6.080 / 6.089 Great Ideas in Theoretical Computer Science
Spring 2008

6.080/6.089 GITCS DATE

Lecture 12

Lecturer: Scott Aaronson *Scribe: Mergen Nachin*

1 Review of last lecture

- NP-completeness in practice. We discussed many of the approaches people use to cope with NP-complete problems in real life. These include brute-force search (for small instance sizes), cleverly-optimized backtrack search, fixed-parameter algorithms, approximation algorithms, and *heuristic algorithms* like simulated annealing (which don't always work but often do pretty well in practice).
- Factoring, as an intermediate problem between P and NP-complete. We saw how factoring, though believed to be hard for classical computers, has a special structure that makes it different from any known NP-complete problem.
- coNP.

2 Space complexity

Now we will categorize problems in terms of how much memory they use.

Definition 1 *L is in PSPACE if there exists a poly-space Turing machine M such that for all x, x is in L if and only if M(x) accepts. Just like with NP, we can define PSPACE-hard and PSPACE-complete problems.*

An interesting example of a PSPACE-complete problem is n-by-n chess:

> Given an arrangement of pieces on an n-by-n chessboard, and assuming a polynomial upper bound on the number of moves, decide whether White has a winning strategy.

(Note that we need to generalize chess to an n-by-n board, since standard 8-by-8 chess is a finite game that's completely solvable in $O(1)$ time.)

Let's now define another complexity class called EXP, which is apparently even bigger than PSPACE.

Definition 2 *Deterministic $f(n)$-Time, denoted DTIME($f(n)$), is the class of decision problems solvable by a Turing machine in time $O(f(n))$.*

Definition 3 *Exponential Time, denoted EXP, equals the union of $DTIME(2^{p(n)})$ over all polynomials p.*

Claim 4 $PSPACE \subseteq EXP$

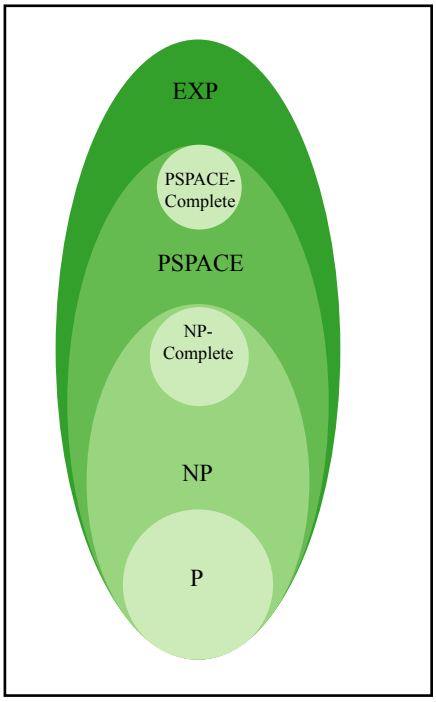

Figure 1: A general picture that we believe

Proof Just like in problem set 2. In order for the machine to halt, any "configuration" must be visited at most once. Assuming an upper bound of $p(n)$ on the number of tape squares, there are $2^{p(n)}$ possible settings of 0's and 1's on the tape, $p(n)$ possible locations for the Turing machine head, and s possible states for the head. So an upper bound on number of steps is $2^{p(n)}p(n)s$, which is exponential in $p(n)$. ■

Claim 5 *$P \subseteq PSPACE$.*

Proof Obviously a P machine can't access more than a polynomial number of memory locations. ■

Claim 6 *$NP \subseteq PSPACE$.*

Proof Enumerate all possible polynomial-size proofs. ■

Remark Does NP=PSPACE? Just like P vs. NP, this is a longstanding open problem! It's conjectured that they're different.

Definition 7 *LOGSPACE is the class of decision problems solvable by a Turing machine with $O(\log n)$ bits of memory.*

Remark But isn't the input already n bits? A technicality: The LOGSPACE machine has *read-only access* to the input. It's only the *read-write* memory that's logarithmically bounded.

Claim 8 *$LOGSPACE \subseteq P$.*

Proof Same as Claim 4, except "scaled down by an exponential." There are at most $s2^{c \log n} c \log n = n^{O(1)}$ possible configurations of a Turing machine with $c \log n$ tape squares. ■

Remark Does LOGSPACE=P? Another longstanding open problem! Again, conjecture is that they are different.

We *can* prove $LOGSPACE \neq PSPACE$, using a Space Hierarchy Theorem similar to the Time Hierarchy Theorem that we saw in Lecture 7. As usual, it's not hard to prove that *more of the same resource* (time, space, etc) provides more computational power than less of it. The hard part is to compare *different* resources (like time vs. space, or determinism vs. nondeterminism).

