6.S078
A FINE-GRAINED APPROACH TO ALGORITHMS AND COMPLEXITY

LECTURE 1

PERSONNEL

Profs. Ryan Williams and Virginia Vassilevska Williams

TA: Nicole Wein

6.S078 WORKLOAD

1. Three Problem sets: worth **50%** of grade, will come out about 2 weeks apart
   Can work with a partner; write up your own solutions.
2. Class Project: worth **50%** of grade
   Can work individually or with 1 partner
   Project proposal (1 page): due **TBA**
   Progress report (2 pages): due **TBA**
   Final presentation: during the last 1-2 weeks of class
3. Flipped Class: After this lecture, most other lectures will be informal discussions of the material. You should read/watch the provided lecture notes (and other materials) before class. We'll release the relevant material late the previous week.

No exams!

WEBSITE AND PIAZZA

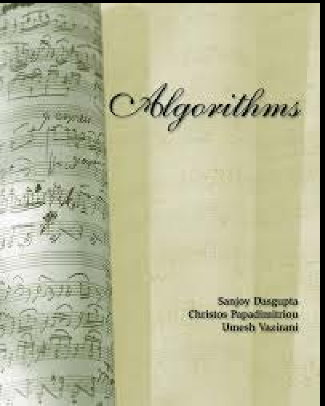
- **http://bit.ly/FGAC20**
- Sign up for Piazza: link on the website
- Announcements will be made on both piazza and the website
- Assignments will be released on piazza

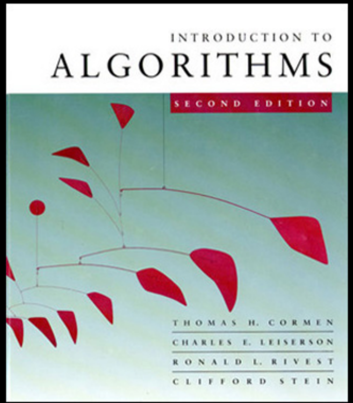
PLAN FOR THE DAY

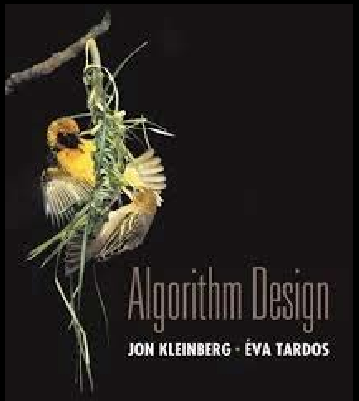
What is this class about?

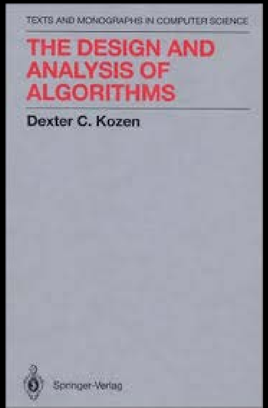
THE CENTRAL QUESTION OF ALGORITHMS RESEARCH

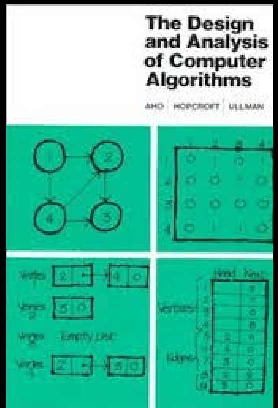
``How fast can we solve fundamental problems, in the worst case?"











etc.

HARD PROBLEMS

For many problems, the known techniques get stuck:

- Very important computational problems from diverse areas
- They have simple, often brute-force, textbook algorithms
- That are slow.
- No improvements in many decades!

Example 1

A CANONICAL HARD PROBLEM

k-SAT

Input: variables $x_1, \ldots, x_n$ and a formula

$F = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ so that each C_i is of the form

$\{y_1 \vee y_2 \vee \ldots \vee y_k\}$ and $\forall i$, y_i is either x_t or $\neg x_t$ for some t.

