Local access to Huge Random Objects

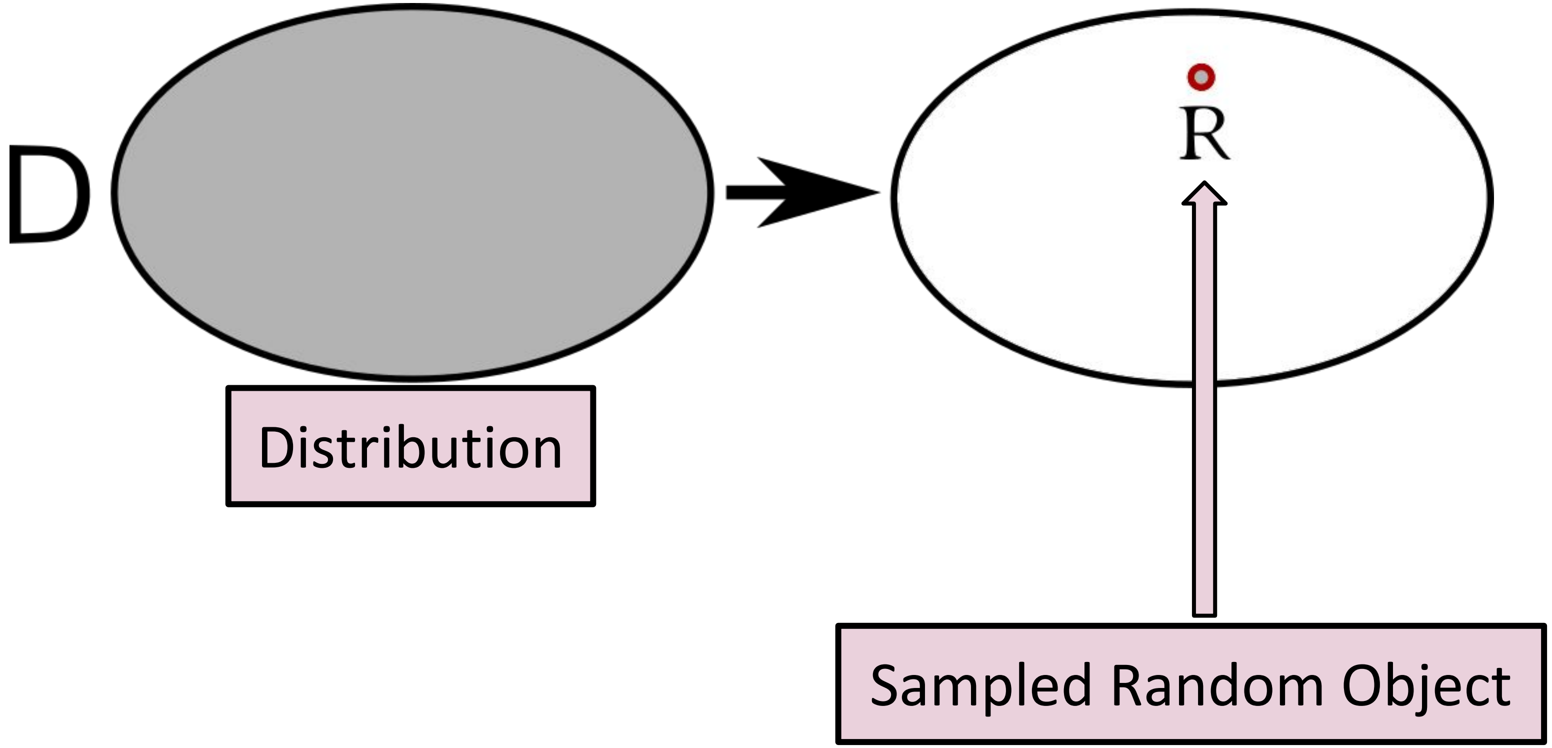
Amartya Shankha Biswas (MIT)

Ronitt Rubinfeld (MIT and TAU)

Anak Yodpinyanee (MIT)

Generating Huge Random Objects

Up front



D

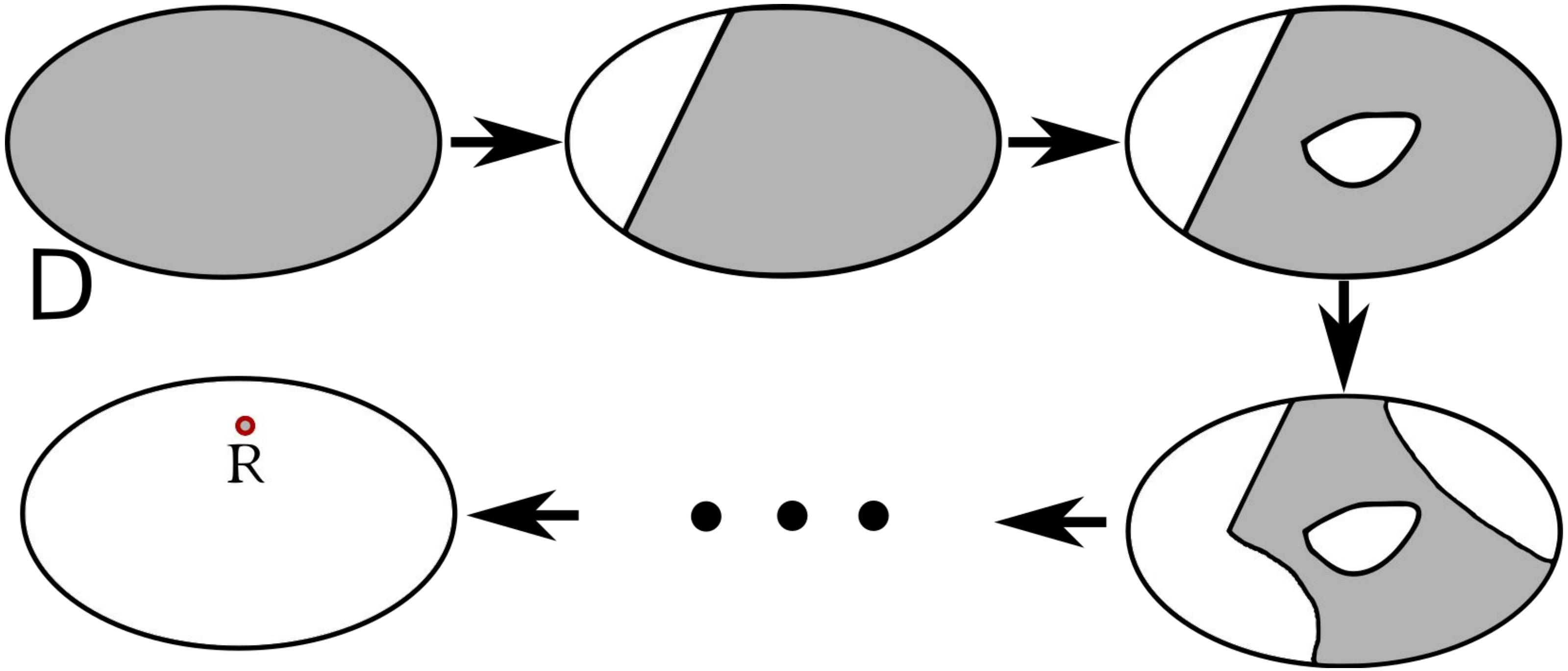
R

Distribution

Sampled Random Object

Partial Sampling

“As needed” (local access queries)

