De Bruijn Sequences with Minimum Discrepancy

Nicolás Álvarez   Verónica Becher   Martín Mereb   Ivo Pajor   Carlos Miguel Soto

July 25, 2024

Abstract

The discrepancy of a binary string is the maximum (absolute) difference between the number of ones and the number of zeroes over all possible substrings of the given binary string. In this note we determine the minimal discrepancy that a binary de Bruijn sequence of order n can achieve, which is n. This was an open problem until now. We give an algorithm that constructs a binary de Bruijn sequence with minimal discrepancy. A slight modification of this algorithm deals with arbitrary alphabets and yields de Bruijn sequences of order n with discrepancy at most 1 above the trivial lower bound n.

Contents

1

1 Statement of Results

A de Bruijn sequence of order n over a b-symbol alphabet is a circular string of length b^n such that every string of length n occurs exactly once in it. The discrepancy of a given string is a non negative integer that indicates the maximum difference, in any substring, between the number of occurrences of the most occurring symbol and the least occurring symbol in that substring. Formally: let Σ be an alphabet and w a string over Σ. Let $|w|_a$ denote the number of occurrences of the symbol a in w. Let $S(w)$ denote the set of all substrings of w where we interpret w as a circular string. The discrepancy of a string w over alphabet Σ is defined as

$$discrepancy(w) = \max_{s \in S(w)} \left(\max_{a \in \Sigma} |s|_a - \min_{c \in \Sigma} |s|_c \right).$$

The maximum and minimum discrepancies attainable by a de Bruijn sequence of order n are $\Theta(2^n/\sqrt{n})$ and $\Theta(n)$ respectively. This was proved by Gabric and Sawada in [3, Theorem 5.3] and [3, Theorem 2.2].

What is the exact minimum discrepancy that a de Bruijn sequence can achieve? Clearly, in every alphabet, the discrepancy of a de Bruijn sequence of order n is at least n, because there must exist a run of n consecutive equal symbols. Here we show that for the binary alphabet the minimum discrepancy achievable by a de Bruijn sequence of order n is exactly n. We give an efficient algorithm to compute it. For alphabets with more than two symbols, our algorithm constructs a de Bruijn sequence of order n with discrepancy at most $n + 1$.

Theorem 1. *There is an algorithm that produces a de Bruijn sequence of order n with discrepancy n in case the alphabet has two symbols, and with discrepancy at most $n + 1$ in case the alphabet has more than two symbols. The algorithm computes in $O(n)$ memory and it outputs each symbol in $O(n)$ time.*

Thus, in the case of the binary alphabet our algorithm yields minimal discrepancy. For larger alphabets we experimentally found that for some values of n there exists a de Bruijn sequence of order n with discrepancy n, but not for others.

Gabric and Sawada in [3] ask whether Huang's [4] algorithm produces binary de Bruijn sequences with minimum discrepancy. The answer is no, but a variant of it does: our algorithm is a variant of Huang's algorithm, twisted to achieve minimum discrepancy. Our algorithm works for any alphabet, thereby, it solves the problem posed by Gabric and Sawada of determining whether the discrepancy bounds for the binary alphabet hold for arbitrary alphabets as well.

2 Structure of the Proof of Theorem 1

The structure of the algorithm and its correctness proof is as follows:

- First, we translate the problem to the language of de Bruijn graphs, using the well known result that Hamiltonian cycles in the de Bruijn graph correspond to de Bruijn sequences [2].

- We define a subgraph of the de Bruijn graph, called the valid subgraph, where all arcs are of a certain form and every path in that graph corresponds to a string with bounded discrepancy.

- We decompose the de Bruijn graph into node-disjoint cycles using the incremented cycle register rule, as is done by Huang [4]. We prove that these cycles are all contained within the valid subgraph.

- We choose a way to connect these cycles in a tree-like structure to form a Hamiltonian cycle, such that the arcs between the cycles are also contained in the valid subgraph.

- Finally, we develop an algorithm to traverse this tree in a depth first manner, completely traversing all cycles and thus producing a Hamiltonian cycle. Since it is contained within the valid subgraph, this guarantees that the resulting Hamiltonian cycle has bounded discrepancy.

3 De Bruijn Graph and Discrepancy

Given an integer n and an alphabet $\Sigma = \{0, 1, \ldots, k-1\}$, where k is a positive integer, the de Bruijn graph $B_n = (V_n, A_n)$ is a graph with node set $V_n = \Sigma^n$, the set of all strings of length n, and an arc $(s, t) \in A_n$ if and only if the suffix of s of length $n-1$ equals the prefix of t of length $n-1$. The alphabet is interpreted modulo k. For example, for $k = 3$ we take $1 + 2 = 0$.

Hamiltonian cycles in the de Bruijn graph correspond to de Bruijn sequences: if $s_0, s_1, \ldots, s_{k-1}$ is a Hamiltonian cycle in B_n, then the sequence

$$s_0[0], s_1[0], \ldots, s_{k-1}[0]$$

consisting of the first symbol of each string is a de Bruijn sequence. Thus, each string of length n appears exactly once in the sequence when viewed cyclically.

Definition (Difference). *The discrepancy of a string w is defined as, discrepancy : $\Sigma^* \to \mathbb{Z}$*

$$discrepancy(w) = \max_{s \in S(w)} \left(\max_{a \in \Sigma} |s|_a - \min_{c \in \Sigma} |s|_c \right).$$

where the value $\max_{a \in \Sigma} |s|_a - \min_{c \in \Sigma} |s|_c$ is the difference of the substring s, denoted $D(s)$.

In order to bound the discrepancy of de Bruijn sequences, we need to bound the difference of all its substrings. A substring of a de Bruijn sequence is a sequence of the form

$$s_i[0], s_{i+1}[0], \ldots, s_j[0]$$

where $s_i, s_{i+1}, \ldots, s_j$ is a contiguous subsequence of the Hamiltonian cycle. Therefore, substrings of a de Bruijn sequence correspond to paths in the de Bruijn graph.