Output: A boolean assignment to $\{x_1,\ldots,x_n\}$ that satisfies all the clauses, or NO if the formula is not satisfiable

Brute-force algorithm: try all 2^n assignments

Best known algorithm: $O(2^{n-(cn/k)} m^d)$ time for const c,d

Goes to 2^n as k grows.

Example 2

???

ANOTHER HARD PROBLEM: LONGEST COMMON SUBSEQUENCE (LCS)

Given two strings on n letters

ATCGGGTTCCTTAAGGG

ATTGGTACCTTCAGG

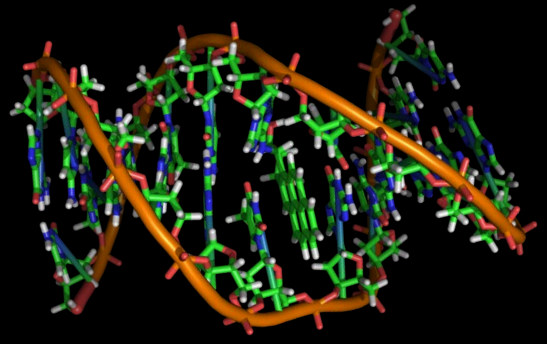
Find a subsequence of both strings of maximum length.

Algorithms:

Classical **$O(n^2)$** time

Best algorithm:
$O(n^2 / \log^2 n)$ time [MP'80]

Applications both in computational biology and in spellcheckers.



Solved daily on huge strings!

(Human genome: 3×10^9 base pairs.)

IN THEORETICAL CS, POLYNOMIAL TIME = EFFICIENT/EASY.

This is for a variety of reasons.

E.g. composing two efficient algorithms results in an efficient algorithm. Also, model-independence.

However, noone would consider an $O(n^{100})$ time algorithm efficient in practice.

If n is huge, then $O(n^2)$ is also inefficient.

WE ARE STUCK ON MANY PROBLEMS, EVEN JUST IN $O(N^2)$ TIME

No $N^{2-\varepsilon}$ time algorithms known for:

➨Many *string matching* problems:
Edit distance, Sequence local alignment, LCS, jumbled indexing …

***General form**: given two sequences of length n, how similar are they?*
All variants can be solved in $O(n^2)$ time by dynamic programming.

ATCGGGTTCCTTAAGGG
ATTGGTACCTTCAGG

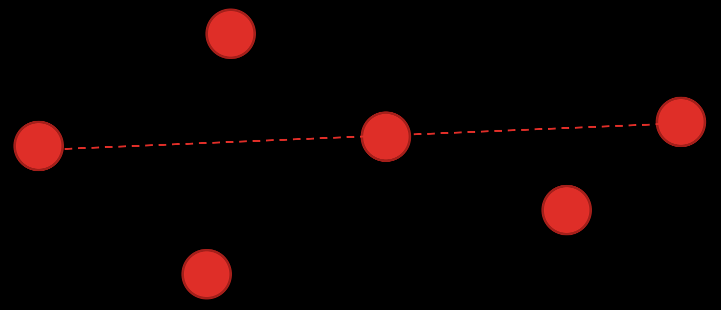
WE ARE STUCK ON MANY PROBLEMS, EVEN JUST IN $O(N^2)$ TIME

No $N^{2-\varepsilon}$ time algorithms known for:

- ➨Many *string matching* problems
- ➨Many problems in *computational geometry*: e.g

Given n points in the plane, are any three collinear?

A very important primitive!



WE ARE STUCK ON MANY PROBLEMS, EVEN JUST IN $O(N^2)$ TIME

No $N^{2-\varepsilon}$ time algorithms known for:

- Many *string matching* problems
- Many problems in *computational geometry*
- Many *graph problems* in sparse graphs: e.g.

Given an n node, O(n) edge graph, what is its diameter?

Fundamental problem. Even approximation algorithms seem hard!

