Motivation:
- Longest Common Subsequence (LCS)
- Edit Distance
- SETH lower bounds

SETH and other hypotheses.

Making SETH more believable: BP-SETH.

Results:
- Simulating branching programs with Edit Distance and LCS.
- Consequences of shaving log factors: $n^2 / \log^{\omega(1)} n$ algorithm for Edit Distance or LCS $\Rightarrow$ NEXP $\not\subseteq$ non-uniform NC$^1$.

Proofs.
Longest Common Subsequence (LCS)

Given two strings of length $n$, what is the length of the longest common subsequence?

$$
A \ T \ G \ C \ T \ T \ C \ G \ G \ C \ A \ A \ A
$$

$$
C \ T \ G \ G \ T \ A \ G \ C \ A \ A \ T \ C
$$
Longest Common Subsequence (LCS)

- Given two strings of length $n$, what is the length of the longest common subsequence?

  $$
  \begin{align*}
  A & \quad T & \quad G & \quad C & \quad T & \quad T & \quad C & \quad G & \quad G & \quad C & \quad A & \quad A \\
  C & \quad T & \quad G & \quad G & \quad T & \quad A & \quad G & \quad C & \quad A & \quad A & \quad T & \quad C
  \end{align*}
  $$

- Dynamic programming, $O(n^2)$ time:

  $$
  LCS(x[i], y[j]) = \max \left\{ LCS(x[i - 1], y[j - 1]) + 1_{x_i = y_j}, \quad LCS(x[i - 1], y[j]), \quad LCS(x[i], y[j - 1]) \right\}
  $$

- Masek and Paterson [1980]: $O(n^2 / \log^2 n)$ time algorithm.
How many insertions, deletions, and substitutions transform one given string into another?

APPLE →
OAPPLE →
ORAPPLE →
ORANPLE →
ORANGLLE →
ORANGE

Dynamic programming: \(O(n^2)\) time for strings of length \(n\).

Masek and Paterson [1980]: \(O(n^2 / \log_2 n)\) time algorithm.
How many **insertions**, **deletions**, and **substitutions** transform one given string into another?

\[
\begin{align*}
\text{APPLE} & \rightarrow \\
O\text{APPLE} & \rightarrow \\
O\text{RAPPLE} & \rightarrow \\
O\text{RANPLE} & \rightarrow \\
O\text{RANGLLE} & \rightarrow \\
O\text{RANGE} & 
\end{align*}
\]

- Dynamic programming: \( O(n^2) \) time for strings of length \( n \).
- Masek and Paterson [1980]: \( O(n^2 / \log^2 n) \) time algorithm.
Local Alignment

Input: two (DNA) sequences of length $n$ and a scoring matrix.

**AGCCCGTCTACGTGCAACCGGAAAGTATA**

**AAACGTGACGAGAGAGAGAACCCATTACGAA**

Output: The optimal alignment of two substrings.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>+1</td>
<td>-1.4</td>
<td>-1.8</td>
<td>-0.7</td>
<td>-1</td>
</tr>
<tr>
<td>C</td>
<td>-1.4</td>
<td>+1</td>
<td>-0.5</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>G</td>
<td>-1.8</td>
<td>-0.5</td>
<td>+1</td>
<td>-1.9</td>
<td>-1</td>
</tr>
<tr>
<td>T</td>
<td>-0.7</td>
<td>-1</td>
<td>-1.9</td>
<td>+1</td>
<td>-1</td>
</tr>
<tr>
<td>-</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>+∞</td>
</tr>
</tbody>
</table>

- Smith and Waterman [1981]: $O(n^2 / \log^2 n)$ time algorithm.
  - Too slow for DNA sequences.
- BLAST [Altschul et al., 1990]: Basic Local Alignment Search Tool.
  - A **heuristic** algorithm for Local Alignment.
  - 56,000 citations.
Conditional lower bounds

Assuming the Strong Exponential Time Hypothesis (SETH), for all $\epsilon > 0$:

- No $O(n^{2-\epsilon})$ time algorithm for Local alignment.  
  [Abboud, Vassilevska Williams, and Weimann, 2014]

- No $O(n^{2-\epsilon})$ time algorithm for Edit Distance.  
  [Backurs and Indyk, 2015]

- No $O(n^{2-\epsilon})$ time algorithm for LCS.  
  [Abboud, Backurs, and Vassilevska Williams, 2015; Bringmann and Künnemann, 2015]
Conditional lower bounds

Assuming the **Strong Exponential Time Hypothesis (SETH)**, for all $\epsilon > 0$:

- No $O(n^{2-\epsilon})$ time algorithm for **Local alignment**.  
  [Abboud, Vassilevska Williams, and Weimann, 2014]
- No $O(n^{2-\epsilon})$ time algorithm for **Edit Distance**.  
  [Backurs and Indyk, 2015]
- No $O(n^{2-\epsilon})$ time algorithm for **LCS**.  
  [Abboud, Backurs, and Vassilevska Williams, 2015; Bringmann and Künemann, 2015]
- No $O(n^{2-\epsilon})$ time algorithm for **similarity measures with alignment gadgets**.  
  [Bringmann and Künemann, 2015]
  - Edit Distance, LCS, Dynamic Time Warping Distance, . . .
Other (tight) SETH-based lower bounds

