

Pyramid Schemes for Eating M&Ms: Enumeration, Generation, and Gray Codes

Elizabeth Hartung ✉ 

Department of Mathematics, Massachusetts College of Liberal Arts, North Adams, MA, USA


Brett Stevens ✉ 

School of Mathematics and Statistics, Carleton University, Ottawa, Canada

Aaron Williams ✉ 

Department of Computer Science, Williams College, Williamstown, MA, USA

Abstract

Consider the following problem. You have a rainbow pyramid of M&Ms with n rows. For example, when $n = 4$ you may have one red, two orange, three yellow, and four green . You want to eat n of the M&Ms in such a way that the remaining M&Ms can be rearranged into a rainbow pyramid with $n - 1$ rows. Two approaches are distinct if a different number from a particular row are eaten. In other words, we only care about the multiset of row frequencies (or colours) that are eaten and not the order in which they are eaten. One solution eats one M&M per row (e.g., $1234 \rightarrow \text{img alt="M&M icons" data-bbox="635 355 655 370}.$ Another eats the entire bottom row (e.g., $4444 \rightarrow \text{img alt="M&M icons" data-bbox="635 370 655 385}).$ How many different solutions are there? We show that the answer is 2^{n-1} . Furthermore, each solution can be naturally encoded with combinatorial objects enumerated by 2^{n-1} including binary words of length $n - 1$, compositions of n , and subsets of $[n - 1]$. Less obviously they are encoded by *M&M permutations* where each value in $[n]$ is at most one position to the right of its position in the identity (e.g., 123, 132, 213, 312 for $n = 3$).

What if at most m from each row can be eaten? When $m = 1$ the only solution is to eat one of each colour. Otherwise, the solutions are counted by Fibonacci ($m = 2$), Tribonacci ($m = 3$), Tetranacci ($m = 4$), and so on, up to 2^{n-1} ($m = n$). Furthermore, solutions can be naturally encoded by limited versions of the aforementioned objects including binary strings avoiding the substring 0^m and M&M permutations where values are limited by moving at most $\ell = m - 1$ positions to the left.

Motivated by the works of Samuel Beckett, we consider minimal-change orders of the solutions. We obtain a satisfying result by filtering the binary reflected Gray code to words avoiding 0^m . For example, when $n = 4$ we have $\text{BRGC}(n) = 000, 100, 110, 010, 011, 111, 101, 001$ and the words avoiding 00 are $\text{BRGC}_\ell(3) = 110, 010, 011, 111, 101$ where $\ell = 1$ is the limit on the run-lengths of 0s. Our bijection then creates solutions that differ in by a single M&M $1244, 2244, 2234, 1234, 1334$. Thus, Beckett's character Murphy can imagine every experience by changing one M&M at a time.

The generalized Gray code $\text{BRGC}_\ell(n)$ was previously defined recursively [Bernini et. al *Acta Informatica* 2015] with its change sequence supporting amortized $\mathcal{O}(1)$ -time generation [Arndt *Matters Computational* 2010]. We uncover a simple greedy definition – flip the leftmost bit that creates a new binary word avoiding 0^m starting from $w = \dots 110^\ell 110^\ell$ – and a successor rule that supports loopless worst-case $\mathcal{O}(1)$ -time generation. Furthermore, the corresponding limited M&M permutations are greedily generated by swapping the smallest value (or the leftmost pair of adjacent values) that gives a valid new permutation (e.g., $\overline{1}243, 21\overline{4}3, \overline{2}134, 1\overline{2}34, 1324$ for $n = 4$ and $\ell = 1$).

We also consider a relaxed version of the problem in which the initial pyramid's n rows have respective widths $r, r + 1, r + 2, \dots, n, n, \dots, n$. Here the answer is an n -term product $\langle n, r \rangle! = 1 \cdot 2 \cdot 3 \cdots r \cdot (r + 1) \cdot (r + 1) \cdots (r + 1)$ that we refer to as a *flatorial number*. Furthermore, the solutions are represented by a generalization of M&M permutations in which each symbol can appear at most r positions to the right of its position in the identity. We complete our investigation by showing that eight distinct classes of permutations are enumerated by flatorial numbers.

2012 ACM Subject Classification Mathematics of computing \rightarrow Combinatorial algorithms; Mathematics of computing \rightarrow Combinatorics on words

Keywords and phrases combinatorial enumeration, generation, Gray code, loopless algorithm

Digital Object Identifier 10.4230/LIPIcs.FUN.2026.23



© Elizabeth Hartung, Brett Stevens, and Aaron Williams; licensed under Creative Commons License CC-BY 4.0

13th International Conference on Fun with Algorithms (FUN 2026).

Editor: John Iacono; Article No. 23; pp. 23:1–23:24



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Funding *Brett Stevens*: supported by the Natural Sciences and Engineering Research Council of Canada (funding reference number RGPIN-2023-04668).

1 Introduction

We investigate a fun problem invented by the second author while eating colourful pyramids of chocolate M&Ms. In brief, an *M&M pyramid of height n* has a top row of one M&M, a second row of two M&Ms, a third row of three M&Ms, and so on, with the property that the colours are the same on each row and different on distinct rows. The *M&M Problem* asks for the number of distinct ways that n of the M&Ms can be removed (i.e., eaten!) so that the remaining M&Ms can be rearranged to form an M&M pyramid of height $n - 1$. Here we care only about the quantity of each colour eaten and not the order that they were enjoyed¹. Figure 1 illustrates that the answer for $n = 4$ is 8. In particular, note that leftmost solution eats the entire bottom green row while the rightmost eats one of each colour.

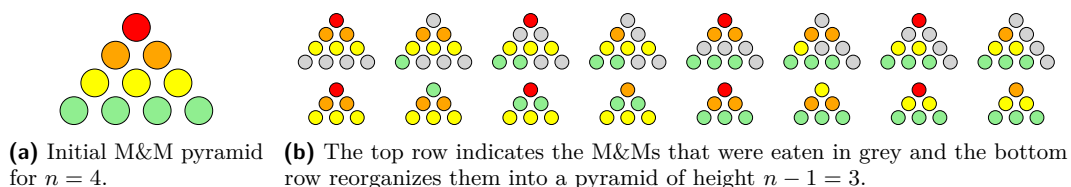


Figure 1 The eight deliciously eaten solutions for $n = 4$.

Section 2 answers this snack-sized problem by proving that there are 2^{n-1} solutions. Furthermore, each solution is naturally represented as a binary string of length $n - 1$, a composition of n , a subset of $[n - 1]$, or as a permutation of $[n]$ in which each symbol appears at most one position to the right relative to its natural position in the identity $12 \dots n$.

In Section 3 we lower the number of possible solutions using a parameter m . In the *limited version* of the problem at most m of each colour can be eaten. There is one solution when $m = 1$ (i.e., eat one of each colour). Otherwise, the answers are Fibonacci ($m = 2$), Tribonacci ($m = 3$), Tetranacci ($m = 4$), and so on (with the base cases omitted), up to the unlimited problem with 2^{n-1} solutions ($m = n$). The corresponding objects are also limited in natural ways, including binary strings avoiding 0^m and permutations in which each value is at most one position to the right and $\ell = m - 1$ positions to the left relative to the identity.

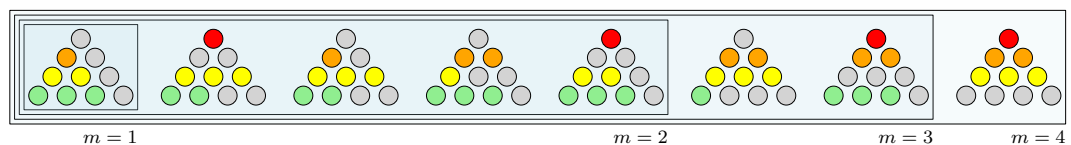
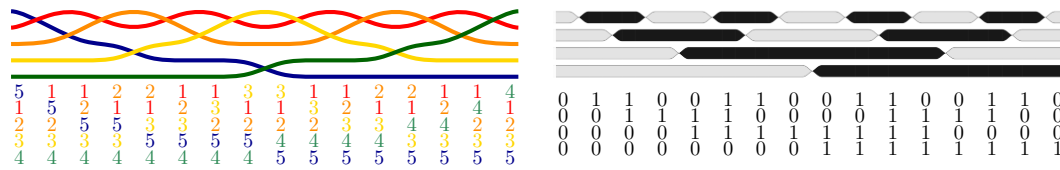


Figure 2 Solutions to the limited problem with $n = 4$ and each $m = 1, 2, 3, 4$. For example, there are seven solutions when $m = 3$ as the only forbidden solution eats the entire bottom row of size 4.

Our first result implies that the number of *M&M permutations* of $[n]$ is 2^{n-1} . Perhaps the most well-known class of permutations of this size is peakless permutations. Those permutations play a central role in the recent *Combinatorial Generation by Permutation*

¹ In other words, we don't care if you "eat the red ones last." This was a slogan for the Canadian version of this type of chocolate candy known as *Smarties*. In the United States you can also play this game with *Smarties* although they are the pastel-colored discs that are known as *Rockets* in Canada! [3, 14].



■ **Figure 3** Gray codes of (a) M&M permutations of $[n]$ and (b) binary strings with $n - 1$ bits. Their change sequence is the binary ruler sequence 1, 2, 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1, 2, 1 (OEIS A001511).

Languages series [25] which is currently on its seventh entry [9, 10]. In particular, the bijection between peakless permutations and binary strings laid the foundation for generalizations of the binary reflected Gray code to larger permutation languages. In this paper we provide a *swap Gray code* for M&M permutations meaning that successive permutations differ by swapping consecutive entries. The order is visualized as a weaving pattern in Figure 3a (using the ribbon style from [40]). If the pattern looks familiar, it is because it is identical to the binary reflected Gray code! Indeed, the crossings in 3a line up exactly with the twists in Figure 3b. Pleasantly, we obtain similar Gray codes for the corresponding objects with parameter m (or $\ell = m - 1$) including binary strings avoiding 0^m , and M&M permutations in which each value moves at most ℓ positions to the left. Furthermore, the corresponding rearranged pyramids (see Figure 6) differ in the colour of a single M&M!

In Section 5 we turn our attention to generation algorithms. We explain how the Gray codes parameterized by m (or $\ell = m - 1$) are generated by simple greedy algorithms. For example, the bitstrings avoiding 0^m are generated by greedily flipping the leftmost bit, while greedily swapping the leftmost pair of values produces the M&M permutations where each value moves at most ℓ positions to the left. Surprisingly, the two greedy algorithms use the same change sequence with bit b_i flipping when adjacent values $p_i p_{i+1}$ swap. The same order also produces a pleasant Gray code for the rearranged M&M pyramids: the colour of a single eaten M&M is changed.

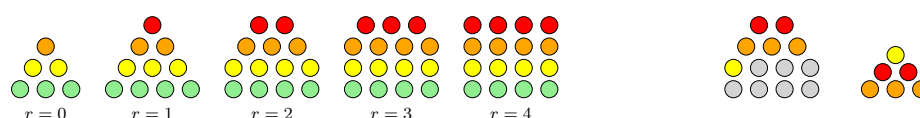
While the algorithms in Section 5 are simple, they are not efficient due to memory usage. In Section 6 we derive the first successor rule for the Gray code of binary strings avoiding 0^m . This allows us to create generation algorithms that are *memoryless* (i.e., previously generated objects are not remembered) and *loopless* (i.e., worst-case $O(1)$ -time per object) which is optimal in terms of time efficiency.

