


Miscellaneous Results

Amartya Shankha Biswas 

CSAIL, MIT
asbiswas@mit.edu

Ronitt Rubinfeld

CSAIL, MIT
ronitt@csail.mit.edu

Anak Yodpinyanee 

CSAIL, MIT
anak@csail.mit.edu

Abstract

Consider a computation on a massive random graph: Does one need to generate the whole random graph up front, prior to performing the computation? Or, is it possible to provide an oracle to answer queries to the random graph "on-the-fly" in a much more efficient manner overall? That is, to provide a *local access generator* which incrementally constructs the random graph locally, at the queried portions, in a manner consistent with the random graph model and all previous choices. Local access generators can be useful when studying the local behavior of specific random graph models. Our goal is to design local access generators whose required resource overhead for answering each query is significantly more efficient than generating the whole random graph.

Our results focus on undirected graphs with independent edge probabilities, that is, each edge is chosen as an independent Bernoulli random variable. We provide a general implementation for generators in this model. Then, we use this construction to obtain the first efficient local implementations for the Erdős-Rényi $G(n, p)$ model, and the Stochastic Block model.

As in previous local-access implementations for random graphs, we support VERTEX-PAIR, NEXT-NEIGHBOR queries, and ALL-NEIGHBORS queries. In addition, we introduce a new RANDOM-NEIGHBOR query. We also give the first local-access generation procedure for ALL-NEIGHBORS queries in the (sparse and directed) Kleinberg's Small-World model. Note that, in the sparse case, an ALL-NEIGHBORS query can be used to simulate the other types of queries efficiently. All of our generators require no pre-processing time, and answer each query using $\mathcal{O}(\text{poly}(\log n))$ time, random bits, and additional space.

2012 ACM Subject Classification **Author:** Please fill in 1 or more `\ccsdsc macro`

Keywords and phrases Dummy keyword

Funding *Amartya Shankha Biswas:* funding

Ronitt Rubinfeld: funding

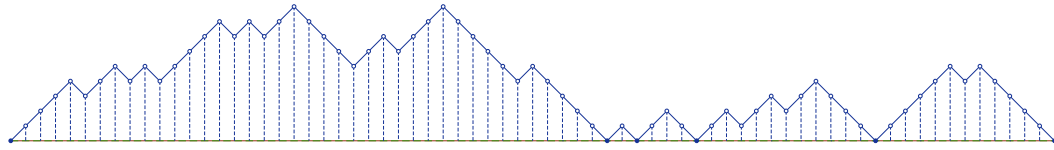
Anak Yodpinyanee: funding

Contents

1	Sampling Catalan Objects	3
1.1	Catalan Trapezoids and Generalized Dyck Paths	4
1.2	Sampling the Height	4
1.2.1	The Simple Case: Far Boundary	5
1.2.2	Path Segments Close to Zero	5
1.3	Supporting “First Return” Queries	7
1.3.1	Maintaining a Boundary Invariant	7
1.3.2	Sampling the Lowest Achievable Height in an Interval	8
1.3.3	Sampling First Position that Touches the “ <i>Mandatory Boundary</i> ”	8
1.3.4	Estimating the CDF	9
1.3.5	Finding the Correct Interval: FIRST-RETURN Query	9
2	Random Coloring of a Graph	10
2.1	Modified Glauber Dynamics	11
2.2	Local Coloring Algorithm	11
A	Dyck Path Generator	15
A.1	Approximating Close-to-Central Binomial Coefficients	15
A.2	Dyck Path Boundaries and Deviations	16
A.3	Computing Probabilities	17
A.4	Sampling the Height	17
A.5	First Return Sampling	20

1 Sampling Catalan Objects

Earlier, we were interested in querying the following random object. In a random permutation of n white marbles and n black marbles, how many white marbles are present in the first k positions. As we have seen before, [GGN10] gives us a method of sampling from this (hypergeometric) distribution. In constructing a generator for the Stochastic Block model, we generalized this by adding more colors (multivariate hypergeometric distribution). We also took this to the extreme where all marbles are distinguishable (i.e. a random permutation), and saw that this could also be implemented efficiently. Now we focus on a more challenging variant of this question with more complicated conditional dependences among the placement of the marbles.



■ **Figure 1** Simple Dyck path with $n = 35$.

We consider a sequence of n white and n black marbles such that every prefix of the sequence has at least as many white marbles as black ones. The number of such sequences is the n^{th} Catalan number C_n . Many important combinatorial objects occur in Catalan families. Dyck paths are one such interpretation of the Catalan numbers that fits in nicely with our setup of white and black marbles. A Dyck path is constructed as a $2n$ step one-dimensional walk with exactly n up and down steps. In Figure 1, each step in the walk moves one unit along the positive x -axis and one unit up or down the positive y -axis. Given these restrictions, we would obtain a 1D random walk pinned to zero on both sides. A Dyck path also has a restriction that the y -coordinate of any point on the random walk is ≥ 0 i.e. the walk never crosses the x -axis. The number of possible Dyck paths (see Theorem 28) is the n^{th} Catalan number $C_n = \frac{1}{n+1} \cdot \binom{2n}{n}$.

We will attempt to support queries to a uniformly random instance of a Dyck path. Specifically, we will want to answer the following queries:

- **HEIGHT(i)**: Returns the position of the path after i steps
- **FIRST-RETURN(i)**: Returns an index $j > i$ such that **HEIGHT(j)** = **HEIGHT(i)** and for any k between i and j , **HEIGHT(k)** is strictly greater than **HEIGHT(i)**.

The **HEIGHT** query seems natural for Dyck paths, but the **FIRST-RETURN** query is important in exploring other Catalan objects. For instance, consider a random well bracketed expression i.e. a uniform distribution over the Dyck language. There is a trivial bijection between Dyck paths and words in this language. The **HEIGHT** query corresponds to asking for the nesting depth at a certain position in the word, and **FIRST-RETURN(i)** returns the position of the matching bracket for position i .

There is also a natural bijection between Dyck paths and rooted ordered trees by letting the path be a transcript of the DFS traversal of a tree. Starting with the root, for each up step we create a new child of the current node, and for each down step, we backtrack up the tree. Here, the **HEIGHT** query returns the depth of a node and the **FIRST-RETURN** query can be used to find the *next child* of a node. Moving forwards, we will focus on Dyck paths for the sake of clarity.

Computing p_d values.
Mention that this is imperfect sampling (close impl.)

Thresholding.

Mention that this is imperfect sampling (close impl.)

Fix $\sqrt{B \log n}$ everywhere.

we have?

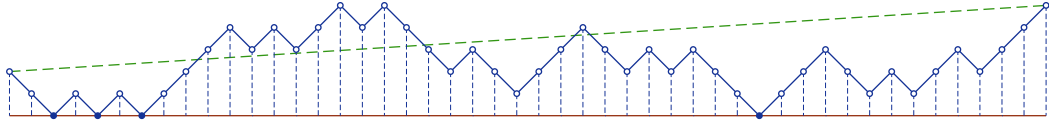
Connect with previous part on SBM

Figure? Degree queries by repeated application.

1.1 Catalan Trapezoids and Generalized Dyck Paths

In order to sample Dyck paths locally, we will need to analyze more general Catalan objects which correspond to numbers in the *Catalan Trapezoid* (presented in [Reu14]). Let $C_k(n, m)$ be the $(n, m)^{th}$ entry of the Catalan trapezoid of order k , where $C_1(n, m)$ corresponds to the Catalan triangle. We can interpret $C_k(n, m)$ as the number of *generalized* Dyck paths. Specifically, we consider a sequence of n up-steps and m down-steps, such that the sum of any initial sub-string is not less than $1 - k$. This means that we start our Dyck path at a height of $k - 1$, and we are never allowed to cross below zero (Figure 2). The total number of such paths is exactly $C_k(n, m)$. For $k = 1$ and $n = m$, we obtain the definition of the simple Dyck path (Figure 1). Now, we state a result from [Reu14] without proof

$$C_k(n, m) = \begin{cases} \binom{n+m}{m} & 0 \leq m < k \\ \binom{n+m}{m} - \binom{n+m}{m-k} & k \leq m \leq n + k - 1 \\ 0 & m > n + k - 1 \end{cases}$$



■ **Figure 2** Complex Dyck path with $n = 25$, $m = 22$ and $k = 3$. Notice that the boundary is shifted.