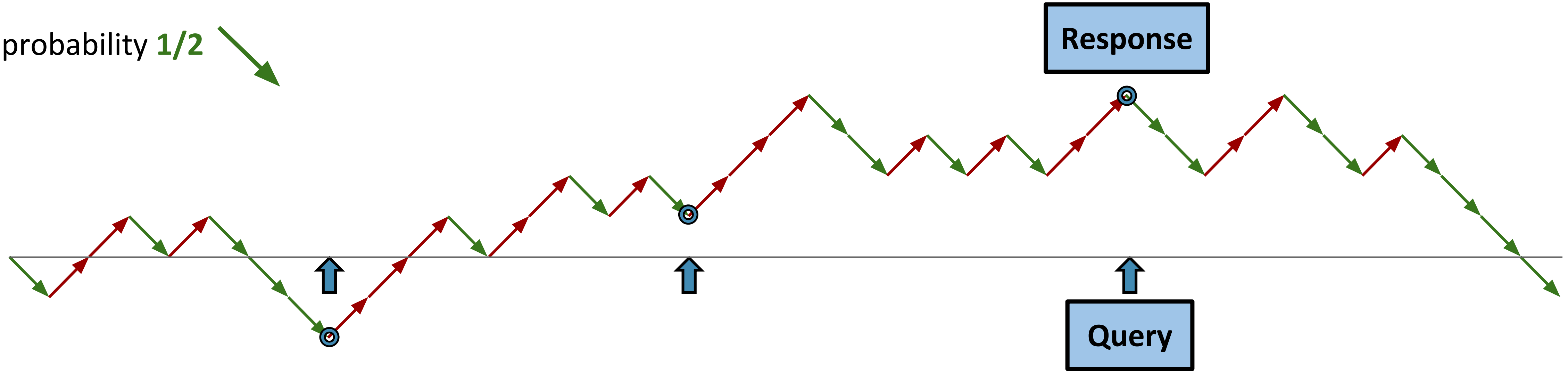


Local Access to 1D Random Walk (on the line)

Query Height(t) returns position of walk at time t

with probability **1/2** ↗

with probability **1/2** ↘



Queries appear in arbitrary order

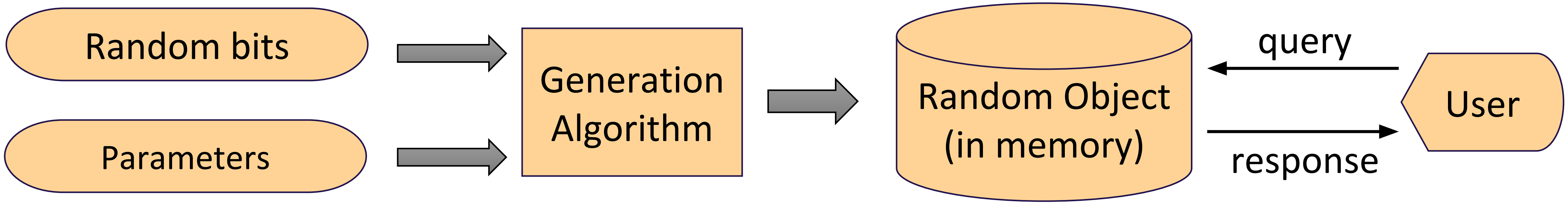
Random graph: Adjacency Matrix

	1	2	3	4	5	6	7	8	9	10
1			0		0		0			
2			0				0	1		
3	0	0		0	0	0	0	0	1	
4			0				0			
5	0		0			1	1			
6			0		1					
7	0	0	0	0	1					
8		1	0							
9			1							
10										

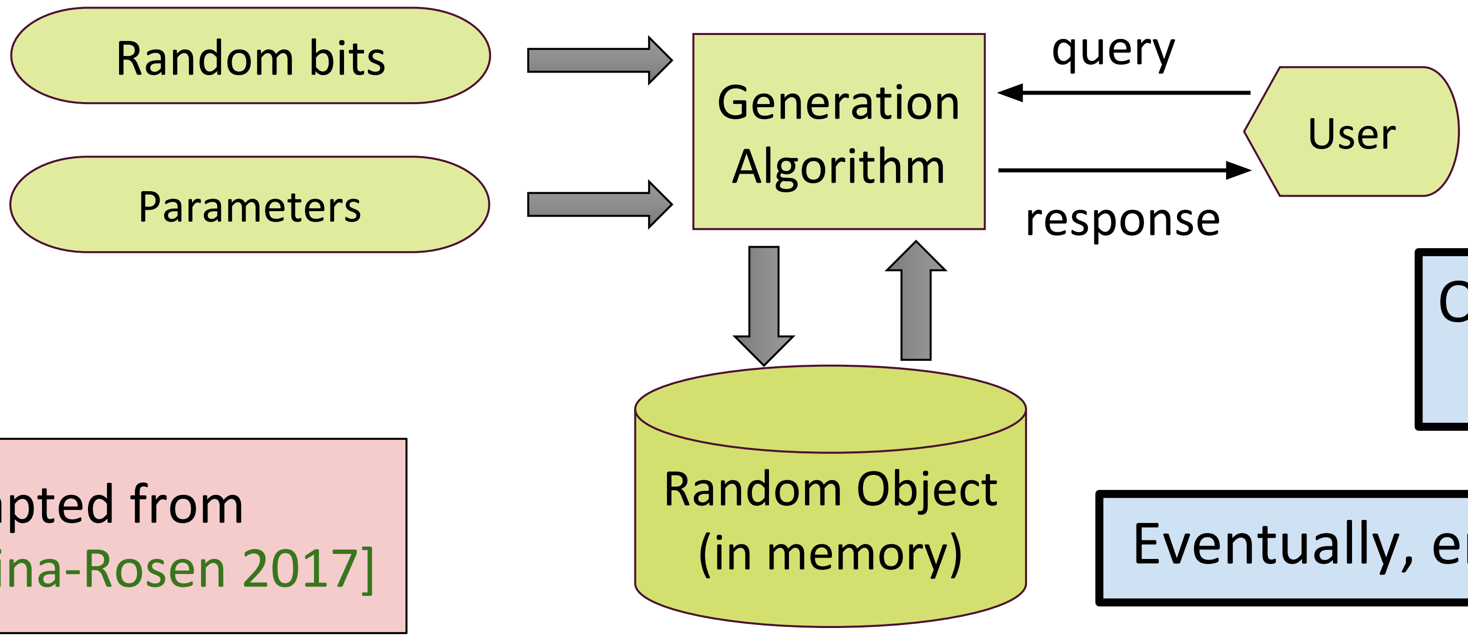
Generate
“on the fly”

Amortize Sampling over Queries

Standard paradigm



“on-the-fly” sampler



Queries reveal partial information

Eventually, entire object is sampled

Figure Adapted from [Even-Levi-Medina-Rosen 2017]

Query Requirements

- **Efficiency**: Use **polylogarithmic**
  - **Time**
  - **Space**
  - **Random Bits**

No pre-processing!