In Section 7 we return to counting M&Ms, but this time we raise the number of possible solutions by introducing a parameter r . In the *relaxed problem* we widen (or narrow) the pyramid with n rows as follows. The top row has r M&Ms, the second row has $r + 1$ M&Ms, the third row has $r + 2$ M&Ms, and so on, except that each row also has width at most n . The initial shapes for $n = 4$ are shown in Figure 4. The goal remains the same: eat M&Ms so that the remaining M&Ms can be rearranged into a standard pyramid shape with n rows.

- When $r = 0$ there is only one solution as the initial shape is a pyramid with $n - 1$ rows.
- When $r = 1$ we have the original problem so the number of solutions is 2^{n-1} .
- When $r = n - 1$ or $r = n$ the number of solutions is $n!$ since any permutation of the colours is possible for the remaining pyramid.

More generally, we prove that the number of solutions is the n -term product $\langle n, r \rangle! = 1 \cdot 2 \cdot 3 \cdots r \cdot r + 1 \cdot r + 1 \cdots r + 1$ which we refer to as a *flatorial number*.

23:4 Pyramid Schemes for Eating M&Ms: Enumeration, Generation, and Gray Codes



(a) The initial pyramid shapes for $r = 0, 1, 2, 3, 4$. The top row has width r and successive rows widen until width n . (b) Eating the M&Ms in grey (left) leaves a pyramid of height $n - 1 = 3$ (right).

■ **Figure 4** (a) The initial shapes of the M&Ms for the relaxed problem for $n = 4$ and $r = 0, 1, 2, 3, 4$ and (b) a solution for $n = 4$ and $r = 2$. The solution in (b) is not possible in the original problem as the new pyramid has two copies of the original top row in red.

2 The M&M Problem

In North America many holidays involve the giving and receiving of treats that often come in different colours, but are otherwise similar. At Easter there are Cadbury's mini eggs; at Halloween there are many examples including M&Ms, Resses Pieces, Rockets, Smarties, Skittles; at Christmas there are Hershey's kisses in holiday colours; Chocolate Chanukkah gelt is sometimes given during Hanukkah which varies in size (or coin denomination) rather than colour. Different candies come in different numbers of colours as shown in Table 1 which can aid you in determining what confectionary to purchase as you explore the mathematics we discuss². To make our work as accessible as possible to a more healthy dietary exploration, we note that mixed nuts could easily include up to nine different types: almond, brazil, cashew, hazelnuts, macadamia, peanut, pecans, pistachios, walnuts. The problem originated with M&Ms and since Mars Wrigley Confectionery does produce at least nineteen different colours, allowing for large pyramids, we will discuss the problem in the context of M&Ms.

■ **Table 1** Readers are encouraged to research these variations of the M&M Problem!

Candy product	number of types (colours)
Smarties 🇨🇦	8
M&Ms, Rockets 🇨🇦, Bassett's Jelly Babies, Wine gums (colours)	6
Skittles, Haribo gummy bears, Wine gums (shapes)	5
Cadbury's mini eggs	4
Reese's Pieces, Holiday Hershey's Kisses	3

When reaching into a bag of M&Ms and pulling out a handful, usually the distribution of colours is not uniform and some colours appear more often and some only a few times. Noticing this in a sample handful of M&Ms one time, the second author started to arrange them on the table in order of the number of times a colour appeared. Frequently these distributions were close to being a pyramid. While it is a natural impulse to start nibbling on some M&Ms from the handful, for a mathematician it was perhaps inevitable to start with the few necessary to create a true pyramid. Once having a pyramid with n different rows (colours), the author wondered how he could eat n of the M&Ms and obtain a pyramid with $n - 1$ different rows.

Given a rainbow pyramid of height n , in how many ways can we remove a total of n so that the remainder can form a rainbow pyramid of height $n - 1$?

² Remember to brush and floss every day while engaging in such research.

The largest colour class in a pyramid with n rows is size n so it is clear that we must eat at least one M&M from the largest class. Once this is eaten there are now two colour classes that have size $n - 1$. One and only one of them must remain at the end when we produce a pyramid with $n - 1$ rows, so we have a choice between eating one M&M from exactly one of these two colour classes. At each stage, after we have eaten one M&M from our choice of exactly two colour classes of size i , all the colour classes of size i and larger in the pyramid with $n - 1$ rows will have been determined and no more M&Ms can be eaten from any of these size classes. We have two colour classes of size $i - 1$ and the colour class of size $i - 1$ in the pyramid with $n - 1$ rows must be one of these. Therefore, we must eat one M&M from exactly one of these classes of size $i - 1$ and an induction proves the following theorem.

► **Theorem 1.** *There are 2^{n-1} ways to remove n objects from a rainbow pyramid of height n so that the remainder can form a rainbow pyramid of height $n - 1$.*

Before moving on to other mathematical connections, we note a recent amusing experience. The second author was eating a rainbow pyramid of a more healthy food (cherry tomatoes) where he has a prejudice against the red tomatoes. Whatever rainbow pyramid he initially started with, he found once he reached the next smaller rainbow pyramid he was inevitably left without any red tomatoes due to this bias. Thus when there are i red tomatoes in a rainbow pyramid with n rows, instead of 2^{n-1} possible eating paths to the next rainbow pyramid, he was left with 2^{n-i} eating paths³. In this case, the red tomatoes were more numerous than the other colours and in his lunch the red colour class was often the largest and frequently he was left with no choices at all. Samuel Beckett has a delightful expression of how our prejudices, both positive and negative, reduce the fullness of our experiences in life (also featuring permutations) in his novel *Murphy* [4].

Murphy receded a little way into the north [of the park] and prepared to finish his lunch. He took the biscuits carefully out of the packet and laid them face upward on the grass, in order as he felt of edibility. They were the same as always, a Ginger, an Osborne, a Digestive, a Petit Beurre and one anonymous. He always ate the first-named last, because he liked it the best, and the anonymous first, because he thought it very likely the least palatable. The order in which he ate the remaining three was indifferent to him and varied irregularly from day to day. On his knees now before the five it struck him for the first time that these prepossessions reduced to a paltry six the number of ways in which he could make this meal. But this was to violate the very essence of assortment, this was red permanganate on the Rima of variety. Even if he conquered his prejudice against the anonymous, still there would be only twenty-four ways in which the biscuits could be eaten. But were he to take the final step and overcome his infatuation with the ginger, then the assortment would spring to life before him, dancing the radiant measure of its total permutability, edible in a hundred and twenty ways!

Overcome by these perspectives Murphy fell forward on his face in the grass, beside those biscuits of which it could be said as truly as of the stars, that one differed from another, but of which he could not partake in their fullness until he had learnt not to prefer any one to any other.

Murphy is “wrestling with the demon of gingerbread” when he is distracted by a woman speaking. While in conversation the woman’s dog eats all the biscuits except the ginger.

³ Exercise left to the reader.

There are other combinatorial classes of objects that have cardinality 2^{n-1} . Here we consider binary strings, subsets, compositions, and a special type of permutation, and show that they have natural bijections to our rainbow pyramid solutions. These results are then summarized in Theorem 2 and illustrated in Example 3 and Figure 5.

In the sequence of binary choices outlined above, we start by eating one M&M from the largest class. Then at stage i we are always choosing between eating one of the existing colour class of size i or one of the colour class most recently eaten. If we encode these choices as $b_i = 1, 0$ respectively, then we form a bijection to the binary strings $b_1 b_2 \cdots b_{n-1}$ of length $n-1$. We note that the sequence of decisions are made in time as the b_i are listed from right to left in this string. If $b_i = 1$, then we will have the opportunity to eat more of that colour, but if $b_i = 0$ then we will not. Thus if $b_i = 0$ this indicates that no M&M of colour i is ever eaten and that colour will be the colour class of size i in the obtained rainbow pyramid. Thus the values of i for which $b_i = 1$ are precisely the colours where at least one M&M of that colour are eaten. The positions of the zeros give a bijection to the subsets of $[n-1]$ via the “uneaten map” and the positions of the ones give another bijection to the subsets of $[n-1]$ via the “(at least one) eaten map” (disregarding the colour n which must always be eaten).

Labelling the M&Ms in the colour class of size i with the label i for simplicity, we can form the sequence of the number of i s that we ate, left to right, from 1 to n , “eaten representation”. In eaten representation e we will say that e_i is the number of items labelled i that are eaten. This is an ordered summation to n , allowing zeros, such that the summand in position i must be no larger than i and the summand in position n is at least 1. Observe that the position of the zeros match the positions of the zeros in the binary string representation. If we drop the zeros we obtain a simple composition of n and thus a map from the eaten representation to the compositions of n . We claim that this map is injective. It is easy to check that this is true for small values of n . For an induction, now suppose that there are two eaten representations, e and f , mapping to the same composition of n and that we have the smallest value of n for which this is possible. Since at least one item labelled n is eaten we know that the rightmost summand of the composition must be the number of n s eaten, and that these two values agree in both composition and thus agree in the eaten representation. For each n eaten after the first, the next smaller colour class is ignored. Thus $e_n = f_n$ and for both eaten representations the next rightmost non-zero value is in position $n - e_n$. Thus the eaten representation from position $n - e_n$ left is a valid eaten representation for a rainbow pyramid of height $n - e_n$ and induction gives the result. Since the number of eaten representations and number of compositions of n both have cardinality 2^{n-1} this gives a bijection between the two sets.

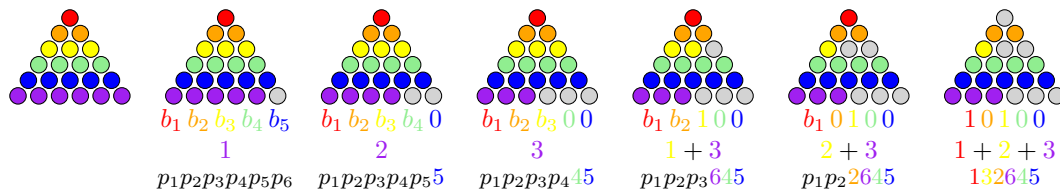
Again labeling the items in the colour class of size i with label i , the starting rainbow pyramid is naturally represented by the identity permutation $12 \cdots n$. After having eaten the n M&Ms to obtain the new rainbow pyramid, add back one M&M of each colour to produce another pyramid of the original size and consider its corresponding permutation, π , of $1, 2, \dots, n$. The positions of the zeros in the binary string representation (and also the zeros in the eaten representation) correspond exactly to the set of colours that were not eaten at all. These colour classes stayed the same size and then add the one additional M&M added so in π have moved right exactly one position from their starting position. Since there were i items labeled i in the starting rainbow pyramid, no label can move more than one position to the right. The new permutation π is uniquely determined by what has been eaten so this gives a bijection to the set of permutations of size n such that each symbol appears at most one position to the right relative to its natural position in the identity $12 \cdots n$. That this family had cardinality 2^{n-1} was observed by Arndt [2] in work that we discuss later on.

