3-SAT is NP-complete. \( \Rightarrow \) Don't expect polynomial time algorithm.

**ETH (Exponential Time Hypothesis)**

3-SAT requires \( 2^n \) time for some \( \alpha > 0 \) (No \( 2^{n/\log n} \) time algorithm)

Naive algo (brute force): \( 2^n \text{poly}(n) \) time

\( n = \# \text{vars} \)

Still interesting (and important) to beat brute force

- \( 2^n \) and \( 2^{n/2} \) are quite different

- Interesting algorithmic ideas

---

"Fine grained complexity"

- \( P \) vs. \( NP \) (\( P \) vs. \( \text{Exp Time} \))

Coarse distinction

Ultimate dream

- Alg with runtime \( c^n \) (\( c > 1 \))

- "Hard" to solve in \( (c-\epsilon)^n \) time

We are very far from such a picture

- \( \exists \) SAT (NP satisfiability with unbounded width clauses) can't be solved in \( (2-\epsilon)^n \) time for any \( \epsilon > 0 \)

\( \Rightarrow \) Strong Exponential Time Hypothesis (SETH)
Today's lecture: Some algorithms for 3-SAT

Trivial: \( O(m \, 2^n) \) time

\( n = \#\text{vars} \)
\( m = \#\text{clauses} \)

\((x_1 \lor x_2 \lor x_3) \land (x_4 \lor x_5 \lor x_6) \land (x_7 \lor x_8 \lor x_9) \land \ldots \)

---

Local search

Suppose we know an assignment \( A \) that is close to a satisfying assignment.

Start with assignment \( A \)

While \( \exists \) at least one unsatisfied clause in \( A \) and haven't looped for \( > r \) steps

a) Pick an arbitrary unsatisfied clause \( x_1 \lor y_1 \lor z_1 \)

b) Branch on each of the vars \( x, y, z \)

\[ A \leftarrow A \uparrow x=1 \]
\[ A \leftarrow A \uparrow y=1 \]
\[ A \leftarrow A \uparrow z=1 \]

At each node of the tree, check if assignment \( A \) is satisfying and halt if so.

Runtime: \( 3 \cdot \text{poly}(n) \)
Claim: If algo didn't terminate before depth \( r \), then one of the leaves will be \( A^* \).

\[ A \leftarrow \text{xyyz} \]

How to pick starting assignment \( A \)?

Try \( A = 0^n \) and \( A = 1^n \)

One of these is within Ham Dist \( \frac{n}{2} \) of \( A^* \)

Corr: \( 3^{n/2} \) poly \( (n) \) time also.

Randomized variant

- Pick several random states \( A \) and search within radius \( r \)

| \( B(r) \) | = \( p = \text{prob} \)

\( 2^n \) that random \( A \) is within \( r \) of \( A^* \)

\( |B(r)| = \sum_{i=0}^{n} \binom{n}{i} \)

3 \( r \) \text{ time}
# trials \geq \frac{1}{p} = \frac{2^n}{B(n)}
\binom{n}{r} \cdot \frac{2^n}{\binom{n}{r}} \cdot 3^n \geq # trials
\Rightarrow \text{Runtime of each trial}
\Rightarrow \text{Runtime} \leq (1.5) \text{ poly}(n)
Optimize \text{ in } r = \frac{n}{4} \text{ is best choice}

Randomized

Random walks algorithm (Schöning 1999)

Fact: Rand algo that succeeds with prob P = \frac{1}{\log n}
\Rightarrow \text{Rand algo of runtime } \frac{\text{poly}(n)}{P}
\text{ that succeeds with prob } 1 - 2^{-n}

1. Pick a random initial assignment A

While there is at least one unsatisfied clause in A & haven’t run for n steps already

(a) Pick an arbitrary unsatisfied clause
(b) Flip the value of a random var. of that clause

① ② ③

① ② ③

① ② ③

① ② ③
Analysis

Let $E$ be the event that $A$ and $A^*$ agree on $\geq \frac{n}{2}$ items.

$\Pr(E) \geq \frac{1}{2}$

$\Pr[\text{algo succeeds} \mid E] \geq \left(\frac{1}{3}\right)^{n/2}$

$\Pr[\text{algo succeeds}] \geq \frac{1}{2} \cdot \left(\frac{1}{3}\right)^{n/2} \geq 3^{n/2} \text{poly}(n) \text{ alg.}$

Better analysis of same algo:

$\Pr[\text{algo succeeds}] \geq \sum_{k=0}^{n} \binom{n}{k} 2^{-n} \left(\frac{1}{3}\right)^k \Pr[\text{dist}(A, A^*) = k]$ 

$= 2^{-n} \sum_{k=0}^{n} \binom{n}{k} \left(\frac{1}{3}\right)^k 2^n = 2^n \left(1 + \frac{1}{3}\right)^n = 2^n \left(\frac{4}{3}\right)^n = \left(\frac{2}{3}\right)^n$

$\Rightarrow$ Also with runtime \( \left(\frac{3}{2}\right)^n \text{ poly}(n) \)
An improved algorithm

Small change: Run loop for $3n$ steps instead of $n$

Analysis: Instead of a beeline from $A$ to $A^*$ analyze the chance of making at most $k$ incorrect steps within first $2k$ steps.

$$\Pr[\text{algo succeed}] \geq \sum_{k=0}^{\infty} \binom{n}{k} 2^{-n} \left( \frac{3}{4k} \right) \left( \frac{2}{3} \right)^{2k}$$

Compute this

$$\geq \left( \frac{3}{4} \right)^n \frac{1}{100^{\sqrt{n}}}$$

Gives $(4/3)^n \text{poly}(n)$ time algo

Almost the best known runtime which is $2 \times (1.31)^n$