# Tutorial on Fine-Grained Complexity

## DIMACS, July 15-19, 2024


Lecturers:
Amir Abboud
Nick Fischer

Organizers:
Karthik C.S.
+ Amir & Nick

Teaching Assistants:
Shyan Akmal
Mursalin Habib
Ron Safier
Nathan Wallheimer



European Research Council
Established by the European Commission

מכון ויצמן למדע
WEIZMANN INSTITUTE OF SCIENCE

DIMACS
Center for Discrete Mathematics and Theoretical Computer Science
Founded as a National Science Foundation Science and Technology Center

NSF

INSAIT
Institute for Computer Science, Artificial Intelligence and Technology

# Lecture 0: Welcome and Overview

- Welcome & Logistics
- Intro and quick overview
- Remarks about the tutorial

# Complexity Theory

| NP-hard | in P |
|---|---|
| k-SAT | Linear Programming |
| Travelling Salesman | All Pairs Shortest Paths |
| Independent Set | Fourier Transform |
| Subset Sum | Edit-Distance |
| Protein Folding | Context-Free Grammar Parsing |
| … | … |

*What about the problems inside P?*

# Complexity Theory

Popular since 1970's
(Traditional) Complexity:

"Polynomial Time = Efficient"

*Polynomial* vs. *exponential?*

*A theory for Small Data*

Popular since 2010's
Fine-Grained Complexity:

"Near-linear Time = Efficient"

$O(n), O(n^{1.5}), O(n^2), \ldots ?$

*A theory for Big Data*

## Longest Common Subsequence (LCS)

Input: two sequences of length n

S = cddcabbabcbaa

T = adbdbbcabacdd

Output: the length of the longest common subsequence

Dynamic Programming: $O(n^2)$

$$M[i,j] = \max \begin{cases} M[i-1,j-1] + (S[i] == T[j]), \\ M[i-1,j], \\ M[i,j-1] \end{cases}$$

[Masek - Paterson ’80]
$O(n^2 / \log^2 n)$

Can we do better?

## *Why care about $n$ vs. $n^2$ vs. $n^3$...?*

*Here's one example where it matters...*

### Local Alignment

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

AGCCCGTCTACGTGCAACCGGGGAAAGTATA
AAACGTGACGAGAGAGAGAACCCATTACGAA

Output: The optimal alignment of two substrings.

C C G – T C T A C G
C C C A T – T A C G
+1 +1 -0.5 -1 +1 -1 +1 +1 +1 +1 = +4.5

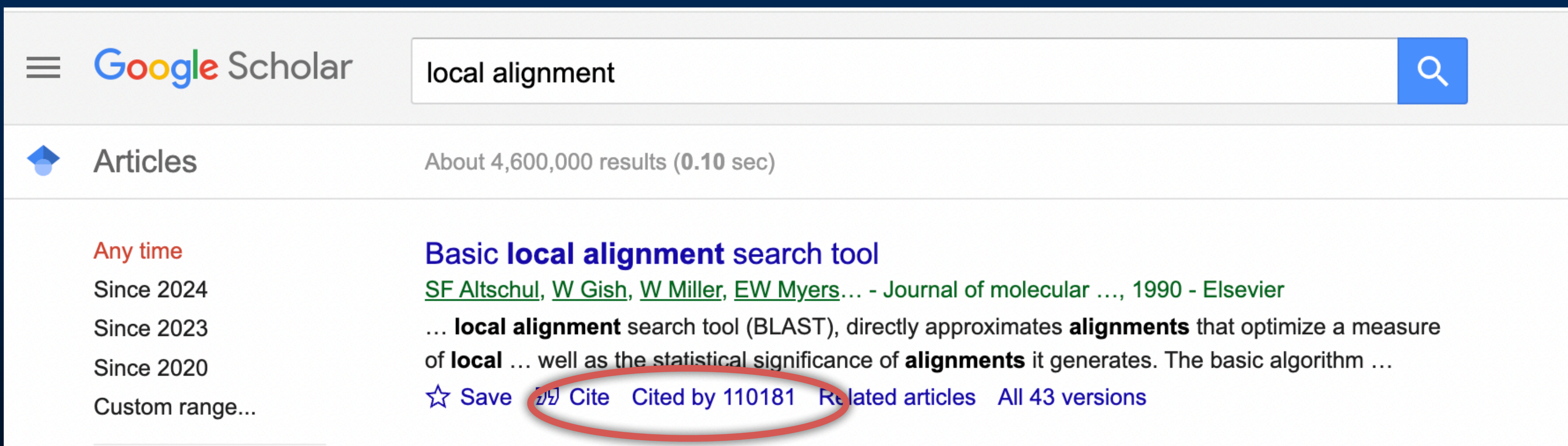
| | A | C | G | T | – |
|---|---|---|---|---|---|
| A | +1 | -1.4 | -1.8 | -0.7 | -1 |
| C | -1.4 | +1 | -0.5 | -1 | -1 |
| G | -1.8 | -0.5 | +1 | -1.9 | -1 |
| T | -0.7 | -1 | -1.9 | +1 | -1 |
| - | -1 | -1 | -1 | -1 | $-\infty$ |

