De Bruijn sequence

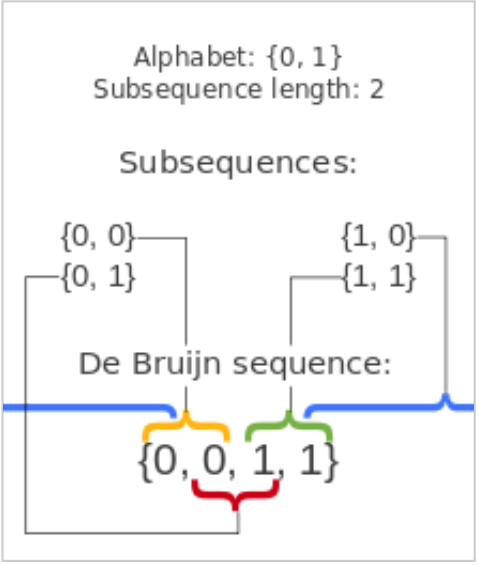
From Wikipedia, the free encyclopedia

In combinatorial mathematics, a k-ary **De Bruijn sequence** $B(k, n)$ of order n, named after the Dutch mathematician Nicolaas Govert de Bruijn, is a cyclic sequence of a given alphabet A with size k for which every possible subsequence of length n in A appears as a sequence of consecutive characters exactly once.

Each $B(k, n)$ has length k^n.

There are $\frac{(k!)^{k^{n-1}}}{k^n}$ distinct De Bruijn sequences $B(k, n)$.

According to de Bruijn,[1] the existence of De Bruijn sequences for each order together with the above properties were first proved, for the case of alphabets with two elements, by Camille Flye Sainte-Marie in 1894,[2] whereas the generalization to larger alphabets is originally due to Tanja van Aardenne-Ehrenfest and himself.



De Bruijn sequence for $k = 2$ and $n = 2$

Contents

History

The earliest known example of a De Bruijn sequence comes from Sanskrit prosody where, since the work of Pingala, each possible three-syllable pattern of long and short syllables is given a name, such as 'y' for short–long–long and 'm' for long–long–long. To remember these names, the mnemoic *yamātārājabhānasalagam* is used, in which each three-syllable pattern occurs starting at its name: 'yamātā' has a short–long–long pattern, 'mātārā' has a long–long–long pattern, and so on, until 'salagam' which has a short–short–long pattern because of the final consonant. This mnemonic, equivalent to a De

Bruijn sequence on binary 3-tuples, is of unknown antiquity, but is at least as old as C. P. Brown's 1869 book on Sanskrit prosody that mentions it and considers it "an ancient line, written by Pāṇini".[3][4][5][6][7]

In 1894, A. de Rivière raised the question in an issue of the French problem journal *L'Intermédiaire des Mathématiciens*, of the existence of a circular arrangement of length 2^n which contains all 2^n binary sequences of length n. The problem was solved, along with the count $2^{2^{n-1}-n}$, by C. Flye Sainte-Marie in the same year.[1] This was largely forgotten, and Martin (1934) proved the existence of such cycles for general alphabet size in place of 2, with an algorithm for constructing them. Finally, when in 1944 Kees Posthumus conjectured the count $2^{2^{n-1}-n}$ for binary sequences, de Bruijn proved the conjecture in 1946, through which the problem became well-known.[1]

Karl Popper independently describes these objects in his *The Logic of Scientific Discovery* (1934), calling them "shortest random-like sequences".[8]

Examples

- Taking $A = \{0, 1\}$, there are two distinct $B(2, 3)$: 00010111 and 11101000, one being the reverse or negation of the other.
- Two of the 2048 possible $B(2, 5)$ in the same alphabet are 00000100011001010011101011011111 and 00000101001000111111011101001101011.

Construction

The De Bruijn sequences can be constructed by taking a Hamiltonian path of an n-dimensional De Bruijn graph over k symbols (or equivalently, a Eulerian cycle of a $(n-1)$-dimensional De Bruijn graph).[9]

An alternative construction involves concatenating together, in lexicographic order, all the Lyndon words whose length divides n.[10]

De Bruijn sequences can also be constructed using shift registers[11] or via finite fields.[12]

Example

Goal: to construct a $B(2, 4)$ De Bruijn sequence of length $2^4 = 16$ using Eulerian ($n - 1 = 4 - 1 = 3$) 3-D De Bruijn graph cycle.

