Lecture 5:

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

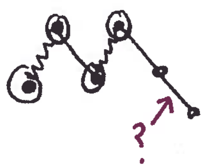
- Property testing: Is the graph Planar?

Sublinear Time Approximation Algorithms via Greedy

Estimating size of maximal matching in degree bounded graph

Why?

- relation to Vertex Cover



  - $VC \geq MM$ ← for each edge in matching, ≥ 1 endpt must be in VC; these are disjoint
  - $VC \leq 2MM$ ← put all MM nodes in VC; if an edge not covered, then violates maximality

- a step towards approx maximum matching

Note: if $\deg \leq d$, Maximal matching $\geq \frac{n}{d}$ ← to see this, run greedy algorithm

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

output depends only on ordering of input edges

Observe:

M maximal, since if $\underset{(u,v)}{e} \notin M$ either u or v alread matched earlier

Oracle reduction Framework

assume given deterministic "oracle" $\mathcal{O}(e)$ which tells you if $e \in M$ or not in one step

- $S' \leftarrow s = \frac{8}{\varepsilon^2}$ nodes chosen iid.

- $\forall v \in S'$

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

- Output $\underbrace{\frac{n}{2s}}_{\text{since 2 nodes matched for each edge in } M} \sum_{v \in S'} X_v + \underbrace{\frac{\varepsilon}{2} \cdot n}_{\text{makes an underestimate unlikely}}$

Behavior of output: Why does it work?

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

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

$$= \frac{n}{2s} \sum_{v \in S'} E[X_v] + \frac{\varepsilon}{2} \cdot n \quad \leftarrow \text{but } E[X_v] = \frac{2|M|}{|V|} = \frac{2|M|}{n}$$

fraction of matched nodes

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

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

$$\parallel$$

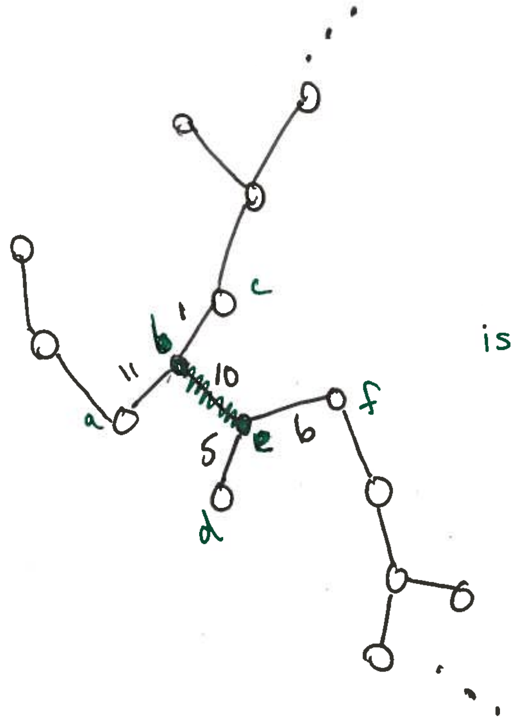
$$\Pr\left[\left|\frac{n}{2s} \sum_{v \in S} X_v - |M|\right| \geq \frac{\varepsilon}{2} n\right] \leq \frac{1}{3} \quad \text{by additive Chernoff-Hoeffding}$$

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)?"

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) (a,b)

1 5 6 11

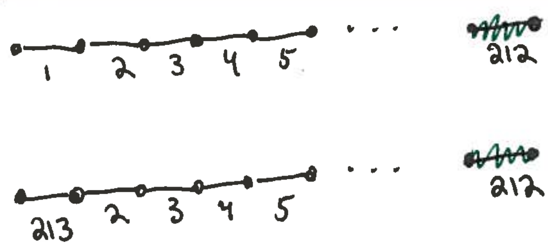
Greedy considers 1st + puts (b,c) into M

so $(b,e) \notin M$!

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 <u>not</u> need any info on adjacent edges with higher order number, since not considered by greedy before e)

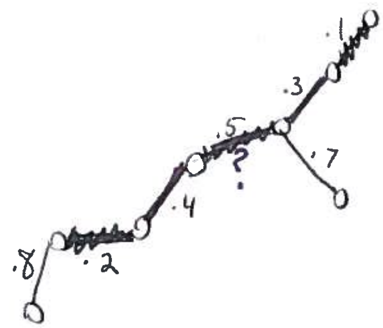
- if any adjacent edge before e in ordering is matched, e is <u>not</u> matched

  else e is matched.

How to break length of dependency chains?

assign <u>random</u> ordering to edges

<u>example</u>



is edge .5 in M?

