6.842 Lecture 3

- The Lovasz Local Lemma (recap + finish)
- Polynomial Identity Testing

Lovasz Local Lemma: Recap & finish

<u>Goal</u>: Show that <u>possibly</u> no bad events happen!

possible tools:

- if independent, then obvious ← easy, no major assumptions on prob of each bad event, but huge assumption on independence
- if not independent, use union bound ← major assumption needed on p_i's.
- What if A_i's have "some" independence?

<u>def</u>. A "independent" of $B_1, B_2, \dots B_k$ if

$$\forall \; J \subseteq [k], \; J \neq \emptyset \quad \text{then} \quad \Pr\left[A \cap \bigcap_{j \in J} B_j\right] = \Pr[A] \cdot \Pr\left[\bigcap_{j \in J} B_j\right]$$

note: $[k]$ means $\{1 \dots k\}$

<u>def</u>. $A_1 \dots A_n$ events

$D = (V, E)$ with $V = [n]$ is

"dependency digraph of $A_1 \dots A_n$"

if each A_i independent of all A_j that are not neighbors in D (i.e. all A_j s.t. $(i,j) \notin E$)

Lovász Local Lemma (symmetric version)

$A_1 \dots A_n$ events s.t. $\Pr(A_i) \le p \ \forall i$
with dependency digraph D s.t. D
has max degree $\le d$.
If $ep(d+1) \le 1$ then

$$\Pr\left[\bigcap_{i=1}^{n} \bar{A}_i\right] > 0$$

Application

Thm. Given $S_1 \dots S_m \subseteq X$ $|S_i| = \ell$

each S_i intersects at most d other S_j's

previously needed $m < 2^{\ell-1}$

If $e \cdot (d+1) \le 2^{\ell-1}$

now no restriction on m but there is a restriction on "degree"

then can 2-color X such that
each S_i not monochromatic

ie. $\mathcal{H}$ is hypergraph with m edges
each containing ℓ nodes + each
intersecting $\le d$ other edges

Stronger assumptions:

(1) For today, assume ℓ, d constants

(2) Binary Entropy: $H(x) \equiv -x\log_2 x - (1-x)\log_2(1-x)$

Let $p = 2 \cdot 2^{(H(\alpha)-1)\cdot \ell}$

$$e d^2 p^{\frac{1}{d+1}} < \frac{1}{2}$$

(3) $2e(d+1) < 2^{\alpha n}$

Algorithm: Given $S_1 \dots S_m \subseteq X$ $\quad |S_i| = \ell \;\; \forall i$

First pass:

for each $j \in X$ pick color red/blue via coin toss

S_i is "bad" if $\le \alpha \cdot \ell$ reds
or $\le \alpha \cdot \ell$ blues

$B \leftarrow \{ S_i \mid S_i \text{ is bad} \}$

1st pass is successful if all "connected components" of B are $\le d \log m$

edge bet A_i & A_j if $A_i \cap A_j \neq \emptyset$
(will change defn later)

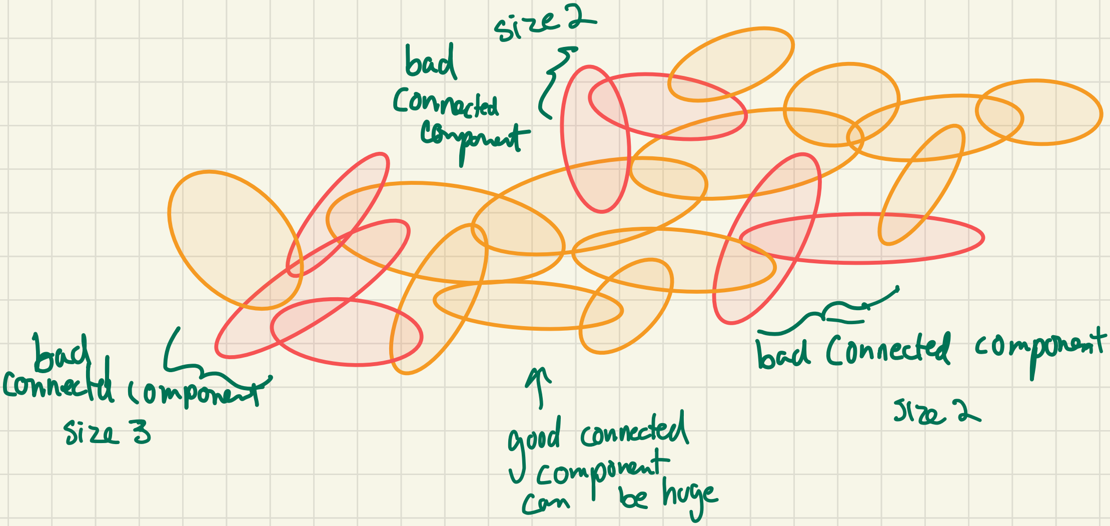
(if not succesful, retry)

Second Pass:

Brute force each connected component (w/o violating their nbrs)

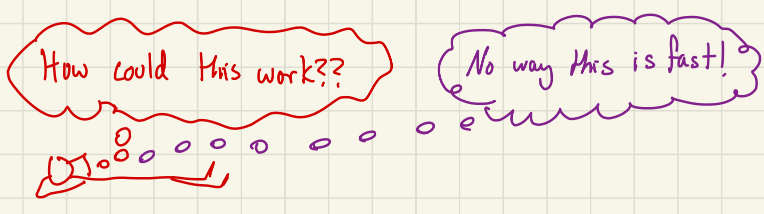
few sets so maybe efficient?

After 1st pass: orange S_i's are "good", red S_i's are "bad"



Some questions:

1. • why is output legal? what if changing S_i's in B makes S_j & B monochromatic?
2. • How fast is pass 2?
3. • How many times to repeat pass 1?



① # Why is output legal?

First pass:

for each $j \in X$ pick color red/blue via coin toss

S_i is "bad" if $\leq \alpha \cdot \ell$ reds
or $\leq \alpha \cdot \ell$ blues

$B \leftarrow \{S_i \mid S_i \text{ is bad}\}$

pass successful if all "connected components" of bad S_i's are $\leq d \log m$

(if not succesful, retry)

Second Pass: Brute force each connected component

Main idea: remaining subproblems each have property that all remaining sets have enough uncolored points so that LLL $\Rightarrow$ soln exists

If S_i not bad & $< \alpha n$ nodes in bad nbrs then S_i will still be bichromatic after recoloring.

If S_i not bad & has $\geq \alpha \ell$ nodes in bad nbrs,

then $\geq \alpha \ell$ nodes get recolored

note: won't use this algorithm →

- if recolored randomly, $\Pr[S_i \text{ is monochrom}] < 2^{-\alpha \ell}$
- using LLL

assumption ✱ + assume $2e(d+1) < 2^{\alpha \ell}$ $\Longrightarrow$ Solution exists!

this was assumption 3

2

How fast is Pass 2?

Main idea: Components small $\Rightarrow$ involve few sets $\Rightarrow$ involve few elements (since assume ℓ is $O(1)$) $\Rightarrow$ can brute force each one separately

size of surviving components $\leq O(d \log m)$

\# settings to vars in a surviving component $\leq 2^{\ell \, O(d \log m)}$

$= m^{O(\ell \cdot d)}$

total time: $\underbrace{\text{\# surviving components}}_{\leq m} \times m^{O(\ell d)} = m^{O(\ell d)}$

if d, ℓ constant: poly(m) time * assumption

else, recurse on components

3 How many times to repeat pass 1?

Complications:

- need "refined" def of "connected component" for pass 2 to work

  why? since need to recolor some nonbad sets that neighbor bad sets

Let's be more careful in our defn. of conn components:

Input → **Hypergraph:** nodes for each $x \in X$

hyperedge S_i corresponds to subset of X

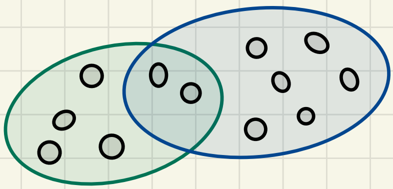
(all $|S_i| = 2 \Rightarrow$ usual notion of graph)