Typically: ***$n \gg 10^6$***

[Smith-Waterman '81] $O(n^2)$ with dynamic programming - too slow!

# *Why care about $n$ vs. $n^2$ vs. $n^3$…?*

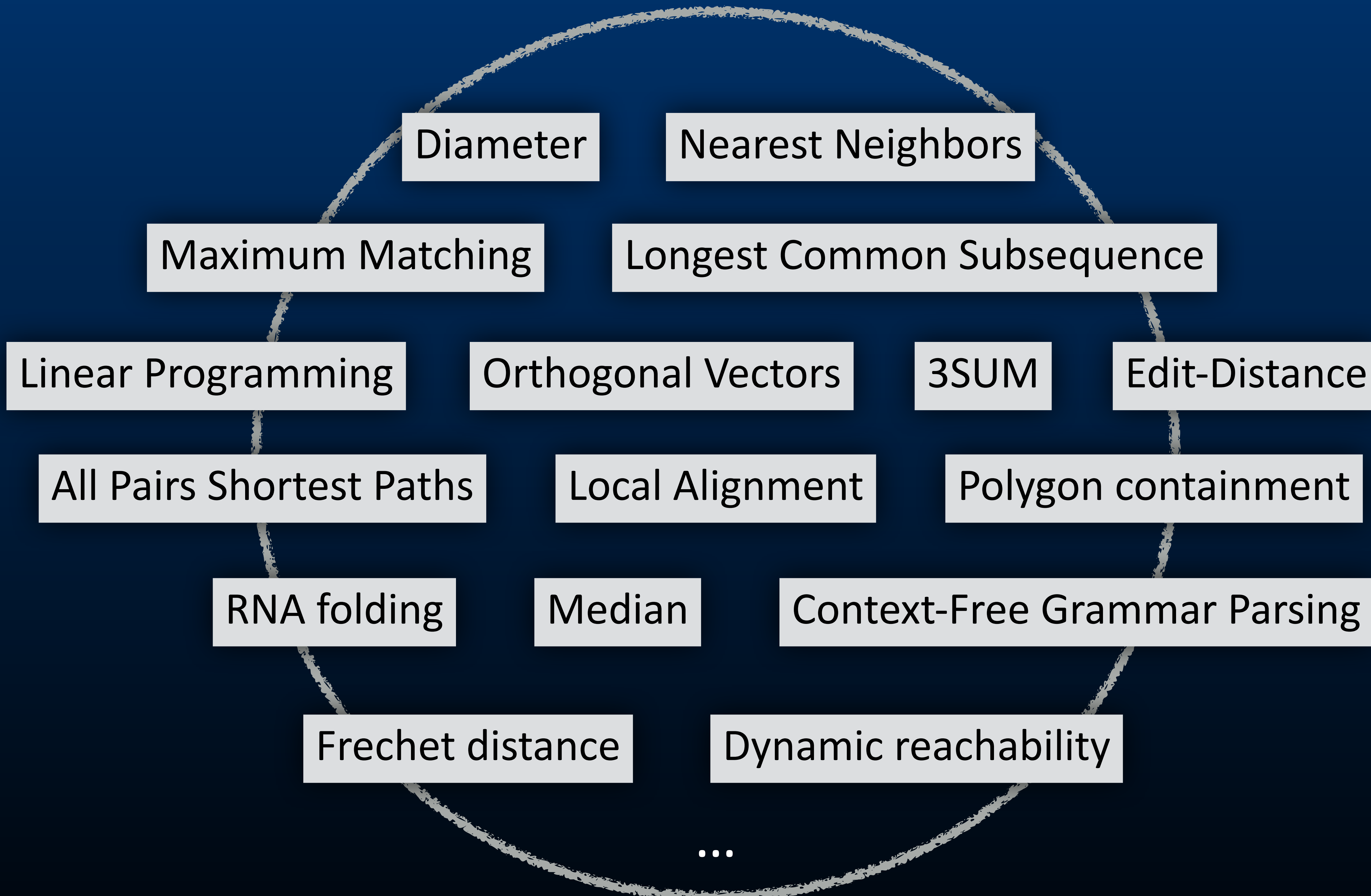
**BLAST**: A *heuristic, linear time* algorithm for **Local Alignment**.




