

Why a combination, nC_k , is always a natural number? Different way to look at the answer accentuates the generation process of factors and prime numbers contained within a particular range of 1 to n .

Amit Kumar

1 Abstract

The concept of combination is crucial to layout groundwork for many fundamental sciences, finance and technical domains. Despite having countless empirical observations that a combination (for a 'chosen n and k ') is always a natural number, a direct proof (inductive or non-inductive) that a combination is always integral, is not rigorously demonstrated in the literature. This article discusses why a combination nC_k will always be a natural number and never fractional or irrational. This article separates an nC_k into mutually exclusive and collectively exhaustive cases and then shows for each case that any term in a denominator have at least one corresponding multiple in the numerator. All k for a particular n (for an nC_k) are visualized through plots to gain better understanding. In the context of the plots, this different way of looking at a Combination nC_k , gives hindsights into the generation process of factors and prime numbers contained within a particular range of 1 to n .

2 Introduction

From countless ages, it has been desired to know the number of ways an arrangement can be obtained or selected out of many possible ways. The quest has travelled from ancient times and has captured the imagination of the great mathematicians like Bhaskara-II[1], Blaise Pascal[2], Pierre-de-Fermat[3], Leonhard Euler[4] to name a very few.

Permutations and Combinations have become fundamental concepts with wide ranging applications in almost every sphere of academic studies or technical endeavors. While a permutation, nP_k , tells how many number of ways k objects can be arranged out of n objects, if we are interested in *exact order of the chosen k items* [6].

$${}^nP_k = \frac{n!}{(n-k)!} \text{ where } n, k \in \mathbb{N} \text{ and } 1 \leq k \leq n \quad (1)$$

e.g. Number of possibilities to choose 5 items out of 9 choices, we will write as ${}^9P_5 = \frac{9!}{(9-5)!}$ (see a nice introduction [7]). It is easy to show that $n!$ (in the numerator) can be broken into a multiplication constituting two parts: one going from n to $(n-k+1)$ and other going from $(n-k)$ to 1. The denominator can also be written as multiplication of numbers from $(n-k)$ to 1 as below.

$$\begin{aligned} {}^nP_k &= \frac{n!}{(n-k)!} \\ &= \frac{(n) \times (n-1) \cdots \times (n-k+1) \times (n-k) \times (n-k-1) \cdots \times 3 \times 2 \times 1}{(n-k) \times (n-k-1) \times (n-k-2) \cdots \times 3 \times 2 \times 1} \\ &= (n) \times (n-1) \cdots \times (n-k+1) \end{aligned} \quad (2)$$

It is evident that the terms in **blue fonts** will cancel each other, in the numerator and the denominator, and we will be left with a product of integer sequence going from n to $(n-k+1)$. Since each participating number in the product is a natural number, the net product is also a natural number.

Combination, nC_k represents number of ways k elements out of n can be chosen, *without any regard to order between the chosen items* [6]. Below are equivalent representations of a combination.

$${}^nC_k = \binom{n}{k} = C(n, k) \quad (3)$$

$$= \frac{{}^nP_k}{k!} \quad (4)$$

$$= \frac{n!}{(n-k)! \times (k)!} \quad (5)$$

where $n!$ means factorial of n and is defined as $n \times (n-1) \times (n-2) \cdots 3 \times 2 \times 1$. Some striking correlations are definitely there such as the one using Pascal's triangle. The n 'th row of the Pascal's triangle [5] is stacked rows of expansion of Binomial Coefficients, where each k 'th coefficient (in the n 'th row) is the value of a combination nC_k (see Table 1 and footnote¹).

$$\begin{array}{cccccc} & & 1 & & & \\ & 1 & 2 & 1 & & \\ & 1 & 3 & 3 & 1 & \\ & 1 & 4 & 6 & 4 & 1 \\ 1 & 5 & 10 & 10 & 5 & 1 \end{array} \parallel \begin{array}{cccccccccccc} & & & & & & {}^1C_0 & & & & & \\ & & & & & & {}^2C_0 & {}^2C_1 & {}^2C_2 & & & \\ & & & & & & {}^3C_0 & {}^3C_1 & {}^3C_2 & {}^3C_3 & & \\ & & & & & & {}^4C_0 & {}^4C_1 & {}^4C_2 & {}^4C_3 & {}^4C_4 & \\ & & & & & & {}^5C_0 & {}^5C_1 & {}^5C_2 & {}^5C_3 & {}^5C_4 & {}^5C_5 \end{array}$$

Table 1: Correlation between Pascal’s triangle (left) to Combinations (right)

Given the Pascal's triangle it is easy to observe and inductively prove [6] that

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \quad (6)$$

or

$${}^nC_k = {}^{n-1}C_{k-1} + {}^{n-1}C_k, \text{ where } 0 < k < n \text{ } (\because {}^nC_0 = {}^nC_n = 1, \text{ need not be computed. [6]}) \quad (8)$$

It can also be observed in table 1 that the combinations (and the Pasacal's triangle) have a symmetry:

$$\begin{aligned} {}^nC_k &= \frac{n!}{(n-k)! \times (k)!} \\ {}^nC_{(n-k)} &= \frac{n!}{(n-(n-k))! \times (n-k)!} \\ &= \frac{n!}{(k)! \times (n-k)!} \\ &= \frac{n!}{(n-k)! \times (k)!} \\ \therefore {}^nC_k &= {}^nC_{(n-k)} \end{aligned}$$

The results (value of combinations) have been observed to be integers (empirically, over and over again based on calculations) for a particular choice of n and k . The focus of this paper is a proof that holds, irrespective of any choice of n and k , and one that transcends manual calculations everytime to show the integral nature. We can think of some known indirect proofs that ‘show’ the combination is integral. *e.g.* We have established above that a permutation, nP_k , is integral. Now by the definition of equation (4), we can say the below:

$${}^n P_k = {}^n C_k \times (k)!$$

\therefore L.H.S. is known to be an integer (see equation (2)), the R.H.S must also be an integer and is shown below.

${}^nC_k \times (k)! = {}^nC_k \times$ an integer value ($\because (k)!$ will always be an integer.)

\therefore to make the R.H.S. an integer the combination ‘must’ be an integer as well. ■

The proof looks promising but arguments like the below dent it. **‘If** the $(k)!$ is an even number **and** nC_k is a fraction number of the nature $x.5$ where x can be any integer, then the multiplication of ${}^nC_k \times (k)!$ will still be an integer. However it is trivial to see that the combination is definitely not integral in ‘this hypothetical’ example.

On similar lines, we can also make use of equation (6) and ‘say’ that by induction on ‘n’ the integrality of the combination ‘will’ hold, subjected to the following base conditions:

- $\binom{n}{k} = 0$ if $n < k$, premise-1
- $\binom{n}{k} = 1$ if $k = 0$ or $k = n$, premise-2

In these inductive steps, all is fine, if $k \geq n$, since the answer is either 0 or 1. However as soon as $n > k > 0$, we go ahead and assume for one of the two terms that it will be an integer. This is like putting the cart ahead of the horse since we set out to prove that ${}^nC_k = \binom{n}{k}$, will always be an integer in the first place. Let us see below

¹Please note that ${}^1C_0 = {}^1C_1 = 1$ as there is exactly one way how 0 or n items can be chosen out of a total n items

what we mean by this statement. Let's try to calculate 7C_3 using the induction by starting at $n = 0$. We can calculate the final value easily 'if we do the calculations manually' as below.

$$\begin{aligned}
\binom{1}{3} &= \binom{0}{3} + \binom{0}{2} = 0 + 0 = 0 \because \binom{0}{k} = 0, \text{ by premise-1} \\
\binom{2}{3} &= \binom{1}{3} + \binom{1}{2} = 0 + 0 = 0, \text{ by premise-1} \\
\binom{3}{3} &= \binom{2}{3} + \binom{2}{2} = 0 + 1 = 1 \left[\binom{2}{2} = 1 \because \binom{n}{n} = 1, \text{ by premise-2} \right] \\
\binom{4}{3} &= \binom{3}{3} + \binom{3}{2} = 1 + 3 = 4 \text{ See the discussion in footnote for this and next 3 steps}^2 \\
\binom{5}{3} &= \binom{4}{3} + \binom{4}{2} = 4 + 6 = 10 \\
\binom{6}{3} &= \binom{5}{3} + \binom{5}{2} = 10 + 10 = 20 \\
\binom{7}{3} &= \binom{6}{3} + \binom{6}{2} = 20 + 15 = 35
\end{aligned}$$

7C_3 is indeed 35 but please note that the only thing we could be sure that the L.H.S will be equal to R.H.S. At no stage it was implicit that the R.H.S will be an integer once $n > k > 0$ and since we were trying to prove that a combination is always integer, we cannot assume itself during our formative steps. Thus we can conclude that a direct, inductive or non-inductive and algebraic, proof that the value of combination will 'always' be an integer for 'any choice of n and k ', is not '**explicitly**' stated in the literature.

While a rigorous, inductive algebraic proof establishing that combinations are integral and only integral in nature, may prove to be hard, it should be possible to prove the same by non-inductive blocks of steps. This paper aims to fill the gap and demonstrate non-inductively and algebraically that for each denominator term there is at least one term in the numerator that is a multiple of it. Thus all the terms in the denominator will be cancelled and the final value of a combination will 'always collapse to an integer (more correctly a natural number)'. At the end of it we visualize our results through graphs and we find an interesting but also a simple connection with the generation process of factors and prime numbers contained within the range 1 to n . We conclude this article by showing that prime numbers do not arise randomly but they have a simple and systematic generation algorithm. To show effectiveness of this algorithm we start with a **single prime number, 2**, and generate the first 1000 prime numbers. These **generated** 1000 primes match exactly the first 1000 **known** prime numbers.

3 Proofs

3.1 Why Combination nC_k is a natural number?

Combination is the number of arrangements, when the order *does not* matter and is given by [6]:

$${}^nC_k = \frac{n!}{(n-k)! \times (k)!} \text{ where } n, k \in \mathbb{N} \text{ and } 1 \leq k \leq n \quad (9)$$

This time we can cancel the denominator's $k!$

$$\begin{aligned}
{}^nC_k &= \frac{(n) \times (n-1) \cdots \times (k+1) \times \textcolor{blue}{(k)} \times \textcolor{blue}{(k-1)} \cdots \times \textcolor{blue}{3} \times \textcolor{blue}{2} \times \textcolor{blue}{1}}{(n-k) \times (n-k-1) \cdots \times 3 \times 2 \times 1 \times \textcolor{blue}{(k)} \times \textcolor{blue}{(k-1)} \cdots \times \textcolor{blue}{3} \times \textcolor{blue}{2} \times \textcolor{blue}{1}} \\
&= \frac{(n) \times (n-1) \cdots \times (k+1)}{(n-k) \times (n-k-1) \cdots \times 3 \times 2 \times 1}
\end{aligned}$$

Thus it can be written down as the below (after cancelling common terms shown in **blue fonts**):

$${}^nC_k = \frac{(n) \times (n-1) \cdots \times (k+2) \times (k+1)}{(n-k) \times (n-k-1) \cdots \times 3 \times 2 \times 1} \quad (10)$$

²To be sure that $\binom{3}{2}$ is an integer we have to calculate it, since this is outside the base condition. Equality $\binom{4}{3} = \binom{3}{3} + \binom{3}{2}$ does not 'necessarily' imply integer nature. This holds for $\binom{4}{2}, \binom{5}{2}, \binom{5}{3}, \binom{6}{3}, \binom{6}{2}, \binom{7}{3} \cdots$ and so on

3.1.1 Different possible values of n and k w.r.t each other.

If $n//2$ is defined as integer portion of division of n by 2, then we can see that one of the two statements, below, must be true:

$$n = (n//2 \times 2) \text{ where } n \text{ is an even number} \quad (11)$$

$$n = (n//2 \times 2) + 1 \text{ where } n \text{ is an odd number} \quad (12)$$

On a similar note, k can be defined as six and six only possibilities w.r.t n :

$$k = n//2 \text{ where } n \text{ is an even number} \quad (13)$$

$$k = n//2 - l \text{ where } n \text{ is an even number and } l \in \mathbb{N}, 1 \leq l \leq n//2 \quad (14)$$

$$k = n//2 + m \text{ where } n \text{ is an even number and } m \in \mathbb{N}, 1 \leq m \leq n//2 \quad (15)$$

$$k = n//2 \text{ where } n \text{ is an odd number} \quad (16)$$

$$k = n//2 - l \text{ where } n \text{ is an odd number and } l \in \mathbb{N}, 1 \leq l \leq (n-1)/2 \quad (17)$$

$$k = n//2 + m \text{ where } n \text{ is an odd number and } m \in \mathbb{N}, 1 \leq m \leq (n+1)/2 \quad (18)$$

Let us take a few concrete examples to understand the equations from equation (11) to equation (18).