- All responses **consistent** with **single valid sample**

- **Output distribution** ε-close to the **true distribution** (ℓ_1 distance)

Example Queries in Erdos-Renyi Graphs: $G(n, p)$

Every edge exists with probability p (independently)

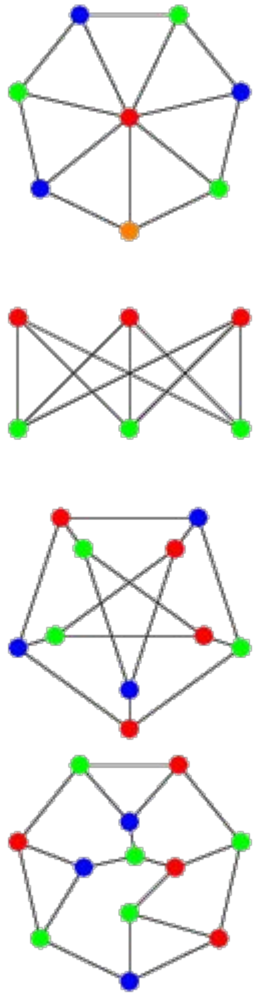
- Vertex-Pair(u, v): Is edge (u, v) present?
- All-Neighbors(v): Return full neighborhood of v
- Degree(v) $\longrightarrow$ **OPEN**
- Next-neighbor(v): **Lexicographically** next neighbor
- Random-neighbor(v): Return random neighbor of v

A harder setting

[B-Rubinfeld-Yodpinyanee]

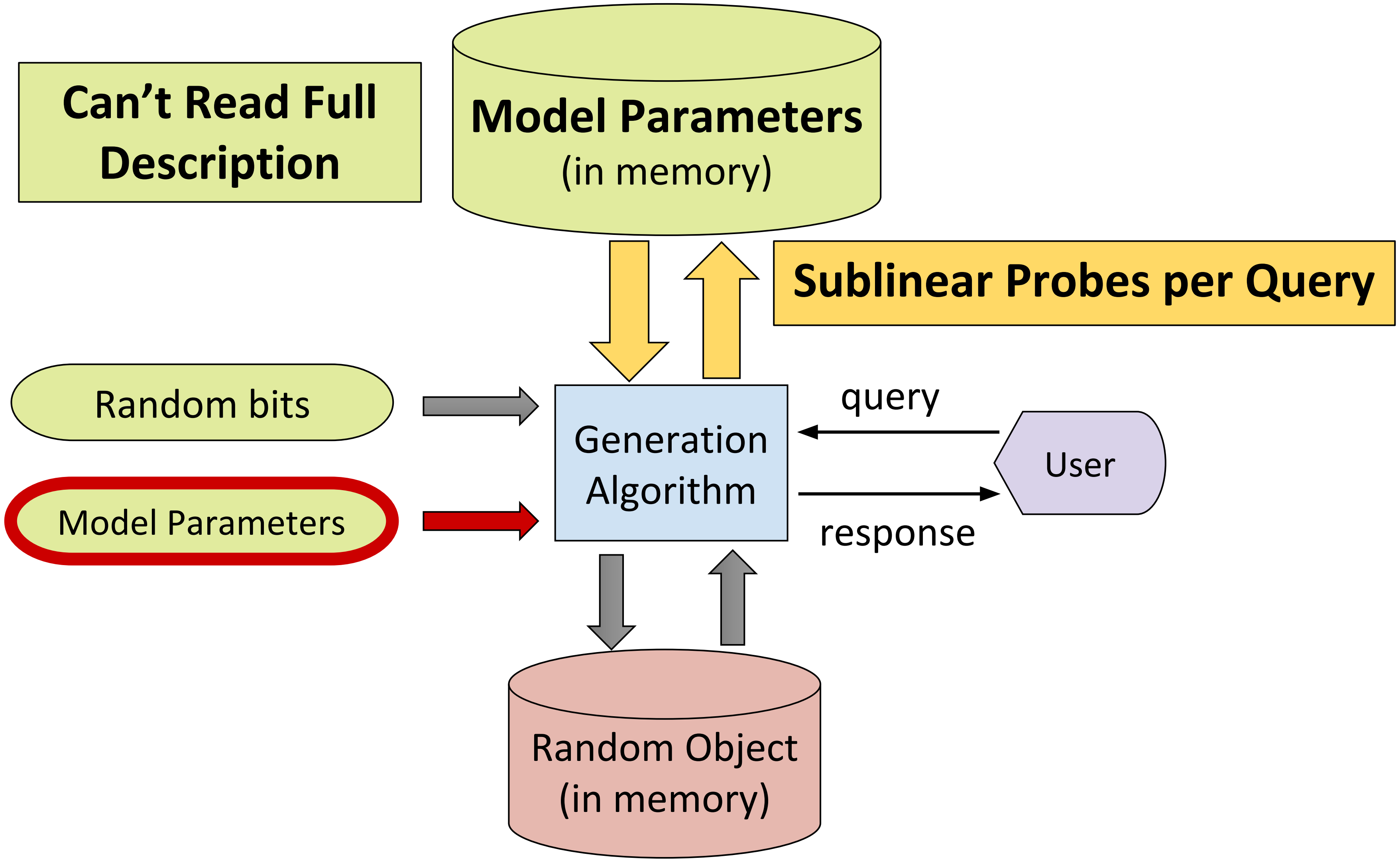
Random (Valid) Coloring of a Graph

- **Input Graph:** G
  - Maximum Degree: Δ
  - Number of colors: $q > \Delta$
- **Output:** Random Valid Coloring of G
  - **Uniform over all valid colorings**
- **Query: Color of single vertex** in **sublinear time**



- All responses must be consistent
- Overall coloring sampled from uniform distribution

Random Objects with **Huge Description Size**



Can't Read Full Description

Model Parameters
(in memory)

Sublinear Probes per Query

Random bits

Model Parameters

Generation Algorithm

query

response

User

Random Object
(in memory)

