Sign up on Piazza! R's OH W 3-4pm
Videos, lecture notes, V's OH TBD

So far: ETH + SETH. Why do people believe them?
(regarding k-SAT) What is known algorithmically?

Super-SETH: \exists \text{ an odd } u \text{-SAT in } 2^{n-\text{poly}(\log n)} \text{ time.}

Alg for 3SAT with few 3-clauses (clauses of width 3)

Thm: 3SAT is in \( O^*(1.5^t) \) time on formulas w/ \( \leq 3 \) clauses

randomized
\[ n^c \cdot 1.5^t \text{ Ex: } \text{poly}(\log n) \Rightarrow \text{poly time,} \]

Randomized reduction from 3SAT to 2SAT \( \in \text{P} \).

Think of \( F \) as set of clauses,
each clause as set of literals.

Have you seen 2SAT \( \in \text{P} \)?

RANDO\( (F) \): Repeat for \( 20 \cdot (1.5)^t \) trials:

Let \( F' \leq F \) be the 2cnf + 1cnf clauses.
For all 3-clauses \( C \in F \),
\[ \text{Rand. remove literal from } C, \text{ get } C'. \text{ Put } C' \text{ in } F'. \]
If \( F' \) is sat then return \text{SAT}.

Return \text{UNSAT}.

Key: \forall F', assign A, \((F' \text{ sat. by } A) \Rightarrow (F \text{ sat. by } A)\).

F \text{ UNSAT } \Rightarrow \text{RANDO}(F) \text{ returns } \text{UNSAT}.

F \text{ sat. by } A \Rightarrow \forall 3\text{-clauses } C \in F,
\[ \text{Pr}(A \text{ sat., } C') \geq 2^{-1/3}. \]

PF: IF \( A \) sat. 2or3 literals \( \in C' \),
then \( A' \) sat. \( C' \).

What's prob \[ \text{RANDO}(F) = \text{UNSAT}? \]
\[ \text{Pr}(A \text{ sat., } C') \geq 2^{-1/3}. \]
Each random lit. is independent,
\[ \text{Pr}(\forall 3\text{-clauses } C, A \text{ sat., } C') \geq (2/3)^t. \]
A sat, \( F' \) at end.
Repeat \( r = 20 \cdot (1.5)^t \) trials
\[ \Rightarrow \text{Pr}( \forall F' \text{ is SAT over all trials}) \leq (1 - (2/3)^r)^t \leq e^{-20(1.5)^t(2/3)^t} \]
prob of error \( < 10^{-9} \).
Branching/Backtracking

These most resemble real-world SAT solvers.

Pick a var, set cleverly. Repeat. Hopefylly "guessed" SAT assign...

If not, back up & try other vars/values

Try to learn from mistakes

Thm: k-SAT on n vars is in $2^{n - \frac{\log n}{k}}$ time

A(F): $F = \emptyset \Rightarrow \text{SAT}$

$\emptyset \not\in F \Rightarrow \text{UNSAT}$

If $F$ has no clauses $\Rightarrow$ every assign. satisfies!

If $F$ has an "all-false" clause

Take $C = \{l_1, \ldots, l_c\} \in F$, $c \leq k$ has $2^c - 1$ sat assigns.

For all $2^c - 1$ sat assigns $a$ to $C$,

$F' \not\in a$ plug $a$ into $F$. Explode clauses set to true by $a$.

return UNSAT,

Correctness: obvious? Only trying SAT assigns...

Time: Worst-case is $c = k$: $T(n) \leq (2^k-1) \cdot T(n-k) + \text{poly}(n)$.

$T(n) \leq (2^k-1)^\frac{1}{k} \cdot \text{poly}(n)$.

$\frac{1}{k} \leq e^\frac{-1}{k}$

Cpr: $\forall k \exists \delta < 1$ k-SAT in $O(2^{\delta n})$ time

[set $\delta = 1 - \frac{1}{\beta(2k)}$]

SETH: $\exists k \forall \delta < 1$ k-SAT in $O(2^{\delta n})$ time

Thm: k-SAT is in $2^{n - \frac{\log n}{k}}$ time.