¹⁰ C_0 means n is even, $k = 0$ and $l = n//2 = 5$: Thus equations (11) and (14) are realized.

¹⁰ C_2 means n is even, $k = 2$ and $l = 3$: Thus equations (11) and (14) are realized.

¹⁰ C_4 means n is even, $k = 4$ and $l = 1$: Thus equations (11) and (14) are realized.

¹⁰ C_5 means n is even and $k = n//2 = 5$: Thus equations (11) and (13) are realized.

¹⁰ C_6 means n is even, $k = 6$ and $m = 1$: Thus equations (11) and (15) are realized.

¹⁰ C_8 means n is even, $k = 8$ and $m = 3$: Thus equations (11) and (15) are realized.

¹⁰ C_{10} means n is even, $k = 10$ and $m = n//2 = 5$: Thus equations (11) and (15) are realized.

¹¹ C_0 means n is odd, $k = 0$ and $l = (n-1)/2 = 5$: Thus equations (12) and (17) are realized.

¹¹ C_2 means n is odd, $k = 2$ and $l = 3$: Thus equations (12) and (17) are realized.

¹¹ C_4 means n is odd, $k = 4$ and $l = 1$: Thus equations (12) and (17) are realized.

¹¹ C_5 means n is odd and $k = n//2 = 5$: Thus equations (12) and (16) are realized.

¹¹ C_6 means n is odd, $k = 6$ and $m = 1$: Thus equations (12) and (18) are realized.

¹¹ C_8 means n is odd, $k = 8$ and $m = 3$: Thus equations (12) and (18) are realized.

¹¹ C_{11} means n is odd, $k = 11$ and $m = (n+1)/2 = 6$: Thus equations (12) and (18) are realized.

We can clearly see based on equations (13) to (18) that there are six possibilities that we separately need to take care and prove. In the following subsections, 3.1.3 to 3.1.8, we will prove each of the above condition.

3.1.2 Setting up the problem

Looking at the equation (10) it is clear that each term in the denominator can be written as $(n-k-x)$, where $x \in \mathbb{W}, 0 \leq x \leq (n-k-1)$ and each term in the numerator can be written as $(k+y)$, where $y \in \mathbb{N}, 1 \leq y \leq (n-k)$. Formalizing the two equations as below:

$$\text{denominator} = \prod_0^{n-k-1} (n-k-x), \text{ where } n \text{ and } k \text{ are constants and } x \text{ is the variable} \quad (19)$$

$$\text{numerator} = \prod_1^{n-k} (k+y), \text{ where } k \text{ is a constant and } y \text{ is the variable} \quad (20)$$

Now we have to show that for every $(n-k-x)$, there exists a natural number a such that $a \times (n-k-x) = (k+y)$ holds true. i.e.

$$a \times (n-k-x) = (k+y), \text{ where } a \in \mathbb{N} \quad (21)$$

3.1.3 nC_k is always a natural number when equation (11) and (13) hold true

$$\begin{aligned}
a \times (n - k - x) &= (k + y) \\
a \times (2 \times n/2 - n/2 - x) &= (n/2 + y) \\
a \times (n/2 - x) &= (n/2 + y) \\
a &= \frac{(n/2 + y)}{(n/2 - x)} \\
y &= a \times (n/2 - x) - n/2 \\
y &= (a - 1) \times n/2 - a \times x
\end{aligned} \tag{22}$$

Let us take a couple of examples.

For ${}^{10}C_5$ and $x = 1$ we get the solution as below

$$\begin{aligned}
y &= (a - 1) \times 5 - a \\
y &= a \times (5 - 1) - 5 \\
y &= a \times 4 - 5
\end{aligned}$$

Clearly if $a = 2$ we get a valid solution and we obtain $y = 3$. Let us see what this means. This means that for denominator $(n - k - x) = (10 - 5 - 1) = 4$, we will get a corresponding numerator which will have its factor as $(k + y) = (5 + 3) = 8$. This is consistent. Let us take another example.

For ${}^{10}C_5$ and $x = 3$ we get the solution as below

$$\begin{aligned}
y &= (a - 1) \times 5 - a \times 3 \\
y &= a \times (5 - 3) - 5 \\
y &= a \times 2 - 5
\end{aligned}$$

Clearly if $a = 3$ we get a valid solution and then $y = 1$. This result can be interpreted as before. This means that for the denominator $(n - k - x) = (10 - 5 - 3) = 2$, we will get a corresponding numerator which will have its factor as $(k + y) = (5 + 1) = 6$. This is also consistent.

For (22), we know that $x \in \mathbb{W}, 0 \leq x \leq (n - k - 1)$ and $y \in \mathbb{N}, 1 \leq y \leq (n - k)$. These two conditions get simplified due to (13) as $0 \leq x \leq (n/2 - 1)$. Let us put extreme conditions of x to see if we can obtain a valid a for the denominator for every y .

$$\begin{aligned}
y &= (a - 1) \times n/2 - a \times 0 \text{ this is due to (22)} \\
y &= (a - 1) \times n/2
\end{aligned}$$

Clearly an a can be chosen such that R.H.S is a natural number. Similarly we can say that,

$$\begin{aligned}
y &= (a - 1) \times n/2 - a \times (n/2 - 1) \text{ this is due to (22)} \\
y &= \cancel{a \times n/2} - n/2 - \cancel{a \times n/2} + a \\
y &= a - n/2
\end{aligned}$$

Clearly an a can be chosen such that R.H.S is a natural number. Thus we saw that an a is available in the range of extreme x values and thus we always have an a so that every denominator is a factor for some numerator. ■

3.1.4 nC_k is always a natural number when equation (11) and (14) hold true

$$\begin{aligned}
a \times (n - k - x) &= (k + y) \\
a \times (2 \times n/2 - n/2 + l - x) &= (n/2 - l + y) \\
a &= \frac{(n/2 - l + y)}{(n/2 + l - x)} \\
y &= (a - 1) \times n/2 + l \times (a + 1) - a \times x
\end{aligned} \tag{23}$$

How do we interpret this? Let us see an example.

${}^{10}C_4$. Here $n = 10$ and thus $n/2 = 5, k = 4, l = n/2 - k = 5 - 4 = 1$. Furthermore let us assume that $x = 1$, i.e. the denominator is $(n - k - 1)$ for which the factor containing numerator has to be found.

Putting the above values in (23), we get $y = (a - 1)5 + 1(a + 1) - a \times 1 = (a - 1)5 + 1$

Clearly $a = 2$ will solve the equation which is consistent with restrictions on y and the value of y will be $y = (2 - 1) \times 5 + 1 = 6$. That means the corresponding multiples will reside in the numerator $(k + y) = (k + 6)$. Putting all the values in equation (21) should get us no contradiction.

$$\begin{aligned} a \times (n - k - x) &= (k + y) \\ 2 \times (10 - 4 - 1) &= (4 + 6) \\ 2 \times (5) &= (10) \\ 10 &= 10 \end{aligned}$$

Please note that $a = 1$ will also give a consistent solution as $y = (1 - 1) \times 5 + 1 = 1$, which means $(k + 1)$ is the desired numerator. Please note that $(n - k - x) = (10 - 4 - 1) = 5$ will have two numerators where the multiples reside, one in $(k + 1) = 5$ and other in $(k + 6) = 10$. We will have an additional example as ${}^{10}C_4$ and $x = 3$. i.e. the denominator is $(n - k - 3)$ and the corresponding numerator, $(k + y)$, has to be found. Using (23), we get $y = (a - 1)5 + 1(a + 1) - a * 3 = (a - 1)5 + 6$

. Again $a = 3$ will solve the equation and y will be $y = (3 - 1) \times 5 + 1 + 3 * 3 = 2$. That means the corresponding factor will reside in the numerator $(k + y) = (k + 2)$.

. To see if it is consistent, we will have to put the values in (21) and we will find that the equality holds.

$$\begin{aligned} a \times (n - k - x) &= (k + y) \\ 2 \times (10 - 4 - 3) &= (4 + 2) \\ 2 \times (3) &= (6) \\ 6 &= 6 \end{aligned}$$

Now according to (19),

$$\begin{aligned} 0 \leq x \leq (n - k - 1) &= 0 \leq x \leq (2 \times n/2 - n/2 + l - 1) \\ &= 0 \leq x \leq (n/2 + l - 1) \end{aligned} \quad (24)$$

We will show that the equations are consistent at the extremes of x and therefore must be true for other values of x in between.

Choosing $x = 0$ in the equation (23), we get

$$\begin{aligned} y &= (a - 1) \times n/2 + l \times (a + 1) - a \times x \\ y &= (a - 1) \times n/2 + l \times (a + 1) - a \times 0 \\ y &= (a - 1) \times n/2 + l \times (a + 1) \end{aligned}$$

Since l is $a \in \mathbb{N}$, so the minimum value l can have is 1. Thus the second term in the R.H.S. above will be positive. To make y as positive integer, there is always a value of a we can choose to satisfy this restriction.

At the other extreme, choosing $x = (n/2 + l - 1)$, we get the below.

$$\begin{aligned} y &= (a - 1) \times n/2 + l \times (a + 1) - a \times (n/2 + l - 1) \\ y &= a \times n/2 - n/2 + l \times a + l - a \times n/2 - a \times l + a \\ y &= \cancel{a \times n/2} - n/2 + \cancel{l \times a} + l - \cancel{a \times n/2} - \cancel{a \times l} + a \\ y &= a - n/2 + l \end{aligned}$$

Again it is evident that an a can be chosen such that y can be in a form consistent with assumptions. Thus we have shown that the equation (23) is consistent and solvable at extremes of x and thus must be solvable between. ■

3.1.5 nC_k is always a natural number when equation (11) and (15) hold true

$$\begin{aligned} a \times (n - k - x) &= (k + y) \\ a \times (2 \times n/2 - n/2 - m - x) &= (n/2 + m + y) \\ a &= \frac{(n/2 + m + y)}{(n/2 - m - x)} \\ y &= (a - 1) \times n/2 - m \times (a + 1) - a \times x \end{aligned} \quad (25)$$

Interpreting this result is as before through an example.