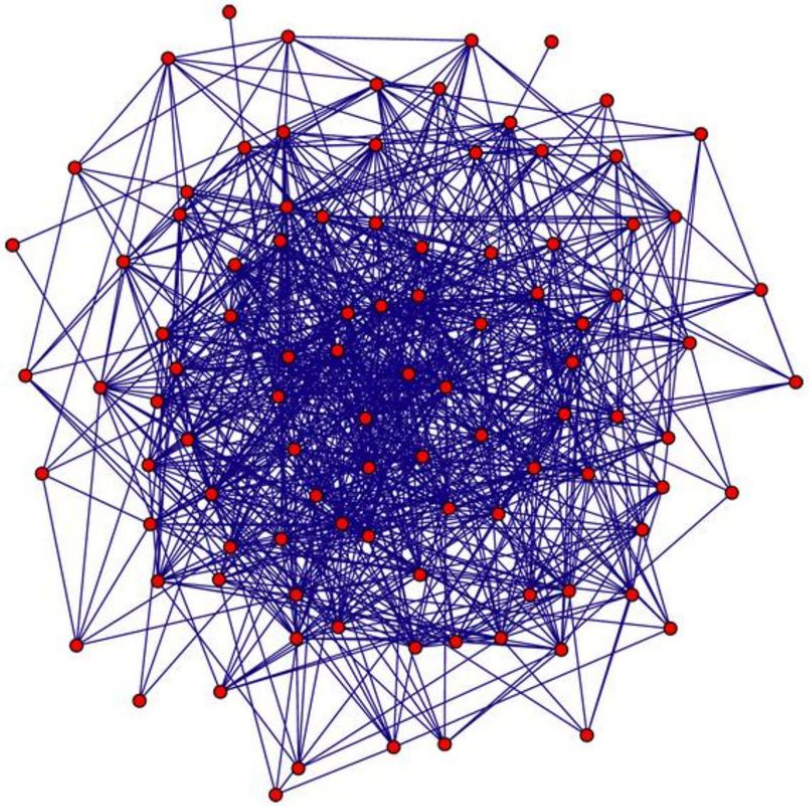
Prior work

Local Access Model from [Goldreich-Goldwasser-Nussboim 03]

- Generators for **huge** random functions, codes, graphs, …
- Important **primitives**
  - Sampling from binomial distribution
  - Interval-sum queries on **random binary strings**
    (see also [Gilbert-Guha-Indyk-Kotidis-Muthukrishnan-Strauss 02])
- Random graphs with **specified property**
  - e.g. Planted clique or Hamiltonian cycle
  - Focus on **indistinguishable** (under small number of queries and poly time)

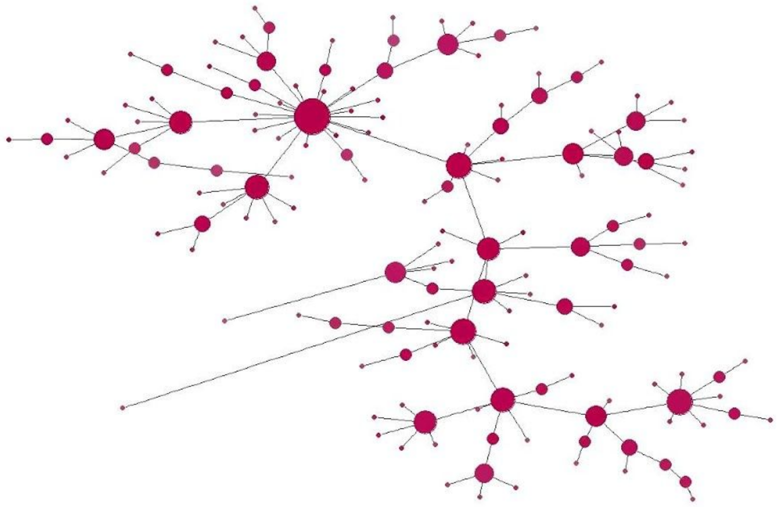
Sparse $G(n, p)$ graph [Naor-Nussboim 2007]

- Degree at most **polylog**
- Queries:
  - **Vertex-Pair**
  - **All-Neighbors**



Implementations of Barabasi-Albert Preferential Attachment Graphs [Even-Levi-Medina-Rosen 2017]

- Graphs generated:
  - Rooted tree/forest structure
  - **Highly sequential** random process
  - Sparse, but **unbounded degree**
- Queries **(no bound on number)**:
  - **Vertex-Pair**
  - **Next-Neighbor**
    (Lexicographically in Adjacency List)



Summary of our Results

Erdos-Renyi $G(n, p)$

- Support all values of p
- Vertex-Pair
- Next-Neighbor
- **Random-Neighbor**

Application:
Random walk in large degree graph!

Today's Talk

Other Random Objects

- General graphs with **Independent edge probabilities** (under mild assumptions)
- 1D random walks
- Random Catalan objects
- (Simple) Domino Tilings

Unbounded Queries

Polylog time space and random bits

Generated objects are **truly random** (not just indistinguishable)

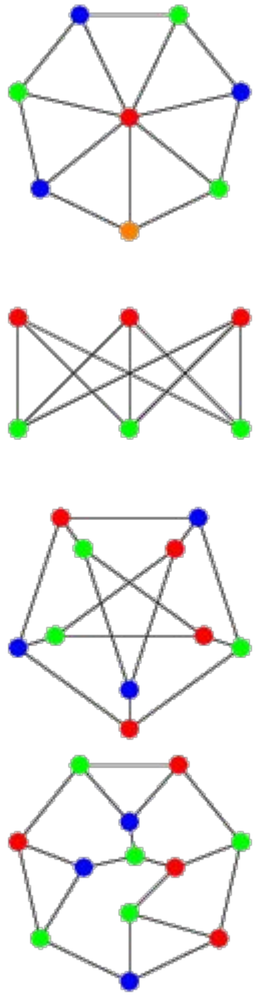
Random (Valid) Coloring of a Graph

- For $q > 9\Delta$
- **Unbounded** Queries
- Query color of specified vertex in **sublinear time**
    - **Not polylog**
- **Memoryless**

Local Computation Algorithms [Rubinfeld, Tamir, Vardi, Xie]
with **specific output distribution**

