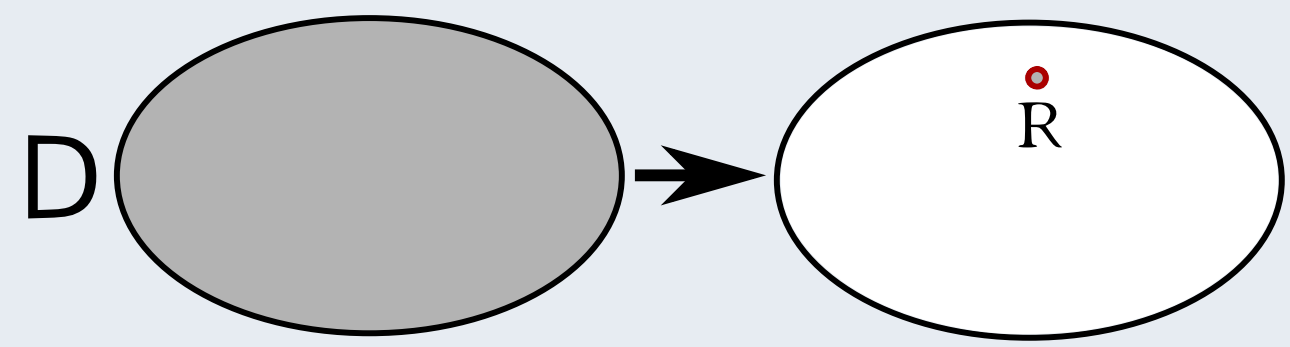


# Local-Access Generators

Amartya Shankha Biswas, Ronitt Rubinfeld, Anak Yodpinyanee  
CSAIL, MIT

## Partial Sampling from a Distribution

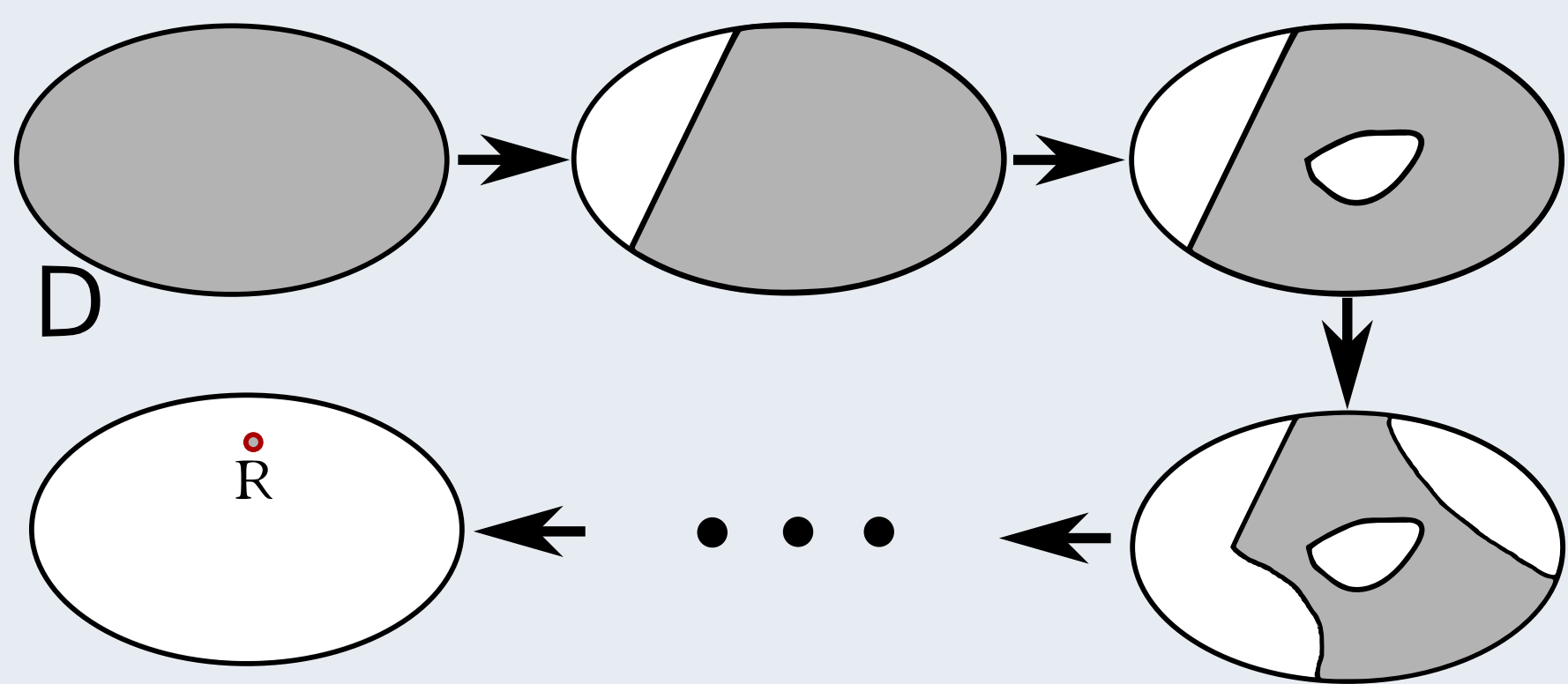
### Full Sampling $R \sim D$ in $\mathcal{O}(T)$ time



