Lecture 5:

- Using Greedy Algorithms to design
  Sublinear Time Algorithms —
  the case of maximal matching

- Property testing:
  Is the graph Planar?
Sublinear Time Approximation Algorithms via Greedy

Estimating size of maximal matching in degree bounded graph

Why?

- Relation to Vertex Cover
  - $VC \geq MM \quad \text{for each edge in matching, } 2 \text{ edges must be in these are disjoint}$
  - $VC \leq 2MM \quad \text{put all } MM \text{ nodes in } VC$ if an edge not covered, then violates maximality

- A step towards approx maximum matching

Note: if $\deg \leq d$, maximal matching $\approx \frac{n}{d}$ \quad \text{to see this, run greedy algorithm}

Greedy Sequential Matching Algorithm:

\[
M \leftarrow \emptyset \\
\forall e = (u,v) \in E, \\
\quad \text{if neither } u \text{ or } v \text{ matched, add } e \text{ to } M \\
\]

Output $M$

Observe: $M$ maximal, since if $e \in M$ either $u$ or $v$ already matched earlier
Oracle reduction Framework

- Assume given deterministic "oracle" $O(e)$ which tells you if $e \in M$ or not in one step.
- $\epsilon \leq \delta = \frac{\delta}{2^2}$ nodes chosen iid.
- For all $v \in S$, $X_v = \sum_{i=1}^{\delta} 1$ if any call to $O(v_i, w_i)$ for $w_i \in N(v)$ returns "yes".

Output $\frac{n}{2s} \sum_{v \in S} X_v + \frac{\epsilon}{2} n$ makes an underestimate unlikely since 2 nodes matched for each edge in $M$.

Behavior of output: Why does it work?

$|M| = \frac{1}{2} \sum_{v \in V} X_v$

$E[\text{output}] = E\left[\frac{n}{2s} \sum_{v \in S} X_v\right] + \frac{\epsilon}{2} n$

$= \frac{n}{2s} \sum_{v \in S} E[X_v] + \frac{\epsilon}{2} n$ but $E[X_v] = \frac{2|M|}{|V|} = \frac{2|M|}{n}$

$= \frac{n}{2s} \cdot 2 |M| \cdot \frac{1}{n} + \frac{\epsilon}{2} n = |M| + \frac{\epsilon}{2} n$

$P_r \left[ \left| \frac{n}{2s} \sum_{v \in S} X_v \right| \leq \frac{\epsilon}{2} n \right] = \frac{1}{3}$ by Chernoff-Hoeffding.

Claim with prob $\geq 2/3$, $|M| \leq \text{output} \leq |M| + 3n$. 

\[\frac{n}{2s} \sum_{v \in S} X_v \leq |M| \leq \text{output} \leq |M| + 3n\]
Implementing the oracle:

Main idea: figure out "what would greedy do on (v,w)?"

- how?
- which input order?
- do we need to figure out all previous nodes?

is (v,e) ∈ M?

adjacent to:
(b,c) (e,d) (e,f) (a,b)

Greedy considers 1st & puts (b,c) into M
so (b,e) ∈ M!
no need to consider rest of graph

Problem: Greedy is "sequential" & has long dependency chains?

Example:
even if you know graph is a line, is edge odd or even in order?
Implementing oracle based on greedy:

Algorithm:

Given \( e \), is \( e \) in \( M \)?

- Recursively find out all decisions for adjacent edges with lower order number (do not need any info on adjacent edges with higher order number, since not considered by greedy before \( e \))

- If any adjacent edge before \( e \) in ordering is matched, \( e \) is not matched
  
else \( e \) is matched.
How to break length of dependency chains?

assign random ordering to edges

example

is edge 5 in M?

