CS1200: Intro. to Algorithms and their Limitations Prof. Salil Vadhan

Lecture 21: The P vs. NP Problem

Harvard SEAS - Fall 2025 *2025-11-13*

1 Announcements

- Salil's OH today 1-1:45pm SEC 3.327, Mon 5:30-6:15pm Zoom.
- PS7 and SRE5 feedback
  - PS7 median and 75th percentile time spent: College 6hrs/8hrs, DCE 9hrs/12hrs.
  - Ethics content feedback: mostly very positive, but some felt that computer scientists shouldn't be making life or death decisions. The main takeaways were *supposed* to be (a) the algorithms we design often make such value-laden decisions when deployed, whether or not we intended it, and (b) computer scientists should engage with those appropriately trained to figure out how to take such considerations into account in algorithm design. One lecture on ethical considerations in cs1200 was not intended to constitute "appropriate training"!
  - Legibility of board becoming more challenging with subscripts and blue chalk. I'll try to improve it!
  - Some want more feedback on your problem set mistakes. You can use regrade requests for this purpose (when you don't understand what you did wrong, or how the staff solution is better)!
  - SRE5 needed more time. We are considering a shorter one for next year! But showing up on time (many people arrived late) and doing sufficient prep can help (both senders and receivers).

Recommended Reading:

- Hesterberg–Vadhan 23
- MacCormick Ch. 14, 17.

2 Search vs. Decision

Q: Does the seeming hardness of $\mathsf{NP}_{\mathsf{search}}$ come from the fact that there are many possible answers? What if we restrict to decision problems, with only two possible answers?

Definition 2.1. A computational problem $\Pi = (\mathcal{I}, \mathcal{O}, f)$ is a *decision problem* if $\mathcal{O} = \underline{\hspace{6cm}}$
and for every $x \in \mathcal{I}$, $|f(x)| = \underline{\hspace{3cm}}$

By definition,

$$\begin{aligned} \mathsf{P} &= \underline{\hspace{12cm}} \\ \mathsf{EXP} &= \underline{\hspace{12cm}} \end{aligned}$$

However, the decision class NP has a more subtle definition in terms of $\mathsf{NP}_{\mathsf{search}}$:

Definition 2.2 (NP)**.** A decision problem $\Pi = (\mathcal{I}, \{\texttt{yes}, \texttt{no}\}, f)$ is in NP if there is a computational problem $\Gamma = (\mathcal{I}, \mathcal{O}, g) \in \mathsf{NP}_{\mathsf{search}}$ such that for all $x \in \mathcal{I}$, we have:

$$\begin{aligned} f(x) = \{\texttt{yes}\} &\quad\Leftrightarrow\quad \underline{\hspace{6cm}} \\ f(x) = \{\texttt{no}\} &\quad\Leftrightarrow\quad \underline{\hspace{6cm}} \end{aligned}$$

Example 2.3. Let's see a couple of examples of problems in NP.

•

•

Another appealing interpretation of the class NP:

Pursuing this viewpoint, it turns out that there is a deep connection between mathematical proofs and NP, and this is one reason that the P vs. NP question is considered to be a central open problem in mathematics as well as computer science.

One nice feature of focusing on decision problems is that we can show that NP contains P:

Lemma 2.4. *$\mathsf{P} \subseteq \mathsf{NP}$.*

Proof.

$\square$

In contrast, as shown in the problem set, $\mathsf{P}_{\mathsf{search}}$ is *not* a subset of $\mathsf{NP}_{\mathsf{search}}$, since $\mathsf{NP}_{\mathsf{search}}$ requires that *all* solutions are short and easy to verify, whereas $\mathsf{P}_{\mathsf{search}}$ only tells us that at least one of the solutions is easy to find.

The "P vs. NP Question" is usually formulated as asking whether $\mathsf{P} = \mathsf{NP}$ (with the answer widely conjectured to be no). It turns out that this decision formulation is equivalent to the search version we have been studying:

Theorem 2.5 (Search vs. Decision)**.** $\mathsf{NP} = \mathsf{P}$ *if and only if* $\mathsf{NP}_{\mathsf{search}} \subseteq \mathsf{P}_{\mathsf{search}}$.

Q: Does this theorem remind you of anything you've seen?

A:

Proof of Theorem 2.5. Suppose that $\mathsf{NP}_{\mathsf{search}} \subseteq \mathsf{P}_{\mathsf{search}}$.

For the converse, assume that $\mathsf{P} = \mathsf{NP}$. Since SAT–DECISION is in NP, it is also in P and hence in $\mathsf{P}_{\mathsf{search}}$. By Lemma 2.6 below, SAT $\leq_p$ SAT–DECISION, so SAT is also in $\mathsf{P}_{\mathsf{search}}$. Since SAT is $\mathsf{NP}_{\mathsf{search}}$-complete, we have $\mathsf{NP}_{\mathsf{search}} \subseteq \mathsf{P}_{\mathsf{search}}$.

$\square$

Lemma 2.6. SAT $\leq_p$ SAT–DECISION.

Proof.

```
SATDecisionToSAT($\varphi$):
    Input   : A CNF formula $\varphi(x_0, \ldots, x_{n-1})$ (and access to an oracle
              $O$ solving SAT–Decision)
    Output  : A satisfying assignment $\alpha$ to $\varphi$, or $\perp$ if none exists.
0 if $O(\varphi) =$ no then return $\perp$;
1 foreach $i = 0, \ldots, n-1$ do
2     if $O($______________________________$) =$ yes then $\alpha_i = 0$;
3     else $\alpha_i = 1$;
4 return $\alpha = (\alpha_0, \ldots, \alpha_{n-1})$
```
Algorithm 2.1: Reduction from SAT–Decision to SAT

Runtime:

Correctness:

□

In most textbooks, the theory of NP-completeness focuses on decision problems (and names like SAT, Graph Coloring, etc. typically refer to the decision versions). As expected, we say a decision problem Π is NP-*complete* if $\Pi \in \mathsf{NP}$, and Π is NP-hard, meaning $\Gamma \leq_p \Pi$ for every problem $\Gamma \in \mathsf{NP}$. All of the $\mathsf{NP}_{\mathsf{search}}$-completeness proofs we have seen are also NP-completeness proofs of the corresponding decision problems, since they all use mapping reductions, which have the property that $R(x)$ has a solution iff x has a solution:

Theorem 2.7. SAT–Decision*,* 3-SAT–Decision*,* Independent Set–Threshold Decision*,* Subset Sum–Decision*, and* Long Path–Decision *are* NP*-complete.*

For NP-completeness proofs as in the above theorem, mapping reductions become even simpler; we only need the polynomial-time algorithm R that transforms `yes` instances to `yes` instances, and `no` instances to `no` instances. We don't need the algorithm S that maps solutions to the search problem on $R(x)$ back to solutions to the search problem on x.

3 The Breadth of NP-completeness

There is a huge variety of NP-complete problems, from many different domains:

The fact that they are all NP-complete means that, even though they look different, there is a sense in which they are really all the same problem in disguise. And they are equivalent in complexity: either they are all easy (solvable in polynomial time) or they are all hard (not solvable in polynomial time). The widely believed conjecture is the latter; $\mathsf{P} \neq \mathsf{NP}$.

4 Two Possible Worlds

If $\mathsf{P} = \mathsf{NP}$, then:

If $\mathsf{P} \neq \mathsf{NP}$, then: