How to Pass a Silicon Valley Software Engineering Interview

Paul Tyma

Get Notes : http://paultyma.blogspot.com

Software Development
March,2008

Who am I

- Day job: Senior Engineer, Google
  - Over 200 engineering interviews
  - Resume reviewer
  - Hiring committee member
- Founder, former CEO, Preemptive Solutions
  - Dotfuscator, Dasho
- Founder, Manybrain, inc.
  - Mailinator.com

What do I know about all this?

- If you're good at interviewing, you likely get a job fast and thus don't get a lot of experience
- If you are an interviewer, you only see one, limited side of the story
- Lots of war stories on the net
  - (from often rather bitter people)
- My (latest) experience
  - Weeks of research/studying
  - 3 interviews

About this talk

- This talk is not about resume making
- This talk is not a secret key to cheating on interviews
- Questions are all over the net
  - (everything here is either original or referenced on the net)
- This is (somewhat) language neutral
- Part Experiential, Academic, Practical

About this talk

- Some explanation about the subjects in this talk
- Interviewing in the valley in general
- Startup Economics
- Types of problems
  - Some on language, but not a ton
  - Some on puzzles
  - Some on bit twiddling
  - Algorithms and Data structures

Silicon Valley Interviews

- Might/might-not be the same as a lot of places
- Social Networks are hot
  - (search was in 2006)
- Big, big, huge, massive, large amounts of data
  - Algorithms and data structures matter
- Understanding of memory usage, disk i/o and performance in general
- Problem solving (as always)

Why the Focus on all this stuff I learned in school?

- This is **not** about all that stuff you forgot in school
- Response time needs to be in milliseconds
- Choosing the wrong algorithm can be the difference between getting or not getting a new user
- Turns out its quite easy to write software that doesn't need to scale

Facebook

- What was Facebook day 1?
  - A database with a PHP front-end
- In PHP, Java, C#, whatever
  - How long would it take you to reproduce Facebook's first incarnation?
- A single Mysql instance with some simple queries probably used to happily query the whole userbase.

Facebook

- What is it today?
- Its not about “that stuff you learned in school”
  - Its about what a company with thousands of (possibly conflicting) queries per second operating on a directed-graph with 50million nodes
    - And of course a few Petabytes of data
    - And 99.99% uptime
- Design decision? A facebook user is (or recently was) currently limited to 5000 friends.

This is about Software Engineering

- A.K.A – coding, programming, web backends, web 2.0
- And the semi-harder-ish-core-ish stuff
- Startup environment
  - And to make money – or to pretend to make money until you can get bought out

Interviewing rules and some incidental fluffy stuff

Job Change Rule #1

(my rule anyway)

- If you are the smartest person where you work... Quit.
- (it might make sense that if you're the dumbest person where you work, you should quit too, but for all different reasons)
- Lots of smart people here
- Seems to me the older I get, the more people I find that are smarter than me
  - Maybe I'm getting dumber
  - Or maybe, this is largely for the same reason you remember your elementary school gymnasium as utterly immense. As I get older, it gets smaller.

Job Change Rule #2

- Do **not** interview for your dream job first
  - Plan on bombing and learning from your first interview
- Your chances are cut significantly
  - You don't know what to expect
  - Not only from the “what to say” category, but right down to the “what to wear”, “when to show up”, “being surprised by who's interviewing you”

Where

Google, Facebook, Slide, Admob

- Consider the technology
- Consider the success chance of the company
- Management
- Revenue
- Consider the Culture (and company size)
- Consider the buyout potential

Strategy: Interviewing

- Basics:
  - Discuss your work
  - Be excited to be there
  - Tell why you want to work there
  - Tell why you can bring them value
  - Never disrespect current employer
  - Exemplify non-work external projects
  - Extracurricular Coding projects are good

Strategy: Interviewing

- Interviewer Mantra
  - Would you want to work with this person?
  - They want smart people – bottom line
- Passionate
- Friendly
- No chips on any shoulders

Typically, interviewing is “honest”

- It's not about your school
- It's not about your resume
- It's not about your experience
- It's about what you can do.
  - Right now.
  - On the whiteboard.
- Of course, good “everything” doesn't hurt
- If the first 3 are great, they'll show up in the 4th

Experiential:

"Hardest" Interview that I had

- 5 page programming assignment just to get the interview
  - Implement Google's pagerank algorithm in Java
  - Implement a k-differences spellchecker based on 2 research papers in C++
  - Implement a webcrawler (i.e. Bot) in Java using breadth-first, depth-first, and some algorithm of your design
- 4 hour interview with 9 developers, CEO, CTO

Startup Economics

(I have seen these numbers – all deals vary)

- *PLEASE evaluate your own deal pursuant to the situation. Numbers here for example only.*
- Pre-funding < 10 employees
  - CTO: 1 to 5%
  - Senior engineer: 0.5%
- Notes
  - “Number of Shares” is irrelevant
  - Titles surprisingly aren't at this stage
  - You will be diluted

Super Rough Estimates

(I have seen these numbers – all deals vary)

