A Simple Formula for Binomial Coefficients Revealed Through Polynomial Encoding

Joseph M. Shunia

November 2023

Abstract

We provide a detailed derivation of a new formula for binomial coefficients by harnessing an underexplored property of polynomial encoding. The formula, $\binom{n}{k} = \left\lfloor \frac{(1+2^n)^n}{2^{nk}} \right\rfloor \bmod 2^n$, is valid for $n > 0$ and $0 \leq k \leq n$. We relate this formula to existing mathematical methods via Kronecker substitution. To showcase the versatility of our approach, we also apply it to multinomials. A baseline computational complexity analysis identifies opportunities for optimization. We conclude by positing an open problem concerning the efficient computation of $\binom{n}{k}$ modulo n using our formula.

1 Introduction

Binomial coefficients, denoted by $\binom{n}{k}$, are fundamental in various mathematical domains. Conventional techniques for their computation involve factorials, Gamma functions, and Pascal's triangle. However, alternative approaches can sometimes offer new computational efficiencies and analytical insights. We present a new formula for binomial coefficients, given by:

$$\binom{n}{k} = \left\lfloor \frac{(1+2^n)^n}{2^{nk}} \right\rfloor \bmod 2^n, \quad \text{for } n > 0 \text{ and } 0 \leq k \leq n \tag{1}$$

Our formula depends only on basic arithmetic operations, yet underscores the deep connections between binomial coefficients, polynomials, and modular arithmetic. It stems from a fascinating property of polynomial encoding that has been rarely explored in mathematical literature. We give a detailed proof, and relate this formula to established techniques like Kronecker substitution which reveals new opportunities for computational optimization.

As a demonstration of the versatility of our methodology, we derive a generalized formula for the coefficients in the multinomial expansion of arbitrary degree univariate unit polynomials. Through this, we show how our technique can be applied to calculate the coefficients of univariate polynomial expansions in general.

We analyze the computational complexity of our binomial formula and compare it to typical factorial-based approaches. An open problem is posed regarding efficient computation of $\binom{n}{k} \pmod n$ using a variant of our formula.

Overall, our work establishes new theoretical results and computational techniques in a fundamental area of mathematics. By examining a specific underexplored property of polynomial encoding and applying it in an original way, we open new research directions with potential for significant impact on computational mathematics.

2 A Property of Polynomial Encoding

Theorem 1. If $P(x)$ is a polynomial with non-negative integer coefficients, and $n \geq 1$ is an integer, then $P(x)$ can be completely determined by the values $P(n)$ and $P(P(n))$.

Proof. Let $q = P(n)$, which gives the sum of the coefficients of P evaluated at n. Consider $P(P(n)) = P(q)$ expressed in base q; the digits correspond exactly to the coefficients of P. The only possible ambiguity arises if $P(q) = q \cdot N$ for some N, but since the coefficients evaluated at $Q(n)$ sum to q, we deduce that $P = q \cdot x^{N-1}$ in this case. □

Remark. This property appears to be under explored in the literature; we were unable to find any papers explicitly describing it. However, it has been the subject of some online discussion and at least one blog post [1]. The proof given above was first proposed by user ARupinski in a MathOverflow post [2] for the case $x = 1$, and has been generalized for the purposes of this paper.

3 Derivation of Binomial Coefficient Formula

3.1 Definition of Polynomial Function

We define $P(x)$ as the polynomial function:

$$P(x) = 1 + x \tag{2}$$

If n is a non-negative integer, then by the Binomial Theorem [3] we have:

$$P(x)^n = (1 + x)^n = \sum_{k=0}^{n} \binom{n}{k} x^k \tag{3}$$

As the coefficients of the terms in $P(x)$ are precisely the binomial coefficients for the n-th row of Pascal's Triangle [4], it is apparent:

$$[x^k]P(x)^n = \binom{n}{k} \tag{4}$$

3.2 Application of the Polynomial Property

The value of $P(1)^n$ is simply the summation of the coefficients of $P(x)^n$. It is easy to see that:

$$P(1)^n = \sum_{k=0}^{n} \binom{n}{k} 1^k = (1 + 1)^n = 2^n \tag{5}$$

$$P(P(1)^n)^n = \sum_{k=0}^{n} \binom{n}{k} (P(1)^n)^k = (1 + P(1)^n)^n \tag{6}$$

$$= \sum_{k=0}^{n} \binom{n}{k} (2^n)^k = (1 + 2^n)^n \tag{7}$$

3.3 Reconstruction of Polynomial Coefficients

We can use the result from Theorem 1 to recover the coefficients of $P(x)^n$. We will achieve this by first encoding $P(P(1)^n)^n$ in base $P(1)^n$, and then recovering the coefficients of $P(x)^n$ from the encoded value.

Lemma 1. The formula to recover the k-th coefficient of $P(x)^n$ is given by:

$$[x^k]P(x)^n = \left\lfloor \frac{(1+P(1)^n)^n \bmod P(1)^{nk+n}}{P(1)^{nk}} \right\rfloor \tag{8}$$

Proof. Given an integer n and a base $b > 1$. Let $L = \lfloor \log_b n \rfloor$. Then the encoded representation of n in base b is the unique representation of n, such that [5]:

$$n = \sum_{k=0}^{L} b^k \left\lfloor \frac{n \bmod b^{k+1}}{b^k} \right\rfloor \tag{9}$$

By inserting $P(1)$ into the equation, we can see:

$$[x^k]P(x)^n = \left\lfloor \frac{P(P(1)^n)^n \bmod (P(1)^n)^{k+1}}{(P(1)^n)^k} \right\rfloor \tag{10}$$

$$= \left\lfloor \frac{(1+P(1)^n)^n \bmod P(1)^{nk+n}}{P(1)^{nk}} \right\rfloor \tag{11}$$

Hence, the lemma is proven. □

3.4 Binomial Coefficient Formula Proof

Theorem 2.

$$\binom{n}{k} = \left\lfloor \frac{(1+2^n)^n}{2^{nk}} \right\rfloor \bmod 2^n, \quad \text{for } n > 0 \text{ and } 0 \le k \le n \tag{12}$$

Proof. **Step 1: Proof that the equation is valid:**
We begin with the identity we established in Lemma 1:

$$[x^k]P(x)^n = \left\lfloor \frac{(1+P(1)^n)^n \bmod P(1)^{nk+n}}{P(1)^{nk}} \right\rfloor \tag{13}$$

By substituting $[x^k]P(x)^n = \binom{n}{k}$ and $P(1) = 2$, we have:

$$\binom{n}{k} = \left\lfloor \frac{(1+2^n)^n \bmod 2^{nk+n}}{2^{nk}} \right\rfloor \tag{14}$$

Next, we define $X = (1+2^n)^n$, $Y = 2^{nk+n}$, $Z = 2^{nk}$. By replacement:

$$\binom{n}{k} = \left\lfloor \frac{X \bmod Y}{Z} \right\rfloor \tag{15}$$

To simplify the expression, we can utilize the following well-known identity for the mod operation [6]:

$$a \bmod b = a - b \left\lfloor \frac{a}{b} \right\rfloor \tag{16}$$

By application of the identity, the expression simplifies as follows:

$$\binom{n}{k} = \left\lfloor \frac{X \bmod Y}{Z} \right\rfloor \tag{17}$$

$$= \left\lfloor \frac{X - Y \left\lfloor \frac{X}{Y} \right\rfloor}{Z} \right\rfloor \tag{18}$$

$$= \left\lfloor \frac{X}{Z} - \frac{Y}{Z} \left\lfloor \frac{X}{Y} \right\rfloor \right\rfloor \tag{19}$$