Do we need to spend  $\mathcal{O}(T)$  upfront?

### $N$ steps of *Partial Sampling*

Each partial step should take  $\tilde{\mathcal{O}}(T/N)$  time.



### Problem Statement

A local-access generator of a random object  $R \sim D$ , provides indirect access to  $R'$  with a *query oracle* s.t.

- All query responses (*partial samples*) are **consistent**
- The **distribution** of  $R'$  is  $\epsilon$ -close to  $D$  in  $L_1$  distance

## Bucketing-Generator & Random-Nighbor Queries

*Problem:* next-neighbor cannot “jump” to a random potential neighbor of  $v$

**Bucketing** Divide each row of the adjacency matrix into contiguous buckets  
 $\Rightarrow$  random neighbor of  $v \approx$  random neighbor in a random bucket of  $v$

*Problem:* Do NOT know  $\deg(v)$ : Must return each neighbor with prob.  $1/\deg(v)$

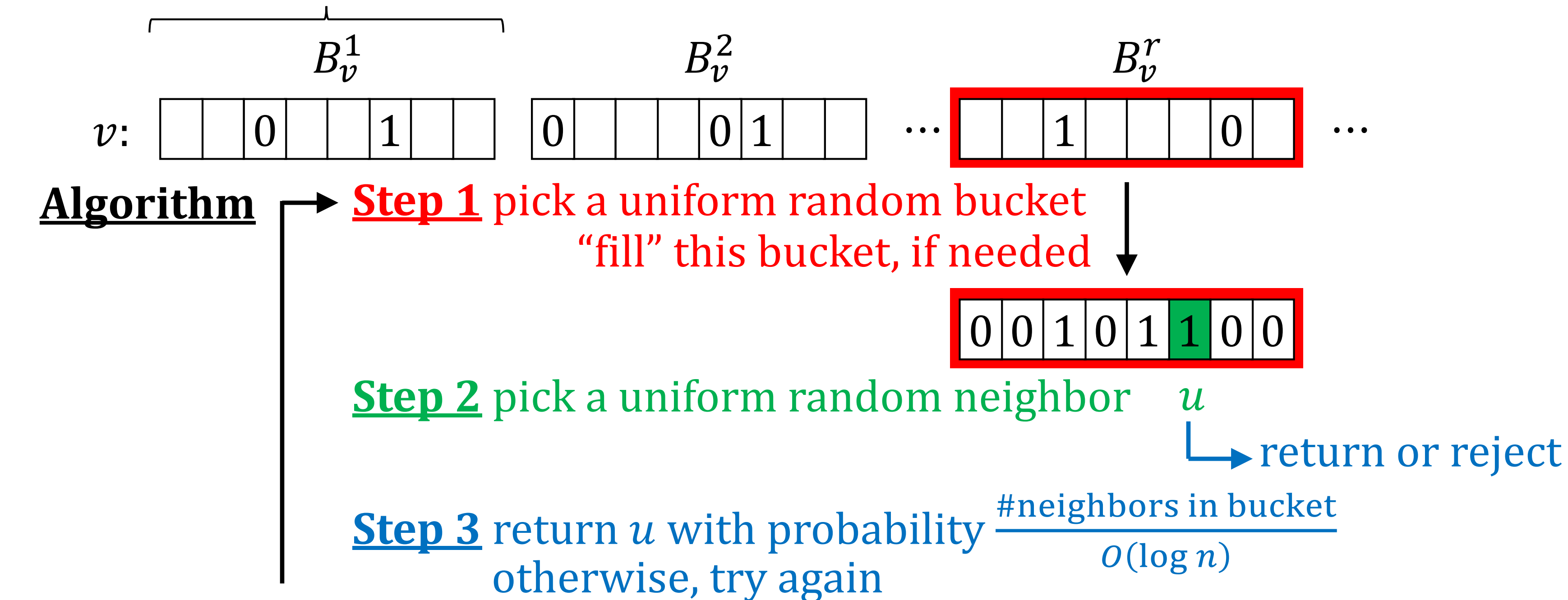
**Rejection Sampling** Normalize probability of returning any specific neighbor