- **k-Dominating Set.**
  [Pătraşcu and Williams, 2010]

- **Diameter Approximation.**
  [Roditty and Vassilevska Williams, 2013]

- **Subtree Isomorphism.**
  [Abboud, Backurs, Hansen, Vassilevska Williams, and Zamir, SODA 2016]

- **Dynamic reachability.**
  [Abboud and Vassilevska Williams, 2014]

- **Frechet distance.**
  [Bringmann, 2014]

- **Succinct Stable Matching.**
  [Moeller, Paturi, and Schneider, 2015]
CNF formula with $n$ variables and $m$ clauses:

$$\phi = (x_1 \lor x_2 \lor \bar{x}_3 \lor \bar{x}_4) \land (\bar{x}_1 \lor \bar{x}_5) \land \cdots \land (x_7 \lor \bar{x}_9 \lor \bar{x}_{n-3} \lor x_n)$$

- **CNF-SAT**: Is a given CNF formula satisfiable?
- **$k$-SAT**: Is a given CNF formula with $k$ literals per clause satisfiable?
CNF formula with \( n \) variables and \( m \) clauses:

\[
\phi = (x_1 \lor x_2 \lor \bar{x}_3 \lor \bar{x}_4) \land (\bar{x}_1 \lor \bar{x}_5) \land \cdots \land (x_7 \lor \bar{x}_9 \lor \bar{x}_{n-3} \lor x_n)
\]

- CNF-SAT: Is a given CNF formula satisfiable?
- \( k \)-SAT: Is a given CNF formula with \( k \) literals per clause satisfiable?

- Best (randomized) algorithms:
  - 3-SAT: \( O(2^{0.388n}) \) [Hertli, 2011]
  - 4-SAT: \( O(2^{0.555n}) \) [Hertli, 2011]
  - \( k \)-SAT: \( 2^{(1-1/O(k))n} \) [Paturi, Pudlák, Saks, and Zane, 2005]
  - CNF-SAT: \( 2^{(1-1/O(\log(m/n)))n} \) [Calabro, Impagliazzo, and Paturi, 2006; Dantsin and Hirsch, 2009]
The Strong Exponential Time Hypothesis (SETH)

Impagliazzo and Paturi [1999]:

**Hypothesis (Exponential Time Hypothesis (ETH))**

$3$-SAT on $n$ variables requires $2^{\delta n}$ time for some $\delta > 0$.

**Hypothesis (Strong Exponential Time Hypothesis (SETH))**

There is no $\epsilon > 0$ such that $k$-SAT on $n$ variables can be solved in $2^{(1-\epsilon)n}$ time for all $k \geq 3$.

- **SETH**: CNF-SAT cannot be solved in $2^{(1-\epsilon)n}$ time.
The Strong Exponential Time Hypothesis (SETH)

Impagliazzo and Paturi [1999]:

Hypothesis (Exponential Time Hypothesis (ETH))

3-SAT on \( n \) variables requires \( 2^{\delta n} \) time for some \( \delta > 0 \).

Hypothesis (Strong Exponential Time Hypothesis (SETH))

There is no \( \epsilon > 0 \) such that \( k \)-SAT on \( n \) variables can be solved in \( 2^{(1-\epsilon)n} \) time for all \( k \geq 3 \).

- **SETH**: CNF-SAT cannot be solved in \( 2^{(1-\epsilon)n} \) time.
- Sparsification: Even for CNF formulas of **linear size** [Impagliazzo, Paturi, and Zane, 2001], i.e., \( m \leq f(k) \cdot n \).
3SUM

- **3SUM**: Given $n$ numbers, are there 3 that sum to 0?
- Easily solvable in $O(n^2)$ time.

**Conjecture**

3SUM on $n$ integers requires $n^{2-o(1)}$ time.
**3SUM**

- **3SUM**: Given $n$ numbers, are there 3 that sum to 0?
- Easily solvable in $O(n^2)$ time.

### Conjecture

**3SUM on $n$ integers requires $n^{2-o(1)}$ time.**

- Baran, Demaine, and Pătraşcu [2005]:
  $O(n^2(\log \log n)^2 / \log^2 n)$ time (randomized) algorithm for integers.

- Grønlund and Pettie [2014]:
  $O(n^2(\log \log n)^2 / \log n)$ time (randomized) algorithm for real numbers.
All-pairs shortest paths (APSP)

- **APSP:** Given a weighted directed graph with $n$ nodes, find the distance between every pair of nodes.
- Floyd-Warshall algorithm [1962]: $O(n^3)$ time.

**Conjecture**

*APSP on $n$ nodes and $O(\log n)$ bit weights requires $n^{3-o(1)}$ time.*
All-pairs shortest paths (APSP)

- **APSP:** Given a weighted directed graph with \( n \) nodes, find the distance between every pair of nodes.
- Floyd-Warshall algorithm [1962]: \( O(n^3) \) time.

