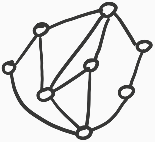
Testable Properties in General Graphs and Random Order Streaming

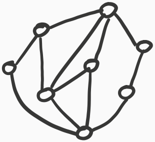
Artur Czumaj, *Hendrik Fichtenberger*, Pan Peng, Christian Sohler

Property Testing: in a Nutshell

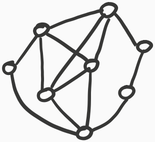


planar ✓

Property Testing: in a Nutshell

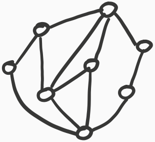


planar ✓

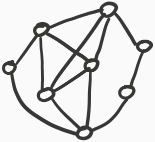


non-planar ✗

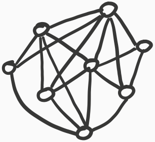
Property Testing: in a Nutshell



planar ✓

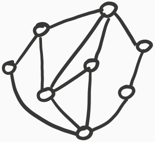


non-planar ✗

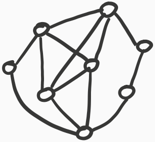


non-planar ✗

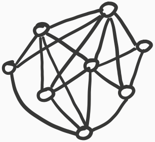
Property Testing: in a Nutshell



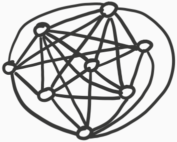
planar ✓



non-planar ✗

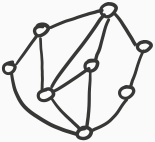


non-planar ✗

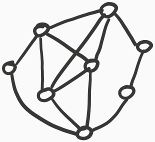


non-planar ✗

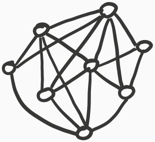
Property Testing: in a Nutshell



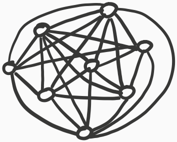
planar ✓



non-planar ✗



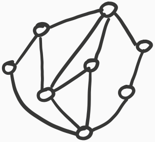
non-planar ✗



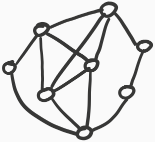
non-planar ✗

time complexity: $\Omega(|V|)$

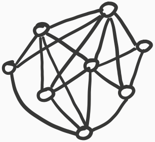
Property Testing: in a Nutshell



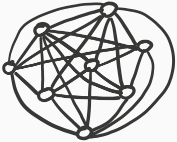
planar ✓



non-planar ✗



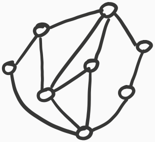
non-planar ✗



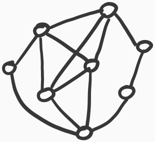
non-planar ✗

time complexity: $\Omega(|V|)$

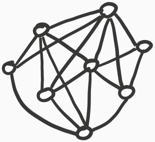
Property Testing: in a Nutshell



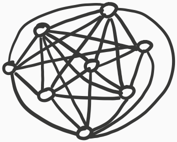
planar ✓



slightly non-planar ✗



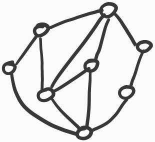
quite non-planar ✗



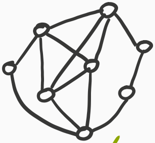
very non-planar ✗

time complexity: $\Omega(|V|)$

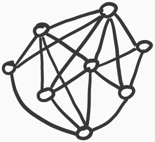
Property Testing: in a Nutshell



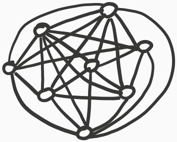
planar ✓



slightly non-planar ✓/✗



