Improved Bounds for Novelty Games

Maxwell Fishelson Michael Han

January 13, 2026

Abstract

We study the *novelty* game, a combinatorial problem in which pk integers from $[1, N]$ are distributed evenly among p non-communicating players, who each output m numbers. The players must collectively ensure that at least one of them outputs a number not in the original list. Focusing on oblivious strategies, we propose a new framework for novelty games and then introduce a sequence of six optimizations based off that framework. These optimizations lead to an improvement on the upper bound compared to the previous state of the art. More specifically, we improve the bound for the $(3, 2, 1)$ game from approximately 1.71×10^6 to 193,050, which is a reduction of over 99.8%. Our techniques also lead to exponential improvements in the general $(p, k, 1)$ game, with a reduction of $e^{2k^{\frac{p}{2}}} \prod_{i=0}^{p-1}(k^i)!$. We also provide two conjectures on the lower bounds of the novelty game and conjecture that our upper bound is tight up to subexponential factors.

Contents

1 Introduction

The novelty game [LS25] is a combinatorial problem where p non-communicating players each receive k inputs from $[1, N]$ and produce m outputs, also from $[1, N]$. In order to win, novelty must hold across **all** possible inputs. In other words, in every possible case, at least one of the players has to produce an output that is novel (outside of the pk inputs).

We call this the (p, k, m) novelty game, and our goal is to reduce the **upper** bound on N, denoted $\mathfrak{B}(p, k, m)$. Note that we focus on single-attempt novelty games, i.e. improving $\mathfrak{B}(p, k, 1)$.

Remark 1. *Although novelty games are defined quite simply, determining precise upper and lower bounds on N remains a challenging task. Try analyzing the $(2, 2, 1)$ game yourself to see if you can find and prove the bound on N.*

Novelty games generalize paradoxical tournaments [LS25] and relate to various existing mathematical problems. Thus, they are a powerful tool for studying computational complexity, designing faster algorithms, producing explicit constructions that expose computational limits, and modeling non-communicating agents in distributed algorithms and graph theory [VW23][RSW22][GGNS23][APY10][AS10].

Contributions. We improve the oblivious bound $\mathfrak{B}(p, k, 1)$ significantly, reducing $\mathfrak{B}(3, 2, 1)$ from $15^7 \approx 171$ million to 193,050 (a 99.8% reduction) via six optimizations: input avoidance, cycle avoidance, reordering ancestors, merging equivalent ancestors, merging neighboring ancestors, and pruning unused outputs. We also conjecture that our oblivious bound is tight up to subexponential factors and discuss the lower bounds of novelty games briefly.

Paper Organization. Section 2 provides preliminaries, Section 3 states main theorems and results, Section 4 provides the optimization framework that we use as a foundation, Section 5 to Section 10 detail optimizations, Section 11 provides the combined optimization results, Section 12 provides the optimization results for multi-attempt variant novelty games, and Section 13 concludes with future directions. Algorithms are detailed in Appendix A, some input/output examples are provided in Appendix B, and the novelty game simulator is provided in Appendix C.

2 Preliminaries

2.1 Notation

This section presents the notation, definitions, and known results that form the foundation of our work. Generally, we try to use the same notation as [LS25], which we are building off of.

Recall that we denote upper bounds on N in general as $\mathfrak{B}(p, k, 1)$. Since we propose multiple optimizations to the original bound in [LS25], we denote the original bound by $\mathfrak{B}_0(p, k, 1)$. We then use $\mathfrak{B}_i(p, k, 1)$ for $1 \leq i \leq 6$ to denote the improved bound after Optimization i.

We denote the set of vertices by V and the set of directed edges by E, so (V, E) is a graph representation of the novelty game. We use $R(V)$ to represent the range of the input set V, that is, the set of all output vertices.

When all players use the same strategy, we say that is an *oblivious* strategy. Otherwise, we say it is a *non-oblivious* strategy. The oblivious bound is generally weaker than the non-oblivious bound, as any oblivious strategy can be considered a special case of some non-oblivious strategy.

We should also note that novelty games can be interpreted as graphs, as this is important for understanding the results in [LS25]. Under this interpretation, each number is treated as a vertex in a graph: when an input number x is used to create an output z, we draw a directed edge from x to z. However, we also introduce a new framework that makes it simpler to understand our optimizations.

A winning strategy S for the novelty game on $X = [N]$ is a function $S\colon X^k \to X$ such that for every input tuple $(x_1, \ldots, x_k) \in X^k$, the output $S(x_1, \ldots, x_k)$ satisfies $S(x_1, \ldots, x_k) \notin \{x_1, \ldots, x_k\}$. Each player applies the same strategy S to their k inputs, and the strategy is winning if, for *every possible* input list $E \subseteq X$ of size pk partitioned among the p players, at least one player outputs a novel value not in E.

We also define the notion of *cycles* in novelty games. Suppose an input x is given to Player 1, who produces an output z that becomes the input to Player 2. This process continues so that each Player i receives the output of Player $i-1$ as input. If, after L steps, the output of Player L equals the original input x, we call this an L-cycle. As shown in the key lemma for establishing winning strategies from [LS25], any winning strategy for a (p, k, m) game must prevent the existence of L-cycles for $1 \le L \le p$.

Similar to [LS25], we also use

$$f(k, d) = 1 + k + k^2 + \cdots + k^{d-1} = \frac{k^d - 1}{k - 1}$$

throughout our paper. We further introduce

$$q = f(k, p+1) = \frac{k^{p+1} - 1}{k - 1}$$

and

$$r = f(k, p) = \frac{k^p - 1}{k - 1}$$

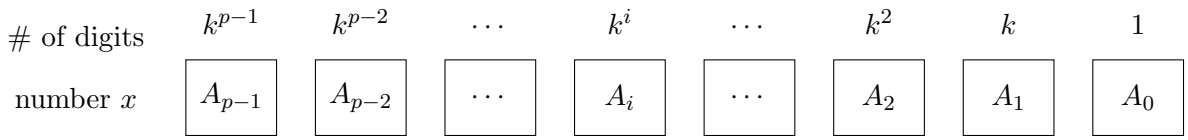
for convenience. In other words, q is the range of each digit, while r is the number of digits in each input. In the $(3, 2, 1)$ game, we have $q = 1 + 2 + 4 + 8 = 15$ and $r = 1 + 2 + 4 = 7$, leading to a bound of $q^r = 15^7$.

If a player receives input x and produces output z, we say that x is the *parent* of z, and z is the *child* of x. This terminology naturally extends to *grandparents*, *grandchildren*, and more generally to *ancestors* and *descendants*. Parents are also referred to as first-level ancestors, while grandparents are second-level ancestors.

There is only one type of oblivious strategy that is known so far, and we refer to it as the *ancestor-bookkeeping* strategy. This was first brought up in [LS25]. Such a strategy encodes information about a set of ancestors from levels 1 to $p-1$ directly within the number itself. This information is then used to distinguish the number from all the necessary ancestors. More specifically, each number is structured as a sequence of last-digit groups, corresponding to different ancestor levels. The maximum number of ancestors at level i is at most k^i, and we must retain all ancestor levels from 0 to $p-1$ to avoid L-cycles for $0 < L \le p$. The total number of digits in each number is given by the sum $k^{p-1} + k^{p-2} + \cdots + k^2 + k + 1 = f(k, p)$. For convenience, we denote the group of digits at level i as A_i, so a number x takes the form $A_{p-1}A_{p-2}\ldots A_1A_0$. See Figure 1 for an illustration, which includes the number of digits within each ancestor group.

We use $\bigoplus$ to denote concatenation, so a number x can be written as $x = \bigoplus_{i=0}^{p-1} A_i(x)$. Since A_0 consists of a single digit, we sometimes refer to it as m, $m(x)$, or m_x. The remaining digits,

Figure 1: The ancestor groups in a number x



$A_{p-1}A_{p-2}\dots A_1$, are collectively denoted by $A(x)$, so that $A(x) = \bigoplus_{i=1}^{p-1} A_i(x)$. We refer to $A(x)$ as the *ancestor digits* of x. In the specific case of the $(3, 2, 1)$ game, each number has the form $A_2A_1A_0$, where A_2 represents the grandparent and A_1 the parent. For convenience, we also denote them as G and P, respectively, so a number takes the form GPm.

Note that we frequently let x and y be inputs, z be the output, a be a child of z, and b be a child of a (and a grandchild of z). If $A_i(x)$ contributes to the construction of $A_{i+1}(z)$, then we have the relation $A_i(x) \subseteq A_{i+1}(z)$. In other words, the ancestor digits at level i of each input are concatenated to form the level $i + 1$ ancestor group of the output.

We write $A_i(x) \succ A_{i+1}(z)$ (read as "$A_i(x)$ converts to $A_{i+1}(z)$") to indicate that the ancestor group $A_i(x)$ becomes part of the group $A_{i+1}(z)$ after one additional player. We extend this notion to multiple players: $A_i(x) \succ A_j(u)$ means that $A_i(x)$ eventually becomes part of $A_j(u)$ (the ancestor group of some descendant u) after $j - i > 0$ steps. Conversely, we write $A_i(x) \not\succ A_j(u)$ to indicate that such a conversion is not possible after $j - i > 0$ players. We also note that the ancestor group convertibility relation $A_i(x) \succ A_j(u)$ depends on the specific ancestor-bookkeeping algorithm used in the strategy. For example, see the change from Equation 13 to Equation 14.

In ancestor-bookkeeping strategies, each number consists of multiple digits, with a designated digit range for each position to allow for novelty. We denote this digit range by $\mathfrak{D}(p, k)$. This digit range is calculated from the contributions of various cycle avoidance. We will use $\mathfrak{C}(L)$ to denote the contribution from L-cycle avoidance, where the corresponding p and k are fixed. We also use $s = \lceil \frac{p}{2} \rceil - 1$ when convenient.

In the case of the $(3, 2, 1)$ game, each number x is represented as a 7-digit array $\overline{x_7x_6x_5x_4x_3x_2x_1}$ [LS25]. We make use of this representation when convenient, so note that we have:

- $x_1 = m_x$ is the unique last digit of x,
- $x_3x_2 = P_x = A_1(x)$ are the parent digits (from the two parents of x),
- $x_7x_6x_5x_4 = G_x = A_2(x)$ are the grandparent digits (from the four grandparents of x).

For convenience, we occasionally use v (from [LS25]) and m_x interchangeably throughout the paper. We also define $\#_d(S)$ as the number of occurrences of digit d in the sequence S. This count will be used in the criteria for checking convertibility relations of the form $A_i(x) \succ A_j(u)$.

2.2 Related Work

The novelty game is similar to other mathematical problems such as the MISSING-STRING problem [VW23] and the range avoidance problem [RSW22][GGNS23]. It is also similar to the Nearest Codeword Problem and the Remote Point Problem, as mentioned in [APY10] and [AS10]. These problems can be used to study the computational complexity of a system, propose faster algorithms for these complex systems, and are even useful in finding explicit constructions that demonstrate the limitations of computation. The novelty game also generalizes paradoxical tournaments in the case of $p = 2$, and thus has applications in distributed algorithms and graph theory, where players model non-communicating agents.

The novelty game is known to be a generalization of the paradoxical tournament game (https://en.wikipedia.org/wiki/Tournament_(graph_theory)#Paradoxical_tournaments). This was shown in [LS25]. More specifically, when $p = 2$, it is known that the novelty game is the same as the k-paradoxical tournament game, and they have the same bounds.

There have been numerous existing results regarding paradoxical tournaments and their bounds.

Paul Erdős had the first result of $\mathfrak{B}(2, k, 1) \leq O(k^2 2^k)$ using the probabilistic method [Erd63]. George and Esther Szekeres improved this bound to $\mathfrak{B}(2, k, 1) \leq (k + 2)2^{k-1} - 1$ [SS65]. Graham and Spencer gave a new bound of $\mathfrak{B}(2, k, 1) \leq O(k^2 2^{2k-2})$ which is nearly the square of the Erdős bound in [Erd63], but they also provided an explicit construction method by using the Paley digraph [GS71]. Although this new constructive bound is asymptotically worse than the non-constructive Erdős bound, explicit constructions using Paley digraphs often lead to significantly smaller values than the general $O(k^2 2^{2k-2})$ bound.

Reid et al. [RMHH04] studied minimal dominating sets and maximal irredundant sets in tournaments, establishing both lower and upper bounds for several graph problems, including the k-paradoxical tournament problem. The webpage https://www.ttested.com/paradoxical-tournaments/ provides a helpful summary of known bounds for paradoxical tournaments. Table 1, adapted from the above webpage, presents several existing bounds for k-paradoxical tournament games. Here, LB stands for Lower Bound, UB for Upper Bound, and ? indicates that the value is unknown. We also add a new column labeled "Paley UB," based on OEIS sequence A362137 [OEI24], which reports bounds obtained via the Paley digraph construction. These bounds are often the smallest known upper bounds.

Note that $\mathfrak{B}(2, 2, 1) = 7$ and $\mathfrak{B}(2, 3, 1) = 19$ have been proven to be optimal bounds for k-paradoxical tournaments. However, the optimal bounds for $k > 3$ remain an open question.

Table 1: The existing bounds for k-paradoxical tournament games

k	Erdos LB	Szekeres LB	Optimal	Paley UB	Erdos UB	Graham UB
1	3	2	3	3	3	3
2	7	7	7	7	21	19
3	15	19	19	19	91	151
4	31	47	?	67	149	1031
5	63	111	?	331	353	6427

We notice that the existing methodology used works well for $\mathfrak{B}(2, k, 1)$, but does not work when $p > 2$. However, they are still very useful in understanding novelty games and the existing bounds on novelty games. Next, we will briefly describe the existing results from [LS25] for novelty games.

2.3 Known General Bounds of Novelty Games

For oblivious strategies, the existing bound from [LS25] is

$$\mathfrak{B}_0(p, k, 1) \leq f(k, p + 1)^{f(k,p)} = q^r.$$

As an example, when $(p, k) = (3, 2)$, this gives $\mathfrak{B}(3, 2, 1) \leq 15^7 \approx 1.71 \times 10^8$. For non-oblivious strategies, tighter bounds can be achieved. From [LS25], we have that

$$\mathfrak{B}(p, k, 1) \leq \mathfrak{B}(2, k, 1)^{k^{p-2}} \quad \text{or} \quad \mathfrak{B}(p, k, 1) \leq (13k^3 \cdot 2^k)^{k^{p-2}}.$$

Oblivious strategies are generally preferred due to their simplicity and structure. Thus, we focus on improving the oblivious bound $\mathfrak{B}_0(p, k, 1)$ through several optimization techniques.

3 Main Theoretical Results

3.1 Improved Oblivious Bound

Theorem 1. *The oblivious bound for novelty games is improved to:* $\mathfrak{B}(p, k, 1) \leq \prod_{i=0}^{p-1} \binom{\mathfrak{D}(p,k)}{k^i}$, *where* $\mathfrak{D}(p, k)$ *is given by Equation 10. This is roughly* $e^{2k^{\frac{p}{2}}} \prod_{i=0}^{p-1}(k^i)!$ *times smaller than* q^r.

This bound, achieved through multiple optimizations that we describe in Section 5–10, significantly reduces the upper bound on N. For example, the upper bound for the $(3, 2, 1)$ game is reduced from 15^7 to 193,050.

This makes the strategy more practical for combinatorial applications.

3.2 Conjectures

Conjecture 1. *For oblivious ancestor-bookkeeping strategies with* $p > 2$ *players, the lower bound is* $\mathfrak{B}(p, k, 1) \geq \prod_{i=0}^{p-1} \binom{k^p+1}{k^i}$.

Conjecture 2. *For all strategies (oblivious and non-oblivious) with* $p \geq 2$ *players, the lower bound is* $\mathfrak{B}(p, k, 1) \geq \mathfrak{B}(2, k, 1)^{k^{p-2}}$.

These conjectures are about lower bounds of novelty games, which are not well studied yet. They suggest our improved oblivious bound is near-tight and motivate future work on lower bounds.

4 Framework

4.1 Overview of Optimizations

We propose six optimizations to reduce $\mathfrak{B}(p, k, 1)$:

1. input avoidance
2. cycle avoidance
3. reordering ancestors
4. merging equivalent ancestors
5. merging neighboring ancestors
6. pruning unused outputs

We use $\mathfrak{B}(3, 2, 1)$ as a running example, and overall we reduce the bound from $15^7 \approx 171$ million to 193,050.

4.2 Existing Strategy

The existing oblivious bound [LS25] is $\mathfrak{B}(p,k,1) \leq q^r$, where q and r are defined in Subsection 2.1.

