y)

t = m(x-y)

This tells us that the pointers meet after m units, at which point the slower pointer is at (m - k) units into the loop and k units from the start of the loop. Therefore, if we move the first pointer back to the start of the linked list and move both pointers forward by k units, they'll meet at the start of the loop.

OBSERVATIONS AND SUGGESTIONS

» This technique is known as Floyd's cycle detection algorithm. It's also used in Pollard's rho technique for factorization.

12.4 Imagine you have an unbalanced binary search tree. Design an algorithm which creates a linked list of all the nodes at each depth (eg, if you have a tree with depth D, you'll have D linked lists). pg 43

SOLUTION

We can do a simple level by level traversal of the tree, with a slight modification of the breath-first traversal of the tree.

In a usual breath first search traversal, we simply traverse the nodes without caring which level we are on. In this case, it is critical to know the level. We thus use a dummy node to indicate when we have finished one level and are starting on the next.

```
struct ListOfLists {
    ListOfLists() : next(NULL), data(NULL) {}
    ListOfLists* next;
    LLNode* data;
};
struct LLNode {
    LLNode(Node* t, LLNode* n) : next(n), tree(t) {}
    LLNode* next;
    Node* tree;
};

ListOfLists* TreeLinkedLists(Node* root) {
    ListOfLists * results = new ListOfLists();
    traverse(root, results);
    return results;
}

void traverse(Node* root, ListOfLists* results) {
    if (root == NULL) return; // nothing to do
    results->head = new LLNode(root, results->head); // prepend tree node
    if (results->next == NULL) {
        // extend results if necessary
        results->next = new ListOfLists();
    }
    traverse(root->left, results->next);
    traverse(root->right, results->next);
}
```

12.5 Implement an algorithm to delete a node in the middle of a single linked list, given only access to that node. pg 43

```
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
```

SOLUTION

The solution to this is to simply copy the data from the next node into this node and then delete the next node.

NOTE: **This problem can not be solved if the node to be deleted is the last node in the linked list. That’s ok—your interviewer wants to see you point that out. Mark it as dummy in that case. If you have a Circular linked list, then this might be all the more interesting.**

```
void deletenode (Struct node* todelete) {
    if (todelete && (todelete->next)) {
        Struct node* temp = todelete->next;
        todelete->value = temp->value;
        todelete->next = temp->next;
        delete(temp); // function to delete the node
    }
}
```

12.6 You have two numbers represented by a linked list, where each node contains a single digit. Write a function that adds the two numbers and returns the sum as a linked list. pg 43

```
EXAMPLE:
input: (3 -> 1 -> 5), (5 -> 9 -> 2)
output: 9 -> 0 -> 7
```

SOLUTION

```
/* Push both the linked lists onto two stacks and then pop them one
by one for addition */
LinkedList *PushToStack(LinkedList *node1, LinkedList *node2) {
    Stack s1;
    Stack s2;
    while (node1) {
        s1.push(node1->data);
        node1 = node1->next;
    }
    while (node2) {
        s2.push(node2->data);
        node1 = node2->next;
    }
    LinkedList *result = NULL;
    int carry = 0;

    while (!s1.empty() && !s2.empty()) {
        int sum = s1.pop() + s2.pop() + carry;
        result = new LinkedList(sum % 10 , result);
        carry = sum / 10;
    }
    while (!s1.empty()) {
        int sum = s1.pop() + carry;
        result = new LinkedList(sum % 10 , result);
        carry = sum / 10;
    }
    while (!s2.empty()) {
        int sum = s2.pop() + carry;
        result = new LinkedList(sum % 10 , result);
        carry = sum / 10;
    }
    return result;
}
```

13.1 Explain the following terms: virtual memory, page fault, thrashing. pg 45

SOLUTION

Virtual memory is a computer system technique which gives an application program the impression that it has contiguous working memory (an address space), while in fact it may be physically fragmented and may even overflow on to disk storage. Systems that use this technique make programming of large applications easier and use real physical memory (e.g. RAM) more efficiently than those without virtual memory.

http://en.wikipedia.org/wiki/Virtual_memory

Page Fault: A page is a fixed-length block of memory that is used as a unit of transfer between physical memory and external storage like a disk, and a page fault is an interrupt (or exception) to the software raised by the hardware, when a program accesses a page that is mapped in address space, but not loaded in physical memory.

http://en.wikipedia.org/wiki/Page_fault

Thrash is the term used to describe a degenerate situation on a computer where increasing resources are used to do a decreasing amount of work. In this situation the system is said to be thrashing. Usually it refers to two or more processes accessing a shared resource repeatedly such that serious system performance degradation occurs because the system is spending a disproportionate amount of time just accessing the shared resource. Resource access time may generally be considered as wasted, since it does not contribute to the advancement of any process. In modern computers, thrashing may occur in the paging system (if there is not 'sufficient' physical memory or the disk access time is overly long), or in the communications system (especially in conflicts over internal bus access), etc.

http://en.wikipedia.org/wiki/Thrash_(computer_science)

13.2 What is a Branch Target buffer? Explain how it can be used in reducing bubble cycles in cases of branch misprediction. pg 45

SOLUTION

Branch misprediction occurs when the CPU mispredicts the next instruction to be executed.

The CPU uses pipelining which allows several instructions to be processed simultaneously. But during a conditional jump, the next instruction to be executed depends on the result of the condition. Branch Prediction tries to guess the next instruction. However, if the guess is wrong, we are penalized because the instruction which was executed must be discarded.

Branch Target Buffer (BTB) reduces the penalty by predicting the path of the branch, computing the target of the branch and caching the information used by the branch. There will be no stalls if the branch entry found on BTB and the prediction is correct, otherwise the penalty will be at least two cycles.

13.3 Describe direct memory access (DMA). Can a user level buffer / pointer be used by kernel or drivers? pg 45

SOLUTION

Direct Memory is a feature which provides direct access (read/write) to system memory without interaction from the CPU. The "DMA Controller" manages this by requesting the System bus access (DMA request) from CPU. CPU completes its current task and grants access by asserting DMA acknowledgement signal. Once it gets the access, it reads/writes the data and returns back the system bus to the CPU by asserting the bus release signal. This transfer is faster than the usual transfer by CPU. Between this time CPU is involved with processing task which doesn't require memory access.

By using DMA, drivers can access the memory allocated to the user level buffer / pointer.

13.4 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 45

SOLUTION

1. The keyboard sends a scan code of the key to the keyboard controller (Scan code for key pressed and key released is different).
2. The keyboard controller interprets the scan code and stores it in a buffer.
3. The keyboard controller sends a hardware interrupt to the processor. This is done by putting signal on "interrupt request line": IRQ 1.
4. The interrupt controller maps IRQ 1 into INT 9.
5. An interrupt is a signal which tells the processor to stop what it was doing currently and do some special task.
6. The processor invokes the "Interrupt handler". CPU fetches the address of "Interrupt Service Routine" (ISR) from "Interrupt Vector Table" maintained by the OS (Processor use the IRQ number for this).
7. The ISR reads the scan code from port 60h and decides whether to process it or pass the control to program for taking action.

13.5 Write a program to find whether a machine is big endian or little endian. pg 45

SOLUTION

```
#define BIG_ENDIAN 0
#define LITTLE_ENDIAN 1
int TestByteOrder() {
    short int word = 0x0001;
    char *byte = (char *) &word;
    return (byte[0] ? LITTLE_ENDIAN : BIG_ENDIAN);
}
```

13.6 Discuss how would you make sure that a process doesn't access an unauthorized part of the stack. pg 45

SOLUTION

As with any ambiguously worded interview question, it may help to probe the interviewer to understand what specifically you're intended to solve. Are you trying to prevent code that has overflowed a buffer from compromising the execution by overwriting stack values? Are you trying to maintain some form of thread-specific isolation between threads? Is the code of interest native code like C++ or running under a virtual machine like Java?

As background, remember that stacks are a thread-specific concept-- there can be multiple stacks in a process.

NATIVE CODE

One threat to the stack is malicious program input, which can overflow a buffer and overwrite stack pointers, thus circumventing the intended execution of the program.

If the interviewer is looking for a simple method to reduce the risk of buffer overflows in native code, modern compilers provide this sort of stack protection through a command line option. With Microsoft's CL, you just pass /GS to the compiler. With GCC, you can use -fstack-protector-all.

For more complex schemes, you could set individual permissions on the range of memory pages representing the stack section you care about. In the Win32 API, you'd use the VirtualProtect API to mark the page PAGE_READONLY or PAGE_NOACCESS. This will cause the code accessing the region to go through an exception on access to the specific section of the stack.

Alternately, you could use the HW Debug Registers (DRs) to set a read or write breakpoint on the specific memory addresses of interest. A separate process could be used to debug the process of interest, catch the HW exception that would be generated if this section of the stack were accessed.

CONTINUED ON NEXT PAGE

However, it's very important to note that under normal circumstances, threads and processes are not means of access control. Nothing can prevent native code from writing anywhere within the address space of its process, including to the stack. Specifically, there is nothing to prevent malicious code in the process from calling VirtualProtect and marking the stack sections of interest PAGE_EXECUTE_READWRITE. Equally so, nothing prevents code from zeroing out the HW debug registers, eliminating your breakpoints. In summary, nothing can fully prevent native code from accessing memory addresses, including the stack, within its own process space.

MANAGED CODE

A final option is to consider requiring this code that should be "sandboxed" to run in a managed language like Java or C# / .NET. By default, the virtual machines running managed code in these languages make it impossible to gain complete access to the stack from within the process.

One can use further security features of the runtimes to prevent the code from spawning additional processes or running "unsafe" code to inspect the stack. With .NET, for example, you can use Code Access Security (CAS) or appdomain permissions to control such execution.

13.7 What are the best practices to prevent reverse engineering of DLLs? pg 45

SOLUTION

Best practices include the following:

» Use obfuscators.

» Do not store any data (string, ect) in open form. Always compress or encode it.

» Use a static link so there is no DLL to attack.

» Strip all symbols.

» Use a .DEF file and an import library to have anonymous exports known only by their export ids.

» Keep the DLL in a resource and expose it in the file system (under a suitably obscure name, perhaps even generated at run time) only when running.

» Hide all real functions behind a factory method that exchanges a secret (better, proof of knowledge of a secret) for a table of function pointers to the real methods.

» Use anti-debugging techniques borrowed from the malware world to prevent reverse engineering. (Note that this will likely get you false positives from AV tools.)

» Use protectors.

13.8 A device boots with an empty FIFO queue. In the first 400 ns period after start-up, 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 45

SOLUTION

While a perfectly optimal solution is complex, an interviewer is most interested in how you approach the problem.

THEORY

First, note that writes do not have to be evenly distributed within a period. Thus a likely worst case is 80 words are written at the end of the first period, followed by 80 more at the start of the next.

Note that the maximum write rate for a full period is exactly matched by a full period of processing (400 ns / ((3 ns + 2 ns)/process) = 80 processed words/period).

As the 2nd period in our example is fully saturated, adding writes from a 3rd period would not add additional stress, and this example is a true worst case for the conditions.

A SAFE QUEUE DEPTH

For an estimate of maximum queue size, notice that these 160 writes take 640 ns (160 writes * 4 ns / write = 640 ns), during which time only 128 words have been read (640 ns / ((3 ns + 2 ns) / word) = 128 words). However, the first read cannot start until the first write has finished, which fills an extra slot in the queue.

Also, depending on the interactions between read and write timing, a second additional slot may be necessary to ensure a write does not trash the contents of a concurrently occurring read.

Thus, a safe estimate is that the queue must be at least 34 words deep (160 - 128 + 1 + 1 = 34) to accommodate the unread words.

FINDING AN OPTIMAL (MINIMAL) QUEUE DEPTH

Depending on the specifics of the problem, it's possible that the final queue spot could be safely removed.

In many cases, the time required to do an edge case analysis to determine safety is not worth the effort. However, if the interviewer is interested, the full analysis follows.

We are interested in the exact queue load during the final (160th) consecutive write to the queue. We can approach this by graphing the queue load from time = 0 ns, observing the pattern, and extending it to time = 716 ns, the time of the final consecutive write.

The graph below shows that the queue load increases as each write begins, and decreases 3 ns after a read begins. Uninteresting time segments are surrounded by [brackets]. Each character represents 1 ns.

```
Writer:
    [ 0 - 79 ns] AAAABBBBCCCCDDDDEEEE [100 - 707 ns] XXXXYYYYZZZZ____
[>= 724 ns]
Worker:
    [ 0 - 79 ns] ____aaaaabbbbbcccccd [100 - 707 ns] opppppqqqqqrrrrr
[>= 724 ns]
Queue load:
    11112221222222223222 [30 + q:] 3333333343333322
```

Y = Writing word 159 @ 712 ns

Z = Writing word 160 @ 716 ns

q = Processing word 127 @ 714 ns

r = Processing word 128

Note that the queue load does in fact reach a maximum of 34 at time = 716 ns.

