Jonathan Crawford – July 22, 2019

Homework 5: NP - The Job Interview 50 Points

Question. We have discussed your programming skills, a variety of algorithms, and some general topics in asymptotic runtime and runspace. One last topic we'd like you to talk about is intractability. Can you briefly describe your concept of intractability, NP-Hardness, and NP-Completeness? (Please answer in general terms.)

Intractable problems remain a fascination because they lack an efficiently provable polynomial time algorithm and we as computer theorists and lifelong learners have a hard time "computing" and accepting this fact. They are often problems that we can picture in our surrounding world and remind us of the organic and unpredictably random world we live in yet try to tame on a daily basis with mathematics and computation. One that comes to mind is the optimal size container to store a set of n items, only achievable by brute force and backtracking. My concept of intractability is that since there are problems that there is an algorithm for, but an efficient one yet to be discovered. The best way to approach these problems is separation of concerns, and to concentrate on areas of the brute force algorithm that can be improved upon. This leaves less to be discovered as science catches up with the unimprovable portions and possibly teases out portions of randomness to allow it.

My concept of NP hardness and NP complete is that it largely revolves around a subset of equivalence for NP problems. When we look at a set of algorithms that we categorize as unique enough to differentiate as distinguishable and NP. Even though they are tackling a different problem the part of the them that implies randomness and unprovability might be a shared concept through any manner of its ambiguity. This implies an umbrella of sorts on the type of problem we classify as NP, and since NP does not equal P we have the possibility that if we discover a provable efficient algorithm for one that means there is one for all of them in this set. When we talk about NP complete problems we are really meaning all the problems that belong to NP and also NP hard. This is because if we believe that NP does not equal P then NP complete would form the intersection over the ambiguity representing the inefficiently solvable problems that would be reduceable if solved.

I believe these problems will all be solved in my lifetime. If you look at the ramp up speed of technology and human discovery over the past 20 years it is hard to see this ambiguity lasting much longer. While I do appreciate the organic randomness of the world around us, I believe any problem that can fit into an algorithm has already proven itself solvable just by first being distinguishably recognizable. It simply becomes a matter of time after that. Google's news algorithm clearly knows my interests because in my feed just this past weekend there was an article about a young mathematician solving the sensitivity conjecture that has stumped computer science for decades in few enough words to fit into a tweet. The point being that once a problem is simply framed, it has already begun the process of being broken down.

Follow-Up #1. What is an example of an NP Complete problem that you find surprising?

To me the most interesting NP complete problem is the decision-based travelling salesman problem. What makes it so interesting is that it is a problem that you can picture in your mind as you drive home today past neighborhoods and street corners. What also makes it interesting, and as you pointed out surprising, is that if you ask a layman unfamiliar with computational theory they would believe the solution is surely straightforward. They can understand the premise, given mapped roads, and global positioning systems.

The interest would grow for a student of computer science as soon as they were exposed to Hamiltonian paths. The fact that the symmetry of one direction to and from a location provides a halving of the possible solutions is inherently mathematical yet common sense and visual. It is also a great way to explain the concept of weighted edges through distance of the paths. From a young age we learn to travel these edges in a path to school, the park, and back home for dinner. We continue this later in life to work, on errands, and back home. I also have read that most solutions found are only within 2-3% of the optimal tour, a figure I find not only surprising but fascinating.

Follow-Up #2. OK, that's all well and good, but how does this whole mess of intractability affect the practical issues facing algorithm design and implementation?

One thing these problems provide that can be very advantageous to practical design is that they provide a reference point for truly difficult struggles we face in everyday implementations. Mainly, the framing and recognizing of the elements of ambiguity (if there are any) in the problem we are trying to tackle. Also, since the greatest researchers and engineers the world over are constantly attempting to solve them much can be gleaned from their thought processes and approaches. What may have failed in terms of NP solvability might provide inspiration to us as lifelong learners in tackling the difficulties of our own working set of problems. Problems like this provided a forum of ideas basically and are necessary in order to keep pushing the boundary further in computer science in that regard. They drive achievement and allow us to know the working limitations of what is possible so that we can remain efficient in our work let alone our algorithms.

References

1. https://intractable.askdefine.com/
2. https://www.quantamagazine.org/mathematician-solves-computer-science-conjecture-in-two-pages-20190725/
3. https://en.wikipedia.org/wiki/NP-completeness
4. https://www.britannica.com/science/NP-complete-problem
5. https://www.geeksforgeeks.org/np-completeness-set-1/