4 Valid Subgraph

4.1 Histograms

Let us define a concept that will help us to calculate the difference of a string.

Definition (Histogram). *Given a string $s \in \Sigma^*$ let $H(s) : \Sigma \to \mathbb{Z}$ be defined as $H(s)(c) = |s|_c$.*

The difference of a histogram is defined as follows.

Definition (Difference of a histogram)**.** *The difference $D(H)$ of a histogram $H : \Sigma \to \mathbb{Z}$ is given by*

$$D(H) = \max_{i,j \in \Sigma} H(i) - H(j).$$

As expected, $D(H(s)) = D(s)$. We use the same symbol to denote the difference of a string and the difference of a histogram. However, the meaning of the symbol will be clear from the context in which it is used. Another operation that we need on histograms is the *partial sum*.

Definition (Partial sum of a histogram)**.** *Given $H : \Sigma \to \mathbb{Z}$, define its partial sum $P(H) : \Sigma \to \mathbb{Z}$ as*

$$P(H)(i) = \sum_{j=0}^{i} H(j).$$

Let K be the constant one histogram. That is, $K : \Sigma \to \mathbb{Z}$ such that $K(b) = 1$ for all $b \in \Sigma$. Notice that if we add K to a histogram, the difference is not affected. Therefore, we can consider the histograms in the quotient space

$$\frac{\Sigma \to \mathbb{Z}}{\langle K \rangle}.$$

4.2   Incremented Cycle Register Rule

Definition (Incremented Cycle Register Rule)**.** *Given a positive integer k, we define $ICR_k : \Sigma^n \to \Sigma^n$ as given by*

$$ICR_k(s) = s[1]s[2]\ldots s[n-1](s[0]+k).$$

Then, $ICR_k(s)$ is the string consisting of a cyclic rotation of the original s, but where the last symbol is incremented by k. Unless otherwise specified, $ICR = ICR_1$.

Consider the subgraph of B_n given by $(\Sigma^n, \{(s, ICR(s)) : s \in \Sigma^n\})$, which we call the ICR-subgraph of B_n. Since ICR is a bijective function, the ICR-subgraph consists of a collection of node-disjoint cycles.

Consider a path $s_0, s_1, \ldots, s_k$ in the ICR-subgraph. Can we reconstruct the histogram of first symbols of these strings knowing only s_0 and s_k?

Lemma 1. *Let $s \in \Sigma^n$ and let $b = s[0]$ be the first symbol of s, then*

$$P(H(s)) - P(H(ICR(s))) \equiv e_b \mod K,$$

where e_b is the indicator function at b.

Proof. By definition, $H(s) - H(ICR(s)) = e_b - e_{b+1}$. If $b + 1 \neq 0$, then $P(e_b - e_{b+1}) = e_b$ and by linearity of P we are done. If $b + 1 = 0$, then $P(e_b - e_{b+1}) = e_b - K$. $\qquad\square$

Lemma 2. *Let $s_0, \ldots, s_k$ be a sequence of strings such that $s_{i+1} = ICR(s_i)$ for all i. Then the following equality holds*

$$P(H(s_0)) - P(H(s_k)) \equiv \sum_{i=0}^{k-1} e_{s_i[0]} \mod K.$$

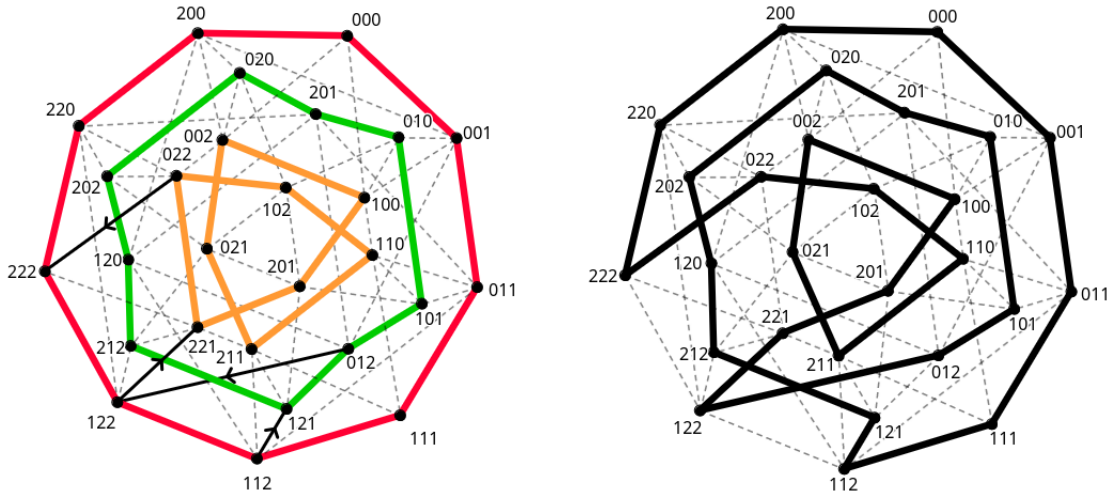*Proof.* Apply Lemma 1 and use the telescopic sum. $\qquad\square$

4.3 Valid Subgraph

Since the ICR-subgraph of B_n is not enough to build a Hamiltonian cycle, we introduce another degree of freedom. We consider $d : \Sigma^n \to \Sigma$ to be a depth assignment, and for each $s \in \Sigma^n$, d_s is its depth.

Definition (Valid Subgraph). *Given a depth assignment $d : \Sigma^n \to \Sigma$, an arc (s,t) in the de Bruijn graph B_n is called* valid *if, for $b = s[0]$ and $c = t[n-1]$,*

- *$b + 1 = c$ and $d_s = d_t$, or*

- *$b + 1 = d_t$ and $c = d_s$*

The set of valid arcs in B_n form the valid subgraph.

In Figure 1 there is an example of a valid subgraph.

(a) Valid subgraph. The dashed lines are the non-valid edges and the lines with arrows are the valid edges that don't belong to any *ICR* cycle. Solid colored lines are the *ICR* cycles.

