The Applied Algorithmic Thinking Club is a faculty-led student club whose goal is to have fun with algorithms!

The activities are oriented to develop skills on problem solving, algorithmic thinking, modeling, and coding in an informal and inviting setting.

**PERKS OF JOINING THE CLUB**

- Problem Solving
- Whiteboard-interview Coaching
- Leetcode/HackerRank Coaching
- Advisement Talks
- Weekly meetings
- Seidenberg Lounge
- Faculty-led by Dr. Christelle Scharff and Dr. Miguel Moutinho

**EXPECTED OUTCOMES**

- Program for fun!
- Target the moon!
- Algorithms to live by!
- Build experience
- Improve academic standing
- Qualify for top jobs and get them!

**FOCUS ON DIFFERENT AREAS**

- Programming hackathons
- Identify and coach teams for programming competitions

**ADDITIONAL ACTIVITIES**

- Measuring Space
Mission

Develop skills on:

- Problem Solving
- Modeling / Abstraction
- Algorithmic Thinking
- Coding
- Performance Engineering

Real system → Abstraction → Model
Interpretation
Methods

Weekly meetings with various focus:

- **Problem oriented**: for students with weak background in Data Structures, Algorithms, and coding.

- **Whiteboard-interview coaching**: algorithms exercises from real tech interviews, compare solutions, analyze optimizations.

- **Leetcode/HackerRank coaching**: coding exercises from automated tools used in real tech interviews.

- **Advisement talks**: job interviews, career development, graduate school, research opportunities, NSF REU programs, etc.
Near-future Plans

- **NSF S-STEM grant proposal**: on the general theme "Programming for all, closing the gap of inequality to the access of early programming education". Attract academically talented students who had little programming experience in their previous education.

- **Programming competitions coaching**: select and train a team of top students to participate in competitions such as ACM and Google Code Jam.

- **Hackathons**: organize regular programming hackathons reaching out to students from other schools.
The World of Computing Some Years Ago

Woz!
I’ve just come up with a new algorithm for blah that runs in only $O(n \log^* n / \log\log n)$!!!

You know Don, “In The Real World” we can’t afford to spend time coding complex algorithms, we go for the easiest to code.

Donald Knuth
Stanford University

Steve Wozniak
Apple Inc. founder

(FICTIONAL DIALOGUE)
The World of Computing Today

FAST ALGORITHMS ARE CRUCIAL!!
Algorithms and Data Structures are also “Real World”

AND EMPLOYERS DO CARE ABOUT IT, A LOT!
**Google Technical Phone Interview Tips**

Please be prepared for the engineers to ask you questions in the following areas:
- Google products (i.e. what you use)
- Coding ability (you will code in a Google Doc)
- Algorithm Design/Analysis
- System Design

The links and information below may help you prepare for your interview:
- [Interviewing at Google](#)
- [Google Products](#)
- [Five Essential Phone Screen Questions by Steve Yegge (Google Engineer)](#)
- [Types of algorithm questions Google asks: TopCoder Tutorials](#)
- [The Official Google Blog: “Baby steps to a new job” by Gretta Cook (Google Engineer)](#)
- [“How to Get Hired” by Dan Kegel (Google Engineer)](#)
- [Student Guide to Technical Development](#)

**BOOKS (#2 was highly recommended by several engineers and quite representative of the types of coding questions asked):**

2. Types of coding questions Google asks: *Programming Interviews Exposed; Secrets to Landing Your Next Job (Programmer to Programmer)* by John Mongan, Noah Suojanen, and Eric Giguere

**TIPS DIRECTLY FROM OUR ENGINEERS:**

One of our engineers drafted this overview of the main areas software engineers should prepare to have a successful interviews with Google:

1.) **Algorithm Complexity:** You need to know Big-O. If you struggle with basic big-O complexity analysis, then you are almost guaranteed not to get hired. For more information on Algorithms you can visit: [http://www.topcoder.com/tc?module=Static&d1=tutorials&d2=alg_index](http://www.topcoder.com/tc?module=Static&d1=tutorials&d2=alg_index)

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

*Strongly recommended* for information on Coding: [Programming Interviews Exposed: Secrets to landing your next job](http://www.wiley.com/WileyCDA/WileyTitle/productCd-047012167X.html)
How are we going to be ``measured'' in industry?

...by how we create new knowledge.

To create new algorithms, The key is to understand how the current were designed.

Don’t settle to be an expert user of the hottest tool, be an expert designer of the next hottest tool!!
Books you can use to catch-up

- Introduction to Algorithms (Third Edition)
- Algorithms (Fourth Edition)
- Introduction to The Design and Analysis of Algorithms
- Data Structures and Algorithm Analysis in Java (Third Edition)
- Algorithm Design
- Data Structures with Outside in Java
Videos you can use to catch-up

Erik Demaine (MIT) Fall 2011
Charles Leiserson (MIT) Fall 2005
Robert Sedgewick (Princeton) part I
Robert Sedgewick (Princeton) part II
Tim Roughgarden (Stanford)
Tom Leighton (MIT) Math for CS

PACE CELEBRATING
RECURSION
algorithms

Al Kwarizmi

Dijkstra

Bellman

Fibonacci

Euclid

Rivest, Shamir, and Adleman