Let us first briefly describe the ancestor-bookkeeping oblivious strategy proposed by [LS25] to help understand how the existing $\mathfrak{B}(3,2,1)$ bound is calculated. In order to achieve the novelty property, the original oblivious strategy keeps track of the last digits of some of its ancestors, stored in the number itself. Intuitively, each number is treated as a stack of last-digit sets, located at different ancestor levels. These ancestor levels are blocks of digits in the number itself. The maximum number of ancestors at a level i is at most k^i, and we need to keep all the ancestor levels from 0 to $p-1$ to avoid L-cycles for $0 < L \leq p$.

We briefly describe the existing strategy in Algorithm 3, which is presented in more depth in [LS25]. When a player receives k inputs, their ancestor groups at level A_i are concatenated to form the A_{i+1} group of the output number by Algorithm 1. In other words, if x is an input and z is the output, then $A_i(x) \succ A_{i+1}(z)$. This process determines all digits of the output number except the last one, $A_0(z)$. The final digit $A_0(z)$ or $m(z)$ is then selected to differ from the last digits of all its ancestors by Algorithm 2, ensuring novelty.

In the case of $(3,2,1)$, each number is in the form of $A_2A_1A_0$, with $|A_2| = 4, |A_1| = 2, |A_0| = 1$. There are two inputs, so there are $2^3 + 2^2 + 2^1 = 14$ ancestors across the three levels. Adding one additional digit for the player's own output gives a required range of $14 + 1 = 15$. Each number consists of $2^2 + 2^1 + 1 = 7$ digits, so the total number of distinct numbers is $15^7 = 170{,}859{,}375$, or approximately 171 million. This example is computed using the ancestor-bookkeeping strategy implemented in Algorithm 1, which also serves as the baseline strategy described in Algorithm 3 that we later optimize.

Another way to think about it, which is directly described in [LS25], is as follows: Let $x = \overline{x_7x_6x_5x_4x_3x_2x_1}$ and $y = \overline{y_7y_6y_5y_4y_3y_2y_1}$ be two 7-digit input numbers. The digits of these are then used to produce the output:

$$z = \overline{x_3x_2y_3y_2x_1y_1v}. \tag{1}$$

As when x and y are known, the first 6 digits of z are determined, we simply have to choose the last digit m_z to guarantee novelty.

Let's present one example for the $(3,2,1)$ novelty game to demonstrate the rule. We will use hexadecimal digits A, B, C, D, E, F to stand for digits $10, 11, 12, 13, 14, 15$ respectively. The example in Figure 11 has two inputs $x = 0123456$ and $y = EDCBA98$, and the output is $z = 45A9687$. The output z has a unique last digit 7 to guarantee that it is a distinct number from all its ancestors.

We propose multiple optimizations based on the existing ancestor bookkeeping strategy. Our proposed optimizations keep the ancestor bookkeeping behavior and slightly modify the algorithm to bookkeep the ancestors and to select the last digit more carefully, in order to reduce the total number of valid outputs and thus the oblivious bound too.

4.3 Theoretical Foundations

We now present the core definitions, assumptions, and results that underlie our analysis. These form the theoretical basis for the bound improvements and constructions developed in the remainder of the paper.

Lemma 1 (Minimal Strategy Lemma)**.** *The set of all outputs must be the same as the set of all inputs for a winning strategy on a minimum number of vertices. In other words,* $R([N]) = [N]$.

Proof. We proceed with proof by contradiction. For the sake of contradiction, assume that there is one x that is not in the set of outputs for a winning strategy S. We can then remove this x from the vertices. We can verify that the strategy S is still a winning strategy for the new set of vertices. But the number of vertices has been reduced by 1, contradicting N is the minimum. So such a x does not exist. □

Lemma 1 tells us that we can check if N is not the minimum for a given winning strategy by computing the set of all outputs. We simply have to check to see if there are any x that do not exist in the output set.

Corollary 1. *$\mathfrak{B}_0(p, k, 1) = q^r$ is not the lower bound for the novelty game.*

Proof. Based on Lemma 1, we just need to show that there exists at least one x that is not a valid output. We proceed with proof by contradiction. If d is the last digit of the last input, then since the strategy requires the last digit of the output be different from d, we know that $x = ...dd$ is not a valid output. □

Lemma 1 tells us it is possible to reduce the existing oblivious bound. One way is to remove all the inputs that are never outputs of the winning strategy, leaving only all the valid outputs. This would reduce the total number of vertices while still maintaining a winning strategy. Lemma 2 provides a specific construction method to accomplish this.

Lemma 2 (Range Reduction Lemma)**.** *Let $X = [N]$ have a winning strategy S whose range is $Y = \{y_1, \ldots, y_M\}$ with $M < N$. Define $X_2 = [M]$. Then there is a bijection*

$$f \colon Y \to X_2, \quad f(y_i) = i,$$

and a corresponding winning strategy S_2 on X_2.

Proof. Define $f(y_i) = i$ for $1 \leq i \leq M$. The new domain X_2, with fewer elements, will have a winning strategy S_2 constructed from S. Given any k inputs from $X_2 = \{1, \ldots, M\}$, apply f^{-1} to obtain their preimages in Y. Use the original strategy S on these values to produce a novel output $y \in Y$. Then let S_2 output $f(y) \in X_2$. Since f is bijective, $f(y)$ differs from all original inputs in X_2, so S_2 is also a winning strategy. □

Lemma 2 provides a systematic way to construct a winning strategy by reducing the number of possible outputs. If the outputs occupy only $M < N$ values, we can remap the strategy from $[N]$ to $[M]$, preserving correctness and reducing the bound accordingly. Though this kind of normalization via bijections appears implicitly in many combinatorial and game-theoretic arguments, we state it here explicitly for clarity and later reference.

By Lemma 1 and Lemma 2, if we reduce the number of valid outputs for a winning strategy, then the bound $\mathfrak{B}(p, k, 1)$ will be reduced. This serves as the theoretical foundation of our framework to improve the bound of novelty games.

Next we will describe our new optimizations for the winning strategy to reduce the oblivious bound.

5 Optimization 1: Input Avoidance

This optimization forms the conceptual foundation for those that follow.

5.1 Input Avoidance in the $(3, 2, 1)$ Game

We begin by considering the worst case presented in [LS25], which is when the input digits are all distinct. In other words, when each digit from 0 to 13 appears exactly once across the two inputs. Note that in this case, the last digit of the output does not have to be a brand new digit. Instead, we can simply reuse one of the existing digits from the two inputs. This allows us to go from needing 15 digits to needing 14 digits. We can further optimize this, bringing it down to needing 13 digits by allowing a digit of overlap between the two inputs.

Let's modify the existing algorithm presented in [LS25] to show that these reductions are valid.

However, instead of simply letting v be a digit distinct from all x_i and y_i as in [LS25], we do some casework. Let d be the number of unique digits present in the two inputs. We define v as follows:

$$v = \begin{cases} \text{one of the unused digits} & \text{if } d < 13 \\ x_1 \text{ if } x_1 \notin \{A(x) \cup A(y)\} \text{ else } y_1 & \text{if } d = 13 \end{cases}$$

We now present several examples in Figure 12 to illustrate the new rule. We will use hexadecimal digits A, B, C, D, E, F to stand for digits $10, 11, 12, 13, 14, 15$ respectively.

- Example 1: Inputs are $x = 0123456$ and $y = CBA9999$, and the output is $z = 4599697$. Because the number of unique digits in the two inputs is $11 < 13$, the output z selects 7 as its last digit, making it a distinct number from all its ancestors. This is actually the same processing as the original algorithm.
- Example 2: Inputs are $x = 0123456$ and $y = CBA9876$, and the output is $z = 4587666$. The number of unique digits in the two inputs is 13, and the output z reuses the same last digit as the two inputs, but it is still a distinct number from all its ancestors.
- Example 3: Inputs are $x = 0123453$ and $y = CBA9876$, and the output is $z = 4587366$. The number of unique digits in the two inputs is 13, and the output reuses the same last digit as the second input y, but it is still a distinct number from all its ancestors.

To show this works for all inputs, we consider two cases.

1. If $d < 13$, then we can simply set v to be different from all x_i and y_i, similar to [LS25].
2. If $d = 13$, we can show that the ancestor digits of the output must be different from the ancestor digits of the two inputs. Begin by noting that we know there is exactly one duplicate digit among the 14 digits of the two inputs. WLOG, assume x_1 is chosen as the last digit of the output. This means the output number is

$$z = \overline{z_7 z_6 z_5 z_4 z_3 z_2 z_1} = \overline{x_3 x_2 y_3 y_2 x_1 y_1 x_1}.$$

We can further split this into two cases:

(a) If $x_1 = y_1$, then we know all the other digits in the inputs are different from each other. Thus, the first 4 digits of z will be different from those of x or y due to our construction (e.g. $z_7 = x_3 \neq x_7$). This makes z different from its parents, which avoids 1-cycles. We also find that z will be different from its grandparents and great-grandparents, thus avoiding 2-cycles and 3-cycles. This is because the last digit of the output $z_1 = x_1$ will be different from $A(z)$, which represents the last digits of z's grandparents and great-grandparents.

(b) If $x_1 \neq y_1$, then we can focus on when one of x_1, y_1 is the duplicate digit. The case when the duplicate is between two ancestor digits is trivial, as we can simply set v to x_1 and it will be different from all the digits in the inputs. Now, to stay consistent with our assumption above, let y_1 be the duplicate digit so we can let $v = x_1$. We have $z \neq x$ as we know $A_2(z) \neq A_2(x)$. Furthermore, since x_1 is different from all the other input digits, we know that z is different from any of its 14 ancestors.

With this optimization, we bring the upper bound on $\mathfrak{B}(3, 2, 1)$ from 15^7 to 13^7, which is a reduction of more than 60%.

5.2 The $(p, k, 1)$ Game

The same logic can be extended to general k values. In general, we reduce the base from q to $q - k$ according to the below algorithm.

1. If the total number of unique digits across all k input numbers is less than $q - k$, then there is always at least one digit unused among them. We can safely choose such a digit as the final digit v of the output number, ensuring its novelty.

2. If the number of unique digits among all k input numbers is exactly $q - k$, then there are precisely k repeated digits among the $q - 1$ ancestor digits. We consider two subcases:

   (a) If all k input numbers share the same last digit, say $m(x_i) = a$ for all $1 \leq i \leq k$, then a must not appear among the other ancestor digits. In this case, we simply set $v = a$. Similarly to the $(3, 2, 1)$ case, it is clear that $z \neq m(x_i)$ for all $1 \leq i \leq k$.

   (b) The other case is when some of the last digits of the k inputs are not equal. This allows us to choose a last digit v from among them such that v does not appear among the other ancestor digits.

   Each input has

   $$r = \frac{k^p - 1}{k - 1} = \frac{q - 1}{k}$$

   digits, so the k inputs contain a total of $q - 1$ digits.

   Furthermore, we know the number of unique digits across all inputs is $d = q - k$. Since $q - 1 > q - k$, by the pigeonhole principle, there is at least one last digit that is not a duplicate of any of the ancestor digits. We can then pick this v as the last digit of the output.

   However, we still have the case where $m(z) = m(x_i)$ for some input x_i. We simply have to show that $A(z) \neq A(x_i)$ to complete the proof. Note that all ancestors with last digit $m(z)$ are among the k inputs which are fully known by the player. As $r = \frac{k^p - 1}{k - 1}$ is much larger than k, we can easily find such an ancestor digit that makes this inequality true. Thus, we have that z is different from any of its $q - 1$ ancestors by the same logic as in the $(3, 2, 1)$ case.

5.3 Summary of Results

The key idea is that when a player produces a new output, we already know the inputs used to make that output. This redundancy exists because the current player already knows the k input numbers they are using to make the output. Thus, even if the player chooses the same last digit in

the output as one of the input numbers, we can still guarantee that the output number is different from all of the input numbers. We can further guarantee that the output will be different from the other $q-1-k$ higher-level ancestors, as we ensure the output's last digit is distinct from the ancestor digits.

Thus, this optimization reduces the digit range from q to $q-k$ in general. Our oblivious bound formula with Optimization 1 is thus:

$$\mathfrak{B}_1(p,k,1) = (q-k)^r.$$

Pseudocode for this algorithm can be found as Algorithm 5 in Appendix A. The below table is a comparison of the original bound and the reduced bound for small values of p and k.

Table 2: Comparison between $\mathfrak{B}_0(p,k,1)$ and $\mathfrak{B}_1(p,k,1)$

old, new	$k=2$	$k=3$	$k=4$	$k=5$
$p=2$	343, 125	28561, 10000	4084101, 1419857	88E7, 30E7
$p=3$	17E7, 62E6	67E19, 24E19	32E39, 11E39	97E66, 35E66
$p=4$	23E21, 86E20	20E82, 75E81	19E214, 70E213	17E450, 65E449
$p=5$	60E54, 22E54	78E308, 28E308	12E1068, 44E1067	13E2804, 51E2803
$p=6$	34E131, 12E131	11E1105, 41E1104	23E5100, 87E5099	37E16758, 13E16758
$p=7$	42E304, 15E304	70E3841, 26E3841	15E23696, 58E23695	64E97452, 23E97452

This optimization motivates further optimizations, as we realize that the digit range q can be further reduced. Next, we present Optimization 2, which brings down the digit range even more by aggressively avoiding cycles.

6 Optimization 2: Cycle Avoidance

As shown in [LS25], if a strategy can avoid cycles of length 1 to p, then it is a winning strategy for the $(p,k,1)$ game. We focus on avoiding cycles across all ancestors, instead of just avoiding the immediate ancestors (inputs).

6.1 The $(3,2,1)$ Game

Through careful construction and some casework, we will reduce the digit range from $q-k=13$ to $\mathfrak{D}(3,2)=11$. To initially describe our strategy, we will focus on x without loss of generality. Any arguments made regarding x and z can easily be applied to y and z.

To guarantee novelty, we want to avoid all L-cycles. In the $(3,2,1)$ game, we have $1 \le L \le 3$. We proceed with casework:

1. To avoid 3-cycles, we claim that having $m_z \notin G_x$ is sufficient. Due to our construction, Algorithm 1, any grandchild b of z must satisfy $m_z \in G_b$. If we have $m_z \notin G_x$, then it follows that $G_b \not\subseteq G_x$. In particular, this implies $G_b \neq G_x$, so $b \neq x$, and we cannot have a 3-cycle.

2. To avoid 1-cycles, note that the condition $m_z \notin G_x$ is sufficient. To prove this, assume for the sake of contradiction that a 1-cycle occurs while $m_z \notin G_x$. This means we need to have

$z = x$, which implies that $P_z = P_x$, $G_z = G_x$, and $m_z = m_x$. However, since $m_x \in P_x \subseteq G_x$, we would have

$$m_z = m_x \in G_x,$$

which violates our assumption. Thus, no 1-cycle can occur when we have $m_z \notin G_x$.

3. To avoid 2-cycles, let's first consider what is necessary to have a 2-cycle. By definition (Algorithm 1), we would need $P_z \succ G_a$ and $m_z \succ P_a$ where a is z's child. Since a 2-cycle means $G_a = G_x$ and $P_a = P_x$, we would also need

$$P_z \succ G_x \quad \text{and} \quad m_z \succ P_x.$$

Now, let's consider two subcases:

(a) $P_z \not\succ G_x$. By the above, we would not have a 2-cycle.

(b) $P_z \succ G_x$. This means it is possible to have $G_a = G_x$, and thus we must enforce $m_z \not\succ P_x$ (i.e. $m_z \notin P_x$) to avoid 2-cycles.

This completes our proof. To summarize, our requirements for the strategy are the following:

$$\begin{cases} m(z) \notin G(x) \\ m(z) \notin P(x) \quad \text{if } P(z) \succ G(x) \end{cases} \tag{2}$$

We now present several examples in Figure 13 to illustrate the new rule. We will use hexadecimal digits A to stand for the digit 10.

- Example 1: Inputs are $x = 0123456$ and $y = A987654$, and the output is $z = 4565644$. Because $A_1(z) = 64$ cannot convert to $A_2(x) = 0123$ or $A_2(y) = A987$, the last digit $m(z)$ does not need to exclude any digits in $A_1(x)$ or $A_1(y)$. So $m(z)$ just needs to exclude the 8 digits in $A_2(x)$ and $A_2(y)$ by taking the smallest value 4. The output reuses the same last digit as the second input (which is just a coincidence), and it is still a distinct number from all its ancestors.