110k citations!

Are there fast algorithms with optimality guarantees?

# The Class P




*Goal:* *Understand the time complexity of important problems.*

# Fine-Grained Complexity or: Hardness in P

Take a problem X in P, say in ***$O(n^2)$*** time.

And prove that:

“ X *probably* *cannot* be solved in $O(n^{2-\varepsilon})$ time. ”

*But how?*

# How do we get $n^2$ and $n^3$ lower bounds?

## Unconditional polynomial lower bounds?

*"Any Turing Machine has to spend $\Omega(n^2)$ time…"*

<u>Time Hierarchy Thm (1965):</u> Some (artificial) problems require $\Omega(n^2)$ time.

But $\Omega(n^2)$ for natural problems, even for SAT, is far out of reach of current techniques. Best lower bound is $3.1n$.

## Lower bounds for restricted algorithms?

e.g. $\Omega(n \log n)$ for sorting in the comparisons-only model.

Not general enough, and only gives partial answers.

NP-hardness is not fine-grained enough…

# Complexity Theory

*How do we prove hardness results?*

Popular since 1970's
(Traditional) Complexity:

*Polynomial* vs. *exponential?*

Reductions!

**P ≠ NP:**
"k-SAT cannot be solved in polynomial time."

Popular since 2010's
Fine-Grained Complexity:

$O(n), O(n^{1.5}), O(n^2), \ldots ?$

# Complexity Theory

*How do we prove hardness results?*

Popular since 1970's
(Traditional) Complexity:

*Polynomial* vs. *exponential?*

Reductions!

My problem is in P
$\downarrow$
$P = NP$

Popular since 2010's
Fine-Grained Complexity:

$O(n), O(n^{1.5}), O(n^2), \ldots ?$

Fine-Grained Reductions!

# Complexity Theory

*How do we prove hardness results?*

Popular since 1970's
(Traditional) Complexity:

*Polynomial* vs. *exponential?*

Reductions!

**P ≠ NP:**
"k-SAT cannot be solved in polynomial time."

Popular since 2010's
Fine-Grained Complexity:

$O(n), O(n^{1.5}), O(n^2), \ldots ?$

Fine-Grained Reductions!