(b) Hamiltonian cycle subgraph of the valid subgraph. Solid lines form the Hamiltonian cycle.

Figure 1: An example of a valid subgraph for $|\Sigma| = 3$ and $n = 3$ where the *ICR* cycle of 000 is assigned depth 1 and the other two *ICR* cycles are assigned depth 2.

Lemma 3. *Given a depth assignment $d : \Sigma^n \to \Sigma$, and $s,t \in \Sigma^n$ such that (s,t) is a valid arc in the de Bruijn graph then:*

$$e_{s[0]} \equiv P(H(s) + e_{d_s}) - P(H(t) + e_{d_t}) \mod K.$$

Proof. Let $b = s[0]$ and $c = t[n-1]$. Observe that for any valid arc:

$$e_b - e_c + e_{d_s} - e_{d_t} = e_b - e_{b+1}.$$

Now,

$$
\begin{aligned}
P(H(s) + e_{d_s}) - P(H(t) + e_{d_t}) &\equiv P\left(H(s) - H(t) + e_{d_s} - e_{d_t}\right) & \mod K \\
&\equiv P(e_b - e_c + e_{d_s} - e_{d_t}) & \mod K \\
&\equiv P(e_b - e_{b+1}) & \mod K \\
&\equiv e_b \equiv e_{s[0]} & \mod K \;\square
\end{aligned}
$$

If the depth assignment is constant in the cycles produced by the ICR rule, then the ICR-subgraph is a subgraph of the valid subgraph. If we have any path in the valid subgraph, then a telescoping equality holds.

Lemma 4. *Let $d : \Sigma^n \to \Sigma$ be a depth assignment and let $s_0, \dots, s_k$ be a path in the valid subgraph, then*

$$
\sum_{i=0}^{k-1} e_{s_i[0]} \equiv P\left(H(s_0) + e_{d_{s_0}}\right) - P\left(H(s_k) + e_{d_{s_k}}\right) \mod K.
$$

The histogram of the sequence of first symbols in a path in the valid subgraph on B_n depends only on the initial and final string. Observe that in the formula for Lemma 4, the histogram ignores the first symbol of the last string (the summation only goes up to $k - 1$). This is not a problem for our application: Suppose that, for every pair of strings $s, t \in \Sigma^n$ we manage to prove a bound

$$
D\left(P\left(H(s) + e_{d_s}\right) - P\left(H(t) + e_{d_t}\right)\right) \leq F, \qquad F \in \mathbb{N}
$$

Then, every de Bruijn sequence that arises from a Hamiltonian cycle $s_0, \dots, s_k$ in the valid subgraph, has discrepancy-bound F. Indeed, if we consider the substring $s_i[0], s_{i+1}[0] \dots, s_j[0]$, then we can bound

$$
D(H(s_i[0], s_{i+1}[0] \dots, s_j[0]))
$$

by applying Lemma 4 on the path $s_i, s_{i+1}, \dots, s_j, s_{j+1}$ with an extra past-the-end element.

4.4 Difference Bounds for Valid Subgraph Paths

We start by ignoring the depths.

Lemma 5 (Difference of partial sums of histograms). *If $s, t \in \Sigma^n$ then*

$$
D(P(H(s) - H(t))) \leq n.
$$

Proof. The histogram $H(s)$ can be written as a sum over all symbols of s,

$$
H(s) = \sum_{i=0}^{n-1} e_{s[i]}.
$$

Similarly,

$$
H(t) = \sum_{i=0}^{n-1} e_{t[i]}.
$$

6

Therefore,

$$P(H(s) - H(t)) = \sum_{i=0}^{n-1} P(e_{s[i]} - e_{t[i]}).$$

Notice that $P(e_{s[i]} - e_{t[i]})$ has a difference of at most 1 because its values are either all 0s and 1s, or all 0s and -1s. Therefore, due to subadditivity we have the desired bound. $\square$

Lemma 6 (Difference of partial sums of histograms with shared symbol). *If $s, t \in \Sigma^n$ and there exists a symbol that appears in both strings, then*

$$D(P(H(s) - H(t))) \leq n - 1.$$

Proof. In the proof of Lemma 5 we obtained that when $s, t \in \Sigma^n$,

$$P(H(s) - H(t)) = \sum_{i=0}^{n-1} P(e_{s[i]} - e_{t[i]}).$$

Since the histograms do not depend on the order of the symbols, we can assume without loss of generality that $s[0] = t[0]$, and therefore the first summand of the sum vanishes, so the remaining $n - 1$ can contribute at most 1 difference each. $\square$

Now we consider difference of partial sums but including depths.

Lemma 7. *If $s, t \in \Sigma^n$ are strings and $d : \Sigma^n \to \Sigma$ is a depth assignment, then*

$$D\left(P\left(H(s) + e_{d_s}\right) - P\left(H(t) + e_{d_t}\right)\right) \leq n + 1.$$

Proof. The formula follows from Lemma 5 using subadditivity of D, linearity of P and the fact that $D(P(e_{d_s} - e_{d_t})) \leq 1$. $\square$

Lemma 8. *If $s, t \in \Sigma^n$ are strings and $d : \Sigma^n \to \Sigma$ is a depth assignment, such that*

- *$d_s = d_t$, or*

- *s and t share at least one symbol*

Then

$$D\left(P\left(H(s) + e_{d_s}\right) - P\left(H(t) + e_{d_t}\right)\right) \leq n.$$

Proof. In the case where $d_s = d_t$, the term $P(e_{d_s}) - P(e_{d_t})$ vanishes and contributes 0 to the total difference, so the difference-bound of n for the non-depth terms applies.

In the case where s and t share a symbol, the $n - 1$ bound on the non-depth terms applies, with at most an extra unit of difference from the $P(e_{d_s}) - P(e_{d_t})$ term. $\square$

The results above lead us to the following two theorems.