Next, we replace X, Y, and Z with their respective values and simplify further:

$$\binom{n}{k} = \left\lfloor \frac{X}{Z} - \frac{Y}{Z} \left\lfloor \frac{X}{Y} \right\rfloor \right\rfloor \tag{20}$$

$$= \left\lfloor \frac{(1+2^n)^n}{2^{nk}} - \frac{2^{nk+n}}{2^{nk}} \left\lfloor \frac{(1+2^n)^n}{2^{nk+n}} \right\rfloor \right\rfloor \tag{21}$$

$$= \left\lfloor \frac{(1+2^n)^n}{2^{nk}} - 2^n \left\lfloor \frac{(1+2^n)^n}{2^{nk+n}} \right\rfloor \right\rfloor \tag{22}$$

$$= \left\lfloor \frac{(1+2^n)^n}{2^{nk}} \right\rfloor - 2^n \left\lfloor \frac{(1+2^n)^n}{2^{nk+n}} \right\rfloor \tag{23}$$

Finally, we again make use of the mod operation identity (16) to simplify:

$$\binom{n}{k} = \left\lfloor \frac{(1+2^n)^n}{2^{nk}} \right\rfloor \bmod 2^n \tag{24}$$

We have arrived at our formula.

Step 2: Proof that the equation holds only for $n > 0$ and $0 \leq k \leq n$: To prove why the equation $\binom{n}{k} = \left\lfloor \frac{(1+2^n)^n}{2^{nk}} \right\rfloor \bmod 2^n$ holds only for $n > 0$ and $0 \leq k \leq n$, we consider the following cases:

1. $n \leq 0$: In this case, the modulus becomes 2^{-n} and the original equation does not make sense.
2. $k < 0$: In this case, we have $2^{-nk} = \frac{1}{2^{nk}}$ and the original expression becomes $2^{nk}(1+2^n)^n \bmod 2^n$, which is always equal to 0 since $2^n \mid 2^{nk}$.
3. $k > n$: In this case, $2^{nk} > (1+2^n)^n > 2^{nn}$, hence $\left\lfloor \frac{(1+2^n)^n}{2^{nk}} \right\rfloor = 0$. We note that $\binom{n}{k} = \binom{k}{n}$ here, which cannot equal 0.

Conclusion: We have shown that our formula is valid and that it holds only for $n > 0$ and $0 \leq k \leq n$. Hence, the theorem is proven. □

3.5 Close Inspection of Binomial Coefficient Formula

Upon initial inspection, it may not be obvious why our formula works:

$$\binom{n}{k} = \left\lfloor \frac{(1+2^n)^n}{2^{nk}} \right\rfloor \bmod 2^n, \quad \text{for } n > 0 \text{ and } 0 \leq k \leq n$$

However, by expanding the numerator $(1+2^n)^n$, we can see:

$$(1+2^n)^n = \sum_{j=0}^{n} \binom{n}{j} (2^n)^j \tag{25}$$

Dividing by the denominator 2^{nk} and taking the sum modulo 2^n gives:

$$\frac{(1+2^n)^n}{2^{nk}} \bmod 2^n = \sum_{j=0}^{n} \binom{n}{j} (2^n)^{j-k} \bmod 2^n \tag{26}$$

The terms with $j < k$ will sum to a value that is less than 1 and therefore, will vanish upon application of the floor operation. The terms with $j > k$ will be divisible by 2^n and therefore, vanish upon application of the mod operation. The only remaining term is $j = k$, which is equal to $\binom{n}{k}$ since $(2^n)^{j-k} = (2^n)^0 = 1$.

4 Connection to Kronecker Substitution

4.1 An Alternative Derivation

The formula presented for computing binomial coefficients can also be derived using Kronecker substitution [7]. However, Kronecker substitution is a technique that it is typically used for polynomial multiplication, rather than deriving explicit coefficient formulas. This technique substitutes a variable x in a polynomial $P(x)$ with a discrete value to yield a finite representation. By performing this substitution in an appropriate base, the digits directly correspond to the coefficients of the original polynomial.

