What if a problem just <u>feels</u> hard to solve efficiently?

Eg: k-SAT

Recall:

In: variables $x_1, x_2, \ldots, x_n$ and a formula

$F = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ of clauses, where

each C_i of the form $\{y_1 \vee y_2 \vee \cdots \vee y_k\}$

+ where y_j is either x_t or $\neg x_t$ for some t.

Out: A boolean assignment to $\{x_1, \ldots, x_n\}$ such that F evaluates to TRUE, or NO if F is not satisfiable.

Toy Example: 'spose $n=3$, $m=2$, $k=3$.

One potential input formula could be:

$$F = \{x_1 \vee x_2 \vee \neg x_3\} \wedge \{\neg x_1 \vee \neg x_2 \vee \neg x_3\}$$

What would we get as output here?

A)

Let x_1 = True, x_2 = False, x_3 = False.

Then indeed, F is satisfiable.

$$\left(F = \{ \underset{T}{x_1} \vee \underset{F}{x_2} \vee \underset{T}{\neg x_3} \} \wedge \{ \underset{F}{\neg x_1} \vee \underset{T}{\neg x_2} \vee \underset{T}{\neg x_3} \} = T \wedge T = T \checkmark \right)$$

But hopefully you get the feel that this process might not scale well...

Naive Algorithm $\longrightarrow$ Try every assignment of x_i's.
(2^n of them)

Turns out, best known algorithm is basically that.

$O(2^{n-(cn/k)} m^d)$ time for constants c, d.

As k grows, this just goes to 2^n.

Therein lies the classical motivation for the "Strong Exponential Time Hypothesis" (SETH)

SETH: For every $\varepsilon > 0$, there is a k s.t.

k-SAT on n variables, m clauses, cannot be solved in $2^{(1-\varepsilon)n}$ poly m time.

Another tricky problem:

Orthogonal vectors (OV)

OV:

In: Sets S, T of N vectors in $\{0,1\}^d$

Out: Are there $u \in S$, $v \in T$ satisfying $u \cdot v = 0$?

Strategy: Reduce an arbitrary instance of k-SAT as an input to OV, whose solution is TRUE $\iff$ satisfiable.

Eg: k-SAT input: $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_4)$

(1.) Split variables into sets:

$V_1 = \{x_1, x_2\}$, $V_2 = \{x_3, x_4\}$

(2) Consider the partial assignments for V_j: ($2^{n/2}$ of them for $j = 1, 2$)

eg:

For V_1: $\{[x_1 = 0, x_2 = 0], [x_1 = 0, x_2 = 1], [x_1 = 1, x_2 = 0], [x_1 = 1, x_2 = 1]\}$

For each partial assignment ϕ of V_j, create $(m+2)$ length vector

$$v(j,\phi) = [\underbrace{0,1 \mid 1,0}_{\substack{V_j \\ \text{Basis}}} \mid \overbrace{\underbrace{\dots}_{\substack{0 \text{ if } \phi \text{ satisfies the clause,} \\ 1 \text{ else}}}}^{m \text{ clauses}}]$$

Eg] For $[0,0] \in V_1$, (i.e. $x_1 = 0$, $x_2 = 0$)

$$v(j,\phi) = v(1,[0,0]) = [\underbrace{0,1}_{\substack{\text{For all} \\ v(1,\phi)}}, 1, 0, 1]$$

Rmk: would be 1,0 for all $v(2,\phi)$

(Recall SAT input: $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_4)$)

$\boxed{\text{Thm:}}$ $v(1,\phi) \cdot v(2,\psi) = 0$ iff $\phi \odot \psi$ is a sat assignment.

$N = 2^{n/2}$ vectors of dimension $d = O(m) \rightarrow$ OV instance.

So, a solution to OV in $N^{2-\delta}\,\text{poly}(d)$ time for $\delta > 0$

$\Rightarrow 2^{n(1-\frac{\delta}{2})}\,\text{poly}(m)$ sol'n to SAT, & SETH is false.

What have we done?

1) - Encountered k-SAT & SETH for hardness

2) - Encountered OV

3) - Reduced k-SAT to an instance of OV, thereby establishing hardness for OV according to SETH.