Generating Research Ideas

To do …

Based on N. Feamster and A. Gray slides

General approach

- Find a problem
- Understand the problem
- Make a plan for a solution, carry it out
- Review the solution

“The most exciting phrase to hear in science, the one that heralds the most discoveries, is not "Eureka!" (I found it!) but 'That's funny...”
Isaac Asimov

Finding problems

- Hop on a trend
- Find a nail that fits your hammer
- Revisit old problems (with new perspective)
- Making life easier
  - Pain points
  - Wish lists
- “*-ations”
  - Generalization
  - Specialization
  - Automation

Hop on a trend – where to find them?

- Funding agencies
  - Funded proposals; calls for proposals
- Conference calls for papers
- Industry/technology trends: trade rags

Finding a nail for your hammer

- Become an expert at something
  - You'll become valuable to a lot of people
- Build a system that sets you ahead of the pack

- Apply your "secret weapon" to one or more problem areas
  - Algorithm
  - System
  - Expertise

Revisiting problems

- Previous solutions may have assumed certain problem constraints
  - Layering is good; not enough memory; …
- What has changed since the problem was "solved"?
  - Processing power
  - Cost of memory
  - New protocols
  - New applications
  - …

Making life easier

- Look for pain points
  - Industry, other researchers, etc. for problems that recur
  - In programming, if you have to do something more than a few times, script!
  - In research, if the same problem is recurring and solved the same silly way, look for a better one …
- Wish lists
  - What systems do you wish you had that would make your life easier?
  - What questions would you like answered to?

Automation, generalization, specialization

- Automation
  - Some tasks are manual and painful
  - Could you automate it? Difficult because it requires complex reasoning
- Generalize from specifics
  - Previous work may outline points in the design space
  - Is there a general algorithm, system, framework, etc., that solves a *larger* class of problems?
- Specialize
  - Find general problems, “problem areas” (taxonomies and surveys)
  - Applying constraints to the problem in different ways may yield a new class of problems (e.g., routing)

General approach

- Find a problem
- Understand the problem
- Make a plan for a solution, carry it out
- Review the solution

Exhaustive search

- Collect data
  - Can enhance your expertise as a side effect
- Model the problem
  - List all of the constraints to a problem space
  - Consider the different angles within your model that you might be able to attack the problem
- Consider many other examples
  - May suggest general framework or approach
  - You may also see a completely different approach

Formalization

- Define metrics
  - Ways to measure the quality of various solutions
  - What constitutes a “good solution”
  - Objective functions can be optimized

- Formalization/modeling can lead to simplifying assumptions (hopefully not over-simplifying)
  - Can also suggest ways to attack the problem
  - …or an algorithm itself

Decomposition

- Given a model, it often becomes easier to break a solution into smaller parts
- Understand each part in detail, and how they interact
- Then revisit the whole

General approach

- Find a problem
- Understand the problem
- Make a plan for a solution, carry it out
- Review the solution

Consider related problems and analogies

- Try to restate the problem, or create an equivalent problem
  - Consider different terminologies and representations
- See if your problem matches a general form already formalized
- Can you use the solution to a related problem?

- Make an analogy to another problem, then look at its solution

Change the problem to one you can solve

- Make simplifying assumptions
  - Violate some of the constraints of the problem
  - Define a sense of approximation to the ideal solution
- Then revisit the original problem
- Make the minimally-simpler problem; then relate the solutions to the two problems
  - “mathematical induction”

Just start, with anything

- Start with a strawman solution, then modify as needed
  - e.g. (in algorithms): Propose a simple algorithm, check its correctness
  - e.g. (in data modeling): Look at simple statistics of a dataset, then dive into anomalies
  - e.g. (in systems): Just whip up some code

Work backward from the goal

- Visualize the solution, and what it must look like, or probably looks like
- See what's needed to get there
- Consider all the solutions that can't work

Solve a part, or each part

- Solve each part separately, then stitch the solutions together
  - Start with the part which is most tractable
  - “divide-conquer-merge”
  - Be careful: it’s always best to avoid separate objective functions when possible
- Perhaps finding a good solution to a part is a good problem in itself

Think in speech or pictures

- Use dialogues with others
  - Or yourself
  - Talk to people who approach things differently
- Draw pictures
  - Add auxiliary elements, to be able to relate to other problems/solutions

Come from all angles

- Keep coming with a new twist on the problem
  - Break out of a thinking pattern or dead end
  - A new twist renews motivation
- Keep track of all your ideas and partially-completed paths
- Finally, let your subconscious work
  - Immersion
  - Stay relaxed
  - Or: use deadlines to force shortcuts

General approach

- Find a problem
- Understand the problem
- Make a plan for a solution, carry it out
- Review the solution

Look back at your solution

- Check that it really works
  - If it works, note the key to why, more abstractly
  - Were all of the constraints, difficulties, and facts used and accounted for
- Try to improve upon it
  - Can you achieve the same thing more directly or easily

What else can your solution do?

- Now you have a hammer
- Can you use the solution for some other problem?
  - A more general form of the problem?
  - An interesting special case?
  - A related problem or analogous problem?

Making a "theory"

- If successful, you may have a "theory" = a framework for characterizing problems and/or solutions
  - Says when it applies, when it doesn't
  - Characterizes the hardness of different problems
    - May identify simple special cases
  - Characterizes the quality of different solutions
    - How long it takes, amount of resources it uses
  - Show/characterize solution meeting criteria
    - correctness, convergence, etc.