Specifically, for a polynomial $P(x)$ with integer coefficients, performing a Kronecker substitution with $x = b$ yields $P(b)$ [7]. Expanding this in base b gives a direct correspondence between the digits and coefficients of the original polynomial. That is:

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \tag{27}$$

$$P(b) = (a_0 b^0 + a_1 b^1 + \ldots + a_n b^n)_b \tag{28}$$

Where $(a)_b$ denotes the representation of integer a in base b.

We can use this approach to derive the binomial coefficient formula as follows:

1. Begin with the polynomial $P(x) = 1 + x$
2. Perform a Kronecker substitution with $x = 2^n$, yielding:
$$P(2^n) = 1 + 2^n = A \tag{29}$$
3. Raise A to the nth power:
$$A^n = (1 + 2^n)^n \tag{30}$$
4. Reverse the Kronecker substitution on A^n in base 2^n to obtain the coefficients. By the digit-coefficient correspondence, these are precisely the binomial coefficients.

Therefore, by carefully selecting the base for the Kronecker substitution, we arrive at the formula presented in Theorem 2. It is important to note that our usage of Kronecker substitution in this manner, while valid, is highly unusual compared to typical applications focused on polynomial multiplication. Although this application of Kronecker substitution produces the same binomial coefficients, its limitations become apparent when we examine the general properties of polynomial coefficients.

4.1.1 Observation

We notice that our key property of polynomials from Theorem 1 is related to Kronecker substitution in the following way: for a given polynomial $P(x)$, $P(1)$ gives the minimum base that can be used with Kronecker substitution to ensure that all coefficients can be recovered completely.

4.2 The Hidden Difficulty of Base Selection

A cursory reading of our derivation might suggest that the results are trivially achieved through Kronecker substitution. Such an interpretation would miss the nuanced challenges associated with choosing an appropriate base for the substitution, especially in the general case. While Kronecker substitution does yield a result that encodes polynomial coefficients as digits in a particular base, it does not provide an explicit method to determine what that base should be to recover a formula for those coefficients. The base selection is typically input specific, and influenced by computational efficiency considerations, such as fast bitwise operations in base 2.

To illustrate this point, consider the difficulty of applying Kronecker substitution to the following polynomial:

$$Q(x)^5 = (1 + 13x + 3x^2 + 7x^3 + 11x^5)^5 \tag{31}$$

Applying Theorem 1, we get:

$$Q(1)^5 = 35^5 \tag{32}$$

$$Q(Q(1))^5 = (1 + 13(35^5) + 3(35^5)^2 + 7(35^5)^3 + 11(35^5)^5)^5 \tag{33}$$

In this example, the base needed for coefficient recovery via Kronecker substitution would require expansion and other additional calculations to identify. Moreover, the step to reverse Kronecker substitution often involves complicated manipulations. While these might yield the coefficients, they are far removed from the original formula and are more computationally intensive. This demonstrates that choosing a suitable base, particularly in the general case, is far from trivial.

Our approach offers a key advantage in this regard: Theorem 1 furnishes us with an elegant way to determine the minimum base that ensures complete recovery of the polynomial's coefficients for all inputs, thus enabling us to derive an explicit formula. This addresses a non-trivial challenge in the use of Kronecker substitution for polynomial coefficient recovery, especially when the polynomial has a more complex structure or higher degree. We demonstrate this further in §5 by deriving an explicit formula for coefficients in the multinomial expansion of univariate unit polynomials.