► **Theorem 2.** *The following objects are equinumerous and each pair has $\mathcal{O}(n)$ -time bijections.*

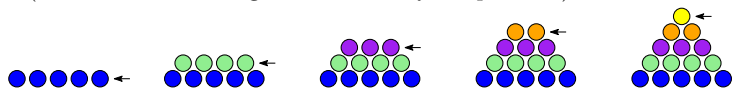
1. Solutions to the rainbow pyramid problem
2. Binary strings of length $n - 1$.
3. Subsets of $[n - 1]$.
4. Compositions of n .
5. Permutations of $[n]$ where each value appears at most one position to the right relative to its natural position in the identity permutation. These are the M&M permutations $\mathbb{M}(5)$.

► **Example 3.** Suppose that we start with a rainbow pyramid of height 6 which is naturally represented by the permutation 123456 and illustrated at the left in Figure 5a. Consider the solution that eats M&Ms in the following order: 6, 6, 6, 3, 3, 1. Eating the first 6 is forced and that fact that the next two M&Ms eaten are 6s indicates that $b_5 = b_4 = 0$: we did not eat a 4 or 5 when we had the opportunity. Our choice to then eat a 3 instead of another 6 gives $b_3 = 1$. Our remaining choices are encoded with $b_2 = 0$ and $b_1 = 1$ giving binary representation 10100. The eaten representation simply records how many of each colour we ate: 1, 0, 2, 0, 0, 3. This corresponds to the composition $1 + 2 + 3$ of $n = 6$.

At the end our rainbow pyramid is represented by 32645. When we add back one M&M of each colour we have the permutation representation 132645. In this permutation 2, 4 and 5 have moved to the right one position relative to their position in the identity permutation. Note that these colours which moved to the right are precisely the colours which were not eaten and correspond exactly to the positions of zeros in the binary string representation. Furthermore subtracting one from the non zero values of the eaten representation give the number of places to the left these colours moved: 6 moved $3 - 1 = 2$ positions to the left; 3 moved $2 - 1 = 1$ position to the left and 1 moved $1 - 1 = 0$ positions to the left.



(a) Each successive decision contributes one bit to the binary representation, a unit value to the composition, and one digit to the permutation (with the final first digit determined by the previous).



(b) Each successive decision determines the next higher row in the pyramid that will be formed after rearranging the remaining M&Ms.

■ **Figure 5** Visualizing the steps taken to create the single solution to the M&M Problem for $n = 6$ from Example 3. The initial pyramid of height $n = 6$ appears on the left of (a) and the final rearranged pyramid of height $n - 1 = 5$ appears on the right of (b).

3 A Limited M&M Problem with Parameter ℓ

Suppose that we want to avoid eating too many M&Ms of the same colour. More specifically, we limit ourselves to eating at most m copies of any given colour, where m is some fixed constant. This leads to a generalization of our original problem parameterized by m .

Given a rainbow pyramid of height n , in how many ways can we remove at most m of each colour so that the remainder can form a rainbow pyramid of height $n - 1$?

Once again we consider solutions as unordered selections of colours, and every solution removes exactly n of the M&Ms. Now let us consider how our various representations are modified under this chromatic limitation. For some objects it is convenient to use $\ell = m - 1$.

Binary strings of length $n - 1$. As in the original problem we know that at least one copy of colour n is removed, and bit $b_i = 1$ when at least one copy of colour i is removed. However, not every binary string of length $n - 1$ encodes a valid solution when the use of each colour is limited. Indeed, the following points show that a solution is valid, if and only if, the binary string avoids substrings of the form 0^m (or equivalently, 0^ℓ is the longest run of 0s allowed).

- Substring $b_{i-m}b_{i-m+1}\cdots b_i = 00\cdots 01 = 0^m1$ indicates m copies of colour i are removed.
- Suffix $b_{n-m+1}b_{n-m+2}\cdots b_n = 00\cdots 0 = 0^m$ indicates m copies of colour n are removed.

Subsets of $[n - 1]$. Consecutive values in the subset (union with $\{0, n\}$) differ by at most m .

Compositions of n . Each part is at most m .

Permutations of $[n]$. Each symbol can move at most one position to the right and at most ℓ positions to the left relative to the identity.

These observations lead to the following theorem. Note that the correspondences with the M&M Problem are new, while those involving the other M&M objects can be found in various textbooks on combinatorics [28].

► **Theorem 4.** *The following objects are equinumerous.*

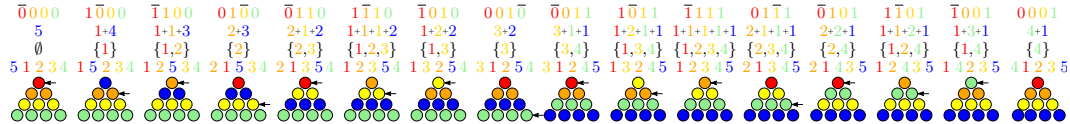
1. *Solutions to the rainbow pyramid problem with limit m .*
2. *Binary strings of length $n - 1$ without any substring of the form 0^m .*
3. *Subsets of $[n - 1]$ union with $\{0, n\}$ in which consecutive values differ by at most m .*
4. *Compositions of n in which each part is at most m .*
5. *Permutations of $[n]$ in which each symbol appears at most one position to the right and at most $\ell = m - 1$ positions to the left relative to the identity.*

4 Gray Codes

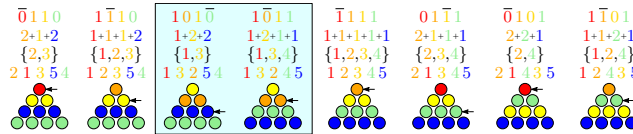
Given a combinatorial object of a particular type and size (e.g., n -bit binary strings, permutations of $[n]$, binary trees with n nodes) a *Gray code* orders the objects such that consecutive objects are similar to each other in some particular way. Humans with various backgrounds have been constructing these types of patterns for hundreds of years. For example, bell-ringers in the 1600s created *plain changes* [16, 52] which orders the $n!$ permutations of $[n]$ so that consecutive permutations differ by swapping two consecutive values in one-line notation (e.g., $\overline{1}23, \overline{2}1\overline{3}, \overline{2}3\overline{1}, \overline{3}2\overline{1}, \overline{3}1\overline{2}, 132$ for $n = 3$). Plain changes was rediscovered multiple times in the 1960s leading to its alternate moniker: the Steinhaus-Johnson-Trotter algorithm [53, 30, 56]. Another Gray code for permutations, this time using prefix-reversals, was discovered in the late 1700s by Klügel [29] (e.g., $\overleftarrow{1}23, \overleftarrow{2}13, \overleftarrow{3}12, \overleftarrow{1}32, \overleftarrow{2}31, \overleftarrow{3}21$).

The obsession with listing all possibilities with small changes between them also extends to Beckett and his works. In particular, *Quad* [5] is a play with four actors in which every non-empty subset of actors appear onstage in successive scenes with a single performer entering or exiting to create the next scene. Furthermore, Beckett required that the actor who had been offstage the longest would always be the one who entered. This idea led to the notion of a *Beckett Gray code* [49, 60, 15]. Similarly, it is easy to imagine Murphy obsessing over similar issues when eating his cookies.

In this section we discuss the binary reflected Gray code, the concepts of sublist and simultaneous Gray code, and how these ideas apply to our M&M objects parameterized by m . The results are illustrated in Figure 6.



(a) Gray codes that correspond to the binary reflected Gray code BRGC(4) found in the top row.



(b) Gray codes that correspond to the limited binary reflected Gray code BRGC₁(4) found in the top row. This sublist has different successors (e.g., 1010 = 1011 is highlighted above versus 1010 = 0010 in (a)).

■ **Figure 6** Gray codes of M&M objects for (a) $n = 5$ and (b) $n = 5$ with limit $m = 2$ (or $\ell = 1$). The Gray codes follow the top rows which are flip Gray codes for (a) $\mathbb{B}(4)$ and (b) $\mathbb{B}_1(4)$. The former uses the binary reflected Gray code BRGC(4), while the latter uses its sublist BRGC₁(4) of words that avoid $0^m = 00$. Subsequent rows contain simultaneous Gray codes for (a) M&M permutations, compositions, subsets, and rearranged pyramids with respective limited versions in (b). In particular, the M&M permutations (a) $\mathbb{M}(5)$ and (b) $\mathbb{M}_1(5)$ appear in their regular swap Gray codes Reg(5) and Reg₁(5), respectively. The rearranged pyramids are also in a Gray code order, where each arrow indicates a single extra M&M to eat (which uniquely determines the M&M colour to not eat) relative to the previous solution.

4.1 Binary Reflected Gray Code

The most widely known Gray code is the *binary reflected Gray code (BRGC)*. The BRGC orders the 2^n binary strings with n bits so that consecutive strings differ in a single bit (e.g., 000, 100, 110, 010, 011, 111, 101, 001 for $n = 3$). The order is attributed to Frank Gray at Bell Labs due to his 1953 patent involving televisions [21]; see Ruskey [41] and Knuth [32] for additional historical details. In particular, the attribution is a strong case of Stigler’s Law of Eponymy [55] as the order appeared in an earlier patent also from Bell Labs [54]!

The word *reflected* refers to writing a list of strings in reverse (e.g., *Algorithms with Fun*) and the *reflected code* (as Gray called it) is constructed recursively by prefixing 0 to every string of length $n - 1$, followed by prefixing 1 to every string of length $n - 1$ in reflected order. This recursive construction is formalized below with base case ϵ (i.e., the empty string)

$$\text{BRGC}(n) = \text{BRGC}(n - 1) \cdot 0, \text{BRGC}(n - 1)^R \cdot 1 \text{ with } \text{BRGC}(0) = \epsilon \tag{1}$$

where \cdot concatenates a bit to every string in a list, and the commas combine two lists.

► **Example 5.** $\text{BRGC}(2) = \text{BRGC}(1) \cdot 0, \text{BRGC}(1)^R \cdot 1 = (0, 1) \cdot 0, (1, 0) \cdot 1 = 00, 10, 11, 01$.

A *change sequence* is a list of successive changes made in a Gray code. For the binary reflected Gray code, the list of indices that are flipped form the well-known *binary ruler sequence* (A001511) [38]. The sequence can be defined recursively as follows with base case \emptyset (i.e., empty sequence)

$$\text{ruler}(n) = \text{ruler}(n-1), n, \text{ruler}(n-1) \text{ with } \text{ruler}(0) = \emptyset. \tag{2}$$

It would be reasonable to expect the second copy of ruler($n - 1$) to be reversed due to the reflection in BRGC(n). This is correct, but not necessary, as ruler(n) is always a palindrome.

► **Example 6.** The bit indices that are flipped in $\text{BRGC}(2) = \bar{0}0, 1\bar{0}, \bar{1}1, 10$ are $\text{ruler}(2) = 1, 2, 1$, so the indices that are flipped in $\text{BRGC}(3)$ are $\text{ruler}(3) = \text{ruler}(2), 3, \text{ruler}(2) = 1, 2, 1, 3, 1, 2, 1$.