**Conjecture**

*APSP on \( n \) nodes and \( O(\log n) \) bit weights requires \( n^{3-o(1)} \) time.*

- Williams [2014]: \( n^3/2^{\Omega(\sqrt{\log n})} \) time randomized algorithm.
- Chan and Williams [SODA, 2016]: \( n^3/2^{\Omega(\sqrt{\log n})} \) time deterministic algorithm.
Hardness of polynomial time problems

- **CNF-SAT**
  - “SETH-hard class”
  - ... LCS
  - k-Dominating-Set
  - Dynamic reachability
  - Sparse Diameter
  - Edit-Distance
  - Frechet

- **3SUM**
  - “3SUM-hard class”
  - ... Polygon Containment
  - 3 points on a line
  - Listing Triangles

- **APSP**
  - “APSP-hard class”
  - ... Radius
  - Dynamic Weighted Matching
  - Negative Triangle
  - Median
Open problem: How do SETH, 3SUM, and APSP relate to each other?

Abboud, Vassilevska Williams, and Yu [2015] identified a problem that is harder than all three.
Open problem: How do SETH, 3SUM, and APSP relate to each other?

Abboud, Vassilevska Williams, and Yu [2015] identified a problem that is harder than all three.

Our work: Making SETH-based lower bounds more believable.

- Edit Distance
- Longest Common Subsequence
- Dynamic Time Warping Distance
- …
Satisfiability of more general functions

\[ f(x_1, x_2, \ldots, x_n) \]
Satisfiability of more general functions

\[ f(x_1, x_2, \ldots, x_n) \]
Satisfiability of more general functions

\[ f(x_1, x_2, \ldots, x_n) \]
Satisfiability of more general functions

\[ f(x_1, x_2, \ldots, x_n) \]

\[ (x_1 \lor x_2 \lor \bar{x}_3 \lor \bar{x}_4) \land (\bar{x}_1 \lor \bar{x}_5) \land \cdots \land (x_7 \lor \bar{x}_9 \lor \bar{x}_{n-3} \lor x_n) \]
NC circuits

- **NC\(^i\)** circuits:
  - AND, OR, NOT gates.
  - Depth \(O(\log^i n)\).
  - Fan-in 2.
  - Polynomial size.

\[ \text{NC} = \bigcup_i \text{NC}^i. \]

\[ \text{NC}^1 \subseteq \text{NC}^2 \subseteq \cdots \subseteq \text{NC}^i \subseteq \cdots \subseteq \text{NC}. \]
NC circuits

- **NC^i circuits:**
  - AND, OR, NOT gates.
  - Depth $O(\log^i n)$.
  - Fan-in 2.
  - Polynomial size.

- $\text{NC} = \bigcup_i \text{NC}^i$.
- $\text{NC}^1 \subseteq \text{NC}^2 \subseteq \cdots \subseteq \text{NC}^i \subseteq \cdots \subseteq \text{NC}$.

**Theorem (Barrington’s Theorem [1989])**

Any fan-in 2, depth $d$ circuit can be converted into an equivalent (deterministic) **branching program** of width 5 and length $4^d$.

NC^1 and NC circuits can be expressed as width 5 branching programs of polynomial and quasi-polynomial length, respectively.
Branching programs (BPs)

- Edge-labelled, directed, layered graph.
- Width: $W$ nodes per layer.
- Length: $T$ layers.
- Each layer is associated with a variable. A variable can be associated with many layers.
- An input $001$ is accepted iff it generates an $s$-$t$ path.
Nondeterministic branching programs

- **Nondeterminism**: Multiple edges with the same label can leave the same vertex.

- **Size**: Number of edges, at most $O(TW^2)$.

- An input $001$ is accepted iff it generates some $s$-$t$ path.
Nondeterministic branching programs

- **Nondeterminism**: Multiple edges with the same label can leave the same vertex.
- **Size**: Number of edges, at most $O(TW^2)$.
- An input 001 is accepted iff it generates some $s$-$t$ path.
- Branching programs can represent (nondeterministic) Turing machines that use space $O(\log W)$ and time $T$. 
A hierarchy of SAT problems

3-SAT ≤ CNF-SAT ≤ AC⁰-SAT ≤ ACC-SAT ≤ NC¹-SAT ≤ NC-SAT ≤ ...

- The satisfiability problem (SAT) exists for functions of varying generality.
- Algorithms for SAT problems:
  - 3-SAT: $O(2^{0.388n})$
  - CNF-SAT: $2^{(1-1/O(\log(m/n)))n}$
  - ACC-SAT: $2^n/n^{\omega(1)}$ [Williams, 2014].
  - NC¹-SAT: Even $2^n/n$ is unknown.
A SETH hierarchy

**NC-SETH:** SAT on NC circuits requires $2^{(1-\epsilon)n}$ time for every $\epsilon > 0$.

**Hypothesis (\(C\)-SETH)**

SAT on functions from \(C\) requires $2^{(1-\epsilon)n}$ time for every $\epsilon > 0$. 
A SETH hierarchy

- **NC-SETH**: SAT on NC circuits requires $2^{(1-\epsilon)n}$ time for every $\epsilon > 0$.

**Hypothesis (C-SETH)**

SAT on functions from $\mathcal{C}$ requires $2^{(1-\epsilon)n}$ time for every $\epsilon > 0$.