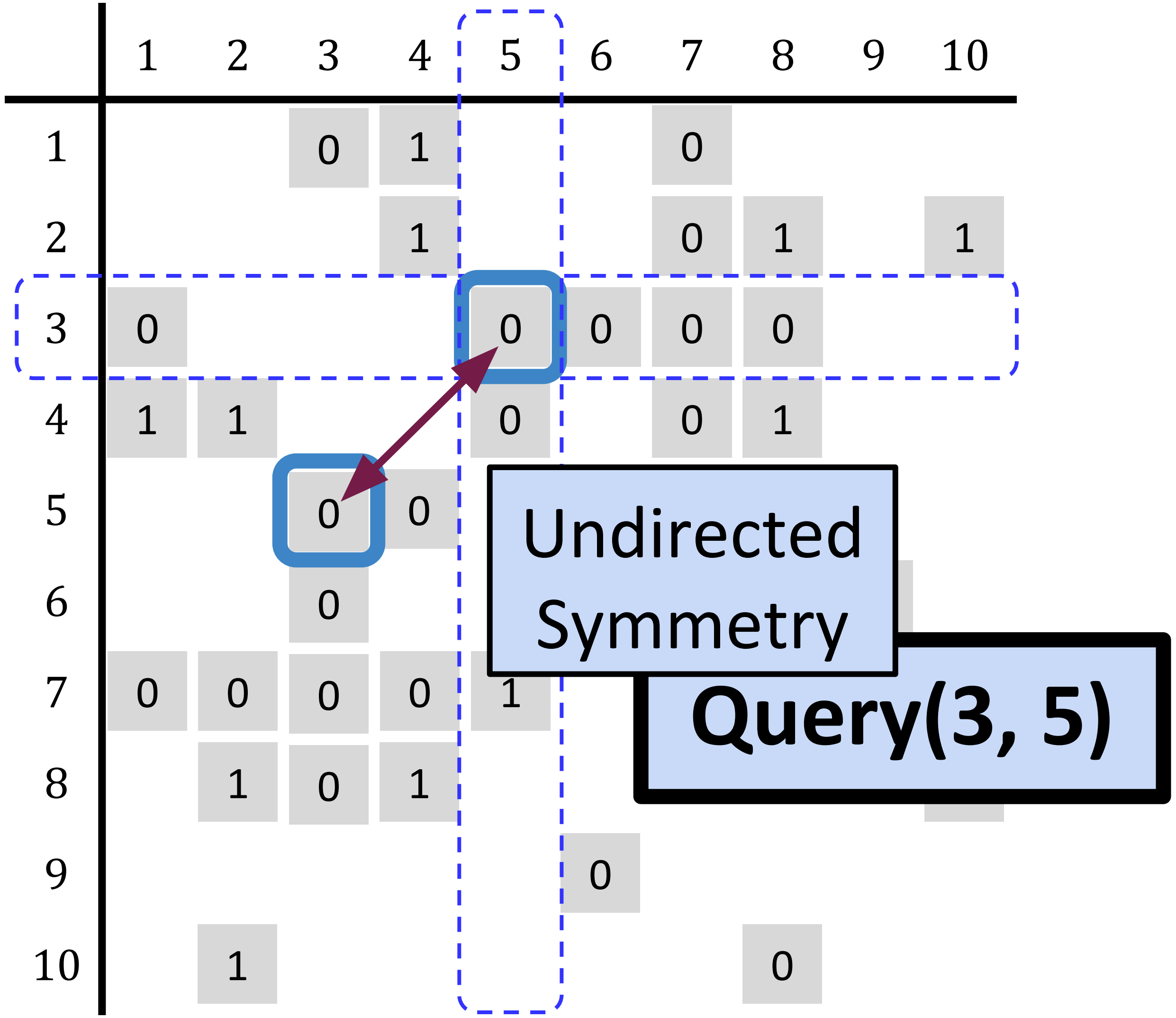
Sequential Markov Chain
works for $q > 2\Delta$

For $q > 9\Delta$
probe complexity is $n^{6.12\Delta/q}$



$G(n, p)$ graphs

Vertex-pair query: Is there an edge from u to v?



	1	2	3	4	5	6	7	8	9	10
1			0	1			0			
2				1			0	1		1
3	0				0	0	0	0		
4	1	1			0		0	1		
5			0	0						
6			0							
7	0	0	0	0	1					
8		1	0	1						
9						0				
10		1						0		

Undirected Symmetry

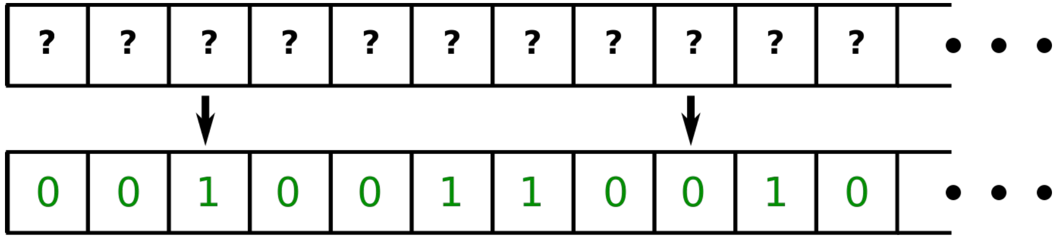
Query(3, 5)

Generate “on the fly”

toss coins as needed

Next-Neighbor and Random-Neighbor

- Dense case: $p > \frac{1}{poly(\log n)}$
  - Flip coins till you see **1**
  - Time: $O(1/p)$



?	?	?	?	?	?	?	?	?	?	?	• • •
0	0	1	0	0	1	1	0	0	1	0	• • •

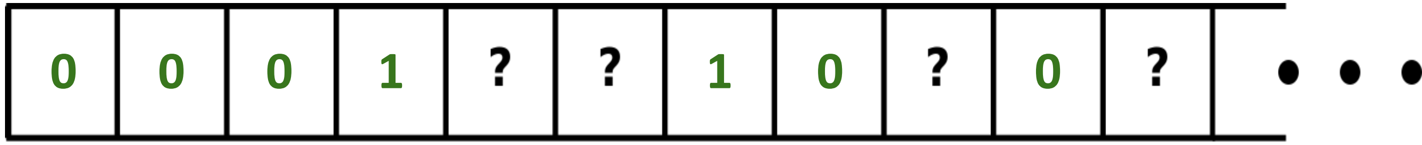
- Sparse Case: $p < \frac{poly(\log n)}{n}$
  - Use All-Neighbors query from [Naor-Nussboim 07]

- Intermediate is harder: e.g. $p = \frac{1}{\sqrt{n}}$
  - Many neighbors
  - Large gaps between neighbors

Can we do $o(1/p)$?

Next-Neighbor and Random-Neighbor

- Dense case: $p > \frac{1}{poly(\log n)}$
  - Flip coins till you see **1**
  - Time: $O(1/p)$



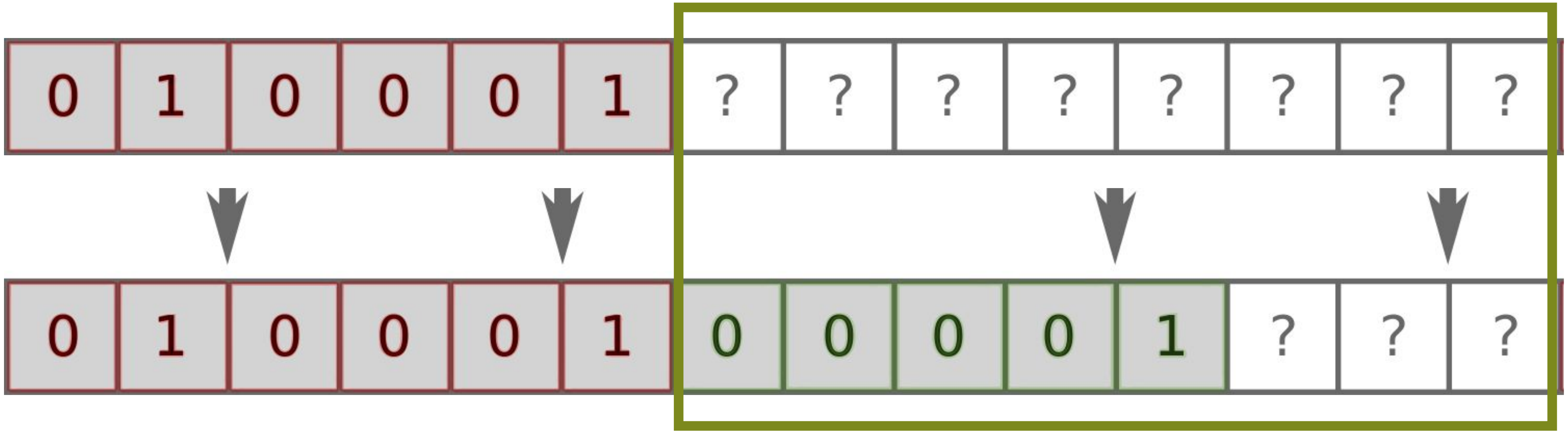
0	0	0	1	?	?	1	0	?	0	?	• • •

- Sparse Case: $p < \frac{poly(\log n)}{n}$
  - Use All-Neighbors query from [Naor-Nussboim 07]

- Intermediate is harder: e.g. $p = \frac{1}{\sqrt{n}}$
  - Many neighbors
  - Large gaps between neighbors

Can we do $o(1/p)$?