1.2 Sampling the Height

Our general recursive step is as follows. We consider a random sequence of length $2B$ comprising of $2U$ up steps (+1) and $2D$ down steps (-1) where the sum of any prefix cannot be less than $k - 1$. Without loss of generality, let's assume that $2D \leq B$. If this were not the case, we could simply flip the sequence and negate the elements. This essentially means that the overall Dyck path is non-decreasing.

We want to sample the height of this path after B steps. This is the same as sampling the number of (+1)s that get assigned to the first half of the elements in the sequence. We define p_d as the probability that exactly $D - d$ (-1)s get assigned to the first half. This means that exactly $U + d$ (+1)s get assigned to the first half. Consequently, the second half will contain exactly $D + d$ (-1)s and $U - d$ (+1)s.

What is d if negative?

Let us first compute this probability.

$$p_d = \frac{D_{left} \cdot D_{right}}{D_{tot}}$$

Here, D_{left} denotes the number of valid starting sequences (first half) and D_{right} denotes the number of valid ending sequences. Here, *valid* means that each half sequence gets the appropriate number of ups and downs and the initial sums never drop below $1 - k$. For, D_{right} , we will start the Dyck path from the end of the $2B$ sequence. In this case the invalidation threshold will be a different k' . This k' is the final height of the $2B$ sequence. So, $k' = k + 2U - 2D = k + 4B - 2D$. We will use this fact extensively moving forward.

Also, D_{tot} is the total number of possible sequences of length $2B$, given the initial conditions. Note that in this case the threshold remains at k .

Frequently?

We will use the following rejection sampling lemma from [GGN10].

► **Lemma 1.** Let $\{p_i\}$ and $\{q_i\}$ be distributions satisfying the following conditions

1. There is a poly-time algorithm to approximate p_i and q_i up to $\pm n^{-2}$
2. Generating an index i according to q_i is closely implementable.
3. There exists a poly($\log n$)-time recognizable set B such that
 - $1 - \sum_{i \in B} p_i$ is negligible
 - There exists a constant c such that for every i , it holds that $p_i \leq \log^{O(1)} n \cdot q_i$

Then, generating an index i according to the distribution $\{p_i\}$ is closely-implementable.

We also use the following lemmas to bound the deviation of the path.

► **Lemma 2.** Consider a contiguous sub-path of a simple Dyck path of length $2n$ where the sub-path is of length $2B$ comprising of U up-steps and D down-steps (with $U + D = 2B$). Then there exists a constant c such that the quantities $|B - U|$, $|B - D|$, and $|U - D|$ are all $< c\sqrt{B \log n}$ with probability at least $1 - 1/n^2$ for every possible sub-path.

► **Lemma 3.** There exists a constant c such that if $k > c\sqrt{B \log n}$, then the distribution of paths sampled without a boundary (hypergeometric sampling) is statistically $\mathcal{O}(1/n^2)$ -close in L_1 distance to the distribution of Dyck paths.

1.2.1 The Simple Case: Far Boundary

By Lemma 3, the problem reduces to the hypergeometric sampling case when $k > \mathcal{O}(\sqrt{B \log n})$. In this case, we can simply approximate the probability as

$$\frac{\binom{B}{D-d} \cdot \binom{B}{D+d}}{\binom{2B}{2D}}$$

This is because unconstrained random walks will not dip below the $1 - k$ threshold with high probability. This problem was solved in [GGN10] using $\mathcal{O}(\text{poly}(\log n))$ resources.

1.2.2 Path Segments Close to Zero

The problem arises when we $k = \mathcal{O}(\sqrt{B \log n})$. In this case we need to compute the actual probability. Using the formula from [Reu14], we find that.

$$D_{left} = \binom{B}{D-d} - \binom{B}{D-d-k} \quad D_{right} = \binom{B}{U-d} - \binom{B}{U-d-k'} \quad D_{tot} = \binom{2B}{2D} - \binom{2B}{2D-k} \quad (1)$$

Here, $k' = k + 2U - 2D$, and so $k' = \mathcal{O}(\sqrt{B \log n})$ (using Lemma 2).

The final distribution we wish to sample from is given by $\{p_d\}_d$ where $p_d = \frac{D_{left} \cdot D_{right}}{D_{tot}}$. To achieve this, we will use Lemma 1 from [GGN10]. An important point to note is in order to apply this lemma, we must be able to compute the p_d values. For now, we will assume that we have access to an oracle that will compute the value for us. Later, in Section , we will see how to construct such an oracle.

Fix reference

Next, we will construct an appropriate $\{q_i\}$ and show that $p_d < \text{poly}(\log n) \cdot q_d$ for all $|d| = \mathcal{O}(\sqrt{B \log n})$ which will allow us to invoke Lemma 1. Note that Lemma 2 implies that the probability mass associated with $|d| > \Theta(\sqrt{B \log n})$ is negligible. In fact we will be able to use the hypergeometric distribution for q_d :

$$q_d = \frac{\binom{B}{D-d} \cdot \binom{B}{D+d}}{\binom{2B}{2D}} = \frac{\binom{B}{D-d} \cdot \binom{B}{U-d}}{\binom{2B}{2D}}$$

First, we consider the case where $k \cdot k' \leq 2U + 1$. In this case, we use loose bounds for $D_{left} < \binom{B}{D-d}$ and $D_{right} < \binom{B}{U-d}$. We also use the following lemma (proven in Section A).

► **Lemma 4.** *When $kk' > 2U + 1$, $D_{tot} > \frac{1}{2} \cdot \binom{2B}{2D}$.*

Combining the three bounds we obtain $p_d < \frac{1}{2}q_d$. Intuitively, in this case the Dyck boundary is far away, and therefore the number of possible paths is only a constant factor away from the number of unconstrained paths (see Section 1.2.1). The case where the boundaries are closer (i.e. $k \cdot k' \leq 2U + 1$) is trickier, since the individual counts need not be close to the corresponding binomial counts. However, in this case we can still ensure that the sampling probability is within poly-logarithmic factors of the binomial sampling probability. We use the following lemmas (proven in Section A).

► **Lemma 5.** $D_{left} \leq c_1 \frac{k \cdot \log n}{\sqrt{B}} \cdot \binom{B}{D-d}$ for some constant c_1 .

► **Lemma 6.** $D_{right} < c_2 \frac{k' \cdot \log n}{\sqrt{B}} \cdot \binom{B}{U-d}$ for some constant c_2 .

► **Lemma 7.** *When $kk' \leq 2U + 1$, $D_{tot} < c_3 \frac{k \cdot k'}{B} \cdot \binom{2B}{2D}$ for some constant c_3 .*

We can now put these lemmas together to show that $p_d/q_d \leq \Theta(\log^2 n)$ and invoke Lemma 1 to sample the value of d . This gives us the height of the Dyck path at the midpoint of the two given points.

► **Theorem 8.** *Given two positions a and b (and the associated heights) in a Dyck path of length $2n$ with the guarantee that no position between a and b has been sampled yet, there is an algorithm that returns the height of the path halfway between a and b . Moreover, this algorithm only uses $\mathcal{O}(\text{poly}(\log n))$ resources.*

Proof. If $b - a$ is even, we can set $B = (b - a)/2$. Otherwise, we first sample a single step from a to $a + 1$, and then set $B = (b - a - 1)/2$. Since there are only two possibilities for a single step, we can explicitly compute an approximation of the probabilities, and then sample accordingly. Now, if $B > \Theta(\log^2 n)$ we can simply use the rejection sampling procedure described above to obtain a $\mathcal{O}(\text{poly}(\log n))$ algorithm. Otherwise, we sample each step individually. Since there are only $2B = \Theta(\log^2 n)$ steps, the sampling is still efficient. \square

► **Theorem 9.** *There is an algorithm that provides sample access to a Dyck path of length $2n$, by answering queries of the form $\text{HEIGHT}(x)$ with the correctly sampled height of the Dyck path at position x using only $\mathcal{O}(\text{poly}(\log n))$ resources per query.*

Proof. The algorithm maintains a successor-predecessor data structure (e.g. Van Emde Boas tree) to store all positions x that have already been queried. Each newly queried position is added to this structure. Given a query $\text{HEIGHT}(x)$, the algorithm first finds the successor and predecessor (say a and b) of x among the already queried positions. This provides us the guarantee required to apply Theorem 8, which allows us to query the height at the midpoint of a and b . We then binary search by updating either the successor or predecessor of x . Once the interval length becomes less than $\Theta(\log^2 n)$, we perform the full sampling (as in Theorem 8) which provides us the height at position x . \square

1.3 Supporting “First Return” Queries

We might want to support more complex queries to a Dyck path. Specifically, in addition to querying the height of a position, we might want to know the next time the path return to that height (if at all). We introduce a new query $\text{FIRST-RETURN}(x)$ which returns the first time the walk returns to $\text{HEIGHT}(x)$ if the step from x to $x+1$ is an up-step.