5 Deriving a Formula for Coefficients in the Multinomial Expansion of Univariate Unit Polynomials

Building upon the methodologies employed in our binomial coefficient derivation, we derive a generalized formula for calculating coefficients within the multinomial expansion of arbitrary degree univariate unit polynomials:

$$\left(\frac{x^D - 1}{x - 1}\right)^n = (1 + x + \ldots + x^{D-1})^n \tag{34}$$

The conventional approach to determine these coefficients utilizes conditional summations of multinomial coefficients $\binom{n}{k_0,k_1,\ldots,k_{D-1}}$, which represent the number of ways specific choices can be made to yield the term x^k [8]:

$$\binom{n}{k_0, k_1, \ldots, k_{D-1}} = \frac{n!}{k_0!k_1! \ldots k_{D-1}!} \tag{35}$$

In the context of our univariate unit polynomial, for each power of x in the expansion, the coefficient will come from all the combinations of powers that sum up to that specific power. Specifically, the coefficient of x^k in the expansion of our polynomial is [9]:

$$[x^k](1 + x + \ldots + x^{D-1})^n = \sum \binom{n}{k_0, k_1, \ldots, k_{D-1}} \tag{36}$$

Where the summation criteria are:

$$k_0 + k_1 + \ldots + k_{D-1} = n \tag{37}$$
$$0 \cdot k_0 + 1 \cdot k_1 + \ldots + (D-1) \cdot k_{D-1} = k \tag{38}$$

5.1 Multinomial Formula and Its Proof

Theorem 3.

$$[x^k] \left(\frac{x^D - 1}{x - 1} \right)^n = \left\lfloor \left(\frac{D^{Dn} - 1}{D^{n+k} - D^k} \right)^n \right\rfloor \bmod D^n,$$
$$\text{for } n > 0 \text{ and } 0 \leq k \leq n \cdot (D-1)$$

Proof. **Step 1: Proof that the equation is valid:**
We define the polynomial function:

$$P_D(x)^n = \left(\frac{x^D - 1}{x - 1} \right)^n = (1 + x + \ldots + x^{D-1})^n \tag{39}$$

In this case, it is clear that:

$$P_D(1)^n = D^n \tag{40}$$

Therefore, we have:

$$P_D(P_D(1)^n)^n = (1 + D^n + \ldots + D^{n(D-1)})^n \tag{41}$$

Observe that the inner sum is equivalent to the summation of the powers of D^n from 0 to $(D-1)$. We note the following identity [10]:

$$\sum_{k=0}^{n-1} n^k = \frac{n^n - 1}{n - 1} \tag{42}$$

By substitution, we have:

$$P_D(P_D(1)^n)^n = \left(\sum_{k=0}^{D-1} D^{nk} \right)^n = \left(\frac{D^{Dn} - 1}{D^n - 1} \right)^n \tag{43}$$

In of our proof of Theorem 2, we established that:

$$[x^k] P(x)^n = \left\lfloor \frac{P(P(1)^n)^n}{P(1)^{nk}} \right\rfloor \bmod P(1)^n \tag{44}$$

By equivalence, we have:

$$[x^k] P_D(x)^n = \left\lfloor \frac{P_D(P_D(1)^n)^n}{P_D(1)^{nk}} \right\rfloor \bmod P_D(1)^n \tag{45}$$

Finally, we replace the values of $P_D(1)^n$ and $P_D(P_D(1)^n)^n$ and simplify to arrive at our original equation:

$$[x^k] \left(\frac{x^D - 1}{x - 1} \right)^n = \left\lfloor \left(\frac{D^{Dn} - 1}{D^n - 1} \right)^n \cdot D^{-nk} \right\rfloor \bmod D^n \tag{46}$$
$$= \left\lfloor \left(\frac{D^{Dn} - 1}{D^n - 1} \cdot D^{-k} \right)^n \right\rfloor \bmod D^n \tag{47}$$
$$= \left\lfloor \left(\frac{D^{Dn} - 1}{D^k (D^n - 1)} \right)^n \right\rfloor \bmod D^n \tag{48}$$
$$= \left\lfloor \left(\frac{D^{Dn} - 1}{D^{n+k} - D^k} \right)^n \right\rfloor \bmod D^n \tag{49}$$