- Example 2: Inputs are $x = 0123452$ and $y = A987653$, and the output is $z = 4565236$. Because $A_1(z) = 23$ can convert to $A_2(x) = 0123$, the last digit $m(z)$ needs to exclude the digits in $A_1(x) = 45$. So $m(z)$ needs to exclude the 8 digits in $A_2(x)$ and $A_2(y)$ and the 2 digits in $A_1(x) = 45$ by taking the smallest value 6. The output is still a distinct number from all its ancestors.

Using this new rule (2), we can now calculate the minimum needed range for each digit. In any case, we would have $m(z) \notin G(x)$ and $m(z) \notin G(y)$, which gives most $|G(x)| + |G(y)| = 8$ restricted digits. Additionally, we focus on the worst case, which is when $P(z) \succ G(x)$ and $P(z) \succ G(y)$ (Case 3b). In this case, we need $m_z \notin P_x$ and $m_z \notin P_y$, giving $|P_x| + |P_y| = 4$ more restricted digits. However, we also need to consider duplicate digits. Notice that since $P(z) \succ G(x)$ and $P(z) \succ G(y)$, $P(z)$ is present in both $G(x)$ and $G(y)$, leading to an overlap of $|P(z)| = 2$ digits. Thus, we end up requiring

$$8 + 4 - 2 + 1 = 11 \text{ distinct digits.}$$

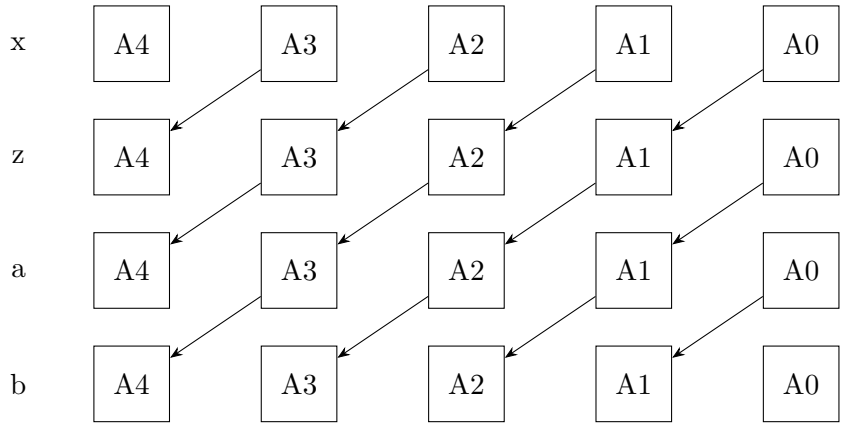
We made some assumptions about what the "worst case" is, but it's fairly straightforward to check the other cases and notice that we never require more than 11 distinct digits.

We've shown above that with our new construction method (Algorithm 7), we always have at most 11 distinct digits in the inputs. Thus, a range of $[0, 11)$ for each digit is sufficient. This means we can reduce the original bound $\mathfrak{B}_0(3, 2, 1) = 15^7 = 170,859,375$ to our new bound $\mathfrak{B}_2(3, 2, 1) = 11^7 = 19,487,171$, which is a 88.6% reduction.

6.2 Theoretical Guarantees

We will first provide the theoretical foundations needed for this optimization, and then we will derive cycle avoidance rules to reduce the bound. We begin by presenting a crucial property. For Figure 2 below, the input is x, the output is z, a is a child of z, and b is a grandchild of z (and a child of a). The arrow between two boxes stands for ancestor group convertibility.

Figure 2: The ancestor group chain-shifting property



Property 1 (Chain-Shifting Property)**.** *Recall that due to the way we do ancestor-bookkeeping (Algorithm 1), we have that* $A_i(x) \succ A_{i+1}(z)$.

To visualize this key property, we can think of each number as a chain of ancestor groups: $\overline{A_{p-1}A_{p-2}...A_2A_1A_0}$. These chains are laid horizontally, with an input directly above its output, as in Figure 2. As we go from one ancestor chain to the next, each ancestor group "shifts" one to the left. In other words, we have $A_0(x) \succ A_1(z), A_1(x) \succ A_2(z)$, and so on. The leftmost ancestor group A_{p-1} is thus "lost," as we do not need to store information about whether there are $(p+1)$-cycles.

Property 2. *The chain-shifting property is also true for* x *and its descendant* u *after* L *levels: if* $i - j = L$*, then we must have* $A_j(x) \succ A_i(u)$.

As our goal is to avoid cycles, we proceed by considering when a cycle can exist.

Lemma 3. *In order for an* L*-cycle to exist between an input* x *and an output* z*, we must have* $A_j(z) \succ A_i(x)$ *for all* $i - j = L - 1$.

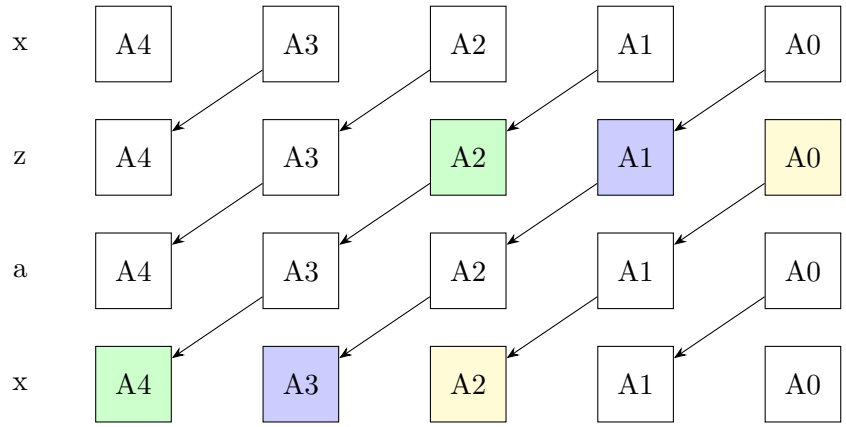
Proof. In order for an L-cycle to exist, the output L levels down from x, which is $L-1$ levels down from z, must be the same as x. Thus, applying Property 2 for all corresponding ancestor groups in z and x gives us Lemma 3. □

The below Figure 3 shows such an example for 3-cycle x-z-a-x existence with the 5-player novelty game. We can easily see that for 3-cycle to exist, we must have $A_2(z) \succ A_4(x)$, $A_1(z) \succ A_3(x)$, and $A_0(z) \succ A_2(x)$.

Now, we present a useful property between cycles of different lengths.

Theorem 2. *If a strategy avoids every* L*-cycle, then it also avoids every* M*-cycle where* $M \mid L$.

Figure 3: The necessary condition for 3-cycle existence with 5 players



Proof. Assume for the sake of contradiction that our strategy avoids all L-cycles, but there still exists an M-cycle. This means there is a number x that returns to itself after M levels. However, we know $L = M \times N$ for some integer N, so we can simply repeat this M-cycle N times to produces an L-cycle. This contradicts our assumption that no L-cycles exist, and so no M-cycles can exist. □

Theorem 3. *If a strategy avoids all cycles of lengths* $\lfloor \frac{p}{2} \rfloor + 1, \lfloor \frac{p}{2} \rfloor + 2, \ldots, p$*, then it also avoids all cycles of lengths* $1, 2, \ldots, p$*.*

Proof. For any integer M with $1 \leq M \leq \lfloor \frac{p}{2} \rfloor$, it's apparent that there exists some L with $\lceil \frac{p}{2} \rceil \leq L \leq p$ such that $M \mid L$. The result follows by a simple application of Theorem 2. □

We can naturally expand the strategy we used in the $(3, 2, 1)$ game to the $(p, k, 1)$ game. Based on Theorem 2 and Theorem 3, our general strategy is as follows:

$$\begin{cases} m(z) \notin A_{p-1}(x) \\ m(z) \notin A_L(x), \text{where } \lfloor \frac{p}{2} \rfloor < L < q - 1 \quad \text{if } \forall j > 0, A_j(z) \succ A_{j+L-1}(x) \end{cases} \tag{3}$$

6.3 A Detailed Example for Necessary Conditions of L-cycle Existence

For the $(7, 2, 1)$ game, the below Figure 4 – 7 illustrate the necessary conditions for L-cycle existence $(4 \leq L \leq 7)$. Note based on Theorem 3, we just need to avoid cycles of $4 \leq L \leq 7$, then we also avoid all cycles of $1 \leq L \leq 7$. The arrow between two boxes stands for ancestor group convertibility relation. These figures help us understand in the next subsection the calculation of the contribution of forbidden digits from cycle avoidance rules.

Figure 4: Necessary condition for 7-cycle existence

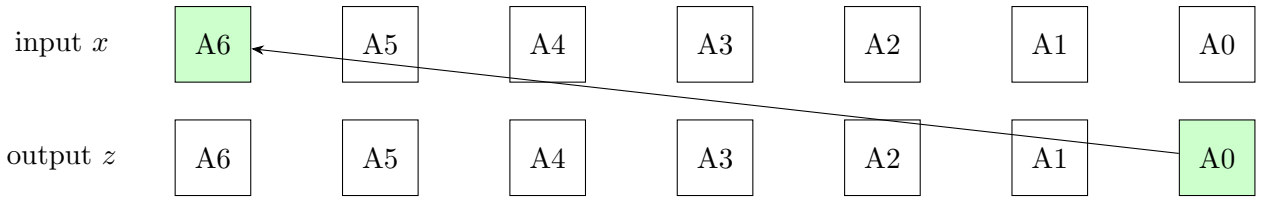


Figure 5: Necessary condition for 6-cycle existence

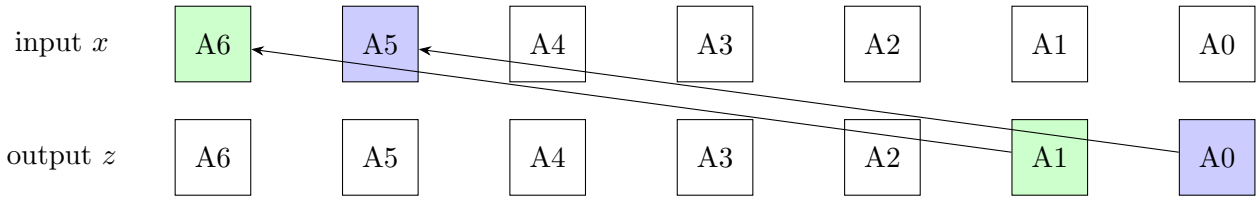


Figure 6: Necessary condition for 5-cycle existence

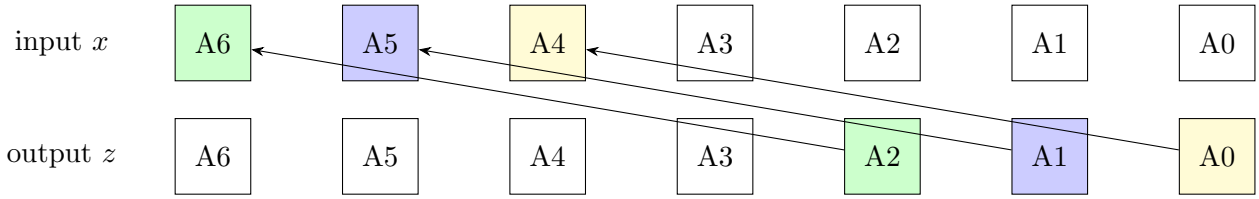
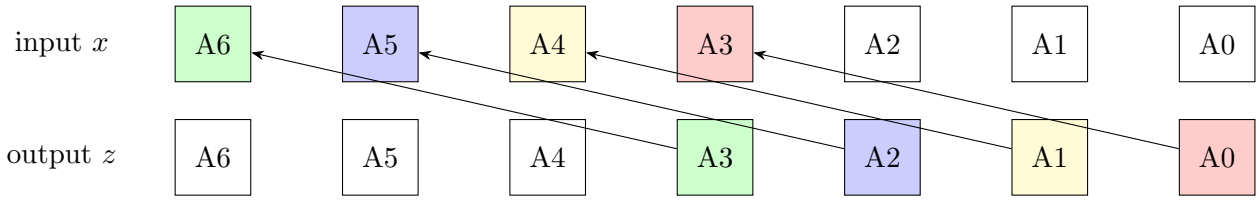


Figure 7: Necessary condition for 4-cycle existence



More specifically, because 7-cycle avoidance means 1-cycle avoidance, 6-cycle avoidance means both 3-cycle avoidance and 2-cycle avoidance, if we can break the necessary conditions for these cycles, then we will have L-cycle avoidance for all $1 \leq L \leq 7$.

6.4 Digit Range Contributions

Recall that we use $\mathfrak{C}(L)$ to denote the forbidden-digit contribution from L-cycle avoidance. The below is the basic formula to calculate the required digit range from the contributions of all L-cycle avoidance

$$\mathfrak{D}(p,k) = \sum_{L=1}^{p} \mathfrak{C}(L) + 1. \tag{4}$$

Based on the theoretical guarantees from Subsection 6.2, we will analyze which values of m_z must be excluded to avoid cycles of various lengths. Each expression below gives the worst-case number of digits that must be excluded to eliminate L-cycles or $(p-j)$-cycles, taking into account overlaps among the inputs. Recall that we denote the output as z and the inputs as $x_1, x_2, \ldots x_k$. We proceed with casework based off the above rule (3).

1. To avoid p-cycles, we must ensure m_z does not appear in the A_{p-1} group of any input (see Figure 4. Since there are at most k^p ancestors in A_{p-1}, the total number of digits we exclude is
$$\mathfrak{C}(p) = k \cdot k^{p-1} = k^p.$$

2. To avoid $(p-1)$-cycles (see Figure 5), we do not need to exclude m_z from all A_{p-2} groups, which would require k^{p-1} digits. This naive way of contribution can be improved. If $A_1(z)$ is

not a subset of any $A_{p-1}(x)$, then m_z faces no restriction. If $A_1(z)$ is a subset of exactly one input's A_{p-1} group, and that input's A_{p-2} contains k^{p-2} distinct digits, then the contribution is k^{p-2}. When $A_1(z)$ is a subset of a second input's A_{p-1} group, this input must have at least k duplicates of $A_1(z)$, so its A_{p-2} group contributes only $k^{p-2}-k$ new digits. Continuing this process for the remaining $k-2$ inputs, the total becomes

$$\mathfrak{C}(p-1) = k^{p-2} + (k-1)(k^{p-2}-k) = k^{p-1} - k^2 + k.$$

In general, let us use F to denote the contribution of the first matched input, and use S to denote the contribution of the second matched input, then the formula for $\mathfrak{C}(L)$ is

$$\mathfrak{C}(L) = F + (k-1)S.$$

3. To avoid $(p-2)$-cycles (see Figure 6), fully excluding m_z from all A_{p-3} groups would cost k^{p-2} digits, but this can be reduced. If one input matches, it contributes $k^{p-3}-k$ since $A_1(z) \succ A_{p-2}$ has been counted for $(p-1)$-cycles and $A_1(z)$ is not new here: note in the worst case the previous $(p-1)$-cycle avoidance must have contributed because the $(p-1)$-cycle contribution is roughly k times of the $(p-2)$-cycle contribution. Each additional input adds $k^{p-3}-k^2-k$ more digits, because its A_{p-3} group contains at least k^2+k repeated elements due to overlap at levels A_1 and A_2. The total is therefore

$$\mathfrak{C}(p-2) = k^{p-3} - k + (k-1)(k^{p-3}-k^2-k) = k^{p-2} - k^3.$$

4. To avoid $(p-3)$-cycles (see Figure 7), the first matched input contributes $k^{p-4}-k^2-k$ digits (because $A_2(z)$ and $A_1(z)$ are not new). Each additional match adds $k^{p-4}-k^3-k^2-k$ digits, reflecting the necessary overlap of k^3+k^2+k values. This gives a total of

$$\mathfrak{C}(p-3) = k^{p-4} - k^2 - k + (k-1)(k^{p-4}-k^3-k^2-k) = k^{p-3} - k^4 - k^2.$$

5. In the general case of avoiding $(p-j)$-cycles, each additional matched input adds

$$k^{p-j-1} - (k^j + k^{j-1} + \cdots + k^2 + k)$$

more forbidden digits for m_z. Thus, the total contribution is:

$$\begin{aligned}
\mathfrak{C}(p-j) &= (k^{p-j-1} - \sum_{i=1}^{j-1} k^i) + (k-1)\left(k^{p-j-1} - \sum_{i=1}^{j} k^i\right) = k^j + k\left(k^{p-j-1} - \sum_{i=1}^{j} k^i\right) \\
&= k^{p-j} - k^{j+1} + k - \sum_{i=1}^{j-1} k^i = k^{p-j} + k^j + k - \sum_{i=1}^{j+1} k^i \\
&= k^{p-j} - k^{j+1} + k - kf(k,j-1) = k^{p-j} + k^j + k - kf(k,j+1).
\end{aligned}$$

That is, the general formula is

$$\mathfrak{C}(p-j) = k^{p-i} - k(k^i-1) - kf(k,i-1) = k^{p-j} + k^j + k - kf(k,j+1). \tag{5}$$

This summation stops when $k^{p-j-1} - \sum_{i=1}^{j} k^i$ becomes negative. At that point, additional matched inputs do not contribute any new forbidden digits. This occurs when $2j \geq p-1$, or equivalently, when $j \geq \frac{p-1}{2}$. In such cases, the contribution is just from the first input, thus it simplifies to $k^{p-j-1} - \sum_{i=1}^{j-1} k^i = k^{p-j-1} - \frac{k(k^{j-1}-1)}{k-1} = k^{p-j-1} - kf(k,j-1)$.