Each edge in this 3-dimensional De Bruijn graph corresponds to a sequence of four digits: the three digits that label the vertex that the edge is leaving followed by the one that labels the edge. If one traverses the edge labeled 1 from 000, one arrives at 001, thereby indicating the presence of the subsequence 0001 in the De Bruijn sequence. To traverse each edge exactly once is to use each of the 16 four-digit sequences exactly once.

For example, suppose we follow the following Eulerian path through these nodes:

000, 000, 001, 011, 111, 111, 110, 101, 011,

110, 100, 001, 010, 101, 010, 100, 000.

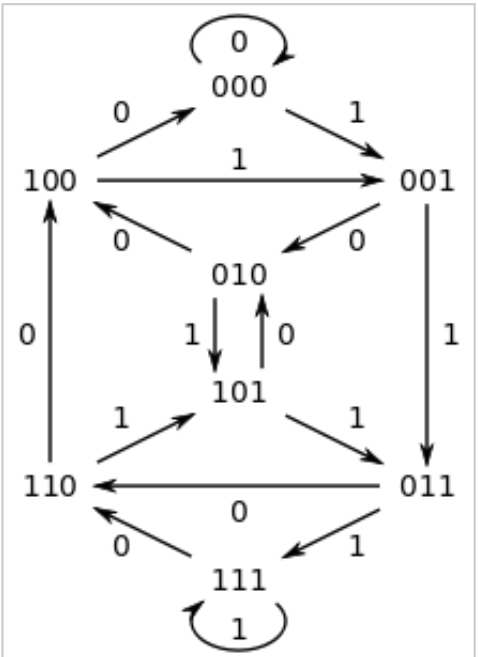
These are the output sequences of length k:

0 0 0 0
_ 0 0 0 1
_ _ 0 0 1 1

This corresponds to the following De Bruijn sequence:

0 0 0 0 1 1 1 1 0 1 1 0 0 1 0 1

The eight vertices appear in the sequence in the following way:



A De Bruijn graph. Every four-digit sequence occurs exactly once if one traverses every edge exactly once and returns to one's starting point (an Eulerian cycle). Every three-digit sequence occurs exactly once if one visits every node exactly once (a Hamiltonian path).

```
    {0  0  0} 0  1  1  1  1  0  1  1  0  0  1  0  1
     0 {0  0  0} 1  1  1  1  0  1  1  0  0  1  0  1
     0  0 {0  0  1} 1  1  1  0  1  1  0  0  1  0  1
     0  0  0 {0  1  1} 1  1  0  1  1  0  0  1  0  1
     0  0  0  0 {1  1  1} 1  0  1  1  0  0  1  0  1
     0  0  0  0  1 {1  1  1} 0  1  1  0  0  1  0  1
     0  0  0  0  1  1 {1  1  0} 1  1  0  0  1  0  1
     0  0  0  0  1  1  1 {1  0  1} 1  0  0  1  0  1
     0  0  0  0  1  1  1  1 {0  1  1} 0  0  1  0  1
     0  0  0  0  1  1  1  1  0 {1  1  0} 0  1  0  1
     0  0  0  0  1  1  1  1  0  1 {1  0  0} 1  0  1
     0  0  0  0  1  1  1  1  0  1  1 {0  0  1} 0  1
     0  0  0  0  1  1  1  1  0  1  1  0 {0  1  0} 1
     0  0  0  0  1  1  1  1  0  1  1  0  0 {1  0  1}
... 0} 0  0  0  1  1  1  1  0  1  1  0  0  1 {0  1 ...
... 0  0} 0  0  1  1  1  1  0  1  1  0  0  1  0 {1 ...
```

...and then we return to the starting point. Each of the eight 3-digit sequences (corresponding to the eight vertices) appears exactly twice, and each of the sixteen 4-digit sequences (corresponding to the 16 edges) appears exactly once.