Theorem 2. *If there exists a depth assignment $d : \Sigma^n \to \Sigma$ such that the valid subgraph of B_n has a Hamiltonian cycle, then such a cycle corresponds to a de Bruijn sequence of order n with discrepancy at most $n + 1$.*

Theorem 3. *If there exists a depth assignment $d : \Sigma^n \to \Sigma$ such that*

- *every pair of strings with different depth share a symbol, and*

- *the valid subgraph of B_n has a Hamiltonian cycle,*

then that cycle corresponds to a de Bruijn sequence of order n with discrepancy at most n.

For $|\Sigma| = 2$, the first condition in Theorem 3 is satisfied if and only if the string with all zeros and the string with all ones are assigned the same depth.

5 Our Hamiltonian Cycle

Since $ICR_1 : \Sigma^n \to \Sigma^n$ is a bijective function, it partitions the space Σ^n into cycles. Let $\mathbf{N}$ be the set of cycles given by the ICR_1 rule. We use this set to build a de Bruijn sequence by joining these cycles to obtain a Hamiltonian cycle.

Definition (Orbit of a node). *We define orbit $: \Sigma^n \to \mathcal{P}(\Sigma^n)$ as*

$$orbit(s) = \{s, ICR(s), ICR^2(s), \dots\}.$$

Alternatively, it can be defined as the equivalence class of the relation defined by:

$$R(s,t) \iff t = ICR^k(s) \text{ for some } k \in \mathbb{Z}.$$

Definition (Cycle Graph). *We define a directed graph $\mathbf{G}$ with node-set $\mathbf{N}$ and an arc $(\mathcal{U}, \mathcal{V})$ between cycles $\mathcal{U}, \mathcal{V} \in \mathbf{N}$ if, and only if, there exists an $s \in \mathcal{U}$ such that $ICR_0(s) \in \mathcal{V}$.*

Let us assume that there exists a directed tree $\mathbf{T}$ rooted at $\mathcal{R} \in \mathbf{N}$ that spans $\mathbf{G}$ (the tree is directed from the root to the leaves). The next section proves the existence of $\mathbf{T}$. The tree $\mathbf{T}$ defines a parent relationship $parent : \mathbf{N} \setminus \{\mathcal{R}\} \to \mathbf{N}$.

Definition (Representative). *Let $\mathcal{U} \in \mathbf{N} \setminus \{\mathcal{R}\}$. The representative s of $\mathcal{U}$ is any (fixed) node $s \in \mathcal{U}$ such that $ICR_0^{-1}(s) \in parent(\mathcal{U})$ and $s[n-1] \equiv d \mod |\Sigma|$, where d is the depth of $\mathcal{U}$ in the tree. We write $rep(\mathcal{U}) = s$, and also $rep(u) = s$ for any $u \in \mathcal{U}$. The set of representatives of all cycles is denoted with $\mathbf{Reps}$.*

Observe that for all $s \in \mathbf{Reps}$, $ICR_0^{-1}(s) \in parent(orbit(s))$. Let us define the following rule:

$$R(s) = \begin{cases} ICR_0(s) & \text{if } ICR_0(s) \in \mathbf{Reps} \\ ICR_2(s) & \text{if } ICR_1(s) \in \mathbf{Reps} \\ ICR_1(s) & \text{otherwise.} \end{cases}$$

Lemma 9. *R is a function.*

Proof. We have to show that there is no $s \in \Sigma^n$ such that $ICR_0(s), ICR_1(s) \in \mathbf{Reps}$. We only need to prove the case $|\Sigma| > 2$, because for $|\Sigma| = 2$, ICR_0 and ICR_2 coincide.

Suppose $ICR_0(s), ICR_1(s) \in \mathbf{Reps}$. Then, by definition, $orbit(s) = orbit(ICR_1(s))$. And, since $ICR_0(s) \in \mathbf{Reps}$, necessarily $orbit(s)$ is the parent of $orbit(ICR_0(s))$. This implies that the depth of $orbit(ICR_0(s))$ is one more than the depth of $orbit(ICR_1(s))$. On the other hand, the last symbol of $ICR_1(s)$ is one more than the last symbol of $ICR_0(s)$, and by definition they match the depths of their respective orbits modulo $|\Sigma|$. Since $|\Sigma| > 2$, this is a contradiction. $\square$

Let us prove the following lemmas about this rule.

Lemma 10 (Bijectivity)**.** *The function R is bijective.*

Proof. Suppose $R(s) = R(t)$, then necessarily the maximal proper suffixes of s and t coincide. It is easy to see that if the rule applied to s and t is the same, then $s = t$, because all ICR_k are injective. Then, we have three cases without symmetries:

- $R(s) = ICR_0(s) = ICR_1(t) = R(t)$: Since $R(s) = ICR_0(s)$, then $R(t) = R(s) \in \mathbf{Reps}$. But this is impossible because $R(t) = ICR_1(t)$.

- $R(s) = ICR_0(s) = ICR_2(t) = R(t)$: we can see that $ICR_1(ICR_0^{-1}(ICR_1(t))) = ICR_2(t)$ for all t. Then, $ICR_0^{-1}(ICR_1(t)) = ICR_1^{-1}(ICR_0(s))$, let's call this string u. Now, we know that $ICR_0(s), ICR_1(t) \in \mathbf{Reps}$ so $ICR_0(u), ICR_1(u) \in \mathbf{Reps}$. If $|\Sigma| > 2$ this is impossible, as shown in Lemma 9.

- $R(s) = ICR_1(s) = ICR_2(t) = R(t)$: this means that $ICR_1(t) \in \mathbf{Reps}$, but this is impossible because $ICR_0(s) = ICR_1(t)$ and then $ICR_0(s) \in \mathbf{Reps}$.