4.2 Sublist Gray Codes

Given a Gray code order for a set of objects, we can create a list of any subset by considering the corresponding sublist. In other words, we can use the induced suborder. If the result is also a Gray code, then we refer to it as a *sublist Gray code*. One of the first results of this type is that inducing the binary reflected Gray code on fixed-weight binary strings (i.e., strings with a fixed number of 1s) produces a transposition Gray code [58].

The binary reflected Gray code also provides a sublist Gray code for the strings that avoid 0^m in which successive strings differ by the flip of a single bit. This result was proven as part of a larger study on avoiding specific substrings (also known as *factors*) in the reflected Gray code for q -ary strings [6]. More generally, Gray codes for words avoiding certain factors have been studied for at least 30 years [50]. We denote the set of binary strings avoiding 0^m as $\mathbb{B}_\ell(n)$, where $\ell = m - 1$ provides the limit on the longest run of 0s that is allowed, and the corresponding sublist of the binary reflected Gray code by $\text{BRGC}_\ell(n)$.

► **Example 7.** The sublist of $\text{BRGC}(3) = 000, 100, 110, 010, 011, 111, 101, 001$ with limit $\ell = 1$ is a flip Gray code $\text{BRGC}_1(3) = \bar{0}\bar{0}\bar{0}, \bar{1}\bar{0}\bar{0}, 110, 010, 011, 111, 101, \bar{0}\bar{0}\bar{1} = \bar{1}10, 01\bar{0}, \bar{0}11, 1\bar{1}1, 101$.

Alternatively, $\text{BRGC}_\ell(n)$ can be defined directly by recursion with base case $\text{BRGC}_0(0) = \epsilon$

$$\begin{cases} 0^n, \text{BRGC}_\ell(n-\ell-1)^R \cdot 10^\ell, \text{BRGC}_\ell(n-\ell)^R \cdot 10^{\ell-1}, \dots, \text{BRGC}_\ell(n-1)^R \cdot 1 & \text{if } \ell = n \quad (3a) \\ \text{BRGC}_\ell(n-\ell-1)^R \cdot 10^\ell, \text{BRGC}_\ell(n-\ell)^R \cdot 10^{\ell-1}, \dots, \text{BRGC}_\ell(n-1)^R \cdot 1 & \text{if } \ell < n. \quad (3b) \end{cases}$$

Note that the strings are ordered by their rightmost 1 with the special case 0^n when $\ell = n$.

► **Example 8.** $\text{BRGC}_2(4) = \text{BRGC}_2(1)^R \cdot 10^2, \text{BRGC}_2(2)^R \cdot 10^1, \text{BRGC}_2(3)^R \cdot 1$ and the sublists produce the order $\bar{1}100, 01\bar{1}0, \bar{0}110, 1\bar{1}10, \bar{1}010, 001\bar{0}, \bar{0}011, 1\bar{0}11, \bar{1}111, 01\bar{1}1, \bar{0}101, 1\bar{1}01, 1001$.

These orders were generated in amortized $O(1)$ -time by Arndt [1]. He recursively generated the change sequence $\text{ruler}_\ell(n)$, which we refer to as the *ruler sequence with limit* ℓ , as follows

$$\begin{cases} 1, \text{ruler}_\ell(n-\ell)^R, n-\ell+2, \text{ruler}_\ell(n-\ell+1)^R, n-\ell+3, \dots, n, \text{ruler}_\ell(n-1)^R & \text{if } \ell = n \quad (4a) \\ \text{ruler}_\ell(n-\ell)^R, n-\ell+2, \text{ruler}_\ell(n-\ell+1)^R, n-\ell+3, \dots, n, \text{ruler}_\ell(n-1)^R & \text{if } \ell < n. \quad (4b) \end{cases}$$

with base case $\text{ruler}_0(0) = \emptyset$. These sequences are not always palindromes, so the reflections are relevant. Note that the 1 in (4a) gives the special transition $\bar{0}0^{n-1} = 10^{n-1}$ in (3a).

► **Example 9.** $\text{ruler}_2(4) = \text{ruler}_2(1)^R, 3, \text{ruler}_2(2)^R, 4, \text{ruler}_2(3)^R$ is the change sequence of $\text{BRGC}_2(4)$ in Example 8. The first two sublists are original ruler sequences (which are palindromes) so the sequence is $\text{ruler}(1), 3, \text{ruler}(2), 4, \text{ruler}_2(3)^R = 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1, 2$.

Additional sublist Gray codes of the BRGC are known to exist [47, 48] as are sublist Gray codes of other orders including plain changes [44] and cool-lex order [42].

4.3 Simultaneous Gray Codes

The previous section explained that one of our M&M objects has a Gray code.

► **Theorem 10** ([1, 6]). *There is a flip Gray code for the binary strings that avoid $0^{\ell+1}$. In fact, the limited binary reflected Gray code $\text{BRGC}_\ell(n)$ is a sublist Gray code of the binary reflected Gray code $\text{BRGC}(n)$ and its change sequence is the limited ruler sequence $\text{ruler}_\ell(n)$.*

What does Theorem 10 tell us about Gray codes for other M&M objects? Given two sets of objects \mathcal{O}_1 and \mathcal{O}_2 and a bijection between them $f : \mathcal{O}_1 \rightarrow \mathcal{O}_2$, a *simultaneous Gray code* is an order that is a Gray code for both \mathcal{O}_1 and \mathcal{O}_2 . In other words, if you take the Gray code for \mathcal{O}_1 and apply f to each object, then a Gray code for \mathcal{O}_2 is obtained.

Simultaneous Gray codes often exist when the bijection between objects is very simple. For example, the binary reflected Gray code is a simultaneous Gray code for binary strings, compositions, and subsets [37] and these results generalize nicely based on the limit ℓ . On the other hand, simultaneous Gray codes are more difficult to attain when the bijections are non-trivial. For example, there are hundreds of objects counted by the Catalan objects [51], yet simultaneous Gray codes for them were quite rare [43, 17] prior to recent developments [25, 22] (see [39] for further discussion of this point).

This brings us to M&M permutations. It turns out that the binary reflected Gray code $\text{BRGC}(n)$ provides a swap Gray code for $\mathbb{M}(n)$. Since the swap operation is used in plain changes, and since “plain” M&Ms are also known as “regular” M&Ms, we’ll refer to this Gray code as *regular order* for M&Ms and denote it $\text{Reg}(n)$. Arndt observed that swap Gray codes are obtained in the same way for any limit ℓ as stated in the following theorem.

► **Theorem 11** ([1]). *There is a swap Gray code for the M&M permutations with limit ℓ . In fact, the limited regular order $\text{Reg}_\ell(n)$ is simultaneous with the limited binary reflected Gray code $\text{BRGC}_\ell(n)$. Furthermore, the change sequence $\text{ruler}_\ell(n)$ is both the sequence of swapped smaller indices or values.*

► **Example 12.** $\text{BRGC}_2(4)$ has change sequence $\text{ruler}_2(4) = 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1$. Therefore, we can swap the corresponding indices to obtain the limited regular order as $\text{Reg}_2(4) = \overline{1}2534, 2\overline{1}5\overline{3}4, \overline{2}1\overline{3}54, 1\overline{2}3\overline{5}4, \overline{1}3\overline{2}54, 3\overline{1}245, \overline{1}3\overline{2}45, \overline{1}2\overline{3}45, 2\overline{1}345, \overline{2}1\overline{4}35, \overline{1}2\overline{4}35, 14\overline{2}35, 14\overline{2}35$; alternatively, the sequence specifies the smaller value that swaps over a larger value $\text{Reg}_2(4) = \overline{1}2\overline{5}34, 2\overline{1}5\overline{3}4, \overline{2}1\overline{3}54, \overline{1}2\overline{3}54, \overline{1}3\overline{2}54, 3\overline{1}245, \overline{1}3\overline{2}45, \overline{1}2\overline{3}45, 2\overline{1}345, \overline{2}1\overline{4}35, \overline{1}2\overline{4}35, 14\overline{2}35$.

A simple consequence of Theorem 11 is that our rearranged M&M pyramids also have a simple and delicious Gray code. Each swap in an M&M permutation corresponds to swapping two adjacent rows in the corresponding rearranged pyramid, or replacing the top row (consisting of one M&M) with the missing colour. Note that these changes can be made by changing the colour of a single M&M. In other words, successive solutions to an M&M Problem (with limit ℓ) differ by the colour of a single eaten M&M!

The next two sections extend Theorems 10–11. Section 5 shows that the Gray codes have simple greedy descriptions, and Section 6 outlines loopless algorithms for generating them.

5 Greedy Algorithms

In the half-century following Gray’s patent, recursion was used in various ways to create new Gray codes, as reflected in Savage’s survey from the late-1990s [45]. In the last decade, it has become more common to view the binary reflected Gray code as the result of a simple greedy algorithm. The algorithm starts the order with the string 0^n , then it repeatedly creates a new next string by flipping the leftmost bit in the most recent string. Or less verbosely: flip the leftmost bit. This process continues until no flips are possible (i.e., flipping any bit in the most recent string recreates a previous string).

► **Example 13.** The binary reflected Gray code $\text{BRGC}(3)$ begins 000, 100, 110, 010. Now consider the next step using the greedy algorithm. It first tries to flip the leftmost bit of the most recent string 010, however, this fails as $\overline{0}10 = 110$ is already in the list. Similarly, $0\overline{1}0 = 000$ is already in the list. But its third priority succeeds as $01\overline{0} = 011$ is not in the list. So 011 is added to the end of the list, and the algorithm continues by operating on it.

Many previous Gray code constructions can be re-expressed in terms of greedy algorithms. For example, plain changes and Klügel’s order from Section 4 are generated from $12 \cdots n$ by swapping the smallest value and reversing the shortest prefix, respectively [59].

The greedy perspective has led to a surge of new results, including fun results on pancake flipping [19] and matroids [34]. It has also been used to generate various subsets of binary strings [57, 27] and solve multiple conjectures [33, 33]. However, the biggest success of the approach is the *Combinatorial Generation via Permutation Languages* series of papers⁴. The series was initially announced in an extended abstract and poster at *Permutations Patterns 2019* [23] before appearing as conventional conference and journal papers [24, 25]. Since then it has grown to its seventh entry [9, 10] with a generalization involving multiset permutations (or *s*-words) recently announced [12]. The greedy approach is featured in Mütze’s updated survey of Gray codes [36] which is regularly updated on arXiv [35] and also appears as a dynamic survey in the *Electronic Journal of Combinatorics* [37].

5.1 New Results

Pleasantly, the Gray codes discussed in Section 4 also have simple greedy algorithms. In fact, the most difficult part of the algorithms is describing the first object. We know from their sublist constructions that the Gray codes should start with the first string in the order induced by the binary reflected Gray code, but what are those strings?

Before providing formulae for these strings, it is helpful to first consider the mere existence of swap Gray codes for M&M permutations $\mathbb{M}(n)$. Consider the permutation $n12 \cdots n-1 \in \mathbb{M}(n)$. Note that every value is already one position to the right, except for n . Therefore, only one swap is valid, namely the swap of the first two positions. Similarly, there is only one valid swap for $n-112 \cdots n-2n \in \mathbb{M}(n)$. Therefore, any swap Gray code for $\mathbb{M}(n)$ must start and end at these two permutations. This illustrates how vitally important the choice of the initial object is in this context.