**SETH:**
"k-SAT cannot be solved even in $O(1.99^n)$ time."

# Complexity Theory

*How do we prove hardness results?*

| Popular since 1970's<br>(Traditional) Complexity: | Popular since 2010's<br>Fine-Grained Complexity: |
|---|---|
| *Polynomial* vs. *exponential?* | $O(n), O(n^{1.5}), O(n^2), \ldots ?$ |
| Reductions! | Fine-Grained Reductions! |
| My problem is in P<br>↓<br>$P = NP$ | My problem is linear<br>↓<br>SETH is false |

# An Example of a Fine-Grained Lower Bound

**Theorem** [AVW'14]:

"If for some $\varepsilon > 0$, we can solve Local Alignment in $O(n^{2-\varepsilon})$ time, then we can solve k-SAT in $O((2-\delta)^n)$ time for some $\delta > 0$ and all $k > 0$."

Faster Local Alignment
e.g. $O(n^{1.99})$
$\longrightarrow$
Faster k-SAT
e.g. $O(1.99^n)$
$\longrightarrow$
SETH is false

$P \neq NP$: "k-SAT cannot be solved in polynomial time."

**ETH:** "k-SAT cannot be solved even in $2^{o(n)}$ time."

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

"k-SAT cannot be solved even in $O(1.99^n)$ time."

# SETH

**k-SAT**: given a k-CNF formula on $n$ variables and $m$ clauses, is it satisfiable?

$$\phi = (x_1 \vee x_2 \vee \bar{x}_3 \vee x_{10}) \wedge \cdots \wedge (x_2 \vee \bar{x}_1 \vee x_4)$$

Fastest algorithms:

$$O\left(2^{\left(1-\frac{1}{ck}\right)\cdot n}\right)$$

k=3: $1.308^n$

k=4: $1.504^n$

k=5: $1.592^n$

... $k \to \infty : 2^n$

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

[Impagliazzo-Paturi'01]

There is no $\varepsilon > 0$ such that for all $k > 2$,

**k-SAT** can be solved in $O((2-\varepsilon)^n)$ time.

SETH: "k-SAT cannot be solved in ***O(1.99^n)*** time."

# The Class P (before)

Diameter

Nearest Neighbors

Maximum Matching

Longest Common Subsequence

Linear Programming

Orthogonal Vectors

3SUM

Edit-Distance

All Pairs Shortest Paths

Local Alignment

Polygon containment

RNA folding

Median

Context-Free Grammar Parsing

Frechet distance

Dynamic reachability

…

# The Class P (after)

**SETH**

Diameter
Closest Pair
Local Alignment
Dynamic Reachability
Single-Source Max-Flow
Subtree Isomorphism
Stable Matching
Edit-Distance
Frechet
LCS
…

*Problem domains:*
Graph Algorithms
Pattern Matching
Bioinformatics
Computational Geometry
Data Structures
Machine Learning
Formal Languages
…

Many problems remain unclassified…

# The Class P (after)

**SETH**

SAT on $n$ variable formulas requires $\Omega((2-\varepsilon)^n)$ time.

Closest Pair
Local Alignment
Dynamic Reachability
Single-Source Max-Flow
Subtree Isomorphism
Stable Matching
Edit-Distance
Frechet
LCS
…

**3SUM**

Finding 3 that sum to zero among $n$ numbers requires $\Omega(n^{2-\varepsilon})$ time.

Polygon Containment
Strips Cover Rectangle
Triangle Enumeration
Compressed Inner Product
Dynamic Max Matching
Set Intersection
…

**APSP**

Computing all distances in $n$ node graphs requires $\Omega(n^{3-\varepsilon})$ time.