WE ARE STUCK ON MANY PROBLEMS, EVEN JUST IN $O(N^2)$ TIME

No $N^{2-\varepsilon}$ time algorithms known for:

- Many *string matching* problems
- Many problems in *computational geometry*
- Many *graph problems* in sparse graphs
- Many other problems …

Why are we stuck?

Are we stuck because of the same reason?

PLAN FOR TODAY

- Traditional hardness in complexity
- A fine-grained approach
- Some simple results

TIME HIERARCHY THEOREMS IN COMPLEXITY THEORY

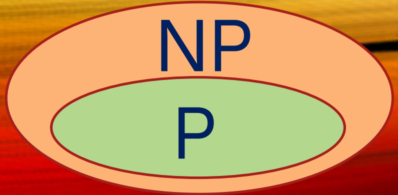
For most natural computational models one can prove:

for any constant c, there *exist* problems solvable in $O(n^c)$ time but not in $O(n^{c-\varepsilon})$ time for any $\varepsilon > 0$.

It is completely unclear how to show that a *particular* problem in $O(n^c)$ time is not in $O(n^{c-\varepsilon})$ time for any $\varepsilon > 0$.

It is not even known if SAT is in linear time!

WHY IS K-SAT HARD?



N – size of input

It also does not apply to problems in P! Unless P=NP

Theorem [Cook, Karp'72]:

k-SAT is **NP-complete** for all $k \geq 3$.

NP-completeness addresses runtime, but it is too coarse-grained!

I.e. k-SAT is considered *hard because*

***"fast" algorithms for it imply "fast" algorithms for many important problems*.**

We'll develop a *fine-grained theory of hardness* that is conditional and mimics NP-completeness.

PLAN

- Traditional hardness in complexity
- A fine-grained approach
- Some simple results

FINE-GRAINED HARDNESS *IDEA*

Idea: Mimic NP-completeness

1. Identify key hard problems

2. Reduce these to all (?) problems believed hard

3. Hopefully form *equivalence classes*

CNF SAT IS CONJECTURED TO BE REALLY HARD

We will see these in detail next lecture!

Two popular conjectures about SAT on n variables [IPZ01]:

ETH (Exponential Time Hypothesis):

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

SETH (Strong Exponential Time Hypothesis): For every $\varepsilon > 0$, there is a k such that k-SAT on n variables, m clauses cannot be solved in $2^{(1-\varepsilon)n}$ poly m time.

So we can use k-SAT as our hard problem and ETH or SETH as the hypothesis we base hardness on.

Strengthening of SETH [CGIMPS'16] suggests these are **not equivalent...**

Fix the model: word-RAM with O(log n) bit words

Given a set S of n vectors in $\{0,1\}^d$, for d = ω(log n) are there u, v 2 S with **u ¢v = 0**?

Hypothesis: Orthog. Vecs. requires $n^{2-o(1)}$ time.

[W'05]: SETH implies this hypothesis!

Easy O(n^2 d) time alg

Best known [AWY'15]: $n^{2 - \Theta(1 / \log (d/\log n))}$

We will see these a lot!

Orthogonal vectors

Next 2 weeks

Hypothesis: APSP requires $n^{3-o(1)}$ time.

More key problems to blame

Weeks 4-5

Given a set S of n integers, are there a, b, c 2 S with **a + b + c = 0**?

3SUM

APSP

All pairs shortest paths: given an n-node weighted graph, find the **distance** between every two nodes.

Weeks 6-7

Easy O(n^2) time alg

[BDP'05]: ~n^2 / $\log^2$ n time for integers

[Chan'18] : ~n^2 / $\log^2$ n time for reals

Hypothesis: 3SUM requires $n^{2-o(1)}$ time.