Step 2: Proof that the equation holds only for $n > 0$ and $0 \leq k \leq n \cdot (D-1)$:
To prove why the equation holds only for $n > 0$ and $0 \leq k \leq n \cdot (D-1)$, we consider the following cases in relation to:

$$[x^k]\left(\frac{x^D-1}{x-1}\right)^n = \left\lfloor \left(\frac{D^{Dn}-1}{D^{n+k}-D^k}\right)^n \right\rfloor \bmod D^n \tag{50}$$

$$= \left\lfloor \left(\frac{D^{Dn}-1}{D^n-1}\right)^n \cdot D^{-nk} \right\rfloor \bmod D^n \tag{51}$$

1. $n \leq 0$: In this case, the modulus becomes D^{-n} and the original equation does not make sense.
2. $k < 0$: In this case, we have $D^{-(-nk)} = D^{nk}$ and thus the original expression is always equal to 0, since $D^n \mid D^{nk}$.
3. $k > n \cdot (D-1)$: In this case, $D^{nk} > \left(\frac{D^{Dn}-1}{D^n-1}\right)^n > D^{nn}$, so the value of the expression is always equal to 0.

Conclusion: We have shown that our formula is valid and that it holds only for $n > 0$ and $0 \leq k \leq n \cdot (D-1)$. Hence, the theorem is proven. □

6 Computational Complexity Analysis of $\binom{n}{k}$ Formula

We present a baseline analysis of the computational complexity associated with naively computing $\binom{n}{k}$ using our formula. This analysis is based on a direct, unoptimized approach. While this offers a foundational understanding, potential optimizations might not be addressed.

6.1 Time Complexity Analysis

Analyzing the time complexity for computing $\binom{n}{k}$ using our formula, we use the representation (14):

$$\binom{n}{k} = \left\lfloor \frac{(1+2^n)^n \bmod 2^{nk+n}}{2^{nk}} \right\rfloor$$

The procedure can be delineated as follows:

1. **Modular Exponentiation**: The main component affecting the time complexity is the modular exponentiation step. Using exponentiation by squaring, the time complexity for computing $a^b \mod c$ is $O(\log b \log c)$ [11]. Given c is exponential in n:

$$T_1(n) = O((nk+n)\log n) \tag{52}$$

2. **Bit Shift Operation**: The floored division by 2^{nk} translates to a bit shift, which has a linear time complexity in terms of bit count:

$$T_2(n) = O(nk) \tag{53}$$

3. **Combining Computations**: Taking both primary operations into account, the overall time complexity is:

$$T(n) = O((nk+n)\log n) + O(nk) = O((nk+n)\log n) \tag{54}$$

However, as n grows, the nk term becomes dominant:

$$T(n) = O(nk \log n) \tag{55}$$

6.2 Space Complexity Analysis

Analyzing the space complexity for computing $\binom{n}{k}$ requires us to consider the primary memory-consuming steps:

1. **Intermediate Results**: Exponentiation by squaring requires storage for intermediate results. For modulo 2^{nk+n} computations, the maximum number of bits needed is $nk + n$. Thus, space complexity is:

$$S_1(n) = O(nk + n) \tag{56}$$

2. **Bit Shift Operation**: The bit shift operation does not require proportional additional storage, leading to:

$$S_2(n) = O(1) \tag{57}$$

3. **Miscellaneous Variables**: Auxiliary variables, including n, k, and loop counters, occupy:

$$S_3(n) = O(1) \tag{58}$$

Combining these, the overall space complexity becomes:

$$S(n) = O(nk + n) \tag{59}$$

When n grows significantly, this simplifies to:

$$S(n) = O(nk) \tag{60}$$

6.3 Comparison to Typical Approach

The typical approach for computing $\binom{n}{k}$ makes use of the factorial formula:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{61}$$

