Lecture 5:

• Greedy algorithms vs. Sublinear time: the case of maximal matching

• Property Testing:

  is the graph planar?
Sublinear time algorithms via greedy:

We focus on problem of estimating size of maximal matching ($MM$) in degree bounded graph.

Why?

- step towards approx maximum matching
- relation to Vertex cover ($VC$)

\[
VC \geq MM \quad \text{for each edge in matching,}\quad \geq 1 \text{ endpt must be in } VC \\
\quad \quad \quad \quad \text{these are disjoint!}
\]

\[
VC \leq 2 \cdot MM \quad \text{put all } MM \text{ nodes in } VC \\
\text{if any edge not covered by } VC \text{ then can add edge to } MM \text{ violating maximality of } MM.
\]
Note (similar to VC)

if degree $\leq \Delta$, maximal matching $\geq \frac{m}{2\Delta}$

why? run process:
place edge $(u,v)$ in $MM$
delete other edges of $u + v$ $(\leq 2\Delta)$ which can no longer be in matching

Greedy Sequential Matching Algorithm:

$M \leftarrow \emptyset$

$\forall e = (u,v) \in E$

if neither $u$ or $v$ matched
add $e$ to $M$

Output $M$

Observation:

$M$ is maximal

why? if $e \notin M$, 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.

Algorithm to estimate $|M|:

- $S \leftarrow \text{set of } s = \frac{\delta}{\varepsilon^2} \text{ nodes chosen iid}
- \forall v \in S'
  \begin{align*}
  &\text{let } X_v \leftarrow 1 \text{ if any call to } O(v, w) \text{ for } w \in N(v) \text{ returns "yes" } \\
  &\text{0 } o.w.
  \end{align*}
- Output $\frac{\delta}{2s} \sum_{v \in S} X_v + \frac{\varepsilon}{2} \cdot n$

since 2 nodes makes underestimate unlikely
matched for each edge in $M$
Behavior of output:

(why a good approximation?)

- $S \leftarrow \text{set of } s=\frac{8}{\varepsilon^2} \text{ nodes chosen iid}$
- $\forall v \in S$, if any call to $O(v,w)$ for $w \in N(v)$ returns "yes"
  - let $X_v \leftarrow \{0\}$ o.w.
- Output $\frac{n}{2s} \sum_{v \in S} X_v + \frac{\varepsilon}{2} \cdot n$

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

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

$$= \frac{n}{2s} \sum_{v \in S} E[X_v] + \frac{\varepsilon}{2} \cdot n$$

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

$$= |M| + \frac{\varepsilon}{2} \cdot n$$

$$Pr\left[1\left(\frac{n}{2s} \sum_{v \in S} X_v + \frac{\varepsilon}{2} \cdot n\right) - E[\text{output}] \geq \frac{\varepsilon}{2} \cdot n\right]$$

$$\leq \frac{1}{3}$$ by additive Chernoff-Hoeffding

Claim with prob $\geq 2/3$, $|M| \leq \text{output} \leq |M| + \varepsilon n$
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 \((b,e) \in M\)? adjacent to

\[(b,c) (e,d) (e,f) (g,b)\]

\[1 5 6 11\]

Greedy considers first
output \((b,c)\) into \(M\)
So \((b,e) \notin M\)

\[\Rightarrow \text{no need to consider rest of graph}\]

Problem: Greedy is “sequential” + has long dependency chains?

Example:

even if you know graph is line,
is edge odd or even in order?
Implementation of oracle:
Input: edge $e$
Output: is $e \in M$?

Algorithm:
• Recursively find all decisions for adjacent edges with lower ordering number (do not need info on adjacent edges with higher number since greedy doesn't consider before $e$)

• if any adj. edge with lower number is matched then $e$ is not matched
  else $e$ is matched

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

Example:
\[
\begin{array}{ccccccc}
& & & & & & \\
1 & 2 & 3 & 4 & 5 & \ldots & 212 \\
& & & & & & \\
2 & 3 & 1 & 2 & 3 & 4 & \ldots & 212 \\
\end{array}
\]

\[
\Rightarrow \text{even if you know graph is line, is edge odd or even in order?}
\]
How to break length of dependency chains?

assign random ordering to edges

Example:

Is edge 0.5 in M?