${}^{10}C_6$. Here $n = 10$ and thus $n/2 = 5, k = 6, m = k - n/2 = 6 - 5 = 1$. Furthermore let us assume that $x = 1$,

i.e. the denominator is $(n - k - 1)$ for which the factor containing numerator has to be found.

. Substituting the above values in (25), we get $y = (a-1)5 - m(a+1) - a*1 = (a-1)5 - 1(1+a) - a = (a-1)5 - 2a - 1$. It is easy to see that the equation is solved for $a = 3$. In such a case y will be equal to $(3-1)*5 - 2*3 - 1 = 3$. i.e. for $(n - k - 1)$, there is a multiple in $(k + 3)$, where $n = 10$ and $k = 6$. It is trivial to verify this.

$$\begin{aligned} a \times (n - k - x) &= (k + y) \\ 3 \times (10 - 6 - 1) &= (6 + 3) \\ 3 \times (3) &= (9) \\ 9 &= 9 \end{aligned}$$

As per (19),

$$\begin{aligned} 0 \leq x \leq (n - k - 1) &= 0 \leq x \leq (2 \times n/2 - n/2 - m - 1) \\ &= 0 \leq x \leq (n/2 - m - 1) \end{aligned} \quad (26)$$

We need to show that there is no contradiction at the extremes (25) and thus it will hold true for middle values as well. Choosing $x = 0$ in equation (25) we arrive at,

$$\begin{aligned} y &= (a - 1) \times n/2 - m \times (a + 1) - a \times x \\ y &= (a - 1) \times n/2 - m \times (a + 1) - a \times 0 \\ y &= (a - 1) \times n/2 - m \times (a + 1) \end{aligned}$$

The above should hold for $y \geq 1$. Thus

$$\begin{aligned} (a - 1) \times n/2 &\geq 1 + m \times (a + 1) \\ a \times n/2 - m \times a &\geq (1 + m + n/2) \\ a &\geq \frac{(1 + m + n/2)}{(n/2 - m)} \end{aligned}$$

The last equation is defined as long as $n/2 \geq m$. This constraint is already in place due to the premise of (15). Thus a value of a can be computed such that (25) is satisfied.

Working at the other extreme of $x = (n/2 - m - 1)$ we observe the below.

$$\begin{aligned} y &= (a - 1) \times n/2 - m \times (1 + a) - a \times (n/2 - m - 1) \\ y &= a \times n/2 - n/2 - m - m \times a - a \times n/2 + a \times m + a \\ y &= \cancel{a \times n/2} - n/2 - m - \cancel{m \times a} - \cancel{a \times n/2} - \cancel{a \times m} + a \\ y &= (a - n/2) - m \end{aligned}$$

As seen, y has allowed solutions at both extremes as there is always an a that can be chosen to satisfy the constraints. ■

3.1.6 nC_k is always a natural number when equation (12) and (16) hold true

$$\begin{aligned} a \times (n - k - x) &= (k + y) \\ a \times (2 \times n/2 + 1 - n/2 - x) &= (n/2 + y) \\ a &= \frac{(n/2 + y)}{(n/2 + 1 - x)} \\ y &= (a - 1) \times n/2 + a - a \times x \end{aligned} \quad (27)$$

The best way to understand the above equation, as before, is through an example.

In ${}^{11}C_5$, $n = 11$, $n/2 = 5 = k$. Let x be equal to 1, then substituting the values in the above equation, we get,

$$\begin{aligned} y &= (a - 1) \times 5 + a - a \times 1 \\ y &= (a - 1) \times 5 + \cancel{a} - \cancel{a \times 1} \\ y &= (a - 1) \times 5 \end{aligned}$$

Clearly $a = 2$ will give a solution and y will be equal to 5. That means for the above setup for $(n - k - 1)$, there will be a factor in $(k + 5)$. It is easy to verify this.

$$\begin{aligned}
a \times (n - k - x) &= (k + y) \\
2 \times (11 - 5 - 1) &= (5 + 5) \\
2 \times (5) &= (10) \\
10 &= 10
\end{aligned}$$

Just like before we can appreciate that the equation (27) has boundaries for extreme values of x . Thus if we show the lower extreme and upper extreme values of x to have a consistent solution, then the proof must hold for values in between.

We remember that $0 \leq x \leq (n - k - 1) = 0 \leq x \leq (2 \times n/2 + 1 - n/2 - 1) = 0 \leq x \leq n/2$. First we will check for the lower limit of 0. Putting 0 in equation (27), we get,

$$\begin{aligned}
y &= (a - 1) \times n/2 + a - a \times x \\
y &= (a - 1) \times n/2 + a - a \times 0 \\
y &= (a - 1) \times n/2 + a
\end{aligned}$$

Clearly a value of a can be chosen such that R.H.S satisfies the constraint that $y \in \mathbb{N}$. When we think of the other extreme, we obtain the following.

$$\begin{aligned}
y &= (a - 1) \times n/2 + a - a \times x \\
y &= (a - 1) \times n/2 + a - a \times (n - k - 1) \\
y &= (a - 1) \times n/2 + a - a \times (2 \times n/2 + 1 - n/2 - 1) \\
y &= (a - 1) \times n/2 + a - a \times (n/2) \\
y &= a - n/2
\end{aligned}$$

Here also it is clear that a value of a can be chosen such that the R.H.S is greater than or equal to 1.

Thus we have shown that a valid y can be obtained by choosing an appropriate a , for two extreme values of x . ■

3.1.7 nC_k is always a natural number when equation (12) and (17) hold true

$$\begin{aligned}
a \times (n - k - x) &= (k + y) \\
a \times (2 \times n/2 + 1 - n/2 + l - x) &= (n/2 - l + y) \\
a &= \frac{(n/2 - l + y)}{(n/2 + 1 + l - x)} \\
y &= (a - 1) \times n/2 + l \times (a + 1) - a \times x + a
\end{aligned} \tag{28}$$

Let us find the value for an example. Suppose we are given ${}^{11}C_4$. Here $n = 11$, $n/2 = 5$, $k = 4$, $l = 1$ and let's say $x = 1$. Then putting these values in the above equation, we get:

$$\begin{aligned}
y &= (a - 1) \times n/2 + l \times (a + 1) - a \times x + a \\
y &= (a - 1) \times 5 + 1 \times (a + 1) - a \times 1 + a \\
y &= (a - 1) \times 5 + \cancel{1 \times a} + 1 - \cancel{a \times 1} + a \\
y &= (a - 1) \times 5 + 1 + a
\end{aligned}$$

Clearly $a = 1$ gives a solution and thus $y = 2$. This argues that for $(n - k - 1)$, there is a factor in $(k + 2)$ for this example. i.e. for $(11 - 4 - 1)$ we have a factor in $(4 + 2)$. It is evident that 6 is on both sides of the equation.

Now we have to show that given an x and l and n , we can get a y such that $y \in \mathbb{N}$. We will show for two extreme values of x , it should hold true for values in between.

Let's recall that $0 \leq x \leq (n - k - 1) = 0 \leq x \leq (2 \times n/2 + 1 - n/2 + l - 1) = 0 \leq x \leq (n/2 + l)$. Starting with $x = 0$ and substituting $x = 0$ in equation (28).

$$\begin{aligned}
y &= (a - 1) \times n/2 + l \times (a + 1) - a \times x + a \\
y &= (a - 1) \times n/2 + l \times (a + 1) - a \times 0 + a \\
y &= (a - 1) \times n/2 + l \times (a + 1) + a
\end{aligned}$$

In the above each term on the right hand side is a positive integer and therefore the net value of y is also going to be a positive integer and thus the restriction that $1 \leq y, y \in \mathbb{N}$
Putting the other extreme value of x in equation (28)

$$\begin{aligned} y &= (a-1) \times n/2 + l \times (a+1) - a \times x + a \\ y &= (a-1) \times n/2 + l \times (a+1) - a \times (n/2 + l) + a \\ y &= a \times n/2 - n/2 - l \times a + l - a \times n/2 - a \times l + a \\ y &= \cancel{a \times n/2} - n/2 - \cancel{l \times a} + l - \cancel{a \times n/2} - \cancel{a \times l} + a \\ y &= a - n/2 + l \end{aligned}$$

Given the limits on l as per equation (17), a value of l can be found such that a consistent value of y can be obtained. ■

3.1.8 nC_k is always a natural number when equation (12) and (18) hold true

$$\begin{aligned} a \times (n - k - x) &= (k + y) \\ a \times (2 \times n/2 + 1 - n/2 - m - x) &= (n/2 + m + y) \\ a &= \frac{(n/2 + m + y)}{(n/2 + 1 - m - x)} \\ y &= (a-1) \times n/2 - m \times (a+1) - a \times x + a \end{aligned} \quad (29)$$

Let us see the equation (29) in action.

Suppose it is given that we have to work on ${}^{11}C_6$, then $n = 11, n/2 = 5, k = 6, m = 1$. Furthermore suppose $x = 3$, i.e. we have to find the multiple of $(n - k - 3)$ in the numerator.

$$\begin{aligned} y &= (a-1) \times n/2 - m \times (a+1) - a \times x + a \\ y &= (a-1) \times 5 - 1 \times (a+1) - a \times 3 + a \\ y &= (a-1) \times 5 - 1 - 3 \times a \\ y &= 5 \times a - 5 - 1 - 3 \times a \\ y &= 2 \times a - 6 \end{aligned}$$

Clearly there is a solution for $a = 4$. Once we have obtained, a then we can plug it in equation (29) to obtain the value of y . We get,

$$\begin{aligned} y &= (a-1) \times n/2 - m \times (a+1) - a \times x + a \\ y &= (4-1) \times 5 - 1 \times (4+1) - 4 \times 3 + 4 \\ y &= 15 - 5 - 12 + 4 \\ y &= 2 \end{aligned}$$

Thus for ${}^{11}C_6$, for the numerator $(n - k - 3)$, there is a factor in $(k + 2)$. It is easy to verify this using (21). Putting the values:

$$\begin{aligned} 4 \times (11 - 6 - 3) &= (6 + 2) \\ 4 \times 2 &= 8 \end{aligned}$$

Thus we arrive at a consistent solution.

What is left to be shown is that given an x, m and n , we can always get a y such that $y \in \mathbb{N}$. Since x has limits in equation (29), it suffices to show that a consistent y can be obtained always if consistent y can be obtained at the lower and upper extremes.