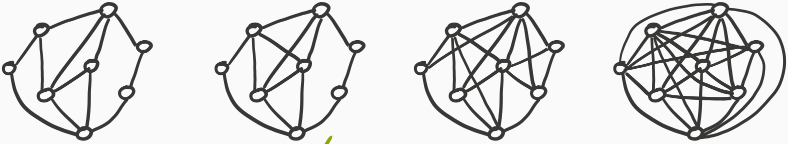
quite non-planar ✗



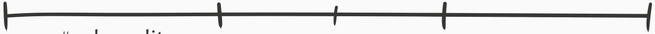
very non-planar ✗

time complexity: $\Omega(|V|)$

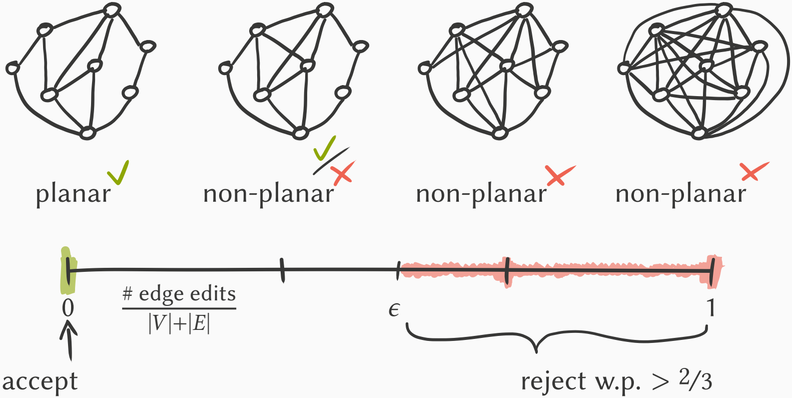
Property Testing: in a Nutshell



planar ✓ non-planar ~~✓~~ ✗ non-planar ✗ non-planar ✗



Property Testing: in a Nutshell



planar ✓

non-planar ~~✓~~ ✗

non-planar ✗

non-planar ✗

0

$\frac{\text{\# edge edits}}{|V|+|E|}$

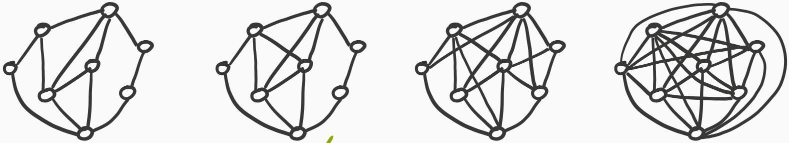
ϵ

1

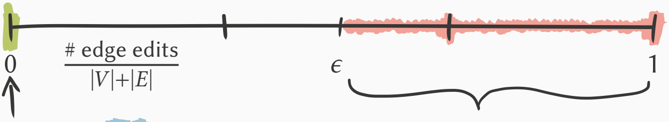
accept

reject w.p. > ²⁄₃

Property Testing: in a Nutshell



planar ✓ | non-planar ~~✓~~ ✗ | non-planar ✗ | non-planar ✗



0

$\frac{\text{\# edge edits}}{|V|+|E|}$

ϵ

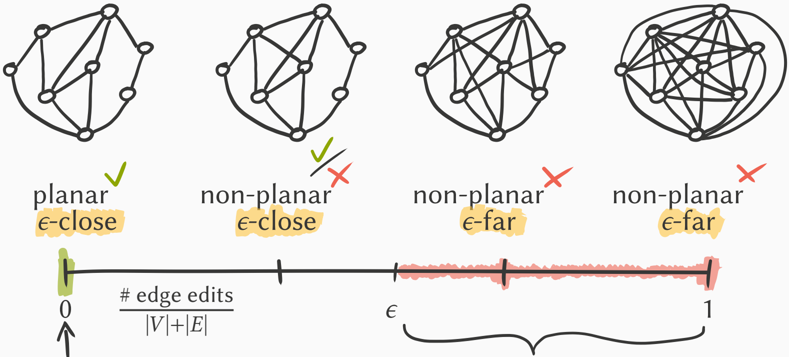
1

accept w.p. $>$ $2/3$

reject w.p. $>$ $2/3$

1-sided error / 2-sided error

Property Testing: in a Nutshell



planar ✓
ϵ-close

non-planar ~~✓~~ ✗
ϵ-close

non-planar ✗
ϵ-far

non-planar ✗
ϵ-far

$\frac{\#\text{ edge edits}}{|V|+|E|}$

0

ϵ

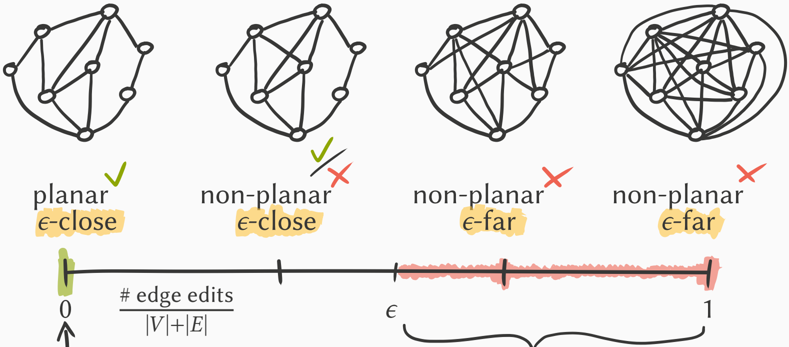
1

accept w.p. > $^2/_3$

reject w.p. > $^2/_3$

1-sided error / 2-sided error

Property Testing: in a Nutshell



planar ✓

ϵ-close

non-planar ~~✓~~ ✗

ϵ-close

non-planar ✗

ϵ-far

non-planar ✗

ϵ-far

0 $\qquad \frac{\#\text{ edge edits}}{|V|+|E|} \qquad \epsilon \qquad 1$

accept w.p. $> {}^2\!/_3$

reject w.p. $> {}^2\!/_3$

1-sided error / 2-sided error

complexity: # queries to data structure

The Bounded-Degree Model

☒ bounded-degree model: $\forall v \in V : d(v) \leq d, d \in O(1), n := |V|$

☒ input structure: adjacency lists (1 query $\hat{=}$ 1 entry)

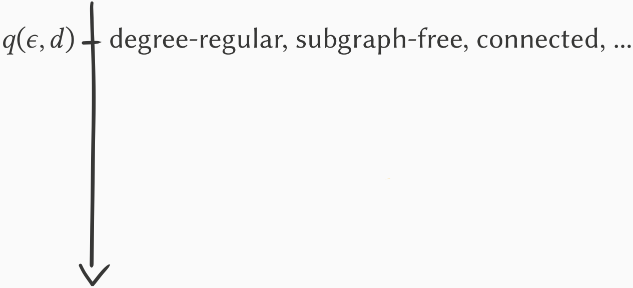
☒ error: 1-sided

The Bounded-Degree Model

☒ bounded-degree model: $\forall v \in V \,:\, d(v) \leq d, d \in O(1), n \,:= |V|$

☒ input structure: adjacency lists (1 query $\hat{=}$ 1 entry)

☒ error: 1-sided



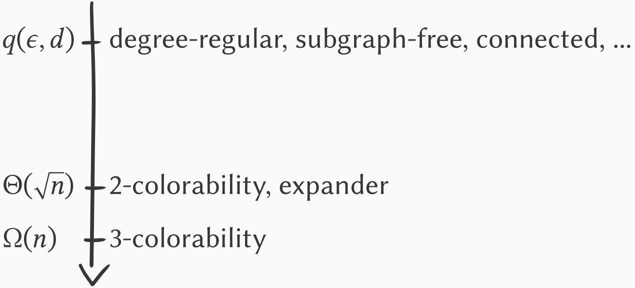
$q(\epsilon, d)$ degree-regular, subgraph-free, connected, ...