Algorithm

The following Python code calculates a De Bruijn sequence, given *k* and *n*, based on an algorithm from Frank Ruskey's *Combinatorial Generation*.[13]

```
def de_bruijn(k, n):
    """
    De Bruijn sequence for alphabet k
    and subsequences of length n.
    """
    try:
        # let's see if k can be cast to an integer;
        # if so, make our alphabet a list
        _ = int(k)
        alphabet = list(map(str, range(k)))

    except (ValueError, TypeError):
        alphabet = k
        k = len(k)

    a = [0] * k * n
    sequence = []

    def db(t, p):
        if t > n:
            if n % p == 0:
                sequence.extend(a[1:p + 1])
        else:
            a[t] = a[t - p]
            db(t + 1, p)
            for j in range(a[t - p] + 1, k):
                a[t] = j
                db(t + 1, t)
    db(1, 1)
    return "".join(alphabet[i] for i in sequence)

print(de_bruijn(2, 3))
print(de_bruijn("abcd", 2))
```

which prints

```
00010111
aabacadbbcbdccdd
```

Uses

The sequence can be used to shorten a brute-force attack on a PIN-like code lock that does not have an "enter" key and accepts the last *n* digits entered. For example, a digital door lock with a 4-digit code would have $B(10, 4)$ solutions, with length 10 000. Therefore, only at most 10 000 + 3 = 10 003 (as the solutions are cyclic) presses are needed to open the lock. Trying all codes separately would require $4 \times 10\,000 = 40\,000$ presses.

The symbols of a De Bruijn sequence written around a circular object (such as a wheel of a robot) can be used to identify its angle by examining the *n* consecutive symbols facing a fixed point. Gray codes can be used as similar rotary positional encoding mechanisms.

De Bruijn cycles are of general use in neuroscience and psychology experiments that examine the effect of stimulus order upon neural systems,[14] and can be specially crafted for use with functional magnetic resonance imaging.[15]

A De Bruijn sequence can be used to quickly find the index of the LSB or MSB in a word using bitwise operations.[16][17] An example of returning the index of the least significant bit from a 32 bit unsigned integer is given below using bit manipulation.

```
unsigned int v;
int r;
static const int MultiplyDeBruijnBitPosition[32] =
{
  0, 1, 28, 2, 29, 14, 24, 3, 30, 22, 20, 15, 25, 17, 4, 8,
  31, 27, 13, 23, 21, 19, 16, 7, 26, 12, 18, 6, 11, 5, 10, 9
};
r = MultiplyDeBruijnBitPosition[((uint32_t)((v & -v) * 0x077CB531U)) >> 27];
```

The index of the LSB in *v* is stored in *r* and if *v* has no set bits the operation returns 0. The constant, 0x077CB531U, in the expression is a De Bruijn sequence.

De Bruijn torus

Main article: De Bruijn torus

A De Bruijn torus is a toroidal array with the property that every *k*-ary *m*-by-*n* matrix occurs exactly once. (It is not necessary that the array be expressed toroidally; the array can be mapped into a 2-dimensional array. Because it is toroidal it "wraps around" on all 4 sides.)

Such a pattern can be used for two-dimensional positional encoding in a fashion analogous to that described above for rotary encoding. Position can be determined by examining the *m*-by-*n* matrix directly adjacent to the sensor, and calculating its position on the De Bruijn torus.

De Bruijn decoding

Computing the position of a particular unique tuple or matrix in a De Bruijn sequence or torus is known as the De Bruijn Decoding Problem. Efficient O(n log n) decoding algorithms exists for special, recursively constructed sequences[18] and extend to the two dimensional case.[19] De Bruijn decoding is of interest, e.g., in cases where large sequences or tori are used for positional encoding.

See also

- De Bruijn graph
- De Bruijn torus
- Normal number
- Linear feedback shift register
- *n*-sequence

Notes

1. De Bruijn (1975).
2. Flye Sainte-Marie, C. (1894), "Solution to question nr. 48", *L'intermédiaire des Mathématiciens* **1**: 107–110