*Problem:* next-neighbor cannot “jump” to a random potential neighbor of  $v$

$\Rightarrow$  suffice to show that **any neighbor** is returned with the **equal** probability

#neighbors in each bucket

$\sim \Theta(1)$  in expectation,  $O(\log n)$  max w.h.p.  $\Rightarrow$  #buckets  $\sim$  #neighbors



$$\Pr[u \text{ returned}] = \frac{1}{\text{\#buckets}} \times \frac{1}{\text{\#neighbors in bucket}} \times \frac{\text{\#neighbors in bucket}}{O(\log n)} \sim \frac{\Omega(1/\log n)}{\text{\#neighbors}}$$

$\Pr[\text{some neighbor returned}] \sim \Omega(1/\log n) \Rightarrow O(\log n)$  tries suffices

**Data Structure** Buckets contains set of known neighbors, and “filled” marker  
 $\Rightarrow$  “fill” with expected  $\Theta(1)$  next-neighbor queries  $\left. \begin{array}{l} O(\log n) \text{ time per query} \\ \tilde{O}(m+n) \text{ space usage} \end{array} \right\}$

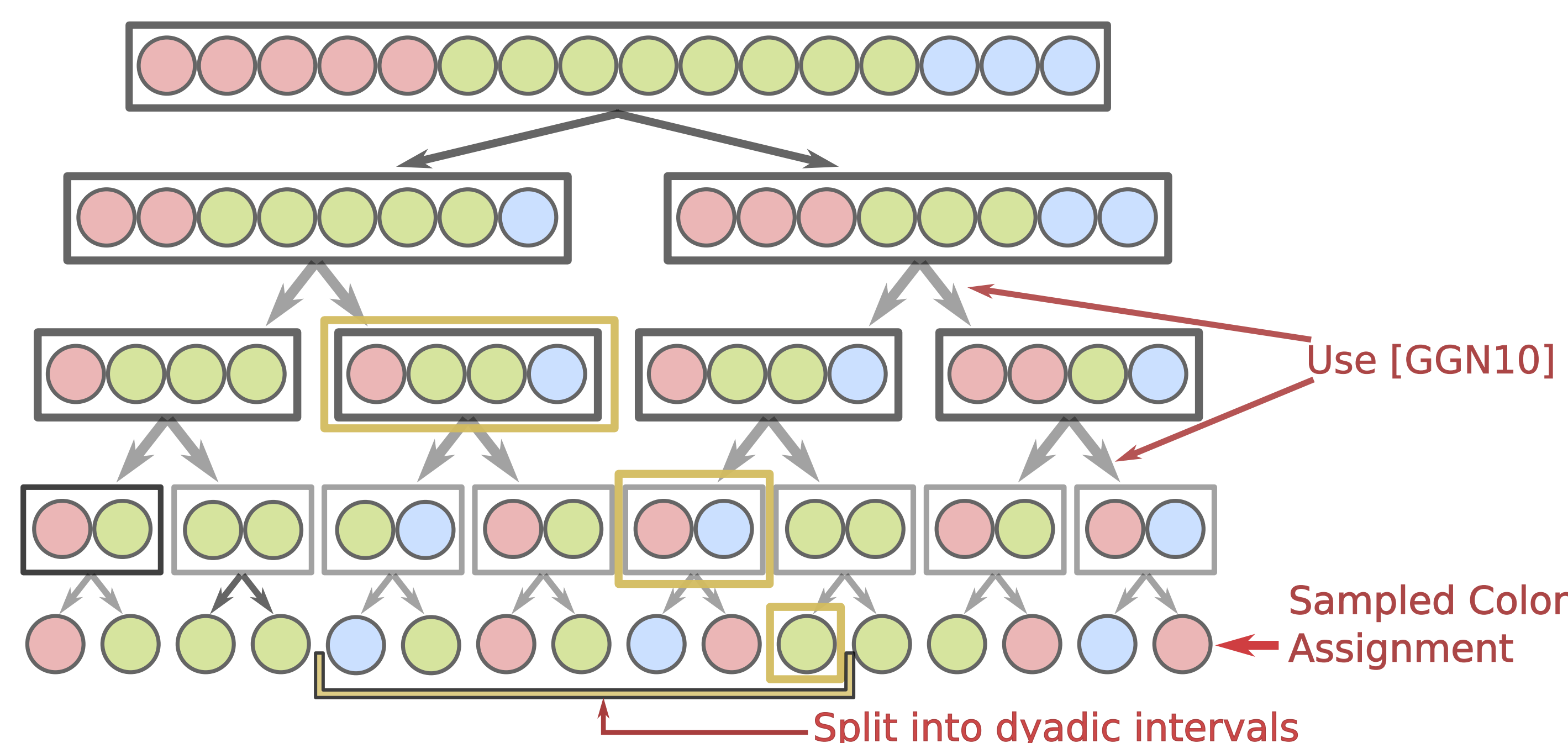