The Bounded-Degree Model

☒ bounded-degree model: $\forall v \in V : d(v) \leq d, d \in O(1), n := |V|$

☒ input structure: adjacency lists (1 query $\hat{=}$ 1 entry)

☒ error: 1-sided



$q(\epsilon, d)$ — degree-regular, subgraph-free, connected, ...

$\Theta(\sqrt{n})$ — 2-colorability, expander

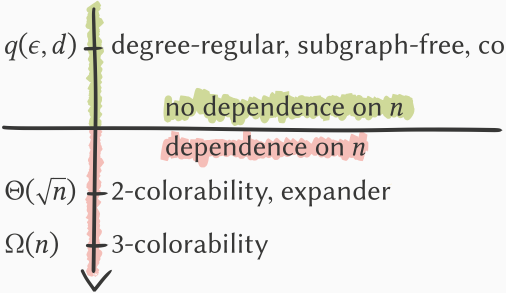
$\Omega(n)$ — 3-colorability

The Bounded-Degree Model

☒ bounded-degree model: $\forall v \in V\ :\ d(v) \leq d, d \in O(1), n\ := |V|$

☒ input structure: adjacency lists (1 query $\hat{=}$ 1 entry)

☒ error: 1-sided



$q(\epsilon, d)$ degree-regular, subgraph-free, connected, ...

no dependence on n

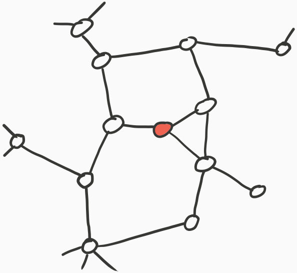
dependence on n

$\Theta(\sqrt{n})$ 2-colorability, expander

$\Omega(n)$ 3-colorability

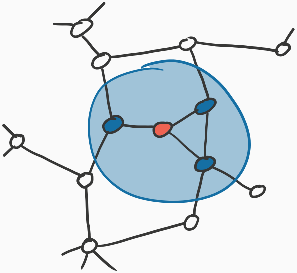
q-Disks / q-Hop Neighborhoods

$\mathrm{disk}_q(v)$: unlabelled subgraph induced by $\mathrm{BFS}(v)$ of depth q



q-Disks / q-Hop Neighborhoods

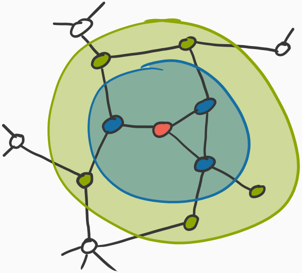
$\text{disk}_q(v)$: unlabelled subgraph induced by $\text{BFS}(v)$ of depth q



disk_1(●)

q-Disks / q-Hop Neighborhoods

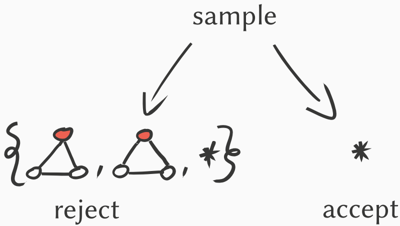
$\text{disk}_q(v)$: unlabelled subgraph induced by $\text{BFS}(v)$ of depth q



$\text{disk}_1($●$)$

$\text{disk}_2($●$)$

Constant-Query Testers



sample

reject

accept

Theorem [GR’09, ...]

A property tester for bounded-degree graphs with constant query complexity $q := q(\epsilon)$ can be transformed into an algorithm that

1. obtains a uniform sample S of $\Theta(q)$ many q-disks
2. rejects iff S is from a family of forbidden sets of q-disks

Goldreich, Ron, STOC’09

The Random-Neighbor Model

☒ bounded-degree model
☒ input structure: adjacency lists
☒ error: 1-sided

The Random-Neighbor Model

- ☒ ~~bounded-degree model~~ general graphs
- ☒ input structure: adjacency lists
- ☒ error: 1-sided

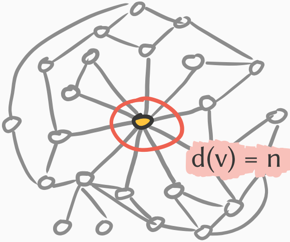
The Random-Neighbor Model

- ☒ ~~bounded-degree model~~ general graphs
- ☒ input structure: adjacency lists
- ☒ error: 1-sided

What can a constant-query
property tester do?

The Random-Neighbor Model

- ☒ ~~bounded-degree model~~ general graphs
- ☒ input structure: adjacency lists
- ☒ error: 1-sided



What can a constant-query property tester do?

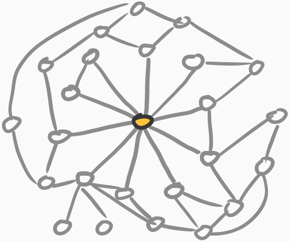
BFS

The Random-Neighbor Model

☒ ~~bounded-degree model~~ general graphs

☒ input structure: adjacency lists

☒ error: 1-sided



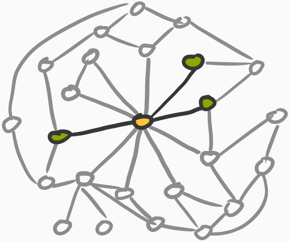
What can a constant-query property tester do?