Skip-sampling for next-neighbor queries: The case of directed graphs



$$\mathbb{P}[k \text{ zeros followed by a } 1] = p(1-p)^k$$

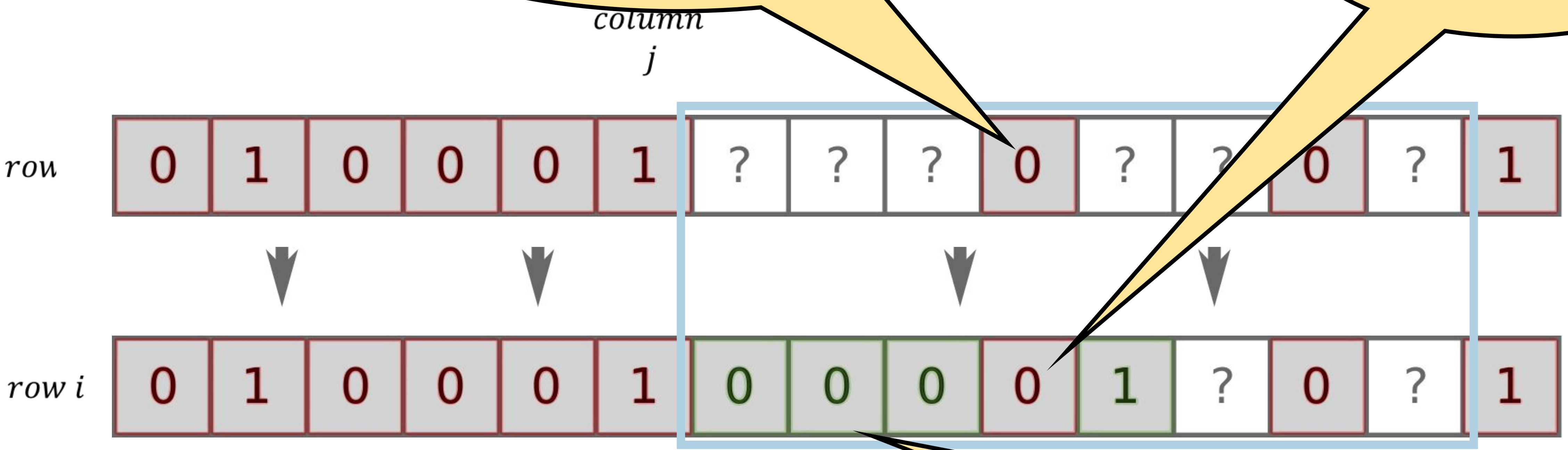
$$\text{CDF} = \sum_{k'=0}^{k} p(1-p)^{k'} = 1-(1-p)^k$$

Binary search on CDF

Skip-sampl bor querie
Undirecte

Some are determined by other neighbor

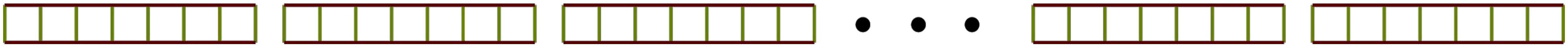
yields correct distribution?



row	0	1	0	0	0	1	?	?	?	0	?	?	0	?	1
row i	0	1	0	0	0	1	0	0	0	0	1	?	0	?	1

Write down all 0s?

Random-Neighbor queries via Bucketing

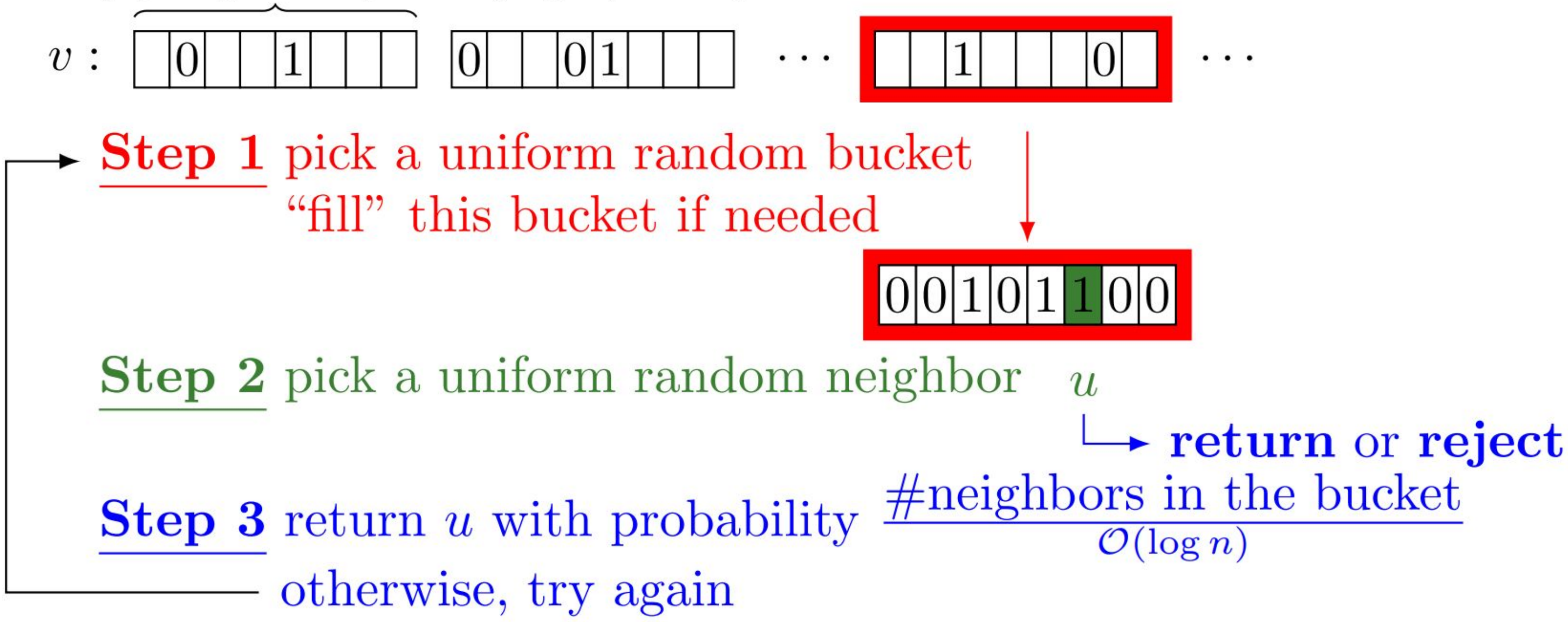


- Equipartition each row into *contiguous* **buckets** such that:
  - Expected # of neighbors in bucket is $\Theta(1)$
  - w.h.p. ⅓ of buckets are non-empty
  - w.h.p. no bucket has more than $\boldsymbol{\log n}$ neighbors
- Always determine a bucket completely