Notice that when multiplying $k!$ and $(n-k)!$, the product contains all integers from 1 to n in the sequence. Thus, many terms in the numerator $n!$ and the denominator $k!(n-k)!$ will cancel out, leaving only k terms in the numerator. This results in the simplified formula:

$$\binom{n}{k} = \prod_{j=1}^{k} \frac{n-j+1}{j} \tag{62}$$

In pseudocode:

Algorithm 1 Binomial Coefficient Computation

```
function BINOMIAL(n, k)
    c ← 1
    for j from 1 to k do
        c ← c × (n − j + 1)
        c ← c/j
    end for
    return c
end function
```

The algorithm above requires k multiplications and divisions to compute $\binom{n}{k}$. Harvey and van Der Heoven (2021) have given an algorithm for integer multiplication which has a time complexity of $O(k \log k)$ [12]. This gives the program an overall time complexity of:

$$T(n) = O(k(k \log k)) = O(k^2 \log k) \tag{63}$$

The algorithm computes $\binom{n}{k}$ directly, so the space complexity is trivially:

$$S(n) = O(k) \tag{64}$$

The time complexity of our formula compares favorably when k is large. Considering the symmetry $\binom{n}{k} = \binom{n}{n-k}$, when $k = \left\lfloor \frac{n}{2} \right\rfloor$ and is therefore as large as possible, the time complexity of both formulas simplifies to $O(n^2 \log n)$. However, our formula requires $O(n^2)$ space to compute, whereas the factorial formula requires only $O(n)$ space.

In practice, our formula may be more efficient to compute in certain cases due to the reduction in overheads associated with incrementing counters and looping. Since our formula uses exponentiation by squaring, it requires only $O(\log n)$ multiplications compared to $O(k)$ multiplications for the factorial formula. While our formula has a higher memory footprint, modern systems often have large amounts of RAM available. A practical analysis of performance and trade offs associated with both approaches is warranted.

6.4 Optimizing Our Binomial Coefficient Formula

The connection to Kronecker substitution we found in §4 reveals potential for computational optimization. Harvey (2009) [13] gives several methods to more efficiently perform multiplications using a modified "multipoint" Kronecker substitution. The techniques provided by Harvey can be applied to our formula in a straightforward manner, however the benefits of doing so have yet to be analyzed. This presents a potential avenue for further research.

7 The Problem of Efficiently Computing $\binom{n}{k} \pmod n$

Let's examine a variant our binomial coefficient formula (23):

$$\binom{n}{k} = \left\lfloor \frac{(1+2^n)^n}{2^{nk}} \right\rfloor - 2^n \left\lfloor \frac{(1+2^n)^n}{2^{nk+n}} \right\rfloor$$

An equivalent definition of the floor function is as follows [6]:

$$\left\lfloor \frac{a}{b} \right\rfloor = \frac{a}{b} - \frac{a \bmod b}{b} \tag{65}$$

By substitution, we arrive at:

$$\binom{n}{k} = \left(\frac{(1+2^n)^n}{2^{nk}} - \frac{((1+2^n)^n \bmod 2^{nk})}{2^{nk}} \right) - 2^n \left(\frac{(1+2^n)^n}{2^{nk+n}} - \frac{((1+2^n)^n \bmod 2^{nk+n})}{2^{nk+n}} \right) \tag{66}$$

Which simplifies to:

$$\binom{n}{k} = \frac{((1+2^n)^n \bmod 2^{nk+n}) - ((1+2^n)^n \bmod 2^{nk})}{2^{nk}} \tag{67}$$

From here, if n is odd, then we can distribute the mod operation for n and use modular exponentiation to efficiently compute $2^{-nk} \bmod n$. The terms which we cannot efficiently compute using modular exponentiation are:

$$\left((1+2^n)^n \bmod 2^{nk}\right) \bmod n \tag{68}$$

$$\left((1+2^n)^n \bmod 2^{nk+n}\right) \bmod n \tag{69}$$

Therefore, if we can find a way to compute the above terms efficiently, then we can calculate $\binom{n}{k}$ modulo n efficiently as well. This is a rather interesting problem of evaluating a nested modular exponentiation involving a large integer power and n as moduli. We leave the generalized version of this as an open problem.