- Previous SETH-based lower bounds relied on the simplicity of CNF formulas.
- **We show**: A much richer structure remains unexplored.
- Our lower bounds are based on BP-SETH:
  - Nondeterministic branching programs.
  - Subexponential size, i.e., $2^{o(n)}$.
  - A class much larger than even NC.
Main result

Theorem

$SAT$ for nondeterministic branching programs with $n$ input variables, width $W$, and length $T$ can be reduced to $Edit$ Distance or $LCS$ on two binary strings of length

$$N = 2^{n/2} T^{O(\log W)}.$$
**Main result**

**Theorem**

SAT for **nondeterministic branching programs** with $n$ input variables, width $W$, and length $T$ can be reduced to **Edit Distance** or **LCS** on two binary strings of length

$$N = 2^{n/2} T^{O(\log W)}.$$  

**Corollary**

If Edit Distance or LCS are solvable in $O(N^{2-\epsilon})$ time, then SAT is solvable in $2^{(1-\delta)n}$ time, for some $\delta > 0$, for:

- branching programs of width 5 and length $2^{o(n)}$,
- circuits with fan-in 2 and depth $o(n)$ (including NC),
- nondeterministic Turing machines with $o(\sqrt{n})$ space, and
- boolean formulas of size $2^{o(n)}$. 
Consequences of shaving logarithmic factors

- Best algorithms for Edit Distance and LCS: $O(n^2 / \log^2 n)$
  [Masek and Paterson, 1980]
  - Can we do better?
Consequences of shaving logarithmic factors

- Best algorithms for Edit Distance and LCS: \( O(n^2 / \log^2 n) \)
  [Masek and Paterson, 1980]
  - Can we do better?
  - “The polynomial method”:
    - \( n^3 / \log^{\omega(1)} n \) time for APSP.
      [Williams, 2014]
    - \( n^2 / \log^{\omega(1)} n \) time for Longest Common Substring.
      [Abboud, Williams, and Yu, 2015]
    - \( n^2 / \log^{\omega(1)} n \) time for Batch Hamming Nearest Neighbors.
      [Alman and Williams, 2015]
  - How about \( n^2 / \log^{\omega(1)} n \) time for Edit Distance or LCS?
Best algorithms for Edit Distance and LCS: $O(n^2 / \log^2 n)$
[Masek and Paterson, 1980]
- Can we do better?
- “The polynomial method”:
  - $n^3 / \log^\omega(1) n$ time for APSP.
    [Williams, 2014]
  - $n^2 / \log^\omega(1) n$ time for Longest Common Substring.
    [Abboud, Williams, and Yu, 2015]
  - $n^2 / \log^\omega(1) n$ time for Batch Hamming Nearest Neighbors.
    [Alman and Williams, 2015]
- How about $n^2 / \log^\omega(1) n$ time for Edit Distance or LCS?
- **We show**: This implies NEXP $\not\subseteq$ non-uniform NC$^1$. 
Consequences of refuting SETH

- Refuting SETH shows that $\text{NEXP}$ cannot be solved by \textit{linear-size series-parallel circuits} [Jahanjou, Miles, and Viola, 2015].
  - Much weaker than $\text{NEXP} \not\subseteq \text{NC}^1$. 
E^{NP}: Decision problems computable in \(2^{O(n)}\) time with oracle access to NP.

NTIME[\(2^{O(n)}\)] \(\subseteq\) E^{NP}, NEXP.

**Theorem (Williams, 2014)**

If satisfiability of \(C\)-circuits with size \(S(n) \leq 2^{o(n)}\) is solvable in \(O(2^n/(n^{10} \cdot S(n)))\) time, then E^{NP} does not have non-uniform \(C\)-circuits of size \(S(n)\).

- E^{NP} can be replaced by NTIME[\(2^{O(n)}\)] when \(C \subseteq NC^1\).
- \(2^n/n^{\omega(1)}\) for ACC-SAT \(\Rightarrow\) NEXP \(\not\subseteq\) non-uniform ACC.
Circuit lower bounds

- $E^{NP}$: Decision problems computable in $2^{O(n)}$ time with oracle access to NP.
- $NTIME[2^{O(n)}] \subseteq E^{NP}, NEXP.$

Theorem (Williams, 2014)

If satisfiability of $C$-circuits with size $S(n) \leq 2^{o(n)}$ is solvable in $O(2^n/(n^{10} \cdot S(n)))$ time, then $E^{NP}$ does not have non-uniform $C$-circuits of size $S(n)$.

- $E^{NP}$ can be replaced by $NTIME[2^{O(n)}]$ when $C \subseteq NC^1$.
- $2^n/n^{\omega(1)}$ for ACC-SAT $\Rightarrow$ NEXP $\not\subseteq$ non-uniform ACC.
- $2^n/n^{\omega(1)}$ for NC$^1$-SAT $\Rightarrow$ NEXP $\not\subseteq$ non-uniform NC$^1$. 
Barrington’s Theorem applied to NC\(^1\) produces BPs with \(W = 5\) and \(T = 4^{O(\log n)} = n^{O(1)}\).

Our reduction constructs strings of length

\[
N = 2^{n/2} T^{O(\log W)} = 2^{n/2} n^{O(1)}.
\]

It follows that:

\[
\frac{N^2}{\log \omega(1)} \text{ time for LCS} \quad \Rightarrow \\
2^{n/n^\omega(1)} \text{ time for NC}^1\text{-SAT} \quad \Rightarrow \\
\text{NEXP} \not\subseteq \text{NC}^1
\]
Outline for the rest of the talk

- **Next:** The reduction from BP-SAT to LCS.
- I will present a simplified, direct reduction.
Outline for the rest of the talk

- **Next:** The reduction from BP-SAT to LCS.
- I will present a simplified, direct reduction.
- Bringmann and Künneemann [2015]: SETH-based lower bounds for sequence-problems with alignment gadgets.
  - Edit Distance, LCS, Dynamic Time Warping Distance, . . .
  - Efficient reductions that work for binary strings.
Next: The reduction from BP-SAT to LCS.