- Could have $\sqrt{n}$ buckets, each of size $\sqrt{n}$

Random Neighbors with rejection sampling

Bucketing: expected #neighbors in a bucket $= \Theta(1)$ expected, $\le \mathcal{O}(\log n)$ w.h.p. $\Rightarrow$ #neighbors $\approx$ #buckets



Step 1 pick a uniform random bucket
“fill” this bucket if needed

Step 2 pick a uniform random neighbor u
→ **return** or **reject**

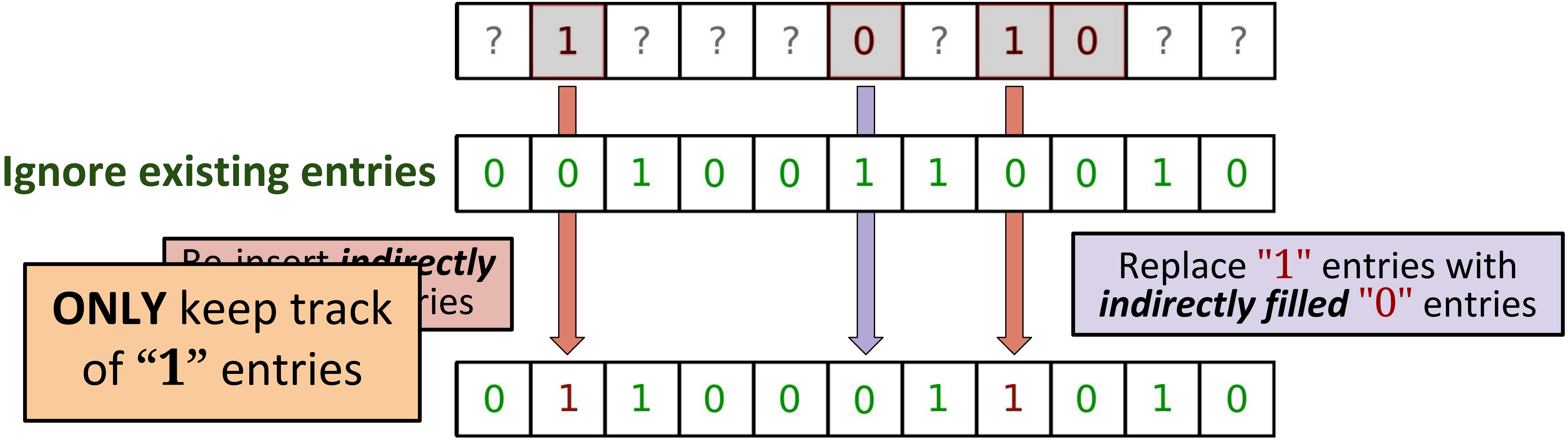
Step 3 return u with probability $\frac{\#\text{neighbors in the bucket}}{\mathcal{O}(\log n)}$
otherwise, try again

$$\mathbb{P}[\text{return } u] = \frac{1}{\#\text{buckets}} \times \frac{1}{\#\text{neighbors in bucket}} \times \frac{\#\text{neighbors in bucket}}{\mathcal{O}(\log n)} \approx \frac{\Omega(1/\log n)}{\#\text{neighbors of } v}$$

$$\mathbb{P}[\text{return any neighbor}] \approx \Omega(1/\log n) \Rightarrow \mathcal{O}(\log n) \text{ iterations suffice}$$

How to fill a bucket?

- Bucket may be ***indirectly*** filled in certain locations
  - "1" entries reported
  - "0" entries not reported but can be queried



?	1	?	?	?	0	?	1	0	?	?

Ignore existing entries

0	0	1	0	0	1	1	0	0	1	0

Re-insert ***indirectly***

ries

ONLY keep track of "**1**" entries

Replace "1" entries with ***indirectly filled*** "0" entries

0	1	1	0	0	0	1	1	0	1	0

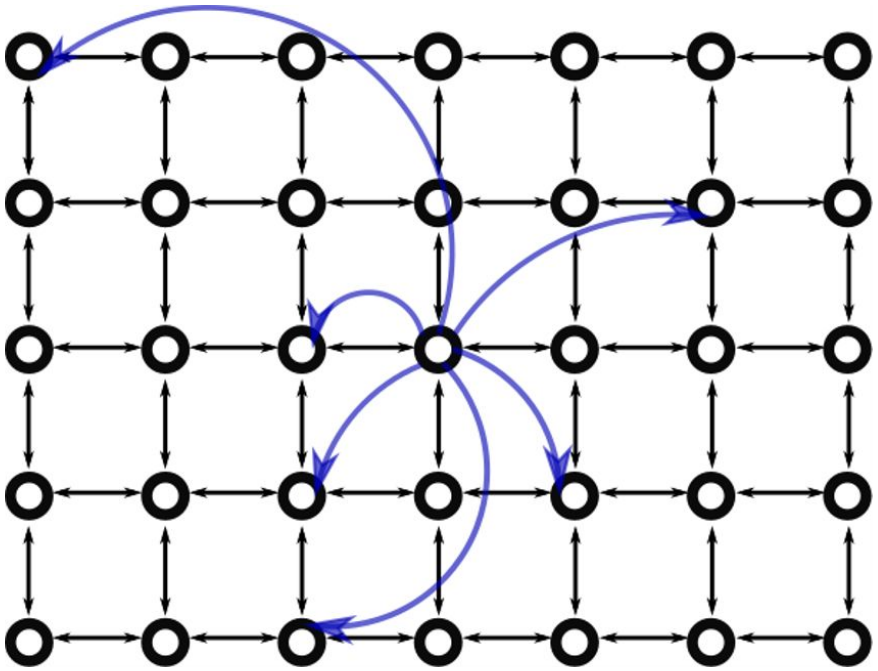
- **Why fast? . . .** # of "1" entries is bounded by $log\ n$

Bucketing provides Next-Neighbor queries too!