- recurse on 3
  - recurse on 1
    - no other adjacent edges so 1 is matched
    - therefore 3 is not matched
    - no need to recurse on 7 since 5 < 7
    - don't know yet about 5, so recurse on 4
  - recurse on 2
    - 8 comes after 2 in order so does not affect greedy's behavior
    - same for 4
    - so 2 is matched
    - 4 is not matched
    - 5 is matched
Implementation of oracle: assume ranks \( r_e \) assign to each edge \( e \)

to check if \( e \in M \):

\[
forall e' \text{ neighboring } e, \\
if r_{e'} < r_e \text{ recursively check } e' \\
if e' \in M \text{ return } "e \in M" \text{ halt} \\
else \text{ continue} \\
return "e \notin M" \\
\]

↑ since no \( e' \) of lower rank then \( e \)

is in \( M \)

Correctness: follows from correctness of greedy

Query complexity:

Claim expected \# queries to graph per oracle query is \( 2^{O(d)} \)

\[ \text{Claim } \implies \text{ total query complexity is } \frac{2^{O(d)}}{\varepsilon^2} \]
pf of Claim

- Consider $\text{Query}_T$ where root node is labelled by original query edge, children of each node are edges adjacent to it.

- Will only query paths that are monotone decreasing in rank.

- $Pr[\text{given path of length } k \text{ explored}] = \frac{1}{(k+1)!}$

- $\# \text{ edges in original graph at dist } \leq k \text{ in tree } \leq d^k$

- $E[\# \text{ edges explored at dist } \leq k] \leq d^k \frac{1}{(k+1)!}$

- $E[\# \text{ total } \# \text{ edges explored}] \leq \sum_{k=0}^{\infty} \frac{d^k}{(k+1)!}$

\[ \leq \frac{e^d}{d} \]
Property Testing

Can we distinguish? in sublinear time?

eg. $P = \text{"planar"}$

Compromise

Can we distinguish graphs with prop $P$ from far from $P$?

eg. $G$ is $\varepsilon$-far from planar if must remove $\geq \varepsilon \cdot d_{\max} \cdot n$ edges to make it planar

Today: Test planarity in time independent of $n$ (but exponential in $\varepsilon$)
Testing $H$-minor freeness

all graphs have max degree $\leq d$

def. $H$ is "minor" of $G$ if can obtain $H$ from $G$ via vertex removals, edge removals, edge contractions

$G$ is "$H$-minor-free" if $H$ not minor of $G$

$G$ is "$\epsilon$-close to $H$-minor-free" if can remove $\leq \epsilon d n$ edges to make it $H$-minor-free

(i.e. $G$ is "$\epsilon$-far")

minor closed property $P$ if $G \in P$ then all minors of $G$ are in $P$

Really Cool Theorem [Robertson+ Seymour]

Every minor-closed property is expressible as a constant # of excluded minors.

Some minor-closed properties: $K_{3,3}$ or $K_5$

planar graphs, bounded tree width, ...

Goal: Testing $H$-minor freeness

Pass if far from $H$-minor-free

Fail if $H$-minor free graphs
more definitions

\( x \) can be a function of \( \epsilon \)

- \( G \) is "\((\epsilon, k)\)-hyperfinite" if
  
  can remove \( \leq \epsilon n \) edges
  
  \( \& \) remain with connected components of size \( \leq k \)

Useful Theorem

Given \( H \) constant that depends only on \( H \)

\( \exists C_H \) s.t. \( \forall 0 < \epsilon < 1 \), every \( H \)-minor free graph of degree \( d \)

is \((\epsilon d, C_H \epsilon^2)\)-hyperfinite.

(i.e. remove \( \leq \epsilon d n \) edges \& components of size \( O(\epsilon^2) \))

Note

Subgraphs of \( H \)-minor free graphs also \( H \)-minor free

\( \& \) so also hyperfinite

but, only remove \#edges in proportion to \#nodes in subgraph
Why is hyperfiniteness useful?

Partition graph $G$ into $G'$

- only const-size connected components remain
- removed only few edges ($\leq Ed_n$)
- if can't do this, $G$ is not $H$-minor free

If $G'$ is close to having property, so is $G$

Constant time

So test $G'$ by picking random components $\tau$ seeing if they have the property

Need a "local" (sublinear) way to determine $G'$, For

now assume we have "partition oracle" $P$

(with parameters $\frac{Ed}{4^i}, k$)

\[ \frac{|\text{fraction edges removed}|}{\text{component size}} \]

Input: vertex $v$

Output: $P[v]$ (v's partition name)

s.t. $\forall v \in V$

(1) $|P[v]| \leq K$

(2) $P(v)$ connected

+ if $G$ is $H$-minor free

with prob $\geq \frac{9}{10}$

$|\{v(u) \in E \mid P(u) \neq P(v)\}| \leq Ed_n \frac{1}{q}$