$\square$

Lemma 11 (Transitivity)**.** *The bijection R is transitive. That is, for any $s, t \in \Sigma^n$ there exists an integer k such that $R^k(s) = t$.*

Proof. Since R is a bijection, it partitions Σ^n into disjoint R-cycles. Suppose there is an ICR-cycle $\mathcal{U} \in \mathbf{N}$ such that $\mathcal{U}$ is not completely contained in any R-cycle. That means $\mathcal{U}$ contains two nodes belonging to different R-cycles. In particular, there are two strings $s, t \in \mathcal{U}$ such that s and $ICR_1(s)$ belong to different R-cycles and, also, t and $ICR_1(t)$ belong to different R-cycles. Thus, without loss of generality, we can assume that $ICR_1(s) \notin \mathbf{Reps}$ (since $\mathcal{U} \cap \mathbf{Reps}$ has size at most one), and also that $\mathcal{U}$ is the deepest node in the tree that intersects two different R-cycles.

Clearly, $R(s) \neq ICR_1(s)$, otherwise s and $ICR_1(s)$ would belong to the same R-cycle. Since $ICR_1(s) \notin \mathbf{Reps}$, the remaining possibility (from the definition of R) is that $ICR_0(s) \in \mathbf{Reps}$. From the definition of $\mathbf{Reps}$, we get that the orbit of $ICR_0(s)$ is a child of the orbit of s. From the definition of R, we get that

$$R(ICR_1^{-1}(ICR_0(s))) = ICR_2(ICR_1^{-1}(ICR_0(s))) = ICR_1(s),$$

and therefore $ICR_1^{-1}(ICR_0(s))$ and $ICR_1(s)$ belong to the same R-cycle. But $ICR_0(s)$ belongs to the same R-cycle as s, so the orbit of $ICR_0(s)$ *also* contains two different R-cycles, which contradicts the assumption that $\mathcal{U}$ is the deepest.

Now assume that there are two adjacent orbits $\mathcal{U}, \mathcal{V} \in \mathbf{N}$ in the tree that are contained in different R-cycles. Without loss of generality, let us assume that $\mathcal{V}$ is the child of $\mathcal{U}$ and let $s = rep(\mathcal{V})$. Since $s \in \mathbf{Reps}$, we know that $R(ICR_0^{-1}(s)) = s$ and that $ICR_0^{-1}(s)$ is in the parent orbit of s (namely, $\mathcal{U}$). This is a contradiction, because we assumed that $\mathcal{U}$ and $\mathcal{V}$ do not belong to the same R-cycle. $\square$

As R is a bijective and transitive function, and it also satisfies the property that the arc $(s, R(s))$ is in the de Bruijn graph, we can construct a Hamiltonian cycle by taking an arbitrary node as the start, let us call it s, and then repeatedly applying the function R until we arrive to s again.

5.1 Hamiltonian cycle in the Valid Subgraph

Now that we have proved that R generates a Hamiltonian cycle in the de Bruijn graph, let us prove that all of its arcs are in the valid subgraph. To construct the valid graph we need to define a depth assignment.

Definition (Depth Assignment). *For a fixed tree of ICR-cycles, we define*

$$d : \Sigma^n \to \Sigma \qquad d(s) = p + 1,$$

where p is the depth of the orbit of s in the tree, modulo $|\Sigma|$.

The resulting valid subgraph for the case where the parent of every cycle $\mathcal{U} \in \mathbf{N} \setminus \{\mathcal{R}\}$ is $\mathcal{R}$ is given in Figure 1, and the unique Hamiltonian path in the subgraph is shown.

Lemma 12. *The arc $(s, R(s))$ is valid.*

Proof. Let b be the first symbol of s and c be the last symbol of $R(s)$. We need to consider the three cases in the definition of $R(s)$:

- Case $ICR_0(s) \in \mathbf{Reps}$: In this case $R(s) = ICR_0(s)$ and therefore $c = b$ and $c = d_{R(s)} - 1$. Also, by definition of $\mathbf{Reps}$, the orbit of $R(s)$ is a child of the orbit of s, so we have $d_s + 1 = d_{R(s)}$. Putting these together we get that $b + 1 = d_{R(s)}$ and $c = d_s$, so the arc is valid.

- Case $ICR_1(s) \in \mathbf{Reps}$: In this case $R(s) = ICR_2(s)$ and therefore $c = b + 2$, and given that $ICR_1(s) \in \mathbf{Reps}$, $b + 1 = d_s - 1$. Since $ICR_2(s) = ICR_1(ICR_0^{-1}(ICR_1(s)))$, the orbit of $ICR_2(s)$ is the parent of the orbit of s, and thus $d_{R(s)} = d_s - 1$. Putting these together we get $b + 1 = d_{R(s)}$ and $c = d_s$, so the arc is valid.

- Remaining case: We have $R(s) = ICR_1(s)$, so $c = b + 1$ and $d_s = d_{R(s)}$, so the arc is once again valid. □

5.2 Discrepancy Bound for the General Case

Due to Theorem 2, we have that the de Bruijn sequence associated with R has discrepancy at most $n + 1$.

5.3 Discrepancy Bound for the Binary Case

For the binary case, we use Theorem 3: since the strings 0^n and 1^n belong to the same orbit, they are assigned the same depth value and therefore the hypotheses of the theorem hold and we have that the de Bruijn sequence associated with R has discrepancy at most n.

5.4 Explicit Spanning Tree

To complete the proof, we have to show a directed spanning tree $\mathbf{T}$ of $\mathbf{G}$ and choose the representatives for each orbit. To do this we define the following:

Definition (Difference Array). *Given $s \in \Sigma^n$, we define its difference array $\Delta(s) \in \Sigma^n$,*

$$\Delta(s)[i] = \begin{cases} s[i-1] - s[i] & \text{if } 0 < i < n \\ s[n-1] - s[0] - 1 & \text{if } i = 0. \end{cases}$$

The extra -1 in the $i = 0$ case ensures the following property:

Lemma 13. *Let $s \in \Sigma^n$, then $\Delta(ICR_1(s)) = ICR_0(\Delta(s))$.*

Proof. Consider the bidirectional infinite sequence

$$c_i = ICR^i(s)[0], \qquad i \in \mathbb{Z}.$$

By the definition of ICR, we have that $c_{i+n} = c_i + 1$. Now consider the following associated (infinite) sequence

$$d_i = c_{i-1} - c_i, \qquad i \in \mathbb{Z}.$$

Since $c_{i+n} = c_i + 1$, we have that d is cyclic modulo n. Also, due to the definition of Δ we have that

$$\Delta(s) = d_0 d_1 \dots d_{n-1} \quad \text{and} \quad \Delta(ICR(s)) = d_1 d_2 \dots d_n.$$