Remark 2. *It is obvious that* $\mathfrak{C}(L) \leq k^L$. *This is because the maximum possible number of unique digits for L-cycle avoidance is*

$$\sum_{i=1}^{k} |A_{L-1}| \leq k \cdot k^{L-1} = k^L.$$

Remark 3. *It can be observed or easily verified that we have the property* $\mathfrak{C}(j) > \sum_{i=1}^{j-1} \mathfrak{C}(j)$. *This is because we have the property* $|A_j| > \sum_{i=0}^{j-1} |A_i|$ *and* $\mathfrak{C}(j)$ *is proportional to* $|A_j|$.

Let $s = \lceil \frac{p}{2} \rceil - 1$, the minimum cycle to avoid is when $j = s$, that is, the minimum cycle to avoid is $p - s$. Now let us calculate the contribution of this $(p - s)$-cycle avoidance, which is the edge case that we must consider.

1. If p is even, the contribution of each additional matched input $k^{p-s-1} - \sum_{i=1}^{s} k^i > 0$. So this $p - s$ cycle has total contribution of $\mathfrak{C}(p - s) = k^{p-s} + k^s + k - kf(k, s + 1)$. This is actually the same form as the standard $\mathfrak{C}(p - j)$ formula.

2. However, if p is odd, then the contribution of each additional matched input $k^{p-s-1} - \sum_{i=1}^{s} k^i < 0$. So this $p - s$ cycle has contribution of $\mathfrak{C}(p - s) = k^{p-s-1} - kf(k, s - 1)$ instead of $k^{p-s} - k^{s+1} + k - kf(k, s - 1)$, because only the contribution of the first input counts. There is a difference of $k^{p-s} - k^{s+1} + k - k^{p-s-1}$ to subtract:

   $$\text{term to subtract} = k^{p-s} - k^{s+1} + k - k^{p-s-1}$$

   Notice that when p is odd, actually $p - s = s + 1$, so it becomes:

   $$\text{term to add} = k^{p-s-1} - k$$

Let us use E to stand for this extra term to add, we have:

$$E = \begin{cases} k^{p-s-1} - k, \text{ where } s = \lceil \frac{p}{2} \rceil - 1 & \text{if } p \text{ is odd} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Then we can have a unified formula for this edge case

$$\mathfrak{C}(p - s) = k^{p-s} + k^s + k - kf(k, s + 1) + E.$$

6.5 Digit Range Reduction

Note that it is only necessary to sum across half the cycles due to Theorem 3. Thus, our general formula is:

$$\mathfrak{D}(p, k) = \sum_{i=0}^{\lceil \frac{p}{2} \rceil - 1} \mathfrak{C}(p - i) + 1. \tag{7}$$

Remark 4. *Let* $s = \lceil \frac{p}{2} \rceil - 1$. *It can be easily verified that* $\mathfrak{D}(p, k) \leq \sum_{i=0}^{s} k^{p-i} = k^{p-s} f(k, s + 1)$, *from the above Equation 7 and Remark 2.*

Substitute $\mathfrak{C}(p-i)$ with the expression in Equation 5, we get

$$\mathfrak{D}(p,k) = k^p + \sum_{i=1}^{\lceil \frac{p}{2} \rceil - 1} (k^{p-i} - k(k^i - 1) - kf(k, i-1)) + E + 1. \tag{8}$$

Let $s = \lceil \frac{p}{2} \rceil - 1$ and using the fact that $\sum_{i=0}^{s} k^i = f(k, s+1)$, we can simplify the formula:

$$\begin{aligned}
\mathfrak{D}(p,k) &= k^p + \sum_{i=1}^{s} (k^{p-i} - k(k^i - 1) - kf(k, i-1)) + E + 1 \\
&= k^p + \sum_{i=1}^{s} (k^{p-i} - k^{i+1}) - \frac{k(\frac{k^s - 1}{k-1} - s)}{k-1} + sk + E + 1 \\
&= \sum_{i=0}^{s} k^{p-i} - \sum_{i=1}^{s} k^{i+1} - \frac{k(f(k,s) - s)}{k-1} + sk + E + 1 \\
&= k^{p-s} f(k, s+1) - k^2 f(k,s) - \frac{k(f(k,s) - s)}{k-1} + sk + E + 1.
\end{aligned} \tag{9}$$

Thus, our final digit range formula is:

$$\mathfrak{D}(p,k) = k^{p-s} f(k, s+1) - k^2 f(k,s) - \frac{kf(k,s)}{k-1} + \frac{sk^2}{k-1} + E + 1, \text{ where } s = \lceil \frac{p}{2} \rceil - 1. \tag{10}$$

To calculate the difference between $\mathfrak{D}(p,k)$ and the original power base $f(k, p+1) = \sum_{i=0}^{p} k^i$, we use Equation 9:

$$\Delta\mathfrak{D} = f(k,p+1) - \mathfrak{D}(p,k) = \sum_{i=0}^{p} k^i - (k^p + \sum_{i=1}^{s} (k^{p-i} - k^{i+1}) - \frac{k(\frac{k^s-1}{k-1} - s)}{k-1} + sk + E + 1)$$

$$= 2\sum_{i=1}^{s} k^{i+1} - sk + \frac{k(\frac{k^s-1}{k-1} - s)}{k-1} - E = 2 \cdot f(k, s+2) - (s+2)k - 2 + \frac{k(\frac{k^s-1}{k-1} - s)}{k-1} - E$$

Thus, we have that

$$\text{ratio}(\Delta\mathfrak{D}) = \frac{f(k,p+1) - \mathfrak{D}(p,k)}{f(k,p+1)} = \frac{2 \cdot f(k,s+2) - (s+2)k - 2 + \frac{k(\frac{k^s-1}{k-1} - s)}{k-1} - E}{f(k,p+1)}$$

$$\approx \frac{2f(k,s+2)}{f(k,p+1)} \approx \frac{2}{k^{p-s-1}} \approx \frac{2}{k^{\frac{p}{2}}}.$$

Remark 5. *We can do the ratio estimation in another way. From Remark 4, we get* $\Delta\mathfrak{D} \geq \sum_{i=1}^{s} k^{p-s-1} = kf(k, p-s-1)$. *Therefore,* $\text{ratio}(\Delta\mathfrak{D}) \approx \frac{kf(k,p-s-1)}{f(k,p+1)} \approx \frac{1}{k^{s+1}} \approx \frac{1}{k^{\frac{p}{2}}}$. *There is only a difference of the factor of 2 from the above estimation.*

6.6 Bound Reduction

We obtain the final bound formula for Optimization 2 by substituting $\mathfrak{D}(p,k)$ for q in the original bound formula $\mathfrak{B}_0(p,k,1) \le q^r$. In other words, we have

$$\mathfrak{B}_2(p,k,1) \le \mathfrak{D}(p,k)^{f(k,p)}.$$

More specifically, we have the following for some common player counts:

- 3 players: $\mathfrak{B}(3,k,1) = (k^3+k+1)^{k^2+k+1}$.
- 4 players: $\mathfrak{B}(4,k,1) = (k^4+k^3-k^2+k+1)^{k^3+k^2+k+1}$.
- 5 players: $\mathfrak{B}(5,k,1) = (k^5+k^4+k+1)^{k^4+k^3+k^2+k+1}$.

We now finish by estimating the ratio between the new bound and the original bound, using $(1-\frac{1}{x})^x \approx \frac{1}{e}$ to finish:

$$\frac{\mathfrak{B}_2(p,k,1)}{\mathfrak{B}_0(p,k,1)} = \left(\frac{\mathfrak{D}(p,k)}{f(k,p+1)}\right)^{f(k,p)} = (1-\text{ratio}(\Delta\mathfrak{D}))^{f(k,p)}$$

$$\approx \left(1-\frac{2}{k^{\frac{p}{2}}}\right)^{k^p} = \left((1-\frac{2}{k^{\frac{p}{2}}})^{\frac{k^{\frac{p}{2}}}{2}}\right)^{2k^{\frac{p}{2}}} \approx \frac{1}{e^{2k^{\frac{p}{2}}}}.$$

The ratio between the new bound $\mathfrak{B}_2(p,k,1)$ with Optimization 2 and the original bound $\mathfrak{B}_0(p,k,1)$ is:

$$\mathfrak{B}_2(p,k,1) \approx \frac{\mathfrak{B}_0(p,k,1)}{e^{2k^{\frac{p}{2}}}} \tag{11}$$

Note that this reduction becomes more significant as p and k get larger.

For small values of p and k, Table 3 shows the digit range reduction between Optimization 2 and the original, while Table 4 compares the original bound and the reduced bound.

Table 3: The digit range reduction between Optimization 2 and without optimization

new, old	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$
$p=2$	$5,7$	$10,13$	$17,21$	$26,31$	$37,43$	$50,57$
$p=3$	$11,15$	$31,40$	$69,85$	$131,156$	$223,259$	$351,400$
$p=4$	$23,31$	$103,121$	$309,341$	$731,781$	$1483,1555$	$2703,2801$
$p=5$	$49,63$	$325,364$	$1281,1365$	$3751,3906$	$9073,9331$	$19209,19608$
$p=6$	$103,127$	$1021,1093$	$5301,5461$	$19231,19531$	$55483,55987$	$136473,137257$
$p=7$	$217,255$	$3142,3280$	$21473,21845$	$96826,97656$	$334297,335923$	$957902,960800$

Table 4: Comparison between $\mathfrak{B}_1(p,k,1)$ and $\mathfrak{B}_2(p,k,1)$

old, new	$k=2$	$k=3$	$k=4$	$k=5$
$p=2$	125, 125	10000, 10000	1419857, 1419857	30E7, 30E7
$p=3$	62E6, 19E6	24E19, 24E18	11E39, 41E37	35E66, 43E64
$p=4$	86E20, 26E19	75E81, 32E79	70E213, 44E210	65E449, 59E445
$p=5$	22E54, 24E51	28E308, 28E302	44E1067, 55E1057	51E2803, 11E2789
$p=6$	12E131, 21E126	41E1104, 56E1094	87E5099, 15E5083	13E16758, 56E16732
$p=7$	15E304, 17E296	26E3841, 12E3820	58E23695, 30E23651	23E97452, 91E97372

6.7 Optimization 2 Algorithm

Recall Property 1 and Property 2. In our algorithm, we divide $A_i(x)$ (where $1 \le i \le p-1$) into multiple equal-length subarrays s of length $|A_j(z)|$. We have $A_j(z) \succ A_i(x)$ if

$$A_j(z) = s \quad \text{for at least one such s.} \tag{12}$$

The ancestor-bookkeeping algorithm is not changed with this optimization. With this in mind, pseudocode for the Optimization 2 algorithm is provided in Algorithm 6 and Algorithm 7, located in Appendix A.

7 Optimization 3: Reordering Ancestors

While Optimization 2 lowers the power base, further reductions are achieved by eliminating redundancy in ancestor digits.

7.1 The $(3,2,1)$ Game

Note that 4321254 is essentially equivalent to 1234254 or 1234524 when serving as input to next player. In other words, the order of digits within an ancestor level do not matter. Thus, we can simply sort them in a fixed order (e.g. non-decreasing). This allows us to merge a lot of equivalent output numbers into a single number, which reduces the bound due to Lemma 2. By doing this, we reduce $\mathfrak{B}_2(3,2,1)$ to $\mathfrak{B}_3(3,2,1) = 726,726$.

We now present several examples in Figure 14 to illustrate the new rule. Note that we are in base-11 and use the digit A to represent the value 10.

- Example 1: Inputs are $x = 0123456$ and $y = 789A564$, and the output is $z = 4556464$. The digits at each ancestor level of z must appear in a fixed, sorted order.
- Example 2: Inputs are $x = 0123554$ and $y = 789A466$, and the output is again $z = 4556464$.

Remark 6. *Example 1 and Example 2 produce the same output due to the reordering of ancestor digits.*

- Example 3: Inputs are $x = 0123453$ and $y = 789A563$, and the output is $z = 4556334$. Since $A_1(z) = 33$ cannot convert to either $A_2(x) = 0123$ or $A_2(y) = 789A$, the last digit $m(z)$ does not need to exclude any digits from $A_1(x)$ or $A_1(y)$. It only needs to avoid the 8 digits in

$A_2(x)$ and $A_2(y)$, so it takes the smallest possible value 4. The output is distinct from all its ancestors.

- Example 4: Inputs are $x = 0123453$ and $y = 789A562$, and the output is $z = 4556236$. Here $A_1(z) = 23$ can convert to $A_2(x) = 0123$, so $m(z)$ must also exclude the digits in $A_1(x) = 45$. It avoids the 8 digits in $A_2(x)$ and $A_2(y)$ plus 2 digits in $A_1(x)$, taking the smallest valid value 6. The output remains distinct from all its ancestors.

7.2 Bound Reduction

We will now calculate $\mathfrak{B}_3(3, 2, 1)$. For each ancestor level i, we are forming sorted arrays of length k^i using digits from a fixed pool of size $\mathfrak{D}(p, k)$. Repetition is allowed and the order of digits does not matter (since each ancestor level is sorted), so this is a classic stars and bars problem (https://en.wikipedia.org/wiki/Stars_and_bars_(combinatorics)). We are choosing k^i elements from $\mathfrak{D}(p, k)$ digits with repetition allowed and order irrelevant. The number of such combinations at each ancestor level is given by the standard formula:

$$\binom{\mathfrak{D}(p,k) + k^i - 1}{k^i}.$$

For the $(3, 2, 1)$ game, we have

$$\mathfrak{B}_3(3, 2, 1) \le \binom{11 + 4 - 1}{4} \cdot \binom{11 + 2 - 1}{2} \cdot 11 = 726726.$$

In general, we have

$$\mathfrak{B}_3(p, k, 1) \le \prod_{i=0}^{p-1} \binom{\mathfrak{D}(p,k) + k^i - 1}{k^i}.$$

For small values of p and k, Table 5 shows a comparison between $\mathfrak{B}_2(p, k, 1)$ and $\mathfrak{B}_3(p, k, 1)$.

Table 5: Comparison between $\mathfrak{B}_2(p, k, 1)$ and $\mathfrak{B}_3(p, k, 1)$

old, new	$k = 2$	$k = 3$	$k = 4$	$k = 5$
$p = 2$	125, 75	10000, 2200	1419857, 82365	30E7, 3705156
$p = 3$	19E6, 726726	24E18, 35E12	41E37, 45E23	43E64, 21E38
$p = 4$	26E19, 55E13	32E79, 46E46	44E210, 47E109	59E445, 59E213
$p = 5$	24E51, 11E33	28E302, 74E151	55E1057, 15E458	11E2789, 13E1097
$p = 6$	21E126, 62E73	56E1094, 87E476	15E5083, 77E1875	56E16732, 17E5561
$p = 7$	17E296, 10E157	12E3820, 53E1455	30E23651, 41E7552	91E97372, 24E27890

7.3 The General Algorithm

As our ancestor-bookkeeping algorithm changes with this optimization, we need to update the check criteria of ancestor group convertibility. If $A_j(z) \succ A_i(x)$, then we need $A_j(z) \subseteq A_i(x)$, as before. Additionally, the number of occurrences of each digit in $A_j(z)$ must be equal to or less than the number of occurrences of the same digit in $A_i(x)$. In other words, if $A_j(z) \succ A_i(x)$, then

$$A_j(z) \subseteq A_i(x) \quad \text{and} \quad \forall d,\ \#_d(A_j(z)) \le \#_d(A_i(x)), \tag{13}$$

where $\#_d(S)$ denotes the number of occurrences of digit d in sequence S. When calling the last digit selection Algorithm 6 of Optimization 2 for this optimization, we will use this updated definition of $A_j(z) \succ A_i(x)$. When doing ancestor-bookkeeping, we replace each ancestor group $A_j(z)$ with its sorted sequence. Pseudocode for the Optimization 3 algorithm is provided in Algorithm 8 located in Appendix A.