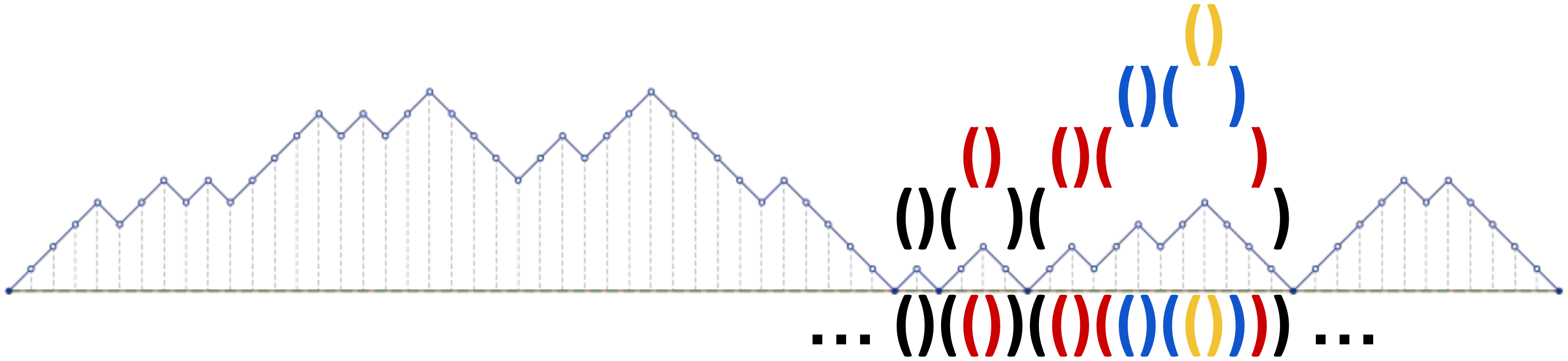
Just process the next bucket in order

General Graphs with Independent Edge Probabilities

- Need mild assumptions on computing sums/products of probabilities
- **Stochastic Block Model**
  - Community structure
  - Probability of edge depends on communities of endpoints
- **Kleinberg’s Small World Model**

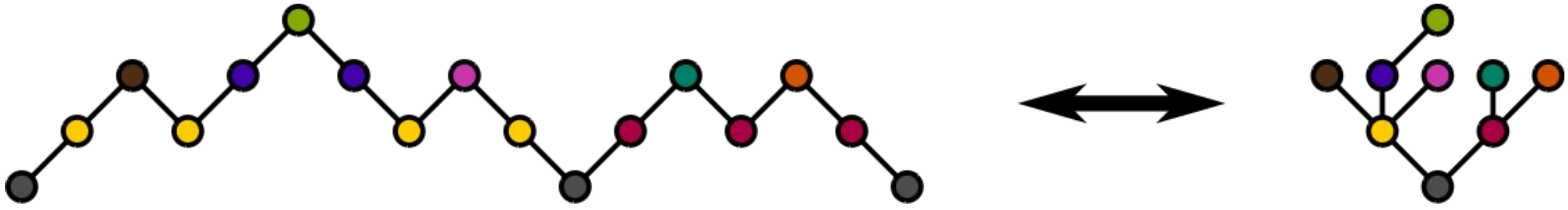


Other Results



- Random walks on the line
- Random Catalan objects
  - Random Dyck paths
    (1D random walk always positive)
  - Well bracketed expressions
  - Random Ordered Trees

- **Height** queries
  - **Depth** queries
    (in brackets and trees)
- **First-Return** queries
  - **Matching-Bracket** queries
  - **Next-Neighbor** in trees

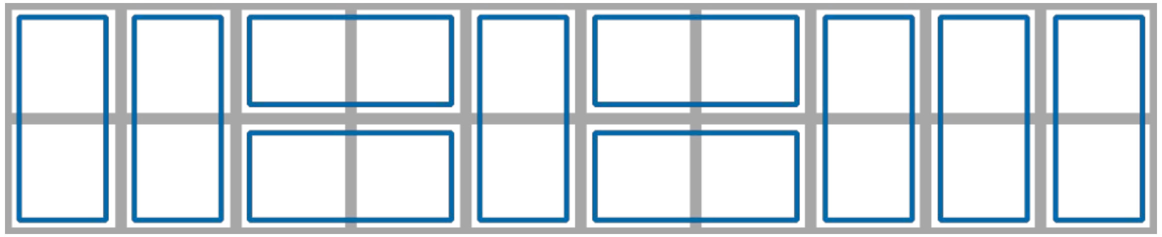


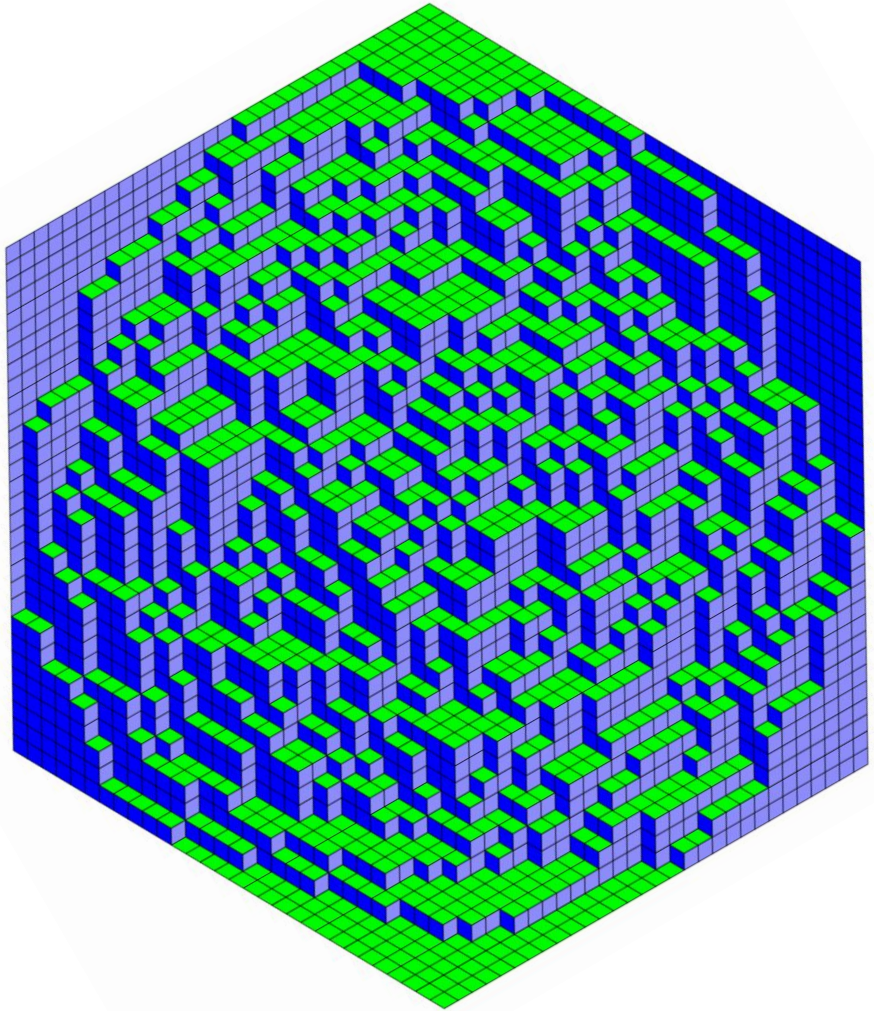
Open Problems: Random Graphs

- Degree queries
- i^{th} neighbor queries
- More complex queries
  - Sample a random triangle/clique
  - Random triangle containing specified vertex/edge

Open Problems: Large Description size

- What about $2\Delta < q < 9\Delta$?
- Random walks on general graphs
- Random satisfying assignment
- Random Linear Extensions of posets
- Random domino tilings (perfect matching)





Thank you!