Problem Statement: Determine an algorithm with time complexity $O(\log^c n)$ for computing:

$$(a^x \bmod b^y) \bmod n \tag{70}$$

Where $n, a, b, x, y \in \mathbb{Z}$. The terms a^x and b^y must satisfy:

$$a^x, b^y = o(2^{2^n}) \tag{71}$$

Solving this problem will enable the computation of binomial coefficients modulo n in $O(\log^c n)$ time using a variant of our formula.

8 Summary

In this paper, we have made several contributions. First, we proved a new formula for computing binomial coefficients via polynomial encoding techniques:

$$\binom{n}{k} = \left\lfloor \frac{(1+2^n)^n}{2^{nk}} \right\rfloor \bmod 2^n, \quad \text{for } n > 0 \text{ and } 0 \leq k \leq n$$

This formula depends only on basic arithmetic operations and illuminates connections between binomial coefficients, polynomial encoding, and modular arithmetic. We gave a baseline analysis of the computational complexity for the formula and compared it to typical approaches. We also left an open problem which, if resolved, would enable efficient computation of $\binom{n}{k} \pmod n$ using the formula.

Furthermore, we demonstrated the broad applicability of our approach by finding a closed form expression for the coefficients of terms in the multinomial expansion of arbitrary degree univariate unit polynomials:

$$[x^k] \left(\frac{x^D - 1}{x - 1} \right)^n = \left\lfloor \left(\frac{D^{Dn} - 1}{D^{n+k} - D^k} \right)^n \right\rfloor \bmod D^n,$$
$$\text{for } n > 0 \text{ and } 0 \leq k \leq n \cdot (D-1)$$

The formulas above stem from connecting binomial coefficients to a key, but rarely explored, property of polynomial encoding. We linked this polynomial property to an established technique, Kronecker substitution, and showed how this linkage presents opportunities for additional research into computational optimizations.

References

[1] Cook J. Polynomial determined by two inputs. `https://johndcook.com/blog/2012/03/27/polynomial-trick/`, 2012. Accessed: 2023-08-05.

[2] Rupinski A. Mathoverflow: Application of polynomials with non-negative coefficients. `https://mathoverflow.net/questions/91827/`, 2012. Accessed: 2023-08-05.

[3] James Stewart. *Calculus: Early Transcendentals*. Cengage Learning, 2007.

[4] Kenneth H Rosen. *Discrete Mathematics and Its Applications*. McGraw-Hill Science/Engineering/Math, 2011.

[5] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT press, 2009.

[6] Ivan Niven, Herbert S Zuckerman, and Hugh L Montgomery. *An Introduction to the Theory of Numbers*. John Wiley & Sons, 2008.

[7] Ralph P Grimaldi. *Discrete and Combinatorial Mathematics*. Pearson Education India, 2004.

[8] Ronald L. Graham, Donald Ervin Knuth, and Oren Patashnik. *Concrete mathematics: a foundation for computer science*. Addison-Wesley Professional, 1994.

[9] Richard A. Brualdi. *Introductory Combinatorics*. Pearson, 2010.

[10] David W. Wilson. Entry a023037 in the on-line encyclopedia of integer sequences. `https://oeis.org/A023037`, 2023. Accessed: 2023-08-07.

[11] Neal Koblitz. *A Course in Number Theory and Cryptography*. Springer-Verlag, 2nd edition, 1994.

[12] Joris van Der Hoeven David Harvey. Integer multiplication in time o(n log n). *Annals of Mathematics*, 2021. doi: 10.4007/annals.2021.193.2.4.hal-02070778v2.

[13] David Harvey. Faster polynomial multiplication via multipoint kronecker substitution. *Journal of Symbolic Computation*, 44, 2009. doi: 10.1016/j.jsc.2009.05.004.