As an interesting note, if the problem had required only 2 ns of the 5 ns processing time to complete a read, the optimal queue depth would decrease to 33.

The below graphs are unnecessary, but show empirically that adding writes from the 3rd period does not change the queue depth required.

CONTINUED ON NEXT PAGE

```
Writer: [ < 796 ns ] ____AAAABBBB [808 - 873 ns] !!@@@@####$$
Worker: [ < 796 ns ] ^^^&&&&&**** [808 - 873 ns] yyyyyzzzzzaa
Queue load: [20 + q] 877788778887 [20 + q:] 112111221122
```

A = Writing word 161

& = Processing word 144

# = Writing word 181

z = Processing word 160 @ 779 ns

```
Writer: [ < 1112 ns] YYYYZZZZ____
Worker: [ < 1112 ns] ^^&&&&&*****
Queue load: [30 + q:] 333343333322
```

Z = Writing word 240 @ 1116 ns

& = Processing word 207 @ 1114 ns

13.9 Write an aligned malloc & free function that takes number of bytes and aligned byte (which is always power of 2) pg 45

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.

SOLUTION

1. We will use malloc routine provided by C to implement the functionality.

   Allocate memory of size (bytes required + alignment – 1 + sizeof(void*)) using malloc.

   alignment: malloc can give us any address and we need to find a multiple of alignment.

   (Therefore, at maximum multiple of alignment, we will be alignment-1 bytes away from any location).

   sizeof(size_t): We are returning a modified memory pointer to user, which is different from the one that would be returned by malloc. We also need to extra space to store the address given by malloc, so that we can free memory in aligned_free by calling free routine provided by C.
2. If it returns NULL, then aligned_malloc will fail and we return NULL
3. Else, find the aligned memory address which is a multiple of alignment (call this p2).
4. Store the address returned by malloc (eg, p1 is just size_t bytes ahead of p2), which will be required by aligned_free.
5. Return p2.

```
void* aligned_malloc(size_t required_bytes, size_t alignment) {
    void* p1; // original block
    void** p2; // aligned block
    int offset = alignment - 1 + sizeof(void*);
    if ((p1 = (void*)malloc(required_bytes + offset)) == NULL) {
        return NULL;
    }
    p2 = (void**)(((size_t)(p1) + offset) & ~(alignment - 1));
    p2[-1] = p1;
    return p2;
}
```

13.10 Write a function called my2DAlloc which allocates a two dimensional array. You should minimize the number of calls to malloc and make sure that the memory is accessible by the notation arr[i][j]. pg 45

SOLUTION

We will use one call to malloc.

Allocate one block of memory to hold the row vector and the array data. The row vector will reside in rows * sizeof(int*) bytes. The integers in the array will take up another rows * cols * sizeof(int) bytes.

Constructing the array in a single malloc has the added benefit of allowing disposal of the array with a single free call rather than using a special function to free the subsidiary data blocks.

```
int** My2DAlloc(int rows, int cols) {
    int header = rows * sizeof(int*);
    int data = rows * cols * sizeof(int);
    int** rowptr = (int**)malloc(header + data);
    int* buf = (int*)(rowptr + rows);
    int k;
    for (k = 0; k < rows; ++k) {
        rowptr[k] = buf + k*cols;
    }
    return rowptr;
}
```

14.1 Write an algorithm such that if an element in an MxN matrix is 0, its entire row and column is set to 0.
pg 47

SOLUTION

```
// Input: Array[M][N]
int i,j;
int row[M] = /* set to all 0's */
int column[N] = /* set to all 0's */;

// Store the row and column index with value 0
for (i=0; i < M; i++) {
    for (j=0; j < N;j++) {
        if (a[i][j] == 0) {
            row[i] = 1; /* Row i will eventually be set to all 0's */
            column[j] = 1; /* Column j will eventually be set to all 0's */
        }
    }
}

// Make the entry arr[i][j] as 0 if either row i or column j as the 0
for (i = 0; i < M; i++) {
    for ( j = 0; j < N; j++) {
        if ((row[i] == 1 || column[j] == 1)) {
            a[i][j] = 0;
        }
    }
}
```

14.2 Given an image represented by a 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 47

SOLUTION

The rotation can be performed in layers, where you perform a cyclic swap on the edges on each layer. In the first for loop, we rotate the first layer (outermost edges). We rotate the edges by doing a four-way swap first on the corners, then on the element clockwise from the edges, then on the element three steps away.

Once the exterior elements are rotated, the interior region's edges are rotated.

```
void rotate(int **matrix, int n) {
    for (int layer = 0; layer < n/2; ++layer) {
        int first = layer;
        int last = n - 1 - layer;
        for(int i = first; i < last; ++i) {
            int offset = i - first;
            int top = matrix[first][i]; // save top

            // left -> top
            matrix[first][i] = matrix[last-offset][first];

            // bottom -> left
            matrix[last-offset][first] = matrix[last][last - offset];

            // right -> bottom
            matrix[last][last - offset] = matrix[i][last];

            // top -> right
            matrix[i][last] = top; // right <- saved top
        }
    }
}
```

14.3 Given a matrix in which each row and each column is sorted, write a method to find an element in it.
pg 47

SOLUTION

Assumptions:

» Rows are sorted left to right in ascending order. Columns are sorted top to bottom in ascending order.

» Matrix is of size MxN.

This algorithm works by elimination. Every move to the left (--col) eliminates all the elements below the current cell in that column. Likewise, every move down eliminates all the elements to the left of the cell in that row.

```
bool FindElem(int **mat, int elem) {
    int row=0, col = n-1;
    while (row < M && col > 0) {
        if (mat[row][col] == elem) {
            return true;
        } else if (mat[row][col] > elem) {
            col--;
        } else {
            row++;
        }
    }
    return false;
}
```

14.4 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 47

SOLUTION

Assumption: Square is of size NxN.

This algorithm does the following:

1. Iterate through every (full) column from left to right.
2. At each (full) column, look at the subcolumns (from biggest to smallest).
3. At each subcolumn, see if you can form a square with the subcolumn as the left side. If so, update currentMaxSize and go to the next (full) column.
4. If N - col <= currentMaxSize, then break completely. We've found the largest square possible. *Why?* At *each column, we're trying to create a square with that column as the left side. The largest such square we could possibly create is N - col. Thus, if N-col <= currentMaxSize, then we have no need to proceed.*

```
int currentMaxSize = 0;
int col = 0;
// Iterate through each column from left to right
while (N - col > currentMaxSize) { // See step 4 above
    foreach sub column sc of col // starting from the biggest) {
        if there is black sided rectangle with left side at the sc {
            update currentMaxSize
            break; // go to next (full) column
        }
    }
}

Time complexity: O(N^2)
```

14.5 Given an NxN matrix of positive and negative integers, write code to find the sub-matrix with the largest possible sum. pg 47

SOLUTION

Brute Force: Complexity O(N^6)

1. Make all possible sub-arrays by having four "for loops" which gives 2 row ids and 2 column ids. Complexity O(N^2 x N^2)
2. Find the sum of each sub-array and compare with the max sum to update it. Complexity O(n^2);

Total complexity : O(N^6)

```
int MaxSum = INT_MIN;
//Generate first column id
for (p1 = 0; p1 < N; p1++) {
  // Generate second column id
  for (p2 = p1+1; p2 < N; p2++) {
     int CurrSum =0;
     // Generate first row id
     for (p3= 0; p3 < N; p3++) {
        // Generate second row id
        for (p4 = 0; p4 < N; p4++) {
           // Find sum of sub array formed by row(p3,p4) column(p1,p2)
           for(row = p3; row < p4; row++) {
              RowSum = 0;
              //calculate sum of current Row with column id from p1 to p3
              for(col=p1; col<p2; col++) {
                 RowSum = RowSum + arr[row][col];
              }
              CurrSum = CurrSum + RowSum;
           }
           if (MaxSum < CurrSum) {
              MaxSum = CurrSum;
           }
        }
     }
  }
}
```

Improvement to N^4 Solution:

As in the above solution, increase the size of row/column required to recalculate previous sums again and again. We can save time by having an array: array[i][j] 0<j<=N, which will contain the sum of all elements from column 1 to j in row i. Time required is O(NM) and for all i array[i][0] = 0.

Now if we want to find, for any row k, the sum of elements from column x to column y, sumxy = array[k][y] - array[k][x-1]: Suppose we have an array sumxy[N], so for each pair for column x and y, sumxy[k] = array[k][y] – array[k][x-1]

Example:

1	2	3
-5	-1	5
6	9	7

After step 1:

1	2	3
-4	1	8
2	10	15

Now, if we want to find sum of rectangle:

-1	5
9	7

Then it would be: sum = (sum[2][1] –sum[0][1]) + (sum[2][2] – sum[0][2])) = (10-2) + (15-3) =20

15.1 Explain what happens, step by step, after you type a URL into a browser. Use as much detail as possible. pg 49

SOLUTION

There's no right, or even complete, answer for this question. This question allows you to go into arbitrary amounts of detail depending on what you're comfortable with. Here's a start though:

1. Browser contacts the DNS server to find the IP address of URL.
2. DNS returns back the IP address of the site.
3. Browser opens TCP connection to the web server at port 80.
4. Browser fetches the html code of the page requested.
5. Browser renders the HTML in the display window.
6. Browser terminates the connection when window is closed.

One of the most interesting steps is Step 1 and 2 - "Domain Name Resolution". The web addresses we type are nothing but an alias to an ip address in human readable form. Mapping of domain names and their associated Internet Protocol (IP) addresses is managed by the domain name system (DNS), which is a distributed but hierarchical entity.

Each domain name server is divided into zones. A single server may only be responsible for knowing the host names and IP addresses for a small subset of a zone, but DNS servers can work together to map all domain names to their IP addresses. That means if one domain name server is unable to find the ip addresses of a requested domain then it requests the information from other domain name servers.

15.2 Explain any common routing protocol in detail. For example: BGP, OSPF, RIP.
pg 49

SOLUTION

Depending on the reader's level of understanding, knowledge, interest or career aspirations, he or she may wish to explore beyond what is included here. Wikipedia and other websites are great places to look for a deeper understanding. We will provide only a short summary.

BGP: Border Gateway Protocol

BGP is the core routing protocol of the Internet. "When a BGP router first comes up on the Internet, either for the first time or after being turned off, it establishes connections with the other BGP routers with which it directly communicates. The first thing it does is download the entire routing table of each neighboring router. After that it only exchanges much shorter update messages with other routers.

BGP routers send and receive update messages to indicate a change in the preferred path to reach a computer with a given IP address. If the router decides to update its own routing tables because this new path is better, then it will subsequently propagate this information to all of the other neighboring BGP routers to which it is connected, and they will in turn decide whether to update their own tables and propagate the information further."

See *http://www.livinginternet.com/i/iw_route_egp_bgp.htm*

RIP: Routing Information Protocol

"RIP provides the standard IGP protocol for local area networks, and provides great network stability, guaranteeing that if one network connection goes down the network can quickly adapt to send packets through another connection. "

"What makes RIP work is a routing database that stores information on the fastest route from computer to computer, an update process that enables each router to tell other routers which route is the fastest from its point of view, and an update algorithm that enables each router to update its database with the fastest route communicated from neighboring routers."

See *http://www.livinginternet.com/i/iw_route_igp_rip.htm*

CONTINUED ON NEXT PAGE

OSPF: Open Shortest Path First

"Open Shortest Path First (OSPF) is a particularly efficient IGP routing protocol that is faster than RIP, but also more complex."

The main difference between OSPF and RIP is that RIP only keeps track of the closest router for each destination address, while OSPF keeps track of a complete topological database of all connections in the local network. The OSPF algorithm works as described below.

» Startup. When a router is turned on it sends Hello packets to all of its neighbors, receives their Hello packets in return, and establishes routing connections by synchronizing databases with adjacent routers that agree to synchronize.

» Update. At regular intervals each router sends an update message called its "link state" describing its routing database to all the other routers, so that all routers have the same description of the topology of the local network.

» Shortest path tree. Each router then calculates a mathematical data structure called a "shortest path tree" that describes the shortest path to each destination address and therefore indicates the closest router to send to for each communication; in other words -- "open shortest path first".

See *http://www.livinginternet.com/i/iw_route_igp_ospf.htm*

15.3 Compare and contrast the IPv4 and IPv6 protocols. pg 49

SOLUTION

Ipv4 and Ipv6 are the internet protocols applied at the network layer. Ipv4 is the most widely used protocol right now and Ipv6 is the next generation protocol for internet.

» Ipv4 is the fourth version of Internet protocol which uses 32 bit addressing whereas Ipv6 is a next generation internet protocol which uses 128 bits addressing.

