Meeting 4.1: A smattering of complexity, continued

I. The usual suspects: P, FP, NP, PSPACE, EXP, #P, ER, ~~R~~, ~~RE~~

II. Reductions and hardness

Note

$$P \subseteq NP \subseteq PSPACE \subseteq EXP \subsetneq ER \subsetneq R \subsetneq RE$$

Expect these are all strict

ER: elementary recursive functions. To unpack, let's have $TIME(F(n))$ be all decision problems that can be solved on Turing machine that runs in time $O(F(n))$, where n is the size (i.e. length) of input.

Likewise, can define $SPACE(F(n))$.

Then

$$ER = \bigcup_{k \geq 1} TIME\Big(\underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}}_{k}\Big)$$

Nice exercise:

$$ER = \bigcup_{k \geq 1} SPACE\Big(\underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}}_{k}\Big)$$

$(SPACE(n) \subseteq TIME(2^n))$

Ex 3-Manifold Homeomorphism (Theorem of G. Kuperberg)

$$L : \{0,1\}^* \times \{0,1\}^* \to \{\text{Yes}, \text{No}\}$$

$$L(x,y) = \begin{cases} \text{Yes} & \text{if } x \text{ and } y \text{ encode homeo. 3-manifolds} \\ \text{No} & \text{otherwise} \end{cases}$$

Why? Geometrization: every 3-manifold can be cut up into pieces in a canonical way so each piece can be endowed with one of 8 Thurston geometries ($\mathbb{E}^3, S^3, \mathbb{H}^3, \ldots$)

Idea of algorithm: geometrize x and y in parallel, then compare their geometric pieces.

$$EXP = \bigcup_{k \geq 1} TIME(2^{n^k}) \overset{\text{"}}{=} 2^{O(poly(n))}$$

$$PSPACE = \bigcup_{k \geq 1} SPACE(n^k)$$

$PSPACE \subseteq EXP$ (Note: if we need $p(n)$ space for an algorithm, the Turing machine running the algorithm can be in at most $O(2^{p(n)})$ possible configurations.)

$$P = \bigcup_{k \geq 1} TIME(n^k).$$

If a problem is in P, we consider it efficiently solvable.

NP: nondeterministic polynomial time.

We say $L \in NP$ if there exists TM M and two polynomials $p(n)$, $q(n)$ such that for all input x to L of length n:

1. If $L(x) = \text{Yes}$, then exists $y \in \{0,1\}^{q(n)}$ such that $M(x,y) = \text{Yes}$.
2. If $L(x) = \text{No}$, $M(x,y) = \text{No}$ for all $y \in \{0,1\}^{q(n)}$
3. $M(x,y)$ runs in time $p(n)$ for all $y \in \{0,1\}^{q(n)}$.

For such a Turing machine we can call the $y \in \{0,1\}^{q(n)}$ "proofs" or "witnesses" or "certificates."

(They are not trustworthy, and M tests their credibility.)

Example SAT ("Boolean satisfiability")

Instances of SAT are Boolean formulas, e.g.

$$(x \vee y \vee z) \wedge (x \vee \neg y \vee t)$$

Problem: given a Boolean formula $f(y_1, y_2, \ldots, y_n)$, decide if there is an input such that f evaluates to 1 (or "True") on that input.

Why is SAT in NP? Take the different possible input to f as the certificates.

If $SAT(f) = Yes$, then of course some input to f evaluates to True. And of course if $SAT(f) = No$, no input will fool the procedure.

Ex Graph 3-colorability

Instance: Graph Γ, e.g. as adjacency matrix

Problem: Decide if Γ has a valid vertex 3-coloring.

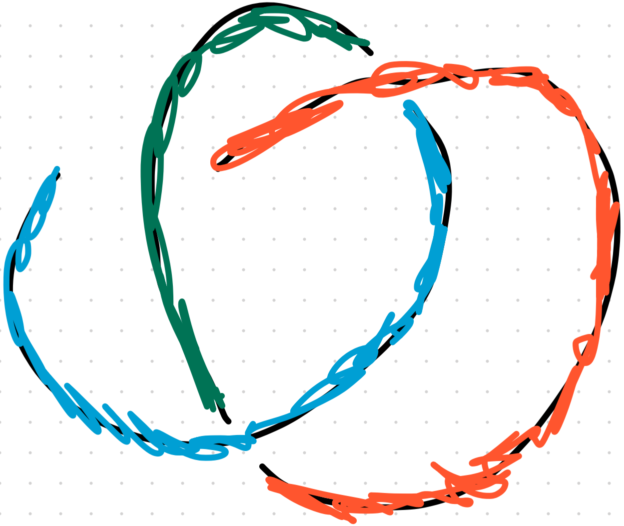
Witness: $\rho: V(\Gamma) \to \{R, G, B\}$

Given a witness, we can quickly verify whether or not it yields valid graph coloring.