B~~F~~S

random / subsampling BFS

The Random-Neighbor Model

- ☒ ~~bounded-degree model~~ general graphs
- ☒ input structure: adjacency lists
- ☒ error: 1-sided



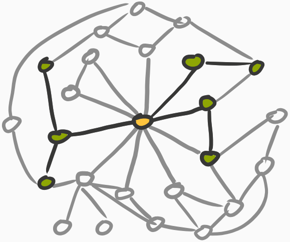
What can a constant-query property tester do?

~~BFS~~

random / subsampling BFS

The Random-Neighbor Model

☒ ~~bounded-degree model~~ general graphs
☒ input structure: adjacency lists
☒ error: 1-sided



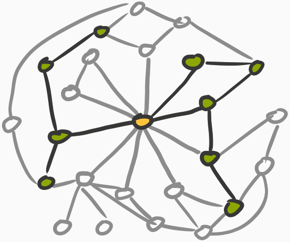
What can a constant-query property tester do?

~~BFS~~

random / subsampling BFS

The Random-Neighbor Model

- ☒ ~~bounded-degree model~~ general graphs
- ☒ input structure: adjacency lists
- ☒ error: 1-sided



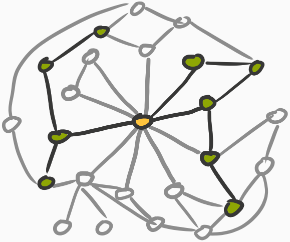
What can a constant-query property tester do?

~~BFS~~

random / subsampling BFS

The Random-Neighbor Model

- ☒ ~~bounded-degree model~~ general graphs
- ☒ input structure: adjacency lists
- ☒ error: 1-sided



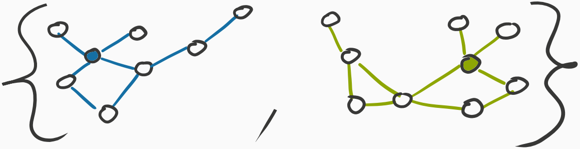
What can a constant-query property tester do?

B~~F~~S

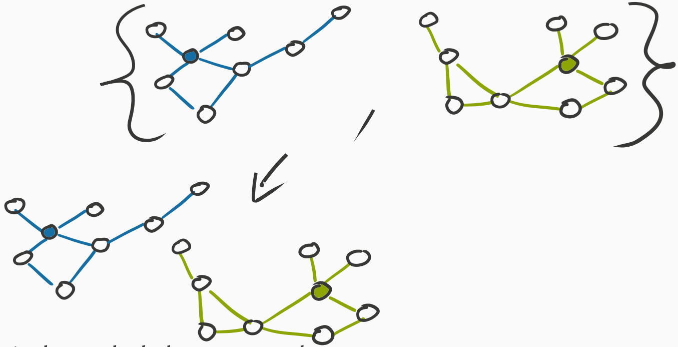
random / subsampling BFS

tester obtains at most q many q-disks (with bounded degree q)

Bounded Degree vs. General Graphs

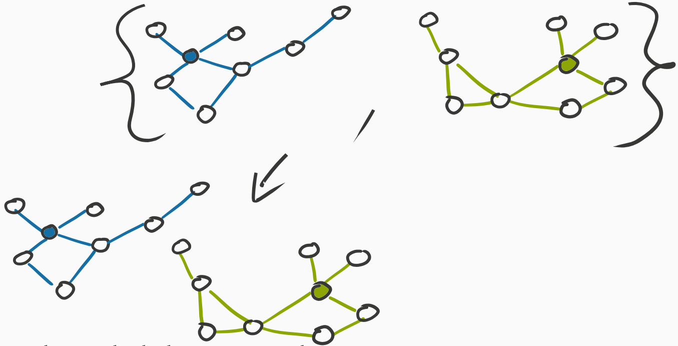


Bounded Degree vs. General Graphs



in bounded-degree graphs:
random q-disks disjoint w.h.p.

Bounded Degree vs. General Graphs

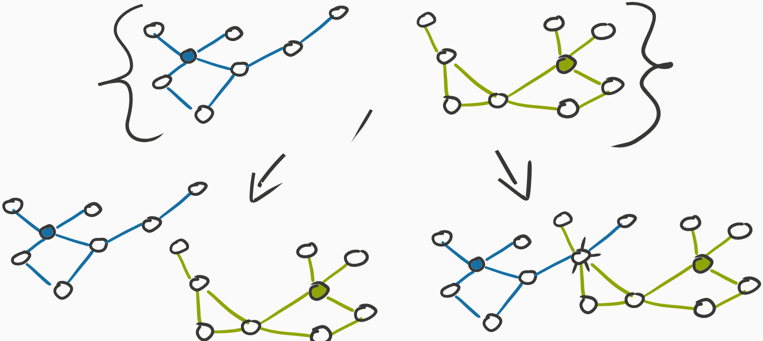


in bounded-degree graphs:
random q-disks disjoint w.h.p.

↓

bijective decomposition ✓
of BFS subgraphs into q-disks

Bounded Degree vs. General Graphs



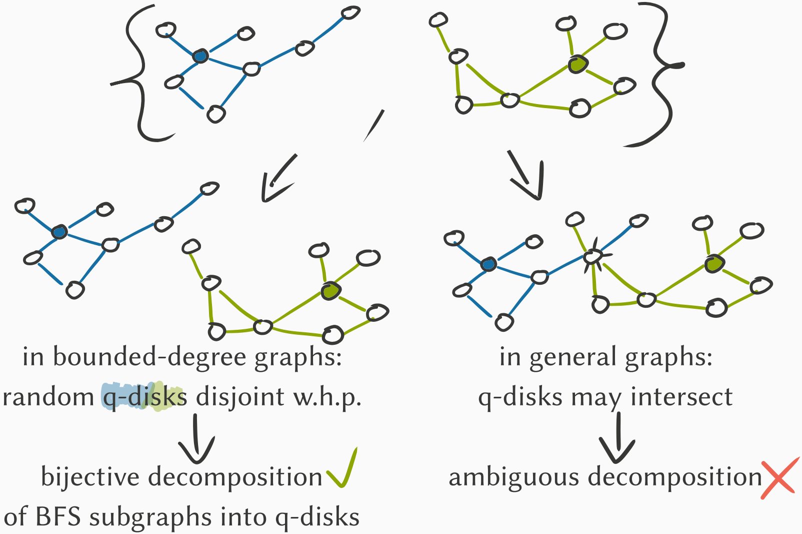
in bounded-degree graphs:
random q-disks disjoint w.h.p.