» Ipv4 allows 4,294,967,296 unique addresses where as Ipv6 can hold 340-undecillion (34, 000, 000, 000, 000, 000, 000, 000, 000, 000, 000, 000, 000) unique Ip address.

» Ipv4 has different class types, these are: A,B,C,D and E. Class A, Class B, and Class C are the three classes of addresses used on IP networks in common practice. Class D addresses are reserved for multicast. Class E addresses are simply reserved, meaning they should not be used on IP networks (used on a limited basis by some research organizations for experimental purposes).

» IPv6 addresses are broadly classified into three categories:

  1. Unicast addresses: A Unicast address acts as an identifier for a single interface. An IPv6 packet sent to a Unicast address is delivered to the interface identified by that address.

  2. Multicast addresses: A Multicast address acts as an identifier for a group/set of interfaces that may belong to the different nodes. An IPv6 packet delivered to a multicast address is delivered to the multiple interfaces.

  3. Anycast addresses: Anycast addresses act as identifiers for a set of interfaces that may belong to the different nodes. An IPv6 packet destined for an Anycast address is delivered to one of the interfaces identified by the address.

» Ipv4 address notation: 239.255.255.255, 255.255.255.0

» IPv6 addresses are denoted by eight groups of hexadecimal quartets separated by colons in between them.

» An example of a valid IPv6 address: 2001:cdba:0000:0000:0000:0000:3257:9652

CONTINUED ON NEXT PAGE

Because of the increase in the population, there is a need of Ipv6 protocol which can provide solution for:

1. Increased address space
2. More efficient routing
3. Reduced management requirement
4. Improved methods to change ISP
5. Better mobility support
6. Multi-homing
7. Security
8. Scoped address: link-local, site-local and global-address space

15.4 What is network/subnet mask? Explain how a host A sends a message/packet to host B when: pg 49

a) both are on same network

b) both are on different networks

Explain which layer takes routing decision and how.

SOLUTION

A mask is a bit pattern used to identify the network/subnet address. The IP address consists of two components: the network address and the host address.

The IP addresses are categorized into different classes which are used to identify the network address.

Example: Consider IP address 152.210.011.002. This address belongs to Class B, so:

Network Mask: 11111111.11111111.00000000.00000000

Given Address: 10011000.11010101.00001011.00000010

By anding Network Mask and IP Address, we get the following network address:

10011000.11010101.00000000.00000000 (152.210.0.0)

Host address: 00001011.00000010

Similarly, a network administrator can divide any network into sub-networks by using subnet mask. To do this, we further divide the host address into two or more subnets.

For example, if the above network is divided into 18 subnets (requiring a minimum of 5 bits to represent 18 subnets), the first 5 bits will be used to identify the subnet address.

Subnet Mask: 11111111.11111111.11111000.00000000 (255.255.248.0)

Given Address: 10011000.11010101.00001011.00000010

So, by anding the subnet mask and the given address, we get the following subnet address: 10011000.11010101.00001000.00000000 (152.210.1.0)

CONTINUED ON NEXT PAGE

How Host A sends a message/packet to Host B:

When both are on same network: the host address bits are used to identify the host within the network.

Both are on different networks: the router uses the network mask to identify the network, and route the packet. The host can be identified using the network host address.

The network layer is responsible for making routing decisions. A routing table is used to store the path information and the cost involved with that path, while a routing algorithm uses the routing table to decide the path on which to route the packets.

Routing is broadly classified into Static and Dynamic Routing based on whether the table is fixed or it changes based on the current network condition.

15.5 What are the differences between TCP/UDP? Explain how TCP handles reliable delivery (explain ACK mechanism), flow control (explain TCP sender's/receiver's window) and congestion control. pg 49

SOLUTION

TCP (Transmission Control Protocol): TCP is a connection-oriented protocol. A connection can be made from client to server, and from then on any data can be sent along that connection.

Reliable - when you send a message along a TCP socket, you know it will get there unless the connection fails completely. If it gets lost along the way, the server will re-request the lost part. This means complete integrity; data will not get corrupted.

Ordered - if you send two messages along a connection, one after the other, you know the first message will get there first. You don't have to worry about data arriving in the wrong order.

Heavyweight - when the low level parts of the TCP "stream" arrive in the wrong order, resend requests have to be sent. All the out of sequence parts must be put back together, which requires a bit of work.

UDP(User Datagram Protocol): UDP is connectionless protocol. With UDP you send messages (packets) across the network in chunks.

Unreliable - When you send a message, you don't know if it'll get there - it could get lost on the way.

Not ordered - If you send two messages out, you don't know what order they'll arrive in.

Lightweight - No ordering of messages, no tracking connections, etc. It's just fire and forget! This means it's a lot quicker, and the network card / OS have to do very little work to translate the data back from the packets.

Explain how TCP handles reliable delivery (explain ACK mechanism), flow control (explain TCP sender's/receiver's window).

For each TCP packet, the receiver of a packet must acknowledge that the packet is received. If there is no acknowledgement, the packet is sent again. These guarantee that every single packet is delivered. ACK is a packet used in TCP to acknowledge receipt of a packet. A TCP window is the amount of outstanding (unacknowledged by the recipient) data a sender can send on a particular connection before it gets an acknowledgment back from the receiver that it has gotten some of it.

For example, if a pair of hosts are talking over a TCP connection that has a TCP window with a size of 64 KB (kilobytes), the sender can only send 64 KB of data and then it must wait for an acknowledgment from the receiver that some or all of the data has been received. If the receiver acknowledges that all the data has been received, then the sender is free to send another 64 KB. If the sender gets back an acknowledgment from the receiver that it received the first 32 KB (which could happen if the second 32 KB was still in transit or it could happen if the second 32 KB got lost), then the sender could only send another additional 32 KB since it can't have more than 64 KB of unacknowledged data outstanding (the second 32 KB of data plus the third).

Congestion Control

The TCP uses a network congestion avoidance algorithm that includes various aspects of an additive-increase-multiplicative-decrease scheme, with other schemes such as slow-start in order to achieve congestion avoidance.

There are different algorithms to solve the problem; Tahoe and Reno are the most well known. To avoid congestion collapse, TCP uses a multi-faceted congestion control strategy. For each connection, TCP maintains a congestion window, limiting the total number of unacknowledged packets that may be in transit end-to-end. This is somewhat analogous to TCP's sliding window used for flow control. TCP uses a mechanism called slow start to increase the congestion window after a connection is initialized and after a timeout. It starts with a window of two times the maximum segment size (MSS). Although the initial rate is low, the rate of increase is very rapid: for every packet acknowledged, the congestion window increases by 1 MSS so that for every round trip time (RTT), the congestion window has doubled. When the congestion window exceeds a threshold ssthresh the algorithm enters a new state, called congestion avoidance. In some implementations (e.g., Linux), the initial ssthresh is large, and so the first slow start usually ends after a loss. However, ssthresh is updated at the end of each slow start, and will often affect subsequent slow starts triggered by timeouts.

16.1 Imagine you have a call center as follows: pg 51

1. Call center has 3 levels of employees: fresher, technical lead (TL), product manager (PM). There can be multiple employees, but only one TL or PM.
2. An incoming telephone call must be allocated to a Fresher who is free.
3. If a fresher can not handle the call, he or she must escalate the call to technical lead.
4. If TL is not free or not able to handle, then the call should be escalated to PM.

   Design the classes and data structures for this problem. Implement a method getCallHandler().

SOLUTION

All three ranks of employees have different work to be done, so those specific functions are profile specific. We should keep these specific things within their respective class.

There are a few things which are common to them, like address, name, job title, age, etc. These things can be kept in one class and can be extended / inherited by others.

Finally, there should be one CallManager class which would route the calls to concerned person.

```
class Person {
 private:
    Name
    Age
    Address
 public:
    /* Get and set functions for above data members */
}
class Fresher : public class Person {
 private:
    Data job specific;
 public:
    /* Get / set and manipulative functions for class data members */
}
class TL : public class Person {
 private:
    Data job specific;
 public:
    /* Get/set and manipulative functions for class data members */
}
class PM: public class Person {
 private:
    Data job specific;
 public:
    /* Get/set and manipulative functions for class data members */
}
class CallHandler {
 private:
    PM pm;
    TL tl;
    Fresher frsh;
 public:
    getCallHandler();
    /* Get / set methods for data member; */
};

void CallHandler : getCallHandler() {
    if ((this->getFirstAvailableFresher())->ReceiveCall()==False)
    {
        /* Fresher couldn’t handle */
        if ((this->getTL())->Free() == False) {
            /* Tech lead is not free, escalate to the manager */
            (this->getPM())->ReceiveCall();
        }
    }
    return;
}
```

16.2 Design a musical juke box using object oriented principles. pg 51

SOLUTION

Let’s first understand the basic system components:

» CD player // plays the cd and produces the sound output

» CD (song collection) // input to the CD player

» Display () // some display about the length of song, remaining time and playlist display

Now, let’s break this down further:

» Playlist creation (includes add, delete, shuffle etc sub functionalities)

» CD selector

» Track selector

» Queueing up a song

» Get next song from playlist

A user also can be introduced:

» Adding

» Deleting

» Credit information

How do we group these functionalities based on Objects (data + functions which go together)?

Object oriented design suggests wrapping up data with their operating functions in a single entity class.

CONTINUED ON NEXT PAGE

```
Class : JukeBox {
 Data:
    cdPlayer
    trackSelector
    user
    cdCollection
 Functions:
    know current track
    know current student
    waitForUser processOneUser
}
Class: user {
 Data:
    UserInfo
 Functions:
    getUser
    addUser
}

Class CDPlayer {
 Data:
    the physical CD player
    playList
    CD
 Functions:
    playTrack
}

Class: playlist {
 Data:
    track
    queue
 Functions:
    studentCollection
    getNextTrackToPlay
    queueUpTrack
}
```

16.3 Design a chess game using object oriented principles. pg 51

SOLUTION

```
class PositionPotentialValue {
    /* compares value of potential game position */
    bool operator < (const PositionValue& pv); };

class ChessPieceBase {
    virtual void estiationParameter0(); /* used by PositionEstimater 
in different circumstances */
    virtual int estiationParameter1();
    virtual bool canBeChecked();
    virtual bool isSupportCastle();
    // other rule-base properties
};

class King : public ChessPieceBase { … };
class Queen : public ChessPieceBase { … };
// other chess piece classes

class Position { // represents chess positions in compact form
    std::vector<ChessPiece*> black;
    std::vector<ChessPiece*> white;
};

class PositionEstimater {
    // calculate value of a position
    static PositionPotentialValue estimate(const Position& p);
};

class PositionBacktracker {
    // get next postion for estimation.
    static Position getNext(Position& p);
};

class ChessPieceTurn { ... }; // represents move of a chess piece

class PlayerBase {
    virtual ChessPieceTurn getTurn(Position& p);
};

class ComputerPlayer : public PlayerBase {
    // actual implementation
    void setDifficulty();
    PositionEstimater estimater;
    PositionBackTracker backtracter;
};

class HumanPlayer : public PlayerBase {...}; // actual implementation


class ChessFormat {...}; // include info about timing, etc.

class GameManager { // keeps track of time, end of the game, etc
    void processTurn(PlayerBase * player);
    bool acceptTurn(ChessPieceTurn * turn);

    Position currentPostion;
    ChessFormat formant;
    void setGameLog(char * filePath);
};
```

16.4 Design the data structures for a generic deck of cards. Explain how you would subclass it to implement particular card games. pg 51

SOLUTION

```
class Card {
 public:
   enum Suit { CLUBS=0x00, SPADES=0x10, HEARTS=0x20, DIAMONDS=0x30 };
   Card(Suit s, int r) {
       assert(1<=r && r<=13);
       card = int(s)+r;
   }
   virtual ~Card() {}; // just in case
   int rank() const { return card & 0x0F; }
   Suit suit() const { return Suit(card & 0x30); }

 private:
   short int card;
};
```

Assume that we're building a blackjack game, so we need to know the value of the cards (face cards are ten, the ace is 11 (most of the time, but that's the job of the Hand class, not the following class).

```
class BlackJackCard : public Card {
 public:
   BlackJackCard(Suit s, int r) : Card(s, r) {}
   int value() const {
       int r = rank();
       if (r == 1) return 11; // aces are 11
       if (r < 10) return r;
       return 10;
   }
   bool isAce() { return rank() == 1; } // might be useful
}
```

16.5 Design the data structures for an online book reader system. pg 51

SOLUTION

Since the problem doesn't describe much about the functionality, let's assume we want to design a basic online reading system which provides the following functionality:

1. User membership creation and extension.
2. Searching the database of books
3. Reading the books

To implement these we may require many other functions:

» get()

» set()

» update()

» display()

Some of the objects required would be:

» User

» Book

» Library

```
class User {
    User_id;
    User_details;
    Account_type;
    Search_Library();
    Read_book();
    Renew_Membership();
    AddUser();
    Get();
    Set();
};

Book {
    Book_id;
    Book_details;
```

```
        AddBook();
        Update();
        Delete()
        Get();
        Set();
    };

    OnlineReaderSystem {
        Book B;
        User U;
        ListenRequest();
        Search();
        Display();
    };
```

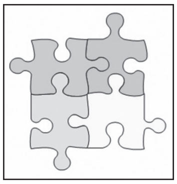
This design is a very simplistic implementation of such a system. We have a class for user to keep all the information regarding the user and an identifier to identify each user uniquely. We can add functionality like registering the user, charging a membership amount and monthly/daily quota, etc.