## Stochastic Block Model

Communities  $\{C_i\}_{i \in [r]}$  partition  $V$ : If  $u \in C_i, v \in C_j$ , then  $\mathbb{P}_{(u,v) \in E} = p_{ij}$ .

**Given sizes of each community  $C_i$  and a range of length  $\ell$**

- Count number of occurrences of each community in any contiguous range
- Sample from *Multivariate Hypergeometric Distribution*

$$\Pr[\mathbf{S}_\ell^C = \langle s_1, \dots, s_r \rangle] = \frac{\binom{C_1}{s_1} \cdot \binom{C_2}{s_2} \cdots \binom{C_r}{s_r}}{\binom{B}{\ell}} \quad \text{where } \ell = \sum_{i=1}^r s_i \text{ and } B = \sum_{i=1}^r C_i$$



## Multivariate Hypergeometric Distribution

[GGN10] solves the special case of  $r = 2$  and  $B = 2\ell$ .

COUNTING-GENERATOR

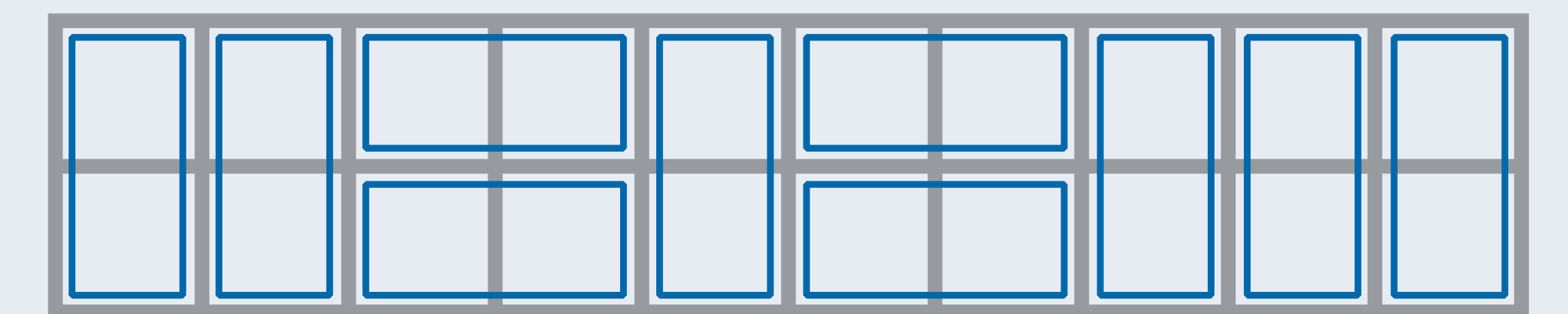
- **Extending to  $B \neq 2\ell$ :** Divide  $\ell$  into dyadic segments.
- **Extending to  $r > 2$ :** Make a tree with a leaf for each  $C_i$ . Every branch in the tree is equivalent to a 2-splitting