↓

bijective decomposition ✓
of BFS subgraphs into q-disks

in general graphs:
q-disks may intersect

Bounded Degree vs. General Graphs



in bounded-degree graphs:
random q-disks disjoint w.h.p.

↓

bijective decomposition ✓
of BFS subgraphs into q-disks

in general graphs:
q-disks may intersect

↓

ambiguous decomposition ✗

The Problem of Intersecting q-Disks

problem: two random BFS visit the same vertex

The Problem of Intersecting q-Disks

problem: two random BFS visit the same vertex

only likely for linear degree vertices

The Problem of Intersecting q-Disks

problem: two random BFS visit the same vertex

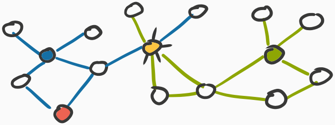
only likely for $\underbrace{\text{linear degree vertices}}_{O(1)\text{ many}}$

The Problem of Intersecting q-Disks

problem: two random BFS visit the same vertex

only likely for $\underbrace{\text{linear degree vertices}}_{O(1)\text{ many}}$

solution: assign unique color to high degree vertices

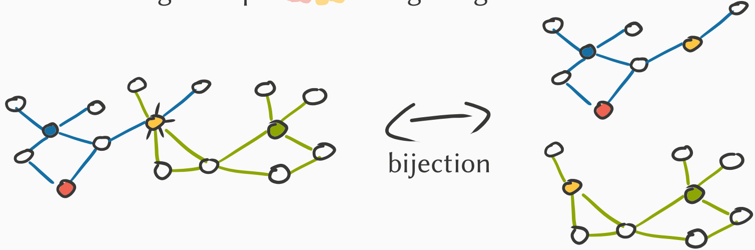


The Problem of Intersecting q-Disks

problem: two random BFS visit the same vertex

only likely for $\underbrace{\text{linear degree vertices}}_{O(1)\text{ many}}$

solution: assign unique color to high degree vertices



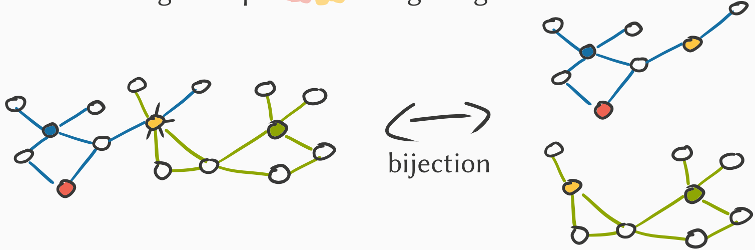
bijection

The Problem of Intersecting q-Disks

problem: two random BFS visit the same vertex

only likely for $\underbrace{\text{linear degree vertices}}_{O(1)\text{ many}}$

solution: assign unique color to high degree vertices



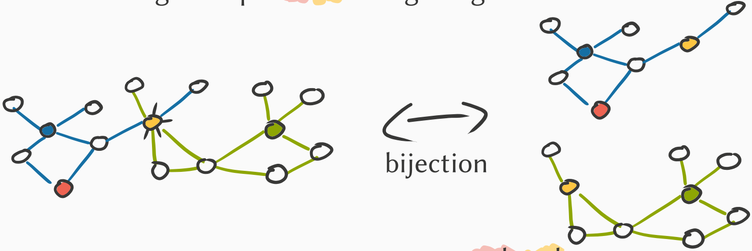
tester: rejects iff it finds a forbidden set of q-disks

The Problem of Intersecting q-Disks

problem: two random BFS visit the same vertex

only likely for $\underbrace{\text{linear degree vertices}}_{O(1)\text{ many}}$

solution: assign unique color to high degree vertices



tester: rejects iff it finds a forbidden set of $\overbrace{\text{q-disks}}^{\text{colored}}$

Constant-Query Testers in the Random-Neighbor Model

Theorem

A property tester *for general graphs in the random-neighbor model* with constant query complexity $q := q(\epsilon)$ can be transformed into an algorithm that

1. obtains a uniform sample S of colored q-disks by performing $\Theta(q)$ random BFS (the number of colors is $O(1)$)
2. rejects iff S is from a family of forbidden sets of colored q-disks

Constant-Query Testers in the Random-Neighbor Model

Theorem

A property tester *for general graphs in the random-neighbor model* with constant query complexity $q := q(\epsilon)$ can be transformed into an algorithm that

1. obtains a uniform sample S of colored q-disks by performing $\Theta(q)$ random BFS (the number of colors is $O(1)$)
2. rejects iff S is from a family of forbidden sets of colored q-disks

why q-disks and not simply forbidden *subgraphs*?