Since $d_n = d_0$, the desired result follows. $\square$

Lemma 13 implies that for each $\mathcal{U} \in \mathbf{N}$, its elements have the same difference array modulo rotations. By convention, we say that $\Delta(\mathcal{U})$ is the lexicographically minimal rotation of $\Delta(s)$ for any $s \in \mathcal{U}$. It is easy to see that the converse is also true: for any $s, t \in \Sigma^n$, if $\Delta(s)$ and $\Delta(t)$ are equal modulo rotations, then $orbit(s) = orbit(t)$. Indeed, consider the string $u = ICR_1^k(s)$. We can vary k such that $\Delta(u) = \Delta(t)$, and then the strings u and t only differ by an added constant, so they must belong to the same orbit.

Also, from the definition of Δ, we have the following identity

$$\sum_{i=0}^{n-1} \Delta(s)[i] = -1,$$

which follows from the fact that every symbol of s cancels out in the sum, except the constant factor -1 in the first position of $\Delta(s)$. Again, the converse is also true. If we have any string $t \in \Sigma^n$ such that

$$\sum_{i=0}^{n-1} t[i] = -1,$$

then there exists a $s \in \Sigma^n$ that satisfies $\Delta(s) = t$, which is unique modulo added constants. To see this, we can choose $s[0]$ arbitrarily, then once $s[i]$ has been determined, the value of $s[i+1]$ follows directly from the equation

$$\Delta(s)[i+1] = t[i+1].$$

The only equation that remains to be satisfied is $\Delta(s)[0] = t[0]$. However, $\Delta(s)$ and t coincide in $n-1$ places and they have the same sum, -1, so they must be identical.

Equipped with the concept of difference array, we can define the tree $\mathbf{T}$ as follows.

Definition (Explicit Tree $\mathbf{T}$). *The root of the tree is the orbit of 0^n and we call it $\mathcal{R}$. For each orbit $\mathcal{U} \in \mathbf{N} \setminus \mathcal{R}$, we define its parent $\mathcal{V} \in \mathbf{N}$ as follows:*

let i be the first non-negative integer such that $\Delta(\mathcal{U})[i] \neq 0$. Consider the array A obtained from decrementing the i-th symbol of $\Delta(\mathcal{U})$ and incrementing the $(i+1)$-th symbol. $\mathcal{V}$ is the only orbit such that $\Delta(\mathcal{V})$ equals A modulo rotations.
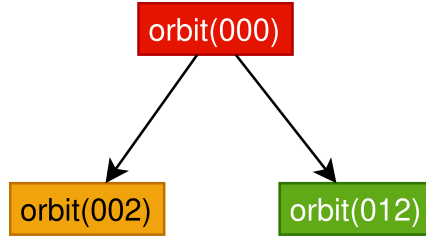
Figure 2: Explicit Tree for the case $n = 3, |\Sigma| = 3$.

Figure 2 gives an example of an Explicit Tree $\mathbf{T}$.

Lemma 14. *The Explicit Tree $\mathbf{T}$ is a spanning tree of $\mathbf{G}$.*

Proof. The existence of i follows from the fact that the sum of elements of the difference array is -1, so it cannot consist solely of zeros. In fact, $i \leq n - 2$; otherwise in $\Delta(\mathcal{U})$ there would be $n - 1$ zeros and, necessarily, the remaining symbol would be -1, which is the difference array of $\mathcal{R}$.

To prove that the parent relationship has no cycles, note that A is lexicographically smaller than $\Delta(\mathcal{U})$, so we have that

$$\Delta(\mathcal{V}) \leq A < \Delta(\mathcal{U})$$

and this ensures that there are no cycles.

Let us see that for every node $\mathcal{U} \in \mathbf{N} \setminus \mathcal{R}$, the pair $(\mathcal{V}, \mathcal{U})$ is an arc of $\mathbf{G}$ when $\mathcal{V}$ is the parent of $\mathcal{U}$ as defined in the definition of the Explicit Tree. It suffices to find an $s \in \mathcal{U}$ such that $ICR_0^{-1}(s) \in \mathcal{V}$, which can also serve as a representative of $\mathcal{U}$. Let i be the first non-negative integer such that $\Delta(\mathcal{U})[i] \neq 0$, and consider the array $k = ICR_0^{i+1}(\Delta(\mathcal{U}))$. That is, the array $\Delta(\mathcal{U})$ rotated such that the first non-zero element moves to the last position. Let $s \in \Sigma^n$ be a string with $\Delta(s) = k$. We can choose the added constant of s as necessary to satisfy the last-symbol constraint in the definition of representative.

We want to show that $ICR_0^{-1}(s) \in \mathcal{V}$. To do this, let us prove that $\Delta(ICR_1(ICR_0^{-1}(s)))$ is a rotation of A, this will imply that $ICR_1(ICR_0^{-1}(s)) \in \mathcal{V}$ and since it is in the same orbit as $ICR_0^{-1}(s)$, it would mean that the latter is also in $\mathcal{V}$.

Observe that $ICR_1(ICR_0^{-1}(s))$ is the same as s but with the last symbol increased by one. The effect that increasing the last symbol of s has on $\Delta(s)$ is that of decreasing the last symbol and increasing the first symbol. Since $\Delta(s) = ICR_0^{i+1}(\Delta(\mathcal{U}))$, we have that $\Delta(ICR_1(ICR_0^{-1}(s)))$ is the same as $ICR_0^{i+1}(\Delta(\mathcal{U}))$ but the last symbol decreased and the first one increased, which is precisely the same as $ICR_0^{i+1}(A)$. This concludes the proof. $\square$

6 Our Algorithm

The algorithm constructs a de Bruijn sequence of order n in the alphabet Σ.