B(F): $F = \emptyset \Rightarrow \text{SAT}$

$\emptyset \not\in F \Rightarrow \text{UNSAT}$

Take $C = \{l_1, \ldots, l_c\} \in F$, $c \leq k$

Call B on: $F$ w/ $l_i$ set

return SAT iff at least one call returns SAT.
Correctness: \( F \) sat. by \( A \rightarrow \forall C, \exists i \ st. l_i \ is \set{1st \ literal \ in} \ C \)
set true by \( A \).

\begin{align*}
\text{worst-case: } c & = n \\
T(n) & \leq \sum_{i=1}^{n-1} T(n-i) + \text{poly}(n). \\
\text{Hairy, but } T(n) & = 2^{n - \frac{n \cdot \log_2(n)}{5 \cdot 2^n}} \text{ works... (see lec. notes)}
\end{align*}

"Master Thm" for Backtracking Recurrences:
Every recurrence of the form \( T(n) \leq T(n-k_1) + \ldots + T(n-k_t) + \text{poly}(n) \)
has solution \( T(n) \leq O(r^n \cdot \text{poly}(n)) \)
where \( r \) is a positive root of \( P(x) = 1 - \frac{\sum_{i=1}^{t} x^{-k_i}}{r} \).

Ex 1: \( T(n) \leq 2 \cdot T(n-1) + \text{poly}(n) \)
\( \quad \Rightarrow \ T(n) \leq O(2^n) \\
\quad \text{P}(x) = 1 - 2x^{-1} \text{ has } x = 2 \text{ as a root} \)

Ex 2: \( T(n) \leq T(n-1) + T(n-2) + \text{poly}(n) \)
\( \quad \text{P}(x) = 1 - x^{-1} - x^{-2} = 0 \Leftrightarrow x^2 - x - 1 = 0 \Leftrightarrow x(x-1) = 1 \Leftrightarrow x = \frac{1 \pm \sqrt{5}}{2} \)

\( \text{positive sol. : } x = \frac{1 + \sqrt{5}}{2} \approx 1.618... \)

PPS2: Fastest known \( k\)-SAT alg., "randomized branching"
Paturi, Pudlak, Saks, Szegedy

Ptime alg. Simplify \( (F) \): If \( \exists \ 1\text{-clause } \{ \overline{x} \} \), set \( \overline{x} \) to true.

\begin{align*}
\text{Generalizing:} & \quad \text{If } \exists \text{ subset } S \subseteq F, |S| \leq 100 \\
& \quad \text{If } \exists \text{ variables } y \in S \text{ st. all SAT assigns } y \text{ to fixed } v \in \{0,1\} \\
& \quad \text{then set } y := v \text{ in } F.
\end{align*}

PPS2 \( (F) \) :
Repeat until all vars assigned
\[
\begin{align*}
\text{Run Simplify } (F) \text{ until } F \text{ no longer changes} \\
\text{Pick random unassigned } x \\
\text{Set } x \text{ to random 0/1 value.}
\end{align*}
\]
Thm: \( F \text{ SAT} \iff \Pr[\text{PPSZ returns SAT assign}] \geq \frac{1}{2^{n-o(n)}} \)

\( (\text{PPSZ}) \quad k\text{-CNF} \quad \exists c \text{ 4CNF } \forall c \text{ F SAT PPSZ needs } \leq 2^{n-o(n)} \text{ time.} \)

"Super-Strong ETH holds for PPSZ"

Local Search

"best" SAT solvers in the 90s used this...

Thm: \( k\text{-SAT is in } 2^{n-o(n)} \text{ time}. \) Super-SETH says "best possible"

\[ \exists c \text{ 4CNF } \forall c \text{ F SAT } 2^{n-o(n)} \text{ time} \]

Thm (P'92): Randomized alg for 2-SAT in \( \text{poly}(n) \) time.

\( \text{LS}(F): \) Let \( A \in \{0,1\}^n \) be random.

\[ 2\text{CNF} \quad \text{Repeat for } 20n^2 \text{ times:} \]

- If \( A \text{ sat, } F \text{ return SAT.} \)
- Else pick clause \( c \in \text{F} \) that \( A \) falsifies.
- Pick \( u \in \text{C} \) at random, \( \epsilon \) prob \( \frac{1}{2} \) for both vars.
- Flip value of \( u \) in \( A \).

Return UNSAT.

\( \text{LS}(F) \text{ returns SAT } \iff F \text{ is sat.} \)

Claim: \( F \text{ is sat, } \Pr[ \text{LS}(F) \text{ returns UNSAT}] < 1/10. \)

Sketch: Let \( A^* \text{ sat, } F, \text{ worst-case: } A^* \text{ is only SAT assign,} \)

Associate LS w/ random walk on a line.