The Streaming Model

- ☒ general graphs
- ☒ input structure: adjacency lists
- ☒ error: 1-sided

Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP'17; Peng, Sohler, SODA'18

The Streaming Model

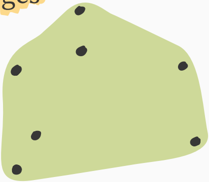
- ☒ general graphs
- ☒ input structure: ~~adjacency lists~~ stream of edges
- ☒ error: 1-sided

Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP'17; Peng, Sohler, SODA'18

The Streaming Model

- ☒ general graphs
- ☒ input structure: ~~adjacency lists~~ stream of edges
- ☒ error: 1-sided

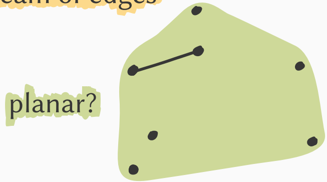
planar?



Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP’17; Peng, Sohler, SODA’18

The Streaming Model

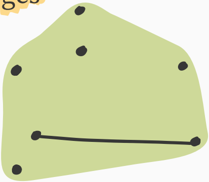
- ☒ general graphs
- ☒ input structure: ~~adjacency lists~~ stream of edges
- ☒ error: 1-sided



Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP'17; Peng, Sohler, SODA'18

The Streaming Model

- ☒ general graphs
- ☒ input structure: ~~adjacency lists~~ stream of edges
- ☒ error: 1-sided



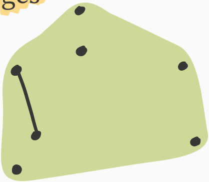
planar?

Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP’17; Peng, Sohler, SODA’18

The Streaming Model

- ☒ general graphs
- ☒ input structure: ~~adjacency lists~~ stream of edges
- ☒ error: 1-sided

planar?

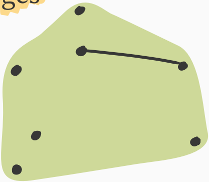


Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP’17; Peng, Sohler, SODA’18

The Streaming Model

- ☒ general graphs
- ☒ input structure: ~~adjacency lists~~ stream of edges
- ☒ error: 1-sided

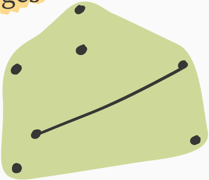
planar?



Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP'17; Peng, Sohler, SODA'18

The Streaming Model

- ☒ general graphs
- ☒ input structure: ~~adjacency lists~~ stream of edges
- ☒ error: 1-sided



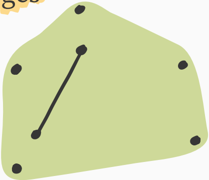
planar?

Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP'17; Peng, Sohler, SODA'18

The Streaming Model

- ☒ general graphs
- ☒ input structure: ~~adjacency lists~~ stream of edges
- ☒ error: 1-sided

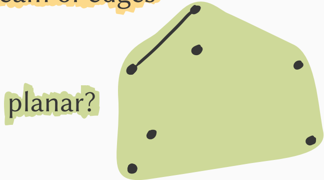
planar?



Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP’17; Peng, Sohler, SODA’18

The Streaming Model

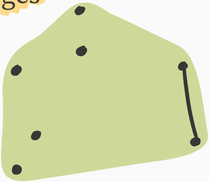
- ☒ general graphs
- ☒ input structure: ~~adjacency lists~~ stream of edges
- ☒ error: 1-sided



Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP'17; Peng, Sohler, SODA'18

The Streaming Model

- ☒ general graphs
- ☒ input structure: ~~adjacency lists~~ stream of edges
- ☒ error: 1-sided



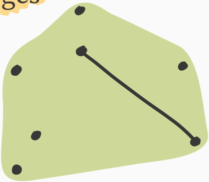
planar?

Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP'17; Peng, Sohler, SODA'18

The Streaming Model

- ☒ general graphs
- ☒ input structure: ~~adjacency lists~~ stream of edges
- ☒ error: 1-sided

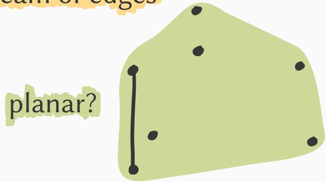
planar?



Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP'17; Peng, Sohler, SODA'18

The Streaming Model

- ☒ general graphs
- ☒ input structure: ~~adjacency lists~~ stream of edges
- ☒ error: 1-sided

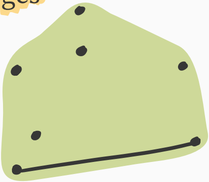


Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP’17; Peng, Sohler, SODA’18

The Streaming Model

- ☒ general graphs
- ☒ input structure: ~~adjacency lists~~ stream of edges
- ☒ error: 1-sided

planar?



Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP'17; Peng, Sohler, SODA'18

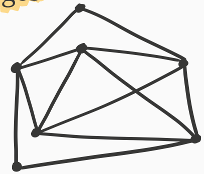
The Streaming Model

- ☒ general graphs
- ☒ input structure: ~~adjacency lists~~ stream of edges
- ☒ error: 1-sided

Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP'17; Peng, Sohler, SODA'18

The Streaming Model

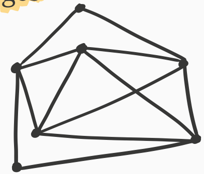
- ☒ general graphs
- ☒ input structure: ~~adjacency lists~~ stream of edges
- ☒ error: 1-sided



Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP'17; Peng, Sohler, SODA'18

The Streaming Model

- ☒ general graphs
- ☒ input structure: ~~adjacency lists~~ stream of edges
- ☒ error: 1-sided



objective: $o(n)$ space

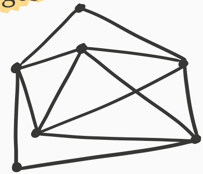
Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP’17; Peng, Sohler, SODA’18