Recall that we defined the transition rule R as follows,

$$R(s) = \begin{cases} ICR_0(s) & \text{if } ICR_0(s) \in \mathbf{Reps} \\ ICR_2(s) & \text{if } ICR_1(s) \in \mathbf{Reps} \\ ICR_1(s) & \text{otherwise.} \end{cases}$$

Algorithm 1: CorrectDifferenceArray
___
Data: $s \in \Sigma^n$

Result:

True if s has the correct difference array to be a representative, False otherwise

$d \leftarrow \Delta(s)$

if $d[n-1] = 0$ **then**
  | **return** *False*

end

$i \leftarrow n - 1$

while $i > 0$ *and* $d[i-1] = 0$ **do**
  | $i \leftarrow i - 1$

end

if $i = 0$ **then**
  | **return** *False*

end

$d_2 \leftarrow ICR_0^i(d)$

return d_2 *is its lexicographically minimal rotation*

Algorithm 2: Transition
___
Data: $s \in \Sigma^n$, depth $\in \Sigma$

Result: A Pair (s', depth') that corresponds to the next node in the path

if *CorrectDifferenceArray*$(ICR_0(s))$ *and* $ICR_0(s)[n-1] = depth + 1$ **then**
  | **return** $ICR_0(s), depth + 1$

end

if *CorrectDifferenceArray*$(ICR_1(s))$ *and* $ICR_1(s)[n-1] = depth$ **then**
  | **return** $ICR_2(s), depth - 1$

end

return $ICR_1(s), depth$

For the algorithm to run in $O(n)$ space, we cannot maintain the tree in memory. Instead, we use the definition of the Explicit Tree **T** and we give an efficient procedure to compute membership to the set **Reps**. This has two parts:

- The difference array of the string must be of the correct form. That is, it should be the lexicographically minimal rotation, shifted so that the first non-zero element is at the end. To be a representative, s also must not be in the root orbit, which we can also check with the difference array.

- The last symbol of the string must be equal to the depth of the orbit modulo $|\Sigma|$.

The first part is implemented in Algorithm 1. The second part is implemented together with the transition function in Algorithm 2. Note that both functions have linear complexity in both time and memory. To determine whether d_2 is the lexicographically minimal rotation we can use Booth's algorithm [1], for example.

6.1 Example Sequences

In Table 1 and Figure 3 (left) we display the output of our algorithm.

$|\Sigma|$	n	Resulting Sequence
2	2	1100
2	3	11101000
2	4	1111001011010000
2	5	11111000101011001001101110100000
2	6	1111110001001100111011000010110101001010111001000110111101000000
3	2	112102200
3	3	111212020101221002110222000
4	2	1121320310223300
4	3	1112123230201312023130301012213320021132203310321003110222333000

Table 1: Example sequences produced by our algorithm.

7 Analysis

7.1 Our Algorithm Behavior for an Arbitrary Base

For base $|\Sigma| = 2$, we know our algorithm produces de Bruijn sequences of discrepancy exactly n, because that is both the upper bound and the trivial lower bound. For the case $n = 1$, we also know our algorithm produces a sequence of discrepancy exactly n, because all de Bruijn sequences for $n = 1$ have discrepancy n. Therefore, in both of these cases our algorithm is optimal.

However, for the case $|\Sigma| > 2$ and $n > 1$, it is still unknown whether the minimum attainable discrepancy is n or $n + 1$. Our algorithm produces a de Bruijn sequence with discrepancy exactly $n+1$ (there is a proof, not included in this paper, that shows our method does not achieve discrepancy equal to n).

7.2 Conjecture on the Minimum Attainable Discrepancy

$|\Sigma|$ \ n	1	2	3	4	5	6	7
2	n	n	n	n	n	n	n
3	n	n+1	n	n			
4	n	n+1	n				
5	n	n					
6	n	n					
7	n	n					
8	n	n					

Table 2: Each cell has the minimum discrepancy attainable for a de Bruijn sequence with the corresponding parameters. The cases that could not be computed are blank.

To compare the behaviour of the algorithm with the actual minimum discrepancy attainable, we made an exhaustive search for the smallest discrepancy that can be obtained with certain parameters of n and $|\Sigma|$, to decide whether our algorithm was optimal or not. The results we obtained can be seen in Table 2. The experimentation is heavily limited due to the doubly exponential nature of the search.

14

n	Our Algorithm	Huang	Random	Weight-range
10	10	12	50	131
11	11	13	71	257
12	12	15	101	468
13	13	16	143	930
14	14	18	203	1723
15	15	19	288	3439
16	16	21	407	6443
17	17	22	575	12878
18	18	24	815	24319
19	19	25	1157	48629
20	20	27	1634	92388
21	21	28	2311	184766
22	22	30	3264	352727
23	23	31	4565	705443
24	24	33	6252	1352090
25	25	35	9192	2704168
26	26	36	13074	5200313
27	27	38	17933	10400613
28	28	40	22672	20058314
29	29	41	34591	40116614
30	30	43	57357	77558775

Table 3: Discrepancy attained by various algorithms for binary de Bruijn sequences. The data for **Huang**, **Random** and **Weight-range** is taken from [3].

Based on the experimental results and given that the de Bruijn graph gets more interconnected with increasing n and $|\Sigma|$, one can expect that achieving a low-discrepancy de Bruijn sequence becomes easier as these parameters increase. We put forward the following conjecture.

Conjecture 1. *The minimal discrepancy of a de Bruijn sequence of order n and alphabet Σ is n, with the exception of the cases $n = 2$ with $|\Sigma| = 3$ and $n = 2$ with $|\Sigma| = 4$.*

7.3 Comparison with Other de Bruijn Sequences

In Table 3 we compare the discrepancy obtained with our algorithm in a binary alphabet with that obtained in previously-described algorithms in the literature:

- Huang's algorithm [4] which attained the previously smallest known discrepancy.

- A uniformly random de Bruijn sequence.