Classical algs: O(n^3) time

[W'14]: n^3 / exp(√ log n) time

FINE-GRAINED HARDNESS

Idea: Mimic NP-completeness

1. Identify key hard problems

2. Reduce these to all (?) other hard problems

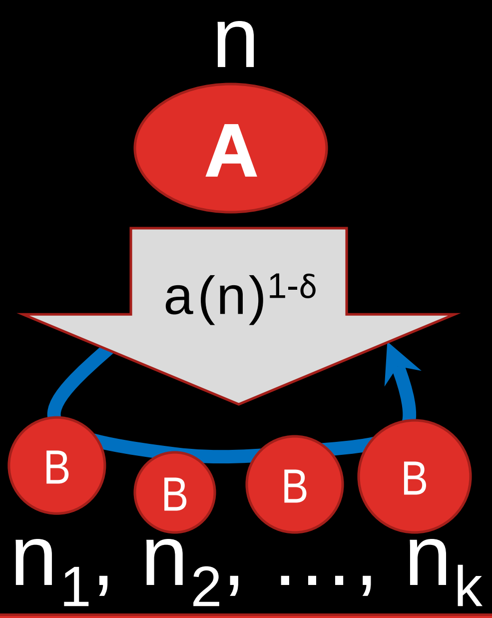
3. Hopefully form *equivalence classes*

FINE-GRAINED REDUCTIONS

Intuition: $a(n)$,$b(n)$ are the naive runtimes for A and B. A reducible to B implies that beating the naive runtime for B implies also beating the naive runtime for A.

- A is (a,b)-reducible to B if
  for every $\varepsilon>0$ $\exists$ $\delta>0$, and an $O(a(n)^{1-\delta})$ time algorithm
  that adaptively transforms any A-instance of size n to B-instances of size $n_1,\ldots,n_k$ so that $\sum_i b(n_i)^{1-\varepsilon} < a(n)^{1-\delta}$.

- **If B is in** $O(b(n)^{1-\varepsilon})$ **time,**
  then A is in $O(a(n)^{1-\delta})$ time.
- Focus on exponents.
- We can build equivalences.



Don’t worry! We will see many examples!

SOME STRUCTURE WITHIN P

Using other hardness assumptions, one can unravel even more structure

N – input size
n – number of variables or vertices

Graph diameter [RV’13,BRSVW’18], eccentricities [AVW’16] , local alignment, longest common substring* [AVW’14], Frechet distance [Br’14], Edit distance [BI’15], LCS, Dyn. time warping [ABV’15, BrK’15], subtree isomorphism [ABHVZ’15], Betweenness [AGV’15], Hamming Closest Pair [AW15], Reg. Expr. Matching [BI16,BGL17]…

Many dynamic problems [P’10],[AV’14], [HKNS‘15], [D16], [RZ’04], [AD’16],…

$N^{2-\varepsilon'}$

$N^{1.5-\varepsilon'}$

$n^{3-\varepsilon}$

$N^{2-\varepsilon}$ Orthog. vectors

[W’04]

$2^{(1-\delta)n}$ k-SAT 8 k

$N^{2-\varepsilon'}$

Huge literature in comp. geom. [GO’95, BHP98, …]: Geombase, 3PointsLine, 3LinesPoint, Polygonal Containment, Planar Motion Planning, 3D Motion Planning …

String problems: Sequence local alignment [AVW’14], jumbled indexing [ACLL’14], …

3SUM

$N^{2-\varepsilon}$

APSP

$n^{3-\varepsilon}$

$N^{1.5-\varepsilon}$

equivalent

In *dense graphs*: radius, median, betweenness centrality [AGV’15], *negative triangle*, second shortest path, replacement paths, shortest cycle [VW’10], …

PLAN

- Traditional hardness in complexity
- A fine-grained approach
- First reductions: from SETH

SETH

SETH: for every $\varepsilon > 0$, there is a k such that k-SAT on n variables, m clauses cannot be solved in $2^{(1-\varepsilon)n}$ poly m time.

