CS167: Reading in Algorithms
Paper-Reading Survival Kit*

Tim Roughgarden†

1 Review: The Five High-Level Questions

Don't forget the five most important questions.

1. **What problem is the paper trying to solve?**
2. **Why is the problem interesting?**
3. **What is the primary contribution?**
4. **How did they do it?**
5. **What are the key take-aways?**

The rest of this document drills down on the fourth question, which for theoretical papers is generally the hardest to answer.

2 Finding Supplementary Material

Reading a paper is not always the quickest way to understanding its contributions. It's often helpful to hear what others have to say about it. So look for:

1. Any books, book chapters, or surveys that discuss the contributions of the paper.
2. Archived lecture (or "scribe") notes from a university graduate course that covered the paper.
3. Slides or videos from oral presentations of the paper. Often these will be on the authors' home pages.

 This list originated in a brainstorming session with Paul Dütting, Vasilis Gkatzelis, Rishi Gupta, Anthony Kim, Kostas Kollias, Okke Schrijvers, Inbal Talgam-Cohen, Joshua Wang, and David Wu.

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

You should also learn how to do a reverse citation search, for example through Google Scholar. This is an easy way to see how follow-up papers describe the paper's contributions.

3 Navigating a Paper

You're not reading a novel. *Don't read the paper linearly from front to back.* There are several corollaries of this principle.

1. Generally one understands a paper through successive refinement; see also [1]. The first pass is generally devoted to understanding the high-level answers to the questions in Section 1. On this pass, a good strategy is to read the relevant parts of the introduction (e.g., the parts that talk about problem motivation, but skipping a detailed literature review), the basics of the model (just enough so that you understand the problem studied), the statements of the primary or simplest results, and the conclusions.

   One you understand what a paper does at a high level, subsequent passes generally focus on a specific technical contribution and how the authors did it. This deeper understanding might lead you to revise your initial answers to the five questions of Section 1.

2. Many researchers prefer to skip the introduction entirely, proceeding directly to the description of the model or problem studied.

3. Never save the conclusions for the end. Many researchers read them first.

4. Try flipping through the paper reading only the theorem statements, plus any figures and tables. With well-written papers, this is often enough to get the gist of the contributions.

5. Differentiate between essential and non-essential technical sections. Often a single technical section showcases the new ideas in the paper, with the others relevant only for experts.

4 On Related Work

Opinions vary widely on how much to pay attention to the discussion of related work. Some researchers ignore it completely; others obsess over it. Here are a few reasons why reading related work can be useful:

1. There are often lots of logical dependencies in the literature — the paper you're reading might well combine algorithms, theorems, or vocabulary from several previous papers. Sometimes to understand one paper you wind up needing to understand several. The advice in Section 2 about looking for surveys and lecture notes is particularly relevant in this case.

2. Sometimes your goal is to understand the gist of a research area — a collection of a dozen or two papers, say — rather than the contributions of any particular paper.

3. Sometimes you want to identify the most influential papers or people in a research area.

If your goal is just to understand the paper at hand, then ignoring the related work section can be a good idea. If the paper uses results from previous work, just accept the previous results as "black boxes" and focus on the contributions of the present paper.

5 Make Things Easier

A simple but crucial technique for understanding a paper is to *specialize it*. Many authors state their results in the most general way possible, and you should push back.

1. Whenever there is a free parameter, see if you can get away with instantiating it to 2 (or $\frac{1}{2}$, or 1, or whatever is the simplest value that makes sense for the context).

2. Identify some canonical special cases. For example, in Lecture #1 we kept in mind a sparse graph with $\approx \sqrt{n}$ vertices of degree $\approx \sqrt{n}$. In Lectures #2-3 with pseudorandom data, we kept in mind the special case of a low-collision-probability distribution that is uniform on a secret set of a given size.

   Many algorithms are initially developed for an "idealized case," and then extend (perhaps with much more work) to a more general setting. For example: for an algorithm for planar graphs, is it simpler to understand when the graph is a grid? Does it help to assume that a graph is regular (i.e., all vertex degrees are the same)? Or that the data is random? When analyzing a randomized algorithm, how do things simplify if you assume that all random variables are equal to their expectations?

3. Try to add assumptions (like those above) until the problem becomes trivial (e.g., solvable by brute-force search or the "obvious algorithm") or ill-posed. Then pull back slightly and try to identify the simplest non-trivial special case.

4. To get a feel for an algorithm, work through some small examples. For each non-trivial idea in the algorithm, find the simplest concrete example that demonstrates the need for that idea.

   Many proofs can also be understood through small examples. E.g., you can trace through an inductive proof when $n = 3$. You might be able to prove an assertion for small examples by brute force.

5. Draw lots of pictures as you try to understood an algorithm or a proof. Often you'll do this in conjunction with working through small examples.

6 Nuts and Bolts

Here are some low-level tips. Most of these are applicable to any kind of mathematical reading.

1. Many people think that the best way to understand an algorithm is to code it up. Analogously, the best way to understand a mathematical result is to try to prove it yourself. When you get stuck, take a look at the proof in the paper — treat the paper's proof like a hint that you find in the back of a textbook. (That's about the level of detail you can expect in a typical conference paper!)

2. For every lemma and theorem, identify all of the hypotheses. Where in the proof is each of them used? For each, how would the proof break if it were dropped? Even better, try to find explicit counterexamples to the statement when each of the hypotheses is dropped.

3. Probability arguments are especially tricky. Even seasoned researchers make basic mistakes, like implicitly assuming that two events are independent when they're not. It's always a good sanity check to ask what a given probability or expectation is over (i.e., to identify the sample space).

7 On Mistakes

Research papers, especially conference versions, usually have far more mistakes than textbooks. Most mistakes are not too serious. Many are typos. Sometimes the authors say one thing when they mean something else. Sometimes authors forget to handle all of the edge cases. These are the types of mistakes that you should correct for yourself as you read the paper — this approach also blends nicely with the advice above to regard the proofs as hints, like in the back of a book, rather than treating them as airtight arguments.

More rarely, a paper has a fatal flaw. Be warned, though, that what seems at first like a fatal flaw is often just an easily fixable mistake or misstatement. Whenever you find yourself not believing a statement, try to find a counterexample that demonstrates that the statement is false. If you succeed in this, see if an extra hypothesis (perhaps unmentioned by the authors) or a minor tweak salvages the statement. If you can't find a counterexample, a good working hypothesis is that the statement really is true — see if you can supply your own proof.

8 After Reading the Paper

After you've put in the hard work of understanding a paper, the following are worth doing.

1. If it's a research area you might continue to work in, archive some notes about the paper so that you can refer to them later. You'll be dismayed at how quickly you

forget the details of the paper, but assuaged by how quickly it comes back when you read back over your notes.

2. Identify any open questions. What's the next step in this line of research? Many papers state open questions explicitly in the conclusions section. Sometimes there are obvious concrete open questions, like improving the result (a better running time, a better approximation guarantee, etc.) or finding a matching lower bound. Think also about non-obvious open questions; see also the next point.

3. In hindsight, what's your opinion of the paper and its results? What are the paper's weaknesses? What do you wish they had proved instead? Would a different model better capture the motivating issues? Answers to these questions can be the basis for future research.

References

[1] S. Keshav. How to read a paper. *Computer Communication Review*, 37(3):83–84, 2007.