- Once VC comes in:
  - Engineers will get all of about 15% of the company to split
- Assuming you're joining an early stage for a big payoff
  - Try for more stock – salary won't make you a big score. Company is cash-tight, maybe offer lower salary in exchange
  - Ask now – this is the only time you can assume your CEO will have little negotiating savvy

Super Rough Estimates

(I have seen these numbers – all deals vary)

- I still see "*I've got a great idea for a startup – now I just need a few engineers*"
- Engineers are still often considered commodities
- After series A venture funding, your prospects go notably down
  - Company can sell for 50million and you get nothing
- They have money now. Salaries will be easier on them. Stock grants are minuscule (to them)

Super Rough Estimates
(I have seen these numbers – all deals vary)

- That doesn't apply if you find a Facebook or Google
  - And call me if you do
- If company valuation went down from series A to series B, owners are heavily diluted
  - Any share you get is wee
  - Company has grown up (although still risky) – focus more on career and experience
  - Big “title jobs” are probably taken and fiercely defended

Interview Questions

Main Types of Questions

- Language
  - APIs? java.lang.Object – sure. STL – probably.
  - APIs after that they are not so important*
    - They're APIs, they are designed to be easy to use
    - * unless the job is in specific APIs
    - Creating APIs is quite hard
- Algorithms/Data Structures
- “No Answer” problem solving

Types of Questions

- Possibly answerable problem solving
- Bits, bytes, and twiddling
- Philosophies
  - Testing, Agile
    - Maybe how to use JUnit
- System Design
  - Design Patterns
  - OO Design
- If its on your resume, it's fair game

Language Questions

Languages

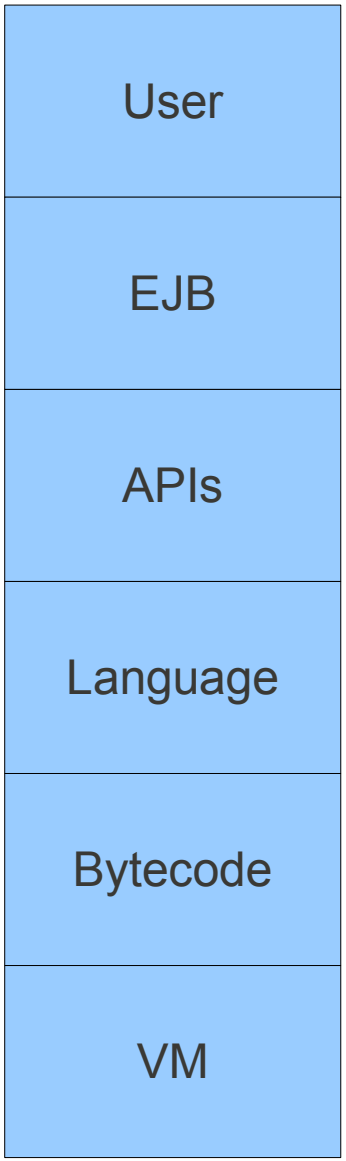
- Reasonable questions are basically anything a developer would likely know after using a language for a year or so.
  - Describe memory leak gotchas using the STL
  - Can you have memory leaks in Java?
  - What does wait, notify, and notifyAll do in java.lang.Object?
  - What is a static method?
- Onsite Interview vs. Phone Screen language questions

Languages

- Java, C++, Python, Groovy, Ruby (Rails)
- But I can learn it !
  - Language master doesn't happen in a few weeks
  - maybe some APIs
- Java
  - Where do you fit?
    - J2EE gluer
    - API ninja
    - Language master
    - Bytecode twiddler

Where do you fit?
(java example)

- Hard to know everything
- Know your strengths
- Position yourself to use them



Binary search the difficulty/level of questions

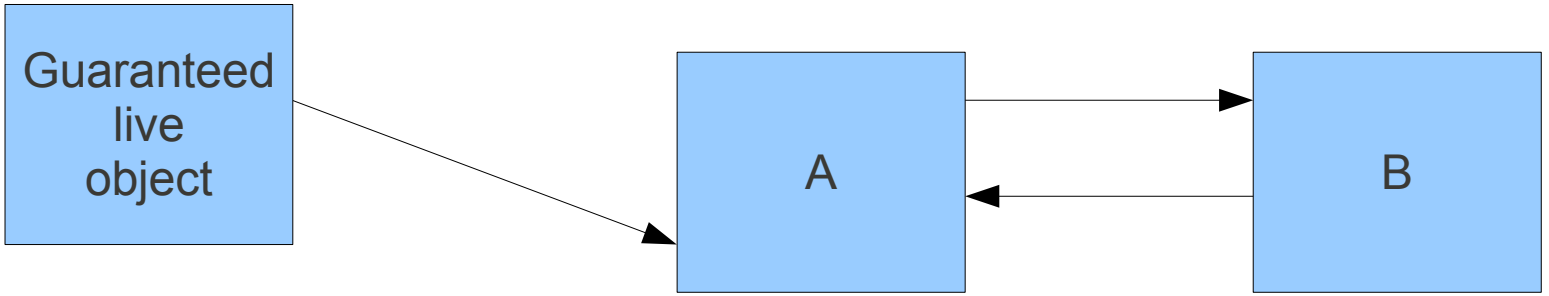
- What does wait and notify do?
  - Going up – What are architectural reasons to use multithreading? (interactivity, efficiency, etc)
  - Going down – Whats the difference between notify and notifyAll?
  - More down – How does (or how should in your opinion) a VM decide who to wake up with notify?