Given a Gray code we let **first** and **last** denote the first and last objects in the order, respectively. (The values of **last** are not needed to describe our greedy algorithms, but they are helpful in understanding why they work.)

Now consider the first string in the limited binary reflected Gray code $\alpha = \text{first}(\text{BRGC}_\ell(n))$. The order begins with strings with suffix 0, so α has suffix 0 so long as $\ell > 0$. Similarly, the suborder of strings with suffix 0 also starts with suffix 0, so α has suffix 00 if $\ell > 1$. Continuing this argument we see that α has suffix 0^ℓ but not suffix $0^{\ell+1}$ as that would be invalid. In other words, it has suffix 10^ℓ . The value 1 indicates a list reflection, so the corresponding suborder begins with suffix 1 and ends with suffix 0. Thus, α has suffix 110^ℓ . The second copy of 1 indicates a second list reflection, so we return to our previous arguments. This results in α having the form $\cdots 110^\ell 110^\ell$ where the start of the string depends on n and ℓ . The last string in the order can be established with a similar argument.

► **Lemma 14.** *The first string in the limited binary reflected Gray code $\text{first}(\text{BRGC}_\ell(n))$ is the unique length n string of the form $\cdots 110^\ell 110^\ell$.*

► **Lemma 15.** *The last string in the limited binary reflected Gray code $\text{last}(\text{BRGC}_\ell(n))$ is the unique length n string of the form $\cdots 110^\ell 110^\ell 1$ with the caveat that its first bit must be 1.*

⁴ The smallest “zig-zag language” is another set of permutations enumerated by 2^{n-1} . The *peakless permutations* avoid the patterns 231 and 132 [7] and the connection between them and the binary reflected Gray code motivated us to consider the Gray codes in this paper.

► **Example 16.** The first and last strings in $\text{BRGC}_2(4)$ are based on the repeating pattern $110^\ell = 1100$. Hence, $\text{first}(\text{BRGC}_2(4)) = 1100$ and $\text{last}(\text{BRGC}_2(4)) = 1001$.

The first and last strings in the limited regular order are simply the M&M permutations that correspond to the above binary strings.

► **Theorem 17.** *The limited binary reflected Gray code $\text{BRGC}_\ell(n)$ of binary strings avoiding $0^{\ell+1}$ is generated by repeatedly flipping the leftmost bit that creates a new string starting from the string in Lemma 14. Similarly, the limited regular order of M&M permutations $\mathbb{M}_\ell(n)$ is generated by swapping the leftmost pair of adjacent values (or the smallest value) starting from the corresponding permutation.*

Note that the instruction to “swap the smallest value” could be ambiguous. For example, given 54123 we wouldn’t know whether to first consider $5\overleftarrow{4}123 = 51423$ or $541\overrightarrow{2}3 = 54213$. However, this distinction never arises as at most one of the swaps is valid and creates a new permutation. This phenomenon is common in greedy Gray code algorithms. In particular, it is present in the greedy description of plain changes and many of its generalizations [24].

► **Example 18.** We know $\text{first}(\text{BRGC}_2(4)) = 1100$, so $\text{first}(\text{Reg}_2(5)) = 12534$. Indeed $\text{Reg}_2(5)$ begins 12534, 21534, 21354, 12354, 13254, 31254. Now consider the next step of the greedy algorithm. It first tries to swap the leftmost pair of indices, but $3\overleftarrow{1}254 = 13254$ is already in the list. Then it tries to swap the next pair of indices $3\overleftarrow{1}254 = 32154$ but this is not a valid swap as 1 is more than one position to the right of its position in the identity. Similarly, $31\overrightarrow{2}54 = 31524$ is invalid as 2 is too far to the right. Finally, $312\overrightarrow{5}4 = 31245$ is successful. Alternatively, the next step swaps the smallest possible value with failures for value 1 ($3\overleftarrow{1}254 = 32154$ and $\overleftarrow{3}1254 = 13254$) then value 2 ($3\overleftarrow{1}254 = 32154$ and $31\overrightarrow{2}54 = 31524$) then value 3 ($\overrightarrow{3}1254 = 13254$) before success in exactly one direction for value 4 ($312\overrightarrow{5}4 = 31245$).

6 Efficient Algorithms

While the algorithms in Section 5 are simple, they are also inefficient. In particular, remembering previously generated objects uses exponential space relative to n even when $\ell = 1$ (as the Fibonacci sequence grows like $O(\phi^n)$ for the golden ratio $\phi \approx 1.62$).

In this section we develop a *successor rule* for the binary strings avoiding $0^{\ell+1}$. In other words, we show how to directly map any such binary string into the next binary string in the Gray code without any additional information. By carefully maintaining $O(n)$ additional space, we are able to repeatedly apply the successor rule in constant time. In other words, we create *loopless algorithm* which generates successive objects in worst-case $O(1)$ -time. Loopless algorithms were pioneered by Ehrlich [18] and are best possible in terms of time-complexity.

To better contextualize these results we first differentiate between enumeration algorithms and generation algorithms.

6.1 Generation Algorithms vs Enumeration Algorithms

The goal of an *enumeration algorithm* is to create every object. They are especially useful for complicated objects when the goal is to store them in a file for subsequent analysis (e.g., see [11] and [13]). Alternatively, we may want to count (i.e., enumerate!) the objects. In these situations it makes sense to measure the time-complexity in terms of the *total time* or *average time* (i.e., total time divided by the number of objects). Regardless of the application, these algorithms are often run from the command-line with output piped to a file (e.g., `./enumerate n > n.txt.`)

In other situations we want to start iterating over the objects immediately, and we won't necessarily complete the process. For example, if we are searching for an object that solves a problem, then we may wish to stop once one is found. In these situations it makes sense to measure how much time is taken to generate the first k objects for all possible values of k . This leads to the notion of a *generation algorithm* and *amortized time* or *worst-case* analyses. Regardless of the exact application, these algorithms are often used as iterators within modern programming languages that support them (e.g., `for x in generate(n)`).

When an application doesn't store or print the objects it can be possible to process each one in $o(n)$ -time (i.e., less than $O(n)$ -time). In the *shared object model* the generation algorithm and application share one object; the generation algorithm modifies this object and notifies the application that the next possibility is ready. To fully take advantage of this model, the generation algorithm must make small changes either in an amortized or worst-case sense. It must also avoid most common types of recursion and it should communicate each change so that the application can avoid scanning the shared object. For example, imagine solving a Knapsack Problem by generating the binary reflected Gray code; each bitstring is an incidence vector of items and each bit-flip changes the associated value of the selected items by adding or subtracting one value. In fact, it is possible to generate and evaluate each feasible solution in amortized $O(1)$ -time [46].

Note that the nomenclature advocated here by the authors is new and non-standard. For example, Ruskey's unpublished but widely used and often cited *Combinatorial Generation* [41] textbook primarily focuses on enumeration algorithms⁵.

6.2 Successor Rule

The successor rule for binary strings in $\text{BRGC}(n)$ without any limitation is well-known [31]. It flips the first bit if the weight of the string (i.e., its number of 1s) is even, and otherwise it flips the bit to the right of the leftmost 1. This is presented below.

► **Definition 19.** Let $b = b_1b_2 \cdots b_n \in \mathbb{B}(n)$ with weight $w = b_1 + b_2 + \cdots + b_n$. Its successor $\text{next}(b) \in \mathbb{B}(n)$ is obtained by flipping the bit at index $\text{flip}(b)$ which defined as

$$\text{flip}(b) = \begin{cases} 1 & \text{if } w \text{ is even} \\ h + 1 & \text{otherwise} \end{cases} \quad (5a)$$

$$(5b)$$

where h is the smallest index with $b_h = 1$.

Note that $\text{flip}(b)$ and $\text{next}(b)$ are undefined precisely when $b = 10^{n-1}$. This is expected and desired behaviour as 10^{n-1} is the last string in $\text{BRGC}(n)$, so it has no successor.

► **Example 20.** Consider $b = b_1b_2b_3b_4b_5b_6 = 001101 \in \mathbb{B}(6)$. Its weight $w = 3$ is odd, so (5a) does not hold and the leftmost bit is not flipped. Instead its leftmost 1 is located at index $h = 3$, so $\text{flip}(b) = h + 1 = 4$ and its successor is $\text{next}(b) = 001\bar{1}01 = 001001$ by (5b).

Now we derive a successor rule for the binary strings avoiding $0^{\ell+1}$ in $\text{BRGC}_\ell(n)$ order. While the limited Gray code $\text{BRGC}_\ell(n)$ has been discussed elsewhere [1, 6] to our knowledge the successor rule has not previously been determined.

The *run-length encoding* of binary string $b \in \mathbb{B}_\ell(n)$ is

$$b_1b_2 \cdots b_n = 1^{f_1} 0^{g_1} 1^{f_2} 0^{g_2} \cdots 1^{f_k} 0^{g_k} \quad (6)$$

⁵ Its preface offers a more accurate (but less catchy) title: *Exhaustive Listing of Combinatorial Objects!*

where each f_i and g_i are positive except possibly for f_1 (i.e., the first run of 1s) and g_k (i.e., the last run of 0s). This partitions the binary string into *blocks* which are maximal substrings of the form 1^*0^* (i.e., some number of 1s followed by some number of 0s). We let $h_j = \sum_{i=1}^{j-1} f_i + g_i$ be the starting index of the i th block.

► **Example 21.** Let $b = b_1b_2b_3b_4b_5b_6b_7b_8b_9 = 001101000$. Its blocks are 00, 110, and 1000.

- The run-lengths of the blocks are $f_1 = 0$, $g_1 = 1$, $f_2 = 2$, $g_2 = 1$, and $f_3 = 1$, $g_3 = 3$.
- The starting indices of the blocks are $h_1 = 1$, $h_2 = 3$, and $h_3 = 6$.

Now we express the successor rule where $\text{flip}_\ell(b)$ provides the index to flip. To make sense of the definition, note that the successor either flips the first bit (i.e., $\text{flip}_\ell(b) = 1$), the second bit in a block (i.e., $\text{flip}_\ell(b) = h_j + 1$), or the last bit (i.e., $\text{flip}_\ell(b) = n$). Also note that if a block equals 110^ℓ (i.e., $f_i = 2$ and $g_i = \ell$), then we cannot flip its second bit as it would create $0^{\ell+1}$. The final consideration is that we don't want to flip the bit that creates the predecessor (i.e., the previous string) and this leads to condition (b) in the definition of j .

► **Definition 22.** Let $b = b_1b_2 \cdots b_n \in \mathbb{B}_\ell(n)$ with run-length encoding $1^{f_1}0^{g_1}1^{f_2}0^{g_2} \cdots 1^{f_k}0^{g_k}$ and weight $w = b_1 + b_2 + \cdots + b_n$. Its successor $\text{next}_\ell(b)$ flips index $\text{flip}_\ell(b)$ defined as

$$\text{flip}_\ell(b) = \begin{cases} 1 & \text{if } w \text{ is even and } (g_1, h_1) \neq (1, \ell) \\ h_j + 1 & \text{otherwise, if } j \text{ exists} \end{cases} \quad (7a)$$

$$(7b)$$

where j is the smallest index of a block satisfying (a) $(f_j, g_j) \notin \{(2, \ell), (1, 0)\}$ and (b) $j \neq 1$ if w is even or $f_1 = 0$. If neither case holds, then $\text{flip}_\ell(b)$ and $\text{next}_\ell(b)$ are undefined.