- Use COUNTING-GENERATOR to sample community counts
- Run the BUCKETING-GENERATOR as before

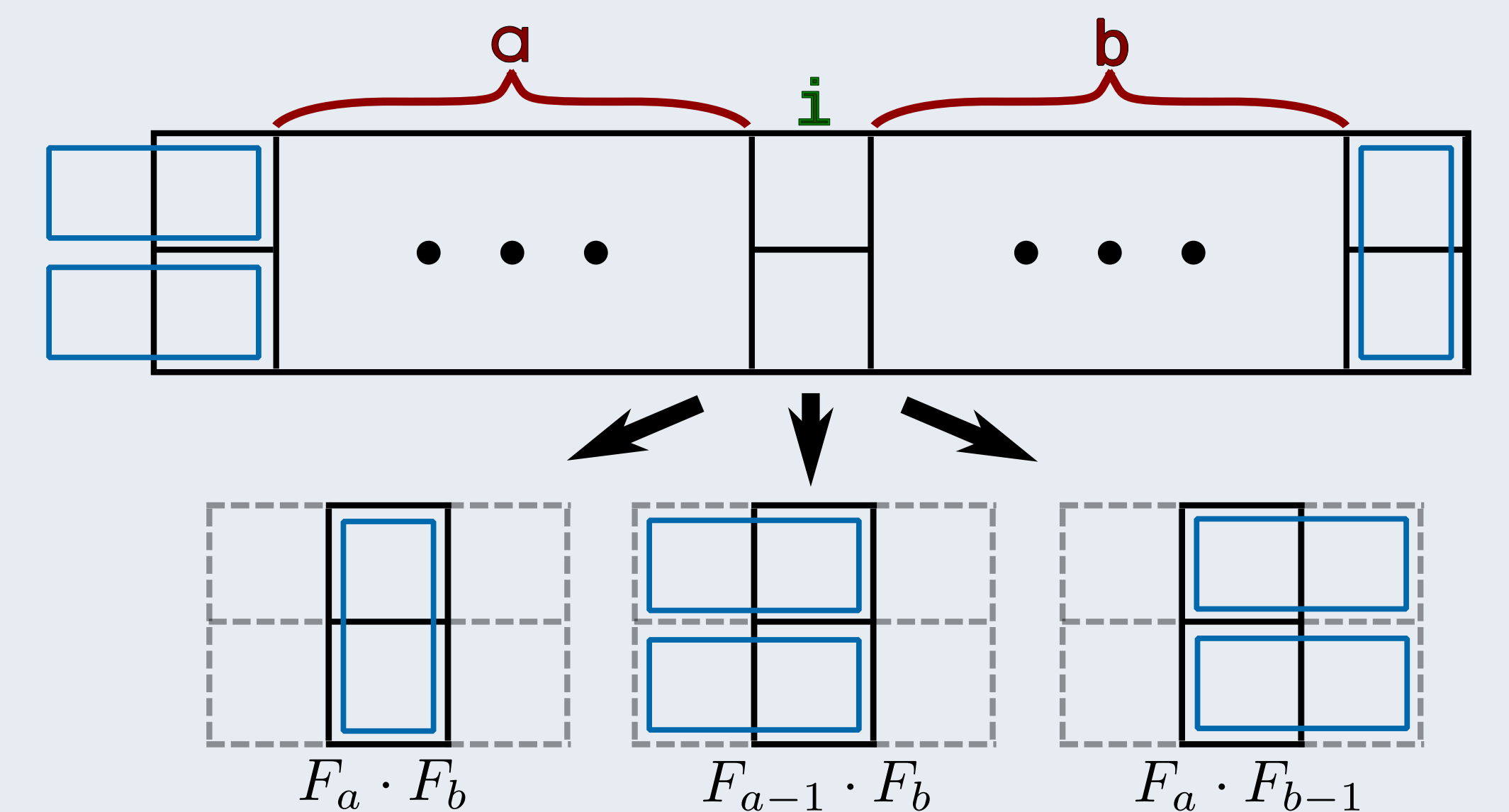
## Work in Progress

### Domino Tiling

A  $2 \times n$  grid tiled with dominoes:  $F_n$  tilings possible.



**Query:** Domino at position  $i$ : *vertical* OR *horizontal*?



Sufficient to approximate  $F_c/F_{c-1}$ : Use  $\phi$  if  $c = \Omega(\log(n))$

**Open:**  $k \times n$  grid for  $k = \omega(1)$  and Dimer model.

## Trivial Example - Sampling $G(n, p)$

**Model:**  $N$  vertex undirected graph: edge probability  $p$

**Query Model:** Given vertices  $u, v$ , is  $(u, v) \in E$ ?

- Just a collection of  $\frac{N(N-1)}{2}$  Bernoulli RVs with bias  $p$ .

