Undergraduate Research @ IIT

Kyle C. Hale
What is research?

1. Find an important problem/question/mystery
2. Think really hard, read a lot
3. Come up with solution/hypothesis
4. Does it work?
   - No: Start again
   - Yes: Try it out
rejoice! and tell the whole world!
What does it involve?

• Thinking (creatively, critically, sometimes wrongly, sometimes rightly) *a lot
• Reading *a lot
• Learning *a lot
• Questioning *a lot
• Brainstorming/Collaborating *a lot
• Experimenting *a lot
• Programming (these days in most fields, esp. CS) *a lot

• Writing *a lot
• Challenging yourself *a lot
• Growing *a lot
• Traveling, meeting interesting people *a lot
• Joining communities
• Having fun! *a lot
What *isn’t* research?

- Boredom
- Someone telling you exactly what to do
- Just another 3 credit hours
- Working on a problem someone else has the answer for
- A 9-5 job
Why should I care?

• It’s **fun**! You have the **freedom** to explore
• You get to be on the **cutting edge**. Your path to becoming an expert
• You can **contribute something** to the world, creating new knowledge
• Leads to flexible, fulfilling, and well-paying **careers**
What about research *in CS*?

• One of the most important and game-changing fields currently

• Technology *is everywhere*, it’s growing, it ain’t slowing down, and it’s all *driven by computing*

• A lot of open problems, challenges

• Fast-moving: things change quickly, will never be bored!
CS Research Areas (broadly construed)

• AI, machine learning, data science
• Systems and Networking
• Theory
• Programming Languages
• CS + X
What does it take to do research?

• **Curiosity**: If you don’t ask questions, you won’t go exploring
• **Drive + Passion**: You must genuinely *care* if you are to make progress
• **Creativity**: It’s not just for the arts! Some of the best research involves new and original ways of rethinking old problems
• **Courage**: You have to learn that *failure is both normal and okay*
  • This is, of course, *unlike* your classes, where you should definitely not fail
• **Dedication**: The best researchers pick a problem and work hard on it
Notice what I didn’t list...

• **Genius/Talent:** Overrated; it sometimes helps, but **curiosity, passion, and perseverance win out big-time**

• **Straight A’s:** The collective sphere of human knowledge couldn’t care less about your GPA
  
  • But *it does* help a professor determine how serious you are about learning...and therefore whether to let you join the lab...so don’t ignore them 😊

• **Knowledge:** I’m more interested in working with people who are **eager and willing to learn** more than with those who already know a lot
Career Paths

• **After undergrad -> grad school (PhD)**
  • Yes, it’s more school, but it’s *not the same*

• **Research Scientist/Engineer**
  • National Labs/Government entities
  • Think Tanks
  • Industry

• **Academic (Professor)**
  • Research + Teaching usually, sometimes just one

• **Consulting**
Opportunities in the Department

• **Informal arrangements**: Generally professors are willing to let you work with them if you’re hard-working and motivated
  • e.g., take CS 497 with a prof (independent study)
  • We sometimes have ad hoc summer (or regular semester funding) to pay you for it

• **CS Honors Research Specialization**: this is new! Perform research as you progress through ugrad, write and present thesis at end. Ideally, publish! Talk to us more if you have ??s

• Research Experience for Undergrad – paid (more on this after)
  • At another institution
  • Here at IIT
OK, sounds great; how do I start?

• **Be engaged in class!**
  • Ask *a lot* of questions; **BE CURIOUS.** Don’t be afraid to look dumb!
  • Sometimes seemingly obvious questions lead to adventures
  • Do these things and **we will for sure notice!**

• **Get to know your professors**
  • We love talking about research! (we’re nerds)
  • Don’t be intimidated, we’re just people
    • **We will make time for you**

• Find out what you’re passionate about, and **approach one of us about doing research**
Important questions in CS
Systems (not exhaustive)

• How to deal with explosion of data (“big data”)?
• How to make extreme amounts of data easy to store, easy to analyze, and easy to compose *efficiently*? -> database systems
• How to make our systems secure?
• How to tame complexity of systems?
• How to design better chip architectures to meet today's challenges?
• How to handle growth of heterogeneous hardware?
• How to make systems *scalable*? (1000s of cores, 1000s of machines)
• How to make systems reliable in the face of failure?
Theory (not exhaustive)

• How to design efficient algorithms for large swaths of (changing) data?
• How to use algorithms to predict behavior and inform decisions?
• Improving computational efficiency across the board (e.g. graph isomorphism, SAT, bin packing, scheduling, etc.)
• P = NP? (Just how hard are certain classes of problems? Can we do better?)
• Someone else would know better 😊
AI + Machine Learning + Data Science (not exhaustive)

• How to build (and understand) intelligent machines
  • That have same capabilities of human intelligence: strong AI
  • That will help make our lives better: increase pattern matching, insight abilities of machines
• How to help machines make insights on large amounts of data?
• How to help machines effectively interact with and understand humans? e.g. Natural language processing, knowledge representation, automated reasoning, problem solving
• How to help machines understand the world and interpret data? e.g. computer vision, knowledge representation, etc.
• How to make sure they AI makes positive (and ethical) impact on humanity
• How to make intelligent machines efficient?
• Transparent machine learning: how do we understand why and how a machine did what it did?
• Are thinking machines conscious? How is our brain different? Philosophy of mind
Programming Languages and Compilers (not exhaustive)

• How do we make it easier to write efficient programs?
• How do we make compilers generate better code?
• How do we automatically port code from one architecture to another?
• What are the right abstractions to current (and new) machines?
• How to create languages that prevent (or at least discourage) bugs?
• How to make programs more secure?
• How to verify that a program does what we think? (verification)
• How to apply tools like machine learning to compilation and program verification?