Software Developer's JOURNAL

new ideas & solutions for professional programmers

Vol.2 No.02 Issue 02/2014 (7) ISSN 1734-3933

OPEN

THE LATEST INNOVATIVE METHODS IN PROGRAMMING

PLAY-BY-POST RPGS ARE ALIVE AND WELL

USER ACTIONS AROUND MVW

LANGUAGES IN UIS

CMS-BASED WEB APPLICATION MAINTENANCE MADE EASY WITH INTEGRATED OO DESIGN

TECHNICAL INTERVIEWING TECHNIQUE: LOOKING FOR AN INTUITIVE NARRATIVE

BY SOUMEN SARKAR JEFF EDMONDS

THE LATEST INNOVATIVE METHODS IN PROGRAMMING

Table of Contents

Hello Software Developer's Journal Readers,

Welcome to our first released issue...

SDJ magazine team pleases to announce launching the first issue of the free Open magazine. In this issue, a lot of tutorials and practice rich articles are embedded for you to develop your SDJ skills and knowledge. Our ultimate goal is to provide our readers with exactly the knowledge and skills they need in their IT careers. Hence, we will be very glad to receive your suggestions of workshops, tutorials, what you need most, etc...

Let's take a look at what you will engage in this free issue, Our experts will teach you the fundamental design patterns in Perl. In addition, you will discover the languages in UI and how to maintain your App localization and reusability. Additionally, you will learn how to improve your mobile product lifecycle and more of other content-rich articles.

We wish to say "Thank You" and express our gratitude to our experts who contributed to this issue and our coming workshops, however, we invite other experts for collaboration for the next issue, due in 4 weeks.

Stay Tuned, along the whole summer, we were preparing a set of practical workshops for you to be released this month.

Python Web Development: Our consultant, J. Tynan Burke, in this workshop, will teach students the basics and the finer points of web templating, using libraries like Boto for handling static files with Amazon S3, using services like Heroku to maximize efficiency and more.

iOS8/swift programming: Zhou Yangbo, our technical expert, assists readers in learning how to use Swift and SpriteKit to programming, AI-Steering Behaviors, use Swift to programming an Vector2D class, create games in Swift and **a lot mor**e.

R programming: Our instructor, Jim Lemon, introduces all about the R programming, fundamentals, functions, base R statistics, R graphics and more.

IF interested in getting real life technical experiences with our rich content SDJ workshops, ssues, tutorials, etc., Or want to get in touch with our team, please feel free to contact Gurkan Mercan at "*gurkan.mercan@bsdmag.org*", **Contact us TODAY and gain a -14 day- free access to all our workshops and issues. OR HURRY UP and contact us this WEEK and enjoy our limited annual offer of subscription for only 300$**.

Hope you enjoy the issue.

Slawek Szeremeta
slawek.szeremeta@sdjournal.org

Editor in Chief: Slawek Szeremeta
slawek.szeremeta@sdjournal.org

Editorial Advisory Board: Shahid H Rathore, Craig Thornton, Hani Ragab, Kishore P V

Special thanks to our Beta testers and Proofreaders who helped us with this issue. Our magazine would not exist without your assistance and expertise.

Publisher: Paweł Marciniak

Managing Director: Ewa Dudzic

Art. Director: Ireneusz Pogroszewski
ireneusz.pogroszewski@sdjournal.org
DTP: Ireneusz Pogroszewski

Marketing Director: Ewa Dudzic

Publisher: Software Media SK
02-676 Warsaw, Poland
Postepu 17D
email: *en@sdjournal.org*
website: *http://sdjournal.org/*

Technical Interviewing Technique: Looking for an Intuitive Narrative

by Soumen Sarkar and Jeff Edmonds

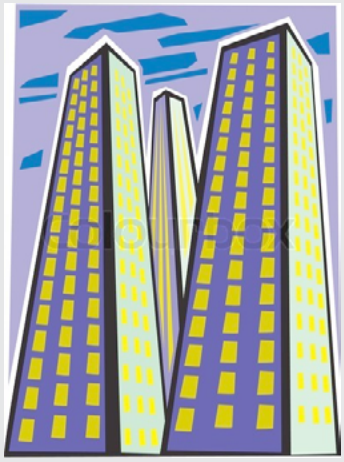
I wanted to write this article to contemplate upon a seemingly vexing problem: In a technical interview, which is a limited contract opportunity, how does the interviewer measure candidate's power of reason and skill of communication on a rationalistic and intuitive basis?