8 Optimization 4: Merging Equivalent Ancestors

This optimization further reduces the bound by merging equivalent ancestors in the outputs.

8.1 The $(3, 2, 1)$ Game

Note that 1112345 is essentially equivalent to 1222345, since their grandparents have the same set of distinct digits, namely $\{1, 2\}$. Thus, instead of simply sorting each ancestor level, we can represent each ancestor level as a set of digits. Again, this allows us to merge a lot of equivalent output numbers into a single number, which reduces the bound due to Lemma 2. By doing this, we reduce $\mathfrak{B}_3(3, 2, 1)$ to $\mathfrak{B}_4(3, 2, 1) = 407,286$.

We now present several examples in Figure 15 to illustrate the new rule. Note that we are in base-11 and use the digit A to represent the value 10.

- Example 1: Inputs are $x = 0123456$ and $y = 789A564$, and the output is $z = 4456464$. The digits at each ancestor level of the output z must merge to its equivalent ancestor group, so the previous 4556464 number from Optimization 3 is never an output now.
- Example 2: Inputs are $x = 0123564$ and $y = 789A466$, and the output is $z = 4456464$. This Example 2 outputs the same number z as Example 1, due to the reordering of the ancestor digits and merging of equivalent ancestor groups.
- Example 3: Inputs are $x = 0123453$ and $y = 789A563$, and the output is $z = 4456334$. Because $A_1(z) = 33$ cannot convert to either $A_2(x) = 0123$ or $A_2(y) = 789A$, the last digit $m(z)$ does not need to exclude any digits in $A_1(x)$ or $A_1(y)$. So $m(z)$ just needs to exclude the 8 digits in $A_2(x)$ and $A_2(y)$ by taking the smallest value 4. The output is a distinct number from all its ancestors.
- Example 4: Inputs are $x = 0123453$ and $y = 789A562$, and the output is $z = 4456236$. Because $A_1(z) = 23$ can convert to $A_2(x) = 0123$, the last digit $m(z)$ needs to exclude the digits in $A_1(x) = 45$. So $m(z)$ needs to exclude the 8 digits in $A_2(x)$ and $A_2(y)$ and the 2 digits in $A_1(x) = 45$ by taking the smallest value 6. The output is still a distinct number from all its ancestors.

8.2 Bound Reduction

We will now calculate $\mathfrak{B}_4(3, 2, 1)$. Note that each ancestor level, the number of unique digit sets of size 1 through n is given by

$$\sum_{i=1}^{n} \binom{\mathfrak{D}(p,k)}{i}.$$

Thus, for $(3, 2, 1)$, we get

$$\mathfrak{B}_4(3, 2, 1) \leq \left[\binom{11}{1} + \binom{11}{2} + \binom{11}{3} + \binom{11}{4}\right] \cdot \left[\binom{11}{1} + \binom{11}{2}\right] \cdot 11 = 407286.$$

In general, we have

$$\mathfrak{B}_4(p,k,1) \leq \prod_{i=0}^{p-1} \sum_{j=1}^{k^i} \binom{\mathfrak{D}(p,k)}{j}.$$

For small values of p and k, Table 6 shows a comparison between $\mathfrak{B}_3(p,k,1)$ and $\mathfrak{B}_4(p,k,1)$.

Table 6: Comparison between $\mathfrak{B}_3(p,k,1)$ and $\mathfrak{B}_4(p,k,1)$

old, new	$k=2$	$k=3$	$k=4$	$k=5$
$p=2$	75, 75	2200, 1750	82365, 54621	3705156, 2175706
$p=3$	726726, 407286	35E12, 48E11	45E23, 17E22	21E38, 25E36
$p=4$	55E13, 60E12	46E46, 38E43	47E109, 59E103	59E213, 18E204
$p=5$	11E33, 46E30	74E151, 16E142	15E458, 28E434	13E1097, 68E1049
$p=6$	62E73, 52E68	87E476, 79E448	77E1875, 11E1784	17E5561, 64E5330
$p=7$	10E157, 47E146	53E1455, 33E1372	41E7552, 16E7188	24E27890, 12E26744

8.3 The General Algorithm

As with the previous optimization, we update the convertibility criteria for $A_j(z) \succ A_i(x)$. In this case, we require

$$A_j(z) \subseteq A_i(x) \quad \text{and} \quad |A_i(x)| - |A_j(z)| \leq k^i - k^j. \tag{14}$$

This reflects the merging of sorted digits into sets of unique digits. When calling Algorithm 6, we apply this modified ancestor group convertibility check.

For ancestor-bookkeeping, the merging of equivalent ancestors is done by:

1. Getting the set of unique digits U in an ancestor group $A_i(z)$
2. Sorting U
3. Getting the minimal digit d_{min} from U
4. Adding the necessary number $(k^i - len(U))$ of occurrences of d_{min} before U to form the new ancestor group

Remark 7. *This new ancestor group is essentially the minimal value among all of its equivalent ancestor groups.*

Pseudocode for the Optimization 4 algorithm is provided in Algorithm 9 located in Appendix A.

9 Optimization 5: Merging Neighboring Ancestors

Next, we can further reduce the bound by merging "neighboring" outputs to a single output.

9.1 The $(3, 2, 1)$ Game

Note that $\mathfrak{D}(3, 2) = 11$, so we are in base 11. We use A to represent the digit 10. We now present several examples in Figure 16 to illustrate the new rule.

Consider when the two inputs are 0123456 and A987456 are the inputs. Then, the output will be 4445664 according to our last optimization (Algorithm 9). However, if we instead generate a new output of 0145064, then 4445664 will never be a valid output, thus reducing the number of valid outputs. Again, this reduces the final bound due to Lemma 2.

Remark 8. *The key idea of this optimization is to combine a set of "neighboring" ancestor groups like* $\{3345, 3445, 3455\}$ *into a single output like* $\{0345\}$.

For completeness, we use an example where inputs are 0126346 and A987346. Focusing on $A_2(z)$, we have 9 possible ancestor groups that turn into 1 ancestor group:

$$\{3334, 3344, 3444, 0034, 0334, 0344, 1134, 1334, 1344\} \Rightarrow \{0134\}.$$

Note that this optimization can also lead to changes in the last digit. For example, when the two inputs are 0126346 and A987346, the output would have been 3334663 according to Algorithm 9. However, with the new Algorithm 10, our output would instead be 0134065, and the last digit $m(z)$ is 5 instead of 3. This is because when we apply Optimization 5, our merging of neighboring ancestors makes $A_1(z)$ goes from 66 to 06. However, this results in $A_1(z) \succ A_0(x)$, as we have $\overline{06} \succ \overline{0126}$. Before, we had $A_1(z) \not\succ A_0(x)$, as $\overline{66} \not\succ \overline{0126}$. Thus, recalling that we need $A_0(z) \not\succ A_1(x)$ for this case (Equation 2), we need to exclude $A_1(x) = 34$ as well when selecting the last digit $A(z)$. This leads to a choice of $A_0(z) = 5$. Despite the introduction of "fake" ancestor digits such as 0 and 1, our output still satisfies the cycle avoidance rules as the new output was an existing output number from Section 8.

With our optimization, we have that

$$\mathfrak{B}_5(3, 2, 1) \leq \binom{11}{4}\binom{11}{2} \cdot 11 = 199650.$$

9.2 Bound Reduction

In general, we can combine

$$\binom{\mathfrak{D}(p,k)}{1}, \binom{\mathfrak{D}(p,k)}{2}, \ldots, \binom{\mathfrak{D}(p,k)}{k^j - 1} \Rightarrow \binom{\mathfrak{D}(p,k)}{k^j}.$$

Thus, we reduce

$$\sum_{j=1}^{k^j} \binom{\mathfrak{D}(p,k)}{j} \Rightarrow \binom{\mathfrak{D}(p,k)}{k^j}.$$

Thus, our final bound with Optimization 5 is

$$\mathfrak{B}_5(p, k, 1) \leq \prod_{i=0}^{p-1} \binom{\mathfrak{D}(p,k)}{k^i}.$$

For small values of p and k, Table 7 shows a comparison between $\mathfrak{B}_4(p, k, 1)$ and $\mathfrak{B}_5(p, k, 1)$.

Table 7: Comparison between $\mathfrak{B}_4(p,k,1)$ and $\mathfrak{B}_5(p,k,1)$

old, new	$k=2$	$k=3$	$k=4$	$k=5$
$p=2$	75, 50	1750, 1200	54621, 40460	2175706, 1710280
$p=3$	407286, 199650	48E11, 28E11	17E22, 11E22	25E36, 18E36
$p=4$	60E12, 25E12	38E43, 22E43	59E103, 40E103	18E204, 14E204
$p=5$	46E30, 18E30	16E142, 96E141	28E434, 19E434	68E1049, 52E1049
$p=6$	52E68, 21E68	79E448, 47E448	11E1784, 78E1783	64E5330, 49E5330

9.3 Approximation

Note that $\binom{n}{k} \leq \frac{n^k}{k!}$. Furthermore, recall the previously derived bound from Optimization 2 and Equation 11. Using this, we can derive that:

$$\mathfrak{B}_5(p,k,1) \leq \prod_{i=0}^{p-1} \binom{\mathfrak{D}(p,k)}{k^i} \leq \prod_{i=0}^{p-1} \frac{\mathfrak{D}(p,k)^i}{(k^i)!} = \frac{\mathfrak{D}(p,k)^{f(k,p)}}{\prod_{i=0}^{p-1}(k^i)!} = \frac{\mathfrak{B}_2(p,k,1)}{\prod_{i=0}^{p-1}(k^i)!} \approx \frac{\mathfrak{B}_0(p,k,1)}{e^{2k^{\frac{p}{2}}} \prod_{i=0}^{p-1}(k^i)!}.$$

The below figure shows the reduction factor $f_5(p,k) = \prod_{i=0}^{p-1}(k^i)!$ for some smaller p,k values:

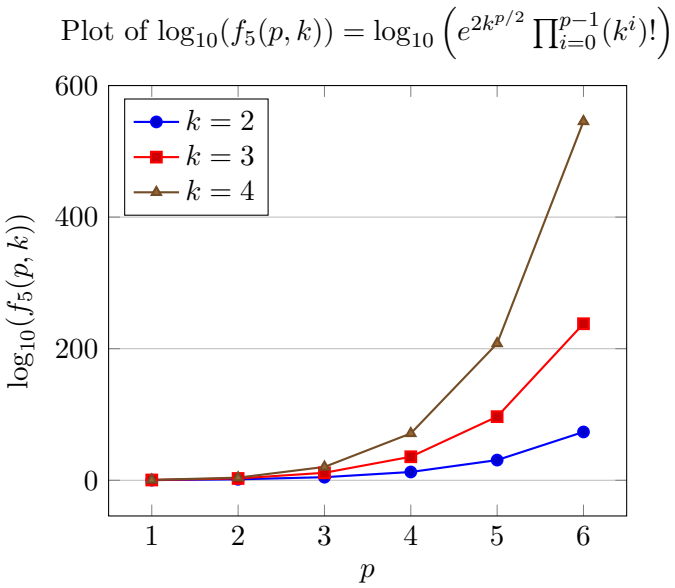


Figure 8: Log-scale plot of $f_5(p,k)$ for $k=2$, $k=3$, and $k=4$.

9.4 The General Algorithm

As with the previous optimization, we update the convertibility criteria for $A_j(z) \succ A_i(x)$. In this case, we require

$$A_j(z) \subseteq A_i(x) \tag{15}$$

Compared to Equation 14, we are getting rid of $|A_i(x)| - |A_j(z)| \leq k^i - k^j$. This condition is no longer needed as we now guarantee that all inputs and outputs have distinct digits at each ancestor level. When calling Algorithm 6, we apply this modified ancestor group convertibility check.

For ancestor-bookkeeping, the merging of neighboring ancestors is done by:

1. Getting the set of unique digits U in an ancestor group $A_i(z)$
2. Getting the set of unique digits W not in U
3. Getting the set F of $(k^i - len(U))$ lowest digits from W
4. New ancestor group is sorted$(U + F)$

Remark 9. *This new ancestor group is essentially the minimal value among all its neighboring ancestor groups.*

Pseudocode for the Optimization 5 algorithm is provided in Algorithm 10 located in Appendix A.

10 Optimization 6: Pruning Unused Outputs

Our final optimization is very minor, simply reducing the number of possible outputs by considering edge cases. Again, this reduces the bound due to Lemma 2.

10.1 The $(3, 2, 1)$ Game

Recall that we are in base-11 and A represents the digit 10. We observe that some numbers like $\{1234099, 12341AA, 12349A9, 12349AA\}$ can never be the output of any player. In fact we can prove the below claim:

Claim 1. *For a valid output z of the $(3, 2, 1)$ game, if $m(z) \geq 9$, then $m(z) \notin A_1(z)$.*

Proof. In order for $m(z) \geq 9$, it is necessary for $A_1(x)$ to be in the restricted digits (that is, $m(z) \not\succ A_1(x)$), because the maximum number of unique digits in $A_2(x)$ and $A_2(y)$ is 8, which can only make $m(z) = 8$ at most. Recall Equation 2. This requires $A_1(z) \succ A_2(x)$. However, since we always need $m(z) \not\succ A_2(x)$, if $A_1(z) \succ A_2(x)$, then we must have $m(z) \not\succ A_1(z)$, which is in fact $m(z) \notin A_1(z)$. □

An equivalent claim of the above (its contrapositive) is as below:

Claim 2. *For a valid output z of the $(3, 2, 1)$ game, if $m(z) \in A_1(z)$, then $m(z) < 9$.*

Proof. Recall Equation 2 for the requirements of a winning strategy for the $(3, 2, 1)$ game. Since $m(z) \notin A_2(x)$ and $m(z) \in A_1(z)$, we must have $A_1(z) \not\subseteq A_2(x)$ (or $A_1(z) \not\succ A_2(x)$), which means no 2-cycles. From our construction method, there will be no contribution of $A_1(x)$ or $A_1(y)$ when selecting the last digit $m(z)$. Since the maximum number of unique digits in $A_2(x)$ and $A_2(y)$ is 8, which can only make $m(z) = 8$ at most. Therefore, $m(z) < 9$. □

We proceed with casework to calculate how many of these invalid outputs there are. We have

$$A_1(z) \in \begin{cases} \{0A, 1A, \ldots, 8A, 9A\} & \text{if } m(z) = A \\ \{09, 19, \ldots, 89, 9A\} & \text{if } m(z) = 9 \end{cases} \quad \text{(10 possibilities in each case).}$$

Thus, the total number of invalid inputs is

$$20 \cdot \binom{11}{4} = 6600,$$

leaving us with a bound of $199650 - 6600 = 193050$. In other words, we have:

$$\mathfrak{B}_6(3, 2, 1) \leq 193050.$$

10.2 The $(p, k, 1)$ Game

In general, we have the below claim:

Claim 3. *For a valid output z of the $(p, k, 1)$ game, if $m(z) \in A_1(z)$, then $m(z) < \mathfrak{D}(p, k) - \mathfrak{C}(p-1) = \mathfrak{D}(p, k) - (k^{p-1} - k^2 + k)$.*

Proof. Recall Equation 3 for the requirements of a winning strategy for the $(p, k, 1)$ game. Since $m(z) \notin A_p(x)$ and $m(z) \in A_1(z)$, we must have $A_1(z) \not\subseteq A_p(x)$ (or $A_1(z) \not\succ A_p(x)$), which means no $(p-1)$-cycles. From our construction method, there will be no contribution of $A_{p-1}(x)$ to the restricted digits when selecting the last digit $m(z)$. Since the maximum number of unique digits in the rest of L-cycle avoidance is $\mathfrak{D}(p, k) - \mathfrak{C}(p-1) - 1$, which can only make $m(z) = \mathfrak{D}(p, k) - \mathfrak{C}(p-1) - 1$ at most. Therefore, $m(z) < \mathfrak{D}(p, k) - \mathfrak{C}(p-1) = \mathfrak{D}(p, k) - (k^{p-1} - k^2 + k)$. □

Claim 4. *For a valid output z of the $(p, k, 1)$ game, if $m(z) \geq \mathfrak{D}(p, k) - \mathfrak{C}(p-1) = \mathfrak{D}(p, k) - (k^{p-1} - k^2 + k)$, then $m(z) \notin A_1(z)$.*

Proof. This is simply the contrapositive of the above claim. □

In general, we have the below more general claim:

Claim 5. *For a valid output z of the $(p, k, 1)$ game, if $m(z) \in \bigcap_{i=1}^{j} A_i(z)$, where $1 \leq j < \lceil \frac{p}{2} \rceil$, then $m(z) < \mathfrak{D}(p, k) - \sum_{i=1}^{j} \mathfrak{C}(p-i) = \mathfrak{D}(p, k) - \sum_{i=1}^{j} (k^{p-i} - k(k^i - 1) - kf(k, i-1))$.*

Proof. Recall Equation 3 for the requirements of a winning strategy for the $(p, k, 1)$ game. Since $m(z) \notin A_p(x)$ and $m(z) \in \bigcap_{i=1}^{j} A_i(z)$, we must have $\forall 1 \leq i \leq j, A_i(z) \not\subseteq A_p(x)$ (or $A_i(z) \not\succ A_p(x)$), which means no $(p-i)$-cycles. From our construction method, there will be no contribution of $A_{p-i}(x)$ to the restricted digits when selecting the last digit $m(z)$. Since the maximum number of unique digits in the rest of L-cycle avoidance is $\mathfrak{D}(p, k) - \sum_{i=1}^{j} \mathfrak{C}(p-i)$, which can only make $m(z) = \mathfrak{D}(p, k) - \sum_{i=1}^{j} \mathfrak{C}(p-i)$ at most. Therefore, $m(z) < \mathfrak{D}(p, k) - \sum_{i=1}^{j} \mathfrak{C}(p-i) = \mathfrak{D}(p, k) - \sum_{i=1}^{j} (k^{p-i} - k(k^i - 1) - kf(k, i-1))$. □

Based on the above Claim 3 – 5, we know there are some numbers that are invalid or unused outputs, that is, the reduction from Optimization 6 is generally nonzero. However, it is difficult to find a closed-form formula for Optimization 6 for general p and k. Even if it is found, it is expected to be a complex expression using the Inclusion-Exclusion Principle (https://en.wikipedia.org/wiki/Inclusion%E2%80%93exclusion_principle). Thus, we provide closed-form estimates for smaller values of $p = 2, 3, 4, 5$ instead.

