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 <u>maximum</u> matching problem
- relation to Vertex Cover (VC)

$|VC| \geq \underbrace{|MM|}_{\text{true for } \underline{\text{any}} \text{ matching not just maximal}}$ ← for each edge in matching

u or v has to be in VC

+ matching edges are node-disjoint

$|VC| \leq 2 \cdot |MM|$ ← put all nodes in MM into VC

$(u,v) \in MM$ put u, v into VC

why is this VC? if any edge not covered by VC we can add to MM → ⇐

$\uparrow$
Contradicts maximality of MM

<u>Note</u> (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:

$M \leftarrow \emptyset$

one by one (in some order)

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

if neither u or v matched previously in this order

add e to M

Output M

Observation:

M is maximal

why? if $\underset{(u,v)}{\overset{}{e}} \notin M$ then either u or v already matched

Oracle Reduction Framework: (Parnas - Ron)

Assume given deterministic "oracle" $\mathcal{O}(e)$ 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 $X_v \leftarrow \begin{cases} 1 & \text{if any call to } \mathcal{O}(v,w) \text{ for } w \in N(v) \text{ returns "yes"} \\ 0 & \text{o.w.} \end{cases}$

- Output $\frac{n}{2s} \sum_{v \in S'} X_v + \underbrace{\frac{\varepsilon}{2} \cdot n}_{\text{unlikely to underestimate}}$

since 2 nodes are matched for each edge in M

average # nodes matched in sample

Behavior of output:

(why a good approximation?)

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

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

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

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

$\leftarrow$ but $E[X_v] = \frac{2|M|}{n}$

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

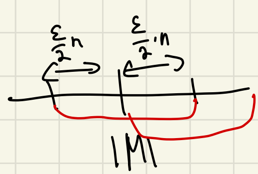
$$= |M| + \frac{\varepsilon}{2} \cdot n$$

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

by additive Chernoff Hoeffding

$$\Pr\left[\left|\frac{n}{2s} \sum_{v \in S'} X_v - |M|\right|\right.$$

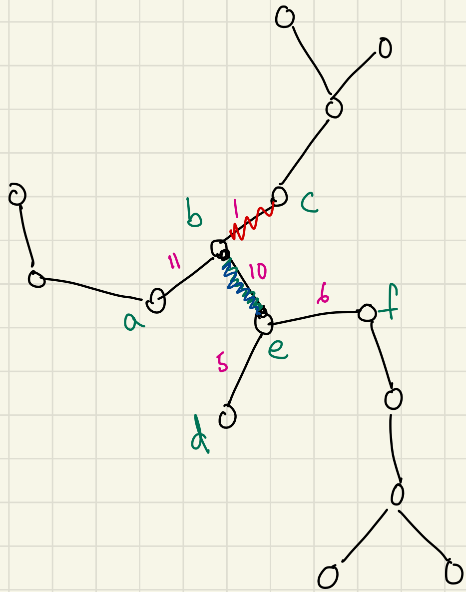
Claim with prob $\geq 2/3$, $|M| \leq \text{output} \leq |M| + \varepsilon \cdot n$



Implementing the oracle:

Main idea: figure out "$\underbrace{\text{what would greedy}}$ do on (v,w)?"

how?
which input order?
do we need to figure out all past choices?

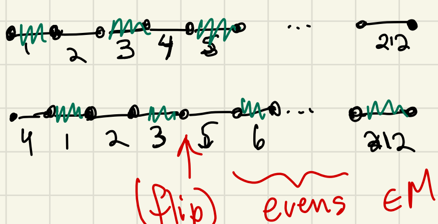


Is $(b,e) \in M$?
adjacent edges: (b,c), (a,b), (d,e), (e,f)
1 11 5 6
comes after 10

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

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

example:



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

Saving grace: assume random order

Implementation of oracle: Input: edge e

Output: is $e \in M$?