Next, we have book class where we will keep all the book's information. We would also implement functions like add/delete/update books.

Finally, we have a manager class for managing online book reader system which would have a listen function to listen for any incoming requests for login. It also provides book search functionality and display functionality. Because the end user interacts through this class, search must be implemented here.

16.6 Implement a jigsaw puzzle in C++. Design the data structures and explain an algorithm to solve the puzzle.
pg 51

SOLUTION



```
// Three types of edges on a puzzle piece.
class Edge {
  enum Type {
     inner, outer, flat,
  }
  Piece parent;
  Type type;
  bool fitsWith(Edge type) { ... }; // Inners & outer fit together.
}
class Piece {
  Edge left, right, top, bottom.
  Orientation solvedOrientation = ...; // 90, 180, etc
}
class Puzzle {
  Piece[][] pieces; /* Remaining pieces left to put away. */
  Piece[][] solution;
  Edge[] inners, outers, flats;
  /* We're going to solve this by working our way in-wards, starting with
   * the corners. This is a list of the inside edges. */
  Edge[] exposed_edges;
  void sort() {
     /* Iterate through all edges, adding each to inners, outers,
      * etc, as appropriate. Look for the corners—add those to
      * solution. Add each non-flat edge of the corner to
      * exposed_edges. */
}
class Puzzle {
  ...
  void solve() {
     foreach edge1 in exposed_edges {
        /* Look for a match to edge1 */
        if (edge.type == Edge.Type.Inner) {
           foreach edge2 in outers {
              if edge1.fitsWith(edge2) {
                      /* We found a match! Remove edge1 from exposed_edges.
                       * Add edge2's piece to solution. Check which edges of edge2
                       * are exposed, and add those to exposed_edges.
                    }
                }
                /* Do the same thing, swapping inner & outer. */
            }
        }
    }
}
```

Overview:

1. We grouped the edges by their type. Because inners go with outers, and vice versa, this enables us to go straight to the potential matches.
2. We keep track of the inner perimeter of the puzzle (exposed_edges) as we work our way inwards. exposed_edges is initialized to be the corner's edges.

17.1 You have a basketball hoop and someone says that you can play 1 of 2 games. pg 53

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?

SOLUTION

Probability of winning Game 1: p

Probability of winning Game 2:

Let s(k,n) be the probability of making exactly k shots out of n. The probability of winning game 2 is s(2,3)+s(3,3). Since, s(k,n) = C(n,k) (1-p)^(n-k) p^k, the probability of winning is 3*(1-p)*p^2+p^3.

Simplified, it becomes 3*p^2 - 2*p^3.

You should play Game1 if P(Game1) > P(Game2):

Game 1 is better if : p < 3*p^2 - 2*p^3.

1 < 3*p - 2*p^2

2*p^2 - 3*p + 1 < 0

(p-1)(p-0.5) < 0

Since p can't be greater than one, the first term is negative.

Therefore, the first game is better if p>0.5.

17.2 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? pg 53

SOLUTION

Any of the 3 ants won't collide if all 3 are moving in clockwise direction, or all three are moving in a counter-clockwise direction. Otherwise, there will definitely be a collision.

How many ways are there for the three ants to move? Each ant can move in 2 directions, so there are 2^3 ways the ant can move. There are only two ways which will avoid a collision, therefore the probability of collision is (2^3 – 2) / (2^3) = 6 / 8 = 3 / 4.

To generalize this to an n-vertex polygon: there are still only 2 ways in which the ants can move to avoid a collision, but there are 2^n ways they can move total. Therefore, in general, probability of collision: (2^n – 2) / 2^n = 1 – 1/2^(n-1).

17.3 Numbers are randomly generated and stored in an array. Write a program to find and maintain the median value as new values are generated. pg 53

SOLUTIONS

Solution #1: Keep the array sorted

If we keep the array sorted, we can find the median element in O(1) time. Unfortunately, in order to keep the array sorted, we need to shift elements in the array each time we insert. This shifting means that an insert will take O(n) time.

Solution #2: Keep an additional data structure (a tree)

Use two priority heaps: a max heap for the values below the median, and a min heap for the values above the median. The median will be largest value of the max heap. When a new value arrives it is placed in the below heap if the value is less than or equal to the median, otherwise it is placed into the above heap. The heap sizes can be equal or the below heap has one extra. This constraint can easily be restored by shifting an element from one heap to the other. Median is available in constant time, so updates are O(lg n).

17.4 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 53

SOLUTION

This question is very similar to another problem: how do you randomly shuffle an array? The optimal solution for this problem involves swapping each element in the array with a (later) element in the array:

```
void shuffleArray(int[] array) {
    for (int j = 0; j < array.length; j++) {
        int index = random(j, n); // random number between j and n
        int temp = array[index];
        array[index] = array[j];
        array[j] = temp;
    }
}
```

Note how, in this solution, once the element at spot j has been swapped, it stays put. That is, after m iterations of the for loop, the first m elements in our randomized array have been decided. And, because this shuffling algorithm is "perfect," we know that these m elements are totally random.

```
int[] pickMRandomly(int[] array, int m) {
    int[] subset = new int[m];
    for (int j = 0; j < m; j++) {
        int index = random(j, n); // random number between j and n
        int temp = array[index];
        array[index] = array[j];
        array[j] = temp;
        subset[j] = temp;
    }
    return subset;
}
```

Explanation / Proof :

Each time, we swap the ith position card with a random card between *i* and *n*. So, for the 1st card, there are n possibilities (that is, [1,n]). For the second card, there are n-1 possibilities. For the ith card, there are *n-i* possibilities. So, when we do this 52 times, the number of possible arrangements is 52*51*50*...1 = 52!. Because no arrangement is more likely than another, we know that this must be a perfect shuffle.

17.5 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 53

SOLUTION

Our first instinct on this problem might be to randomly pick elements from the array and put them into our new subset array. But then, what if we pick the same element twice? Ideally, we'd want to somehow "shrink" the array to no longer contain that element. Shrinking is expensive though because of all the shifting required.

Instead of shrinking / shifting, we can swap the element with an element at the beginning of the array and then "remember" that the array now only includes elements j and greater. That is, when we pick subset[0] to be array[k], we replace array[k] with the first element in the array. When we pick subset[1], we consider array[0] to be "dead" and we pick a random element y between 1 and array.size(). We then set subset[1] equal to array[y], and set array[y] equal to array[1]. Elements 0 and 1 are now "dead." Subset[2] is now chosen from array[2] through array[array.size()], and so on.

```
int[] pickMRandomly(int[] array, int m) {
  int[] subset = new int[m];
  for(int j = 0; j < m; ++j) {
     index = random(j, array.size()-1); // random number in [j,n]
     subset[j] = array[index];
     array[index] = array[j]; // array[j] is now "dead"

     /* potentially unnecessary, depending on how much we can
      * modify the array */
     array[j] = subset[j];
  }
  return subset;
}
```

17.6 Write a method to generate a random number between 1 and 7, given a method that generates a random number between 1 and 5. pg 53

SOLUTION

First, observe that we cannot do this in a guaranteed finite amount of time. Why? Let's see by a parallel example: How would you use rand2() to create rand3()?

Observe that each call of rand2() and the corresponding decision you make can be represented by a decision tree. On each node, you have two branches. You take the left one when rand2() equals 0 (which happens with 1/2 probability). You take the right one when rand2() equals 1 (which happens with 1/2 probability). You continue branching left and right as you continue to call 1/2. When you reach a leaf, you return a result of 1, 2 or 3 (your rand3() results).

» What's the probability of taking each branch? 1/2.

» What's the probability to reach a particular leaf node? 1/2^j (for some j).

» What the probability of returning 3 (for example)? We could compute this by summing up the probabilities of reaching each leaf node with value 3. Each of these paths has probability 1/2^j, so we know that the total probability of returning 3 must be a series of terms of reciprocal powers of 2 (eg, 1/2^x + 1/2^y + 1/2^z + …).

We also know, however, that the probability of returning 3 must be 1/3 (because rand3() should be perfectly random). Can you find a series of reciprocal powers of 2 that sum to 1/3? No, because 3 and 2 are relatively prime.

So… we can similarly conclude that to solve this problem, we will need to accept a small (infinitesimally small) chance that this process will repeat forever. That's ok.

So, how do we solve this?

In order to generate a random number between 1 and 7, we just need to uniformly generate a larger range than we are looking for and then repeatedly sample until we get a number that is good for us. We will generate a base 5 number with two places with two calls to the RNG.

```
int rand7() {
  while (1) {
     int num = 5*(rand5()-1) + rand5() - 1;
     if (num < 21) return num % 7;
  }
}
```

18.1 Picture a computer screen with multiple windows and a mouse. Each window can be represented as a rectangle, and the mouse is represented by a (x, y) coordinate. If you click on the screen, the top most window should become active. Describe an algorithm to return the top most window when the mouse is clicked. pg 55

SOLUTION

There are several approaches with some trade-offs.

Approach 1:

For every window, check if the point is contained in it. If so, keep pointer to the window. Then for all such window pointers, find the window with lowest of z-order. Return this window.

This solution will run in O(N) time and will require no [well, O(1)] preprocessing time and O(N) storage. If we need to find such a window only once, that is the optimal algorithm. But, if we have to process multiple queries this solution can be slow. So there exists another approach:

Approach 2:

Preprocessing: Gather all unique x-coordinates of window corners, keep them in X_array of size m. Sort them. Do the same for y-coordinates, keeping them in Y_array of size n. Create m*n array L of references for window.

For each window, use binary search to find x_min, x_max, y_min, y_max, such that for all i : x_min <= i <= x_max and j : y_min <= j <= y_max, point P(X_array[i], Y_array[j]) is contained in the window. For each such pair (i, j), add a reference to the window from L[i, j] if L[i, j] is empty or if z-order of window is less than z-order of existing element.

Preprocessing requires O(N^2) space and O(N^3) processing time, but now we can do very efficient searches with this structure.

Search: For every query point P(x, y) find with binary search i, j : X_array[i] <= x <= X_array[i + 1]; Y_array[j] <= y <= Y_array[j + 1] and return L[i, j] as answer.

18.2 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 55

SOLUTION

Let's assume we have some scripts which are scheduled to get the data via FTP at the end of the day. Where do we store the data? How do we store the data in such a way that we can do various analyses of it?

Proposal #1

Keep the data in text files . This would be very difficult to manage and update, as well as very hard to query. Keeping unorganized text files would lead to a very inefficient data model.

Proposal #2

We could use a database. This provides the following benefits:

» Logical storage of data.

» Facilitates an easy way of doing query processing over the data.

Example: return all stocks having open > N AND closing price < M

Advantages:

» Makes the maintenance easy once installed properly.

» Roll back, backing up data, and security could be provided using standard database features. We don't have to "reinvent the wheel."

CONTINUED ON NEXT PAGE

Proposal #3

If requirements are not that broad and we just want to do a simple analysis and distribute the data, then XML could be another good option.

Our data has fixed format and fixed size: company_name , open, high, low, closing price. The XML could look like this:

```
<root>
<date value="2008-10-12">
      <company name="foo">
         <open>126.23</open>
         <high>130.27</close>
         <low>122.83</low>
         <closing price>127.30
      </company>
      <company name="bar">
         <open>52.73</open>
         <high>60.27</close>
         <low>50.29</low>
         <closing price>54.91
      </company>
   </date>
   <date value="2008-10-11"> . . . </date>
</root>
```

Benefits:

» Very easy to distribute. In fact, nowadays, XML is the standard data model to share / distribute data across the web.

» Efficient parsers are available to parse the data and extract out only desired data.

» We can add new data to the XML file by carefully appending data. We would not have to re-query the database.

However, querying of the data could be difficult.