Now $0 \leq x \leq (n - k - 1) = 0 \leq x \leq (2 \times n/2 + 1 - n/2 - m - 1) = 0 \leq x \leq (n/2 - m)$. Given this premise, we first substitute the lower limit of x which is 0.

$$\begin{aligned} y &= (a-1) \times n/2 - m \times (a+1) - a \times x + a \\ y &= (a-1) \times n/2 - m \times (a+1) - a \times 0 + a \\ y &= (a-1) \times n/2 - m \times (a+1) + a \\ y &= a \times (n/2 - m + 1) - (n/2 + m) \end{aligned}$$

So as long as $a \times (n//2 - m + 1) - (n//2 + m) \geq 1$, the RHS will be a natural number and consistent y can be obtained.

$$\begin{aligned} a \times (n//2 - m + 1) - (n//2 + m) &\geq 1 \\ a \times (n//2 - m + 1) &\geq (n//2 + m + 1) \\ a &\geq \frac{(n//2 + m + 1)}{(n//2 - m + 1)} \end{aligned} \quad (30)$$

Thus for the above to hold true $(1 + n//2 - m) \geq 0$.
Due to constraints imposed by (3.1.8)

$$\begin{aligned} m &\leq (n + 1)/2 \\ m &\leq (2 \times n//2 + 1 + 1)/2 \\ m &\leq (2 \times n//2 + 2)/2 \\ m &\leq (n//2 + 1) \\ 0 &\leq (1 + n//2 - m) \end{aligned}$$

So the requirement is already satisfied for (30) and thus for $x = 0$ it is always possible to have a $y \in \mathbb{N}$. Now let us see the other extreme of x which is $(n//2 - m)$. Putting this value in equation (29).

$$\begin{aligned} y &= (a - 1) \times n//2 - m \times (a + 1) - a \times x + a \\ y &= (a - 1) \times n//2 - m \times (a + 1) - a \times (n//2 - m) + a \\ y &= a \times n//2 - n//2 - m \times a - m - a \times n//2 + a \times m + a \\ y &= \cancel{a \times n//2} - n//2 - \cancel{m \times a} - m - \cancel{a \times n//2} + \cancel{a \times m} + a \\ y &= (a - n//2) - m \end{aligned}$$

If $(a - n//2) - m \geq 1$, the RHS will always be a positive integer and a consistent y can be obtained.

$$\begin{aligned} (a - n//2) - m &\geq 1 \\ a &\geq (n//2 + m + 1) \end{aligned} \quad (31)$$

Since the R.H.S is a positive integer it is always possible to get such a value of a and thus y can be obtained for the other extreme of x , as well. ■

4 Visualizing the relationship between x, y and a

For (22), (23), (25), (27), (28) and (29), we can have plots, where for each plot x (as contained in the term $n - k - x$) is on the abscissa while y (as contained in the term $k + y$) is on ordinate for different values of nC_k (with implicit l or m if applicable). The *python3* code for these plots (Figure 1-12) is given in appendix A. Let's take an example ${}^{25}C_7$ and then expand on it further to summarise the 6 equations mentioned above. For the discussion in this sub-section, please refer to the Figure 13.

Visualizing x : Each x is attached to a value in the denominator and the value of x increases from left to right having the limits $0 \leq x \leq (n - k - 1)$. Thus in the attached figure, the value of x goes from 0 to $(25 - 7 - 1) = 17$. Using any value of x , a corresponding denominator term can be created by substituting the value of x in $(n - k - x)$. For example a value of $x = 6$ means that it corresponds to the denominator value of $25 - 7 - 6 = 12$.

Visualizing y : Each y is associated with a value in the numerator and the value of y increases from bottom to top having the limits $1 \leq y \leq (n - k)$. Thus in the attached figure, the value of y goes from 1 to $(25 - 7) = 18$. Using any value of y , a corresponding numerator term can be created by substituting the value of y in $(k + y)$. For example a value of $y = 17$ means that it corresponds to the numerator value of $7 + 17 = 24$.

Visualizing a : A particular a is generated by an intersection of a particular x and y and tells that a specific denominator (for a corresponding x), when multiplied by the value of a gives the corresponding y . For example, there is an intersection value of $x=6$ and $y=17$ with the value of a as 2. This can be construed as the following: 12 (corresponding to $x = 6$) when multiplied by $a=2$, gives 24 (corresponding to $y = 17$). Thus a is nothing but a multiple for a term in the denominator which gives a value in the numerator.

Horizontal Picture: For a particular horizontal row, we notice that all the dots (values of a) are different multiples of the right most number in that row. If we neglect the right most number, we realize that when the n 'th term from the left (LTR³) is multiplied by the n 'th term from the right (RTL⁴) gives the right most number. If a number is in the center then it should be multiplied by itself to get the right most number. *e.g.* We can see that for $y = 17$, we have terms 2, 3, 4, 6, 8, 12 and the right most 24. We realise that $2 * 12 = 24$, $3 * 8 = 24$ and $4 * 6 = 24$. Similarly for the value of $y = 9$, we find that the numbers are 2, 4, 8 and the right most 16. Clearly $2 * 8 = 16$ and $4 * 4 = 16$.

Vertical Picture: For a particular vertical row, we notice that it corresponds to a denominator. The dots on the vertical line show, how many and which, numerator values will correspond to the particular value of denominator (x for the vertical line). *e.g.*, a value of $x = 12$ will have three numerators which can contain potential multiples of the denominator corresponding to the value of x . Here for $x = 12$, the value of y are 5, 11, 17. If we do the math we will find that this means that the denominator is $(25 - 7 - 12 = 6)$, will have multiples in $7 + 5 = 12$, $7 + 11 = 18$ and $7 + 17 = 24$.

Calculation of a combination: We realise that calculations of a combination can be shortened.

$$\begin{aligned}
{}^nC_k &= \frac{(n) \times (n-1) \cdots \times (k+1) \times (k) \times (k-1) \cdots \times 3 \times 2 \times 1}{(n-k) \times (n-k-1) \cdots \times 3 \times 2 \times 1 \times (k) \times (k-1) \cdots \times 3 \times 2 \times 1} \\
&= \frac{(n) \times (n-1) \cdots \times (k+1)}{(n-k) \times (n-k-1) \cdots \times 3 \times 2 \times 1} \\
&= \frac{(n) \times (n-1) \cdots \times (n-k+1) \times (n-k) \times (n-k-1) \cdots (k+1)}{(n-k) \times (n-k-1) \cdots \times (k) \times (k-1) \cdots (3) \times (2) \times (1)} \\
&= \frac{(n) \times (n-1) \cdots \times (n-k+1)}{(k) \times (k-1) \cdots (3) \times (2) \times (1)}
\end{aligned}$$

Using this information the calculated value of the combination is as below:

$$\begin{aligned}
{}^{25}C_7 &= \frac{25 \times 24 \times 23 \times 22 \times 21 \times 20 \times 19}{7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1} \\
&= 480700
\end{aligned}$$

Locating the points on the graph for calculating the value of a combination: In the light of the above discussion (4), we realise the following.

1. y axis can be divided in two parts: the top part corresponds to $(n) \cdots \times (n-k+1)$ while the lower part corresponds to $(n-k) \times (n-k-1) \cdots (k+1)$. Clearly as in condensed equation 4 the lower part can be discarded and is denoted by red shaded box.
2. Similarly x axis can be divide in two parts: the left part corresponds to $(n-k) \times (n-k-1) \cdots \times (k+1)$ and the right part corresponds to $(k) \times (k-1) \cdots (3) \times (2) \times (1)$. Here, again the left part can be discarded (highlighted by red shaded box).
3. This leaves us with green and yellow shaded area for denominator and numerator respectively 14.
4. The steps of calculation, to get a value for combination, remain as defined before.

5 Delving deeper into the graphs

Based on the images seen before it is clear that the values of a for which an x has a corresponding y have a set of striking patterns if we analyze all k combinations of nC_k for a given n .

1. For a given n , graph is most dense for nC_1 and least dense for ${}^nC_{n-1}$. In other words, maximum number of a (and thus x and y) points are present on the graph in the former case and least points are present on the graph in the later case.
2. Vertical lines can be drawn for each x (which goes into the denominator term $(n-k-x)$).

³LTR is a direction left to right

⁴RTL is a direction and denotes right to left

3. For nC_1 , graph, the position of dots on the graph (a dot is a valid solution connecting x and y through a value of a), for any vertical line can be easily predicted. For the right most vertical line (equal to the value of $(n-1-1)$) for x , the dots start at the value of 2 for the lowest value of $y=1$ and then keep incrementing by 1 for every 1 position towards the top for a value of $(n-1)$ corresponding to the $y=(n-1)$. For the next left line (corresponding to the $(n-2-1)$), the dots start at the value of 1 for the lowest value of $y=2$ and then keep incrementing at every 2 position by 1. For the preceding subsequent line at $(n-3-1)$, the dots start at the value of 1 for the lowest value of $y=3$ and then keep incrementing at every 3 position. This pattern holds for every preceding line. For a line at position $(n-k-1)$, the points start at the value of 1 for the lowest value of $y=k$ and then keep incrementing at every k location by 1.
4. Horizontal lines can be drawn for each y (which goes into the numerator term $(k+y)$).
5. For nC_1 , graph, the position of dots on the graph, for any horizontal line can be easily predicted. For a value of y we consider $y+1$ and we find that the value of a for the dots are nothing but the factors of $y+1$. Clearly prime numbers have no factors other than themselves and 1.
6. As we go towards higher k (starting from 1 and go towards $(n-1)$) for the nC_k , the nC_1 graph shifts diagonally (left and downwards) and terms outside the boundaries of $1 \leq y \leq (n-k)$ and $0 \leq x \leq (n-k-1)$ keep ‘falling off’ the graph and ‘disappear’. This we can call as the ‘destruction picture’ or ‘low-to-high picture’. Please see the movie (25_C.k_264.mp4) by visiting and downloading/cloning URL <https://github.com/amit7urmc/CombinationsAndPrimes>.
7. As we go from higher to lower k (starting from $(n-1)$ and towards 1) for the nC_k , the nC_1 graph shifts diagonally (right and upwards) and terms within the boundaries of $1 \leq y \leq (n-k)$ and $0 \leq x \leq (n-k-1)$ keep ‘joining’ the graph and ‘appear’. This we can call as the ‘construction picture’ or ‘high-to-low picture’. Please see the movie (25_C.k_reverse_264.mp4) by visiting and downloading/cloning URL <https://github.com/amit7urmc/CombinationsAndPrimes>.
8. In either of the pictures (destruction or construction), we can pause at any nC_k and we can calculate the value of combination as in 4.

6 Conclusion

This article discusses about nC_k , where if n is even it can be written as $2 * n/2$ (where $n/2$ is the integer part of division of n by 2) while it can be stated as $2 * n/2 + 1$ if it is odd. Furthermore k , can take only three values: $(n/2)$, $(n/2 - l)$ or $(n/2 + m)$, where l and m are positive integers greater than 1, and they denote *less* and *more* respectively.

The denominator is composed of product of $(n-k-x)$ terms, while the numerator is composed of products of $(k+y)$ terms, with the following constraints.

$$\begin{aligned} 0 &\leq x \leq (n-k-1) \\ 1 &\leq y \leq (n-k) \end{aligned}$$

With these premises we obtain the following relationship table between y and x (given n and k). Please note that l and m need not be supplied and can always be calculated if n and k are given. However l and m must be known/calculated before using any of the below equations. Also, in the below table, the equation must be solved for all possible values of a using the following algorithm.