C/C++

- Classic
  - Reverse a string in place
  - Sort of bit twiddling
- Virtuality
- Performance
- Memory and leaks
- Joel:
  - Does a given point lie within a rectangle?

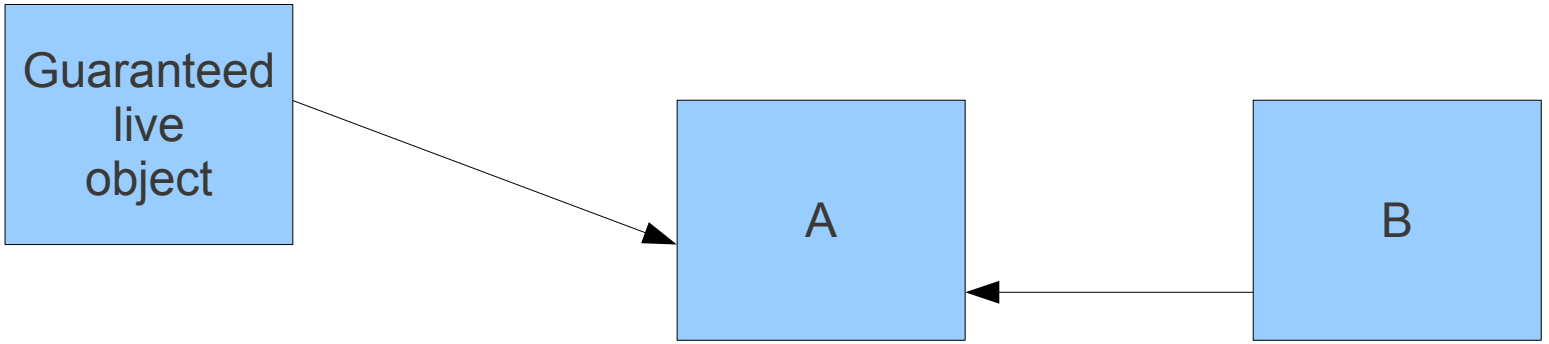
Hidden Depth Question

- In Java, what is “finalize()”?
  - Let's say B has one



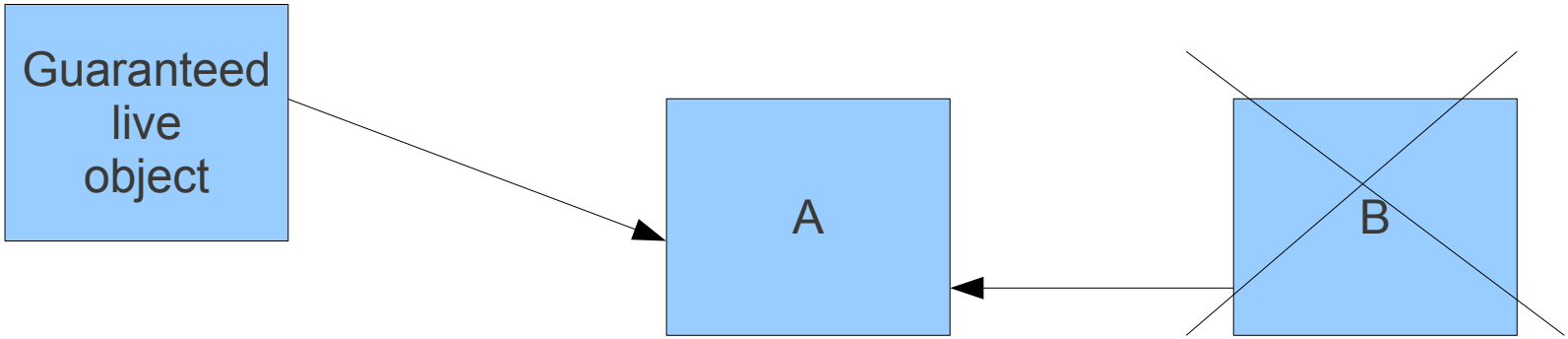
Hidden Depth Question

- Now, is B garbage?
  - GC runs
  - Runs B's finalizer
  - Destroys B



Hidden Depth Question

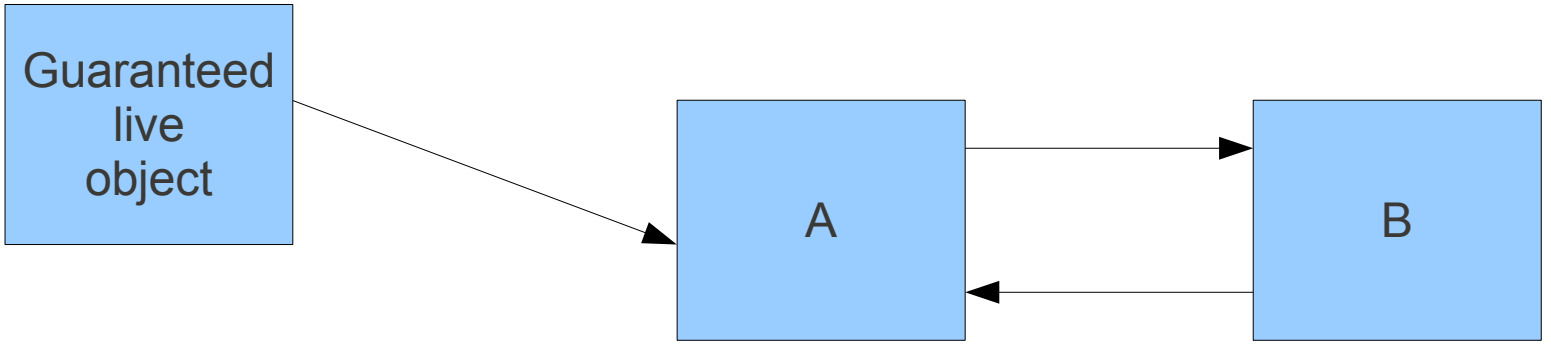
- Now, is B garbage?
  - GC runs
  - Runs B's finalizer
  - Destroys B



Hidden Depth Question

```
public void finalize() {
   a.b = this;
   }

   How would YOU solve this problem?
```



Java `final`

`simple on the outside, gooey on the inside`

- Normal answers:
  - cannot change final variables
  - cannot override final methods
  - cannot subclass final classes
- I've heard:
  - cannot change a method
  - cannot send values to final methods
  - final class is the last class written – the program is done!

Java `final`

- Deeper
  - Say we want `final double pi = 3.14159`
    - should we make that static or not?
    - Can you tell me when we'd want a static final vs. an instance final?
  - Final methods can't be overridden
    - Can you think of any performance implications of this?
  - When should you make fields (or methods) final in general?

Java `final`

local vars

- Do you know what an inner class is?
- Can method contained inner classes access local variables?

```
public Bar foo(int x) {   // compiler error
   return new Bar() {
      public void doLittle() {
         System.out.println(x);
         }
      }
   }
```

Strategy: Languages

- Be honest on your skills – too easy to prove
  - (ignore this if you're interviewing for management)
- Know the basics of a language
  - Could you be a C++ dev without fully understanding pointers?
  - Could you be a Java dev without knowing java.lang.Object inside out?