18.3 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.pg55

SOLUTION

For data block allocation, we can use bitmask vector and linear search (see "Practical File System Design") or B+ trees (see Wikipedia).

```
#include <map>
#include <vector>
#include <string>
struct DataBlock {
  char data[DATA_BLOCK_SIZE];
};
DataBlock dataBlocks[NUM_DATA_BLOCKS];
struct INode {
  std::vector<int> datablocks;
};
struct MetaData {
  int size;
  Data last_modifed;
  Data created;
  char extra_attributes;
};
std::vector<bool> dataBlockUsed(NUM_DATA_BLOCKS);
std::map<string, INode *> mapFromName;
sruct FSBase;
struct File : public FSBase {
private:
  std::vector<INode> * nodes;
  MetaData metaData;
};
```

```
struct Directory : pubic FSBase; {
  std::vector<FSBase* > content;
};

struct FileSystem {
  init();
  mount(FileSystem*);
  unmount(FileSystem*);
  File createFile(cosnt char * name) { /* modifies MetaData */ }
  Directory createDirectory (const char * name) { /* modified MetaData */ }
  void openFile(File * file, FileMode mode) {
     // mapFromName to find INode corresponding to file
  }
  void closeFile(File * file) { … }
  void writeToFile(File * file, void * data, int num) {
     // Modifies all underlying structures.
     // Searches for next available datablock with dataBlockUsed
     // Modifies INode structure
     // Modifies MetaData
     // Accesses datablock using INode structure
  }
  void readFromFile(File * file, void * res, int numbutes, int postion) {
     // Accesses datablock using INode structure
  }
};
```

18.4 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 55

SOLUTION

What is our chat server?

This is something you should discuss with your interviewer, but let's make a couple of assumptions: imagine we're designing a basic chat server that needs to support a small number of users. People have a contact list, they see who is online vs offline, and they can send text-based messages to them. We will not worry about supporting group chat, voice chat, etc. We will also assume that contact lists are mutual: I can only talk to you if you can talk to me. Let's keep it simple.

What specific actions does it need to support?

» User A signs online

» User A asks for their contact list, with each person's current status.

» Friends of User A now see User A as online

» User A adds User B to contact list

» User A sends text-based message to User B

» User A changes status message and/or status type

» User A removes User B

» User A signs offline

What can we learn about these requirements?
We must have a concept of users, add request status, online status, and messages.

What are the core components?
We'll need a database to store items in and an "always online" application as the server. We might recommend using XML for the communication between the chat server and the clients, as it's easy for a person and a machine to read.

CONTINUED ON NEXT PAGE

What are the key objects and methods?

We have listed the key objects and methods below. Note that we have hidden many of the details, such as how to actually push the data out to a client.

```
enum StatusType {
  online, offline, away;
}

class Status {
  StatusType status_type;
  string status_message;
}

class User {
  string username;
  string display_name;
  User[] contact_list;
  AddRequest[] requests;
  bool updateStatus(StatusType stype, string message) { … };
  bool addUserWithUsername(string name);
  bool approveRequest(string username);
  bool denyRequest(string username);
  bool removeContact(string username);
  bool sendMessage(string username, string message);
}
/* Holds data that from_user would like to add to_user
class AddRequest {
  User from_user;
  User to_user;
}
class Server {
  User getUserByUsername(string username);
}
```

What problems would be the hardest to solve (or the most interesting)?

Q1 *How do we know if someone is online—I mean, really, really know?*

While we would like users to tell us when they sign off, we can't know for sure. A user's connection might have died, for example. To make sure that we know when a user has signed off, we might try regularly pinging the client to make sure it's still there.

Q2 *How do we deal with conflicting information?*

We have some information stored in the computer's memory and some in the database. What happens if they get out of sync? Which one is "right"?

Q3 *How do we make our server scale?*

While we designed out chat server without worrying—too much– about scalability, in real life this would be a concern. We'd need to split our data across many servers, which would increase our concern about out of sync data.

Q4 *How we do prevent denial of service attacks?*

Clients can push data to us—what if they try to DOS us? How do we prevent that?

18.5 Describe the data structures and algorithms that you would use to implement a garbage collector in C++. pg 55

SOLUTION

In C++, garbage collection with reference counting is almost always implemented with smart pointers, which perform reference counting. The main reason for using smart pointers over raw ordinary pointers is the conceptual simplicity of implementation and usage.

With smart pointers, everything related to garbage collection is performed behind the scenes - typically in constructors / destructors / assignment operator / explicit object management functions.

There are two types of functions, both of which are very simple:

```
RefCountPointer::type1() {
  /* implementation depends on reference counting organisation.
   * There can also be no ref. counter at all (see approach #4) */
  incrementRefCounter(); }

RefCountPointer::type2() {
  /* Implementation depends on reference counting organisation.
   * There can also be no ref. counter at all (see approach #4).
  decrementRefCounter();
  if (referenceCounterIsZero()) {
     destructObject();
  }
}
```

CONTINUED ON NEXT PAGE

There are several approaches for reference counting implementation in C++:

1. Simple reference counting.

```
struct Object { };
struct RefCount {
  int count;
};
struct RefCountPtr {
  Object * pointee;
  RefCount * refCount;
};
```

Advantages: performance.

Disadvantages: memory overhead because of two pointers.

2. Alternative reference counting.

```
struct Object { … };
struct RefCountPtrImpl {
  int count;
  Object * object;
};
struct RefCountPtr {
  RefCountPtrImpl * pointee;
};
```

Advantages: no memory overhead because of two pointers.

Disadvantages: performance penalty because of extra level of indirection.

CONTINUED ON NEXT PAGE

3. Intrusive reference counting.

```
struct Object { … };
struct ObjectIntrusiveReferenceCounting {
  Object object;
  int count;
};
struct RefCountPtr {
  ObjectIntrusiveReferenceCounting * pointee;
};
```

Advantages: no previous disadvantages.

Disadvantages: class for intrusive reference counting should be modified.

4. Ownership list reference counting. It is an alternative for approach 1-3. For 1-3 it is only important to determine that counter is zero—its actual value is not important. This is the main idea of approach # 4.

All Smart-Pointers for given objects are stored in doubly-linked lists. The constructor of a smart pointer adds the new node to a list, and the destructor removes a node from the list and checks if the list is empty or not. If it is empty, the object is deleted.

```
struct Object { };
struct ListNode {
  Object * pointee;
  ListNode * next;
}
```

19.1 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 57

SOLUTION

```
/* Input: a[], b[], n (number of elements in a) and
 * m (number of elements in b)
int k = m + n - 1; // Index of last location of array b
int i = n - 1; // Index of last element in array b
int j = m - 1; // Index of last element in array a

// Start comparing from the last element and merge a and b
while (i >= 0 && j >= 0) {
  if (a[i] > b[j]) {
     a[k--] = a[i--];
  } else {
     a[k--] = b[j--];
  }
}
while (j >= 0) {
  a[k--] = b[j--];
}
```

Note: You don't need to copy the contents of a after running out of b's. They are already in place.

19.2 Write a method to sort an array of strings so that all the anagrams are next to each other.
pg 57

SOLUTION

The basic idea is to implement a normal sorting algorithm where you override the compareTo method to compare the "signature" of each string. In this case, the signature is the alphabetically sorted string.

```
public int compareTo(String that) {
  String sig1 = this.sort(); // sort the characters in string 1
  String sig2 = that.sort(); // sort the characters in string 2
  return sig1.compareTo(sig2);
}
```

Now, just sort the arrays, using this compareTo method instead of the usual one.

19.3 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. pg 57

SOLUTION

Assumptions:

A is the array. l and u are lower and upper indexes of the array. x is the key that we want to search.

We can do this with a modification of binary search.

```
int search(int a[], int l, int u, int x) {
  while (l <= u) {
     int m = (l + u) / 2;
     if (x == a[m]) {
        return m;
     } else if (a[l] <= a[m]) {
        if (x > a[m]) {
           l=m+1;
        } else if (x >=a [l]) {
           u = m-1;
        } else {
           l = m+1;
        }
     }
     else if (x < a[m]) u = m-1;
     else if (x <= a[u]) l = m+1;
     else u = m-1;
  }
  return -1;
}
```

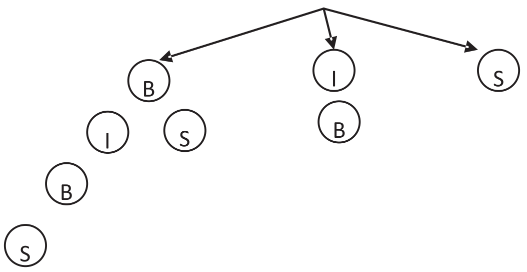
NOTE: What about duplicates? You may observe that the above function doesn't give you an efficient result in case of duplicate elements. However, if your array has duplicate entries then we can't do better than O(n) which is as good as linear search.

For example, if the array is [2,2,2,2,2,2,2,2,3,2,2,2,2,2,2,2,2,2,2], there is no way to find element 3 until you do a linear search.

19.4 Given a string s and an array of smaller strings, T, design a method to search s for each small string in T. pg 57

SOLUTION

First, create a suffix tree for s. For example, if your word were bibs, you would create the following tree:



Then, all you need to do is search for each string in T in the suffix tree. Note that if "B" were a word, you would come up with two locations.

For more details on suffix tree and implementation with source code see:

- *http://mila.cs.technion.ac.il/~yona/suffix_tree/suffix_tree.h*
- *http://mila.cs.technion.ac.il/~yona/suffix_tree/suffix_tree.c*
- *http://mila.cs.technion.ac.il/~yona/suffix_tree/project_report.rtf*

19.5 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 57

SOLUTION

When an interviewer gives a size limit of 2GB, it should tell you something - in this case, it suggests that they don't want you to bring all the data into memory.

So what do we do? We only bring part of the data into memory..

Algorithm:

How much memory do we have available? Let's assume we have X MB of memory available.

1. Divide the file into K chunks, where X * K = 2 GB. Bring each chunk into memory and sort the lines as usual using any O(n log n) algorithm. Save the lines back to the file.
2. Now bring the next chunk into memory and sort.
3. Once we're done, merge them one by one.

The above algorithm is also known as external sort. Step 3 is known as N-way merge

The rationale behind using external sort is the size of data. Since the data is too huge and we can't bring it all into memory, we need to go for a disk based sorting algorithm.

20.1 Write an algorithm to implement a queue using two stacks. pg 59

SOLUTION

As we know, pushing elements onto a stack and then popping all of them will cause the elements to appear in reverse order. But, if we repeat this process again, the elements will be back in their original order.

Thus, the implementation of insert and delete functions of a queue are somewhat like this:

```
Insert(int value):
  // Push onto stack1
  stack1.push(value);
Delete():
```

To return the bottom-most element of the stack, we do the following:

```
if (!stack2.empty()) {
  /* stack2 was already filled by previous calls of dequeue, so just
   * retrieve the next item.
  return stack2.pop();
}

while (!stack1.empty()) {
  /* We know that stack2 is empty, so fill it with the next portion
   * of data.
  stack2.push(stack1.pop());
}
return stack2.pop(); // retrieve the next item.
```

20.2 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 59

SOLUTION

You can implement this by having each node in the stack keep track of the minimum beneath itself. Then, to find the min, you just look at what the top element thinks is min.

When you push an element onto the stack, the element is given the current minimum. It sets its "local min" to be the min(previous minimum, my value).

```
class NodeWithMin inherits Node {
  int min; ...
}

class StackWithMin inherits Stack {
  void push(int value) {
     int new_min = Math.min(value, min())
     Node n = new Node(value, new_min);
     super.push(n);
  }

  int min() {
     if (peek() == null) {
        return MIN_INT;
     } else {
        return peek().min;
     }
  }
}
```

There's just one issue with this: if we have a large stack, we waste a lot of space by keeping track of the min for every single element. Can we do better?

CONTINUED ON NEXT PAGE

We can (maybe) do a bit better than this by using an additional stack which keeps track of the mins.

```
class StackWithMin {
  Stack s1, s2;
  ...

  void push(int value) {
     if (value <= min()) {
        S2.push(value);
     }
     s1.push(value);
  }

  int pop() {
     int value = s1.pop();
     if (value == min()) {
        s2.pop();
     }
  }

  int min() {
     if (s2.peek() == null) {
        return MIN_INT;
     } else {
        return s2.peek();
     }
  }
}
```

Why might this be more space efficient? If many elements have the same local min, then we're keeping a lot of duplicate data. By having the mins kept in a separate stack, we don't have this duplicate data (although we do use up a lot of extra space because we have a stack node instead of a single int).

20.3 Describe how you could use a single array to implement three stacks. pg 59

SOLUTION

Approach 1:

Divide the array in three equal parts and allow the individual stack to grow in that limited space (note: [means inclusive, while (means exclusive of the end point).

» for stack 1, we will use [0, n/3)

» for stack 2, we will use [n/3, 2n/3)

» for stack 3, we will use [2n/3 n)

This solution is based on the assumption that we do not have any extra information about the usage of space by individual stacks and that we can't either modify or use any extra space. With these constraints, we are left with no other choice but to divide equally.

Approach 2:

In this approach, any stack can grow as long as there is any free space in the array.