- recurse on .3
  - recurse on .1
    - no other adjacent edges ~~so~~
      .1 is matched
    - therefore .3 is <u>not</u> 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 doesn't affect Greedy's behavior
    - same for .4
    - so .2 is matched
  - .4 is <u>not</u> matched
- .5 is matched

Implementation of oracle: assume ranks r_e assign 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 than e is in M

Correctness: follows from correctness of greedy

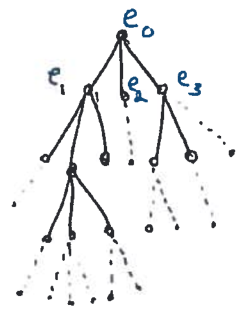
Query complexity:

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

Claim + Parnas-Ron $\Rightarrow$ total query complexity is $\dfrac{2^{O(d)}}{\varepsilon^2}$

Pf of Claim

- Consider QueryTree where root node labelled by original query edge, Children of each node are edges adjacent to it.



- will only query paths that are monotone decreasing in rank

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

- $\#$ edges in original graph at dist $= k$ in tree $\leq d^k$

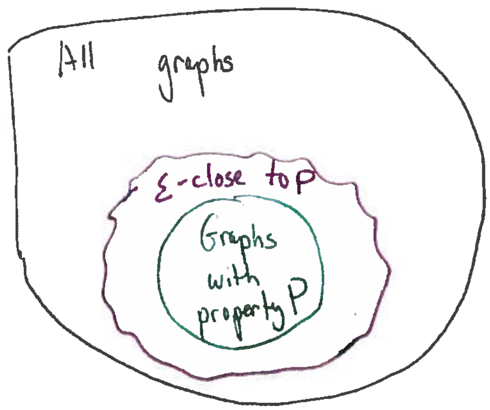
- $E[\#\text{ edges explored at dist } = k] \leq \dfrac{d^k}{(k+1)!}$

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

$$\leq \frac{e^d}{d}$$

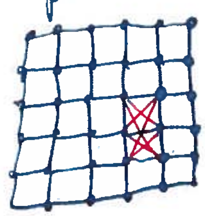
∎

Property Testing



Can we distinguish? in sublinear time?

eg. P = "planar"



Compromise

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

e.g. G is ε-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 ε)

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 "ε-close to H-minor-free" if

can remove $\leq \varepsilon d n$ edges to make it

H-minor-free

(o.w. G is "ε-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

no

planar graph, bounded tree width, ...

Goal: Testing H-minor freeness

Pass H-minor free graphs

Fail if far from H-minor free

more definitions

k can be a function of ε

- G is "(ε, k)-hyperfinite" if

  can remove $\leq \varepsilon n$ edges

  & remain with connected components of size $\leq k$

Useful Thm

Given H

$\exists\ C_H$ (constant that depends only on H) s.t. $\forall\ 0 < \varepsilon < 1$, every H-minor free graph of deg $\leq d$

is $(\varepsilon d, C_H^2/\varepsilon^2)$-hyperfinite.

(i.e. remove $\leq \varepsilon d n$ edges & components of size $O(1/\varepsilon^2)$)

note

Subgraphs of H-minor free graphs also H-minor free

& so also hyperfinite

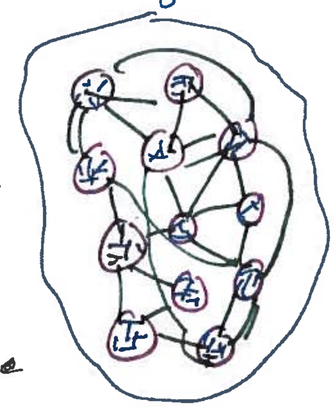
but, only remove # edges in proportion to # nodes in subgraph

Why is hyperfiniteness useful?

Partition graph G into G'

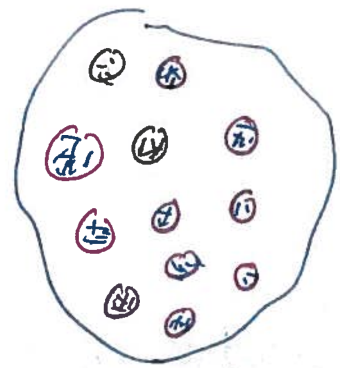
how in sublinear time? { - only const size connected components remain
- removed only few edges ($\le \varepsilon d n$)
- if can't do this, G is not H-minor free

G



$\Downarrow$ remove the few green edges

G'



Easy to test since collection of constant sized graphs!!

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

Constant time { so test G' by picking random components & seeing if they have the property

Need a "local" (sublinear) way to determine G'. For now assume we have "partition oracle" P

(with parameters $\frac{\varepsilon d}{4}$, k)

fraction edges removed; component size

<u>Input</u>: vertex v

<u>Output</u>: $P[v]$ (v's partition name)

s.t. $\forall v \in V$ (1) $|P[v]| \le k$

(2) $P(v)$ connected

& if G is H-minor free

with prob $\ge \frac{9}{10}$ $\quad |\{(u,v)\in E \mid P(u) \ne P(v)\}| \le \frac{\varepsilon d n}{4}$