CS167: Reading in Algorithms
Focus Questions[*]

Tim Roughgarden[†]

1 Introduction

These notes give suggestions about questions to keep in mind when reading a paper, giving a presentation on a paper, or writing up your own research. These questions are meant largely as guidelines — not all of them will apply to every research paper, and different students will have different favored approaches for understanding a research paper.

Here are the five most important questions. Examples, more details, and further questions are given in the subsequent sections.

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?**

In your oral presentations, you should make every effort to answer questions 1, 2, 3, and 5, with the additional time spent giving intuition for the answer to question 4. One easy recipe for your paper responses is to write a paragraph or so about each of the first three questions.

2 What Did They Do?

2.1 The Three Most Important Questions

The first set of questions seek to understand the statement and importance of the paper's contribution.

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

1. **What problem is the paper trying to solve?** In many algorithms paper, the answer to this question will literally be a computational problem. For example, in our first lecture, the problem was, given a graph, to count the number of triangles in the graph. Next week, we'll discuss a paper where the computational problem is dimension reduction [1].

   In other papers, the problem is less precisely defined. For example, in the Mitzenmacher-Vadhan paper [3], the problem was one of modeling — figuring out a model of data that leads to performance predictions that are more accurate than with worst-case data. Next week, we'll see another example, in a paper that strives for a good model for reasoning about Map-Reduce-type algorithms [2].

2. **Why is the problem interesting?** In answering this question, it's worth remembering that a paper can be interesting for many different reasons. It's important to be able to critically evaluate papers, and pretty much all research papers can be criticized in one way or another (the assumptions are too strong, the guarantees are too weak, etc.). It's equally important to take the "glass half-full" approach and identify the positive aspects of the paper. If you undertake theoretical research yourself in the future, you'll find that it's difficult to prove *anything* remotely interesting, let alone the "perfect theorem."

   For example, the problem studied might be immediately useful for an interesting application, either directly or as a subroutine. This was the case with triangle counting [4]. Or perhaps the problem is interesting theoretically, for example by highlighting the power or limitations of a specific algorithmic technique.

   Note that you may disagree with the authors about whether or not the problem studied is interesting. Such disagreement is healthy. At the same time, it's useful to know why other people believe that a problem is important, even if you don't.

3. **What is the primary contribution?** In many algorithms papers, the answer is clear: a new algorithm for the problem being studied. This was the case with the triangle-counting paper[4], and will again be the case with the dimension-reduction paper we discuss next week.

   The pseudorandom data paper [3], on the other hand, did not offer any new ways to solve any problems — rather, it gave a theoretical explanation of why the existing solutions do indeed work well. The second paper we discuss next week [2] also contributes a new model, which is meant to guide the design of Map-Reduce-type algorithms.

Any reading of a research paper should strive to identify answers to the three questions above. You probably also want to know more, such as how and why the proposed solutions work. But answers to the three questions above are generally more likely to be remembered in the long term than anything else.

Similarly, in your oral presentations, you should make sure that the three questions above are answered as clearly as possible. While the questions may sound simple enough, conveying the answers clearly can be a challenge, and might take the bulk of your allotted time. A presentation that does not answer one of these questions cannot be considered a success. Conversely, in a presentation that makes clear the answers to these three questions, many other sins can be forgiven.

For those of you that wind up writing your own research papers, you want to make it as easy as possible for a casual reader to answer all of the questions above.

2.2 Gravy

The primary contribution of a paper is generally offered, implicitly or explicitly, as "better than X according to the metric Y". The next two questions seek to explicitly identify "Y" and "X," respectively. These questions are less universally applicable and less crucial than the three above, but are still a useful guide for interpreting a paper.

(extra) **In what sense is the contribution "good"?** The following question is nearly equivalent: what criteria are being used to compare two different candidate solutions? Example criteria that are precisely defined include the running time or approximation guarantee of an algorithm, either theoretically or on synthetic or real-world data sets. More nebulous criteria include the "simplicity" or "practicality" of an algorithm, or the "accuracy" of a model. For example, the proposed triangle-counting algorithms [4] were offered as algorithms that "parallelize well." The modeling contribution in [3] of pseudorandom data is a "better model" than the traditional worst-case data model, in that its predictions of data structure performance are "more accurate."

(extra) **What is the "obvious" or "baseline" solution?** Most problems have a naive (but not necessarily bad) solution — brute-force search (as in the counting triangle application), the "standard model" (e.g., the worst-case data model), the well-known solution you learned in undergrad algorithms, a research paper on the same problem from two years ago, etc. What is it? In what sense(s) is the paper's alternative, and probably more complicated, solution better?

3 How Did They Do It?

For papers in algorithms, the following question is often the hardest one to understand and, once understood, the hardest to explain to others.

4. **How did they do it?**

For theoretical papers, this question boils down to understanding the proofs. Detailed understanding can take a long time, and it's not unusual to spend hours on a single page or even a single paragraph. Given a budget on time (to understand or to explain to others), here are some things to focus on.

(a) High-level proof outline. If you're lucky, the authors provided one for you, which you can then study carefully. Alternatively, you can start be reading the statements of all the propositions, lemma, and theorems in the paper and try to understand how they fit together. What is the overall architecture of the argument?

(b) A key lemma. For example, in our discussion of hashing pseudorandom data, we emphasized the Leftover Hash Lemma, which is the key idea behind the more general statement for block sources of arbitrary length. We also didn't give the full proof of the Leftover Hash Lemma — only the first step, to emphasize the role of the lemma's hypotheses in the proof.

Similarly, in the triangle counting lecture, we provided a modest result that supported the intuition that delegating counting to low-degree vertices could yield a big savings in the running time. Precisely, we proved a run time bound of $O(m^{3/2})$ on the cleverer algorithm, as opposed to the $\Omega(m^2)$ worst-case running time suffered by the simpler solution (e.g., on a star graph).

(c) Simple special cases and illuminating examples. The way most papers are written suppresses the process by which most papers are created. Typically, researchers use concrete examples and well-chosen special cases to come up with new ideas, and then for their paper state the new ideas and their consequences in the most general terms possible. Unfortunately, they often omit the original motivating examples and special cases entirely. Reverse engineering them can unlock the secrets of a paper.

For experimental papers, the answer to the question has a different nature — what experiments did they run? What data sets were used to argue their central thesis?

4 Lessons Learned

The final important question identifies the bottom line of the paper — what have we learned?

5. **What are the key take-aways?**

Sometimes the answer to this question is quite close to that of the third question (the primary contribution), albeit with the benefit of hindsight. Well-written papers will answer the above question explicitly in a conclusions section. Example answers include: a good way to solve a computational problem (e.g., in triangle-counting, delegating work to low-degree vertices); a good way to think about a problem or concept (e.g., universal hashing is as good as random hashing, provided the data is pseudorandom); or a cool new problem or technique.

References

[1] Nir Ailon and Bernard Chazelle. Faster dimension reduction. *Commun. ACM*, 53(2):97–104, 2010.

[2] Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for mapreduce. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 938–948, 2010.

[3] Michael Mitzenmacher and Salil P. Vadhan. Why simple hash functions work: exploiting the entropy in a data stream. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 746–755, 2008.

[4] Siddharth Suri and Sergei Vassilvitskii. Counting triangles and the curse of the last reducer. In *Proceedings of the 20th International Conference on World Wide Web (WWW)*, pages 607–614, 2011.