We sequentially allocate space to the stacks and we link new blocks to the previous block. This means any new element in a stack keeps a pointer to the previous top element of that particular stack.

In this implementation, we face a problem of unused space. For example, if a stack deletes some of its elements, the deleted elements may not necessarily appear at the end of the array. So, in that case, we would not be able to use those newly freed spaces.

To overcome this deficiency, we can maintain a free list and the whole array space would be given initially to the free list. For every insertion, we would delete an entry from the free list. In case of deletion, we would simply add the index of the free cell to the free list.

In this implementation we would be able to have flexibility in terms of variable space utilization but we would need to increase the space complexity.

20.4 Write a C 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: pg 59

Push | Pop | Top | IsEmpty | IsFull

SOLUTION

Sorting can be performed with one more stack. The idea is to pull an item from the original stack and push it on the other stack. If pushing this item would violate the sort order of the new stack, we need to remove enough items from it so that it's possible to push the new item. Since the items we removed are on the original stack, we're back where we started. The algorithm is O(N^2) and appears below.

```
void sort_stack(Stack * src, Stack * dest) {
  while (!src->IsEmpty()) {
     int tmp = src->Pop();
     while(!dest->IsEmpty() && dest->Top() > tmp) {
        src->Push(dest->Pop());
     }
     dest->Push(tmp);
  }
}
```

20.5 The Towers of Hanoi is a classical mathematical puzzle in which you have N rods and K disks of different sizes which can slide onto any rod. The puzzle starts with disks sorted in ascending order of size from top to bottom (eg, each disk sits on top of an even larger one). You have the following constraints:pg59

1. Only one disk can be moved at a time.
2. A disk is slid off of the top of one rod onto the next rod.
3. A disk can only be placed on top of a larger disk.

In programming terms, this can be simulated with stacks and the operations remove-block, put-block, findblock, isempty, isfull. Write a program to sort the disks in ascending order, given 10 disks and 5 rods.

SOLUTION

First of all, you should know that there is solution for any K when N >= 3. Let us first consider case N = 3.

```
void move_disks(int from, int to, int N) {
    int found = find_rod(from);
    move_disks(from, found, N - 1);
    move_disks(from, to, 1);
    move_disk(found, to, N -1)
}
solution call : move_disks(0, 1, K).
```

Now that you can see the pattern, here is a solution for arbitrary constant K:

```
void move_disks(int from, int to, int N) {
    int found = find_rod(from);
    move_disks(from, found, N - K + 2);
    std::vector<ind> found_v;
    for (int i = 0; i < K - 1; ++i) {
        found_v.push_back(find_rod(from));
        move_disks(from, found_v.back(), 1);
    }
    move_disks(from, to, 1);
    for (int i = K - 2, i >= 0; --i) {
        move_disks(found_v[i], to, 1);
    }
    move_disks(found, to , N - K + 2);
}
solution call : move_disks(0, 1, K);
```

21.1 Write a method to replace all spaces in a string with '%20'. pg 61

SOLUTIONS

The algorithm is as follows:

1. Count the number of spaces during the first scan of the string.
2. Parse the string again from the end and for each character:
   - » If a space is encountered, store "%20".
   - » Else, store the character as it is in the newly shifted location.

```
// Assume parsed string has sufficient free space at the end
ReplaceFun(char * str, int length) {
  int SpaceCount = 0, NewLength, i=0;
  for (i = 0; i < length; i++) {
    if (str[i] == ' ') {
      SpaceCount++;
    }
  }
  NewLength = length + SpaceCount * 2;
  Str[NewLength] = '\0';
  for (i = length – 1, i >= 0; i--) {
    if (str[i] == ' ') {
      str[NewLength -1 ] = '0';
      str[NewLength -2 ] = '2';
      str[NewLength -3 ] = '%';
      NewLength = NewLength -3;
    } else {
      str[NewLength - 1] = str[i];
      NewLength = NewLength -1;
    }
  }
}
```

21.2 Given an integer between 0 - 999,999, print an English phrase that describes the integer (eg, “One Thousand, Two Hundred and Thirty Four.”) pg 61

SOLUTION

```
// Input: Number to be converted to word string
// len: Number of digits in num
// str: Buffer for result
void numtostring(int num , int len, char str[]) {
  char *wordarr1[] = {“”,“One ”, “Two ”, “Three ”, “Four ”,
                      “Five ”, “Six ”, “Seven ”, “Eight ”,”Nine ”};
  char* wordarr11[] ={“”, “Eleven ”, “Twelve ”, “Thirteen ”,
                      “Fourteen ”, “Fifteen ”, “Sixteen ”,
                      “Seventeen ”, “Eighteen ”, “Nineteen ”};
  char* wordarr10[] = {“”,“Ten ”, “Twenty ”, “Thirty ”, “Forty ”,
                      “Fifty ”, “Sixty ”, “Seventy ”, “Eighty ”,
                      “Ninety ”};
  char* wordarr100[] = {“”, “Hundred ”, “Thousand ”};
  int tmp;

  if (num == 0) {
     strcat(str , “Zero”);
  } else {
     if (len > 3 && len % 2 == 0) {
        len++;
     }
     do {
        // Number greater than 999
        if (len > 3) {
           tmp = (num / (int)pow(10,len-2));
           // If tmp is 2 digit number and not a multiple of 10
           if (tmp / 10 == 1 && tmp%10 != 0) {
              strcat(str, wordarr11[tmp % 10] ) ;
           } else {
              strcat(str, wordarr10[tmp / 10]);
```

```
                strcat(str, wordarr1[tmp%10]);
            }
            if(tmp) {
                strcat(str, wordarr100[len / 2]);
            }
            num = num % (int)(pow(10,len-2));
            len = len-2;
        } else { // Number is less than 1000
            tmp = num / 100;
            if (tmp != 0) {
                strcat(str, wordarr1[tmp]);
                strcat(str, wordarr100[len / 2]);
            }
            tmp = num % 100 ;
            if(tmp/10 == 1 && tmp % 10 != 0) {
                strcat(str, wordarr11[tmp % 10] ) ;
            } else {
                strcat(str, wordarr10[tmp/10]);
                strcat(str, wordarr1[tmp%10]);
            }
            len = 0;
        }
    } while(len > 0);
  }
}
```

21.3 Write a method to decide if two strings are anagrams or not. pg 61

SOLUTION

There are two easy ways to solve this problem:

Solution #1: Sort the strings

```
bool anagram(string s, string t) {
  return s.sort() == t.sort()
}
```

Solution #2: Check if the two strings have identical counts for each unique char.

```
bool anagram(string s, string t) {
  if (s.length() ! = t.length()) return false;
  int letters[256] = /* initialize to 0 */;
  int num_unique_s = 0;
  int num_completed_t = 0;
  foreach (char c in s) { // count number of each char in s.
     if (letters[c] == 0) ++num_unique_chars;
     ++letters[c];
  }
  for (int i = 0; i < t.length(); ++i) {
     int c = (int)t[i];
     if (letters[c] == 0) { // We found more of char c in t than in s.
        return false
     }
     --letters[c];
     if (letters[c] == 0) {
        ++num_completed_t;
        if (num_completed_t == num_unique_s) {
           // it’s a match if t has been processed completely
           return i == t.length() - 1;
        }
     }
  }
}
```

21.4 Implement an algorithm to determine if a string has all unique characters. What if you can not use additional data structures? pg 61

SOLUTION

For simplicity, assume char set is ASCII (if not, we need to increase the storage size. The rest of the logic would be the same). *NOTE*: This is a *great* thing to point out to your interviewer!

```
bool isChar(string str) {
    bool char_set[256];
    for (int i = 0; i < 256; ++i) char_set[i] = false;
    for (int i = 0; i < str.size(); ++i) {
        int val = str[i];
        if (char_set[val]) return true;
        char_set[val] = true;
    }
    return false;
}
```

Time complexity is O(n), where n is the length of the string, and space complexity is O(n).

Potential improvement: Use a bit vector to hash the presence of a char.

If we are not allowed to use additional space, we could do the following:

1. Check every char of the string with every other char of the string for duplicate occurrences. This will take O(n^2) time and no space.
2. If we are allowed to destroy the input string, we can modify quicksort to look for duplicates.

Modified quicksort algorithm:

```
bool isChar(string str, int first, int last) {
    bool isChar(string str, int first, int last) {
    if (first < last) return false;
    if (first == last) return false;
    if (str[first] == str[last]) return true;
    int pivot = partition(str, first, last); // standard partition
    if (isChar(str, first, pivot - 1)) return true;
    if (0 < pivot && str[pivot - 1] == str[pivot]) return true;
    if (isChar(str, pivot + 1, last)) return true;
    if (pivot < last && str[pivot] == str[pivot + 1]) return true;
}

bool isChar(string str) {
    return isChar(str, 0, str.length() - 1);
}
```

21.5 Given a sorted array of strings which is interspersed with empty strings, write a method to find the start location of a given string. pg 61

```
Example: find “bc” in [“a”, “”, “”, “”, “”, “b”, “c”, “”, “”, “”,
“d”, “”]
```

SOLUTION

Use ordinary binary search, but when you hit an empty string, advance to the next non-empty string; if there is no next non-empty string, search the left half.

```
int search(char **strings, char * str, int first, int last) {
  while (first <= last) {
     // Ensure there is something at the end
     while (first <= last && !strings[last]) --last;
     if (last < first) return -1; // this block was empty, so fail
     int mid = (last + first) >> 1;
     while (!strings[mid]) ++mid; // will always find one
     int r = strcmp(strings[mid], str);
     if (r == 0) return mid;
     if (r < 0) {
        first = mid + 1;
     } else {
        last = mid - 1;
     }
  }
}
```

21.6 Code: Reverse C-Style String. (C-String means that “abcd\n” is actually represented as six characters)
pg 61

SOLUTION

```
#include <stdio.h>
/************************************************
Name: - reverse
Arguments: - pointer to string to be reversed
************************************************/
void reverse(char *str) {
  char * end = str;
  char tmp;
  if (str) {
     while (*end) ++end;
     --end;
     while (str < end) {
        tmp = *str;
        *str++ = *end;
        *end-- = tmp;
     }
  }
}
```

21.7 Given two strings, s1 and s2, write code to check if s2 is a rotation of s1 using only one call to strstr (eg, "waterbottle" is a rotation of "erbottlewat"). pg 61

SOLUTION

Just do the following checks

1. Check if length(s1) == length(s2). If not, return false.
2. Else, concatenate s1 with s1 and see whether s2 is substring of the result.

   input: s1 = apple, s2 = pleap ==> apple is a substring of pleappleap

   input: s1 = apple, s2 = ppale ==> apple is not a substring of ppaleppale

```
boolean isRotation(const char *s1, const char *s2) {
  if strlen(s1) == strlen(s2) {
     return strstr(strcat(s1,s1),s2);
  }
  return false;
}
```

21.8 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: pg 61

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 encode xml element (as char *) as Byte *

FOLLOW UP

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

SOLUTION

Part 1: Solution

```
#include <sstream>
#include <string>
#include <map>

using namespace std;
typedef map<string, int> TagMapType;

void addKeyValuePair(const string& keyValue, stringstream& where,
TagMapType& tagMap) {
  int eq_pos = keyValue.find_first_of('=');
  where << keyValue.substr(0, eq_pos) << " ";
  int key_start = keyValue.find_first_of('=', eq_pos);
  int key_end = keyValue.find_last_of('=', keyValue.length() - 1);
  where << tagMap[keyValue.substr(key_start, key_end - key_start)];
}
```

CONTINUED ON NEXT PAGE

```
string encode_xml_element(const char * original_element, TagMapType&
                          tagMap, const char * END) {
  char inChar;
  string inString;
  stringstream inputStream(stringstream::in | stringstream::out);
  stringstream outputStream(stringstream::in);

  inputStream << original_element;

  inputStream >> inChar;
  inputStream >> inString ; // Element

  outputStream << inString << " ";
  bool cont = true;
  while (cont) {
     inputStream >> inString;
     if (*inString.rbegin() == '>') {
        cont = false;
     }
     if (!cont) {
        if (inString.length() == 1) {
           break;
        }
     }
     addKeyValuePair(inString, outputStream, tagMap);
  }
  outputStream << END;
  return outputStream.str();
}
```

Part 2: Is there anything you can do to compress this further?

You can treat the file as a general stream of characters and use any number of compression techniques: Shannon–Fano coding, Huffman coding or Arithmetic coding.