The Streaming Model

☒ general graphs

☒ input structure: ~~adjacency lists~~ stream of edges

☒ error: 1-sided



objective: $o(n)$ space

- some problems $\Omega(n)$ in adversarial-order streams

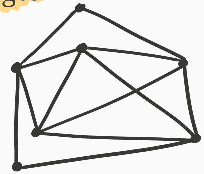
Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP'17; Peng, Sohler, SODA'18

The Streaming Model

☒ general graphs

☒ input structure: ~~adjacency lists~~ stream of edges

☒ error: 1-sided



objective: $o(n)$ space

- some problems $\Omega(n)$ in adversarial-order streams
- trivial if number of edges is $O(n)$

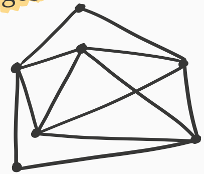
Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP’17; Peng, Sohler, SODA’18

The Streaming Model

☒ general graphs

☒ input structure: ~~adjacency lists~~ stream of edges

☒ error: 1-sided



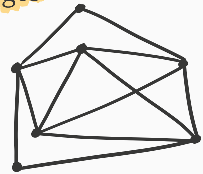
objective: $o(n)$ space

- some problems $\Omega(n)$ in adversarial-order streams
- trivial if number of edges is $O(n)$
- recent model: random-order streams

Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP'17; Peng, Sohler, SODA'18

The Streaming Model

☒ general graphs
☒ input structure: ~~adjacency lists~~ stream of edges
☒ error: 1-sided



objective: $o(n)$ space

- some problems $\Omega(n)$ in adversarial-order streams
- trivial if number of edges is $O(n)$
- recent model: random-order streams

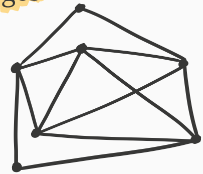
already known:

- estimate several graph parameters in general graphs [PS'18]

Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP'17; Peng, Sohler, SODA'18

The Streaming Model

☒ general graphs
☒ input structure: ~~adjacency lists~~ stream of edges
☒ error: 1-sided



objective: $o(n)$ space

- some problems $\Omega(n)$ in adversarial-order streams
- trivial if number of edges is $O(n)$
- recent model: random-order streams

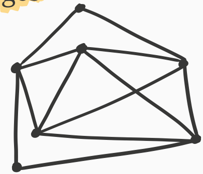
already known:

- estimate several graph parameters in general graphs [PS'18]
- transform constant-query testers for bounded-degree graphs to $O(\log n)$-space streaming testers [MMPS'17]

Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP'17; Peng, Sohler, SODA'18

The Streaming Model

☒ general graphs
☒ input structure: ~~adjacency lists~~ stream of edges
☒ error: 1-sided



objective: $o(n)$ space

- some problems $\Omega(n)$ in adversarial-order streams
- trivial if number of edges is $O(n)$
- recent model: random-order streams

already known:

- estimate several graph parameters in general graphs [PS'18]
- transform constant-query testers for bounded-degree graphs to $O(\log n)$-space streaming testers [MMPS'17]