The utility of this kind of query can be seen in other interpretations of Catalan objects. For instance, if we interpret it as a well bracketed expression, $\text{FIRST-RETURN}(x)$ returns the position of the bracket matching the one started at x . If we consider a uniformly random rooted tree, the function effectively returns the next child of a vertex.

Explain why

We will use the following asymptotic formula for *close-to-central* binomial coefficients.

► **Lemma 10.** *If $k = \frac{n \pm c\sqrt{n}}{2}$ where $c = o(n^{1/6})$, we can approximate $\binom{n}{k}$ up to constant factors by the expression $\frac{2^n}{\sqrt{n}} \cdot e^{-c^2/2}$.*

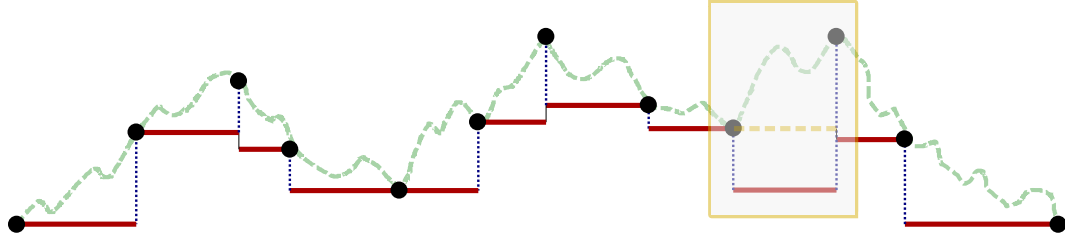
We maintain a threshold $\mathcal{T} = \Theta(\log^4 n)$. If an un-sampled interval in the Dyck path has length less than \mathcal{T} , then we sample the entire interval. So, for intervals with length $B > \mathcal{T}$, the maximum deviations are bounded by $\mathcal{O}(\sqrt{B \log n}) = \mathcal{O}(\log^{2.5} n)$ with high probability. This means that if we write the deviation as $c\sqrt{n}$, we see that $c = \mathcal{O}(\sqrt{\log n})$ which is $o(B^{1/6})$.

1.3.1 Maintaining a Boundary Invariant

Why?

Consider all positions that have been queried already $\langle x_1, x_2, \dots, x_m \rangle$ (in increasing order) along with their corresponding heights $\langle h_1, h_2, \dots, h_m \rangle$.

► **Proposition 11.** *For any two adjacent positions x_i, x_{i+1} whose heights have been sampled as h_i, h_{i+1} , the Dyck path between positions x_i and x_{i+1} is constrained to lie above $\min(h_i, h_{i+1})$.*



■ **Figure 3** Error in third segment.

It is not even clear that this is always possible. After sampling the height of a particular position x_i as h_i (with $x_{i-1} < x_i < x_{i+1}$), the invariant is potentially broken on either side of x_i . We will re-establish the invariant by sampling an additional point on either side. This proceeds as follows for the interval between x_i and x_{i+1} (see error in Figure 3):

1. Sample the lowest height h achieved by the walk between x_i and x_{i+1} .
2. Sample a position x such that $x_i < x < x_{i+1}$ and $\text{HEIGHT}(x) = h$.

Since h is the minimum height along this interval, sampling the point x suffices to preserve the invariant.

1.3.2 Sampling the Lowest Achievable Height in an Interval

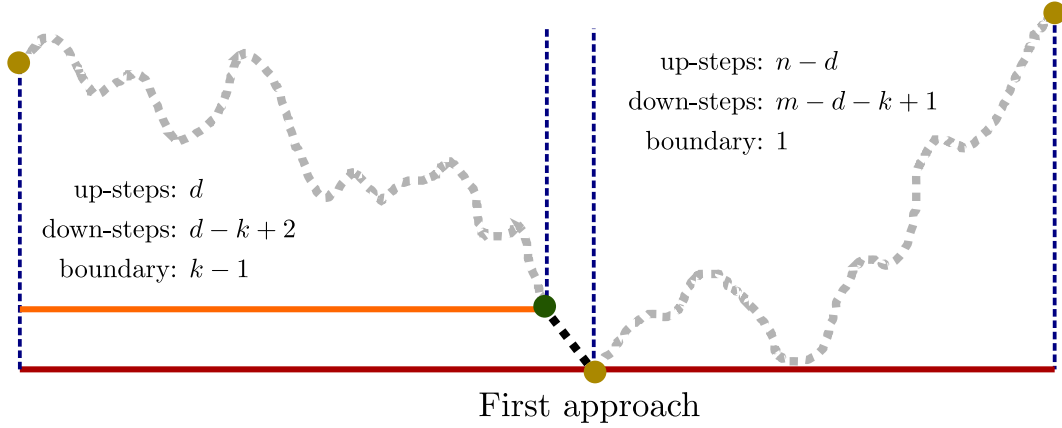
For the first step, we need to sample the lowest height of the walk between x_i and x_{i+1} . Notice that we can assume $x_i < x_{i+1}$ without loss of generality (if $x_i > x_{i+1}$, swap them and proceed). Let's say that the boundary is currently $k' - 1$ units below h_i .

We know how to count the number of possible Dyck paths for any given boundary. Dividing by the total number of possible paths gives us precisely the CDF we need. This allows us to binary search to find the boundary.

We will use D_k to denote the number of paths that respect a boundary which is $k - 1$ units below h_i . So, in the first step, we compute $p = D_{k'/2}/D_{k'}$. This means that with probability p , the path never reaches height $h_i - k'/2$. Otherwise, the path must reach $h_i - k'/2$ but not $h_i - k'$. Note that we can also calculate the total number of such paths as $D_{k'} - D_{k'/2}$. We repeat this procedure, essentially performing binary search, until we find a k such that the path reaches height $h_i - k + 1$ (potentially multiple times), but never goes below it.

1.3.3 Sampling First Position that Touches the “Mandatory Boundary”

Now that we have a “mandatory boundary” k , we just need to sample a position x with height $h = x_i - k + 1$. In fact, we will do something stronger by sampling the *first* time the walk touches the boundary after x_i .



■ **Figure 4** Zooming into the error in Figure 3

We will parameterize the position x the number of up-steps between x_i and x (See Figure 4). This quantity will be referred to as d such that $x - x_{i+1} = 2d + k - 1$. Given a specific d , we want to compute the number of valid paths that result in d up-steps before the first approach to the boundary. We will calculate this quantity by counting the total number of paths to the left and right of the first approach and multiplying them together.

Since we only care about getting a asymptotic (up to poly(log n) factors) estimate of the probabilities, it suffices to estimate the number of paths asymptotically as well.

► **Lemma 12.** If $d > \log^4 n$, then $D_{left}(d) = \Theta\left(\frac{2^{2d+k}}{\sqrt{d}} e^{-r_{left}(d)} \cdot \frac{k-1}{d+k-1}\right)$ where $r_{left}(d) = \frac{(k-2)^2}{2(2d+k-2)}$.

► **Lemma 13.** If $U+D-2d-k > \log^4 n$, then $D_{right}(d) = \Theta\left(\frac{2^{U+D-2d-k}}{\sqrt{U+d-2d-k}} e^{-r_{right}(d)} \cdot \frac{U-D+k}{U-d+1}\right)$ where $r_{right}(d) = \frac{(U-D-k-1)^2}{4(U+D-2d-k+1)}$.