21.9 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. pg 62

```
EXAMPLE:
input: DAMP, LIKE
output: DAMP -> LAMP -> LIMP -> LIME -> LIKE
```

SOLUTION

The algorithm is fairly straightforward. We basically do a breadth-first search, where each word branches to all words in the dictionary that are one edit away. The interesting part is how to implement this—should we build a graph as we go? We could, but there's an easier way. We can instead use a "backtrack map." In this backtrack map, if B[v] = w, then you know that you edited v to get w. When we reach our end word, we can use this backtrack map repeatedly to reverse our path. See the code below:

```
start: the word we're starting with
stop: the word we're ending with
Q: action queue—words that we must still explore
V: visited map—maps from word to bool
B: backtrack map (word to word)-

Insert start word into Q and V.
While Q is not empty do
   Dequeue word w from Q
   For each possible word v from w with one edit operation
      If v == stop
         Create a linked list called list
         Append v to list
         While w is not the start word
            w = B[w]
            Prepend v to list
         Return list
      If v is a dictionary word
         If not Visited[v]
            Enqueue v in Q
            Visited[v] = true
            B[v] = w
   Failure, since we've visited everything reachable from the start.
```

Let n be the length of the start word and m be the number of like sized words in the dictionary. The runtime of this algorithm is O(n m) since the while loop will dequeue at most m unique words. The for loop is O(n) as it walks down the string applying a fixed number of replacements for each character.

21.10 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 62

```
EXAMPLE:
  input:
     file: as was is the as the yahoo you me was the and
     words: was, as
  output: 2
```

SOLUTION

Assumption: The order of words matters. If that is not the case, we can easily repeat the search and flip the order of the words.

Traverse the file and for every occurrence of word1 and word2, compare difference of positions and update the current minimum.

```
pos = 0;
min = MAX_INTEGER/2;
word1_pos = word2_pos = -min;

while (!end of file) {
  current_word = getword();
  if (current_word == word1) {
     word1_pos = pos;
     // uncomment if word order doesn't matter
     // distance = word1_pos - word2_pos;
     // if (min > distance)
     // min = distance;
  } else if (current_word == word2) {
     word2_pos = pos;
     distance = word2_pos - word1_pos;
     if (min > distance)
        min = distance;
  }
  ++pos;
}
```

21.11 Write a program to find the longest word made of other words. pg 62

```
EXAMPLE:
input: test, tester, testertest, testing, testingtester
output (longest word): testingtester .
```

SOLUTION

The solution below does the following:

1. Sort the array by size, putting the longest word at the front
2. For each word, split it in all possible ways. That is, for "test", split it into {"t", "est"}, {"te", "st"} and {"tes", "t"}.
3. Then, for each pairing, check if the first half and the second both exist elsewhere in the array.
4. "Short circuit" by returning the first string we find that fits condition #3.

What is the time complexity of this?

» Time to sort array: O(n log n)

» Time to check if first / second half of word exists: O(d) per word, where d is the average length of a word.

» Total complexity: O(n log n + n * d). Note that d is fixed (probably around 5—10 characters). Thus, we can guess that for short arrays, the time is estimated by O(n * d) , which also equals O(number of characters in the array). For longer arrays, the time will be better estimated by O(n log n).

» Space complexity: O(n).

Optimizations: If we didn't want to use additional space, we could cut out the hash table. This would mean:

» Sorting the array in alphabetical order

» Rather than looking up the word in a hash table, we would use binary search in the array

» We would no longer be able to short circuit.

CODE ON NEXT PAGE

```
#include <iostream>
#include <conio.h>
#include <algorithm>
#include <vector>
#include <string>
typedef vector<string> VS;
using namespace std;

typedef vector<string> VS;

void print_longest_made_from_two(VS&v) {
  /* create a lookup table to efficiently figure out if a word is in
the array */
  std::map<std::string, bool> lookup;
  for (int i = 0; i < v.size(); ++i) {
     lookup[v[i]] = true;
  }
  /* sort words by size */
  sortWordsBySize(v);

  for (int i = 0; i < v.size(); ++i) {
     const string&s = v[i];
     // Split a word in all possible ways
     for (int j = 1; j < s.length() - 1; ++j) {
        std::string first = s.substr(0, j); // first half std::string
        second = s.substr(j, s.length() - j); // second half
        if (lookup[first] && lookup[second]) {
           cout << s;
        }
     }
  }
  cout << “No such word”;
}
```

22.1 How would you test a pen? pg 64

SOLUTION

To illustrate the technique in this problem, let us guide you through a mock-conversation.

Interviewer: How would you test a pen?

Candidate: Let me find out a bit about the pen. Who is going to use the pen?

Interviewer: Probably children.

Candidate: Ok, that's interesting. What will they be doing with it? Will they be writing, drawing, or doing something else with it?

Interviewer: Drawing.

Candidate: Ok, great. On what? Paper? Clothing? Walls?

Interviewer: On clothing.

Candidate: Great. What kind of tip does the pen have? Felt? Ball point? Is it intended to wash off, or is it intended to be permanent?

Interviewer: It's intended to wash off.

…. many questions later ...

Candidate: Ok, so as I understand it, we have a pen that is being targeted at 5—10 year olds. The pen has a felt tip and comes in red, green, blue and black. It's intended to wash off clothing. Is that correct?

…

The candidate has now a problem that is significantly different from what it initially seemed to be. Thus, the candidate might now want to test:

1. Does the pen wash off with warm water, cold water, and luke warm water?
2. Does the pen wash off after staying on the clothing for several weeks? What happens if you wash the clothing while the pen is still wet?
3. Is the pen safe (eg—non-toxic) for children?

and so on...

22.2 How would you test an ATM in a distributed banking system? pg 64

SOLUTION

The first thing to do on this question is to clarify assumptions. Ask the following questions:

» Who is going to use the ATM? Answers might be "anyone," or it might be "blind people" - or any number of other answers.

» What are they going to use it for? Answers might be "withdrawing money," "transferring money," "checking their balance," or many other answers.

» What tools do we have to test? Do we have access to the code, or just the ATM machine?

Remember: a good tester makes sure she knows what she's testing!

Here are a few test cases for how to test just the withdrawing functionality:

» Withdrawing money less than the account balance

» Withdrawing money greater than the account balance

» Withdrawing money equal to the account balance

» Withdrawing money from an ATM and from the internet at the same time

» Withdrawing money when the connection to the bank's network is lost

» Withdrawing money from multiple ATMs simultaneously

22.3 How would you load test a webpage without using any test tools? pg 64

SOLUTION

Load testing helps to identify a web application's maximum operating capacity, as well as any bottlenecks that may interfere with its performance. Similarly, it can check how an application responds to variations in load.

To perform load testing, we must first identify the performance-critical scenarios and the metrics which fulfill our performance objectives. Typical criteria include:

» response time

» throughput

» resource utilization

» maximum load that the system can bear.

Then, we design tests to simulate the load, taking care to measure each of these criteria.

In the absence of formal testing tools, we can basically create our own. For example, we could simulate concurrent users by creating thousands of virtual users. We would write a multi-threaded program with thousands of threads, where each thread acts as a real-world user loading the page. For each user, we would programmatically measure response time, data I/O, etc.

We would then analyze the results based on the data gathered during the tests and compare it with the accepted values.

22.4 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 64

SOLUTION

There are two primary types of testing we should do:

Validation of input/output:

We should validate both the input and output to make sure that each are valid. This might entail:

1. Checking whether input is within the board limit
   - » Attempt to pass in negative numbers
   - » Attempt to pass in x which is larger than the width
   - » Attempt to pass in y which is larger than the width

   Depending on the implementation, these should either return false or throw an exception.
2. Checking if output is within the valid set of return values. (Not an issue in this case, since there are no "invalid" Boolean values).

Functional testing:

Ideally, we would like to test every possible board, but this is far too big. We can do a reasonable covering of boards however. There are 6 pieces in chess, so we need to do something like this:

```
foreach piece a:
  for each other type of piece b (6 types + empty space)
     foreach direction d
        Create a board with piece a.
        Place piece b in direction d.
        Try to move – check return value.
```

22.5 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 64

SOLUTION

The question largely depends on the type of application being diagnosed. However, we can give some general causes of random crashes.

1. Random variable: The application uses some random number or variable component which may not be fixed for every execution of the program. Examples include: user input, a random number generated by the program, or the time of day.
2. Memory Leak: The program may have run out of memory. Other culprits are totally random for each run since it depends on the number of processes running at that particular time. This also includes heap overflow or corruption of data on the stack.

It is also possible that the program depends on another application / external module that could lead to the crash. If our application, for example, depends on some system attributes and they are modified by another program, then this interference may lead to a crash. Programs which interact with hardware are more prone to these errors.

In an interview, we should ask about which kind of application is being run. This information may give you some idea about the kind of error the interviewer is looking for. For example, a web server is more prone to memory leakage, whereas a program that runs close to the system level is more prone to crashes due to system dependencies.

23.1 What’s the difference between a thread and a process? pg 66

SOLUTION

Processes and threads are related to each other but are fundamentally different.

A process can be thought of as an instance of a program in execution. Each process is an independent entity to which system resources (CPU time, memory, etc.) are allocated and each process is executed in a separate address space. One process cannot access the variables and data structures of another process. If you wish to access another process’ resources, inter-process communications have to be used such as pipes, files, sockets etc.

A thread uses the same stack space of a process. A process can have multiple threads. Key difference between processes and threads is that multiple threads share parts of their state. Typically, one allows multiple threads to read and write the same memory (no processes can directly access the memory of another process). However, each thread still has its own registers and its own stack, but other threads can read and write the stack memory.

A thread is a particular execution path of a process; when one thread modifies a process resource, the change is immediately visible to sibling threads.

23.2 How can you measure the time spent in a context switch? pg 66

SOLUTION

This is a tricky question, but let's start with a possible solution.

A context switch is the time spent switching between two processes (eg, bringing a waiting process into execution and sending an executing process into waiting/terminated state). This happens in multitasking. The operating system must bring the state information of waiting processes into memory and save the state information of the running process.

In order to solve this problem, we would like to record timestamps of the last and first instruction of the swapping processes. The context switching time would be the difference in the timestamps between the two processes.

Let's take an easy example: Assume there are only two processes, P1 and P2.

P1 is executing and P2 is waiting for execution. At some point, the OS must swap P1 and P2—let's assume it happens at the Nth instruction of P1. So, the context switch time for this would be Time_Stamp(P2_1) – Time_Stamp(P2_N)

Easy enough. The tricky part is this: how do we know when this swapping occurs? Swapping is governed by the scheduling algorithm of the OS. We can not, of course, record the timestamp of every instruction in the process.

Another issue: there are many kernel level threads which are also doing context switches, and the user does not have any control over them.

Overall, we can say that this is mostly an approximate calculation which depends on the underlying OS. One approximation could be to record the end instruction timestamp of a process and start timestamp of a process and waiting time in queue.

If the total timeof execution of all the processes was T, then the context switch time = T – (SUM for all processes (waiting time + execution time)).

23.3 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 66

SOLUTION

```
using namespace std;
/* Place holder for thread synchronization lock */
class Lock {
public:
  Lock() { /* placeholder code to create the lock */ }
  ~Lock() { /* placeholder code to deallocate the lock */ }
  void AcquireLock() { /* placeholder to acquire the lock */ }
  void ReleaseLock() { /* placeholder to release the lock */ }
};

/* Singleton class with a method that creates a new instance of the
 * class of the type of the passed in template if it does not
 * already exist. */
template <class T> class Singleton {
 private:
  static Lock lock;
  static T* object;
 protected:
  Singleton() { };
 public:
  static T * instance();
};
```

CONTINUED ON NEXT PAGE

```
Lock Singleton::lock;

T * Singleton::Instance() {
  /* if object is not initialized, acquire lock */
  if (object == 0) {
     lock.AcquireLock();
     /* If two threads simultaneously check and pass the first "if"
      * condition, then only the one who acquired the lock first should
      * create the instance */
     if (object == 0) {
        object = new T;
     }
     lock.ReleaseLock();
  }
  return object;
}

int main() {
  /* foo is any class defined for which we want singleton access
  Foo* singleton_foo = Singleton<foo>::Instance(); */
  return 0;
}
```

The general method to make a program thread safe is to lock shared resources whenever write permission is given. This way, if one thread is modifying the resource, other threads can not modify it.

23.4 Design a class which provides a lock only if there are no possible deadlocks.
pg 66

SOLUTION

For our solution, we implement a wait/die deadlock prevention scheme.

```
// Wait/Die
// O - older thread
// Y - younger thread
// O needs a resource held by Y O waits
// Y needs a resource held by O Y dies

const int NUM_RESOURCES = 100;
Resource rscs[NUM_RESOURCES];
Thread thds[2];

bool can_acquire_resource(Thread * t, Resource * r) {
  Thread * ot = &thds[0] == t ? &thds[1] : &thds[0];
  if (!is_resource_acquired(ot, r)) {
     return true;
  }
  if (timestapm(t) < timestamp(ot)) {
     // t is older
     os_wait_for_resource(t, r);
     return true;
  } else {
     // t is younger
     os_finish_process(t);
     return false;
  }
}
```

23.5 Suppose we have the following code: pg 66

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

SOLUTION

PART A

i) Can you explain a multithread synchronization mechanism?