3. C. P. Brown, 1869, *Sanskrit Prosody and Numerical Symbols Explained*, p. 28 (https://archive.org/stream/sanskritprosody00browgoog#page/n44/mode/2up)
4. Subhash Kak, 2000, Yamātārājabhānasalagāṃ an interesting combinatoric sūtra (http://202.41.82.144/rawdatauploadupload/insa/INSA_2/200059d2_123.pdf), Indian Journal of History of Science, 35.2 (2000), 123–127.
5. Rachel W. Hall. Math for poets and drummers (http://www.sju.edu/~rhall/mathforpoets.pdf). *Math Horizons* **15** (2008) 10–11.
6. Donald Ervin Knuth (2006). *The Art of Computer Programming, Fascicle 4: Generating All Trees – History of Combinatorial Generation* (http://books.google.com/?id=56LNfE2QGtYC&pg=PA50&dq=Pingala). Addison–Wesley. p. 50. ISBN 978-0-321-33570-8.
7. Stein, Sherman K. (1963), "Yamátárájabhánasalagám", *The Man-made Universe: An Introduction to the Spirit of Mathematics*, pp. 110–118. Reprinted in Wardhaugh, Benjamin, ed. (2012), *A Wealth of Numbers: An Anthology of 500 Years of Popular Mathematics Writing*, Princeton Univ. Press, pp. 139–144.
8. Karl Popper (2002) [1934]. *The logic of scientific discovery* (http://books.google.com/?id=0a5bLBbe_dMC&pg=PA295&cd=1#v=onepage&q=1111000010011010). Routledge. p. 294. ISBN 978-0-415-27843-0.
9. Klein, Andreas (2013), *Stream Ciphers* (http://books.google.com/books?id=GYpEAAAAQBAJ&pg=PA59), Springer, p. 59, ISBN 9781447150794.
10. According to Berstel & Perrin (2007), the sequence generated in this way was first described (with a different generation method) by Martin (1934), and the connection between it and Lyndon words was observed by Fredricksen & Maiorana (1978).
11. Goresky, Mark; Klapper, Andrew (2012), "8.2.5 Shift register generation of de Bruijn sequences", *Algebraic Shift Register Sequences* (http://books.google.com/books?id=sd9AqHeeHh4C&pg=PA174), Cambridge University Press, pp. 174–175, ISBN 9781107014992.
12. Ralston, Anthony (1982), "de Bruijn sequences—a model example of the interaction of discrete mathematics and computer science", *Mathematics Magazine* **55** (3): 131–143, doi:10.2307/2690079 (https://dx.doi.org/10.2307%2F2690079), MR 653429 (https://www.ams.org/mathscinet-getitem?mr=653429). See in particular "the finite field approach", pp. 136–139.
13. "De Bruijn sequences" (http://hg.sagemath.org/sage-main/file/9e29a3d84c48/sage/combinat/debruijn_sequence.pyx). *Sage*. Retrieved 12 November 2011.
14. GK Aguirre, MG Mattar, L Magis-Weinberg. (2011) "de Bruijn cycles for neural decoding" (https://cfn.upenn.edu/aguirre/wiki/public:de_bruijn).. *NeuroImage* 56: 1293–1300
15. "De Bruijn cycle generator" (http://cfn.upenn.edu/aguirre/wiki/public:de_bruijn_software).
16. Anderson, Sean Eron (1997–2009). "Bit Twiddling Hacks" (http://graphics.stanford.edu/~seander/bithacks.html). Stanford University. Retrieved 2009-02-12.
17. Busch, Philip (2009). "Computing Trailing Zeros HOWTO" (http://7ooo.mooo.com/text/ComputingTrailingZerosHOWTO.html). Retrieved 2015-01-29.
18. Tuliani (2001).
19. Hurlbert & Isaak (1993).

References

- van Aardenne-Ehrenfest, T.; de Bruijn, N. G. (1951), "Circuits and trees in oriented linear graphs" (http://alexandria.tue.nl/repository/freearticles/597493.pdf) (PDF), *Simon Stevin* **28**: 203–217, MR 0047311 (https://www.ams.org/mathscinet-getitem?mr=0047311).
- Berstel, Jean; Perrin, Dominique (2007), "The origins of combinatorics on words" (http://www-igm.univ-mlv.fr/~berstel/Articles/2007Origins.pdf) (PDF), *European Journal of Combinatorics* **28** (3): 996–1022, doi:10.1016/j.ejc.2005.07.019 (https://dx.doi.org/10.1016%2Fj.ejc.2005.07.019), MR 2300777 (https://www.ams.org/mathscinet-getitem?mr=2300777).
- de Bruijn, N. G. (1946), "A combinatorial problem" (http://www.dwc.knaw.nl/DL/publications/PU00018235.pdf) (PDF), *Proc. Koninklijke Nederlandse Akademie v. Wetenschappen* **49**: 758–764, MR 0018142 (https://www.ams.org/mathscinet-getitem?mr=0018142), *Indagationes Mathematicae* **8**: 461–467.
- de Bruijn, N. G. (1975), *Acknowledgement of Priority to C. Flye Sainte-Marie on the counting of circular arrangements of 2^n zeros and ones that show each n-letter word exactly once*