If there is an $2^{(1-\varepsilon)n}$ poly m time algorithm for some $\varepsilon > 0$ that can solve SAT on CNF Formulas (for all k) on n variables and m clauses, then SETH is false.

FAST OV IMPLIES SETH IS FALSE [W'04]

F- CNF-formula on n vars, m clauses

E.g. $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_4)$

Split the vars into V_1 and V_2 on $n/2$ vars each

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

For j=1,2 consider the **partial assignments** of V_j **: there are $2^{n/2}$ of them.**

E.g. for V_1: $\{[x_1 = 0, x_2 = 0], [x_1 = 0, x_2 = 1], [x_1 = 1, x_2 = 0], [x_1 = 1, x_2 = 1]\}$

OV: Given a set S of N vectors in $\{0, 1\}^d$, are there u, v 2 S with **u ¢v = 0**?

Given F, we want to create a set of vectors S in $\{0,1\}^d$ so that there is an orthogonal pair if and only if F is satisfiable and $|S| \sim 2^{n/2}$ and $d \sim m$.

FAST OV IMPLIES SETH IS FALSE [W'04]

F- CNF-formula on n vars, m clauses

Split the vars into V_1 and V_2 on n/2 vars each

For j=1,2 and each **partial assignment** ϕ of V_j create (m+2) length vector v(j, ϕ):

E.g. $(x_1 \lor x_2) \land (\neg x_1 \lor x_3 \lor x_4) \land (\neg x_3 \lor \neg x_4)$

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

$v(1,[x_1 = 0, x_2 = 0])$ = [0, 1, 1, 0, 1]

clauses

0	1	0	1	0	...	...	...	...	1

for all v(**1**, ϕ)

0 if ϕ satisfies the clause, 1 otherwise

1	0	0	0	1	...	...	...	...	1

for all v(**2**, ϕ)

0 if ϕ satisfies the clause,
1 otherwise

FAST OV IMPLIES SETH IS FALSE

0	1	0	1	0	…	…	…	…	1

for all v(**1**, ϕ)

0 if ϕ satisfies the clause, 1 otherwise

1	0	0	0	1	…	…	…	…	1

for all v(**2**, ψ)

0 if ψ satisfies the clause,
1 otherwise

Claim: v(1, ϕ) ¢v(2, ψ) = 0 iff ϕ ⊙ ψ is a sat assignment.