Dynamic Max Matching
Stochastic Context-Free Grammar Parsing
Negative Triangle
Dynamic Max Flow
Replacement Paths
Median
…

Many problems remain unclassified…

# 3SUM

**3SUM:** Given *n* integers, are there 3 that sum to 0?

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| -15 | -6 | 33 | 8 | 1 | -21 | 4 | -30 | 7 | ... | 107 |

*A famous conjecture in computational geometry:*

<u>The 3-SUM Conjecture:</u>

"3-SUM cannot be solved in $O(n^{1.99})$ time."

# The Class P

**SETH**

↓

- Diameter
- Closest Pair
- Local Alignment
- Dynamic Reachability
- Single-Source Max-Flow
- Subtree Isomorphism
- Stable Matching
- Edit-Distance
- Frechet
- LCS
- …

**3SUM**

↓

- Colinearity
- Polygon Containment
- Strips Cover Rectangle
- Triangle Enumeration
- Compressed Inner Product
- Dynamic Max Matching
- Set Intersection
- …

# All Pairs Shortest Paths

APSP: Given a weighted graph on n nodes and $n^2$ edges, compute the distance between every pair of nodes.

| Author | Runtime | Year |
| --- | --- | --- |
| *Fredman* | $n^3 \log \log^{1/3} n / \log^{1/3}$ | 1976 |
| *Takaoka* | $n^3 \log \log^{1/2} n / \log^{1/2}$ | 1992 |
| *Dobosiewicz* | $n^3 / \log^{1/2} n$ | 1992 |
| *Han* | $n^3 \log \log^{5/7} n / \log^{5/7}$ | 2004 |
| *Takaoka* | $n^3 \log \log^2 n / \log n$ | 2004 |
| *Zwick* | $n^3 \log \log^{1/2} n / \log n$ | 2004 |
| *Chan* | $n^3 / \log n$ | 2005 |
| *Han* | $n^3 \log \log^{5/4} n / \log^{5/4}$ | 2006 |
| *Chan* | $n^3 \log \log^3 n / \log^2 n$ | 2007 |
| *Han, Takaoka* | $n^3 \log \log n / \log^2 n$ | 2012 |
| *Williams* | $n^3 / 2^{\Omega(\sqrt{\log n})}$ | 2014 |

Classical Algs: $O(n^3)$

*Bellman-Ford, Dijkstra,…*

Conjecture:
APSP cannot be solved in $O(n^{3-\epsilon})$ time.

# The Class P (before)

Diameter

Nearest Neighbors

Maximum Matching

Longest Common Subsequence

Linear Programming

Orthogonal Vectors

3SUM

Edit-Distance

All Pairs Shortest Paths

Local Alignment

Polygon containment

RNA folding

Median

Context-Free Grammar Parsing

Frechet distance

Dynamic reachability

...

# The Class P (after)

**SETH**

- Diameter
- Closest Pair
- Local Alignment
- Dynamic Reachability
- Single-Source Max-Flow
- Subtree Isomorphism
- Stable Matching
- Edit-Distance
- Frechet
- LCS
- …

**3SUM**

- Colinearity
- Polygon Containment
- Strips Cover Rectangle
- Triangle Enumeration
- Compressed Inner Product
- Dynamic Max Matching
- Set Intersection
- …

**APSP**

- Radius
- Dynamic Max Matching
- Stochastic Context-Free Grammar Parsing
- Negative Triangle
- Dynamic Max Flow
- Replacement Paths
- Median
- …

Many problems remain unclassified…

# Technical Remarks

- We will ignore $\log n,\ \log^{O(1)} n,\ 2^{\sqrt{\log n}}$ or any $n^{o(1)}$ factors.
  - Many reductions have such overheads.
- We allow randomness.
  - The conjectures are assumed to holds against randomized algorithms too.
  - Many reductions use randomness.
- We use the (standard) Word RAM model with $w = O(\log n)$.
  - You can do any operations on words in constant time: addition, multiplication, random access, hashing, etc.
  - Since we allow log factors and randomness, this is not too important.
- Numbers are assumed to be in a polynomial range.
  - Integers in $\{-n^{O(1)}, \ldots, +n^{O(1)}\}$, real numbers with precision $1/n^{O(1)}$.