<table>
<thead>
<tr>
<th>Target!</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>\ldots</th>
<th>( n )</th>
<th>node i</th>
<th>\text{set of } A \in {0,1}^n \text{ that differ in } i \text{ bits from } A^*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assign</td>
<td>( A^* )</td>
<td>( n )</td>
<td>( (\downarrow) )</td>
<td>( \downarrow ) assign</td>
<td>( = { A \mid h(A, A^*) = i } )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LS has some \( A \) in memory, associated w/ some node \( i \).

If \( A = A^* \), at node 0, we're done.
If \( A \neq \emptyset \), \( LS \) flips the value of a \( \textbf{var} \) in a clause \( \{x, y, z\} \)

\[ \text{prob } \frac{3}{2} \text{ of moving } \leftarrow \text{ node } i \rightarrow \text{ node } i-1, \]

\[ \text{prob } \frac{1}{2} \text{ of moving to } i+1. \]

How long until we reach node 0?

Random walk on \((n+1)\)-node line graph.

\[ \text{Pr} \left( \text{reach node } 0 \text{ after } \leq 20 \cdot n^2 \text{ steps } \right) \geq \frac{3}{4}. \]

Can use lower bounds on tail of binomial distribution, or results on cover times of random walks:

\[ X_v = \# \text{ steps before every node is visited, starting at node } v \]

\[ \text{Thm } E[X_v] \leq 2 \cdot n \cdot |V| \cdot |E| \leq 2n^2. \]

\[ \text{Pr(\text{don't reach } 0 \text{ after } 20 \cdot n^2 \text{ steps})} \leq \text{Pr}(X_n \geq 10 \cdot E[X_v]) \leq \frac{1}{10}. \]

Markov’s inequality.

Random walk for 3SAT? Probabilities change!

Only \( \frac{1}{3} \) prob. of moving toward node 0,

\( \frac{2}{3} \) prob of moving away.

\[ \text{Thm: } k\text{-SAT is in } 2^{n-o(n)} \text{ time.} \]

**Schön(F): Choose random } A \in \{0, 1\}^n\]

\[ \text{Repeat } \frac{n}{k} \text{ times:} \]

- If } A \text{ sat } F \text{ return } A
- Else let } C \text{ be falsified by } A\]

\[ \text{Pick random } u \in C, \text{ flip value of } u \text{ in } A \]

Return "FAIL"
Let $A^x$ sat. $F$.

**Claim:** $Pr[\text{Schön}(F) = A^x] \geq \frac{1}{2^n \cdot e^{-n/4k^2} \cdot (n+1)}$  

So we can repeat Schön for $O(2^{n-cn})$ times, and get a sat. assign. to $F$ whp.

**Pf:** $E = "\text{initial } A \text{ differs from } A^x \text{ in } \leq \frac{n}{4k} \text{ bits}"

$Pr[E] = \frac{\binom{n}{n/4k}}{2^n} \leq \# \text{ strings w/ } n/4k 1's

$\implies h(A, A^x) = \frac{n}{4k}$

$Pr[\text{ L.S. for } n/4k \text{ steps finds } A^x | E] \geq \left(\frac{1}{k}\right)^{n/4k}$

for each var chosen, $1/k$ prob. that flipping gets 1 bit closer.

If we are correct $n/4k$ times in a row, we must have $A^x$.

$Pr[\text{Schön}(F) = A^x] \geq Pr[E] \cdot Pr[\text{ L.S. finds } A^x | E]$

**Pf of Claim:** Use

1. $1 - x \leq e^{-x}$

2. $\binom{n}{x} \geq \frac{(\frac{n}{x})^x \cdot (1 - \frac{x}{n})^{n-x}}{(n+1)}$

$\frac{\binom{n}{n/4k}}{2^n} \cdot \frac{1}{k^{n/4k}} \geq \frac{k^{n/4k} \cdot (1 - \frac{n}{4k})^{n-n/4k}}{2^n \cdot (n+1) \cdot (1 - \frac{n}{4k})^{n-n/4k}}$

What's happening? Instead of spending $\binom{n}{n/4k}$ time to try all assignments "within $n/4k$" of an assign., use formula to search the Hamming ball in only $\approx k^{n/4k}$ time.
Can derandomize Schön! Make only deterministic choices. See lec notes.
- derandomize choice of assignment: use covering codes of radius \( \delta \).
- derandomize var. flips: try all \( k \) choices for the flipped var. 
- back track if no SAT assign. found after \( k \) flips \( \Rightarrow \Pr(k) \leq \frac{2^n}{\binom{2n}{n}} \).

