The ecosystem around NP

Evgenij Thorstensen

V18

The general picture

We have P, NP and coNP, and then up there in the clouds PSPACE.

Outline of lecture

P vs. NP — why do we believe it?

NP and coNP weirdness.

Counting problems: the class #P

The famous $\mathsf{P} \neq \mathsf{NP}$ conjecture

We can't prove it. Reasons why this is difficult to prove will show up later in the course.

Fundamental question about the nature of computation. Can search be reduced to verification? If not, why?

Seems obvious that $\mathsf{P} \neq \mathsf{NP}$ (philosophically).

Why do we believe it to be true (what evidence do we have)?

Invisible electric fence

For a lot of problems (of then unknown complexity), weirdly specific numeric bounds happen.

Approximation algorithm for set cover can find a cover at most $\ln(n)$ larger than optimal.

Many years later, turns out that finding a cover at most $(1 - \epsilon)\ln(n)$ is NP-complete.

More here: https://www.scottaaronson.com/blog/?p=1720

Another invisible fence example

Quantum computing: Can solve factoring efficiently.

Surprise: Factoring is not known to be NP-complete, and believed not to be.

Quantum analogues of Pand NPhave the same hard separation problem (can't prove that they are different).

NP-intermediate problems

If $\mathsf{P} \neq \mathsf{NP}$, then there must be problems in $\mathsf{NP} \setminus \mathsf{P}$ that are *not NP-complete*.

Ladner's theorem. The idea is to construct a problem that alternates SAT instances with nothing.

The SAT instances get bigger and bigger, so they exclude more and more polynomial-time algorithms.

The nothing gaps get bigger too, excluding reductions from SAT to this problem.

Some NP-intermediate candidates

Factoring, obviously. But factoring is quite special.

Problem (Graph isomorphism)

Given two graphs G *and* H, *are they isomorphic?*

Subgraph isomorphism is NP-complete. Graph isomorphism is ???

Somehow nothing NP-complete reduces to graph isomorphism.

Counting the *number* of isomorphisms is polynomial-time reducible to deciding if one exists!

The class coNP

$\mathsf{coNP} = \{L \mid \bar{L} \in \mathsf{NP}\}$.

This is not the same as the complement of NP itself.

Polynomial-time verifiers for the no-instances. UNSAT is in this class.

If a formula is satisfiable, it is *not* a member of UNSAT.

Some facts about coNP

If $\mathsf{P} = \mathsf{NP}$, then $\mathsf{NP} = \mathsf{coNP}$. Consequently, if $\mathsf{NP} \neq \mathsf{coNP}$, then also $\mathsf{P} \neq \mathsf{NP}$.

However, if $\mathsf{P} \neq \mathsf{NP}$, we could still have $\mathsf{NP} = \mathsf{coNP}$.

NP and coNP

Two classes of interest: $\mathsf{NP} \cap \mathsf{coNP}$ and
$\mathsf{DP} : \{L_1 \cap L_2 \mid L_1 \in \mathsf{NP}, L_2 \in \mathsf{coNP}\}$

They are not the same class.

DP shows up in problems where we need a minimal solution.

Problem (Homomorphic equivalence)

Given graphs G and H, are they homomorphically equivalent?

Problem (Core)

Given graphs G and H, is H homomorphically equivalent to G, and the minimal such? (meaning that there is no subgraph of H with this property)

Cores

A homomorphism $G \to H$ is a function $h : V_G \to V_H$ such that $(v, w) \in E_G$ implies $(h(v), h(w)) \in E_H$.

Very useful notion. 3COL reduces to Hom.

Theorem

A simple undirected graph G is k-colourable if and only if there exists a homomorphism from G to K_k.

K_k is the complete simple undirected graph on k vertices.

Cores

For H to be a core of G, G, H must be a member of

- the NP language “H hom. equivalent to G” and
- not be a member of the language “subgraph of H is hom. equiv. to G”

Hence this is a DP problem.

What about NP $\cap$ coNP?

This is the class of problems where we can verify both yes and no instances in polynomial time.

Graph isomorphism is believed to be here, but not proven to be.

This class is not known to have complete problems.

Counting problems

Consider this variant of SAT: How many satisfying assignments does a given formula have?

Clearly this is at least as hard as SAT itself.

In fact, it is suspected to be a lot harder.

The class #P is the complexity class for counting the number of solutions to NP problems.

Formally, number of accepting paths in the NTM.

Properties of #P

It has complete problems (SAT).

Most curious, however: It has complete problems from P!

2SAT is in P. $\#2SAT$, however, is #P-complete.

Same for DNFSAT, a “trivial” decision problem.