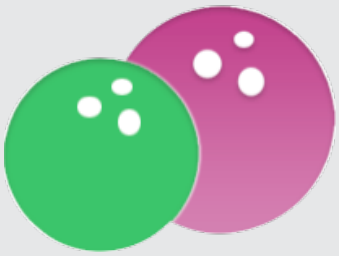
We came to the conclusion that the whole thing depends upon:

- Candidate's preparedness (technical and communication)
- Interviewer's preparedness (interviewing and communication)
- Luck (both on the part of candidate and interviewer)

We think we have a good technical problem, which we hope to be able to do a convincing articulation of the above three.

The Problem





You're standing in front of a 100-story building with two identical bowling balls. You've been tasked with testing the bowling balls' resilience. The building has a stairwell with a window at each story from which you can (conveniently) drop bowling balls. To test the bowling balls you need to find the first floor at which they break. It might be the 100th floor or it might be the 37th floor, but if it breaks somewhere in the middle you know it will break at every floor above. If a ball is not broken, please reuse the ball. Devise an algorithm, which guarantees you'll find the first floor at which one of your bowling balls will break. You're graded on your algorithm's worst-case running time which translates to how many tries you require (the ball may break or it may not).

Interview Structure

This article will propose a guided structure for technical interview in the backdrop of this particular question. The guidance from the interviewer, in my humble opinion, is quite crucial so that the interviewee's power of reason and communication skill can be drawn out and assessed while the interviewer retains control of the engagement experience. In other words, *the interviewer cannot sit back and relax!* We are proposing the following structure:

- Interviewer identifies the constraint of two bowling balls (space complexity constraint) and states that s/he will relax this constraint initially with gradual tightening
- Interviewer identifies the constraint of the number of tries (time complexity constraint) and states that s/he is looking for minimizing worst cases
- The interviewer states that the interview is interactive and advises the candidate to listen carefully and ask questions
- The interviewer then states that following four cases will be considered in an interactive manner:

Case #	Space Complexity	Worst Case Time Complexity
Case 1	Can use only one ball	How many tries?
Case 2	Can use as many balls	How many tries?
Case 3	Can use only two balls	How many tries?
Case 4	Can use only three balls	How many tries?

One Ball Case

One ball case is quite important due to following reasons:

- Icebreaker (interviewee starts talking about a trivial case)

This is a trivial case. Since the algorithm must work for all floors, the only option is to try floor 1 to 100 one-by-one. This also removes any irrelevant details from a candidate's mind like to repeat dropping may make the ball brittle.

Let's assign 10 points out of 100 for this case.

As Many Balls Case

Now that, candidate is talking, let's try to engage on as many balls case. Does the candidate say 100 balls, or 50 balls or something less? If the candidate does not zero on binary search algorithm within five minutes, it is not looking good! Binary search is quite obvious – from a divide and conquer point of view. First drop one ball from floor 50 (= 100/2). If it breaks, then use other balls for lower half (floors 1 through 49) using divide and conquer to halve the problem size at each drop. If the ball does not break then try upper half (floors 51 to 100) using divide and conquer to halve the problem size at each drop. If the candidate gets binary search then you may ask why binary search is applicable to this problem? What is sorted (since binary search is applicable on sorted data structure)?

So the answer is, we need maximum ceil (log2 (100)) = ceil (6.x) = 7 balls/tries. Also note that there is no separation of time complexity (number of tries) and space complexity (number of balls) in this case.

```
Time Complexity = Space Complexity = 7
```

Let's assign 40 points out of 100 for this case. So if the candidate has made till this point, s/he is half way through (10 + 40 = 50 points out of 100).

Two Balls Case

It is time to separate space and time complexity. We constrain the space complexity to two (balls). It should be obvious that time complexity is more than two – come on there are 100 floors! Actually, it has to be at least seven (see previous section). Ask the candidate on the minimum bound of time complexity in this case and look for seven. If the candidate chooses a step size of 14 then there needs to be at least 7 tries (100/14 = 7. X) till the first ball breaks or it is certain that it will break. Let's take an example; let's say the ball breaks

at 100 (the topmost floor). In this case, there will be 7 tries (14, 28, 42,…, 84, 98) and then the candidate can try 99 and 100 ==> total 7 + 2 = 9 tries. If the ball breaks in 97, then the number of tries = 7 (the first ball breaks in floor 98) + 13 (try 2nd ball 85, 86,…, 96, 97) = 20. At this point the candidate could recognize that there are two variables at play:

- Interval size
- Number of intervals

The following equation may help:

```
(Interval size) x (Number of intervals) >= 100
```

For example, if the interval size is 4, then we will need 25 intervals (not good). On the other hand, the interval size of 25 is not desirable as well since if the first ball breaks, then we do need to try linearly within the last interval. We have a situation where the product of two variables is fixed (= 100) and we need to minimize the sum of the two variables. This is achieved by setting the two variables equal to each other:

```
Interval size = Number of intervals = N
```

Then we have,

N x N = 100

→ N = 10 (square root of 100)

A quick calculation shows that maximum number of tries is 10 + 9 = 19. Checking back, if the answer is 99 then we lose the first ball at floor 100 (10, 20,…, 100) and then we need to try linearly (91, 92,…, 99) with the remaining balls.