1. Decide n is even or odd.
2. given k and n , decide if l or m is there and if yes what value.
3. Get range of x such that $0 \leq x \leq (n-k-1)$
4. Solve for all values of a (for each x) using the below table such that $1 \leq y \leq (n-k)$.
5. For each a get a feasible and corresponding y .

n	k	y
Even	$n/2 \pm 0$ (i.e. $k = n/2$)	$y = (a-1) \times n/2 - a \times x$
Even	$n/2 - l$ (i.e. $k < n/2$)	$y = (a-1) \times n/2 + l \times (a+1) - a \times x$
Even	$n/2 + m$ (i.e. $k > n/2$)	$y = (a-1) \times n/2 - m \times (a+1) - a \times x$
Odd	$n/2 \pm 0$ (i.e. $k = n/2$)	$y = (a-1) \times n/2 + a - a \times x$
Odd	$n/2 - l$ (i.e. $k < n/2$)	$y = (a-1) \times n/2 + a + l \times (a+1) - a \times x$
Odd	$n/2 + m$ (i.e. $k > n/2$)	$y = (a-1) \times n/2 + a - m \times (a+1) - a \times x$

There can be an equivalent table where all the a terms are brought together. The below table can help find a quicker since all other terms are known. We can keep increasing a by 1 so that y stays in the range as per premise.

n	k	y
Even	$n//2$ (i.e. $k = n//2$)	$y = a \times (n//2 - x) - n//2$
Even	$n//2 - l$ (i.e. $k < n//2$)	$y = a \times (n//2 - x + l) - n//2 + l$
Even	$n//2 + m$ (i.e. $k > n//2$)	$y = a \times (n//2 - x - m) - n//2 - m$
Odd	$n//2$ (i.e. $k = n//2$)	$y = a \times (n//2 - x + 1) - n//2$
Odd	$n//2 - l$ (i.e. $k < n//2$)	$y = a \times (n//2 - x + l + 1) - n//2 + l$
Odd	$n//2 + m$ (i.e. $k > n//2$)	$y = a \times (n//2 - x - m + 1) - n//2 - m$

7 Other implications of the current work

In the light of analyzing the previous graphs, some implications naturally follow. They are about factors and prime numbers contained within the range 1 to n and more importantly their generation process. We find, as below, that the process is not random rather it is simple and systematic.

- In the light of these images and in the hindsight, it is easy to see now that prime numbers do not appear, on the number-line, at random afterall! There is a definite and simple rule governing their appearance on the number line. To simplify the discussion and summary consider this analogy:
 - In the beginning there is only one known prime in the list of primes i.e. [2] on the number line.
 - Now each number (in the list of primes so far) hops one step on the integer number line. If for any prime number, the number of steps modulus the prime number itself is zero (i.e. if it is an exact multiple of that prime or logically $[(\text{number of steps}) \bmod (\text{the particular prime number})] == 0$), then we discard that number from the consideration of being a prime. However if for **all** numbers (in the list of primes till then) there is some remainder on modulus, then that means this **new step or the new number** is not a multiple of any known prime. Thus we have generated a prime number.
 - If in the above step we generate a prime then we add it to the list of known primes and the hopping continues until as long as desired. Before the subsequent hops, however, this newly generated prime number ‘joins’ the hopping game by being included in the list of prime numbers.
 - Thus it seems that in this ‘hopping game’ along the integer line, any new numbers is ‘spared’ if it is an exact multiple of any known prime number till that step. However if all known prime numbers ‘desert’ that number then this ‘new orphan’ number ‘sprouts and comes into existence’ and immediately joins the hopping game. There seems to be no end to this and this can continue indefinitely.
- To prove this computationally, we downloaded first known 1000 primes from <https://t5k.org/lists/small/1000.txt> and compared it against the primes generated by the algorithm mentioned above (see appendix B for the *python3* code). There is a perfect match!

References

- [1] Wikipedia. History of combinatorics, 2024. [Online; accessed 26-July-2024].
- [2] Wikipedia. Blaise pascal, 2024. [Online; accessed 26-July-2024].
- [3] Wikipedia. Pierre de fermat, 2024. [Online; accessed 26-July-2024].
- [4] Wikipedia. Leonhard euler, 2024. [Online; accessed 26-July-2024].
- [5] Wikipedia. Pascal’s triangle, 2024. [Online; accessed 01-August-2024].
- [6] Chuang-Chong Chen and Khee-Meng Koh. Principles and Techniques in Combinatorics. World Scientific, 1992.
- [7] Scott Hartshorn. Probability - A Beginner’s guide To Permutations and Combinations. Amazon Kindle, 2016.

Software used in this research work.

1. L^AT_EX was used for article writing.
2. Python 3.10 was used as the programming languages
3. Figures were generated with the help of Matplotlib (a Python library) and LibreOffice Impress.

Appendices

A Code to generate the plots as in 6

```
#!/usr/bin/python
"""
This module visualizes the graphs for a particular combination nCk.
create_chart_combination functions creates all x and y so that it can be plotted.
There are two functions:
(1) create_all_cases_graph: for all k(s) for a given n
(2) create_one_case_graph: for a specific k for a given n
There are 6 helper function depending n is even and if k has 1 or m.
All the files are saved to the current directory.
"""
from math import ceil, sqrt
from typing import Union
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator

## Constants. Please do not change
GRAPH_TITLE = '$ ^{d}C_{d}$'
TEXTCOORDS = 'offset pixels'
#####

def helper_even_not_1_m(each_x:int,
                        n:int,
                        k:int,
                        a:list[int],
                        y: list[int],
                        indiv_solution_params:list[tuple[Union[int,bool], Union[int,bool],
                                                         Union[int,bool]]],

                        starting_guess:int=1,
                        l:bool=False,
                        not_1_m:bool=True,
                        m:bool=False)->None:

    """
    This function prepares y and individual params if n is even and both l amd m are zero for k
    """
    while True:
        y_ = (starting_guess-1)*n//2 - (starting_guess*each_x)
        if y_ > (2*(n//2)-k):
            break
        if y_ >= 1:
            a.append(starting_guess)
            indiv_solution_params.append((l,not_1_m,m,starting_guess))
            starting_guess += 1
        y_s = list(map(lambda a_dash: (a_dash-1)*n//2 - (a_dash*each_x),a))
        y.append(y_s)

def helper_even_1(each_x:int,
                  n:int,
                  k:int,
                  a:list[int],
                  y: list[int],
                  indiv_solution_params:list[tuple[Union[int,bool], Union[int,bool],
                                                   Union[int,bool]]],

                  starting_guess:int=1,
                  l:bool=True,
                  not_1_m:bool=False,
                  m:bool=False)->None:

    """
    This function prepares y and individual params if n is even and m is zero for k but not 1.
    """
    while True:
```

```

y_ = (starting_guess-1)*(n//2) + 1*(starting_guess+1) - (starting_guess*each_x)
if y_ > (2*(n//2)-k):
    break
if y_ >= 1:
    a.append(starting_guess)
    indiv_solution_params.append((l,not_l_m,m,starting_guess))
    starting_guess += 1
y_s = list(map(lambda a_dash: (a_dash-1)*(n//2) + 1*(a_dash+1) - (a_dash*each_x),a))
y.append(y_s)

def helper_even_m(each_x:int,
                  n:int,
                  k:int,
                  a:list[int],
                  y: list[int],
                  indiv_solution_params:list[tuple[Union[int,bool], Union[int,bool],
                                                    Union[int,bool]]],

                  starting_guess:int=1,
                  l:bool=False,
                  not_l_m:bool=False,
                  m:bool=True)->None:
    """
    This function prepares y and individual params if n is even and l is zero for k but not m.
    """
    while True:
        y_ = (starting_guess-1)*(n//2) - m*(starting_guess+1) - (starting_guess*each_x)
        if y_ > (2*(n//2)-k):
            break
        if y_ >= 1:
            a.append(starting_guess)
            indiv_solution_params.append((l,not_l_m,m,starting_guess))
            starting_guess += 1
        y_s = list(map(lambda a_dash: (a_dash-1)*(n//2) - m*(a_dash+1) - (a_dash*each_x),a))
        y.append(y_s)

def helper_noteven_not_l_m(each_x:int,
                           n:int,
                           k:int,
                           a:list[int],
                           y: list[int],
                           indiv_solution_params:list[tuple[Union[int,bool], Union[int,bool],
                                                             Union[int,bool]]],

                           starting_guess:int=1,
                           l:bool=False,
                           not_l_m:bool=True,
                           m:bool=False)->None:
    """
    This function prepares y and individual params if n is odd and both l and m are zero for k.
    """
    while True:
        y_ = (starting_guess-1)*(n//2) + starting_guess - (starting_guess*each_x)
        if y_ > ((2*(n//2)+1)-k):
            break
        if y_ >= 1:
            a.append(starting_guess)
            indiv_solution_params.append((l,not_l_m,m,starting_guess))
            starting_guess += 1
        y_s = list(map(lambda a_dash: (a_dash-1)*(n//2)+a_dash - (a_dash*each_x),a))
        y.append(y_s)

def helper_noteven_l(each_x:int,
                    n:int,
                    k:int,
                    a:list[int],
                    y: list[int],
                    indiv_solution_params:list[tuple[Union[int,bool], Union[int,bool],
                                                      Union[int,bool]]],

                    starting_guess:int=1,
                    l:bool=True,
                    not_l_m:bool=False,
                    m:bool=False)->None:
    """
    This function prepares y and individual params if n is odd and m is zero for k but not l.
    """
    while True:
        y_ = (starting_guess-1)*(n//2) + starting_guess + l*(starting_guess+1) - (starting_guess
        *each_x)

```

```

        if y_ > ((2*(n//2)+1)-k):
            break
        if y_ >= 1:
            a.append(starting_guess)
            indiv_solution_params.append((l, not_l_m, m, starting_guess))
            starting_guess += 1
    y_s = list(map(lambda a_dash: (a_dash-1)*(n//2)+a_dash +1*(a_dash+1) - (a_dash*each_x), a))
    y.append(y_s)

def helper_noteven_m(each_x:int,
                    n:int,
                    k:int,
                    a:list[int],
                    y: list[int],
                    indiv_solution_params:list[tuple[Union[int, bool], Union[int, bool],
                                                    Union[int, bool]]],

                    starting_guess:int=1,
                    l:bool=False,
                    not_l_m:bool=False,
                    m:bool=True)->None:
    """
    This function prepares y and individual params if n is odd and l is zero for k but not m.
    """
    while True:
        y_ = (starting_guess-1)*(n//2) +starting_guess -m*(starting_guess+1) - (starting_guess*
                                                                              each_x)

        if y_ > ((2*(n//2)+1)-k):
            break
        if y_ >= 1:
            a.append(starting_guess)
            indiv_solution_params.append((l, not_l_m, m, starting_guess))
            starting_guess += 1
    y_s = list(map(lambda a_dash: (a_dash-1)*(n//2)+a_dash -m*(a_dash+1) - (a_dash*each_x), a))
    y.append(y_s)

def create_chart_combination(n: int, k:int)->list[list[int], list[int], list[int]]:
    """
    Premise (See the accompanying PDF)
    -----
    | n      | k      | y
    -----
    | Even   | n//2   | y = (a-1)\times n//2 - a\times x
    | Even   | n//2-1 | y = (a-1)\times n//2 + 1 \times (a+1) - a\times x
    | Even   | n//2+m | y = (a-1)\times n//2 - m \times (a+1) - a\times x
    | Odd    | n//2   | y = (a-1)\times n//2 + a - a\times x
    | Odd    | n//2-1 | y = (a-1)\times n//2 + a + 1\times(a+1) - a\times x
    | Odd    | n//2+m | y = (a-1)\times n//2 + a -m\times(a+1) - a\times x
    -----
    The chart can be made in the following step.
    1. Decide n is even or odd.
    2. given k and n, decide if l or m is there and if yes what value.
    3. Get range of x such that 0 <= x <= (n-k-1)
    4. Solve for a(s) and y(s) for each x with restrain that 1 <= y <= (n-k).
    5. Return a list of three lists [x, y, a] # for plotting
    """
    is_even = n%2 == 0
    l, not_l_m, m = n//2-k if k < n//2 else None, \
                    n//2 if k==n//2 else None, \
                    k-n//2 if k > n//2 else None
    x = list(range(0, n-k, 1))
    y = []
    solution_params = []
    for each_x in x:
        a = []
        starting_guess = 1
        indiv_solution_params = []
        if is_even:
            if not_l_m:
                helper_even_not_l_m(each_x, n, k, a, y,
                                    indiv_solution_params, starting_guess, l, not_l_m, m)
            elif l:
                helper_even_l(each_x, n, k, a, y,
                              indiv_solution_params, starting_guess, l, not_l_m, m)
            elif m:
                helper_even_m(each_x, n, k, a, y,
                              indiv_solution_params, starting_guess, l, not_l_m, m)
        else:

```



```

        if not_l_m:
            helper_noteven_not_l_m(each_x, n, k, a, y,
                                   indiv_solution_params, starting_guess, l, not_l_m, m)
        elif l:
            helper_noteven_l(each_x, n, k, a, y,
                             indiv_solution_params, starting_guess, l, not_l_m, m)
        elif m:
            helper_noteven_m(each_x, n, k, a, y,
                             indiv_solution_params, starting_guess, l, not_l_m, m)
        solution_params.append(indiv_solution_params)
    returned_x = []
    returned_y = []
    returned_params = []
    for each_y_list_index, each_y_list in enumerate(y):
        solution_params_ = solution_params[each_y_list_index]
        for each_y_index, each_y in enumerate(each_y_list):
            returned_x.append(x[each_y_list_index])
            returned_y.append(each_y)
            returned_params.append(solution_params_[each_y_index])
    returned_results = [returned_x, returned_y, returned_params]
    return returned_results

def create_one_case_graph(n: int, k: int, figure_path: None=None)->None:
    """
    Unlike the create_all_case_graphs, this function takes a specific n and k and create a
    graph.
    """
    fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(12,12))
    ax.set_xlim(right=n+1)
    ax.set_xlim(left=-1)
    ax.set_ylim(top=n+1)
    ax.set_ylim(bottom=0)
    ax.set_xticks(ticks=list(range(1,n-k,1)), labels=[str(i) for i in range(1,n-k,1)])
    ax.yaxis.set_major_locator(MaxNLocator(integer=True, min_n_ticks=1))
    ax.xaxis.set_major_locator(MaxNLocator(integer=True, min_n_ticks=1))
    x, y, params = create_chart_combination(n,k)
    ax.scatter(x=x, y=y, c=[[0,0,0]], marker='o')
    ax.grid(which='both')
    ax.set_title(GRAPH_TITLE%(n,k))
    for data_index, _ in enumerate(params):
        param = _
        annotation = f"a={param[3]}"
        xy = (x[data_index], y[data_index])
        ax.annotate(annotation, xy=xy, xytext=(10,-20),textcoords=TEXTCOORDS)
    ax.set_xlabel('x as in denominator (n-k-x)')
    ax.set_ylabel('y as in numerator (k+y)')
    if figure_path:
        fig.savefig(f'{figure_path}/{n}_{k}_alone.png')
    else:
        fig.savefig(f'./{n}_{k}_alone.png')
    plt.close()

def create_all_cases_graph(n: list[int, int], figure_path:None=None)->None:
    """
    Supplied one even and one add number. 0th index should be even while 1st index should be
    odd.
    """
    assert n[0]%2==0 and n[1]%2
    markers = ["o", "v", "1", "s", "p", "P", "*", "H", "D", "X"]
    fig, axs1 = plt.subplots(nrows=int(ceil(sqrt(n[0]))), ncols=int(ceil(sqrt(n[0]))), sharey=
                             True, sharex=True)

    for axs_row in axs1:
        for axs_col in axs_row:
            axs_col.yaxis.set_major_locator(MaxNLocator(integer=True, min_n_ticks=1))
            axs_col.xaxis.set_major_locator(MaxNLocator(integer=True, min_n_ticks=1))
    ## Draw for even numbers
    counter = 0
    for k in range(1,n[0],1):
        if(k%int(ceil(sqrt(n[0])))==0):
            counter += 1
            x, y, params = create_chart_combination(n[0],k)
            axs1[counter][k%int(ceil(sqrt(n[0])))].scatter(x=x, y=y, c=[[0,0,0]], marker=markers[k%
                                                         (len(markers)-1)])
            axs1[counter][k%int(ceil(sqrt(n[0])))].grid(which='both')
            axs1[counter][k%int(ceil(sqrt(n[0])))].set_title(GRAPH_TITLE%(n[0],k))
            for data_index, _ in enumerate(params):
                param = _

```

```

        annotation = f"a={param[3]}"
        xy = (x[data_index], y[data_index])
        axs1[counter][k%int(ceil(sqrt(n[0])))].annotate(annotation, xy=xy, xytext=(10,-10),
            textcoords=TEXTCOORDS)

fig.suptitle("Graph for various nCk, where n is even")
fig.supxlabel("x as in denominator (n-k-x)")
fig.supylabel("y as in numerator (k+y)")
if figure_path:
    fig.savefig(f"{figure_path}/{n[0]}Ck.png")
else:
    fig.savefig(f"./{n[0]}Ck.png")

fig, axs2 = plt.subplots(nrows=int(ceil(sqrt(n[1]))), ncols=int(ceil(sqrt(n[1]))), sharey=
    True, sharex=True)

for axs_row in axs2:
    for axs_col in axs_row:
        axs_col.yaxis.set_major_locator(MaxNLocator(integer=True, min_n_ticks=1))
        axs_col.xaxis.set_major_locator(MaxNLocator(integer=True, min_n_ticks=1))

## Draw for odd numbers
counter = 0
for k in range(1,n[1],1):
    if(k%int(ceil(sqrt(n[1])))==0):
        counter += 1
    x, y, params = create_chart_combination(n[1],k)
    axs2[counter][k%int(ceil(sqrt(n[1])))].scatter(x=x, y=y, c=[[0,0,0]], marker=markers[k%
        (len(markers)-1)])
    axs2[counter][k%int(ceil(sqrt(n[1])))].grid(which='both')
    axs2[counter][k%int(ceil(sqrt(n[1])))].set_title(GRAPH_TITLE%(n[1],k))
    for data_index, _ in enumerate(params):
        param = _
        annotation = f"a={param[3]}"
        xy = (x[data_index], y[data_index])
        axs2[counter][k%int(ceil(sqrt(n[1])))].annotate(annotation, xy=xy, xytext=(10,-10),
            textcoords=TEXTCOORDS)

fig.suptitle("Graph for various nCk, where n is odd")
fig.supxlabel("x as in denominator (n-k-x)")
fig.supylabel("y as in numerator (k+y)")

if figure_path:
    fig.savefig(f"{figure_path}/{n[1]}Ck.png")
else:
    fig.savefig(f"./{n[1]}Ck.png")
plt.close()

if __name__ == "__main__":
    create_all_cases_graph([2,3])
    create_all_cases_graph([4,5])
    create_all_cases_graph([6,7])
    create_all_cases_graph([8,9])
    create_all_cases_graph([10,11])
    create_all_cases_graph([12,13])

N=25
for k_ in range(1,N,1):
    create_one_case_graph(N, k_)

```

B Code to generate the prime numbers (as discussed in section 5 and item number 2)

```

#!/usr/bin/python
import os
"""
This module generates prime numbers and the idea is inspired by the accompanying PDF.
"""
class GeneratePrime:
    """
    This class can be used to create prime numbers as per the algorithm in the accompanying PDF
    """

```

```

def __init__(self, file_to_write: str="./primeListGenerated.txt")-> None:
    """
    A list of prime numbers discovered and their current steps.
    To save computing the length in every iteration the list length is also maintained.
    file_to_write is a file where generated prime numbers are saved.
    """
    self.prime_numbers_discovered = [2]
    self.running_numbers = 2
    self.prime_numbers_discovered_list_length = 1
    self.file_to_write = file_to_write

def yield_prime(self, upto_n:int)->None:
    """
    upto_n denotes how many prime numbers need to be written to the file.
    """
    with open(self.file_to_write, "w", encoding='utf-8') as file_handle:
        file_handle.write("primeNumbersGenerated")
        file_handle.write("\n2")
        counter = 0
        while True:
            skip_flag = False
            self.running_numbers += 1
            for i in range(self.prime_numbers_discovered_list_length):
                if not self.running_numbers%self.prime_numbers_discovered[i]:
                    skip_flag = True
                    break

            if not skip_flag:
                self.prime_numbers_discovered.append(self.running_numbers)
                self.prime_numbers_discovered_list_length += 1
                file_handle.write(f"\n{self.running_numbers}")
                counter += 1
            if counter >= upto_n-1:
                break

def compare_against_known_result(self, know_result_file:str="./PrimeNumbersTop1000.txt")->
    bool:
    """
    A known result file is taken as input.
    Care must be taken that it has at least same number of primes
    as the upto_n parameter in the yield_prime method.
    """
    present_absent = None
    with open(know_result_file, "r", encoding='utf-8') as known_handle, \
        open(self.file_to_write, "r", encoding='utf-8') as calculated_handle:
        known_result = known_handle.readlines()[1:]
        calculated_result = calculated_handle.readlines()[1:]
        assert len(known_result) == len(calculated_result)
        present_absent = [known_result[i]==calculated_result[i] \
            for i in range(len(known_result))]
    if all(present_absent):
        return True
    return False

if __name__ == "__main__":
    UPTO_N = 1000
    FULL_MATCH_STR = "full_match"
    SOME_MISMATCH_STR = "some_mismatch"
    CURRENT_DIR = os.path.dirname(os.path.abspath(__file__))
    primeObj = GeneratePrime(f"{CURRENT_DIR}/primeListGenerated.txt")
    primeObj.yield_prime(UPTO_N)
    FULL_MATCH = primeObj.compare_against_known_result(f"{CURRENT_DIR}/PrimeNumbersTop1000.txt"
    )
    print(f"We tried to match first known {UPTO_N} primes with those generated by this program.
    \
    The answer is {FULL_MATCH_STR if FULL_MATCH else SOME_MISMATCH_STR}")

```

This code is also available in github <https://github.com/amit7urmc/CombinationsAndPrimes>.