I will present a simplified, direct reduction.

Bringmann and Künemann [2015]: SETH-based lower bounds for sequence-problems with alignment gadgets.
- Edit Distance, LCS, Dynamic Time Warping Distance, . . .
- Efficient reductions that work for binary strings.

We prove a reduction from BP-SAT to their framework.
- This gives efficient reductions to, e.g., Edit Distance and LCS on binary strings.
Next: The reduction from BP-SAT to LCS.

I will present a simplified, direct reduction.

Bringmann and Kündemann [2015]: SETH-based lower bounds for sequence-problems with alignment gadgets.

- Edit Distance, LCS, Dynamic Time Warping Distance, . . .
- Efficient reductions that work for binary strings.

We prove a reduction from BP-SAT to their framework.

- This gives efficient reductions to, e.g., Edit Distance and LCS on binary strings.

We also prove a reduction to k-LCS; LCS on k strings.

Alignment gadgets and k-LCS will not be covered in this talk.
Previous reductions showing SETH-hardness for LCS:
- Williams [2004]: CNF-SAT to **Orthogonal Vectors** (OV).
- Abboud, Backurs, and Vassilevska Williams [2015]: OV to LCS-Pair, and LCS-Pair to LCS.
Definition (Satisfying pair problem)

Let $A, B \subseteq \{0, 1\}^{n/2}$ and $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be given. Is there a pair $(a, b) \in A \times B$ with $f(a, b) = 1$?

- The problem is equivalent to SAT when $A = B = \{0, 1\}^{n/2}$.
- BP-SAT-Pair: The **satisfying pair problem on branching programs**.
Definition (Satisfying pair problem)

Let $A, B \subseteq \{0, 1\}^{n/2}$ and $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be given. Is there a pair $(a, b) \in A \times B$ with $f(a, b) = 1$?

- The problem is equivalent to SAT when $A = B = \{0, 1\}^{n/2}$.
- BP-SAT-Pair: The satisfying pair problem on branching programs.
- BP-SAT-Pair generalizes Orthogonal Vectors.
The **Connecting-Pair Problem**?

Given two lists of $2^{n/2}$ subgraphs of an underlying graph on poly(n) nodes

$|A|=|B|=2^{n/2}$

is there a pair that connects $s$ and $t$?
**LCS-Pair**

**Definition (LCS-Pair)**

Given two sets of strings $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$, and a threshold $X$. Is there a pair $(a_i, b_j)$ with $LCS(a_i, b_j) \geq X$?

\[\begin{align*}
A & = 1000111 \\
& \quad 1100110 \\
& \quad 0100111 \\

B & = 1110101 \\
& \quad 0011101 \\
& \quad 1010000 \\
\end{align*}\]

Abboud, Backurs, and Vassilevska Williams [2015] gave a reduction from LCS-Pair to weighted LCS, and then to LCS.
Definition (LCS-Pair)

Given two sets of strings $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$, and a threshold $X$. Is there a pair $(a_i, b_j)$ with $LCS(a_i, b_j) \geq X$?

- Abboud, Backurs, and Vassilevska Williams [2015] gave a reduction from LCS-Pair to weighted LCS, and then to LCS.
Let $f : \{0, 1\}^n \to \{0, 1\}$ be defined by a given BP.
For each $a \in A = \{0, 1\}^{n/2}$ and $b \in B = \{0, 1\}^{n/2}$ we construct strings $G(a)$ and $\overline{G}(b)$, such that:

\[
LCS(G(a), \overline{G}(b)) = X \quad \text{if } f(a, b) = 1
\]
\[
LCS(G(a), \overline{G}(b)) \leq X - 1 \quad \text{otherwise}
\]

\[
A \quad |A| = |B| = N = 2^{n/2} \quad B
\]

$G(a_1) = 1000111$ \hspace{2cm} $\overline{G}(b_1) = 1110101$

$G(a_2) = 1100110$ \hspace{2cm} $\overline{G}(b_2) = 0011101$

$G(a_3) = 0100111$ \hspace{2cm} $\overline{G}(b_3) = 1010000$
From BP-SAT-Pair to LCS-Pair

Let \( f : \{0, 1\}^n \rightarrow \{0, 1\} \) be defined by a given BP.

For each \( a \in A = \{0, 1\}^{n/2} \) and \( b \in B = \{0, 1\}^{n/2} \) we construct strings \( G(a) \) and \( \overline{G}(b) \), such that:

\[
\begin{align*}
LCS(G(a), \overline{G}(b)) &= X \quad \text{if } f(a, b) = 1 \\
LCS(G(a), \overline{G}(b)) &\leq X - 1 \quad \text{otherwise}
\end{align*}
\]

\[
\begin{array}{c|c|c|c}
A & |A| = |B| = N = 2^{n/2} & B \\
G(a_1) = 1000111 & \overline{G}(b_1) = 1110101 \\
G(a_2) = 1100110 & \overline{G}(b_2) = 0011101 \\
G(a_3) = 0100111 & \overline{G}(b_3) = 1010000
\end{array}
\]

For the remainder of the talk: \( a \) and \( b \) are fixed.
Recursive construction (like Savitch’s Theorem)

Assume $u$ and $v$ are at distance $2^k$ from each other.