* recurse on 0.3
  * recurse on 0.1
    * no other adjacent edges so 0.1 matched
    * therefore 0.3 not matched
  * no need to recurse on 0.7 since 0.5 < 0.7
* recurse on 0.4
  * recurse on 0.2
    * 0.8 comes after 0.2
    * 0.4 also
    * so 0.2 matched
    * so 0.4 not matched
* 0.5 matched
Implementation of oracle:

assume ranks re assigned to each edge e to check if e ∈ M:

∀ e' neighboring e,

  if r_e', < r_e recursively check e'

  & if e' ∈ M, return "e ∈ M" & halt

else continue

Return "e ∈ M"

\[ \text{since no e' of lower rank than e is in M} \]

Correctness:

follows from correctness of greedy

Query complexity:

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

Claim + Parnas-Ron reduction \( \Rightarrow \) total query complexity is \( \frac{2d}{\epsilon^2} \)
Pf of Claim:

• Consider query tree:
  - root node labelled by original query edge
  - children of each node are adjacent edges

• will only query paths that are monotone decreasing in rank

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

• \# edges in original graph at dist = k in tree is at most \( d^k \)

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

• \( E[\text{total } \# \text{edges explored}] \leq \sum_{k=0}^{\infty} \frac{d^k}{(k+1)!} \leq e^d \)
Can we distinguish graphs with property \( P \) from \( \not\exists \) close to \( P \)?

E.g., \( G \) is \( \varepsilon \)-far from planar if must remove \( \geq \varepsilon \cdot \Delta \cdot n \) edges to make it planar.
Today & next time:

test planarity in time independent of \( n \)
(but exponential in \( \varepsilon \))

for graphs with max degree \( \Delta \)

What is a planar graph?
Can be drawn on plane s.t. edges don't intersect

\[
\begin{align*}
K_3 & \quad \text{Yes} \\
K_{2,2} & \quad \text{NO? actually, yes} \\
K_{3,3} & \quad \text{NO!}
\end{align*}
\]
Cool characterization of planar graphs:

def. $H$ is "minor" of $G$ if

can obtain $H$ from $G$ via vertex removals, edge removals or edge contractions

Minor closed properties:

Let $P$ be a set of graphs

e.g. $P =$ planar graphs $\leftarrow$ minor-closed

$P =$ bipartite graphs $\not\leftarrow$ not minor-closed

$P$ is "minor closed" if

$\forall G \in P$ then all minors of $G$ are in $P$
Minor free graph families:

def G is "H-minor-free" if H is not a minor of G.

Thm [Kuratowski]

G is planar iff G is $K_{3,3}$ and $K_5$ minor-free.

Another example: bounded tree width (there are more...)

Really cool theorem: [Robertson & Seymour]

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

Can't hope to distinguish in sublinear time!

$$\text{def } G \text{ is } \varepsilon\text{-close to } H\text{-minor-free}$$

if can remove at most $$\varepsilon\cdot \Delta n$$ edges to make it $$H$$-minor-free

Specifically:

$$\text{def } G \text{ is } \varepsilon\text{-close to planar iff can remove at most } \varepsilon\cdot \Delta n \text{ edges to make it }$$

$$\leq$$ planar \iff \varepsilon$$K_{3,3}+K_{5}$$-free

else $$G$$ is $$\varepsilon$$-far

Goal: Given $$G$$

- if $$G$$ planar, PASS
- if $$G$$ $$\varepsilon$$-far from planar, FAIL

with prob \( \geq \frac{2}{3} \) arbitrary const \( \geq 1/2 \)
Plan for tester: use nice property of Planar (all all $H$-minor-free) graph families.

Can always remove small fraction of edges $\leq \epsilon$

d break up graph into tiny connected components $\leq \text{const}$

**Def.** $G$ is "$(\epsilon, k)$-hyperfinite" if

- Can remove $\leq \epsilon n$ edges
- Remain with connected components of size $\leq k$

**Useful Thm**

Given $H$, $\exists$ const $C_H$ s.t.

For $0 < \epsilon < 1$, every $H$-minor-free graph $G$ of $\deg \leq \Delta$ is $\left( \epsilon \cdot \Delta, \frac{C_H}{\epsilon^3} \right)$-hyperfinite

Remove $\leq \epsilon \cdot \Delta n$ edges

Components of size $O(\frac{1}{\epsilon^2})$ no dependence on $n$
Note: subgraphs of $H$-minor free graphs are also $H$-minor free, so also hyperfinite. But only remove edges in proportion to the number of nodes in subgraph. \( \Rightarrow \) can recurse & break up further.