Deal with
smaller val-
ues of d

Deal with
other values
of d

1.3.4 Estimating the CDF

► **Lemma 14.** $D_{left}(d) \cdot D_{right}(d) = \Theta \left(\frac{2^{U+D}}{\sqrt{d(U+D-2d-k)}} e^{-r(d)} \cdot \frac{k-1}{d+k-1} \cdot \frac{U-D+k}{U-d+1} \right)$ where $r(d) = \mathcal{O}(\log^2 n)$.

Proof. This follows from the fact that both $r_{left}(d)$ and $r_{right}(d)$ are $\mathcal{O}(\log^2 n)$. \square

► **Corollary 15.** The probability p_d of sampling d as the number of up-steps before the first approach to the boundary can be approximated as:

$$p_d = \Theta \left(\frac{2^{U+D} \cdot \frac{(k-1)(U-D+k)}{\sqrt{d(U+D-2d-k)(d+k-1)(U-d+1)}} \cdot e^{-\lfloor r(d) \rfloor}}{\binom{U+D}{U} - \binom{U+D}{D-k}} \right)$$

This is because the floor function only affects the value of the exponential by a factor of at most e .

D_{total} is not correct

► **Corollary 16.** We define a piecewise continuous function

$$\hat{q}(d) = \frac{2^{U+D} \cdot \frac{(k-1)(U-D+k)}{\sqrt{d(U+D-2d-k)(d+k-1)(U-d+1)}} \cdot e^{-\lfloor r(d) \rfloor}}{\binom{U+D}{U} - \binom{U+D}{D-k}}$$

We claim that $p_d = \Theta \left(\int_d^{d+1} \hat{q}(d) \right)$. Note that this integral has a closed form for a fixed value of $\lfloor r(d) \rfloor$.

Let the maximum value of $r(d)$ be $r_{max} = \mathcal{O}(\log^2 n)$.

► **Corollary 17.** We can compute the approximate normalized probabilities

$$q_d = \frac{\int_d^{d+1} \hat{q}(d)}{\int_1^n \hat{q}(d)}$$

such that $p_d = \Theta(q_d)$. Furthermore, we can also compute the CDF of q_d as:

$$Q_d = \frac{\int_1^d \hat{q}(d)}{\int_1^n \hat{q}(d)}$$

This allows us to sample from the distribution q_d and use Lemma 1 to indirectly sample from p_d .

Needs a theorem

► **Theorem 18.** Kinesthetics

1.3.5 Finding the Correct Interval: First-Return Query

As before, consider all positions that have been queried already $\langle x_1, x_2, \dots, x_m \rangle$ (in increasing order) along with their corresponding heights $\langle h_1, h_2, \dots, h_m \rangle$.



► **Lemma 19.** *For any position x_i , assuming that Proposition 11 holds, we can find the interval $(x_{k-1}, x_k]$ that contains $\text{FIRST-RETURN}(x_i)$. We do this by setting k to be either the smallest index $f > i$ such that $h_f \leq h_i$ or setting $k = i + 1$.*

Proof. We assume the contrary i.e. there exists some $k \neq f$ and $k \neq i + 1$ such that the correct interval is $(x_{k-1}, x_k]$. Since $h_f < h_i$, the position of first return to h_i happens in the range $(x_i, x_f]$. So, the only possibility is $i + 1 < k \leq f - 1$. By the definition of h_f , we know that both h_k and h_{k-1} are strictly larger than h_i . Proposition 11 implies that the boundary for this interval $(h_{k-1}, h_k]$ is at $\min(h_{k-1}, h_k) > h_i$. So, it is not possible for the first return to be in this interval. \square

The good news is that there are only two intervals that we need to worry about. Now the challenge is to find the smallest index $f > i$ such that $h_f \leq h_i$. One solution is to maintain an interval tree over the range $[2n]$ storing the position of the boundary. Specifically, we have a balanced binary tree with $2n$ leaves with the i^{th} leaf storing the boundary at position i . Each internal node stores the minimum value amongst all the leaves in its sub-tree. In this setting, we can binary search for f by guessing a bound f' and performing a *range minimum query* over the interval $(x_i, x_{f'}]$. Overall, this requires $\mathcal{O}(\log n)$ range queries each of which makes $\mathcal{O}(\log n)$ probes to the binary tree.

However, we cannot explicitly maintain or even construct this tree, and updates can be as expensive as $\Theta(n)$. To mitigate this, we start with just a root node (indicating that the initial boundary is 1 everywhere) and build the tree dynamically as needed. We perform updates using *lazy propagation* by only propagating updates down to the children (creating children if necessary) when needed. So, at any given time, some nodes in the tree may not hold the correct value, but the correct value must be present on the path to the root.

► **Theorem 20.** *There is an algorithm using $\mathcal{O}(\text{poly}(\log n))$ resources per query that provides sample access to a Dyck path of length $2n$ by answering queries of the form $\text{FIRST-RETURN}(x_i)$ with the correctly sampled position y ; where $y > x_i$ is the position where the Dyck path first returns to $\text{HEIGHT}(x_i)$ after position x_i .*

Proof. We first query the interval $(x_i, x_{i+1}]$ to find a first return using Theorem 18. If a return is not found, we calculate f using . Since $x_{f-1} < x_i \leq x_f$ by definition, the interval $(x_{f-1}, x_f]$ must contain a position at height h_i . We sample a point in the middle of this interval and fix the boundary invariant by sampling another point, essentially breaking it up into $\mathcal{O}(1)$ sub-intervals each at most half the size of the original. Based on the new samples, we find the sub-interval containing the first return in $\mathcal{O}(1)$ time. We repeat up to $\mathcal{O}(\log n)$ times until the current interval size drops below the threshold \mathcal{T} . Then we spend $\tilde{\mathcal{O}}(\mathcal{T})$ time to brute force sample this interval and find the first return position (if it wasn't revealed in previous steps). \square

2 Random Coloring of a Graph

We wish to locally sample an uniformly random coloring of a graph. A q -coloring of a graph $G = (V, E)$ is a function $\sigma : V \rightarrow [q]$, such that for all $(u, v) \in E$, $\sigma_u \neq \sigma_v$. We will consider only bounded degree graphs, i.e. graphs with max degree $\leq \Delta$. Otherwise, the coloring problem becomes NP-hard.

Using the technique of path-coupling, Vigoda showed that for $q > 2\Delta$, one can sample an uniformly random coloring by using a MCMC algorithm.

The Markov Chain proceeds in T steps. The state of the chain at time t is given by $\mathbf{X}^t \in [q]^{|V|}$. Specifically, the color of vertex v at step t is \mathbf{X}_v^t .

In each step of the Markov process, a pair $(v, c) \in V \times [q]$ is sampled uniformly at random. Subsequently, if the recoloring of vertex v with color c does not result in a conflict with v 's neighbors, i.e. $c \notin \{X_u^t : u \in \Gamma(v)\}$, then the vertex is recolored i.e. $X_v^{t+1} \leftarrow c$.

After running the MC for $T = \mathcal{O}(n \log n)$ steps we reach the stationary distribution (ϵ close), and the coloring is an uniformly random one.

Exact Bound: $t_{mix}(\epsilon) \leq \left(\frac{q-\Delta}{q-2\Delta}\right) n (\log n + \log(1/\epsilon))$

cite book
(Peres, Lyons)

2.1 Modified Glauber Dynamics

Now we define a modified Markov Chain that proceeds in epochs. We denote the initial coloring of the graph by \mathbf{X}^0 and the state of the coloring after the k^{th} epoch by \mathbf{X}^k . In the k^{th} epoch \mathcal{E}_k :

- Sample $n = |V|$ colors $\langle c_1, c_2, \dots, c_n \rangle$ from $[q]$, where c_v is the proposed color for vertex v .
- For each vertex v , we set \mathbf{X}_v^k to c_v if for all neighbors w of v , $\mathbf{X}_w^k \neq c_v$ and $\mathbf{X}_w^{k-1} \neq c_v$.

This procedure is a special case of the *Local Glauber Dynamics* presented in [FG18]. The goal in [FG18] is to find a simultaneous update rule that causes few conflicts among neighbors (and converges to the correct distribution). Notice that we can have adjacent nodes update in the same epoch, and this can also be implemented locally using . However for the sake of succinctness and because this would only improve a constant factor in the exponent, we use their update rule and avoid a tedious path coupling argument.

what?