In computer science, a semaphore is a protected variable or abstract data type which constitutes the classic method for restricting access to shared resource.

```
P(Semaphore s) { // Acquire Resource
  wait until s > 0, then s := s-1;
  /* Testing and decrementing s must be atomic to avoid race
   * conditions */
}
V(Semaphore s) { // Release Resource
  s := s+1; /* must be atomic */
}
Init(Semaphore s, Integer v) {
  s := v;
}
```

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

```
Semaphore s_a(0);
Semaphore s_b(0)
A {
  /***/
  s_a.release(1);
}
B {
  s_a.acquire(1);
  /****/
  s_b.release(1)
}
C {
  s_b.acquire(1)
  /******/
}
```

PART B

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(.....);
```

Question continued on following page

PART B

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

```
Semaphore s_a(0);
Semaphore s_b(0);
Semaphore s_c(1);
A {
  s_c.acqure(1);
  /***/
  s_a.release(1);
}
B {
  s_a.acqure(1);
  /****/
  s_b.release(1);
}
C {
  s_b.acqure(1);
  /******/
  s_c.release(1);
}
```

24.1 Given a sorted (increasing order) array, write an algorithm to create a binary tree with minimal height. pg 68

SOLUTION

We will try to create a binary tree such that for each node, the number of nodes in the left subtree and the right subtree are equal, if possible.

Algorithm:

1. Insert into the tree the middle element of the array.
2. Insert (into the left subtree) the left subarray elements
3. Insert (into the right subtree) the right subarray elements
4. Recurse

```
struct Node {
  int data;
  struct Node * left;
  struct Node * right;
};

struct Node * CreateBinaryTree(int *array, int leftIndex, int rightIndex) {
  if (rightIndex < leftIndex) {
     return NULL;
  }
  int current = leftIndex + rightIndex;
  Node * tree = (Node*)malloc(sizeof(struct Node));
  tree->data = array[current];
  tree->left = CreateBinaryTree(array, leftIndex, current - 1);
  tree->right = CreateBinaryTree(array, current + 1, rightIndex);
  return tree;
}
```

24.2 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 68

SOLUTION

The idea is very simple: the difference of min depth and max depth should not exceed 1, since the difference of the min and the max depth is the maximum distance difference possible in the tree.

```
int maxDepth(Tree * root) {
  if (!root) {
     return 0;
  }
  return 1 + max(maxDepth(root->left), maxDepth(root->right));
}
int minDepth(Tree * root) {
  if (!root) {
     return 0;
  }
  return 1 + min(minDepth(root->left), minDepth(root->right));
}
bool isBalanced(Tree * root) {
  return maxDepth(root) - minDepth(root) <= 1
}
```

24.3 Design an algorithm and write code to find the first common ancestor of two nodes in a binary search tree. Avoid storing additional nodes in a data structure. pg 68

SOLUTION

Attempt #1:

If each node has a link to its parent, we could trace p and q's paths up until they intersect.

Attempt #2:

Alternatively, you could follow a chain in which p and q are on the same side. That is, if p and q are both on the left of the node, branch left to look for the common ancestor. When p and q are no longer on the same side, you must have found the first common ancestor.

```
public Tree commonAncestor(Tree root, Tree p, Tree q) {
  if (covers(root.left, p) && covers(root.left, q))
     return commonAncestor(root.left, p, q);
  if (covers(root.right, p) && covers(root.right, q))
     return commonAncestor(root.right, p, q);
  return root;
}
private boolean covers(Tree root, Tree p) { /* is p a child of root? */
  if (root == null) return false;
  if (root == p) return true;
  return covers(root.left, p) || covers(root.right, p);
}
```

What is the running time of this algorithm? One way of looking at this is to see how many times each node is touched. *Covers* touches every child node, so we know that every single node in the tree must be touched at least once, and many nodes are touched multiple times.

Attempt #3:

For any node r, we know the following:

1. If p is on one side and q is on the other, r is the first common ancestor.
2. Else, the first common ancestor is on the left or the right side.

So, we can create a simple recursive algorithm called search that calls search(left side) and search(right side) looking how many nodes (p or q) are placed from the left side and from the right side of the current node.

If there are 2 nodes on one of sides, then we have to check, if the child node on this side is p or q (because in this case the current node is the common ancestor). If the child node is neither p nor q, we should continue search further (starting from the child).

If one of searched nodes (p or q) is located on the right side of the current node, then the other node is located on the other side, so the current node is the common ancestor.

```
// Checks how many nodes are located under this root
public static int covers(Node root, Node p, Node q) {
  int ret = NO_NODES_FOUND;
  if(root == null) return ret;
  if(root == p || root == q) ret += 1;
  ret += covers(root.left, p, q);
  if(ret == TWO_NODES_FOUND) // Found p and q
     return ret;
  return ret + covers(root.right, p, q);
}
public static Node commonAncestor(Node root, Node p, Node q) {
   if(q == p && (root.left == q || root.right == q)) {
     return root;
  }
  // Check how many nodes are located on the left side
  int nodesFromLeft = covers(root.left, p, q);
  if(nodesFromLeft == TWO_NODES_FOUND) {
     if(root.left == p || root.left == q) return root;
     else return commonAncestor(root.left, p, q);
  }
  // Check how many nodes are located on the right side
  int nodesFromRight = covers(root.right, p, q);
  if(nodesFromRight == TWO_NODES_FOUND) {
     if(root.right == p || root.right == q) return root;
     else return commonAncestor(root.right, p, q);
  }
  if (nodesFromLeft == ONE_NODE_FOUND && nodesFromRight == ONE_NODE_
FOUND) return root;
  else return null;
}
```

24.4 Write an algorithm to find the 'next' node (eg, in-order successor) of a given node in a binary search tree where each node has a link to its parent. pg 68

SOLUTION

We approach this problem by thinking very, very carefully about what happens on in-order traversal. On in-order traversal, we visit X.left, then X, then X.right.

So, if we want to find X.successor(), we do the following:

1. If X has a right child, then the successor must be on the right side of X (because of the order in which we visit nodes). Specifically, the left-most child must be the first node visited in that subtree.

2. Else, we go to X's parent (call it P).

2.a. If X was a left child (P.left = X), then P is the successor of X

2.b. If X was a right child (P.right = X), then we have fully visited P, so we call successor(P).

```
struct node* inorderSucc(struct node* e) {
   if (e != NULL) {
      struct node* p ;
      // found right children -> return 1st inorder node on right
      if (e->parent == NULL || e->right != NULL) {
         p = leftMostChild(e->right);
      } else {
         // Go up until we're on left instead of right (case 2b)
         while (p = e->parent) {
            if (p->left == e) break;
            e = p;
         }
      }
      return p;
   }
   return NULL;
}

struct node* leftMostChild(struct node * e) {
   while (e->left) e = e->left;
   return e;
}
```

24.5 Given a directed graph, design an algorithm to find out whether there is a route between two nodes. pg 68

SOLUTION

This problem can be solved by just simple graph traversal, such as depth first search or breadth first search. We start with one of the two nodes and, during traversal, check if the other node is found. We should mark any node found in the course of the algorithm as 'already visited' to avoid cycles and repetition of the nodes.

```
void search(Graph g, Node start, Node end) {
  Queue q = new Queue()
  foreach Node u in g.vertices
     state[u] = NOT_VISITED
  state[start] = VISITING
  q.enqueue(start)
  while (!q.isEmpty()) {
     u = q.dequeue()
     foreach v in u.adjacent_vertices() {
        if state[v] == NOT_VISITED {
           if (v == end) return true
           else {
              state[v] = VISITING
              q.enqueue(v)
           }
        }
     }
     state[u] = VISITED
  }
  return false;
}
```

24.6 How would you design the data structures for a very large social network (Facebook, Linked In, etc)? Describe how you would design an algorithm to show the connection, or path, between two people (eg, Me -> Bob -> Susan -> Jason -> You). pg 68

SOLUTION

Approach:

Forget that we're dealing with millions of users at first. Design this for the simple case.

We can construct a graph by assuming every person is a node and if there is an edge between two nodes, then the two people are friends with each other.

```
class Person {
  Person[] friends;
  // Other info
}
```

If I want to find the connection between two people, I would start with one person and do a simple breadth first search.

But... oh no! Millions of users!

When we deal with a service the size of Orkut or Facebook, we cannot possibly keep all of our data on one machine. That means, that our simple Person data structure from above doesn't quite work—our friends may not live on the same machine as us. Instead, we can replace our list of friends with a list of their IDs, and traverse as follows:

1. For each friend ID
2. int machine_index = lookupMachineForUserID(id);
3. Go to machine machine_index
4. Person friend = lookupFriend(machine_index);

There are more optimizations and follow up questions here than we could possibly discuss, but here are just a few thoughts.

Optimization: Reduce Machine Jumps

Jumping from one machine to another is expensive. Instead of randomly jumping from machine to machine with each friend, try to batch these jumps—eg, if 5 of my friends live on one machine, I should look them up all at once.

Optimization: Smart Division of People and Machines

People are much more likely to be friends with people who live in the same country as them. Rather than randomly dividing people up across machines, try to divvy them up by country, city, state, etc. This will reduce the number of jumps.

Question: Breadth First Search usually requires "marking" a node as visited. How do you do that inthis case?

Usually, in BFS, we mark a node as visited by setting a flag visited in its node class. Here, we don't want to do that (there could be multiple searches going on at the same time—it's bad to just edit our data). In this case, we could mimic the marking of nodes with a hashtable to lookup a node id and whether or not it's been visited.

Other Follow-Up Questions:

» In the real world, servers fail. How does this affect you?

» How could you take advantage of caching?

» Do you search until the end of the graph (infinite)? How do you decide when to give up?

» In real life, some people have more friends of friends than others, and are therefore more likely to make a path between you and someone else. How could you use this data to pick where you start traversing?

24.7 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 68

SOLUTION

Let's approach this problem by simplifying it. What if the path had to start at the root? In that case, we would have a much easier problem:

> Start from the root and branch left and right, computing the sum thus far on each path. When we find the sum, we print the current path. Note that we don't stop just because we found the sum! Why? Because we could have the following path (assume we are looking for the sum 5): 2 + 3 + –4 + 3 + 1 + 2. If we stopped once we hit 2 + 3, we'd miss several paths (2 + 3 + -4 + 3 + 1, and 3 + -4 + 3 + 1 + 2). So, we keep going along every possible path.

Now, what if the path can start anywhere? In that case, we make a small modification. On every node, we look "up" to see if we've found the sum. That is—rather than asking "does this node start a path with the sum?," we ask "does this node complete a path with the sum?"

```
void findsum(node* head, int sum, int[] buffer, int level) {
    if (*head == NULL) return;
    int temp = sum;
    buffer[level] = head->data;

    /* Look up through the path we've traversed to see if our path
     * equals sum */
    for (int i = level; i >= 0; i--) {
        temp = temp - buffer[i];
        if (temp == 0) {
            /* We've found sum by starting at the current node and
             * counting upwards until level i *.
            print(buffer, level, i);
        }
    }
    findsum(head->left, sum, buffer, left + 1); // traverse left
    findsum(head->right, sum, buffer, left + 1); // traverse right
}
void print(int* buf, int start, int end) {
    for(int i = start; i<=end; i++) {
        printf(buffer[i]);
    }
}
```

What is the time complexity of this algorithm? Well, if a node is at level r, we do r amount of work (that's in the looking "up" step). We can take a guess at O(n lg n) (n nodes, doing an average of lg n amount of work on each step), or we can be super mathematical:

» There are 2^r nodes at level r.

» 1*2^1 + 2*2^2 + 3*3^2 + 4*2^4 + … d * 2^d

  = sum(r * 2^r, r from 0 to depth)

  = 2 (d-1) * 2^d + 2

» n = 2^d ==> d = lg n

» NOTE: 2^lg(x) = x

» O(2 (lg n - 1) * 2^(lg n) + 2) = = O(2 (lg n - 1) * n) = O(n lg n)

Following similar logic, our space complexity is O(n lg n).

24.8 Given a directed graph, find a minimal set of vertices which touch all edges within a graph. pg 68

A vertex is said to 'touch' an edge if the edge either originates or terminates with that vertex.

SOLUTION

NOTE: This is a classic NP complete problem vertex cover, so we cannot solve it in polynomial time complexity.

One way to solve this is to generate all possible combination of nodes and checking if it covers all the edges in the graph. Pick the set which has a minimum number of vertices.

Algorithm:

```
min_size = Infi
set_index = -1;
1.  Generate all possible vertex set S
2.  for all set si in S
3.  if union of edges covered by each vertex in set si == E
        if (si.size() < min_size)
                min_size=si.size()
                set_index = i;
4.  return i;
5.  End
```