C Figures

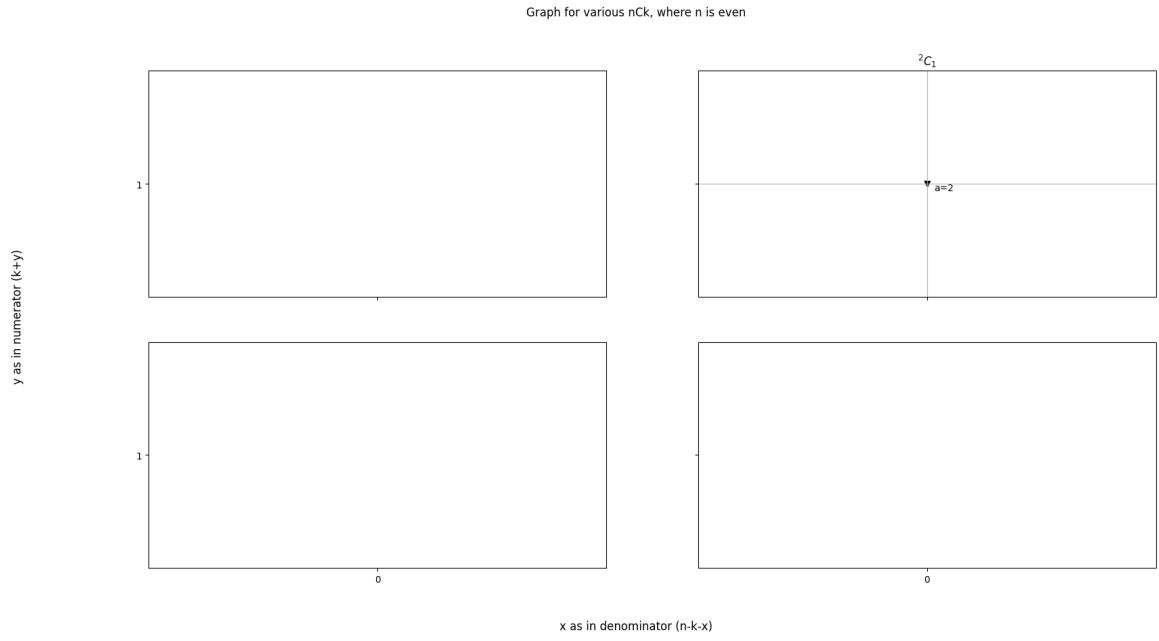


Figure 1: Plot of 2C_k

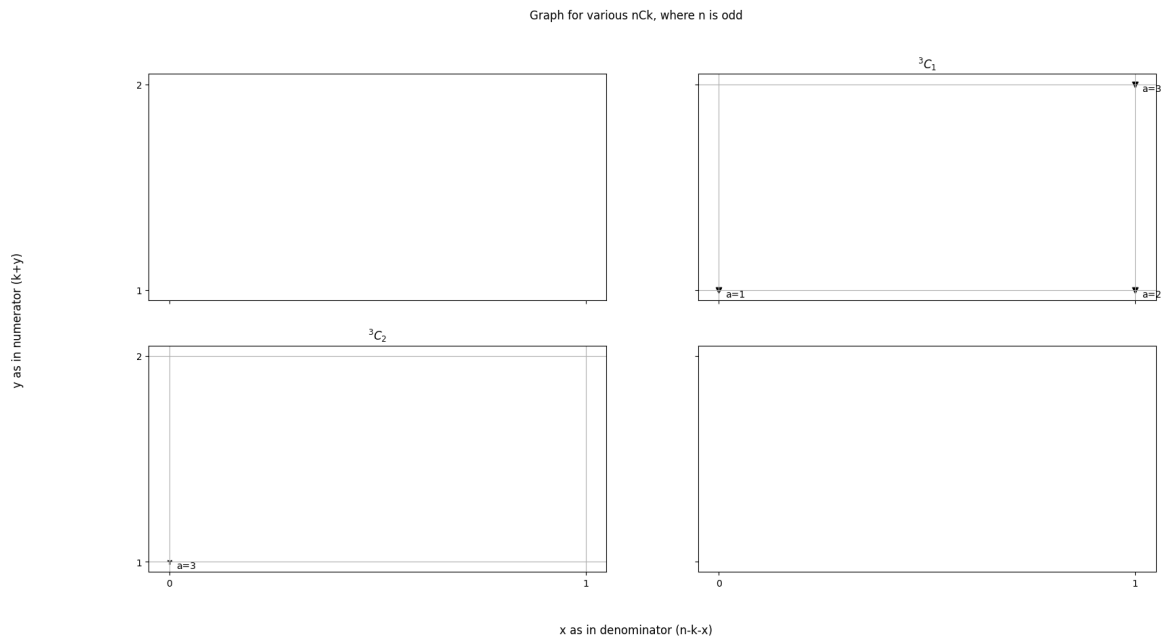


Figure 2: Plot of 3C_k

Graph for various nCk , where n is even

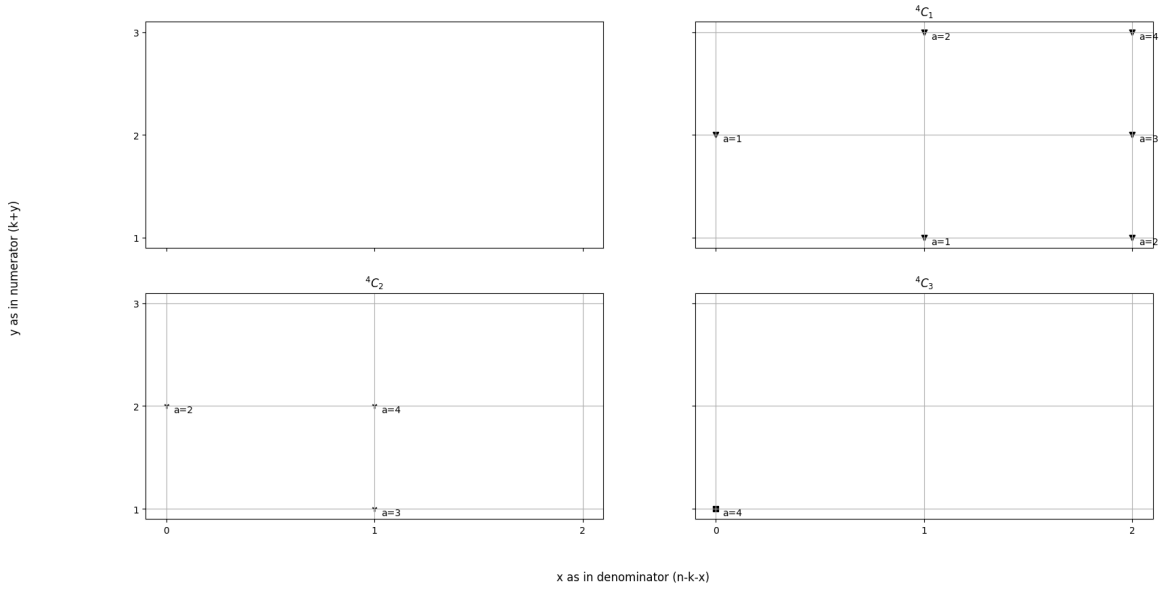


Figure 3: Plot of 4C_k

Graph for various nCk , where n is odd

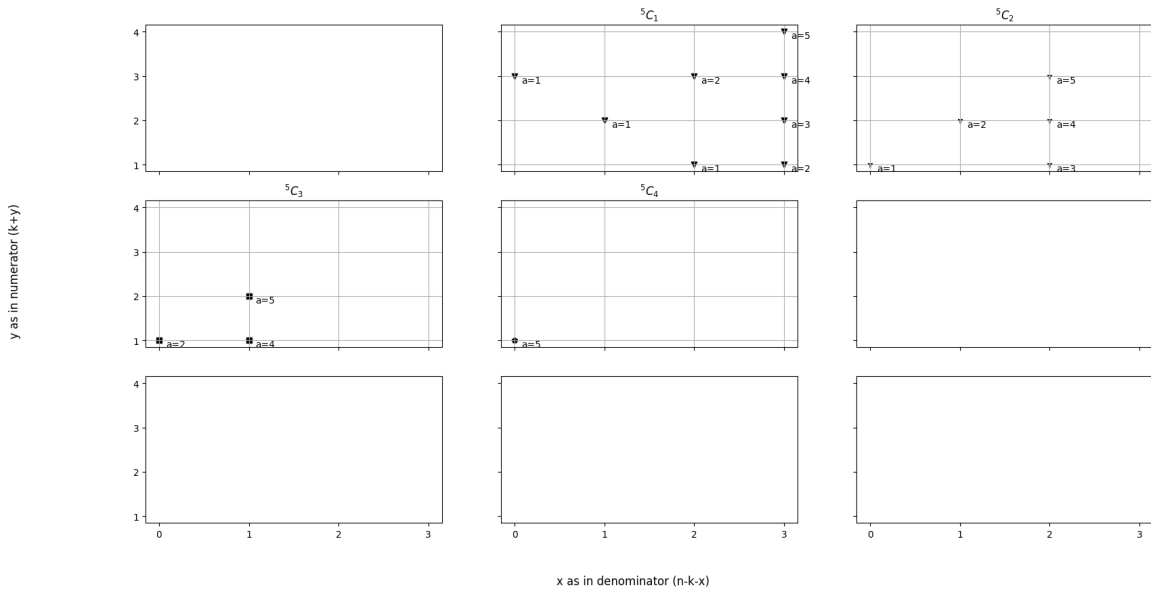


Figure 4: Plot of 5C_k

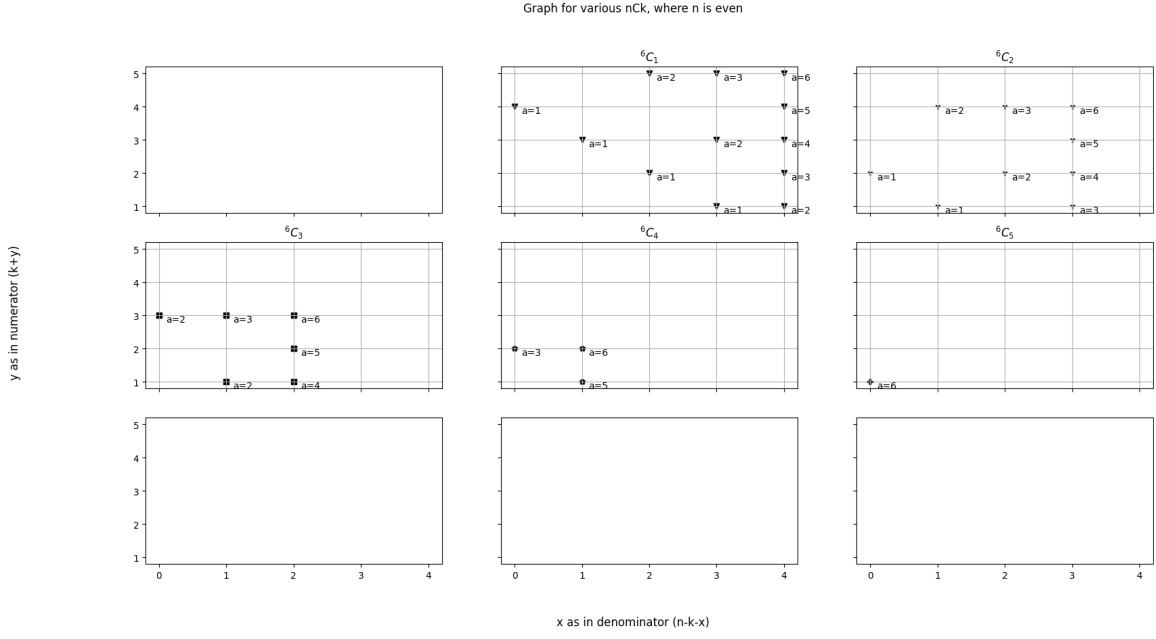


Figure 5: Plot of 6C_k

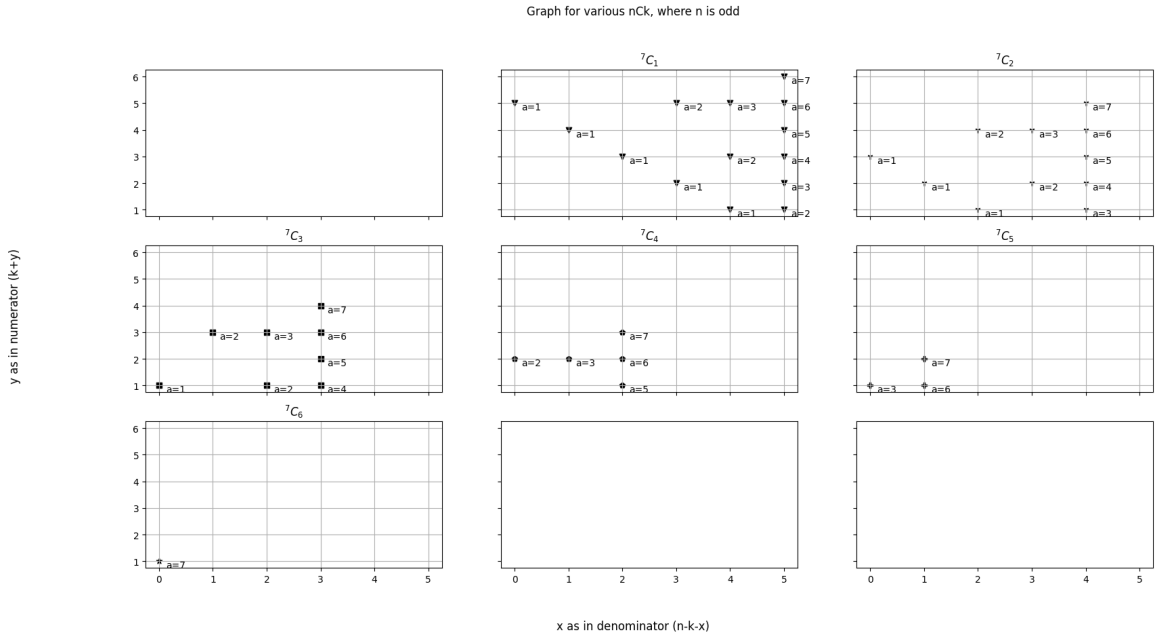