For the path coupling argument, we define the standard pre-metric on the space for all possible colorings (not necessarily valid ones). Given two colorings X and Y , we define $d(X, Y)$ as the number of vertices v such that $X_v \neq Y_v$.

Wording: a vertex needs to avoid $\leq 2\Delta$ colors in order to be accepted

We define a coupling $(X, Y) \rightarrow (X', Y')$ where X and Y differ only at a single vertex v such that $X_v = c_X$ and $Y_v = c_Y$. Now, we pick a random permutation of the vertices along with uniformly sampled colors:

$$\langle (v_1, c_1), (v_2, c_2), \dots, (v_n, c_n) \rangle = \langle (\pi_1, c_1), (\pi_2, c_2), \dots, (\pi_n, c_n) \rangle$$

Cite Path Coupling

Now, for each (v_i, c_i) in order, we update the coloring of X and Y as follows:

- If the current color of v_i as well as c_i is in $\{c_X, c_Y\}$, then the X chain picks the color c_i and the Y chain picks the other color.

► **Lemma 21.** If $q = 2\alpha\Delta$ and $d(X, Y) = 1$, then $\mathbb{E}[d(X', Y')] \leq 1 - \left(1 - \frac{1}{2\alpha}\right) e^{-3/\alpha} + \frac{1/2\alpha}{1-1/\alpha}$

► **Corollary 22.** If $q \geq 9\Delta$ and $d(X, Y) = 1$, then $\mathbb{E}[d(X', Y')] < \frac{1}{e^{1/3}}$

► **Theorem 23.** If $q \geq 9\Delta$, then the Markov Chain is mixed after $\tau_{mix}(\epsilon) = 3 \ln n + 3 \ln(1/\epsilon)$ epochs.

Proof.

□

Write proof

2.2 Local Coloring Algorithm

Given a vertex v , the local-access generator has to output the color of v after running $t \leq 2 \ln n$ epochs of *Modified Glauber Dynamics*. We will define the number of colors as $q = \alpha\Delta$ where $\alpha > 1$.

Each epoch is a vector of color samples $\mathbf{C}^i \sim_{\mathcal{U}} [q]^n$. Note that these values are fully independent and as such any \mathbf{C}_v^i can be sampled trivially. We also use \mathbf{X}^i to denote the

final vector of vertex colors at the end of the i^{th} epoch. Finally, we define indicator variables χ_v^i to denote if the color for vertex v was accepted at the i^{th} epoch; $\chi_v^i = 1$ if and only if for all neighbors $w \in \Gamma(v)$, we satisfy the condition $\mathbf{C}_v^i \neq \mathbf{X}_w^{i-1}$ and $\mathbf{C}_v^i \neq \mathbf{C}_w^i$. So, the color of a vertex v after the t^{th} epoch \mathbf{X}_v^t is set to be \mathbf{C}_v^i where $i \leq t$ is the largest index such that $\chi_v^i = 1$. Algorithm 1 shows the procedure for querying the value of \mathbf{X}_v^t .

Algorithm 1 Generator

```

1: procedure COLOR( $v, t$ )
2:   for  $i \leftarrow [t, t-1, t-2 \dots 1]$  do
3:     if ACCEPTED( $i, v$ ) then
4:       return  $\mathbf{C}_v^i$ 
5: procedure ACCEPTED( $v, t$ )
6:    $c \leftarrow \mathbf{C}_v^t$ 
7:   for  $w \leftarrow \Gamma(v)$  do
8:      $flag \leftarrow 0$ 
9:     for  $t' \leftarrow [t, t-1, t-2, \dots, 1]$  do
10:      if  $t' \neq t$  or  $\mathcal{I}_w^{(t)} < \mathcal{I}_v^{(t)}$  then
11:        if  $\mathbf{C}_w^{(t')} = c$  and ACCEPTED( $w, t'$ ) then
12:           $flag \leftarrow 1$ 
13:          while  $t' < t$  do
14:             $t' \leftarrow t' + 1$ 
15:            if ACCEPTED( $w, t'$ ) then
16:               $flag \leftarrow 0$ 
17:              break
18:          if  $flag = 1$  then
19:            return 0
20:          break
21:   return 1

```

When the algorithm is asked for the final color of v , it finds the last epoch in which v was accepted. Since there are only $\mathcal{O}(\log n)$ epochs, we focus our attention on the subroutine ACCEPTED that samples the value of χ_v^t . The algorithm iterates through the neighbors w of v , and check for conflicts with the proposed color $c = \mathbf{C}_v^t$. The condition $c \neq \mathbf{C}_w^t$ can easily be checked by sampling \mathbf{C}_w^t in the current epoch. If no conflict is seen, the next step is to check whether $c \neq \mathbf{X}_w^{t-1}$.

We first iterate through all the epochs in reverse order to check whether the color c was ever proposed for vertex w . If not, we can ignore w , and otherwise let's say that the last proposal for c was at epoch t' i.e. $\mathbf{C}_w^{t'} = c$. Now, we need to recursively check if this proposal was ACCEPTED. If it was, we move to epoch $t' + 1$ to see if w 's color was replaced. If not, we check epoch $t' + 2$ and so on until we reach epoch $t - 1$. At this point we have seen that $\chi_w^{t'} = 1$ (color c was accepted) and every subsequent proposal until the current epoch was rejected i.e. $\mathbf{X}_w^{t-1} = c$ and this leads to a conflict with v 's current proposal for color c and hence $\chi_v^t = 0$. If at any of the iterations, we see that a different proposal was accepted, then w does not cause a conflict and we can move on to the next neighbor. If we exhaust all the neighbors and don't find any conflicts then $\chi_v^t = 1$.

► **Lemma 24.** *The probability that any given proposal is rejected $\mathbb{P}[\chi_v^t = 0]$ is at most $1/\alpha$. Moreover, this upper bound holds even if we condition on all the values in \mathbf{C} except \mathbf{C}_v^t .*

Proof. A rejection can occur due to a conflict with at most 2Δ possible values in $\{C_w^t, X_w^{t-1} | w \in \Gamma(v)\}$. Since there are $2\alpha\Delta$ colors, the rejection probability is at most $1/\alpha$. \square

So, the number of probes required to check whether a color c (assigned at epoch t') was overwritten at some epoch before t is:

$$\left[T_{t'+1} + \mathcal{B}\left(\frac{1}{\alpha}\right) \cdot T_{t'+2} + \mathcal{B}\left(\frac{1}{\alpha^2}\right) \cdot T_{t'+3} + \cdots + \mathcal{B}\left(\frac{1}{\alpha^{t-t'-2}}\right) \cdot T_{t-1} \right] \quad (2)$$

► **Lemma 25.** For $\alpha > 4$, the expected time to sample a single χ_v^t is $\mathbb{E}[T_t] = \mathcal{O}(t\Delta e^{2t/\alpha})$.

Proof. We formulate a recurrence for the expected number of probes to $\{\chi^{t'}\}_{t' \in [t]}$ used by the algorithm. We will use $\mathcal{B}(p)$ to refer to the Bernoulli random variable with bias p . When checking a single neighbor w , the algorithm iterates through all the epochs t' such that $C_w^{t'} = c$ (technically, only the last occurrence matters, but we are looking for an upper bound). If such a t' is found (this happens with probability $1/q$ independently for each trial), there is one recursive call to $T_{t'}$. Regardless of what happens, let's say the algorithm queries $T_{t'+1}, T_{t'+2}, \dots, T_{t-1}$ until an **ACCEPTED** proposal is found. Adding an extra $T_{t'}$ term to Equation 2 and summing up over all neighbors and epochs we get the following:

$$T_t \leq \Delta \cdot \sum_{t'=1}^t \mathbb{P}[C_w^{t'} = c] \cdot \left[T_{t'} + T_{t'+1} + \mathcal{B}\left(\frac{1}{\alpha}\right) \cdot T_{t'+2} + \mathcal{B}\left(\frac{1}{\alpha^2}\right) \cdot T_{t'+3} + \cdots \right] \quad (3)$$

$$\cdots + \mathcal{B}\left(\frac{1}{\alpha^{t-t'-2}}\right) \cdot T_{t-1} \right] \quad (4)$$

$$\leq \Delta \cdot \mathcal{B}\left(\frac{1}{q}\right) \left[\sum_{t'=1}^{t-1} T_{t'} + \sum_{t'=1}^{t-1} T_{t'} \cdot \left(1 + \mathcal{B}\left(\frac{1}{\alpha}\right) + \mathcal{B}\left(\frac{1}{\alpha^2}\right) + \cdots \right) \right] \quad (5)$$