3 Why do we believe $P \neq NP$ if we can't prove it?

- Hardness of solving NP-complete problems in practice: the empirical case.
- There are "vastly easier" problems than NP-complete ones (like factoring) that we already have no idea how to solve in P.
- As Gödel argued, P=NP would mean mathematical creativity could be automated. God would not be so kind!
- We know that $LOGSPACE \neq PSPACE$. But this means either $LOGSPACE \neq P$, or $P \neq NP$, or $NP \neq PSPACE$! And if one is true, then why not all three?

Incidentally, let me tell you one of the inside jokes of complexity theory. Let LINSPACE be the set of problems solvable in linear space. Then one of the very few separations we can prove is that $LINSPACE \neq P$. Why? Well, suppose $P = LINSPACE$. Then $P = PSPACE$ also. Why? Pad the inputs! But that means LINSPACE=PSPACE, which is ruled out by the Space Hierarchy Theorem!

The punchline is, while we know P and LINSPACE are different, we have no idea which one is not contained in the other one (or as is most likely, whether neither is contained in the other one). We just know that they're different!

4 Why is it so hard to prove $P \neq NP$?

- Because $P \neq NP$!
- Because there really are lots of clever, non-obvious polynomial-time algorithms. Any proof that 3SAT is hard will have to *fail* to show 2SAT is hard, even though the "handwaving intuition" seems the same for both (there are 2^n possible solutions, and clearly each one takes at least constant time to check!). Simple as it seems, this criterion already rules out almost every amateur $P \neq NP$ proof in Prof. Aaronson's inbox...
- Let NPSPACE (Nondeterministic PSPACE) be the class of problems solvable by a PSPACE machine that can make nondeterministic transitions. Then by analogy to $P \neq NP$, you might conjecture that $PSPACE \neq NPSPACE$. But *Savitch's Theorem* shows that this conjecture is false: PSPACE=NPSPACE. So any proof of $P \neq NP$ will have to fail when the polynomially-bounded resource is space rather than time.

- The Baker-Gill-Solovay argument. Techniques borrowed from logic and computability theory, like the ones used to prove the Time Hierarchy Theorem, all seem to *relativize*. In other words, they all seem to go through without change if we assume there's a magical oracle in the world that solves some problem for free. As an example, recall how the proof of unsolvability of the halting problem could easily be adapted to show the unsolvability of the "Super Halting Problem" by "Super Turing Machines". By contrast, any solution to the P vs. NP problem will have to be *non-relativizing*. In other words, it will have to be sensitive to whether or not there's an oracle around. Why? Well, because there are some oracle worlds where P=NP, and others where $P \neq NP$! Specifically, let A be any PSPACE-complete problem. Then I claim that $P^A = NP^A = PSPACE$. (Why?) On the other hand, we can also create an oracle B such that $P^B \neq NP^B$. For every input length n, either there will exist a $y \in \{0,1\}^n$ such that $B(y) = 1$, or else $B(y)$ will equal 0 for all $y \in \{0,1\}^n$. Then given an n-bit input, the problem will be to decide which. Certainly this problem is in NP^B (why?). On the other hand, at least intuitively, the only way a deterministic machine can solve this problem is by asking the oracle B about exponentially many y's. Baker, Gill, and Solovay were able to formalize this intuition (details omitted), thus giving an oracle world where $P \neq NP$.

5 Starting a new unit: Randomness

People have debated the true nature of randomness for centuries. It seems clear what we mean when we say two dice have a 1/36 probability of landing snake-eyes: that if you roll infinitely many times, in the limit you'll get snake-eyes 1/36 of the time. On the other hand, if you go to Intrade.com (at the time of this writing), you can find the market estimates a 70% probability for Obama to win the primary and a 30% probability for Hillary to win. But what does such a probability mean in philosophical terms? That if you literally reran the election infinitely many times, in the limit Obama would win 70% of the time? That seems ridiculous! You can make the puzzle sharper by supposing the election has already happened, and people are trying to guess, what's the probability that when the votes are counted, this candidate will win. Well, the votes are already there! Or at least, what the voting machine says are the votes are there. Or suppose someone asks you for the probability that P=NP. Well, they're either equal or not, and they've been that way since before the beginning of the universe!

Interestingly, in the context of Intrade, probability does have a pretty clear meaning. To say that Obama is estimated by the market to have a 70% chance of winning means, if you want to buy a futures contract that will pay a dollar if he wins and nothing if he loses, then the cost is 70 cents. There's a whole field called decision theory that *defines* probabilities in terms of what bets you'd be willing to make. There are even theorems to the effect that, if you want to be a rational bettor (one who can't be taken advantage of), then you have to act as if you believe in probabilities whether you like them or not.

And then there's the physics question: are the laws of physics fundamentally probabilistic, or is there always some additional information that if we knew it would restore determinism? I'm sure you all know the story of Einstein saying that God doesn't play dice, and quantum mechanics saying God does play dice, and quantum mechanics winning. The world really does seem to have a fundamental random component.

Fortunately, for CS purposes we don't have to worry about many of these deep issues. We can just take the laws of probability theory as axioms.