Langford Pairings Exercise Answers

Anakin Dey

Last Edited on 2/13/23 at 16:42

Langford Pairing for $n = 4$

$[4, 1, 3, 1, 2, 4, 3, 2]$ or $[2, 3, 4, 2, 1, 3, 1, 4]$

Non-existence of Langford Pairings for $n = 1$ or $n = 2$

If $n = 1$, then we have no other numbers to put between the two 1's and so no pairing can exist. If $n = 2$, then the two 2's must be at the ends of the list: $[2, \_, \_, 2]$.

Program for Generating a Single Langford Pairing

See the function `generate_single_pairing` on GitHub

Exercise 6a of [Knu11, Chapter 7]

Note to the reader: I recommend you implement these formulas using Sympy (ask in the Discord for how to use it) and compute the polynomials for small n to gain intuition. These formulas are not intuitive, especially how everything cancels out. This is common in combinatorics when trying to deal with polynomials to count objects.

Let L_n be the number of Langford pairings for n, not counting reversals as distinct pairings, and let

$$f(x_1, \ldots, x_{2n}) = \prod_{k=1}^{n} \left(x_k x_{n+k} \sum_{j=1}^{2n-k-1} x_j x_{j+k+1} \right).$$

We want to show that

$$\sum_{x_1, \ldots, x_{2n} \in \{-1,1\}} f(x_1, \ldots, x_{2n}) = 2^{2n+1} L_n.$$

Think of each x_i as a location variable for the resulting list. For $n = 3$, $f(x_1, \ldots, x_6)$ is equal to

$$x_1 x_4 \big(x_1 x_3 + x_2 x_4 + x_3 x_5 x_4 x_6\big) \cdot x_2 x_5 \big(x_1 x_4 + x_2 x_5 + x_3 x_6\big) \cdot x_3 x_6 (x_1 x_5 + x_2 x_6)$$

So each of the terms in the parentheses correspond to possible positions for various numbers. For example $\big(x_1 x_3 + x_2 x_4 + x_3 x_5 x_4 x_6\big)$ lists out the valid positions for the two 1's. Expanding this sum out yields terms such that $x_2^2 x_4^2 x_3^2 x_6^2 x_1^2 x_5^2$ corresponding to the pairing $[3, 1, 2, 1, 3, 2]$. In general, terms containing all of the x_i's as powers of two will correspond to Langford pairings. Since all the x_i's in these terms have even degree, plugging in 1 or -1 will always yield 1 at the end. Every other term not corresponding to a Langford pairing will have at least one term of degree 1. Iterating over every possible $x_1, \ldots, x_{2n}$ will lead to the cancelling of all these other terms. Thus, all we are left with is the sum of 1's, each 1 corresponding to a valid pairing.

There are 2^{2n} possible values for the tuple $(x_1, \ldots, x_{2n})$ where each x_i takes on one of two values, leading to an overcount by a factor of 2^{2n}. Then we also consider reversals of a Langford pairing to be the same, leading to an overcount of a factor of 2. This explains the 2^{2n+1} coefficient on L_n.

For those interested, part (b) of the question utilizes the Gray Code from last week's meeting to speed up the computation and part (c) uses some clever math to further speed it up. Knuth has some good writing on this in his solutions in [Knu11].

Exercise 15 of [Knu22, Chapter 7.2.2.1]

There are two ways we can modify our formulation of Langford pairing as an exact cover problem to exclude finding both a pairing and it's reverse, leading to a significant speedup. Recall that each option takes the form $i: \left[l_j, l_k\right]$, $k = j+i+1$, meaning i appears in slot j and k in the list. Our two options study where the smallest or one of the largest elements must appear in the resulting pairing.

Option 1: Let $[n \text{ even}]$ be a quantity equal to 1 if n is even and 0 otherwise. We can exclude an option $i: \left[l_j, l_k\right]$ if both $i = n - [n \text{ even}]$ and $j > \frac{n}{2}$. This corresponds to pairings where the first time i appears is after the first quarter of the list and where the second time i appears in in the last quarter of the list. In the reversal of these pairings, i would appear in the first quarter, and thus this excludes reversals.

This excludes $\left\lfloor \frac{n}{2} \right\rfloor$ options.

Option 2: We could more simply omit pairings where $i = 1$ and $j \geq n$. Thus, the second 1 must appear strictly in the second half. Thus in the reversal, the first i must appear at a slot $j < n$. This also omits the reversals.

This excludes $n-1$ options.

Program for Generating all Langford Pairings

See the function `find_pairings` on GitHub

References

[Knu11] Donald E. Knuth. *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1*. Addison-Wesley Professional, 2011. ISBN: 0201038048.

[Knu22] Donald E. Knuth. *The Art of Computer Programming, Volume 4B: Combinatorial Algorithms, Part 2*. 1st. Addison-Wesley Professional, 2022. ISBN: 0201038064.