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 problem
- relation to Vertex Cover (VC)

\[ |VC| \geq |MM| \leftarrow \text{for each edge in matching } \begin{array}{c} \rightarrow \text{u or v has to be in VC} \\ \text{true for any matching not just maximal} \\ + \text{matching edges are node-disjoint} \end{array} \]

\[ |VC| \leq 2 \cdot |MM| \leftarrow \text{put all nodes in MM into VC} \begin{array}{c} \rightarrow \text{uv} \in MM \text{ put uv into VC} \\ \text{why is this VC? if any edge not covered by VC we can add to MM } \end{array} \]
Note (similar to VC)

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

why? run process

when place edge \((u, v)\) into MM

delete other edges of \(u\) or \(v\) \((\leq 2\Delta)\) which can no longer be in matching

all other edges are fair game

Greedy Sequential Matching Algorithm:

\[
\begin{align*}
M &\leftarrow \emptyset \\
\forall e = (u, v) \in E &\quad \text{one by one (in some order)} \\
\text{if neither } u \text{ or } v \text{ matched} &\quad \text{previously in this order} \\
\text{add } e \text{ to } M \\
\end{align*}
\]

Output \(M\)

Observation:

\(M\) is maximal

why? if \(e \not\in M\) then either \(u\) or \(v\) already matched
Oracle Reduction Framework: (Parnas - Ron)

Assume given deterministic "oracle" \( O(e) \)
which tells you if \( e \in M \) or not in one step.

Algorithm to estimate \( |M| \):

- \( S \leftarrow \) set of \( s = \frac{8}{\varepsilon^2} \) nodes chosen iid
- \( \forall v \in S' \)
  - let \( X_v \leftarrow 1 \) if any call to \( O(v,w) \) for \( w \in N(v) \) returns "yes"
  - \( 0 \) o.w.
- Output \( \frac{n}{2s} \sum_{v \in S'} X_v + \frac{\varepsilon}{2} \cdot n \)

Since 2 nodes are matched for each edge in \( M \), average of \# nodes matched in sample unlikely to underestimate.
Behavior of output:

(why a good approximation?)

\[ S \leftarrow \text{set of } s = \frac{8}{\varepsilon^2} \text{ nodes chosen iid} \]

\[ \forall v \in S \text{ let } X_v \leftarrow \begin{cases} 1 & \text{if any call to } G(v, u) \text{ for } u \in N(v) \text{ returns "yes" } \\ 0 & \text{o.w.} \end{cases} \]

\[ \text{Output } \frac{n}{2s} \sum_{v \in S} X_v + \frac{\varepsilon}{2} n \]

not \[ |M| = \frac{1}{2} \sum_{v \in V} X_v \]

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

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

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

\[ = |M| + \frac{\varepsilon}{2} n \]

\[ P_r \left[ \left| \frac{n}{2s} \sum_{v \in S} X_v + \frac{\varepsilon}{2} n \right| - E[\text{output}] \right] \geq \frac{\varepsilon}{2} n \] \leq \frac{1}{3} \text{ 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)?"

Is \((b,e) \in M\)?

adjacent edges:
\((b,c), (a,b), (d,e), (e,f)\)

Since \((b,c)\) is 1st edge considered
\((b,c) \in M\)
\(\Rightarrow (b,e) \notin M\)

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

even if you know graph is line
but don't know greedy order

Example:

```
1 2 3 4 5 6 7 8 9 10 11
```

\(\text{(flip) evens } \notin M\)
Saving grace: assume random order

Implementation of oracle:

Input: edge e
Output: is e ∈ M?

Algorithm:

- Recursively find all decisions for adjacent edges with lower ordering number (do not need to know what greedy did on higher order # since not considered 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:

```
1 2 3 4 5 ... 2/2
A13 1 2 3 4 ... 2/2
```

Even in line 13 e is odd or even in dependency chain
How to break length of dependency chains?

Assign random ordering to edges (ranks are numbers ∈ [0,1])

Example:

Is edge 0.5 in M?
recurse on 0.3
recurse on 0.1
no other adjacent edges so added to M
so 0.3 not matched

no need to recurse on 0.7

recurse on 0.4
recurse on 0.2
all of 0.2's nbrs are bigger so 0.2 ∈ M
so 0.4 ∈ M

so greedy puts 0.5 ∈ M
Implementation of oracle:

Assume ranks $r_e$ assigned to each edge $e$ to check if $e \in M$:

$\forall e'$ neighboring $e$,
- if $r_{e'} < r_e$ recursively check $e'$
  - if $e' \in M$, return "$e \in M$" and halt
- else continue

Return "$e \in M$"

As since no $e'$ of lower rank is in $M$

Correctness: exactly following greedy so follows from correctness of greedy

Query complexity:

Claim expected $g$ queries to graph per oracle query is $2^{2n}$

Claim + Parnas-Ron oracle reduction $\Rightarrow$ total query complexity

(claim)
**Pf of Claim:**

- Consider query tree:
  - root node labelled by original query edge
  - children of each node are all adjacent edges
  - will only go down paths that are decreasing in rank
- \( \Pr[\text{given path of length } K \text{ explored}] = \frac{1}{K!} \)
- # edges in original graph at dist = k in tree is
  \[ E[\text{edges explored at dist } k] \leq \frac{(2\Delta)^k}{K!} \]
- \( E[\text{total # edges explored}] \leq \sum_{K=0}^{\infty} \frac{(2\Delta)^K}{K!} \leq O(\Delta) \)
  
  \[ \frac{e}{\Delta} \]
Property Testing

examples of P:
- planar
- bipartite
- no small cuts
- no triangles
- Connected

All graphs

$\epsilon$-close to $P$

Graphs with property $P$

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

E.g. $G$ is $\epsilon$-far from planar if must remove $\geq \epsilon \cdot \Delta \cdot n$ edges to make it planar.