Ex Knot 3-colorability

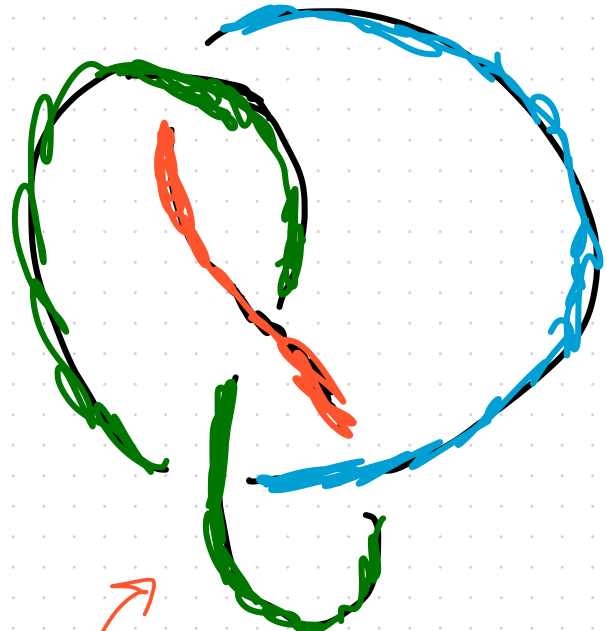
Instance: Knot diagram

Problem: Decide if we can color connected arcs of diagram with 3-colors so at each crossing, either 3 colors are seen, or just 1. Also require that we use all 3 colors.



Witnesses:

$\rho : \{\text{arcs}\} \to \{R, G, B\}$



Can check it's not 3-colorable.

II. Reductions and hardness

(also "polynomial time many-one")

Given two decision problems L and K, a Karp reduction r is polynomial time computable function such that for all $x \in \{0,1\}^*$ we have $L(x) = K(r(x))$.

$$\begin{tikzcd}
\{0,1\}^* \arrow[r, "L"] \arrow[d, "r"'] & \{\text{Yes}, \text{No}\} \\
\{0,1\}^* \arrow[ur, "K"'] &
\end{tikzcd}$$

Such an r is a reduction from L to K.

We interpret K as being at least as hard as L.

Another type of reduction:

polynomial-time Turing reductions (aka "Cook reductions")

Given a problem K, an oracle Turing machine for K is a usual Turing machine, together w/ a black box that solves instances of K in one time step.

P^K = all problems solvable in poly time on a Turing machine w/ oracle for K.

We say L is Cook reducible to K if $L \in P^K$.

Karp reduction $\Longrightarrow$ Cook reduction, but not converse.

A problem K is NP-hard if for all $L \in NP$, there exists a Karp reduction from L to K. If, moreover, $K \in NP$, we say K is NP-complete.

NP-complete = "in NP" + "NP-hard"

NP-hard is transitive under Karp reduction.

Theorem (Cook - Levin)

SAT is NP-complete.

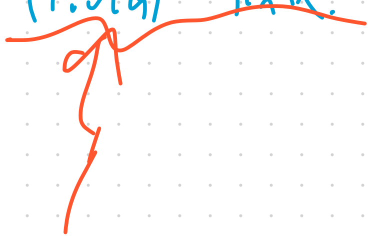
So is 3-SAT

Ex (de Mesmay, Rieck, Sedgwick, Tancer)

Trivial sublink problem is NP-~~complete~~ hard.

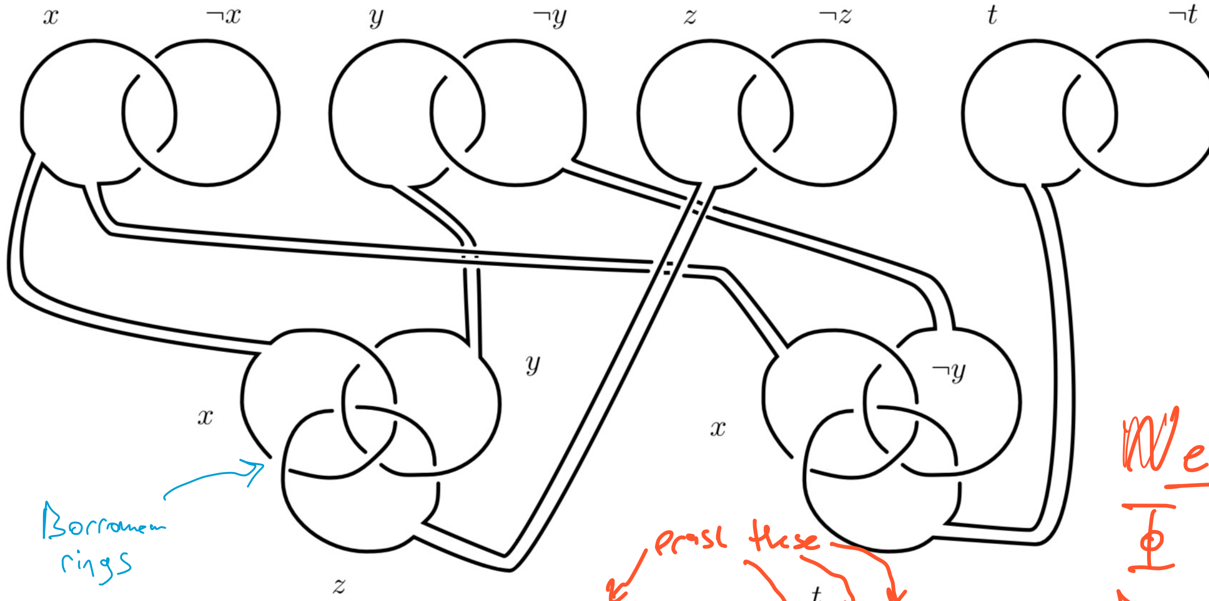
Instance: a link diagram L and natural number n

Problem: decide if L has an n-component unlink that is a trivial link.



e.g.

Proof sketch Reduce from 3-SAT



1 0 0 1 1 1

Figure 1: D_Φ for $\Phi = (x \vee y \vee z) \wedge (x \vee \neg y \vee t))$

Convert Φ to link diagram

D_Φ and n = # variables in Φ

Need to check:

Φ is satisfiable if and only if D_Φ has an n-component sub unlink.