# Fine-Grained Reductions

Thm: If $A \;{}_a\!\!\rightarrow_b\; B$ and B is in $O(n^{b-\varepsilon})$ time then A is in $O(n^{a-\delta})$ time.

Thm: If $A \;{}_a\!\!\rightarrow_b\; B$ and $B \;{}_b\!\!\rightarrow_c\; C$ then also $A \;{}_a\!\!\rightarrow_c\; C$.

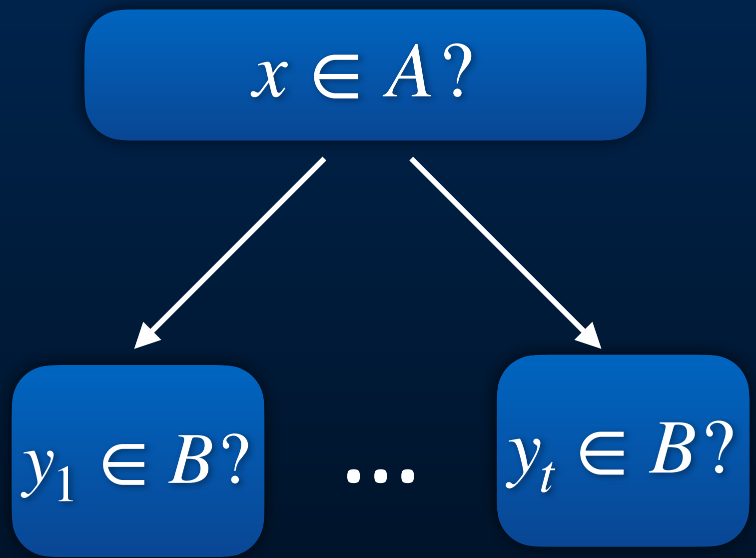
Definition: $A \;{}_a\!\!\rightarrow_b\; B$

A fine-grained $(a, b)$-reduction from A to B is an algorithm $\mathscr{A}^B$ for A with oracle access to B, such that:

$\forall \varepsilon > 0 : \;\; \exists \delta > 0 :$ for all input x of size n:

1. $\mathscr{A}^B(x)$ is correct w.p. $\geq 1 - 1/n^{10}$
2. $\mathscr{A}^B(x)$ runs in $O(n^{a-\delta})$ time.
3. Let $y_1, \ldots, y_t$ be the oracle calls, then:

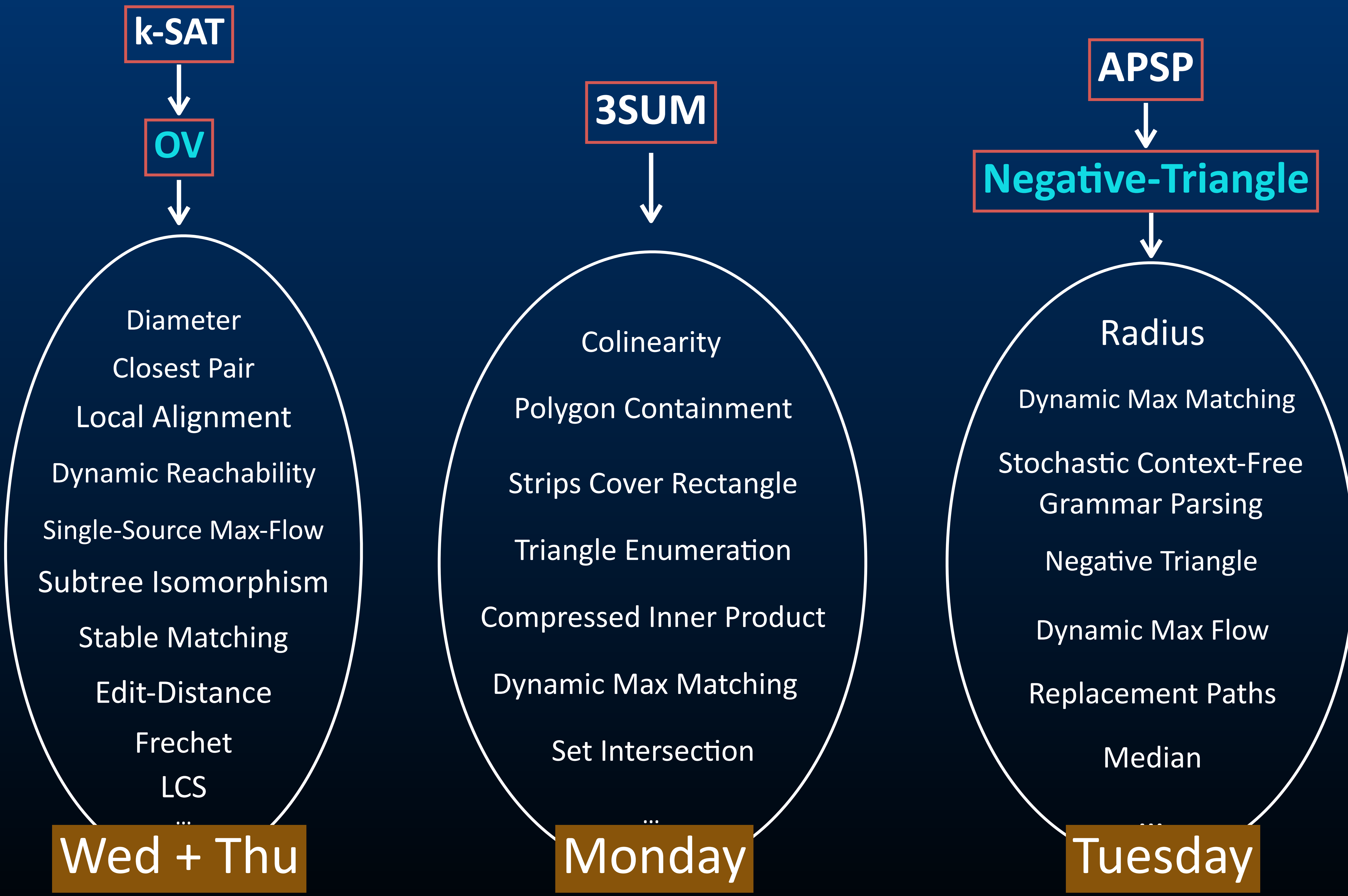
$$\sum_{i=1}^{t} |y_i|^{b-\varepsilon} = O(n^{a-\delta})$$




[Vassilevska & Williams '10]

# Tutorial Objectives

- Goal 0: The ability to understand FGC results.
- Goal 1: The ability to prove your own FGC results.
    - We will highlight the simplest hard problems


k-SAT
OV
Diameter
Closest Pair
Local Alignment
Dynamic Reachability
Single-Source Max-Flow
Subtree Isomorphism
Stable Matching
Edit-Distance
Frechet
LCS
...
Wed + Thu
3SUM
Colinearity
Polygon Containment
Strips Cover Rectangle
Triangle Enumeration
Compressed Inner Product
Dynamic Max Matching
Set Intersection
...
Monday
APSP
Negative-Triangle
Radius
Dynamic Max Matching
Stochastic Context-Free Grammar Parsing
Negative Triangle
Dynamic Max Flow
Replacement Paths
Median
...
Tuesday

# Tutorial Objectives

- Goal 0: The ability to understand FGC results.
- Goal 1: The ability to prove your own FGC results.
  - We will highlight the simplest hard problems
- Goal 2: Intimacy with the theory and with current research.
  - This is the purpose of the afternoon lectures (and Friday).
- *Most importantly: To have fun thinking about basic problems!*