Note that Definition 22 generalizes Definition 19. More specifically, $\text{flip}_{n+1}(b) = \text{flip}(b)$. This is because (7a) always holds when w is even (i.e., if $b_1 = 1$ then $g_1 < n$). Similarly, the indices specified in (5b) and (7b) are the same as $j = 1$ whenever w is odd.

Also note that Definition 22 is undefined precisely when b is the string in Lemma 15.

► **Example 23.** Consider $b = 1100110010$ within $\text{BRGC}_2(10)$. Its weight $w = 5$ is odd, so (7a) does not apply. The value of j is 3 because the first two blocks have the form $110^\ell = 1100$ while the third block is 10. So we apply (5b) by flipping the bit at index $\text{flip}_2(b) = h_3 + 1 = 10$ which produces $\text{next}_2(b) = 11001100\bar{1}0 = 1100110011$.

6.3 Loopless Implementations

To create a loopless algorithm for $\text{BRGC}_\ell(n)$ we need to be able to apply the successor rule in $O(1)$ -time in the worst-case. In other words, we need to store additional information that allows us to determine $\text{next}_\ell(b)$ in $O(1)$ -time, and we need to be able to update any such information in $O(1)$ -time.

Our additional information consists of the weight w of the current string and two sequences.

- The first sequence keeps track of where the runs of 1s begin in the current string.
- The second sequence allows us to skip over each block of the form 110^ℓ once. These skips are needed to avoid creating illegal substrings $1\bar{1}0^\ell = 10^m$ via the successor rule and several such blocks may be simultaneously skipped over at the same time.

Given this additional information we can apply the successor rule in $O(1)$ -time. Similarly, we can update the additional information in constant-time. More specifically, we use the locations of consecutive runs of 1s and the values of surrounding bits to determine when a block of the form $1\bar{1}0^{\{\ell\}}$ has been created. We also store the skipping information as offsets that allows us to clear the need to skip an arbitrary number of consecutive blocks in constant-time. This leads to the following theorem.

Suppose that the value of d_{n-1} has been chosen, and now consider the choices for d_{n-2} . Note that $d_{n-2} \in \{c_{n-r-1}, c_{n-r}, \dots, c_n\} - \{d_{n-1}\}$. This is because the bottom $r + 1$ rows of the initial pyramid have width at least $n - 2$ and the color d_{n-1} has already been used. Also note that c_{n-r-1} is not a valid colour when $r = n$ or $r = n - 1$. So the number of choices for d_{n-2} is $r + 1$ if $0 \leq r \leq n - 2$ or r if $r \geq n - 1$.

In the same way, the number of choices for d_{n-3} is $r + 1$ if $0 \leq r \leq n - 3$, or r if $r = n - 2$, or $r - 1$ if $r \geq n - 1$. As a result, the total number of ways of choosing $d_{n-1}, d_{n-2}, \dots, d_1$ is exactly $\langle n, r \rangle!$. ◀

Interestingly, the proof of Theorem 27 gives us another way to prove our original theorem. In the standard M&M Problem there are two choices for the M&M colour that will become row $n - 1$ at the bottom of the new pyramid. More specifically, the choices are colour n or colour $n - 1$ as these are the colours with frequency at least $n - 1$ in the initial pyramid. After this choice is made there are two choices for the colour of row $n - 2$ in the new pyramid. More specifically, the choices are colour $n - 2$ together with colour n or colour $n - 1$ (depending on which is not used for the bottom row of the new pyramid). This continues until we reach the result of 2^{n-1} . To visualize this argument note that exactly half of the eight new pyramids in Figure 1b have a bottom row of green while the others have a bottom row of yellow, then for each of those four pyramids there are two different colours on the next row, and so on.

8 One-Way Bounded Permutations

Given the results of the previous sections, it becomes natural to ask if there are other classes of permutations counted by 2^{n-1} or more generally the flatorial numbers. In this section we use a tree-based argument to prove that eight distinct classes have this property [26]. These classes are *one-way bounded* permutations on $[n]$ for a fixed parameter $0 \leq r \leq n$.

Let S_n be the set of all permutations of $[n] = \{1, 2, \dots, n\}$ in one-line notation. Consider the following two subsets of S_n for $n = 4$ and a fixed parameter $r = 1$ explained below.

$$\text{rgt}(4, 1) = \{\underline{1}234, \underline{2}134, \underline{23}14, \underline{32}14, \underline{234}1, \underline{243}1, \underline{324}1, \underline{423}1\} \tag{8}$$

$$\text{inv}(4, 1) = \{1\underline{2}34, 12\underline{4}3, 1\underline{3}24, 1\underline{34}2, \underline{2}134, \underline{21}43, \underline{23}14, \underline{234}1\} \tag{9}$$

$$0000 \quad 0001 \quad 0010 \quad 0011 \quad 0100 \quad 0101 \quad 0110 \quad 0111$$

The first set contains permutations $p_1 p_2 \dots p_n$ in which $p_i^{-1} > i$ implies $i \leq r$. In other words, if a value i appears further to the right than it does in the identity, then $i \leq r$. Since $r = 1$ in (8), only value 1 (underlined) can appear to the right of its position in id. The second set contains permutations whose inversion vector⁶ (shown below) $v_1 v_2 \dots v_n$ has $v_i \leq r$ for all i . In other words, every value is inverted with at most r smaller values. Since $r = 1$ in (9), each value is inverted with at most one smaller value.

The two sets have the same cardinality; we will prove that this is true for all n and r . Moreover, this is true for eight classes of *one-way bounded permutations* that are listed in Definition 29. None of our equivalences are obtained by reversing indices $p_n \dots p_2 p_1$ and/or inverting values $(n - p_1 + 1)(n - p_2 + 1) \dots (n - p_n + 1)$. These trivial modifications known as the *symmetries of the square* extend our results to $8 \cdot 4 = 32$ equinumerous types. For example, the “low rights” in (8) become “high lefts” (i.e., $p_i^{-1} < i$ implies $i \geq n - r + 1$) by reversing and inverting. Our results are summarized in Table 2. While each type is fairly

⁶ The *inversion vector* counts the smaller inverted values. That is, $v_i = |\{j \mid j < i \text{ and } p_j^{-1} > p_i^{-1}\}|$.

elementary, we were unable to find many references in the literature. Table 3 shows that only three of the types are listed in corresponding OEIS entries. Short descents with $r = 2$ arise using classic pattern avoidance [7]: $\text{dsc}(n, 2) = \text{Av}_n(312, 231)$. Some non-classical avoidance results also appear in the OEIS sequences [8, 20].

■ **Table 2** Eight types of one-way bounded permutations (see Definition 29). Each is counted by the flatorial $\langle n, r \rangle!$ via a family subtree with large ($\ell - r$ to ℓ) or small (ℓ and 1 to r) branch labels. The sample permutations satisfy their bound for $r = 4$ (or larger) but not $r = 3$. For example, $p = 61584327 \in \text{dsc}(8, 4) \setminus \text{dsc}(8, 3)$ as the smaller value in each descent is ≤ 4 but not ≤ 3 .

low descents	short descents	low rights	short rights	large inverts	small inverts	early lefts	short lefts
insert value	insert value	swap value	swap value	insert index	insert index	swap index	swap index
small labels	large labels	small labels	large labels	small labels	large labels	small labels	large labels
61584327	18564732	78516243	53814276	1783642	13584726	75483216	15768243
$\text{dsc}(n, r)$	$\text{dsc}(n, r)$	$\text{rgt}(n, r)$	$\overline{\text{rgt}}(n, r)$	$\text{inv}(n, r)$	$\overline{\text{inv}}(n, r)$	$\text{flt}(n, r)$	$\overline{\text{flt}}(n, r)$

■ **Table 3** Flatorial numbers for small r . Three of our permutation types appear in the OEIS entries. †Several sequences match the initial terms and have another constraint (e.g., A179357).

r	Formula ($n \geq r$)	$n = 1, 2, 3, \dots$	OEIS	short rights	short ascent	short lefts
1	$1! \cdot 2^{n-1}$	1, 2, 4, 8, 16, 32, 64, ...	A000079	Arndt (2009)		
2	$2! \cdot 3^{n-2}$	1, 2, 6, 18, 54, 162, 486, ...	A025192		Lewis (2006)	
3	$3! \cdot 4^{n-3}$	1, 2, 6, 24, 96, 384, 1536, ...	A084509		Knuth (2022)	
4	$4! \cdot 5^{n-4}$	1, 2, 6, 24, 120, 600, 3000, ...	A179364†			Hardin (2010)
5	$5! \cdot 6^{n-5}$	1, 2, 6, 24, 120, 720, 4320, ...	A179365†			Hardin (2010)

8.1 Family Trees of Permutations & One-Way Bounded Permutations

We consider four *family trees* of permutations. Each of these trees generate two of the classes of one-way bounded permutations given in Definition 29; the classes of permutations that these trees generate is shown in Table 2. Each tree has root 1 and S_ℓ appears at level ℓ . At level ℓ , the branches are labeled with values $b \in [\ell]$. The branch ℓ always inserts ℓ as the rightmost symbol, and identity permutations are on rightmost paths. The remaining children are obtained as follows, where $b \in [\ell - 1]$ is the branch label.

- *Insert value tree*: insert ℓ to the left of value b .
- *Swap value tree*: insert ℓ as the rightmost symbol, then swap it with the value b .
- *Insert index tree*: insert ℓ to the left of the index b .
- *Swap index tree*: insert ℓ as the rightmost symbol, then swap it with index b .

For each family tree we consider two subtrees parameterized by a given value of $r \geq 0$.

- The *large label subtree* includes branches with labels $\ell - r$ to ℓ at level ℓ .
- The *small label subtree* includes branches with labels ℓ and 1 to r at level ℓ .

So at level ℓ the subtrees include branches with label ℓ and (at most) the r smallest or largest remaining labels. Thus, the number of nodes at each successive level increases from 1 up to $r+1$ giving Remark 28. Family trees and subtrees appear in Figures 7–8.

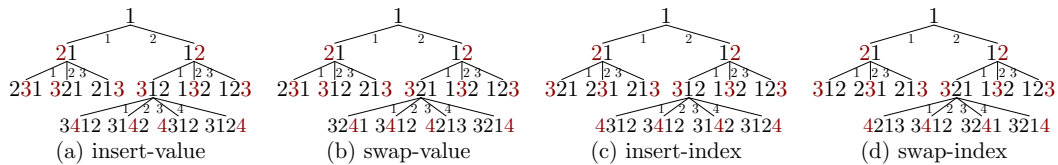
► **Remark 28.** A family subtree with large (or small) labels has $\langle \ell, r \rangle!$ nodes at level ℓ .

► **Definition 29.** Let $p = p_1 p_2 \dots p_n$ be a permutation, $p^{-1} = p_1^{-1} p_2^{-1} \dots p_n^{-1}$ its inverse, and $v = v_1 v_2 \dots v_n$ its inversion vector. Value $i \in [n]$ is an invert if it is inverted with some smaller value (i.e., $v_i > 0$). For example, $p = 2413$ has $p^{-1} = 3142$, $v = 0102$, and inverts 2 and 4. We define the following one-way bounded permutations for a fixed $r \geq 1$.