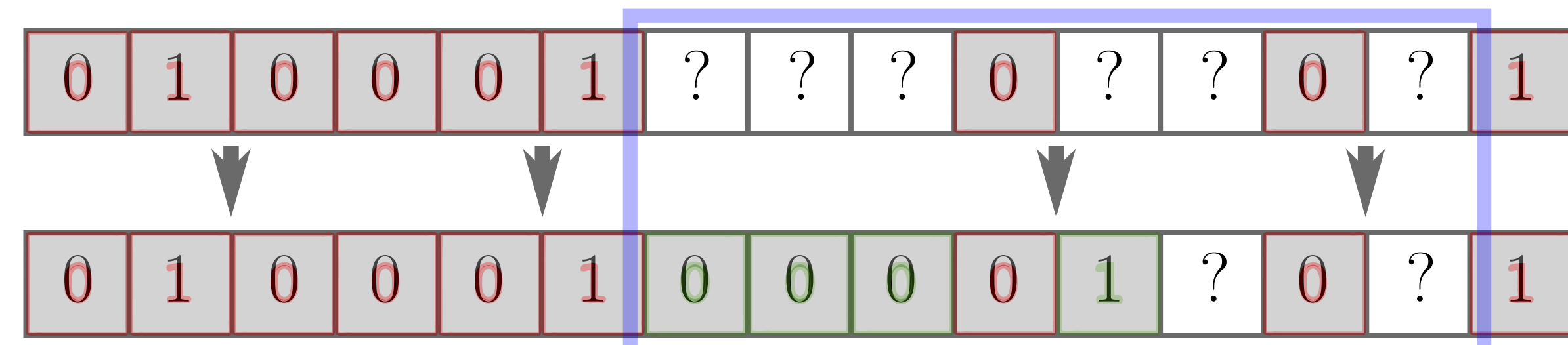
## Find Next-Neighbor (*skip-sampling*)

**Adjacency List query:** Return neighbors of  $v$  in order.

$$\mathbb{P}[k \text{ non-neighbors before next-neighbor}] = p(1-p)^k$$

- Can sample from this distribution in  $\tilde{\mathcal{O}}(1)$  time [ELMR17]
- Avoid sampling each 0 separately

**Issue:** Adjacency matrix is symmetric So, each zero must also appear in the corresponding column of  $v$



If the sampled neighbor is a 0, discard and resample.

**Cannot afford too many re-samplings.**