Schönig's actual analysis is somewhat better:

allows for possibility of walking some fraction of steps in the "bad" direction, but walking more steps in "good" direction

\( \frac{1}{\binom{2n}{n}} \) prob of reaching \( \mathcal{A}^* \) from \( \mathcal{A} \)
can get \( \frac{1}{(\frac{n}{2n})^{\binom{2n}{n}}} \). Improves \( O(1) \) factor in exponent. See lec. notes.

An Equivalent Version of SETH

Important to know for proving hardness from SETH.

Recall: SETH \( \forall \epsilon > 0 \) \( \exists \epsilon \)-SAT not in \( 2^{(1-\epsilon)n} \) time

Suppose I could refute a variant of SETH:
\( \forall \epsilon > 0 \), CNF-SAT with \( cn \) clauses in \( 1.9^n \) time.

\( \approx \) doesn't care about clause width, only "density" of clauses

Claim: \( \forall \epsilon \Rightarrow \Rightarrow \text{SETH} \)

\( \forall \epsilon \), \( \exists \epsilon \)-SAT, \( \forall \epsilon \)-CNF \( F \), reduce to

\( \forall \epsilon \rightarrow \) can solve each \( F' \) in \( 1.9^n \) time.

Set \( \epsilon = 0.01 \rightarrow 1.9^n \cdot 2^{0.01n} \leq 1.92^n \). \( \square \)

So SETH \( \Rightarrow \) \( \forall \epsilon \). Call this "MBSETH"  

MBSETH: \( \forall \epsilon > 0 \), \( \exists \epsilon \)-CNF-SAT on \( cn \) clauses not in \( 2^{(1-\epsilon)n} \) time

This: \( \text{MBSETH} \Leftrightarrow \text{SETH} \) (!)

\( \Rightarrow \): (\( \Leftarrow \)) already done

(\( \Rightarrow \)): Given \( 2^\delta n \) time alg for \( k \)-SAT, \( \forall \epsilon > 0 \), \( \exists \epsilon \)-CNF-SAT \( cn \) clauses in \( 2^{(1-\epsilon)n} \) time
Let \( \mu \) be a parameter.

\[ \text{CNF, \( \mu \) clauses} \]

**SPARSO(\( \mu \))**:

1. **Base case**:
   - If \( F \) is \( \mu \)-SAT with \( n \) vars, solve in \( 2^{\Delta n} \) time.
   - Take \( \text{CEF} \) s.t. \( C = (l_1 u l_2 u \ldots u l_k u k+1 u \ldots) \)
   - Call \( \text{SPARSO} \) on:
     - \( F - \{ \forall \}(l_1 u l_2 u \ldots u l_k u k+1 u \ldots) \)  \( \triangleq \) replace \( C \) w/ first \( k \) literals
     - \( F \) w/ \( l_i = 0, \ldots, l_k = 0 \) \( \triangleq \) all \( k \) bits set \( \text{false} \)
   - Return \( \text{UNSAT} \) \( \triangleright \) 1) \& 2) return \( \text{UNSAT} \).

At base case, if 2 was called \( i \) times, there are \( \leq n - \mu k \) vars

\[ \Rightarrow \quad n/\mu \leq \sum_{i=0}^{\mu k} \binom{n+i}{i} \cdot 2^{-\Delta i} \leq 2^{\Delta n} \cdot (n/\mu)^{\mu k} \leq (2^{-\mu k})^i \]

\[ \text{(A)} \]

\[ \text{Set } \kappa := 10 \log(2) \cdot \log(\mu) \]

\[ \Rightarrow \quad \left( \frac{c n + n \kappa}{n \kappa} \right) < 2 \Delta n \]

\[ \Rightarrow \quad \left( \frac{c (n+k)}{n \kappa} \right) < 2 \Delta n \]

\[ \Rightarrow \quad \delta + \epsilon = \frac{1}{2} + \delta + \epsilon < 1, \quad \delta, \epsilon < 1 \]

\[ \Rightarrow \quad \text{algorithm runs in } 2^{(\delta \pm \epsilon) n} \text{ time.} \]

---

For more details, see lecture notes.