In the second step, we just group all the terms from the same epoch together. Using Lemma 24 and the fact that $\mathbb{P}[C_w^{t'} = c]$ is independent of all other events, we can write a recurrence for the expected number of probes.

$$\mathbb{E}[T_t] \leq \Delta \cdot \frac{1}{\alpha\Delta} \left[\sum_{t'=1}^{t-1} T_{t'} + \sum_{t'=1}^{t-1} T_{t'} \cdot \left(1 + \frac{1}{\alpha} + \frac{1}{\alpha^2} + \cdots \right) \right] \quad (6)$$

Now, we make the assumption that $\mathbb{E}[T_{t'}] \leq e^{2t'/\alpha}$ and show that this satisfies the expectation recurrence. First, we sum the geometric series:

$$\sum_{t'=1}^{t-1} \mathbb{E}[T_{t'}] = \sum_{t'=1}^{t-1} e^{2t'/\alpha} < \frac{e^{2t/\alpha} - 1}{e^{2/\alpha} - 1} < \frac{e^{2t/\alpha}}{e^{2/\alpha} - 1}$$

The expectation recurrence to be satisfied then becomes:

$$\mathbb{E}[T_t] \leq e^{2t/\alpha} \cdot \frac{1}{\alpha} \cdot \frac{1}{e^{2/\alpha} - 1} \cdot \left[1 + \frac{\alpha}{\alpha - 1} \right] = e^{2t/\alpha} \cdot \frac{2\alpha - 1}{\alpha(\alpha - 1)(e^{2/\alpha} - 1)} = e^{2t/\alpha} \cdot f(\alpha)$$

We notice that $f(\alpha)$ is upper bounded by 1 in the domain $\alpha > 4$ (in fact $\lim_{\alpha \rightarrow \infty} f(\alpha) = 1$). This can easily be verified by taking the derivatives and using L'Hôpital's rule. Thus, our recurrence is satisfied. Finally, we note that each probe potentially takes time $\mathcal{O}(t\Delta)$ to iterate through all the neighbors in all epochs resulting in a total runtime of $\mathcal{O}(t\Delta e^{2t/\alpha})$. \square

► **Corollary 26.** *Instead of looking through all the epochs in order, we can use the coloring generator to find the locations directly.*

where?

► **Theorem 27.** *Given adjacency list query access to a graph with n nodes, maximum degree Δ , and $q = 2\alpha\Delta$ colors, we can sample the color of any given node in an $(1/n)$ -approximate uniformly random coloring of the graph in a consistent manner using only $\mathcal{O}(n^{12/\alpha}\Delta \log n)$ time space and random bits. This is sublinear for $\alpha > 12$ and the sampled coloring is $1/n$ -close to the uniform distribution in L_1 distance.*

Proof. We use the mixing time from Theorem 23 $\tau_{mix}(1/n) = 6 \ln n$ in conjunction with Lemma 25. So, the overall runtime becomes $\mathcal{O}(n^{12/\alpha}\Delta \log n)$. \square

References

- FG18** Manuela Fischer and Mohsen Ghaffari. A simple parallel and distributed sampling technique: Local glauher dynamics. In *32nd International Symposium on Distributed Computing (DISC 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.
- GGN10** Oded Goldreich, Shafi Goldwasser, and Asaf Nussboim. On the implementation of huge random objects. *SIAM Journal on Computing*, 39(7):2761–2822, 2010.
- Reu14** Shlomi Reuveni. Catalan’s trapezoids. *Probability in the Engineering and Informational Sciences*, 28(03):353–361, 2014.
- Spe14** Joel Spencer. *Asymptopia*, volume 71. American Mathematical Soc., 2014.
- Sta15** Richard P Stanley. *Catalan numbers*. Cambridge University Press, 2015.

A Dyck Path Generator

► **Theorem 28.** *There are $\frac{1}{n+1} \binom{2n}{n}$ Dyck paths for length $2n$ (construction from [Sta15]).*

Proof. Consider all possible sequences containing $n+1$ up-steps and n down-steps with the restriction that the first step is an up-step. We say that two sequences belong to the same *class* if they are cyclic shifts of each other. Because of the restriction, the total number of sequences is $\binom{2n}{n}$ and each class is of size $n+1$. Now, within each class, exactly one of the sequences is such that the prefix sums are *strictly greater* than zero. From such a sequence, we can obtain a Dyck sequence by deleting the first up-step. Similarly, we can start with a Dyck sequence, add an initial up-step and consider all $n+1$ cyclic shifts to obtain a *class*. This bijection shows that the number of Dyck paths is $\frac{1}{n+1} \binom{2n}{n}$. \square

A.1 Approximating Close-to-Central Binomial Coefficients

We start with Stirling's approximation which states that

$$m! = \sqrt{2\pi m} \left(\frac{m}{e}\right)^m \left(1 + \mathcal{O}\left(\frac{1}{m}\right)\right)$$

We will also use the logarithm approximation when a better approximation is required:

$$\log(m!) = m \log m - m + \frac{1}{2} \log(2\pi m) + \frac{1}{12m} - \frac{1}{360m^3} + \frac{1}{1260m^5} - \dots \quad (7)$$

This immediately gives us an asymptotic formula for the central binomial coefficient as:

► **Lemma 29.** *The central binomial coefficient can be approximated as:*

$$\binom{n}{n/2} = \sqrt{\frac{2}{\pi n}} 2^n \left(1 + \mathcal{O}\left(\frac{1}{n}\right)\right)$$

Now, we consider a “off-center” Binomial coefficient $\binom{n}{k}$ where $k = \frac{n+c\sqrt{n}}{2}$.

Cite Asymptopia

► **Lemma 30.** *Proof from [Spe14]*

$$\binom{n}{k} = \binom{n}{n/2} e^{-c^2/2} \exp(\mathcal{O}(c^3/\sqrt{n}))$$

Proof. We consider the ratio: $R = \binom{n}{k} / \binom{n}{n/2}$:

$$R = \frac{\binom{n}{k}}{\binom{n}{n/2}} = \frac{(n/2)!(n/2)!}{k!(n-k)!} = \prod_{i=1}^{c\sqrt{n}/2} \frac{n/2 - i + 1}{n/2 + i} \quad (8)$$

$$\Rightarrow \log R = \sum_{i=1}^{c\sqrt{n}/2} \log \left(\frac{n/2 - i + 1}{n/2 + i} \right) \quad (9)$$

$$= \sum_{i=1}^{c\sqrt{n}/2} -\frac{4i}{n} + \mathcal{O}\left(\frac{i^2}{n^2}\right) = -\frac{c^2 n}{2} + \mathcal{O}\left(\frac{(c\sqrt{n})^3}{n^2}\right) = -\frac{c^2}{2} + \mathcal{O}\left(\frac{c^3}{\sqrt{n}}\right) \quad (10)$$

$$\Rightarrow \binom{n}{k} = \binom{n}{n/2} e^{-c^2/2} \exp(\mathcal{O}(c^3/\sqrt{n})) \quad (11)$$

\square

A.2 Dyck Path Boundaries and Deviations

► **Lemma 31.** *Given a random walk of length $2n$ with exactly n up and down steps, consider a contiguous sub-path of length $2B$ that comprises of U up-steps and D down-steps i.e. $U + D = 2B$. Both $|B - U|$ and $|B - D|$ are $\mathcal{O}(\sqrt{B \log n})$ with probability at least $1 - 1/n^4$.*