10.3 Case-by-Case Reductions

1. For $\mathfrak{B}(2,k,1)$ the reduction is zero, as we cannot apply Optimization 6.
2. For $\mathfrak{B}(3,k,1)$ only $A_1(z)m(z)$ contributes to the reduction, yielding
$$\binom{\mathfrak{D}(p,k)-1}{k-1}\cdot k\cdot\binom{\mathfrak{D}(p,k)}{k^2}.$$
3. For $\mathfrak{B}(4,k,1)$ likewise only $A_1(z)m(z)$ contributes to the reduction, giving
$$\binom{\mathfrak{D}(p,k)-1}{k-1}\cdot(k^3-k^2+k)\cdot\binom{\mathfrak{D}(p,k)}{k^3}\cdot\binom{\mathfrak{D}(p,k)}{k^2}.$$
4. For $\mathfrak{B}(5,k,1)$ both $A_2(z)m(z)$ and $A_1(z)m(z)$ contribute to the reduction. This leads to a reduction of at least the below part due to $A_1(z)m(z)$ contribution
$$\binom{\mathfrak{D}(p,k)-1}{k-1}\cdot(k^4-k^2+k)\cdot\binom{\mathfrak{D}(p,k)}{k^4}\cdot\binom{\mathfrak{D}(p,k)}{k^3}\cdot\binom{\mathfrak{D}(p,k)}{k^2}.$$
5. For $\mathfrak{B}(p,k,1)$ with $p>5$, each of $A_{\lceil\frac{p}{2}\rceil-1}(z)m(z),\ldots,A_1(z)m(z)$ contributes to the reduction. This leads to a reduction of at least the below part due to $A_1(z)m(z)$ contribution
$$\binom{\mathfrak{D}(p,k)-1}{k-1}\cdot(k^{p-1}-k^2+k)\cdot\prod_{i=2}^{p-1}\binom{\mathfrak{D}(p,k)}{k^i}.$$

As this optimization yields only a marginal improvement to the bound, we omit its algorithm description from Appendix A.

11 Combined Optimizations

We now provide a brief summary of our results after applying all 6 optimizations. In general, we get:
$$\mathfrak{B}_5(p,k,1)\le\prod_{i=0}^{p-1}\binom{\mathfrak{D}(p,k)}{k^i}.$$
More precisely, we have:
$$\mathfrak{B}_6(p,k,1)\le\prod_{i=0}^{p-1}\binom{\mathfrak{D}(p,k)}{k^i}-\binom{\mathfrak{D}(p,k)-1}{k-1}\cdot(k^{p-1}-k^2+k)\cdot\prod_{i=2}^{p-1}\binom{\mathfrak{D}(p,k)}{k^i}.$$

Table 8 shows the reduction to $\mathfrak{B}(3,2,1)$ after applying each optimization.

Table 8: Reduction of the oblivious $\mathfrak{B}(3,2,1)$ bound

$\mathfrak{B}(3,2,1)$	original	opt1	opt2	opt3	opt4	opt5	opt6
bound	15^7	13^7	11^7	726726	407286	199650	193050
reduction	0	108,110,858	43,261,346	18,760,445	319,440	207,636	6,600
ratio	100%	36.7%	11.4%	0.425%	0.238%	0.117%	0.113%

Thus, we reduce the original bound of $15^7 \approx 171$ million to just 193,050, which is less than 0.113% of the original.

The following tables show similar reductions for other parameter settings. As these parameters are relatively large, we report the base-10 logarithm of the reduction ratio.

Table 9: Reduction of the oblivious $\mathfrak{B}(3,3,1)$ bound

$\mathfrak{B}(3,3,1)$	original	opt1	opt2	opt3	opt4	opt5	opt6
bound	40^{13}	37^{13}	31^{13}	3.5×10^{13}	4.8×10^{12}	2.8×10^{12}	2.7×10^{12}
log(ratio)	0	1.014	3.314	16.745	18.737	19.292	19.301

Table 10: Reduction of the oblivious $\mathfrak{B}(3,4,1)$ bound

$\mathfrak{B}(3,4,1)$	original	opt1	opt2	opt3	opt4	opt5	opt6
bound	85^{21}	81^{21}	69^{21}	4.5×10^{24}	1.7×10^{23}	1.1×10^{23}	1.1×10^{23}
log(ratio)	0	1.012	4.379	36.522	39.804	40.204	40.207

Table 11: Reduction of the oblivious $\mathfrak{B}(4,2,1)$ bound

$\mathfrak{B}(4,2,1)$	original	opt1	opt2	opt3	opt4	opt5	opt6
bound	31^{15}	29^{15}	23^{15}	5.5×10^{14}	6.0×10^{13}	2.5×10^{13}	2.4×10^{13}
log(ratio)	0	1.000	4.477	17.559	19.768	20.649	20.672

Table 12: Reduction of the oblivious $\mathfrak{B}(4,3,1)$ bound

$\mathfrak{B}(4,3,1)$	original	opt1	opt2	opt3	opt4	opt5	opt6
bound	121^{40}	118^{40}	103^{40}	4.6×10^{47}	3.8×10^{44}	2.2×10^{44}	2.1×10^{44}
log(ratio)	0	1.004	6.442	82.076	89.180	89.727	89.733

12 Multiple-Attempt Variants

For multiple-attempt (p,k,m) novelty games, we prove that $\mathfrak{B}(p,k,m) < \mathfrak{B}(p,k,1)$ for ancestor-bookkeeping strategies. This is to answer the open question 3 in the paper [LS25]:

Question 3. Can we find a better bound for $\mathfrak{B}(p,k,k-1)$ than $\mathfrak{B}(p,k,1)$?

The answer is YES for oblivious strategies.

12.1 Naive Bound Improvement

Let us take the simpler $(3,3,2)$ game as the example. For single-attempt variant $(3,3,1)$ game, we need a power base $q = \frac{3^{3+1}-1}{3-1} = 40$, and an exponent $\frac{3^3-1}{3-1} = 13$, so the bound is $\mathfrak{B}(3,3,1) = 40^{13}$.

When each player can output two numbers, then a player can output two different numbers with the same last digit. This means that even if the last digit of the output is the same as one ancestor, as long as this ancestor digit occurs only once in all ancestors, then we can still guarantee novelty. If we reduce the number of digits from 40 to 20, then out of the 39 digits of all ancestors, there must exist one digit whose occurrence is 1 in the worst case. Then we pick this digit and it still guarantees novelty.

Note that in the extreme case of all the ancestor digits of the output number are same, we can replace one of the ancestor digits with a different digit, to allow output $m < k$ different numbers for a player. This introduces some "fake" information regarding one ancestor's last digit, however, this only makes the restriction "stricter", thus still guaranteeing the novelty of the output numbers.

After halving the base from 40 to 20, the oblivious bound is now $\mathfrak{B}(3,3,2) = 20^{13}$.

In general, if we reduce the number of required unique digits from q to $\lceil \frac{q}{m} \rceil$, then out of the $q-1$ digits of all ancestors, there must exist one digit whose occurrence is less than m in the worst case. Then we pick this digit as the last digit of all m output numbers. And in the extreme case of all ancestor digits being the same, we can always introduce one "fake" new digit to make it possible.

Therefore, in general, the power base can be reduced by a factor of $\frac{1}{m}$, and the oblivious bound for multiple-attempt variants:

$$\mathfrak{B}(p,k,m) \leq (\lceil \frac{f(k,p+1)}{m} \rceil)^{f(k,p)}$$

This is significantly smaller than $\mathfrak{B}(p,k,1)$, the reduction factor is about $m^{f(k,p)} \approx m^{k^{p-1}}$.

12.2 Optimization 1 of Input Avoidance and Optimization 2 of Cycle Avoidance

The previous result is just a naive optimization, based on the original $\mathfrak{B}_0(p,k,1)$ bound. We can do more optimizations similarly as what we have done for $\mathfrak{B}(p,k,1)$ optimizations.

Optimization 1 of input avoidance and Optimization 2 of cycle avoidance to reduce the power base can still be applied for the multiple attempt variants. We also need to allow at least m variations for the same last digit. This can be implemented by allowing the first m reorderings of the ancestor digits of the output with the same last digit. So the optimized bounds become:

$$\mathfrak{B}_1(p,k,m) \leq (\lceil \frac{q-k}{m} \rceil)^{f(k,p)}$$

$$\mathfrak{B}_2(p,k,m) \leq (\lceil \frac{\mathfrak{D}(p,k)}{m} \rceil)^{f(k,p)}$$

12.3 Optimization 3 of Reordering Ancestors

Optimization 3 of reordering ancestors can also be applied since it does not change the number of occurrences of a digit of ancestors. Optimization 4 of merging equivalent ancestors cannot be applied for the multiple attempt variants, because it changes the number of occurrences of a digit of ancestors.

At the end, we also need to allow at least m variations for the same last digit. This can be implemented by allowing the first m reorderings in the increasing order. This adds an additional factor of m into the formula. So the optimized bound becomes:

$$\mathfrak{B}_3(p,k,m) \le m \prod_{i=0}^{p-1} \binom{\lceil \frac{\mathfrak{D}(p,k)}{m} \rceil + k^i - 1}{k^i}$$

12.4 Optimization 5 of Merging to Stricter Ancestors

Optimization 5 of merging to stricter-ancestors can be applied with some slightly adjustment: we set the repeat count of a digit to a cap of m, instead of 1 for the $(p,k,1)$ case. That is, if a digit has more than m occurrences in the ancestor group A_i, then we replace the extra occurrences with "fake" digits, to guarantee that no single digit exceeds m occurrences. This is always doable because the total number of occurrences of the forbidden ancestor digits is $\mathfrak{D}(p,k)-1$ and we have $\lceil \frac{\mathfrak{D}(p,k)}{m} \rceil$ digits to choose from.

This is the bounded integer composition problem, so the optimized bound becomes:

$$\mathfrak{B}_5(p,k,m) \le m \prod_{i=0}^{p-1} \sum_{j=0}^{\lfloor \frac{k^i}{m+1} \rfloor} (-1)^j \binom{\lceil \frac{\mathfrak{D}(p,k)}{m} \rceil}{j} \binom{k^i - j(m+1) + \lceil \frac{\mathfrak{D}(p,k)}{m} \rceil - 1}{\lceil \frac{\mathfrak{D}(p,k)}{m} \rceil - 1}$$

13 Conclusion and Future Work

13.1 Summary of Contributions

We established several theorems concerning novelty games and used them to optimize the winning strategy, yielding significant improvements to the best known oblivious bounds. In particular, we improved the bound $\mathfrak{B}(p,k,1)$ from the original q^r, with our new bound being

$$\mathfrak{B}(p,k,1) \le \prod_{i=0}^{p-1} \binom{\mathfrak{D}(p,k)}{k^i}.$$

13.2 Future Directions

One possible direction is to find a revised algorithm and a subset of valid outputs and see if their descendants are the same as themselves, thus reducing the upper bound. Or find a better algorithm to beat the existing upper bound for some smaller special p,k values.

Another direction is to find a better lower bound on N.

We will use the same $\mathfrak{B}(p,k,1)$ symbol to denote the lower bound. The only known lower bound for general novelty games is the trivial estimate from [LS25]:

$$\mathfrak{B}(p,k,1) \ge pk+1.$$

Establishing a stricter lower bound would therefore represent meaningful progress in the theory of novelty games.

13.2.1 Combinatorial Entropy Encoding

In information theory, entropy encoding refers to any lossless compression method that seeks to approach the lower bound on average code length determined by Shannon's source coding theorem (https://en.wikipedia.org/wiki/Shannon%27s_source_coding_theorem). Typical schemes include Huffman coding and arithmetic coding, which assign shorter codewords to more probable symbols (https://en.wikipedia.org/wiki/Entropy_coding).

In the context of the $(p, k, 1)$ novelty game, both types of known strategies (oblivious and non-oblivious) can be viewed as strategies for encoding ancestry information into each output number. More specifically:

1. For oblivious strategies, the output is constructed by explicitly concatenating all ancestor labels or input-history data into the novelty number. This approach behaves like a naïve fixed-length code, with description length growing in proportion to the number of ancestors.

2. For non-oblivious strategies, the output embeds a reference to a previously generated novelty number, along with just enough extra information to ensure the new number is novel. This mirrors dictionary or delta coding methods, where prior symbols are reused to reduce the overall description length.

Oblivious strategies thus yield outputs with large description lengths, analogous to ignoring any redundancy in the source. Non-oblivious strategies achieve more compact outputs by exploiting structure in the sequence of novelty numbers, similar to how entropy-coding schemes reduce average code length by reusing common patterns.

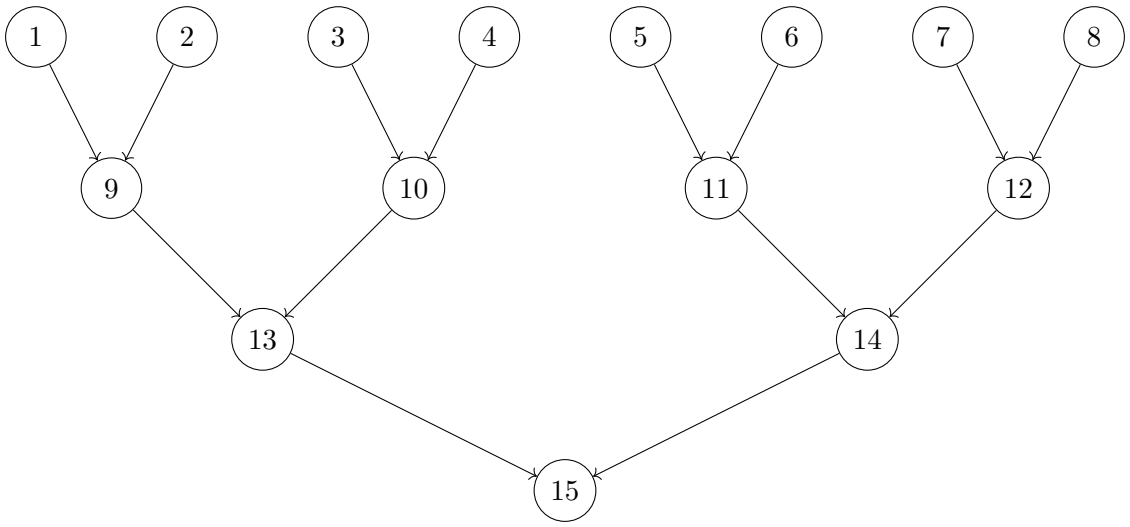
13.2.2 Novelty Game Interpreted as Complete k-ary Tree

The $(p, k, 1)$ novelty game can be interpreted as a complete k-ary tree. We can draw an inverted complete tree. The top level contains the p-th level ancestors, there are k^p such ancestors. The next level contains k^{p-1} ancestors, and so on. The bottom level has only one vertex. As long as each vertex in this tree is novel to its own ancestors, then this is a winning strategy for this specific input set. A winning strategy must hold novelty for all such possible input sets.