Let's assign 20 points out of 100 for this sub-case. If the candidate has made out this far, then the score is at 70 (10 + 40 + 20).

Two Balls Case (Optimized)

At this point, the candidate could be told that s/he did quite well. However, number of tries could be improved through a better strategy. Assuming the optimum number is N, how can we derive it? At this point, the narrative could change. We do not know how the candidate will respond. May be this problem is already known to the candidate? However, this complication does not change the intent of this article. The intent of this article is to show how a multi stage problem (with increasing level of sophistication) could possibly used to assess interviewee's critical thinking ability.

Getting back to the problem... this is what we know about calculating N. Let's start with our initial assertion that we have a strategy of N ball drops. Let's forget for a moment that we have 100 floors. Let's say we have N floors – what is the best strategy with the two balls? We think this should be obvious – drop the first ball from floor N. Lets say it breaks – in that case, next ball can be used linearly (one-by-one) for floors 1 to (N -1). So if the ball breaks at floor N or lower, we are still maintaining the logical consistency that we have a strategy of N ball drops.

```
Invariant → 1 + (N - 1) = N
```

What happens if the ball does not break? Well, in that case we can do the second drop from the floor (N + (N – 1)). Why are we going (N – 1) floor above floor N? This is because at this point we have used two drops and if the ball breaks at floor (N + (N – 1)), we still can use remaining (N – 2) tries to check intermediate floors.

```
Invariant → 1 + 1 + (N - 2) = N
```

Again, we are still maintaining the logical consistency that we have a strategy of N ball drops. This act maintaining of the initial assertion that we have a strategy of N drops and not violating this assumption is called *Maintaining The Invariant* (the heart of any algorithm). A crucial thing to observe now is.

We are going higher (from Nth floor to N + N – 1 floor) while maintaining the invariant that maximum number of tries is <= N:

```
Loop Invariant
Proving Your Algorithms
```

- Start Small (initial condition)
- Make Progress (N è N+1 or N → N-1)
- Maintain Invariant

How long this can continue? Well, this can continue till the sequence comes down to last term 1:

N ((N – 1) on top of N ((N – 2) on top of (N + (N – 1)) (... 3 on top of previous (2 on top of previous (1 on top of previous.

If we add all these intervals, at the end, we must reach the 100th floor. So we have the following equation:

```
N + (N – 1) + (N – 2) + ... + 3 + 2 + 1 >= 100

→ N (N + 1) / 2 >=100
→ N = 13.X
→ N = 14
```

Let's assign 30 points out of 100 for this sub-case. Checking back for floor 100 case, we go up to floor 99 in 11 tries (14 + 13 + 12 + 11 + 10 + 9 + 8 + 7 + 6 + 5 + 4) and the ball break at floor 100 (total 12 (11 + 1) tries.

If the candidate has made out this far, then the score is at 100 (10 + 40 + 20 + 30). Full score!!

Three Balls Case

Whew! This has been quite a journey. However, we do not need to stop here. Let's consider 20-point bonus question:

What If You Can Drop Three Balls?

My response would be to use one ball to halve the problem size. Drop one ball from 50th floor and expect it to break. If it breaks, then use other two balls for floors 1 through 49 with two balls optimized algorithms described in the previous section. In that case:

```
N (N + 1) / 2 = 49

→ N = 9. X

→ N = 10
```

Counting the first ball drop (to halve the problem space), we have to add one:

```
N = 10 + 1 = 11
```

If the ball does not break, continue using the third ball to halve the problem size and when the third ball breaks, use other two balls as per two-ball-optimized algorithm. At this point, it is quite evident what to do with four or five balls. For example, for four balls, the answer is N = 7 + 2 = 9. If the candidate gets this as well then s/he gets 120 (full score+ 20 bonus points).

This situation is screaming for the candidate to be hired if the narrative appeals to both logic and feeling. The art of knowing what the candidate knows takes us to the next section.

Objective Epistemology by Ayn Rand

There is a branch of philosophy on pondering the business of knowing: Objective Epistemology. Ayn Rand published Introduction to Objectivist Epistemology, a monograph on the Objectivist theory of concepts in 1967.

Objective Epistemology by Ayn Rand

How do we know what we know? Is reason a reliable source of knowledge – or is it superseded by mystical revelation or emotional intuition? Can we be certain about our knowledge – or must we always remain in doubt?