Proof. We consider the random walk as a sequence of unbiased random variables $\{X_i\}_{i=1}^{2n} \in \{0, 1\}^{2n}$ with the constraint $\sum_{i=1}^{2n} X_i = n$. Here, 1 corresponds to an up-step and 0 corresponds to a down step. Because of the constraint, X_i, X_j are negatively correlated for $i \neq j$ which allows us to apply Chernoff bounds. Now we consider a sub-path of length $2B$ and let U denote the sum of the X_i s associated with this subpath. Using Chernoff bound with $\mathbb{E}[X] = B$, we get:

$$\mathbb{P}\left[|U - B| < 3\sqrt{B \log n}\right] = \mathbb{P}\left[|U - B| < 3\frac{\sqrt{\log n}}{\sqrt{B}}B\right] < e^{\frac{9 \log n}{3}} \approx \frac{1}{n^3}$$

Since U and D are symmetric, the same argument applies. \square

► **Corollary 32.** *With high probability, every contiguous sub-path in the random walk (with U up and D down steps) satisfies the property with high probability. Specifically, if $U + D = 2B$, then $|B - U|$ and $|B - D|$ are upper bounded by $c\sqrt{B \log n}$ w.h.p. $1 - 1/n^2$ for all contiguous sub-paths (for some constant c).*

Proof. We can simply apply Lemma 31 and union bound over all n^2 possible contiguous sub-paths. \square

► **Lemma 2.** *Consider a contiguous sub-path of a simple Dyck path of length $2n$ where the sub-path is of length $2B$ comprising of U up-steps and D down-steps (with $U + D = 2B$). Then there exists a constant c such that the quantities $|B - U|$, $|B - D|$, and $|U - D|$ are all $< c\sqrt{B \log n}$ with probability at least $1 - 1/n^2$ for every possible sub-path.*

Proof. As a consequence of Theorem 28, we can sample a Dyck path by first sampling a *balanced* random walk with n up steps and n down steps and adding an initial up step. We can then find the corresponding Dyck path by taking the unique cyclic shift that satisfies the Dyck constraint (after removing the initial up-step). Any interval in a cyclic shift is the union of at most two intervals in the original sequence. This affects the bound only by a constant factor. So, we can simply use Corollary 32 to finish the proof. Notice that since $|U - D| \leq |B - U| + |B - D|$, $|U - D| = \mathcal{O}(\sqrt{B \log n})$ comes for free. \square

► **Lemma 3.** *There exists a constant c such that if $k > c\sqrt{B \log n}$, then the distribution of paths sampled without a boundary (hypergeometric sampling) is statistically $\mathcal{O}(1/n^2)$ -close in L_1 distance to the distribution of Dyck paths.*

Proof. We use \mathcal{D} and \mathcal{R} to denote the set of all valid Dyck paths and all random sequences respectively. Clearly, $\mathcal{D} \subseteq \mathcal{R}$. Let c be a constant satisfying Corollary 32. Since the random walk/sequence distribution is uniform on \mathcal{R} , and by Corollary 32 we see that at least $1 - 1/n^2$ fraction of the elements of \mathcal{R} do not violate the boundary constraint. Therefore, $|\mathcal{D}| \geq (1 - 1/n^2)|\mathcal{R}|$ and so the L_1 distance between $\mathcal{U}_{\mathcal{D}}$ and $\mathcal{U}_{\mathcal{R}}$ is $\mathcal{O}(1/n^2)$. \square

A.3 Computing Probabilities

Oracle for estimating probabilities:

► **Lemma 33.** *Given a Dyck sub-path problem within a global Dyck path of size $2n$ and a probability expression of the form $p_d = \frac{D_{left} \cdot D_{right}}{D_{total}}$, there exists a $\text{poly}(\log n)$ time oracle that returns a $(1 \pm 1/n^2)$ multiplicative approximation to p_d if $p_d = \Omega(1/n^2)$ and returns 0 otherwise.*

Proof. We first compute a $1 + 1/n^3$ multiplicative approximation to $\ln p_d$. Using $\mathcal{O}(\log n)$ terms of the series in Equation 7, it is possible to estimate the logarithm of a factorial up to $1/n^c$ additive error. So, we can use the series expansion from Equation 7 up to $\mathcal{O}(\log n)$ terms. The additive error can also be cast as multiplicative since factorials are large positive integers.

The probability p_d can be written as an arithmetic expression involving sums and products of a constant number of factorial terms. Given a $1 \pm 1/n^c$ multiplicative approximation to $l_a = \ln a$ and $l_b = \ln b$, we wish to approximate $\ln(ab)$ and $\ln(a+b)$. The former is trivial since $\ln(ab) = \ln a + \ln b$. For the latter, we assume $a > b$ and use the identity $\ln(a+b) = \ln a + \ln(1+b/a)$ to note that it suffices to approximate $\ln(1+b/a)$. We define $\hat{l}_a = l_a \cdot (1 \pm \mathcal{O}(1/n^c))$ and $\hat{l}_b = l_b \cdot (1 \pm \mathcal{O}(1/n^c))$. In case $\hat{l}_b - \hat{l}_a < c \ln n \implies b/a < 1/n^c$, we approximate $\ln(a+b)$ by $\ln a$ since $\ln(1+b/a) = \mathcal{O}(1/n^c)$ in this case. Otherwise, using the fact that $l_a - l_b = o(n^2)$, we compute:

$$1 + e^{\hat{l}_b - \hat{l}_a} = 1 + \frac{b}{a} \cdot e^{\mathcal{O}(\frac{l_b - l_a}{n^c})} = 1 + \frac{b}{a} \cdot \left(1 \pm \mathcal{O}\left(\frac{1}{n^{c-2}}\right)\right) = \left(1 + \frac{b}{a}\right) \cdot \left(1 \pm \mathcal{O}\left(\frac{1}{n^{c-2}}\right)\right)$$

In other words, the value of c decreases every time we have a sum operation. Since there are only a constant number of such arithmetic operations in the expression for p_d , we can set c to be a high enough constant (when approximating the factorials) and obtain the desired $1 \pm 1/n^3$ approximation to $\ln p_d$. If $\ln p_d < -3 \ln n$, we approximate $p_d = 0$. Otherwise, we can exponentiate the approximation to obtain $p_d \cdot e^{-\mathcal{O}(\ln n/n^3)} = p_d (1 \pm \mathcal{O}(1/n^2))$. \square

A.4 Sampling the Height

fix

- $d < c \cdot \sqrt{B} \log n$
- $k < c \cdot \sqrt{B} \log n \implies U - D < c \cdot \sqrt{B} \log n$
- $k' < c \cdot \sqrt{B} \log n$
- $B > \log^2 n \implies \sqrt{B} \log n < B$

► **Lemma 34.** *For $x < 1$ and $k \geq 1$,*

$$1 - kx < (1 - x)^k < 1 - kx + \frac{k(k-1)}{2} x^2.$$

► **Lemma 5.** $D_{left} \leq c_1 \frac{k \cdot \log n}{\sqrt{B}} \cdot \binom{B}{D-d}$ for some constant c_1 .

Proof. This involves some simple manipulations.

$$D_{left} = \binom{B}{D-d} - \binom{B}{D-d-k} \quad (12)$$

$$= \binom{B}{D-d} \cdot \left[1 - \frac{(D-d)(D-d-1) \cdots (D-d-k+1)}{(B-D-d+k)(B-D-d+k-1) \cdots (B-D-d+1)} \right] \quad (13)$$

$$\leq \binom{B}{D-d} \cdot \left[1 - \left(\frac{D-d-k+1}{B-D-d+k} \right)^k \right] \quad (14)$$

$$\leq \binom{B}{D-d} \cdot \left[1 - \left(\frac{U+d+k-(U-D+d+k-1)}{U+d+k} \right)^k \right] \quad (15)$$

$$\leq \binom{B}{D-d} \cdot \left[1 - \left(\frac{U+d+k-\mathcal{O}(\log n \sqrt{B})}{U+d+k} \right)^k \right] \quad (16)$$

$$\leq \Theta\left(\frac{k \log n}{\sqrt{B}}\right) \cdot \binom{B}{D-d} \quad (17)$$

□

► **Lemma 6.** $D_{right} < c_2 \frac{k' \log n}{\sqrt{B}} \cdot \binom{B}{U-d}$ for some constant c_2 .