  - How GC algorithms work? Not likely. Identify garbage objects in a graph? Yes.

Strategy: Languages

- Language learning comes from doing
- Love whatever language you're interviewing for
- Coding on your own time is a strong indicator
- Open source
  - Always an avenue to do real work in a language

Philosophy and Systems

Strategy: Philosophy

- Many developers are religious about something
    - (maybe thats the most obvious statement in the talk)
  - a language
    - ruby, groovy, python, erlang
  - a development team strategy
    - agile, TDD, pair programming
- Leave your attitude at home
- Anything that is highly popular must have some good points, acknowledge them regardless of your own biases

Strategy: Philosophy

- Stay Neutral
- Unless....
  - Your interviewer is a raging zealot
  - Then become his disciple
    - (or her zealot, but in my experience, more men tend to be raving lunatics about some computing philosophy)
- Or stay neutral – loving is better than hating

Strategy: Systems

- How would you design XYZ?
  - Design an Airline reservation system
  - Maybe be overly encapsulated with verbal qualification
  - Java vs. C++
- How would you test XYZ?
- Give a use case for XYZ

Strategy: Systems

- What's your favorite Design Pattern?
- Singleton
  - Can you describe how to implement it?
  - Ok, so it enforces only one instance
    - why not just use an entirely static class then?

Algorithms

A small (but happy) refresher

Running Times

(informal enough to make an algorithms professor wince)

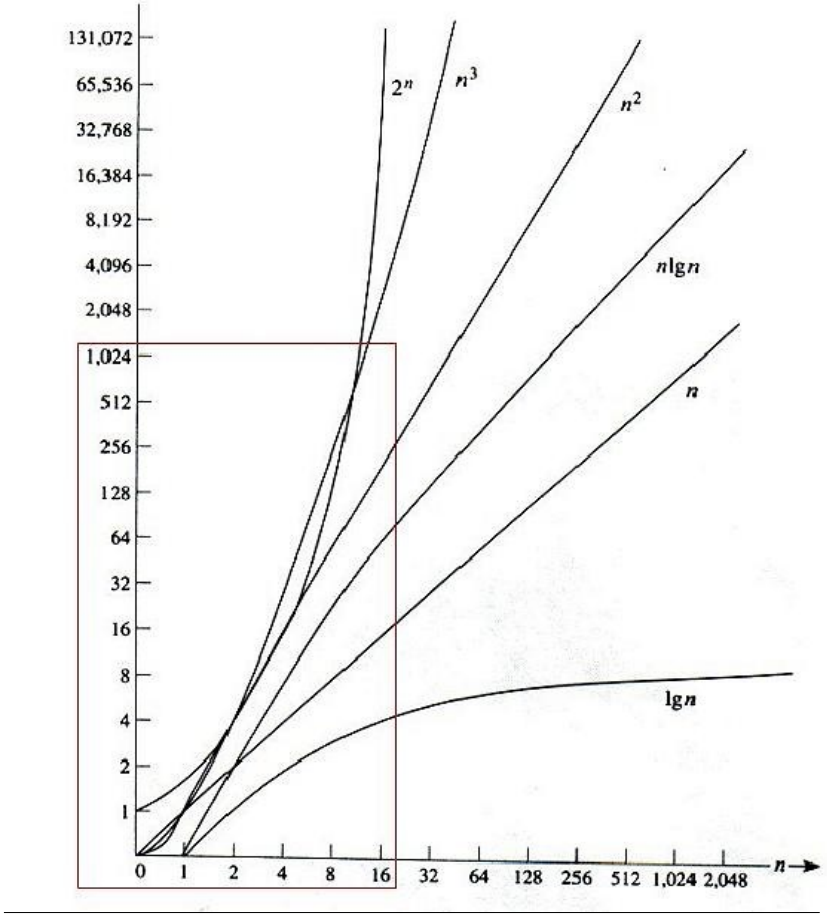
- Big 'O' notation
  - Best, Worst, Average, Expected
- Less about how it runs, more about how it grows as input grows
- Short Version:
  - On the order of ...
- Long Version:
  - As the input dataset grows in size, the number of actions required to complete the task will grow similarly to the function ...