(http://alexandria.tue.nl/repository/books/252901.pdf) (PDF), T.H.-Report 75-WSK-06, Technological University Eindhoven.
- Fredricksen, Harold; Maiorana, James (1978), "Necklaces of beads in *k* colors and *k*-ary de Bruijn sequences", *Discrete Mathematics* **23** (3): 207–210, doi:10.1016/0012-365X(78)90002-X (https://dx.doi.org/10.1016%2F0012-365X%2878%2990002-X), MR 523071 (https://www.ams.org/mathscinet-getitem?mr=523071).
- Hurlbert, Glenn; Isaak, Garth (1993), "On the de Bruijn torus problem" (http://math.la.asu.edu/~hurlbert/papers/DBTP.pdf) (PDF), *Journal of Combinatorial Theory*, Series A **64** (1): 50–62, doi:10.1016/0097-3165(93)90087-O (https://dx.doi.org/10.1016%2F0097-3165%2893%2990087-O), MR 1239511 (https://www.ams.org/mathscinet-getitem?mr=1239511).
- Martin, M. H. (1934), "A problem in arrangements" (http://www.ams.org/journals/bull/1934-40-12/S0002-9904-1934-05988-3/S0002-9904-1934-05988-3.pdf) (PDF), *Bulletin of the American Mathematical Society* **40** (12): 859–864, doi:10.1090/S0002-9904-1934-05988-3 (https://dx.doi.org/10.1090%2FS0002-9904-1934-05988-3), MR 1562989 (https://www.ams.org/mathscinet-getitem?mr=1562989).
- Ralston, Anthony (1982), "de Bruijn sequences—a model example of the interaction of discrete mathematics and computer science", *Mathematics Magazine* **55** (3): 131–143, doi:10.2307/2690079 (https://dx.doi.org/10.2307%2F2690079), MR 653429 (https://www.ams.org/mathscinet-getitem?mr=653429).
- Tuliani, Jonathan (2001), "de Bruijn sequences with efficient decoding algorithms", *Discrete Mathematics* **226** (1-3): 313–336, doi:10.1016/S0012-365X(00)00117-5 (https://dx.doi.org/10.1016%2FS0012-365X%2800%2900117-5), MR 1802599 (https://www.ams.org/mathscinet-getitem?mr=1802599).

External links

- Weisstein, Eric W., "de Bruijn Sequence" (http://mathworld.wolfram.com/deBruijnSequence.html), *MathWorld*.
- "Sloane's A166315 : Lexicographically smallest binary de Bruijn sequences (http://oeis.org/A166315)", *The On-Line Encyclopedia of Integer Sequences*. OEIS Foundation.
- De Bruijn sequence (http://chessprogramming.wikispaces.com/De+Bruijn+sequence)
- Combinatorial Object Server (http://www.theory.csc.uvic.ca/~cos/), includes a De Bruijn sequence generator among many others
- CGI generator (http://www.hakank.org/comb/debruijn.cgi)
- Applet generator (http://www.hakank.org/comb/deBruijnApplet.html)
- Javascript generator and decoder (http://jgeisler0303.github.io/deBruijnDecode/). Implementation of J. Tuliani's algorithm.
- Door code lock (http://www.stefangeens.com/000435.html)
- Minimal arrays containing all sub-array combinations of symbols: De Bruijn sequences and tori (http://lcni.uoregon.edu/~dow/Geek_art/Minimal_combinatorics/Minimal_arrays_containing_all_combinations.html)

Categories: Binary sequences | Enumerative combinatorics