Does the candidate demonstrate an intuitive pathway to his/her knowledge? Does the candidate work with a mind map of complexity tradeoff, loop invariant, problem size variation, back checking, quick calculation etc.

For example, consider this case: Candidate does very well till score 100 (two ball case) but cannot do the three ball case (20 bonus points). What does this mean? Does the candidate know the problem, but s/he cannot tackle a simple variation?

Applying Ayn Rand to Tech Interview

How do we know what the candidate knows?

Remember that the narrative must feel close to heart – the whole process should appeal to the interviewer. The authenticity of the interaction and engagement should stand out.

Where is this Going?

So far, this article has been about how to think about the algorithm (making progress while maintaining invariant). Let us now strip all algorithmic thinking from this article. What else remains? This is what we think this article has besides algorithm:

- Structuring Interview
- Problem Design
- Having an objective philosophy (abstract architecture of engagement)

This is my point besides algorithms or beyond technical:

Conducting a good technical interview is not easy!

Thought leadership needs to come from the interviewer to create a platform or high potential for interviewer to perform. This creativity from interviewers is needed since we are engaging in human interaction.

Other Interview Problems

Before looking for other similar problems, let us look at the design of this problem:

- Trade off space complexity with time complexity
- Increasing level of sophistication
- Natural (not contrived) description

A problem with space-time tradeoff is by definition computationally elegant (evidence is Turing Machine or Computer Architecture). Increasing the level of sophistication is needed for structuring interview. A problem described in terms of physical setting (100 story building, dropping balls) requires the candidate to model the problem and analyze the solution in that model space. So if you look for another interview problem, please keep these three attributes in mind.

Conclusion

We will conclude by going back to introduction. We said, the success depends on three factors:

- Candidate's preparedness (technical and communication)
- Interviewer's preparedness (interviewing and communication)
- Luck (both on the part of candidate and interviewer)

Hopefully, we could articulate on item 1 and 2 above. We all need luck since several things need to line up in order to make a good hire in today's competitive job market for software engineers. Good luck with your interview!

References

Blog Post: Interview Questions: Two Bowling Balls. Jesse Farmer on Tuesday, April 15, 2008, *http://20bits.com/article/interview-questions-two-bowling-balls*, This blog post has a different take on the same problem.

Book: How to Think About Algorithms, Professor Jeff Edmonds, *http://www.cambridge.org/us/academic/subjects/computer-science/algorithmics-complexity-computer-algebra-and-computational-g/how-think-about-algorithms*

There are many algorithm texts that provide lots of well-polished code and proofs of correctness. This book is not one of them. Instead, this book presents insights, notations, and analogies to help the novice describe and think about algorithms like an expert. By looking at both the big picture and easy step-by-step methods for developing algorithms, the author helps students avoid the common pitfalls. He stresses paradigms such as loop invariants and recursion to unify a huge range of algorithms into a few meta-algorithms. Part of the goal is to teach the students to think abstractly. Without getting bogged with formal proofs, the book fosters a deeper understanding of how and why each algorithm works. These insights are presented in a slow and clear manner accessible to second- or third-year students of computer science, preparing them to find their own innovative ways to solve problems.

Book: Introduction to objective epistemology, Ayn Rand, *http://en.wikipedia.org/wiki/Introduction_to_Objectivist_Epistemology*

The majority of the book is Rand's summation of the Objectivist theory of concepts and solution to the problem of universals. An additional essay by Peikoff discusses the analytic–synthetic distinction. A second edition published in 1990 includes transcripts of a discussion session Rand conducted on epistemology.

About the Authors

Soumen Sarkar *is a Senior Technical Product Manager in Platform team of WalmartLabs where he manages several products based on frameworks that operate in a high-scale, distributed, multi-tenancy private cloud environment. Before WalMartLabs, Soumen worked in high scale platforms with Akamai, Nokia, Yahoo and eBay. He graduated from Indian Institute of Technology (IIT) with masters in Electrical Engineering. E-mail: soumen.sarkar@gmail.com.*

Professor Jeff Edmonds *is at the Theory Group of Department of Computer Science & Engineering, York University. Jeff Edmonds received his Ph.D. in 1992 at the University of Toronto in theoretical computer science. His thesis proved that certain computation problems require a given amount of time and space. He did his post doctorate work at the ICSI in Berkeley on secure multi-media data transmission and in 1995 became an Associate Professor in the Department of Computer Science at York University, Canada. He has worked extensively at IIT Mumbai, India, and University of California San Diego. He is well published in the top theoretical computer science journals on topics including complexity theory, scheduling, proof systems, probability theory, combinatorics, and, of course, algorithms.*