Running Times

- What's a faster algorithm:
  - O(n^2)
  - O(n log n)
- Somewhat like asking, what's faster:
  - a car
  - or a tricycle
- But won't O(n^2) eventually be slower than O(n log n) as n gets bigger?

Running Times

- Common running times
  - (for interview questions anyway)
- O(n)
- O(log n)
- O(n log n)
- O(n^2)
- O(1)
- O(2^n)



Real Life Running Time

- O(1) – constant (or O(2) or O(99) or O(203843))
  - Return the first item in a shopping list
  - What if the shopping list had 100 items
  - What if it had a bazillion?
  - Generally we mean “average case”
    - Here however, best, worst and average are the same

Real Life Running Time

- O(log n)
  - n is the number of names in the phone book
  - Looking up a number in a phone book
  - How many pages must you look-at to find the # ?
  - In the average case?
  - In the best case?
  - In the worst case?

Logarithms give me the Willies

- O(log n)
  - Every step breaks the problem in two
  - Finding a specific number in a set of already sorted numbers
  - Weighing something using scales
- Divide and conquer

Real Life Running Time

- O(m)
  - m is the number of names in the list
  - Find student “Bob Jones” in an unsorted list of students
  - How many names must you look at ?
  - In the average case?
  - In the best case?
  - In the worst case?

- Jane Smith
- Fred Wilkins
- Ravi Shankar
- Bob Jones
- Sam Bettson
- Ned Ludd

Real Life Running Time

- O(m log n)
  - Look up the number (in a phone book) for every student in an unsorted list
  - How many pages must you open to find each name and how many times must you do that?
  - In the average case?
  - In the best case?
  - In the worst case?

- Jane Smith
- Fred Wilkins
- Ravi Shankar
- Bob Jones
- Sam Bettson
- Ned Ludd

Real Life Running Time

- O(m^2)
  - How many pairs of students (in the unsorted list) have an average age of 20?
  - Run average for first student and “all others”. Do the same for second, then third, etc.
  - In the average case?
  - In the best case?
  - In the worst case?

- Jane Smith, 18
- Fred Wilkins, 20
- Ravi Shankar, 19
- Bob Jones, 21
- Sam Bettson, 20
- Ned Ludd, 22

Real Life Running Time

- Running time does not forgive
- It's “on the order of” :
  - O(n + n-1 + n-2 + n-3 + ... + 1) = O(n^2)
  - O(n-2) = O(n)
  - O(n) + O(n^2) + O(n) + O(n) = O(n^2)
  - O(293874) = O(123) = O(1)
    - “constant” means that its running time does not change as n does.

Running Times
(one of my favorite questions)

- You are given a 10000 element array which contains pointers to objects holding information about a particular student at a College.
- Sort them by age
- What is the running time?
  - number of interviewees that got this: ~60-70%
  - number of moms that got this: ~100%

Answers I have gotten:

- Call “sort()”
- Create a custom Comparator, then call sort()
- n^2
- Quicksort, etc.
- n log n
- n

Running Times
(same question!)