Reachability gadgets: We recursively define strings $RG_u \rightarrow v^k(a)$ and $RG_u \rightarrow v^k(b)$, such that:

$$\text{LCS}(RG_u \rightarrow v^k(a), RG_u \rightarrow v^k(b)) = X^k$$ if $u \xrightarrow{} v$

$$\text{LCS}(RG_u \rightarrow v^k(a), RG_u \rightarrow v^k(b)) \leq X^k - 1$$ otherwise

$G(a) = RG_s \rightarrow t \log T(a)$ and $G(b) = RG_s \rightarrow t \log T(b)$. 

Abboud, Hansen, Vassilevska Williams, and Williams

Simulating Branching Programs with Edit Distance
Recursive construction (like Savitch’s Theorem)

Assume $u$ and $v$ are at distance $2^k$ from each other.

Reachability gadgets: We recursively define strings $RG_{k}^{u \rightarrow v}(a)$ and $RG_{k}^{u \rightarrow v}(b)$, such that:

$$LCS(RG_{k}^{u \rightarrow v}(a), \overline{RG}_{k}^{u \rightarrow v}(b)) = X_k \text{ if } u \sim v$$

$$LCS(RG_{k}^{u \rightarrow v}(a), \overline{RG}_{k}^{u \rightarrow v}(b)) \leq X_k - 1 \text{ otherwise}$$
Recursive construction (like Savitch’s Theorem)

Assume $u$ and $v$ are at distance $2^k$ from each other.

Reachability gadgets: We recursively define strings $RG_k^{u \rightarrow v}(a)$ and $RG_k^{u \rightarrow v}(b)$, such that:

$$LCS(RG_k^{u \rightarrow v}(a), RG_k^{u \rightarrow v}(b)) = X_k$$

if $u \leadsto v$

$$LCS(RG_k^{u \rightarrow v}(a), RG_k^{u \rightarrow v}(b)) \leq X_k - 1$$

otherwise

$G(a) = RG_{s \rightarrow t}^T(a)$ and $\overline{G}(b) = RG_{log T}(b)$.
The base case, $k = 0$

- For $k = 0$, $u$ and $v$ are in neighboring layers.
- $u \leadsto v$ iff there is an edge from $u$ to $v$.
- This depends on the variable $x_i$ associated with $u$.
- $x_i$ is determined by $a$ if $i \leq n/2$, and by $b$ otherwise.
The base case, \( k = 0 \)

- For \( k = 0 \), \( u \) and \( v \) are in neighboring layers.
- \( u \leftrightarrow v \) iff there is an edge from \( u \) to \( v \).
- This depends on the variable \( x_i \) associated with \( u \).
- \( x_i \) is determined by \( a \) if \( i \leq n/2 \), and by \( b \) otherwise.

\[
\begin{align*}
R_{G_0}^{u \rightarrow v}(a) &= \begin{cases} 
1 & \text{if } (u, v) \text{ is present or } i > n/2 \\
\emptyset_1 & \text{otherwise}
\end{cases} \\
R_{G_0}^{u \rightarrow v}(b) &= \begin{cases} 
1 & \text{if } (u, v) \text{ is present or } i \leq n/2 \\
\emptyset_2 & \text{otherwise}
\end{cases} \\
LCS(R_{G_0}^{u \rightarrow v}(a), R_{G_0}^{u \rightarrow v}(b)) &= \begin{cases} 
1 & \text{if } (u, v) \text{ is present} \\
0 & \text{otherwise}
\end{cases}
\end{align*}
\]
The base case, $k = 0$

- For $k = 0$, $u$ and $v$ are in neighboring layers.
- $u \sim v$ iff there is an edge from $u$ to $v$.
- This depends on the variable $x_i$ associated with $u$.
- $x_i$ is determined by $a$ if $i \leq n/2$, and by $b$ otherwise.

$$RG_0^{u \rightarrow v}(a) = \begin{cases} 1 & \text{if } (u, v) \text{ is present or } i > n/2 \\ \emptyset_1 & \text{otherwise} \end{cases}$$

$$RG_0^{u \rightarrow v}(b) = \begin{cases} 1 & \text{if } (u, v) \text{ is present or } i \leq n/2 \\ \emptyset_2 & \text{otherwise} \end{cases}$$

$$LCS(RG_0^{u \rightarrow v}(a), RG_0^{u \rightarrow v}(b)) = \begin{cases} 1 & \text{if } (u, v) \text{ is present} \\ 0 & \text{otherwise} \end{cases}$$

- $X_0 = 1$. 
$k > 0$

$u \sim v$ iff \begin{align*}
&u \sim W_1 \quad \text{AND} \quad W_1 \sim v \\
&u \sim W_2 \quad \text{AND} \quad W_2 \sim v \\
&u \sim W_3 \quad \text{AND} \quad W_3 \sim v
\end{align*}
**AND and OR gadgets**