Algorithm:

- recursively find all decisions for adjacent edges with lower ordering number

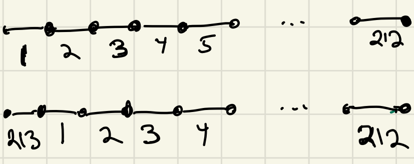
(do not need to know what greedy did on higher order #'s since not considered before e)

- 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:

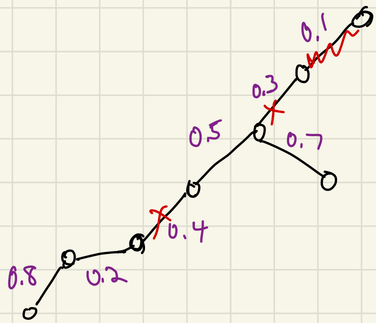


even in line is e is odd or even in dependency chain

How to break length of dependency chains?

assign random ordering to edges
(ranks are numbers $\in [0,1]$)

<u>Example</u>:



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 <u>not</u> 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
$0.2 \in M$

so $0.4 \notin M$

so greedy puts $0.5 \in M$

Implementation of oracle:

assume ranks $r_e \leftarrow (0,1)$ 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 "$e \notin M$" & halt

else continue

return "$e \in M$" ← since no e' of lower rank is in M

Correctness: exactly following greedy so follows from correctness of greedy

Query complexity:

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

Claim + Parnas-Ron oracle reduction ⇒ total query complexity to estimate MM is $\frac{1}{\varepsilon^2} \cdot 2^{O(d)}$

∀ e' neighboring e,

- if $r_{e'} < r_e$ recursively check e'

  & if $e' \in M$, return "$e \notin M$" & halt

  else continue

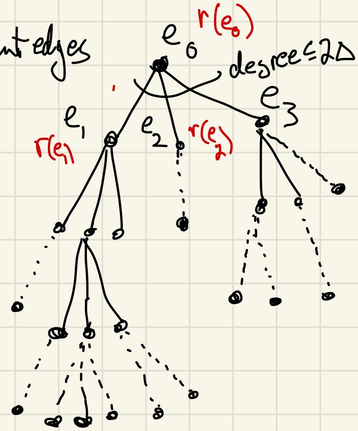
return "$e \in M$"

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

- $\Pr[\text{given path of } \underbrace{\text{length } k}_{\#\text{ edges}} \text{ explored}] = \frac{1}{k!}$

- # edges in original graph at dist = k in tree is $\leq (2\Delta)^k$

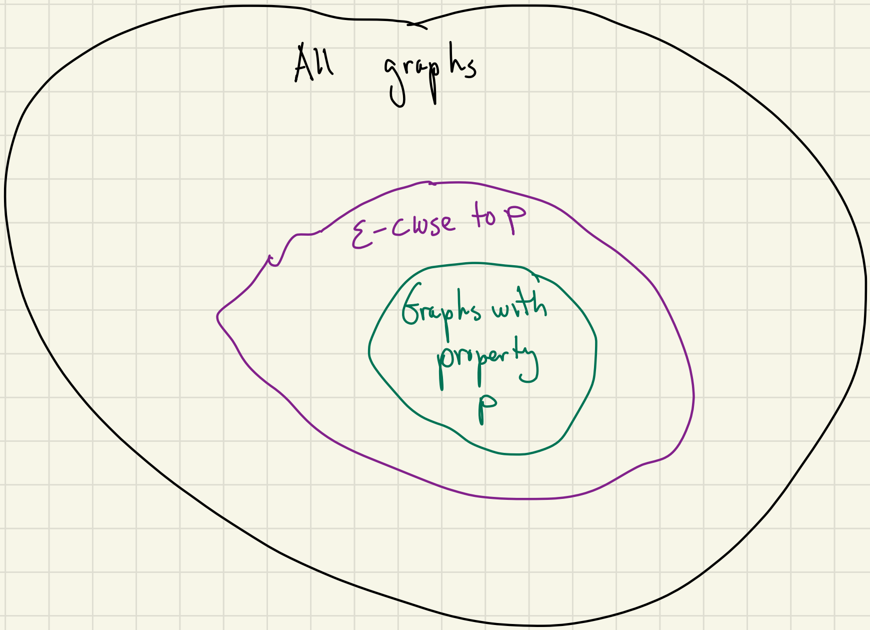
- $E[\#\text{edges explored at dist } k] \leq \frac{(2\Delta)^k}{k!}$

- $E[\text{total } \#\text{ edges explored}] \leq \sum_{k=0}^{\infty} \frac{(2\Delta)^k}{k!}$

$$\leq e^{O(\Delta)}$$

$$\frac{e}{\Delta}$$

Property Testing



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

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

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