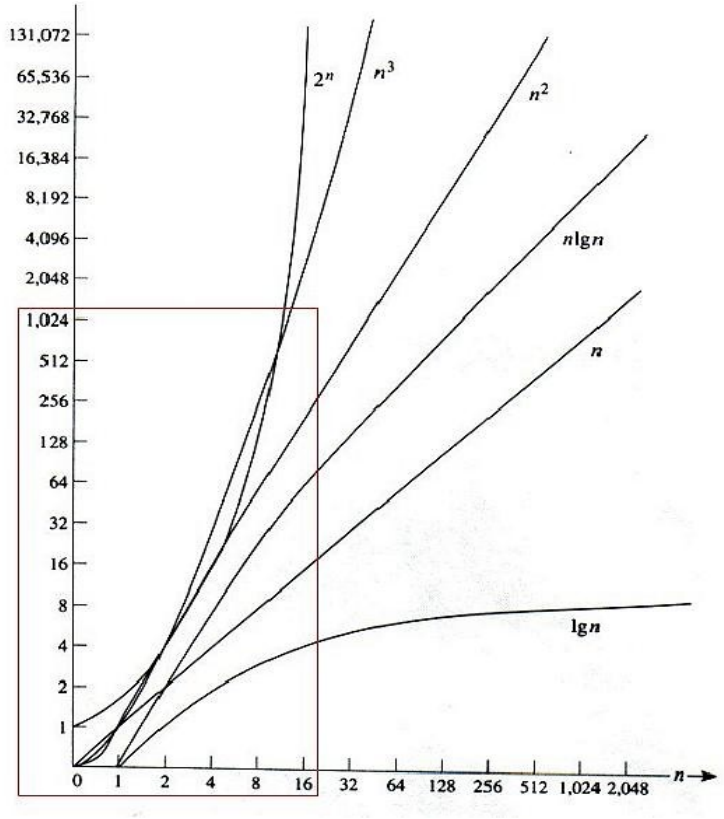
- You are in a gymnasium with a stack of 10000 pieces of paper before you. On each page is written a different student's name and their age.
- Sort these papers by age.
- What is the running time?
- i.e., How many times must you look at each piece of paper?

Running Times

- Why is this important?
  - n^2 (bubble sort)
    - 10000 * 10000 = **100,000,000**
  - n log n (quick sort)
    - 10000 * $\log_2(10000)$ = 10000 * 14 = **140,000**
  - n (linear sort)
    - **10000**

Running Times

- Good interview questions tend to have a naïve solution and a harder, faster, less naïve solution
- Start with “correct”
  - correct is GOOD
  - make it fast thereafter
    - which is gooder



Real Life Running Time
(again)

- O(n^2) -- (switching n to be # of students)
  - How many pairs of students (in the unsorted list) have an average age of 20?
  - Run average for first student and “all others”. Do the same for second, then third, etc.
  - In the average case?
  - In the best case?
  - In the worst case?
  - (we sure?)

- Jane Smith, 18
- Fred Wilkins, 20
- Ravi Shankar, 19
- Bob Jones, 21
- Sam Bettson, 20
- Ned Ludd, 22

Real Life Running Time

- How many pairs of students (in the unsorted list) have an average age of 20?
  - Count the number of students with each age – O(n)
  - For all pairs of students age 20 add 1
    - Combination(age[20], 2)
    - O(age[20])

$$\frac{n!}{k!\,(n-k)!}$$

  - For all buckets i less than 20:
    - Add size(age[i])*size(age[(20-i)+20]))
    - O(m/2) = O(m)
  - O(n) + O(age[20]) + O(m) = O(n+m)

- Jane Smith, 18
- Fred Wilkins, 20
- Ravi Shankar, 19
- Bob Jones, 21
- Sam Bettson, 20
- Ned Ludd, 22

Algorithm Strategy

- Problem solving in this context is rather fair
  - Allowance is given that its an interview
- We can do this easy
  - Or we can do it “real easy”
- Look for extra information about the input
  - i.e., data is a limited range

Algorithm Strategy

- Runtime relations
  - If you find an O(n^2), there is a fair chance there is an O(nlogn) hiding in there – and sorting is the key to getting to it.
- Consider preprocessing the data
  - Again, sort it? Sum it? Count it?

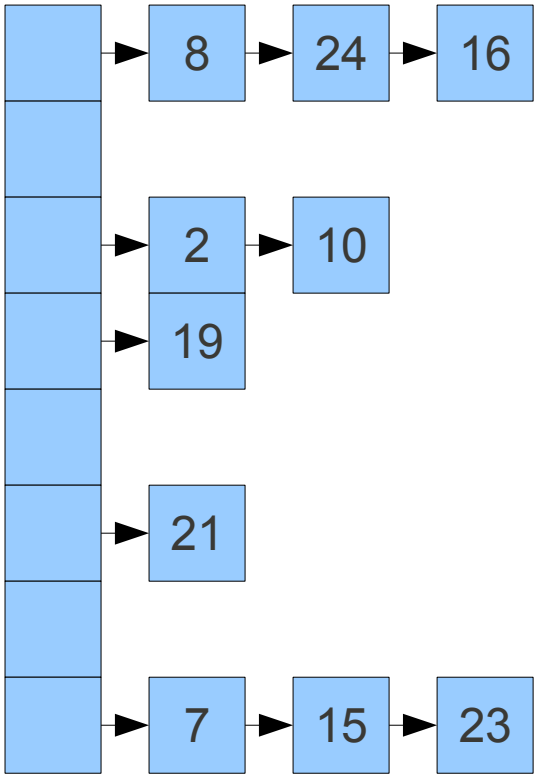
Algorithm Strategy

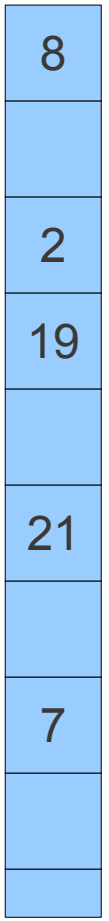
- If you think an interview will be algorithm heavy
  - Go write a binary tree or hashmap from scratch
  - Learn all the running times of common data structures
  - Whiteboard everything – it usually helps and shows what you're thinking