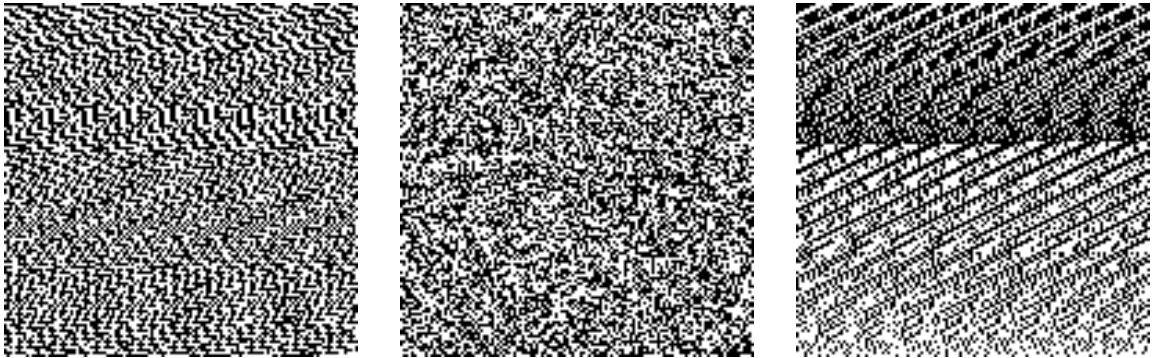- The Weight-range construction [3], which is proven to have the asymptotically maximum discrepancy possible.

Figure 3: Graphical representation for the binary sequences of order $n = 14$ produced by our algorithm (left), a de Bruijn sequence chosen uniformly at random (center) and the de Bruijn sequence with asymptotically maximum discrepancy in [3] (right). The symbols of the sequence are displayed in row-major order. Zero is white, one is black.

8 Code

We include a full C++ implementation of the proposed algorithm.

```cpp
#include <vector>
#include <algorithm>
#include <iostream>
using namespace std;

// n is the order of the de Bruijn string
int n;

// base is the size of the alphabet
int base;

// Checks if s is its lexicographically minimal
// rotation in O(n). Modified Duval.
// Implementation from:
//    https://stackoverflow.com/a/73966629
bool min_lex(const vector<int> &s) {
    for(int i=0, j=1; j < 2*n; j++) {
        int a = s[i%n], b = s[j%n];
        if(b < a) return false;
        else if(a < b) i = 0;
        else i++;
    }
    return true;
}

// The difference array of s, defined as
//    res[i] = s[i-1] - s[i]
// with the convention that s[-1] = s[n-1]-1
// this ensures ICR induces cyclic shift of
// difference array
vector<int> diff_array(const vector<int> &s) {
    vector<int> res(n);
    for(int i = 0; i < n; i++) {
        res[i] = s[(i-1+n)%n] - s[i] - (i==0);
        res[i] = (res[i] + base) % base;
    }
    return res;
}

// Incremented Cyclic Register rule
vector<int> icr(const vector<int> &s, int k) {
    vector<int> t = s;
    rotate(t.begin(), t.begin() + 1, t.end());
    t.back() = (t.back() + int(k))%base;
    return t;
}
```

```cpp
// Checks if s is in Reps.
// This is true iff the difference array of s
// is positioned such that the lexicographically
// minimal rotation ends its first run of zeros
// at the last element.
// This ensures that applying ICR_0^-1 to s
// decreases the first element after the run.
bool reps(const vector<int> &s, int depth) {
    auto da = diff_array(s);
    if(da.back() == 0) return false;

    int i = n-1;
    while(i != 0 && da[i-1] == 0) i--;
    if(i == 0) return false;

    rotate(da.begin(), da.begin()+i, da.end());
    if(!min_lex(da)) return false;

    return (depth-s.back()) % base == 0;
}

// Generate the full de Bruijn sequence,
// with the transition function defined by
//    P(s) = ICR_0(s) if ICR_0(s) is in Reps
//           ICR_2(s) if ICR_1(s) is in Reps
//           ICR_1(s) otherwise
vector<int> generate() {
    vector<int> result;
    vector<int> s(n, 0);
    int depth = 0;
    do {
        auto a = icr(s, 0);
        auto b = icr(s, 1);
        auto c = icr(s, 2);
        if(reps(a, depth+1)) s = a, depth++;
        else if(reps(b, depth)) s = c, depth--;
        else s = b;
        result.push_back(s.back());
    } while(s != vector<int>(n, int(0)));
    return result;
}

int main() {
    cout << "enter n and base: " << flush;
    cin >> n >> base;
    auto result = generate();

    for(int i : result) cout << i;
    cout << endl;
}
```

References

[1] Kellogg S. Booth. Lexicographically least circular substrings. *Information Processing Letters*, 10(4):240–242, 1980.

[2] Nicolaas G. de Bruijn. A combinatorial problem. *Nederl. Akad. Wetensch., Proc.*, 49:758–764 = Indagationes Math. 8, 461–467 (1946), 1946.

[3] Daniel Gabric and Joe Sawada. Investigating the discrepancy property of de bruijn sequences. *Discrete Mathematics*, 345(4):112780, 2022.

[4] Yue Jiang Huang. A new algorithm for the generation of binary de Bruijn sequences. *Journal of Algorithms*, 11(1):44–51, 1990.

Nicolás Álvarez
ICC CONICET Argentina - `nico.alvarez@gmail.com`

Verónica Becher
Departamento de Computación, Facultad de Ciencias Exactas y Naturales & ICC
Universidad de Buenos Aires & CONICET Argentina- `vbecher@dc.uba.ar`

Martín Mereb
Departamento de Matemática, Facultad de Ciencias Exactas y Naturales & IMAS
Universidad de Buenos Aires & CONICET Argentina- `mmereb@gmail.com`

Ivo Pajor
Departamento de Computación, Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires & Argentina- `pajorivo@gmail.com`

Carlos Miguel Soto
Departamento de Computación, Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires & Argentina- `miguelsotocarlos@gmail.com`