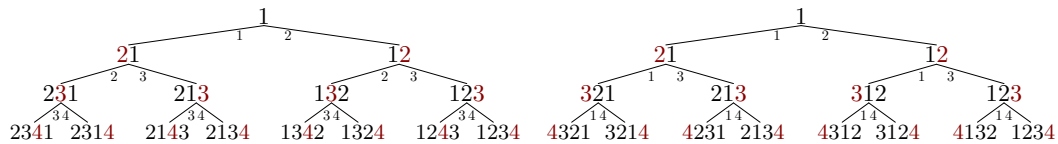
- (a) p has low descents, denoted $p \in \underline{\text{dsc}}(n, r)$, if $p_{i+1} < p_i$ implies $p_{i+1} \leq r$.
That is, the smaller value in a descent is at most r . For example, in 61584327 each smaller value in a descent has $r \leq 4$.
- (b) p has short descents, denoted $p \in \overline{\text{dsc}}(n, r)$, if $p_i - p_{i+1} \leq r$.
That is, each descent has height at most r . For example, in 12734685 , each descent has height $r \leq 4$.
- (c) p has low rights, denoted $p \in \underline{\text{rgt}}(n, r)$, if $p_i^{-1} > i$ implies $i \leq r$.
That is, only small values can move to the right. For example, in 53861472 , only values $r \leq 4$ have moved to the right.
- (d) p has short rights, denoted $p \in \overline{\text{rgt}}(n, r)$, if $p_i^{-1} - i \leq r$, or equivalently, $p_i \geq i - r$.
That is, values can move at most r spaces to the right. In 53841276 , values have moved at most 4 spaces to the right.
- (e) p has large inverts, denoted $p \in \underline{\text{inv}}(n, r)$, if $v_i > 0$ implies $v_i \geq i - r$.
That is, any invert is inverted with at least $i - r$ smaller values. In 75483216 , any invert i is inverted with at least $i - 4$ smaller values.
- (f) p has small inverts, denoted $p \in \overline{\text{inv}}(n, r)$, if $v_i \leq r$ for all i .
That is, each value is inverted with at most r smaller values. For example, in 15724368 , each value is inverted with at most 4 smaller values.
- (g) p has early lefts, denoted $p \in \underline{\text{ft}}(n, r)$, if $p_i^{-1} < i$ implies $p_i^{-1} \leq r$.
That is, if a value is moved to the left, it must be in the first r positions of p . In 75384216 , values that moved left are in the first $r = 4$ positions.
- (h) p has short lefts, denoted $p \in \overline{\text{ft}}(n, r)$, if $p_i^{-1} \geq i - r$, or equivalently, $p_i \leq i + r$.
That is, values can move at most r spaces to the left. For example, in 1682473 , values have moved at most 4 spaces to the left.

► **Theorem 30.** For all $n \geq 1$ and $r \geq 1$, each of the sets of one-way bounded permutations in Definition 29 are counted by flatorial numbers, $\langle n, r \rangle!$.

Proof. By Remark 28, for each set of one-way bounded permutations, we need only show that the permutations in the set are exactly the nodes in the associated subtree with large labels or small labels. We illustrate this for low rights using the swap value tree with small labels; the other proofs are similar. A *right* is a value v with $p_v^{-1} > v$.



■ **Figure 7** Four family trees up to level $\ell = 3$ with one node at level $\ell = 4$. A branch with label $b < \ell$ inserts ℓ (a) before value b or (c) at index b , or inserts ℓ in the last position and then swaps it with (b) value b or (d) index b . A branch with label $b = \ell$ inserts ℓ into the last position.



■ **Figure 8** The insert index family subtree with large (left) and small (right) labels for $r = 1$. Note that the ℓ branches are included in both subtrees.

23:20 Pyramid Schemes for Eating M&Ms: Enumeration, Generation, and Gray Codes

Consider the swap value tree, in which each child of $p_1 p_2 \dots p_{\ell-1}$ at branch b is obtained by appending ℓ , and then swapping ℓ with b for $b < \ell$. Each branch $b < \ell$ creates a right: b is now in position $\ell > b$. Suppose $p = p_1 p_2 \dots p_{\ell-1} \in \text{rgt}(\ell - 1, r)$. The children of p in $\text{rgt}(\ell - 1, r)$ are exactly those with branches $b \leq r$ (in which rights with values at most r are created), and branch ℓ (in which no right is created). On the other hand, if v is a right in permutation p , then v will be a right in all descendants of p , since v either remains in its same position for branches $b \neq v$, or moves further to the right at any branch $b = v$. Therefore, if $p \notin \text{rgt}(n, k)$, then none of its descendants can be. ◀

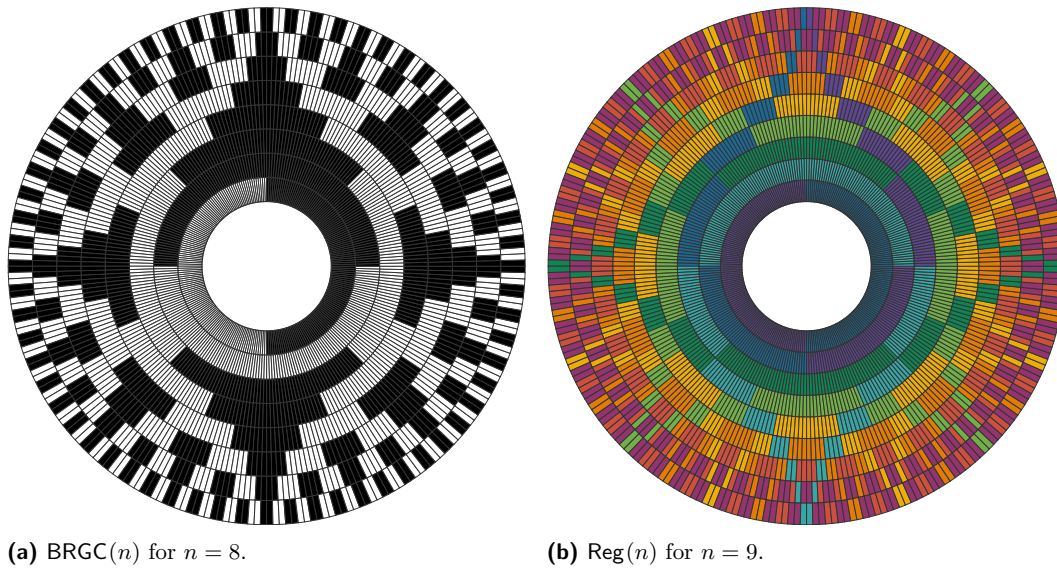
Table 4 illustrates one-way bounded permutations of types (a)–(d) from Definition 29.

■ **Table 4** One-way bounded permutations for $n = 4$ with $r = 2$ (left) and $n = 5$ with $r = 1$ (right). Note that there are $\langle 4, 2 \rangle! = 18$ and $\langle 5, 1 \rangle! = 16$ of each type (a)–(d) (as well as for types (e)–(h)).

(a) low descents	(b) short descents	(c) low rights	(d) short rights
1234	12345	1234	12345
1324	21345	1243	12354
1342	23145	1324	12435
1423	23415	1342	12543
2134	23451	1423	13245
2314	23514	1432	13254
2341	24135	2134	14325
2413	24513	2143	15432
3124	25134	2314	21345
3142	31245	2431	21354
3214	34125	3124	21435
3241	34512	3142	21543
3412	35124	3214	32145
3421	41235	3421	32154
4123	45123	4213	43215
4132	51234	4231	54321
4213		4312	
4231		4321	

■ **Table 5** The parameter ℓ denotes the limitation that at most ℓ M&Ms may be eaten (Section 3). The parameter r refers to right-bounded permutations (Section 8). Three r -bounded permutation types appear in the OEIS entries. †Several sequences match the initial terms and have another constraint (e.g., A179357).

ℓ	r	Name	Formula ($n \geq r$)	$n = 1, 2, 3, \dots$	OEIS	reference
1		Mononacci		1, 1, 1, 1, 1, 1, 1, ...	A057427	
2		Fibonacci		1, 1, 2, 3, 5, 8, 13, 21, 34, ...	A000045	
3		Tribonacci		1, 1, 2, 4, 7, 13, 24, 44, 81, ...	A000073	
4		Tetranacci		1, 1, 2, 4, 8, 15, 29, 56, 108, ...	A000078	
5		Pentanacci		1, 1, 2, 4, 8, 16, 31, 61, 120, ...	A001591	
6		Hexanacci		1, 1, 2, 4, 8, 16, 32, 63, 125, ...	A001592	
7		Heptanacci		1, 1, 2, 4, 8, 16, 32, 64, 127, ...	A122189	
1		∞ -nacci	$\langle n, 1 \rangle! = 2^{n-1}$	1, 2, 4, 8, 16, 32, 64, ...	A000079	Arndt (2009): short rights
2			$\langle n, 2 \rangle! = 2! \cdot 3^{n-2}$	1, 2, 6, 18, 54, 162, 486, ...	A025192	Lewis (2006): short ascents
3			$\langle n, 3 \rangle! = 3! \cdot 4^{n-3}$	1, 2, 6, 24, 96, 384, 1536, ...	A084509	Knuth (2022): short ascents
4			$\langle n, 4 \rangle! = 4! \cdot 5^{n-4}$	1, 2, 6, 24, 120, 600, 3000, ...	A179364†	Hardin (2010): short lefts
5			$\langle n, 5 \rangle! = 5! \cdot 6^{n-5}$	1, 2, 6, 24, 120, 720, 4320, ...	A179365†	Hardin (2010): short lefts



■ **Figure 9** Binary reflected Gray code for binary words and regular order for M&M permutations. Individual words are read outside-in with successive words in clockwise order starting at 12 o'clock. For example (a) starts 00000000 then 10000000 while (b) begins 912345678 then 192345678.

9 Final Remarks

9.1 Summary

Our investigation started by trying to count the number of solutions to a fun eating problem. By varying the problem in two ways we obtain a natural progression of counting functions from 1 to Fibonacci to 2^n to $n!$ as shown in Table 5. This progression includes variations of Fibonacci numbers (e.g., Tribonacci) and the newly named flatorial numbers.

We also considered how to efficiently order and generate the solutions to the original and limited problems, including new foundational results on the limited binary reflected Gray code $\text{BRGC}_\ell(n)$. Finally, to ensure that the fun never ends, we articulated eight distinct classes of permutations that are also enumerated by flatorials!

9.2 Future Work

A natural enumeration question arises from simultaneously applying our two parameters ℓ and r . A natural Gray code question is if the solutions to our relaxed problem can be appropriated represented and ordered. A natural algorithmic question is if the new sets of one-way permutations can be efficiently generated.

References

- 1 Jörg Arndt. *Matters Computational: ideas, algorithms, source code*. Springer Science & Business Media, 2010.
- 2 Jörg Arndt. *Permutations with special properties*, pages 277–290. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-14764-7_11.
- 3 Linda J Barth. *New Jersey Originals: Technological Marvels, Odd Inventions, Trailblazing Characters & More*. Arcadia Publishing, 2018.
- 4 Samuel Beckett. *Murphy*. Picador, 1973.