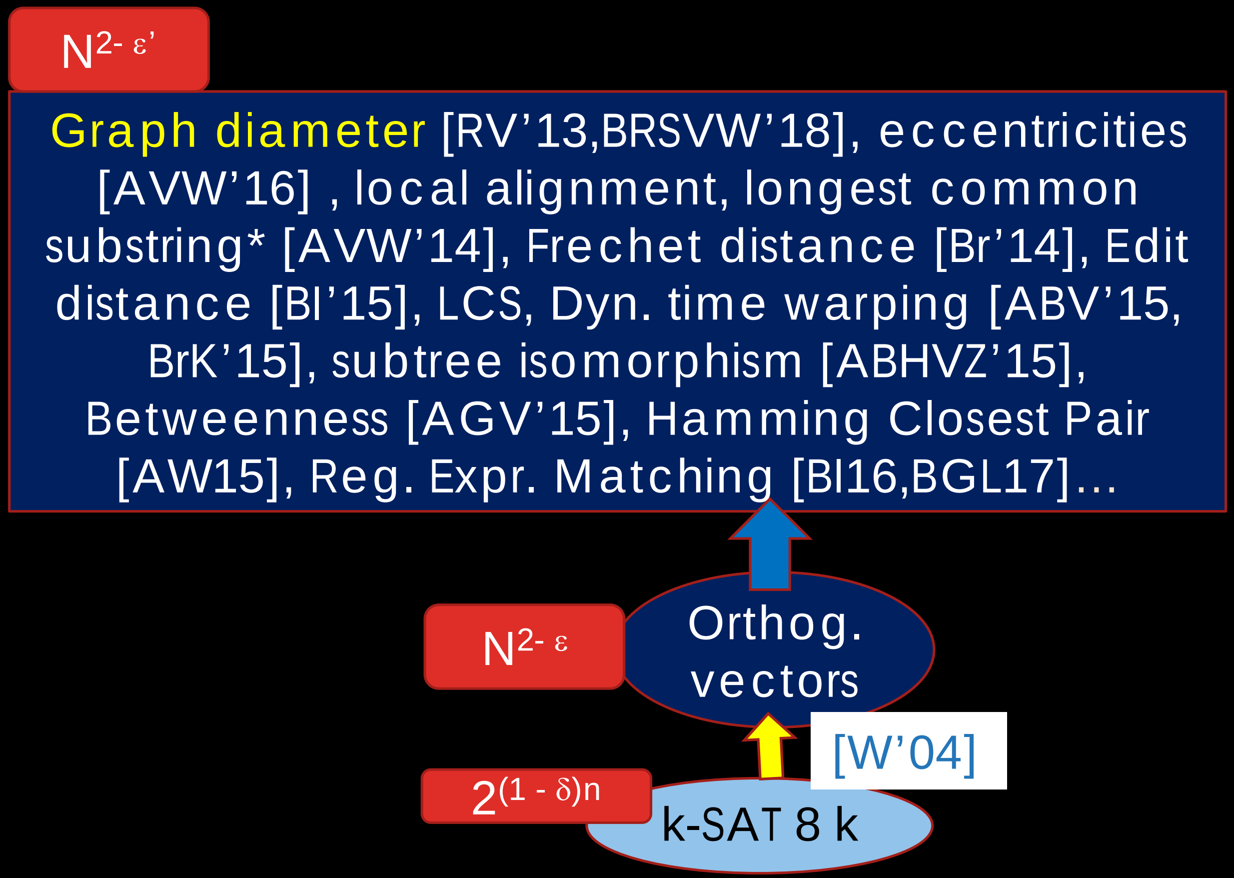
N = $2^{n/2}$ vectors of dimension $d = O(m) \rightarrow$ an OV instance.

So $N^{2-\delta}\ poly(d)$ time for OV for $\delta > 0$ implies $2^{n(1-\frac{\delta}{2})}\ poly(m)$ time for SAT and SETH is false.

Diameter:

Given $G = (V, E)$, determine $D = \max_{u,v \in V} d(u, v)$.

$\frac{3}{2}$ – **Approximate Diameter**: output D’ such that $\frac{2D}{3} \leq D' \leq D$.



Say G has m edges, n vertices.

Using BFS: $O(mn)$ time Diameter.
Best known even in sparse graphs.

RV’13: 3/2-Approximate Diameter in $\tilde{O}\left(m^{\frac{3}{2}}\right)$ time – better than mn in sparse graphs!