Data Structures Strategy

- HashTable versus BinaryTree
- Data structures often have size to speed trade-offs
- Positionality
  - Many data structures rely on the presence or absence of an element at a specific place
  - Consider a hashtable that has effectively no load-factor
    - it is allowed only one element per bucket

Data Structures Strategy





What's the Running Time

- How many times must you look/touch/examine/compare each piece of data
- Ask yourself, can this problem even possibly be done without examining all data at least once
  - Summing, sorting, etc. (not necessarily searching)
- Careful with running time equivalence
  - n*(n-1)*(n-2)*.....*1 = n^2
  - O(23947394873) and O(1) are both constant

“No Answer” Problem Solving

I hate these
luckily they are rare

How Many...

- How many hamburgers are eaten in the USA everyday?
- How many Walmarts are there in the USA?
- How many cars are there in the world?

Surprisingly...

- The correct answer to all of these is....

Surprisingly...

- The correct answer to all of these is....

"TRY"

Strategy

- Think out loud
- Ask if you don't know a statistic
- Pick round numbers(!)
  - 2, 5, 10
- Look like you're having fun
- There is no answer – you can't be exactly right
  - Thus wrong to some degree is expected
- So.. how many hamburgers are eaten in the USA each day?

Puzzle Questions

- These stink....
  - Say you have a piece of rope that is guaranteed to burn for 1hour. However, no part is guaranteed to burn at a specific rate. How can you use 2 such ropes and a lighter to measure 45 minutes?
- Strategy
  - Whiteboard it
  - Try Lots of things
  - Try to steer the interviewer to more concrete questions

Summary

- Pick your “space” in your language – and define yourself as such
  - (or extend your space before the interview)
- At least memorize running times of insertion and lookups of hashtables, trees, and lists. And of sorting.
- Plenty of questions on the net
  - Even company specific posts

Question Evolution

Correct, to fast, to faster

Problem Evolution

- Problem:
  - We receive 500 emails/sec which average 2048byte each
  - We wish to search each email and filter those that contain a specific search term
    - Opposite of what search engines do
  - “darn”
  - The search term may be embedded - i.e. “goshdarn”, “darnit”, “goshdarnit”
  - we ignore case concerns

Problem Evolution

- Best case (minimum amount of work)
  - every email starts with “darn”
  - 4 comparisons per email
  - 500 per second
  - 2000 comparisons per second

```
Darn that dog. I wish he'd stop chewing slippers
```

Problem Evolution

- Average Case
  - no email contains “darn” and some normal number of letter “d”
  - 2048 comparisons per email
  - 500 per second
  - 1,024,000 comparisons per second
    - Times some constant relative to the numbers of “d” we encounter

```
That silly dog. I wish he'd stop chewing
  slippers
```

Problem Evolution

- Worst case (maximum amount of work)
  - All emails contain balked data and search string works against us
  - 2048 comparisons per email
  - 500 per second
  - 1,024,000 comparisons per second
  - * 4 to find

```
Search for “aaab” in:

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

Interview question

- This question is good because:
  - Its practical
  - It requires problem solving and analysis
  - And shows understanding of algorithms
- Assumptions:
  - Not many emails have the bad word
    - i.e. the common case is no-match

In Java

```
public boolean isBadEmail(String emailBody) {
   // let's ignore case for simplicity
   if (emailBody.contains(“darn”)) {
      return true;
      }
   return false;
}
```

Can we do Better?

```
That silly dog. Darn him chewing my slippers.

dddddddddddddddddddarn <-- match!
```

Can we do Better?

- What if we only searched for the “n”?

```
That silly dog. Darn him chewing my slippers.
   n   n   n   n   n
                  rn
                 arn
                darn  < -- match!
```

- (we're leaving out details, what if we encountered an 'r' somewhere?)

Running times for a total miss

- The naïve way (i.e., Java String.indexOf)
  - O(n)
  - worst: O(n*m)
- The “skip to the n” way
  - O(n/m)
  - worst: O(n*m)

Problem Evolution

- Average Case
  - 2048 comparisons per email - 500 per second
  - 1,024,000 comparisons per second
  - With Skip-to-'n' we only do 256,000 per second
    - ¼ the work !
    - Formally this is Boyer-Moore
    - Can we do any better?