Monemizadeh, Muthukrishnan, Peng, Sohler, ICALP'17; Peng, Sohler, SODA'18

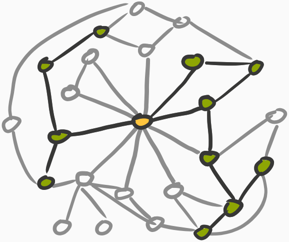
Streaming Tester For General Graphs

idea: sample $\Theta(q)$ vertices and simulate BFS in stream

Streaming Tester For General Graphs

idea: sample $\Theta(q)$ vertices and simulate BFS in stream

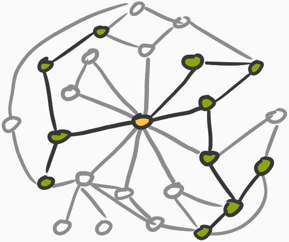
query tester's BFS



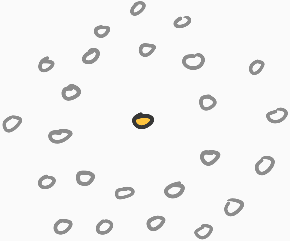
Streaming Tester For General Graphs

idea: sample $\Theta(q)$ vertices and simulate BFS in stream

query tester's BFS



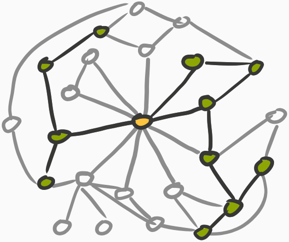
streaming tester's BFS



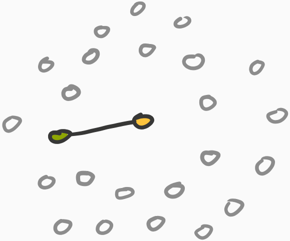
Streaming Tester For General Graphs

idea: sample $\Theta(q)$ vertices and simulate BFS in stream

query tester's BFS



streaming tester's BFS

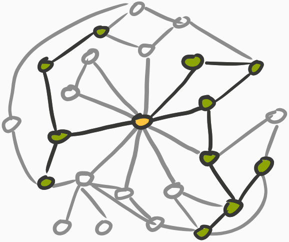


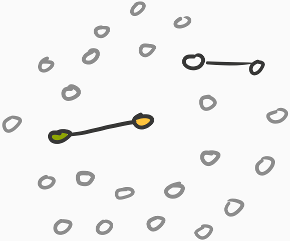
Streaming Tester For General Graphs

idea: sample $\Theta(q)$ vertices and simulate BFS in stream

query tester's BFS

streaming tester's BFS

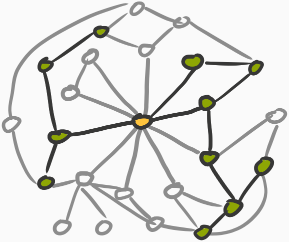




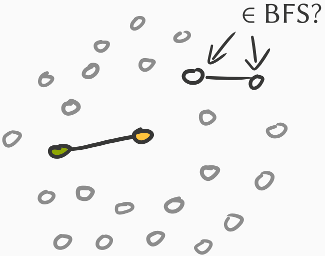
Streaming Tester For General Graphs

idea: sample $\Theta(q)$ vertices and simulate BFS in stream

query tester's BFS



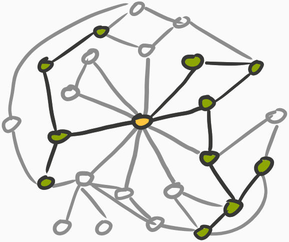
streaming tester's BFS



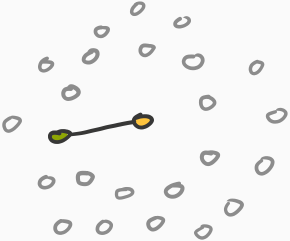
Streaming Tester For General Graphs

idea: sample $\Theta(q)$ vertices and simulate BFS in stream

query tester’s BFS



streaming tester’s BFS

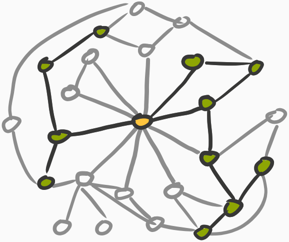


condition on: edges are streamed in a *good* order

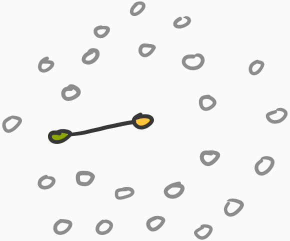
Streaming Tester For General Graphs

idea: sample $\Theta(q)$ vertices and simulate BFS in stream

query tester's BFS



streaming tester's BFS



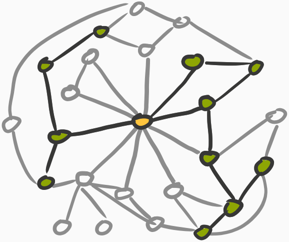
condition on: edges are streamed in a *good* order

can bound probability by $\Omega(1)$

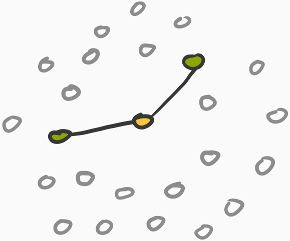
Streaming Tester For General Graphs

idea: sample $\Theta(q)$ vertices and simulate BFS in stream

query tester's BFS



streaming tester's BFS



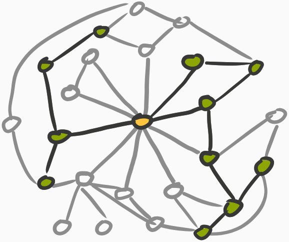
condition on: edges are streamed in a *good* order

can bound probability by $\Omega(1)$

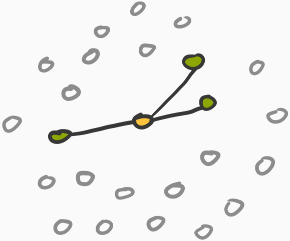
Streaming Tester For General Graphs

idea: sample $\Theta(q)$ vertices and simulate BFS in stream

query tester’s BFS



streaming tester’s BFS



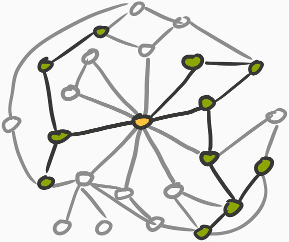
condition on: edges are streamed in a *good* order

↖ can bound probability by $\Omega(1)$

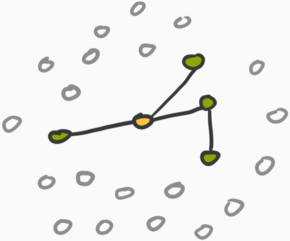
Streaming Tester For General Graphs

idea: sample $\Theta(q)$ vertices and simulate BFS in stream

query tester's BFS



streaming tester's BFS



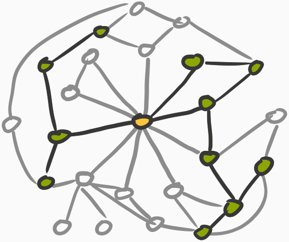
condition on: edges are streamed in a *good* order

can bound probability by $\Omega(1)$

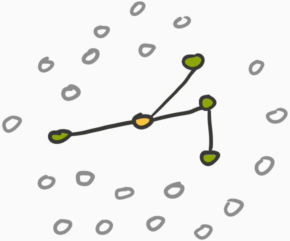
Streaming Tester For General Graphs

idea: sample $\Theta(q)$ vertices and simulate BFS in stream

query tester's BFS



streaming tester's BFS



condition on: edges are streamed in a *good* order

can bound probability by $\Omega(1)$

then: bound probabilities to see colored q-disks independently

From Constant-Query to Streaming Testers

Theorem

Every constant-query property tester for general graphs in the random-neighbor model with one-sided error and constant query complexity admits a $O(\log n)$ space random order streaming tester.

From Constant-Query to Streaming Testers

Theorem

Every constant-query property tester for general graphs in the random-neighbor model with one-sided error and constant query complexity admits a $O(\log n)$ space random order streaming tester.

open problem: similar result for testers with two-sided error