not directed in this case

Dependency digraph: nodes for each S_i

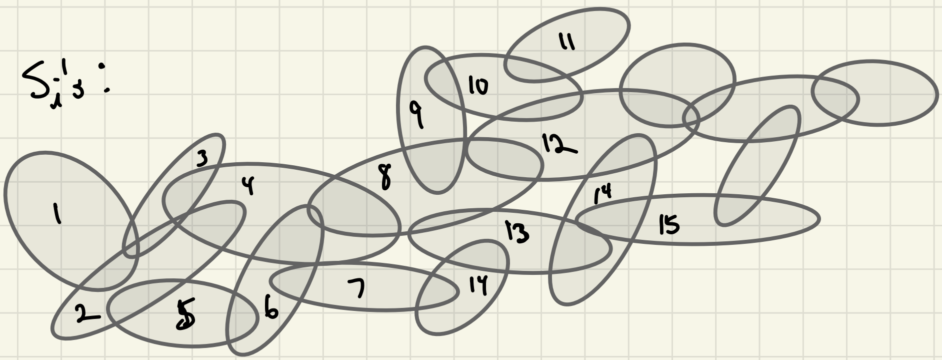
regular type of edge! → edge between S_i & S_j if intersect

hyperedge S_i

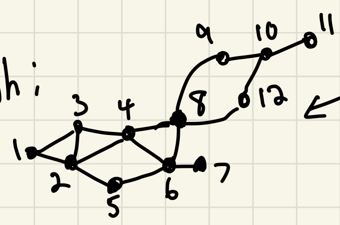


S_j hyperedge

All S_i's:

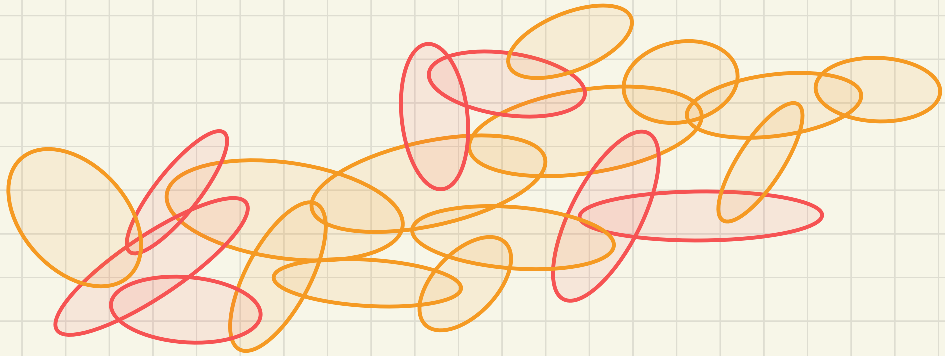


Piece of Dependency Digraph:



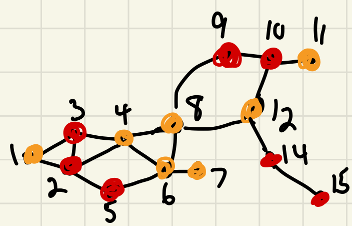
assumption ⇒ this graph has degree $\leq d$

After 1^{st} pass: orange S_i's are "good", red S_i's are "bad"



How should we define "connected component"?

Try 1: use dependency graph

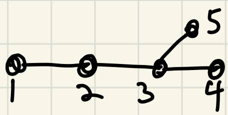


degree $\le d$ by assumption

we will see a difficulty with this soon

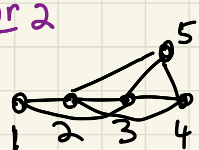
Try 2: use "square" of dependency graph: connect nodes of dist 1 or 2

example:



graph G

$\Rightarrow$



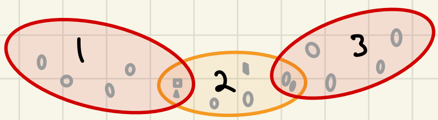
graph G^2

degree of "square" graph:

deg ≤ # nodes that can be reached in 1 or 2 steps in original graph

$$\le \underset{\text{1 step}}{d} + \underbrace{\underset{\text{1st step}}{d} \cdot \underset{\text{2nd step}}{d}}_{\text{2 steps}} \le 2d^2$$

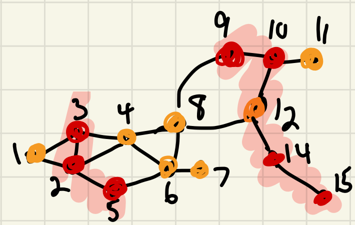
why square graph?