- 5 Samuel Beckett. *The Collected Shorter Plays of Samuel Beckett: All That Fall, Act Without Words, Krapp's Last Tape, Cascando, Eh Joe, Footfall, Rockaby and others*. Grove/Atlantic, Inc., 2010.
- 6 Antonio Bernini, Stefano Bilotta, Renzo Pinzani, Ahmad Sabri, and Vincent Vajnovszki. Gray code orders for q -ary words avoiding a given factor. *Acta Informatica*, 52(7):573–592, 2015. doi:10.1007/S00236-015-0225-2.
- 7 David Bevan. Permutation patterns: basic definitions and notation. *arXiv preprint arXiv:1506.06673*, 2015.
- 8 David Bevan, Derek Levin, Peter Nugent, Jay Pantone, Lara Pudwell, Manda Riehl, and ML Tlachac. Pattern avoidance in forests of binary shrubs. *Discrete Mathematics & Theoretical Computer Science*, 18(Permutation Patterns), 2016.
- 9 Sofia Brenner, Jean Cardinal, Thomas McConville, Arturo Merino, and Torsten Mütze. Combinatorial generation via permutation languages. vii. supersolvable hyperplane arrangements. *arXiv preprint arXiv:2507.14327*, 2025.
- 10 Sofia Brenner, Jean Cardinal, Thomas McConville, Arturo Merino, and Torsten Mütze. Traversing regions of supersolvable hyperplane arrangements and their lattice quotients. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4953–4968. SIAM, 2026.
- 11 Gunnar Brinkmann, Gilles Caporossi, and Pierre Hansen. A constructive enumeration of fusenes and benzenoids. *Journal of Algorithms*, 45(2):155–166, 2002. doi:10.1016/S0196-6774(02)00215-8.
- 12 Samuel Buick, Madeleine Goertz, Amos Lastmann, Kunal Pal, Helen Qian, Sam Tacheny, Aaron Williams, Leah Williams, and Yulin Zhai. Exhaustive generation of pattern-avoiding s -words. In *Proceedings of the 23rd International Conference on Permutation Patterns*, pages 107–110, 2025.
- 13 James Chapman, Judith Foos, Elizabeth J Hartung, Andrew Nelson, and Aaron Williams. Pairwise disagreements of kekulé, clar, and fries numbers for benzenoids: A mathematical and computational investigation. *MATCH Communications in Mathematical and in Computer Chemistry*, 80(1):189–206, 2018.
- 14 Paul Chrystal. *The History of Sweets*. Pen and Sword, 2021.
- 15 Mark Cooke, Chris North, Megan Dewar, and Brett Stevens. A note on Beckett-Gray codes and the relationship of Gray codes to data structures. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 126:195–200, 2025.
- 16 Richard Duckworth. *Tintinnologia, Or, the Art of Ringing: Wherein is Laid Down Plain and Easie Rules for Ringing All Sorts of Plain Changes*. Project Gutenberg, 2010.
- 17 Stephane Durocher, Pak Ching Li, Debajyoti Mondal, Frank Ruskey, and Aaron Williams. Cool-lex order and k -ary Catalan structures. *Journal of Discrete Algorithms*, 16:287–307, 2012. doi:10.1016/J.JDA.2012.04.015.
- 18 Gideon Ehrlich. Loopless algorithms for generating permutations, combinations, and other combinatorial configurations. *Journal of the ACM (JACM)*, 20(3):500–513, 1973. doi:10.1145/321765.321781.
- 19 Harrah Essed and Wei Therese. The harassed waitress problem. In *International Conference on Fun with Algorithms*, pages 325–339. Springer, 2014. doi:10.1007/978-3-319-07890-8_28.
- 20 Alice LL Gao and Sergey Kitaev. On partially ordered patterns of length 4 and 5 in permutations. *The Electronic Journal of Combinatorics*, 26(3), 2019. doi:10.37236/8605.
- 21 Frank Gray. Pulse code communication. *United States Patent Number 2632058*, 1953.
- 22 Petr Gregor, Torsten Mütze, and Namrata. Combinatorial generation via permutation languages. vi. binary trees. *European Journal of Combinatorics*, 122:104020, 2024. doi:10.1016/J.EJC.2024.104020.
- 23 Elizabeth Hartung, Hung P Hoang, Torsten Mütze, and Aaron Williams. Exhaustive generation of pattern-avoiding permutations. *Permutation Patterns 2019*, pages 81–83, 2019.

- 24 Elizabeth Hartung, Hung P Hoang, Torsten Mütze, and Aaron Williams. Combinatorial generation via permutation languages. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1214–1225. SIAM, 2020.
- 25 Elizabeth Hartung, Hung P Hoang, Torsten Mutze, and Aaron Williams. Combinatorial generation via permutation languages. i. fundamentals. *Transactions of the American Mathematical Society*, 375:2255–2291, 2022.
- 26 Liz Hartung, Brett Stevens, and Aaron Williams. One-way bounded permutations and the flatorial numbers. In *Proceedings of the 23rd International Conference on Permutation Patterns*, pages 55–58, 2025.
- 27 Nathanaël Hassler, Vincent Vajnovszki, and Dennis Wong. Efficient generation of some greedy binary Gray codes. *RAIRO-Theoretical Informatics and Applications*, 59:16, 2025. doi:10.1051/ITA/2025014.
- 28 Silvia Heubach and Toufik Mansour. *Combinatorics of compositions and words*. Chapman and Hall/CRC, 2009.
- 29 Carl Friedrich Hindenburg. *Collection of combinatorial-analytical treatises*, volume 2. ben Gerhard Fleischer the Younger, 1796.
- 30 Selmer M Johnson. Generation of permutations by adjacent transposition. *Mathematics of computation*, 17(83):282–285, 1963.
- 31 Donald E Knuth. *Art of Computer Programming, Volume 4, Fascicle 2, The: Generating All Tuples and Permutations*. Addison-Wesley Professional, 2013.
- 32 Donald E Knuth. *Art of Computer Programming, Volume 4, Fascicle 4, The: Generating All Trees—History of Combinatorial Generation*. Addison-Wesley Professional, 2013.
- 33 Bowie Liu, Dennis Wong, Chan-Tong Lam, and Sio-Kei Im. Generating pivot Gray codes for spanning trees of complete graphs in constant amortized time. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 833–870. SIAM, 2026.
- 34 Arturo Merino, Torsten Mutze, and Aaron Williams. All your bases are belong to us: listing all bases of a matroid by greedy exchanges. In *11th International Conference on Fun with Algorithms (FUN 2022)*, volume 226, page 22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
- 35 Torsten Mütze. Combinatorial Gray codes—an updated survey. *arXiv preprint arXiv:2202.01280*, 2022. arXiv:2202.01280.
- 36 Torsten Mutze. Combinatorial Gray codes: an updated survey. *Electronic Journal of Combinatorics*, 30(3), 2023.
- 37 Torsten Mutze. Combinatorial Gray codes: an updated survey. *Electronic Journal of Combinatorics*, DS26, 2024.
- 38 OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences, 2026.
- 39 Vincent Pilaud and Aaron Williams. Skipping ropes: An efficient Gray code algorithm for generating wiggly permutations. In *19th International Symposium on Algorithms and Data Structures (WADS 2025)*, pages 46–1. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025.
- 40 Yuan Qiu and Aaron Williams. Generating signed permutations by twisting two-sided ribbons. In *Latin American Symposium on Theoretical Informatics*, pages 114–129. Springer, 2024. doi:10.1007/978-3-031-55598-5_8.
- 41 Frank Ruskey. Combinatorial generation. *Working version (1j-CSC 425/520)*, 2003.
- 42 Frank Ruskey, Joe Sawada, and Aaron Williams. Binary bubble languages and cool-lex order. *Journal of Combinatorial Theory, Series A*, 119(1):155–169, 2012. doi:10.1016/J.JCTA.2011.07.005.
- 43 Frank Ruskey and Aaron Williams. Generating balanced parentheses and binary trees by prefix shifts. In *Proceedings of the fourteenth symposium on Computing: the Australasian theory*, volume 77, pages 107–115, 2008. URL: <http://crpit.scem.westernsydney.edu.au/abstracts/CRPITV77Ruskey.html>.

- 44 Ahmad Sabri and Vincent Vajnovszki. Restricted steinhaus-johnson-trotter list. In *ICGT*, 2014.
- 45 Carla Savage. A survey of combinatorial Gray codes. *SIAM review*, 39(4):605–629, 1997. doi:10.1137/S0036144595295272.
- 46 Joe Sawada and Aaron Williams. Efficient oracles for generating binary bubble languages. *the electronic journal of combinatorics*, pages P42–P42, 2012.
- 47 Joe Sawada, Aaron Williams, and Dennis Wong. Inside the binary reflected Gray code: Flip-swap languages in 2-Gray code order. In *International Conference on Combinatorics on Words*, pages 172–184. Springer, 2021. doi:10.1007/978-3-030-85088-3_15.
- 48 Joe Sawada, Aaron Williams, and Dennis Wong. Flip-swap languages in binary reflected Gray code order. *Theoretical Computer Science*, 933:138–148, 2022. doi:10.1016/J.TCS.2022.08.024.
- 49 Joe Sawada and Dennis Chi-Him Wong. A fast algorithm to generate Beckett-Gray codes. *Electronic Notes in Discrete Mathematics*, 29:571–577, 2007.
- 50 Matthew B Squire. Gray codes for a -free strings. *The Electronic Journal of Combinatorics*, pages R17–R17, 1996.
- 51 Richard P Stanley. *Catalan numbers*. Cambridge University Press, 2015.
- 52 Fabian Stedman. *Campanalogia: or The art of ringing improved*. Good Press, 2025.
- 53 Hugo Steinhaus. *One hundred problems in elementary mathematics*. Courier Corporation, 1963.
- 54 George Robert Stibitz. Binary counter. *United States Patent Number 2307868*, 1943.
- 55 Stephen M Stigler. Stigler’s law of eponymy. *Transactions of the New York academy of sciences*, 39(1 Series II):147–157, 1980.
- 56 Hale F Trotter. Algorithm 115: perm. *Communications of the ACM*, 5(8):434–435, 1962. doi:10.1145/368637.368660.
- 57 Vincent Vajnovszki and Dennis Wong. Greedy Gray codes for Dyck words and ballot sequences. In *International Computing and Combinatorics Conference*, pages 29–40. Springer, 2023. doi:10.1007/978-3-031-49193-1_3.
- 58 Herbert S Wilf. A unified setting for sequencing, ranking, and selection algorithms for combinatorial objects. *Advances in Mathematics*, 24(3):281–291, 1977.
- 59 Aaron Williams. The greedy Gray code algorithm. In *Workshop on Algorithms and Data Structures*, pages 525–536. Springer, 2013. doi:10.1007/978-3-642-40104-6_46.
- 60 Dennis Chi-Him Wong. *Fast algorithms to generate Beckett-Gray codes and coil-in-the-box codes*. PhD thesis, University of Guelph, 2007.