Note for non-oblivious strategies, the mappings between two adjacent levels can be different, since such a mapping corresponds to a specific player.

Figure 9 illustrates a 4-level complete binary tree for the $(3, 2, 1)$ novelty game.

Figure 9: Complete binary tree for the $(3, 2, 1)$ novelty game



With the help of interpreting novelty games as complete k-ary tree, we have the below claims:

Claim 6. $\mathfrak{B}(p, k, 1) \geq k^p + 1$.

Proof. If $N \leq k^p$, then we can put all the distinct numbers at the topmost level of the complete k-ary tree. (If $N < k^p$, then there will be some empty vertices at the top level, and we can fill the remaining empty vertices with any of those N numbers). For any specific strategy, when the topmost level vertices are specified, the remaining vertices in the tree are fully determined. To meet the novelty property, the bottom vertex must have a distinct number from all its ancestors, that is, all other vertices in this tree. If $N \leq k^p$, then we have no new number for this bottom vertex, so there is no any winning strategy. This means that we must have at least $k^p + 1$ distinct numbers. Therefore, $\mathfrak{B}(p, k, 1) \geq k^p + 1$. □

This $k^p + 1$ bound is a slightly improved lower bound than $pk + 1$, however, it is still a very trivial lower bound.

Claim 7. $\mathfrak{D}(p, k) \geq k^p + 1$ *for oblivious ancestor-bookkeeping strategies of the* $(p, k, 1)$ *game.*

Proof. Consider the ancestor bookkeeping strategy as a complete k-ary tree. For a $(p + 1)$-level k-ary tree, the topmost level contains k^p ancestors, and the next level contains k^{p-1} ancestors, ..., and the bottom level contains only one single output number. For a specific strategy, as long as the topmost k^p ancestors are chosen, then the numbers at all other levels are fully determined. The k^p ancestors are completely independent from one another, so it is always possible to choose different last digits for them. For ancestor-bookkeeping strategies, the last digits of those k^p ancestors are not kept in the output number due to space restriction, and we must let the last digit of the output be distinct from all these k^p ancestors. And we need one more unique digit for the output number itself. Therefore, $\mathfrak{D}(p, k) \geq k^p + 1$. □

13.2.3 Conjectures

We will provide two conjectures on the lower bound of novelty games to elicit further research.

Conjecture 1. *For oblivious ancestor-bookkeeping strategies with $p > 2$ players, the lower bound is*

$$\mathfrak{B}(p, k, 1) \geq \prod_{i=0}^{p-1} \binom{k^p + 1}{k^i}.$$

This conjecture comes from the observation or the above claim that $\mathfrak{D}(p, k) \geq k^p + 1$, and then the same Optimizations 3,4,5 can apply, but Optimization 6 does not apply, so we can derive this formula for the lower bound.

This is probably also the lower bound for all oblivious strategies, since all oblivious strategies are this ancestor-bookkeeping type strategies.

Some conjectured lower bounds for smaller p, k values:

$$\mathfrak{B}(3, 2, 1) \geq \binom{9}{4}\binom{9}{2}\binom{9}{1} = 40824$$

$$\mathfrak{B}(3, 3, 1) \geq \binom{28}{9}\binom{28}{3}\binom{28}{1} = 633556123200 \approx 6.3E11$$

$$\mathfrak{B}(4, 2, 1) \geq \binom{17}{8}\binom{17}{4}\binom{17}{2}\binom{17}{1} = 133767233600 \approx 1.3E11$$

Conjecture 2. $\mathfrak{B}(p, k, 1) \geq \mathfrak{B}(2, k, 1)^{k^{p-2}}$.

This should be the lower bound for all strategies, including both oblivious and non-oblivious strategies.

This comes from the recursive novelty property (Subsubsection 13.2.4): $\mathfrak{B}(p,k,1) \geq \mathfrak{B}(p-1,k,1)^k$, which comes from the observation that all inputs must contribute to the output equally: If any single input does not contribute to the output at all, then we can always choose the same value as the output to be one ancestor of this input, to violate the novelty property. As a result, the most efficient output is the simple concatenation of the novelty components from all k inputs, which is similar as the most efficient coding is equal-length Huffman encoding if all letters are equal probability. That is, to get a $(p,k,1)$ strategy from a $(p-1,k,1)$ strategy, the best way is by using k-dimensions, and each dimension is $\mathfrak{B}(p-1,k,1)$.

One intuitive thinking is using information theory. There is some limit for $\mathfrak{B}(p,k,1)$, which is due to the inherent entropy of the probability space of all the strategies (mappings from $\binom{N}{k}$ inputs to $N-k$ outputs). Such an entropy can be described by at least $log_2 N = log_2 \mathfrak{B}(p,k,1)$ bits, by algorithm complexity or Kolmogorov complexity or AIT (Algorithm Information Theory https://en.wikipedia.org/wiki/Algorithmic_information_theory). Note this entropy is irreducible. And because the k inputs are i.i.d. (independent and identically distributed), the new entropy of the $(p+1,k,1)$ novelty game is the addition of them, thus the new entropy can only be described by at least $k \cdot log_2 N$ bits, which means $\mathfrak{B}(p+1,k,1) \geq N^k$.

Another thinking or way to understand is: each number is a probability variable or entropy variable, which stands for the number of cases or combinations in itself to allow room for novelty. If this variable contains more cases, then it means more entropy and thus require more digits/bits to encode. With a single player, each input/output is almost fixed, with very low or lowest possibilities, that is why $\mathfrak{B}(1,k,1)$ can be encoded in $k+1$ ways. With two and more players, the number of possibilities increase exponentially (the number of highest level ancestors becomes k times more with one more player), thus this conjecture.

The k-ary complete tree (Figure 9) may help to understand it too. Each level adds k more times of ancestors, which adds k times of digits to encode or represent such added complexity.

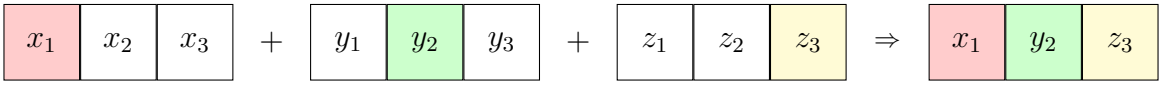
13.2.4 Recursive Novelty Property

Figure 10 illustrates the recursive novelty property for novelty games. This is for $k=3$ case. Each input x,y,z needs to provide $k=3$ novelty components at the same time to guarantee that there is always one novelty component available for the output to pick. For example, x_1,x_2,x_3 are 3 novelty components (or numbers) for the $(p-1,k,1)$ game. The output u consists of at least one novelty component from each input, to ensure novelty with all ancestors of x,y,z. Therefore, we must have this property $\mathfrak{B}(p,k,1) \geq \mathfrak{B}(p-1,k,1)^k$.

Note each input must contribute at least one novelty component to the output u. If an input like x does not contribute to the output u at all, then we will be able to freely choose the ancestor of this input x, that is, we can let one ancestor of x be exactly the same as the output u, to find a counterexample to violate the novelty property. Therefore, each input must contribute to the output.

Also, since all the k inputs are equal, the novelty component contribution to the output must also be equal.

Figure 10: Recursive novelty property for $k=3$ case



Claim 8. $\mathfrak{B}(p,k,1) \leq (k+1)^{k^{p-1}}$.

Proof. Using this recursive novelty approach, we can get a more trivial non-oblivious strategy: start from $\mathfrak{B}(1,k,1) = k+1$, we can get $\mathfrak{B}(2,k,1) = \mathfrak{B}(1,k,1)^k = (k+1)^k$, and then $\mathfrak{B}(3,k,1) = \mathfrak{B}(2,k,1)^k = (k+1)^{k^2}$. And for general p, we get $\mathfrak{B}(p,k,1) = (k+1)^{k^{p-1}}$. □

We know that there is an optimized bound for $\mathfrak{B}(2,k,1) = O(k^2 2^k)$, which is smaller than our new trivial bound $\mathfrak{B}(2,k,1) = (k+1)^k$. However, there is no such further optimization for $\mathfrak{B}(3,k,1)$, therefore, the conjecture is $\mathfrak{B}(p,k,1) \geq \mathfrak{B}(2,k,1)^{k^{p-2}}$, better than the above $\mathfrak{B}(1,k,1)^{k^{p-1}}$.

If this conjecture is true, and since we already know that there exists such a winning strategy [LS25] for $N = \mathfrak{B}(2,k,1)^{k^{p-2}}$, then we know the exact bound for novelty games: $\mathfrak{B}(p,k,1) = \mathfrak{B}(2,k,1)^{k^{p-2}}$.

Another future direction is to find a better $\mathfrak{B}(3,k,1)$ result than $\mathfrak{B}(2,k,1)^k$ using other novel approaches like graph or projective geometry methods, then this conjecture can be improved to be $\mathfrak{B}(p,k,1) = \mathfrak{B}(3,k,1)^{k^{p-3}}$.

14 Acknowledgments

The author would like to thank his excellent mentor, Maxwell Fishelson, for his continued guidance and advice. The author also would like to thank MIT PRIMES and its organizers for providing him with this incredible opportunity.

15 References

References

[APY10] Noga Alon, Rina Panigrahy, and Sergey Yekhanin. Deterministic approximation algorithms for the nearest codeword problem. In Manindra Agrawal, Lance Fortnow, Thomas Thierauf, and Christopher Umans, editors, *Algebraic Methods in Computational Complexity*, volume 9421 of *Dagstuhl Seminar Proceedings (DagSemProc)*, pages 1–13, Dagstuhl, Germany, 2010. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[AS10] Vikraman Arvind and Srikanth Srinivasan. The remote point problem, small bias spaces, and expanding generator sets. In Jean-Yves Marion and Thomas Schwentick, editors, *27th International Symposium on Theoretical Aspects of Computer Science*, volume 5 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 59–70, Dagstuhl, Germany, 2010. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[Erd63] Paul Erdös. On a problem in graph theory. *The Mathematical Gazette*, 47(361):220–223, 1963.

[GGNS23] Karthik Gajulapalli, Alexander Golovnev, Satyajeet Nagargoje, and Sidhant Saraogi. Range avoidance for constant-depth circuits: Hardness and algorithms. *arXiv preprint arXiv:2303.05044*, 2023.

[GS71] Ronald L Graham and Joel H Spencer. A constructive solution to a tournament problem. *Canadian Mathematical Bulletin*, 14(1):45–48, 1971.

[LS25] Ulysse Léchine and Thomas Seiller. Kolmogorov time hierarchy and novelty games. working paper or preprint, May 2025.

[OEI24] OEIS. A362137 smallest size of an n-paradoxical tournament built as a directed paley graph., 2024.

[RMHH04] K Brooks Reid, Alice A McRae, Sandra Mitchell Hedetniemi, and Stephen T Hedetniemi. Domination and irredundance in tournaments. *Australasian journal of combinatorics*, 29:157–172, 2004.

[RSW22] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. On the range avoidance problem for circuits. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 640–650. IEEE, 2022.

[SS65] Esther Szekeres and George Szekeres. On a problem of schütte and erdös. *The Mathematical Gazette*, pages 290–293, 1965.

[VW23] Nikhil Vyas and Ryan Williams. On oracles and algorithmic methods for proving lower bounds. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*, volume 251 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 99:1–99:26, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

A Full Algorithms

This appendix provides pseudocode for the ancestor-bookkeeping strategy and its optimizations for the novelty $(p, k, 1)$ game, as referenced in previous sections. Algorithm 1 and Algorithm 3 describe the baseline strategy, while Algorithm 5–10 implement Optimizations 1–5. Optimization 6 consists of highly specific casework, so we did not find it worthwhile to include a general algorithm for it.

Note in the below algorithms, when a number x is represented as base-t array, the index of the right-most digit is 0. When converting x to base-t array, the number of digits in the array is $\frac{k^p-1}{k-1}$. Also for an array A, we use $sorted(A)$ to denote the sorted result of this array A in non-decreasing order.

Algorithm 1 The original ancestor-bookkeeping algorithm for $(p, k, 1)$

Input: Parameters p, k; inputs $x_1, \ldots, x_k$
$t \leftarrow \frac{k^{p+1}-1}{k-1}$
Get the base-t representation array of $x_1, \ldots, x_k$
Get A_i subarrays of all k inputs, each input x_j is in form of $A_{p-1}A_{p-2}...A_2A_1A_0$ or $\bigoplus_{i=0}^{p-1} A_i$
for i in range $[1, p-1]$ **do**
    $A_i(z) = \bigoplus_{j=1}^{k} A_{i-1}(x_j)$
end for
Output: ancestor digits $A(z)$

Algorithm 2 The original last digit selection algorithm for $(p, k, 1)$

Input: Parameters p, k; inputs $x_1, \ldots, x_k$, and the ancestor digits $A(z)$ of output z
$t \leftarrow \frac{k^{p+1}-1}{k-1}$
Get the base-t representation array of $x_1, \ldots, x_k$
Get A_{p-1} subarrays of all k inputs
$s \leftarrow$ the set of all unique digits in $A(z)$ of the output and the A_{p-1} subarrays of all k inputs
$v \leftarrow$ smallest digit in the set $[0 : t-1] - s$ ▷ such a digit always exists
Output: the last digit v

Algorithm 3 The original $(p, k, 1)$ algorithm without any optimization in Subsection 4.2

Input: Parameters p, k; inputs $x_1, \ldots, x_k$
$t \leftarrow \frac{k^{p+1}-1}{k-1}$
Get the base-t representation array of $x_1, \ldots, x_k$
Get ancestor digits $A(z)$ of the output using the above ancestor-bookkeeping Algorithm 1
Get the last digit v of the output using the above last digit selection Algorithm 2
Get the value of the new base-t array $A(z)v$ and assign to z
Output: z

Algorithm 4 The last digit selection algorithm for Optimization 1: Input Avoidance

Input: Parameters p, k; inputs $x_1, \ldots, x_k$
$t \leftarrow \frac{k^{p+1}-1}{k-1} - k$
$s \leftarrow$ the set of all unique digits of the k inputs
$d \leftarrow |s|$
if $d < t$ **then**
  $v \leftarrow$ smallest digit in the set $[0 : t-1] - s$ ▷ such a digit always exists
else
  $v \leftarrow m(x_i)$, where $m(x_i) \notin A(x_i)$ ▷ such a digit always exists
end if
Output: the last digit v

Algorithm 5 The $(p, k, 1)$ algorithm for Optimization 1: Input Avoidance

Input: Parameters p, k; inputs $x_1, \ldots, x_k$
$t \leftarrow \frac{k^{p+1}-1}{k-1} - k$
Get the base-t representation array of $x_1, \ldots, x_k$
Get A_i subarrays of all k inputs, each input x_j is in form of $\overline{A_{p-1}A_{p-2}...A_2A_1A_0}$
Run the ancestor-bookkeeping Algorithm 1 to create ancestor digits $A(z)$ of the output z
Get the last digit v of the output with new last digit selection Algorithm 4
Get the value of the new base-t array $\overline{A(z)v}$ and assign to z
Output: z

Algorithm 6 The last digit selection algorithm for Optimization 2: Cycle Avoidance

Input: Parameters p, k; inputs $x_1, \ldots, x_k$, and $A(z) = A_{p-1}A_{p-2}...A_2A_1$ of output z
$t \leftarrow \mathfrak{D}(p, k)$ ▷ Digit range from Equation 10
Get the base-t representation array of $x_1, \ldots, x_k$
Get A_i subarrays of all k inputs, each input x_j is in form of $A_{p-1}A_{p-2}...A_2A_1A_0$
Now generate the last digit v of the output number with below new rules
$s \leftarrow$ the set of all unique digits in the A_{p-1} subarrays of the k inputs
for i in range $[1, \lceil \frac{p}{2} \rceil - 1]$ **do** ▷ Theorem 3
  for j in range $[1, k]$ **do** ▷ All inputs
    $f \leftarrow$ FALSE ▷ No-cycle flag
    for m in range $[1, i]$ **do** ▷ Lemma 3
      if $A_m(z) \not\succ A_{p-i-m+1}(x_j)$ **then** ▷ Equation 12 for $\succ$ check
        $f \leftarrow$ TRUE ; break
      end if
    end for
    if f not TRUE **then**
      Add unique digits of $A_{p-i-1}(x_j)$ to set s
    end if
  end for
end for
$v \leftarrow$ smallest digit in the set $[0 : t-1] - s$ ▷ such a digit always exists
Output: the last digit v