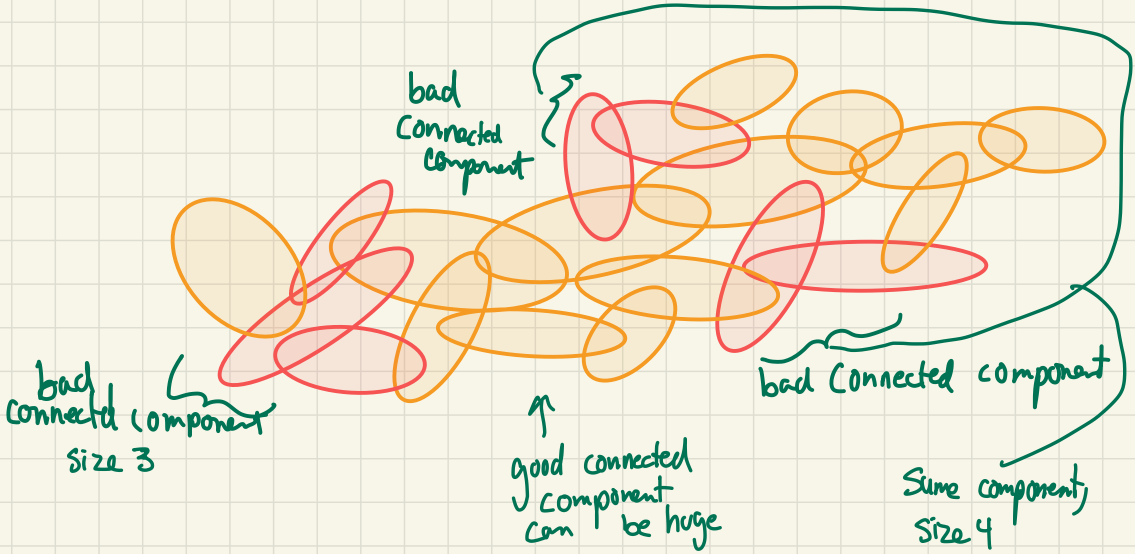
1 + 3 both cause elts in 2 to be recolored

$\Rightarrow$ step 2 needs to recolor 1, 2, 3 simultaneously

For this lecture, "connected" component means all nodes reachable in square graph



After 1st pass: orange S_i's are "good", red S_i's are "bad"



In pass 2, might need to fix neighbors of bad components:

recall: If S_i not bad & has $\geq \alpha \ell$ nodes in bad nbrs, then $\geq \alpha \ell$ nodes get recolored

say S_i "survives" if bad
or has $\geq \alpha \ell$ nodes in bad nbrs

We will show that connected components of "bad" sets S_i are small: $O(\log n)$

Algorithm needs to recolor bad sets

+ possibly some of their nbrs in original graph (the ones that survive):

each bad set S_i has $\leq d$ nbrs

$\Rightarrow$ total size (# S_i's) of component to recolor is $O(d \log n)$

How many repetitions of Pass 1?

fact for $H(x) = -x\log_2 x - (1-x)\log_2(1-x)$

$$\forall S_i, \quad \Pr[S_i \text{ bad}] \leq 2 \cdot \sum_{i \leq \alpha n} \binom{\ell}{i} / 2^{\ell} \leq \underbrace{2 \cdot 2^{(H(\alpha)-1)\ell}}_{\text{define this to be } p}$$

$\leq p$

$\approx 2^{-c\ell}$ for some const c

Given dependency digraph G,

put edge between S_i & S_j if $S_i \cap S_j \neq \emptyset$

if $S_{i_1}, S_{i_2}, \ldots, S_{i_m}$ are independent set

no edges between them

(so $S_{i_k} \cap S_{i_\ell} = \emptyset \;\; \forall\, i_k, i_\ell$)

then $\Pr[S_{i_1} \ldots S_{i_m} \text{ all in } B] \leq p^m$

since mutually independent

First try

Show no big component survives:

$$\Pr[\text{specific } \underset{\text{size } s}{\text{big}} \text{ component survives}]$$

size s

$$\le \Pr[\text{big independent set in component survives}]$$

size $s' < s$

$$\le p^{s'}$$

$$\Pr[\text{any big component survives}]$$

$$\le \#\ \text{potential big components in dependency graph} \cdot p^{s'}$$

every possible connected subgraph of original graph. lots of these

what is a good bound?

$\binom{n}{s}$? way too big!!

Can use degree bound to improve!!

how does s' compare to s?

if component is clique, then s' could be 1

but, use degree bound!