Proof.

$$D_{right} = \binom{B}{U-d} - \binom{B}{U-d-k'} \quad (18)$$

$$= \binom{B}{U-d} \cdot \left[1 - \frac{(U-d)(U-d-1) \cdots (U-d-k'+1)}{(B-U-d+k')(B-U-d+k'-1) \cdots (B-U-d+1)} \right] \quad (19)$$

$$\leq \binom{B}{U-d} \cdot \left[1 - \left(\frac{U-d-k'+1}{B-U-d+k'} \right)^{k'} \right] \quad (20)$$

$$\leq \binom{B}{U-d} \cdot \left[1 - \left(\frac{2D-U-d-k+1}{2U-D+k+d} \right)^{k'} \right] \quad (21)$$

$$\leq \binom{B}{U-d} \cdot \left[1 - \left(\frac{U+k+d-(2U-2D+2d+2k-1)}{U+k+d} \right)^{k'} \right] \quad (22)$$

$$\leq \binom{B}{U-d} \cdot \left[1 - \left(\frac{U+k+d-\mathcal{O}(\log n \sqrt{B})}{U+k+d} \right)^{k'} \right] \quad (23)$$

$$\leq \Theta\left(\frac{k' \log n}{\sqrt{B}}\right) \cdot \binom{B}{U-d} \quad (24)$$

□

change state-
ment

► **Lemma 35.** $D_{tot} \geq \binom{2B}{2D} \cdot \left[1 - \left(1 - \frac{k'}{2U+1} \right)^k \right]$.

Proof.

$$D_{tot} = \binom{2B}{2D} - \binom{2B}{2D-k} \quad (25)$$

$$= \binom{2B}{2D} \cdot \left[1 - \frac{(2D)(2D-1) \cdots (2D-k+1)}{(2B-2D+k)(2B-2D+k-1) \cdots (2B-2D+1)} \right] \quad (26)$$

$$\geq \binom{2B}{2D} \cdot \left[1 - \left(\frac{2D-k+1}{2B-2D+1} \right)^k \right] \quad (27)$$

$$\geq \binom{2B}{2D} \cdot \left[1 - \left(\frac{2U - (2U-2D+k-1)}{2U+1} \right)^k \right] \quad (28)$$

$$\geq \binom{2B}{2D} \cdot \left[1 - \left(\frac{(2U+1)-k'}{2U+1} \right)^k \right] \quad (29)$$

$$\geq \binom{2B}{2D} \cdot \left[1 - \left(1 - \frac{k'}{2U+1} \right)^k \right] \quad (30)$$

$$(31)$$

□

Reference previous lemma

► **Lemma 4.** When $kk' > 2U+1$, $D_{tot} > \frac{1}{2} \cdot \binom{2B}{2D}$.

Proof. When $kk' > 2U+1 \implies k > \frac{2U+1}{k'}$, we will show that the above expression is greater than $\frac{1}{2} \binom{2B}{2D}$. Defining $\nu = \frac{2U+1}{k'} > 1$, we see that $(1 - \frac{1}{\nu})^k \leq (1 - \frac{1}{\nu})^\nu$. Since this is an increasing function of ν and since the limit of this function is $\frac{1}{e}$, we conclude that

$$1 - \left(1 - \frac{k'}{2U+1} \right)^k > \frac{1}{2}$$

□

► **Lemma 7.** When $kk' \leq 2U+1$, $D_{tot} < c_3 \frac{k \cdot k'}{B} \cdot \binom{2B}{2D}$ for some constant c_3 .

Proof. Now we bound the term $1 - \left(1 - \frac{k'}{2U+1} \right)^k$, given that $kk' \leq 2U+1 \implies \frac{kk'}{2U+1} \leq 1$. Using Taylor expansion, we see that

$$1 - \left(1 - \frac{k'}{2U+1} \right)^k \quad (32)$$

$$\leq \frac{kk'}{2U+1} - \frac{k(k-1)}{2} \cdot \frac{k'^2}{(2U+1)^2} \quad (33)$$

$$\leq \frac{kk'}{2U+1} - \frac{k^2 k'^2}{2(2U+1)^2} \quad (34)$$

$$\leq \frac{kk'}{2U+1} \left(1 - \frac{kk'}{2(2U+1)} \right) \quad (35)$$

$$\leq \frac{kk'}{2(2U+1)} \leq \frac{kk'}{\Theta(B)} \quad (36)$$

$$(37)$$

□

A.5 First Return Sampling

► **Lemma 12.** *If $d > \log^4 n$, then $D_{left}(d) = \Theta\left(\frac{2^{2d+k}}{\sqrt{d}} e^{-r_{left}(d)} \cdot \frac{k-1}{d+k-1}\right)$ where $r_{left}(d) = \frac{(k-2)^2}{2(2d+k-2)}$.*

Proof. In what follows, we will drop constant factors: Refer to Figure 4 for the setup. The left section of the path reaches one unit above the boundary (the next step would make it touch the boundary). The number of up-steps on the left side is d and therefore the number of down steps must be $d + k - 2$. This includes d down steps to cancel out the upwards movement, and $k - 2$ more to get to one unit above the boundary. The boundary for this section is $k' = k - 1$. This gives us:

$$D_{left}(d) = \binom{2d+k-2}{d} - \binom{2d+k-2}{d-1} \quad (38)$$

$$= \binom{2d+k-2}{d} \left[1 - \frac{d}{d+k-1}\right] = \binom{2d+k-2}{d} \frac{k-1}{d+k-1} \quad (39)$$

Now, letting $z = 2d + k - 2$, we can write $d = \frac{z-(k-2)}{2} = \frac{z-\frac{k-2}{\sqrt{z}}\sqrt{z}}{2}$. Using Lemma 2, we see that $\frac{k-2}{\sqrt{z}}$ should be $\mathcal{O}(\sqrt{\log n})$. If this is not the case, we can simply return 0 because the probability associated with this value of d is negligible. Since $z > \log^4 n$, we can apply Lemma 30 to get:

$$D_{left}(d) = \Theta\left(\left(\frac{z}{z/2}\right) e^{\frac{(k-2)^2}{2z}} \frac{k-1}{d+k-1}\right) = \Theta\left(\frac{2^{2d+k}}{\sqrt{d}} e^{\frac{(k-2)^2}{2(2d+k-2)}} \frac{k-1}{d+k-1}\right)$$

□

► **Lemma 13.** *If $U+D-2d-k > \log^4 n$, then $D_{right}(d) = \Theta\left(\frac{2^{U+D-2d-k}}{\sqrt{U+D-2d-k}} e^{-r_{right}(d)} \cdot \frac{U-D+k}{U-d+1}\right)$ where $r_{right}(d) = \frac{(U-D-k-1)^2}{4(U+D-2d-k+1)}$.*

Proof. The right section of the path starts from the original boundary. Consequently, the boundary for this section is at $k' = 1$. The number of up-steps on the right side is $U - d$ and the number of down steps is $D - d - k + 1$. This gives us:

$$D_{right}(d) = \binom{U+D-2d-k+1}{U-d} - \binom{U+D-2d-k+1}{U-d+1} \quad (40)$$

$$= \binom{U+D-2d-k+1}{U-d} \left[1 - \frac{D-d-k-1}{U-d+1}\right] \quad (41)$$

$$= \binom{U+D-2d-k+1}{U-d} \frac{U-D+k}{U-d+1} \quad (42)$$

Now, letting $z = U + D - 2d - k + 1$, we can write $U - d = \frac{z+(U-D+k-1)}{2} = \frac{z+\frac{U-D+k-1}{\sqrt{z}}\sqrt{z}}{2}$. Using Lemma 2, we see that $\frac{U-D+k-1}{\sqrt{z}}$ should be $\mathcal{O}(\sqrt{\log n})$. If this is not the case, we can simply return 0 because the probability associated with this value of d is negligible. Since $z > \log^4 n$, we can apply Lemma 30 to get:

$$D_{right}(d) = \Theta\left(\left(\frac{z}{z/2}\right) e^{\frac{(U-D+k-1)^2}{2z}} \frac{U-D+k}{U-d+1}\right) \quad (43)$$

$$= \Theta\left(\frac{2^{U+D-2d-k}}{\sqrt{U+D-2d-k}} e^{\frac{(U-D+k-1)^2}{2(U+D-2d-k+1)}} \frac{U-D+k}{U-d+1}\right) \quad (44)$$

□