```
That silly dog. I wish he'd stop chewing
  slippers
```

Question Evolution

Expanding the problem

Problem Evolution

- What if we wanted to look for many words?
  - darn
  - durn
  - heck
  - ... and 97 more

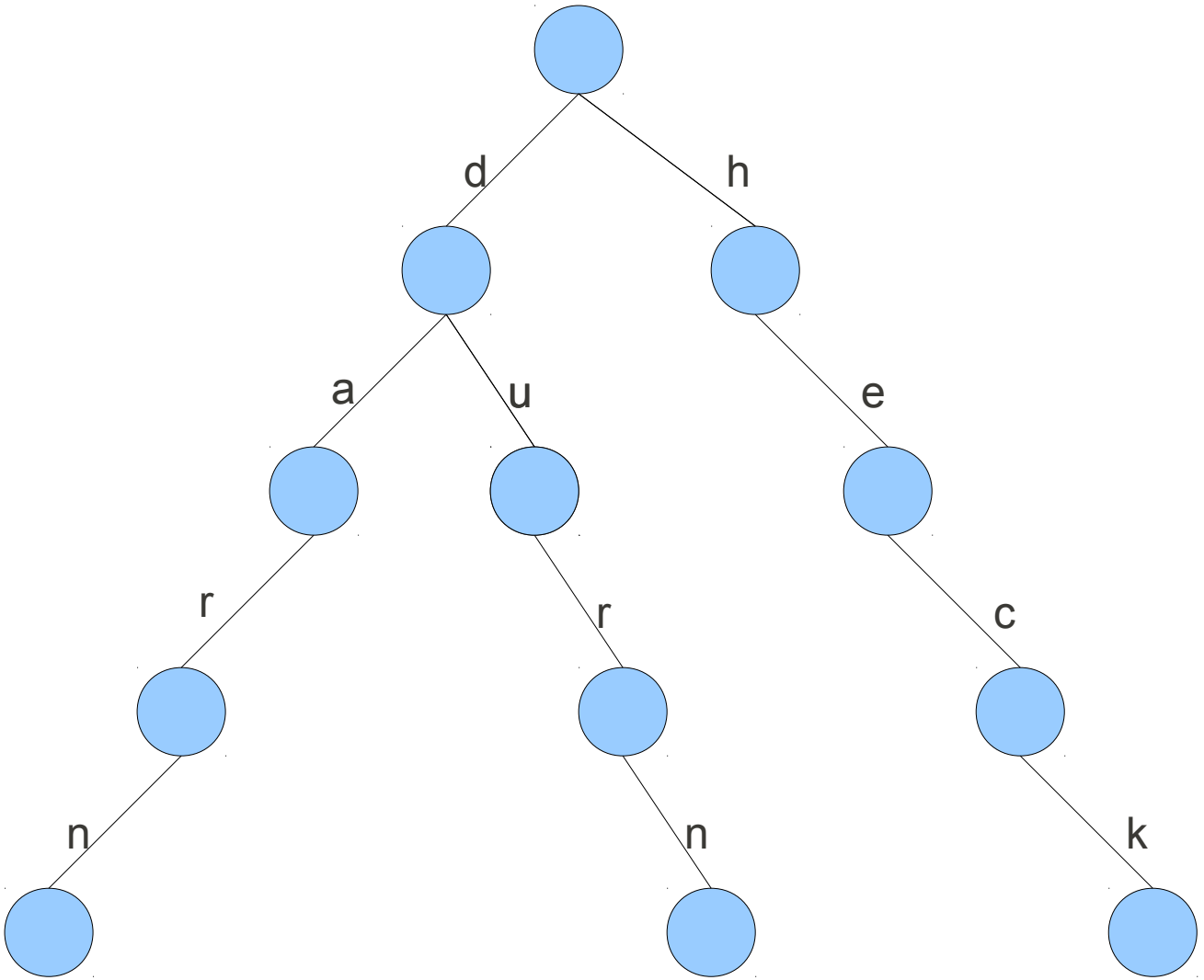
```
That silly dog. I wish he'd stop chewing
  slippers
```

Running times for a total miss

- The naïve way (i.e., Java String.indexOf)
  - O(kn) – where k = number of search terms
  - 1,024,000 comparisons per search term per sec
  - 102,400,000 comparisons per second
  - What about 1000 terms? 10000?
- Boyer-Moore
  - 25,600,000 comparisons per second
- This is starting to get hairy
  - Note: this is actual problem I needed to solve

Problem Evolution

- How about organizing our data better
- We can do “smarter” searching
- We can use a reTRIEval data structure



Running times for a total miss

- Reminder: 100 terms - The naïve way
  - 102,400,000 comparisons per second
- Boyer-Moore
  - 25,600,000 comparisons per second
- Trie
  - 1,024,000 comparisons per second
  - for **any** number of search terms
  - Note: we lost boyer-moored-ness

Problem Summary

- Interview problems generally have a naïve (but correct) solution
  - and a faster (equally correct) one
- Often
  - O(n^2) improves to O(nlogn) (sorting)
  - O(nlogn) improves to O(n) (limited-range sorting)
  - O(n) improves to constant O(1) (list search)
- Correctness is most important
  - wincing if you see an O(n^2) doesn't hurt

References

- Source of graph, good algorithm overview
  - http://216.249.163.93/bob.pilgrim/445/order_of_complexity.html
- Interview questions:
  - http://www.sellsbrothers.com/fun/msiview/default.aspx?content=question.htm
- Algorithms books
  - Cormen, Leiserson, Rivest - http://theory.lcs.mit.edu/~clr/
  - Sedgewick - http://www.cs.princeton.edu/~rs/
- Google on “Google Interview Questions” or “Microsoft Interview Questions”
- Joel's “Write the code to find out if a point lies within a rectangle”
- “Engineer's view of Venture Capital”