Plan: hope to show no big component survives.

if big component C survives, — this doesn't exist whp

can get good bound on # bounded degree subtrees! → then C has a big subtree that survives — this doesn't exist whp

since bounded degree → then can find (less) big independent set in subtree — show this doesn't exist whp ("with high probability")

Well Known fact:

subtrees of size u in graph of degree $\le \Delta$ is $\le n \cdot \frac{1}{(\Delta-1)(u+1)} \binom{\Delta u}{u}$

#nodes $= n$

$$\le n(e\Delta)^u$$

much much better than $\binom{n}{u}$ when Δ is constant

Given subtree of size u,

it has indep set of size $\geq \dfrac{u}{\Delta+1}$

interesting if $\Delta \ll n$

why?

each round:

- I gets bigger by 1
- subtree gets smaller by $\leq \Delta+1$

Repeat

$I \leftarrow$ arbitrary node u in subtree

remove u + all nbrs of u from subtree

Until subtree is empty

$\Rightarrow$ # rounds $= |I| \geq \dfrac{u}{\Delta+1}$

New try:

Show no big component survives:

$$E[\#\text{ of size} > S \text{ subtrees that survive}]$$

$$\le \sum_{i=S}^{m} E[\#\text{ size } i \text{ subtrees that survive}]$$

hiding an indicator argument in there :)

$$\le \sum_{i=S}^{m} (\#\text{ size } i \text{ subtrees}) \times \Pr[\text{size } i \text{ subtree survives}]$$

$$\le \sum_{i=S}^{m} m \cdot \underbrace{(e d^2)^i \times \left(p^{\frac{i}{d+1}}\right)}_{\left(e\, \underbrace{d^2 p^{\frac{1}{d+1}}}_{\text{assume this is } < 1/2}\right)^i}$$

(*)

$$\le \sum_{i=S}^{m} m \cdot \frac{1}{2^i} < \frac{m}{2^{S-1}}$$

for $S = \log 4m$

$$\le \frac{m}{4m} = 1/4$$

upper bound on expected # of big components

By Markov's ≠:

i.e. ≥ 1

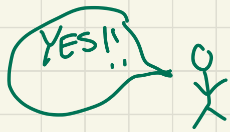
$$\Pr[\#\text{ of size} \geq \log 4m \text{ subtrees } > 0] < \frac{1}{4}$$

so $\Pr[\#\text{ components of size} \geq \log 4m \text{ is} > 0] < 1/4$

$\Longrightarrow$ expected # times to repeat first pass ≤ 4

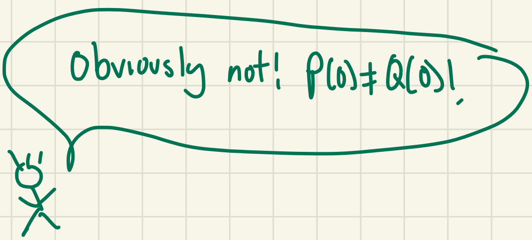
Polynomial Identity Testing

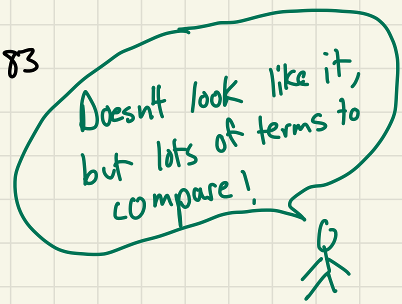
Is $P(x) = (x+1)^2$ the same as $Q(x) = x^2+2x+1$?



What about $P(x) = (x+3)^{38}(x-4)^{83}$

& $Q(x) = (x-4)^{38}(x+3)^{83}$





Problem: given 2 polynomials P, Q

is $P \equiv Q$?

i.e. is $P(x) = Q(x) \; \forall x$?

Problem': given polynomial R

is $R \equiv 0$?

i.e. is $R(x) = 0 \; \forall x$?

Let $R(x) = P(x) - Q(x)$ then $R \equiv 0$ iff $P \equiv Q$

Fact: If $R \not\equiv 0$ has degree $\leq d$ then

R has at most d roots (recall: a "root" is x st. $R(x) = 0$)

Algorithm for deciding whether $R \equiv 0$:

pick $d+1$ distinct inputs $x_1 \dots x_{d+1}$

if $\forall i$ $R(x_i) = 0$ output "$R \equiv 0$"

else ($\exists i$ st. $R(x_i) \neq 0$) output "$R \not\equiv 0$"

Runtime: $O(d)$ evaluations of R