- **Given:** Strings $a_1, \ldots, a_k, b_1, \ldots, b_k$.
- Let $z_i = LCS(a_i, b_i)$, and assume $z_i \leq Y$.
- Construct two strings $A = \text{AND}(a_1, \ldots, a_k)$ and $B = \text{AND}(b_1, \ldots, b_k)$ such that:
  
  \[
  \begin{align*}
  LCS(A, B) & = Y' \quad \text{if } z_1 = \cdots = z_k = Y \\
  LCS(A, B) & < Y' \quad \text{otherwise}
  \end{align*}
  \]
AND and OR gadgets

- **Given:** Strings $a_1, \ldots, a_k, b_1, \ldots, b_k$.
- Let $z_i = LCS(a_i, b_i)$, and assume $z_i \leq Y$.
- Construct two strings $A = \overline{AND}(a_1, \ldots, a_k)$ and $B = \overline{AND}(b_1, \ldots, b_k)$ such that:

  $$\begin{align*}
  LCS(A, B) &= Y' \quad \text{if } z_1 = \cdots = z_k = Y \\
  LCS(A, B) &< Y' \quad \text{otherwise}
  \end{align*}$$

- Construct two strings $A' = \overline{OR}(a_1, \ldots, a_k)$ and $B' = \overline{OR}(b_1, \ldots, b_k)$ such that:

  $$\begin{align*}
  LCS(A', B') &= Y'' \quad \text{if } \exists i : z_i = Y \\
  LCS(A', B') &< Y'' \quad \text{otherwise}
  \end{align*}$$
Combining AND and OR

\[ RG^u \rightarrow v_k(a) = \text{OR}(\text{AND}(RG^u \rightarrow w_{1k-1}(a), RG^{w_{1k-1}} \rightarrow v_k(a)), \ldots, \text{AND}(RG^u \rightarrow w_{W_{k-1}}(a), RG^{w_{W_{k-1}}} \rightarrow v_k(a))) \]

\[ RG^u \rightarrow v_k(b) = \overline{\text{OR}}(\overline{\text{AND}}(RG^u \rightarrow w_{1k-1}(b), RG^{w_{1k-1}} \rightarrow v_k(b)), \ldots, \overline{\text{AND}}(RG^u \rightarrow w_{W_{k-1}}(b), RG^{w_{W_{k-1}}} \rightarrow v_k(b))) \]
Combining AND and OR

\[ RG_k^{u \rightarrow v}(a) = \text{OR}(\text{AND}(RG_{k-1}^{u \rightarrow w_1}(a), RG_{k-1}^{w_1 \rightarrow v}(a)), \ldots, \text{AND}(RG_{k-1}^{u \rightarrow w_W}(a), RG_{k-1}^{w_W \rightarrow v}(a))) \]

\[ \overline{RG_k^{u \rightarrow v}}(b) = \overline{\text{OR}(\text{AND}(\overline{RG}_{k-1}^{u \rightarrow w_1}(b), \overline{RG}_{k-1}^{w_1 \rightarrow v}(b)), \ldots, \text{AND}(\overline{RG}_{k-1}^{u \rightarrow w_W}(b), \overline{RG}_{k-1}^{w_W \rightarrow v}(b)))} \]

- The weight increases by less than a factor $16W^2$ at each level.
- Total weight: $(16W^2)^{\log T} = T^{O(\log W)}$.
Implementing AND

- **Given:** Strings $a_1, a_2, b_1, b_2$ with weight $w$.
- Let $z_i = LCS(a_i, b_i)$, and assume $z_i \leq Y$.
- Construct two strings $A = AND(a_1, a_2)$ and $B = \overline{AND}(b_1, b_2)$ such that:

  $$LCS(A, B) = Y'$$  
  if $z_1 = z_2 = Y$

  $$LCS(A, B) < Y'$$  
  otherwise
Implementing AND

- **Given:** Strings \( a_1, a_2, b_1, b_2 \) with weight \( w \).
- Let \( z_i = \text{LCS}(a_i, b_i) \), and assume \( z_i \leq Y \).
- Construct two strings \( A = \text{AND}(a_1, a_2) \) and \( B = \overline{\text{AND}}(b_1, b_2) \) such that:

\[
\text{LCS}(A, B) = Y' \quad \text{if } z_1 = z_2 = Y
\]
\[
\text{LCS}(A, B) < Y' \quad \text{otherwise}
\]

- \( w(\#) = 2w \)
- \( Y' = 2Y + 2w \)
- \( w(A) = w(B) = 4w \)
Implementing OR

- **Given:** Strings $a_1, \ldots, a_k, b_1, \ldots, b_k$ with weight $w$.
- Let $z_i = LCS(a_i, b_i)$, and assume $z_i \leq Y$.
- Construct two strings $A = OR(a_1, \ldots, a_k)$ and $B = OR(b_1, \ldots, b_k)$ such that:
  
  $$LCS(A, B) = Y' \quad \text{if } \exists i : z_i = Y$$
  $$LCS(A, B) < Y' \quad \text{otherwise}$$
Implementing OR

- **Given:** Strings $a_1, \ldots, a_k, b_1, \ldots, b_k$ with weight $w$.
- Let $z_i = LCS(a_i, b_i)$, and assume $z_i \leq Y$.
- Construct two strings $A = OR(a_1, \ldots, a_k)$ and $B = OR(b_1, \ldots, b_k)$ such that:
  \[
  LCS(A, B) = Y' \quad \text{if } \exists i : z_i = Y \\
  LCS(A, B) < Y' \quad \text{otherwise}
  \]

  

\[
\begin{align*}
  w(\#) &= w($) = 2w \\
  Y' &= Y + 4(k - 1)w \\
  w(A), w(B) &\leq 4k^2w
\end{align*}
\]
Implementing OR

**Given:** Strings $a_1, \ldots, a_k, b_1, \ldots, b_k$ with weight $w$.

Let $z_i = LCS(a_i, b_i)$, and assume $z_i \leq Y$.