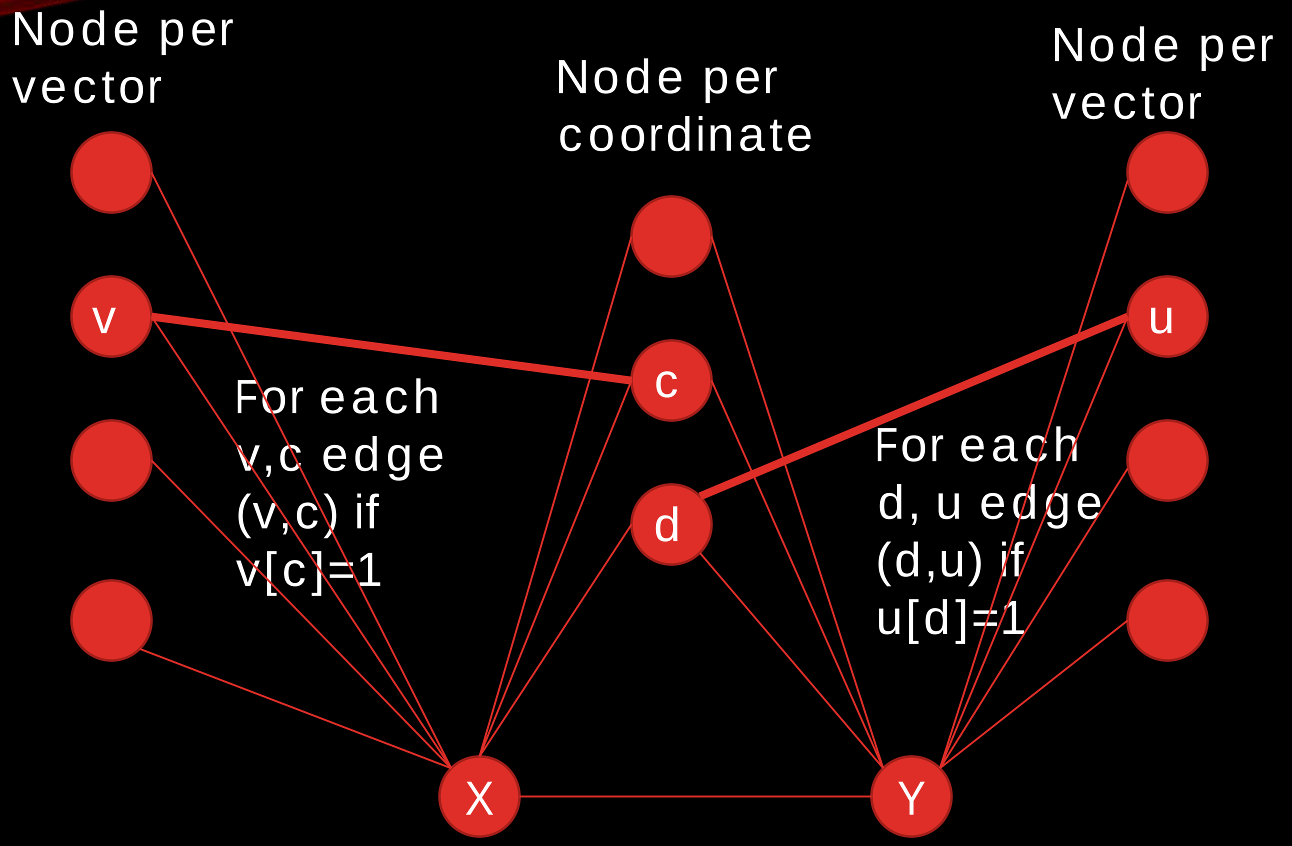
We’ll show $3/2-\epsilon$ – Diameter for $\epsilon > 0$ requires $mn^{1-o(1)}$ time under SETH.

Hard: distinguishing between Diameter 2 or 3 in sparse graphs.

Reduce from OV on n vectors; due to "Sparsification Lemma" can assume dimension is $d = O(\log n)$...

[RV'13]

DIAMETER 2 OR 3



Diameter is 3 if exists orthogonal pair, and is 2 otherwise.

Thm: Diameter 2 or 3 in $O(m^{2-\varepsilon})$ time implies $O(n^{2-\delta})$ time for OV and hence SETH is false.

Any two vector nodes from the same side are at dist 2.

Any coordinate is at dist 2 from everyone, X and Y are at dist 2 from everyone.

Two vectors u and v from different sides are at

dist 2 if exists a c with **u[c]=v[c]=1**, and at **dist 3** otherwise.

Graph has $O(n)$ nodes and since $d = O(\log n)$, $m = \tilde{O}(n)$ edges

SEE YOU NEXT TIME!

- Check Piazza and the website for the lecture notes.
- Please read them before the next class!
- PS: If there's any related topic you'd like more lecture notes to read (e.g., NP-completeness) please let us know via piazza!
(You can also email, but it's better if other students can see your questions too, so they can upvote it!)