Algorithm 7 The $(p, k, 1)$ algorithm with Optimization 2: Cycle Avoidance

Input: Parameters p, k; inputs $x_1, \ldots, x_k$
$t \leftarrow \mathfrak{D}(p, k)$ ▷ Digit range from Equation 10
Get the base-t representation array of $x_1, \ldots, x_k$
Get A_i subarrays of all k inputs, each input x_j is in form of $A_{p-1}A_{p-2}...A_2A_1A_0$
Run the ancestor-bookkeeping Algorithm 1 to create ancestor digits $A(z)$ of the output z
Get the last digit v of the output with new last digit selection Algorithm 6
Get the value of the new base-t array $\overline{A(z)v}$ and assign to z
Output: z

Algorithm 8 The $(p, k, 1)$ Algorithm with Optimization 3: Reordering Ancestors

Input: Parameters p, k; inputs $x_1, \ldots, x_k$
$t \leftarrow \mathfrak{D}(p, k)$ ▷ Digit range from Equation 10
$y \leftarrow \overline{A(y)v}$ from Algorithm 7 ▷ Equation 13 for $\succ$ check
$L \leftarrow 1$ ▷ The ancestor level
$low \leftarrow 1$, $high \leftarrow k$ ▷ Bounds of $A_1(y)$
while $L < p$ **do**
  $A_L(y) \leftarrow sorted(A_L(y))$ ▷ $A_L(y) = [low : high]$
  $L \leftarrow L + 1$
  $low \leftarrow high + 1$
  $high \leftarrow high + k^L$
end while
Reconstruct the base-t number $\overline{A(y)v}$ and assign to z
Output: z

Algorithm 9 The $(p, k, 1)$ algorithm with Optimization 4: Merging Equivalent Ancestors

Input: Parameters p, k; inputs $x_1, \ldots, x_k$
$t \leftarrow \mathfrak{D}(p, k)$ ▷ Digit range from Equation 10
$y \leftarrow \overline{A(y)v}$ from Algorithm 7 ▷ Equation 14 for $\succ$ check
$L \leftarrow 1$ ▷ The ancestor level
$low \leftarrow 1$, $high \leftarrow k$ ▷ Bounds of $A_1(y)$
while $L < p$ **do**
  $U \leftarrow$ the set of unique digits of $A_L(y) = [low, high]$
  $d_{min} \leftarrow min(U)$
  $A_L(y) \leftarrow [d_{min}] * (k^L - len(U)) \bigoplus sorted(U)$
  $L \leftarrow L + 1$
  $low \leftarrow high + 1$
  $high \leftarrow high + k^L$
end while
Reconstruct the base-t number $\overline{A(y)v}$ and assign to z
Output: z

Algorithm 10 The $(p, k, 1)$ algorithm with Optimization 5: Merging Neighboring Ancestors

Input: Parameters p, k; inputs $x_1, \ldots, x_k$
$t \leftarrow \mathfrak{D}(p, k)$ ▷ Digit range from Equation 10
$A(z) \leftarrow$ from Algorithm 1
$L \leftarrow 1$ ▷ The ancestor level
$low \leftarrow 1$, $high \leftarrow k$ ▷ Bounds of $A_1(z)$
while $L < p$ **do**
  $U \leftarrow$ the set of unique digits of $A_L(z) = [low, high]$
  $W \leftarrow [\mathfrak{D}(p, k)] - U$
  $F \leftarrow$ the set of $(k^L - len(U))$ lowest digits from W ▷ F is set of "fake" digits
  $A_L(z) \leftarrow sorted(U + F)$
  $L \leftarrow L + 1$
  $low \leftarrow high + 1$
  $high \leftarrow high + k^L$
end while
$v \leftarrow$ from updated $A(z)$ with Algorithm 6 ▷ Equation 15 for $\succ$ check
Reconstruct the base-t number $\overline{A(z)v}$ and assign to z
Output: z

B Examples of Inputs and Output for Various Algorithms

This appendix provides some examples to illustrate the output number generation from some specific input numbers, to help understand the algorithms of the baseline strategy and various optimizations. The below Figure 11 is an example of inputs and output for the original baseline algorithm of the $(3, 2, 1)$ novelty game. And other figures are input/output examples for various optimizations.

The original algorithm Figure 11 simply concatenates all ancestor groups of the inputs at the same level and selects one unused unique digit as the last digit $m(z)$ to construct the output.

For Optimization 1 of input avoidance Figure 12, notice that the last digit $m(z)$ of the output number can be the same as one of the input numbers which does not appear in the ancestor digits of all inputs. For Optimization 2 of cycle avoidance Figure 13, notice that the last digit $m(z)$ of the output number can also be the same as one of the input numbers, although the last digit selection rule (2) is different from Optimization 1.

For Optimization 3 of reordering ancestors Figure 14, its last digit $A_0(z)$ or $m(z)$ is the same as Optimization 2 of cycle avoidance, while the only difference is the digit order in each ancestor group $A_i(z)$ is fixed in non-decreasing order. As a result, some numbers like $4565644, 5546464$ become invalid outputs, and are replaced by the same valid output 4556464.

For Optimization 4 of merging equivalent ancestors Figure 15, notice that each ancestor group is replaced by its equivalent ancestor group. $4565, 5546, 5646$ now become invalid ancestor group digits and are replaced by the same equivalent ancestor group 4456.

For Optimization 5 of merging neighboring ancestors Figure 16, notice that each ancestor group is replaced by its "neighboring" ancestor group. $1413, 1314, 0304$ become invalid ancestor group digits and are replaced by the same neighboring ancestor group 0134. As a result, the ancestor group digits of a valid output must contain unique digits in sorted order, thus significantly reducing the number of valid outputs.

For Optimization 6 of pruning unused outputs, we can see from Figure 17 that it is impossible to reach $m(z) = 9$ or $m(z) = A$ if $A_1(z)$ contains the digit 9 or A. Therefore, we can remove such unused outputs from the set of vertices to reduce the bound.

Figure 11: Original algorithm or baseline algorithm

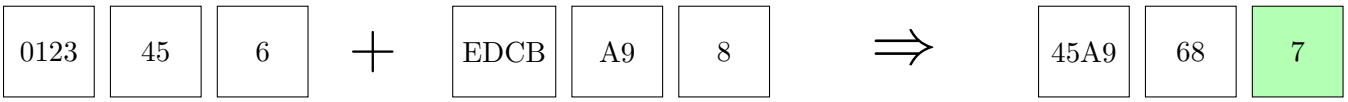


Figure 12: Optimization 1 algorithm of input avoidance

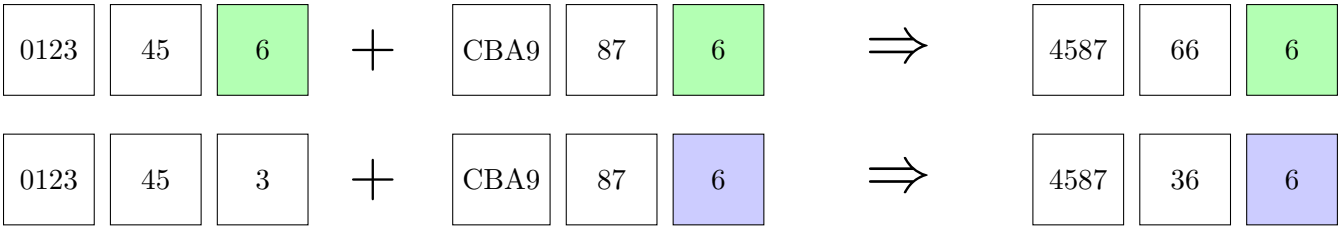


Figure 13: Optimization 2 algorithm of cycle avoidance

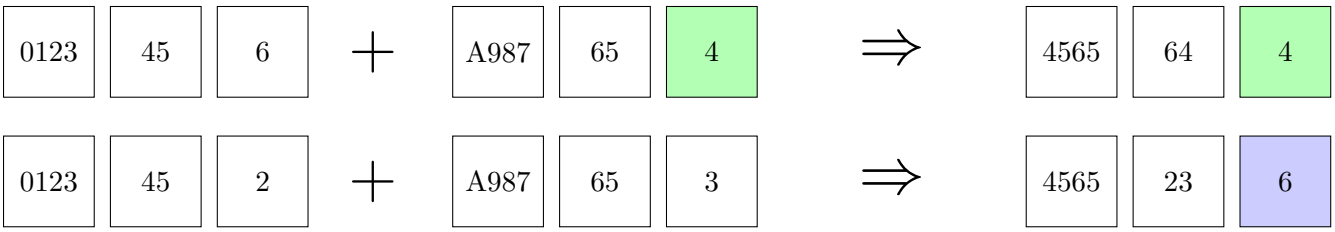


Figure 14: Optimization 3 algorithm of reordering ancestors

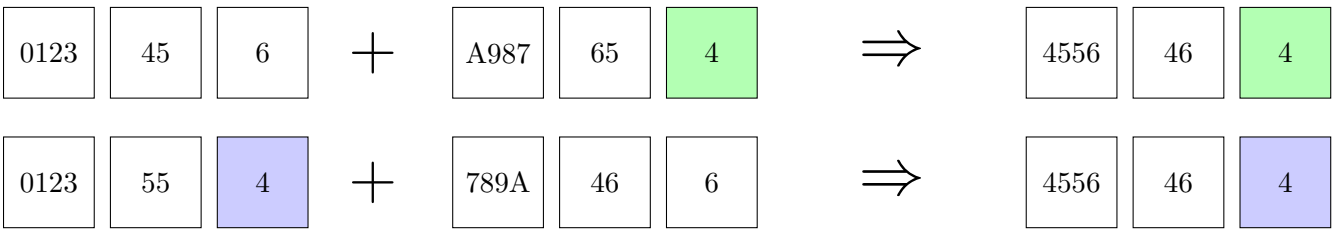


Figure 15: Optimization 4 algorithm of merging equivalent ancestors

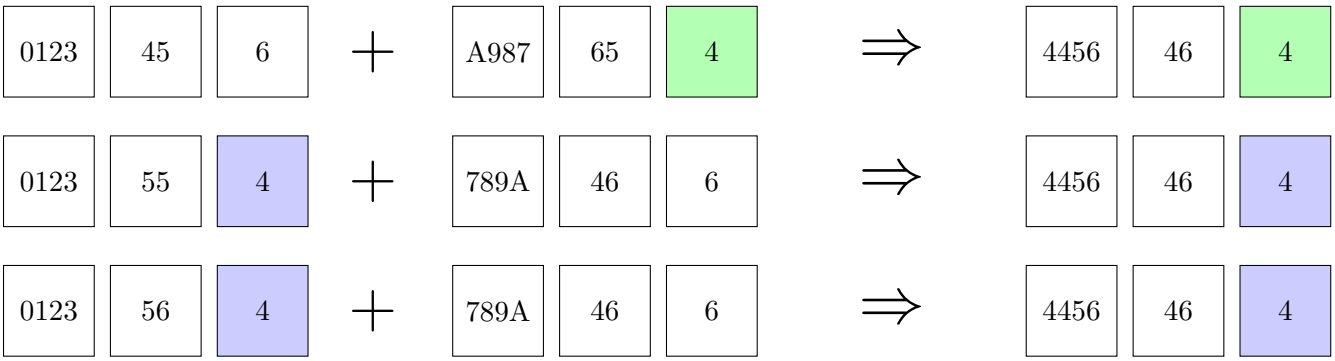


Figure 16: Optimization 5 algorithm of merging neighboring ancestors

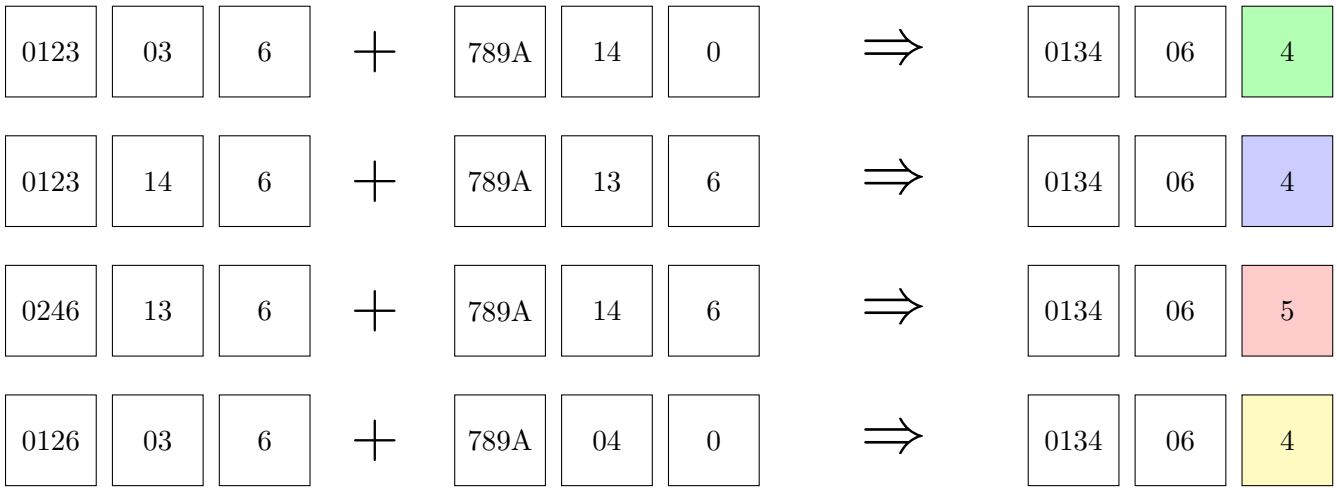
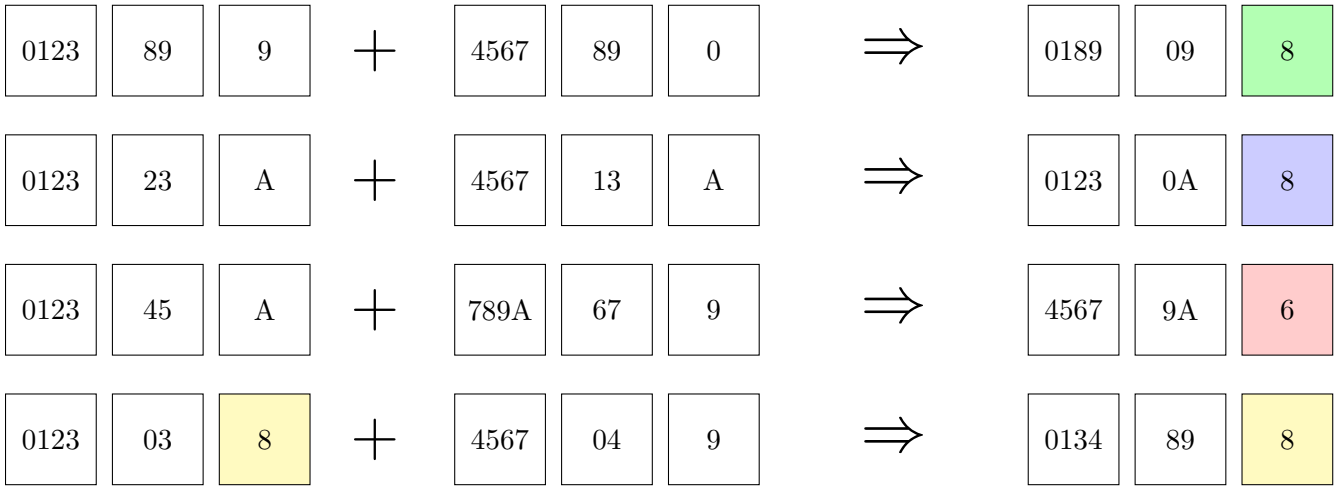


Figure 17: Optimization 6 algorithm of pruning unused outputs



C Novelty Game Simulator

I wrote some Python code to simulate the calculation of outputs for various inputs under different optimizations of the $(p, k, 1)$ novelty game. The code can be found here: https://github.com/michaellhan/2025primes.

The script allows modifications to the number of players, the number of inputs per player, and the optimization levels. It can also compute the bounds for all the optimizations.