Construct two strings $A = OR(a_1, \ldots, a_k)$ and $B = OR(b_1, \ldots, b_k)$ such that:

\[ LCS(A, B) = Y' \quad \text{if } \exists i : z_i = Y \]
\[ LCS(A, B) < Y' \quad \text{otherwise} \]

- $w(\#) = w(\$) = 2w$
- $Y' = Y + 4(k - 1)w + 2w$
- $w(A), w(B) \leq 4k^2w$

Also, append $@i$ with weight 2w to each $a_i$ and $b_i$. 
From LCS-Pair to LCS

Definition (LCS-Pair)

Given two sets of strings \( A = \{ a_1, \ldots, a_N \} \) and \( B = \{ b_1, \ldots, b_N \} \), and a threshold \( X \). Is there a pair \((a_i, b_j)\) with \( \text{LCS}(a_i, b_j) \geq X \)?

Abboud, Backurs, and Vassilevska Williams [2015] gave a reduction from LCS-Pair to weighted LCS, and then to LCS.
From LCS-Pair to LCS

Definition (LCS-Pair)

Given two sets of strings \( A = \{a_1, \ldots, a_N\} \) and \( B = \{b_1, \ldots, b_N\} \), and a threshold \( X \). Is there a pair \((a_i, b_j)\) with \( \text{LCS}(a_i, b_j) \geq X \)?

Abboud, Backurs, and Vassilevska Williams [2015] gave a reduction from LCS-Pair to **weighted LCS**, and then to LCS.

\[
\begin{align*}
  &a_1 &a_2 &a_3 &a_1 &a_2 &a_3 \\
b_1 &b_2 &b_3 
\end{align*}
\]
**From LCS-Pair to LCS**

**Definition (LCS-Pair)**

Given two sets of strings $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$, and a threshold $X$. Is there a pair $(a_i, b_j)$ with $LCS(a_i, b_j) \geq X$?

Abboud, Backurs, and Vassilevska Williams [2015] gave a reduction from LCS-Pair to **weighted LCS**, and then to LCS.
From LCS-Pair to LCS

Definition (LCS-Pair)

Given two sets of strings $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$, and a threshold $X$. Is there a pair $(a_i, b_j)$ with $\text{LCS}(a_i, b_j) \geq X$?

Abboud, Backurs, and Vassilevska Williams [2015] gave a reduction from LCS-Pair to weighted LCS, and then to LCS.

Abboud, Hansen, Vassilevska Williams, and Williams Simulating Branching Programs with Edit Distance 38/39
From LCS-Pair to LCS

Definition (LCS-Pair)
Given two sets of strings $A = \{a_1, \ldots, a_N\}$ and $B = \{b_1, \ldots, b_N\}$, and a threshold $X$. Is there a pair $(a_i, b_j)$ with $LCS(a_i, b_j) \geq X$?

- Abboud, Backurs, and Vassilevska Williams [2015] gave a reduction from LCS-Pair to **weighted LCS**, and then to LCS.

```
<table>
<thead>
<tr>
<th>a_1 #</th>
<th>$</th>
<th>a_2 #</th>
<th>$</th>
<th>a_3 #</th>
<th>$</th>
<th>a_1 #</th>
<th>$</th>
<th>a_2 #</th>
<th>$</th>
<th>a_3 #</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ $ $</td>
<td>#</td>
<td>b_1</td>
<td></td>
<td>$ $</td>
<td>#</td>
<td>b_2</td>
<td></td>
<td>$</td>
<td>#</td>
<td>b_3</td>
</tr>
</tbody>
</table>
```

- $w(||) \gg w(\$) \gg w(#) = X - 1$. 

Abboud, Hansen, Vassilevska Williams, and Williams Simulating Branching Programs with Edit Distance 38/39
Open problems

- Strengthen other SETH-based lower bounds.
- Is SETH false?
- How do SETH, 3SUM, and APSP relate to each other?
  - Is BP-SETH harder than all three?
  - Can APSP and 3SUM be formulated as satisfying pair problems?
  - A good place to start: The $n^{3/2^{\Omega(\sqrt{\log n})}}$ APSP algorithm by Williams [2014] relies on connections to ACC.
Open problems

- Strengthen other SETH-based lower bounds.
- Is SETH false?
- How do SETH, 3SUM, and APSP relate to each other?
  - Is BP-SETH harder than all three?
  - Can APSP and 3SUM be formulated as satisfying pair problems?
  - A good place to start: The $n^{3/2}/2^{\Omega(\sqrt{\log n})}$ APSP algorithm by Williams [2014] relies on connections to ACC.

Thank you for listening!