Figure 6: Plot of 7C_k

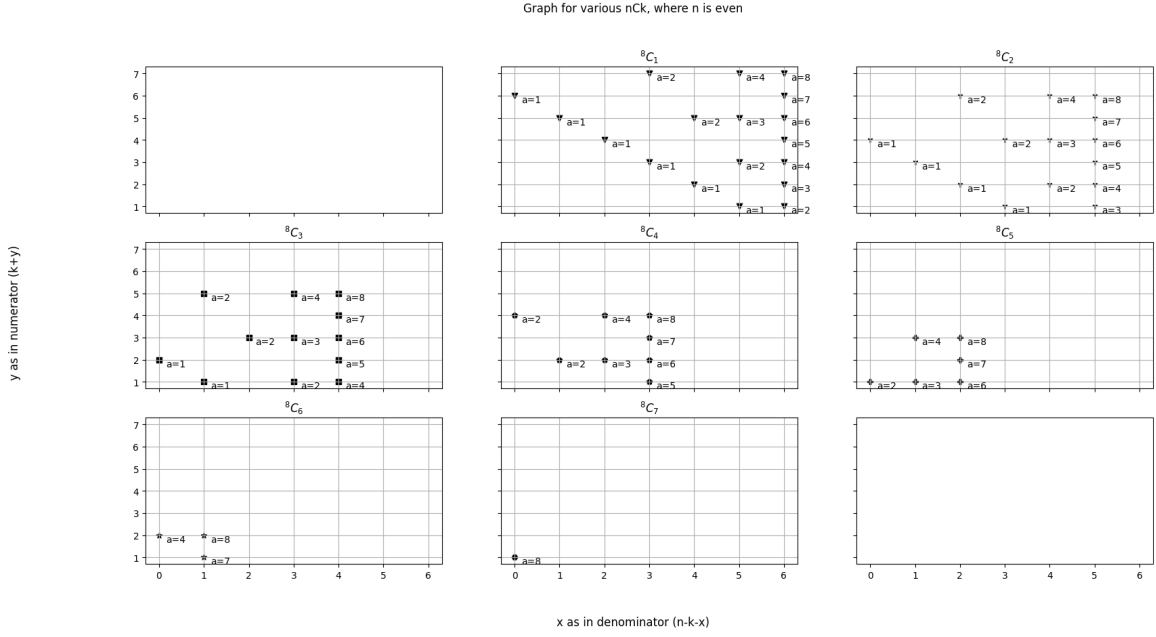


Figure 7: Plot of 8C_k

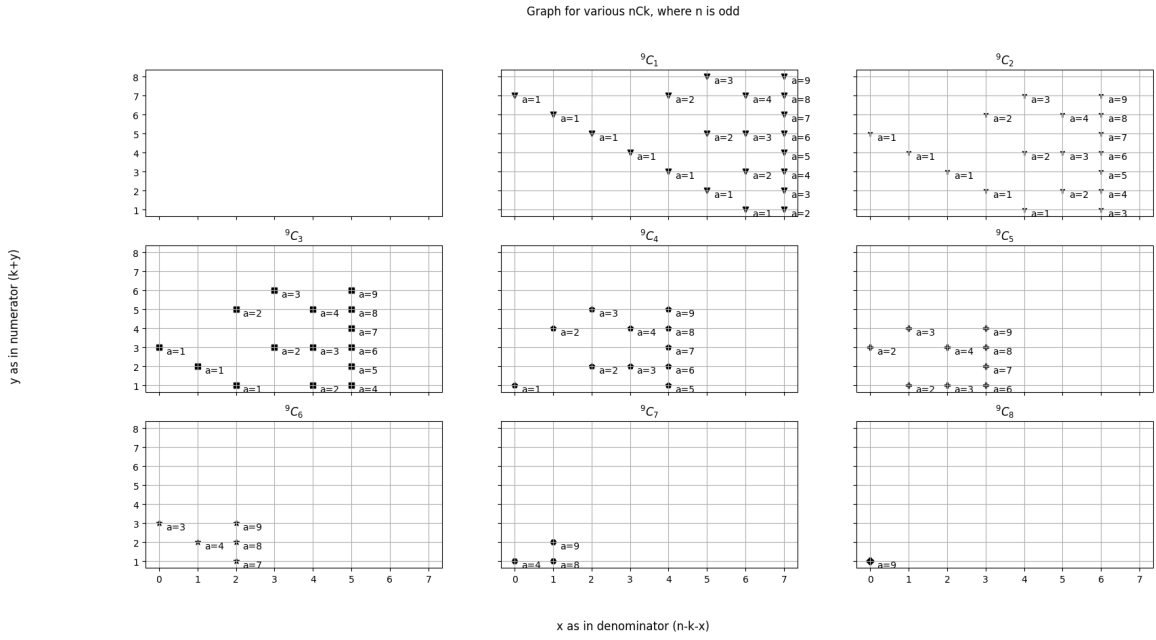


Figure 8: Plot of 9C_k

Graph for various nC_k , where n is even

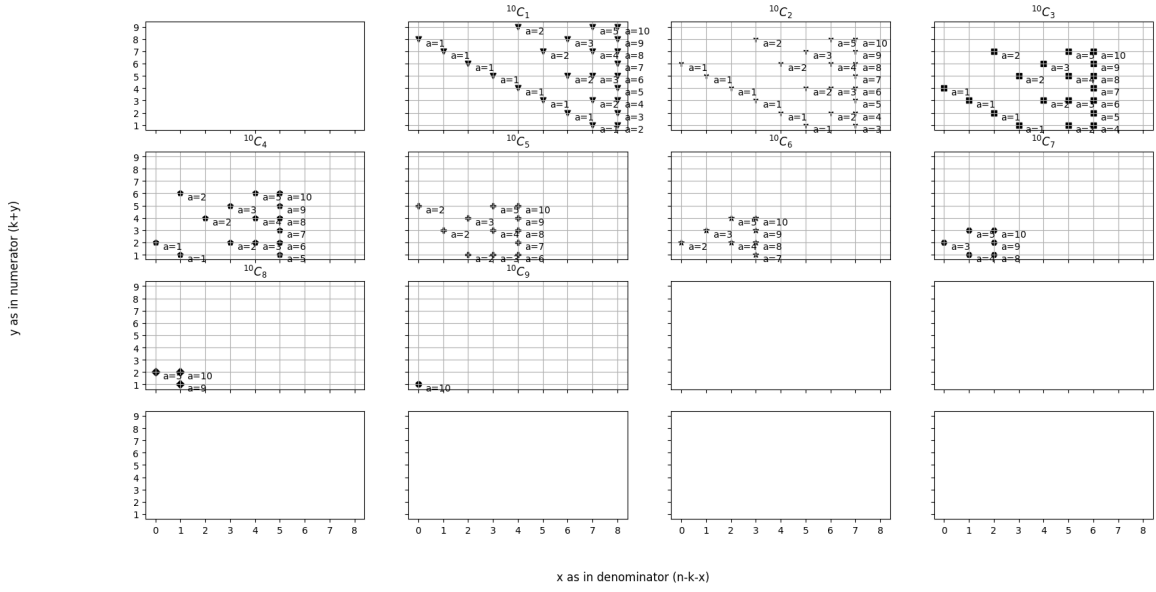


Figure 9: Plot of $^{10}C_k$

Graph for various nC_k , where n is odd

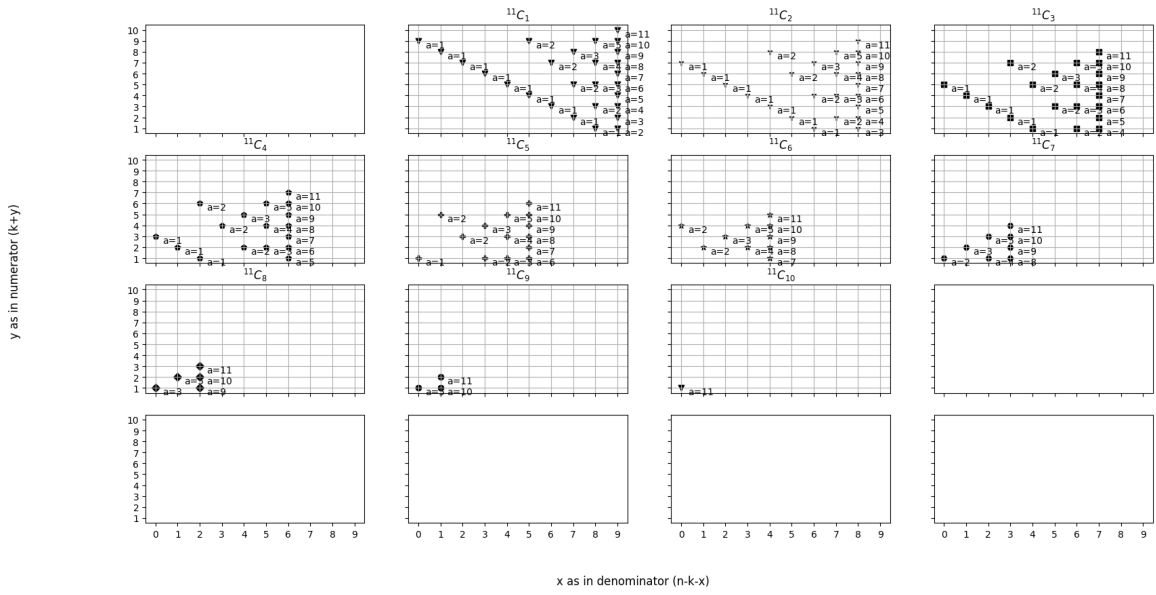


Figure 10: Plot of $^{11}C_k$

Graph for various nC_k , where n is even

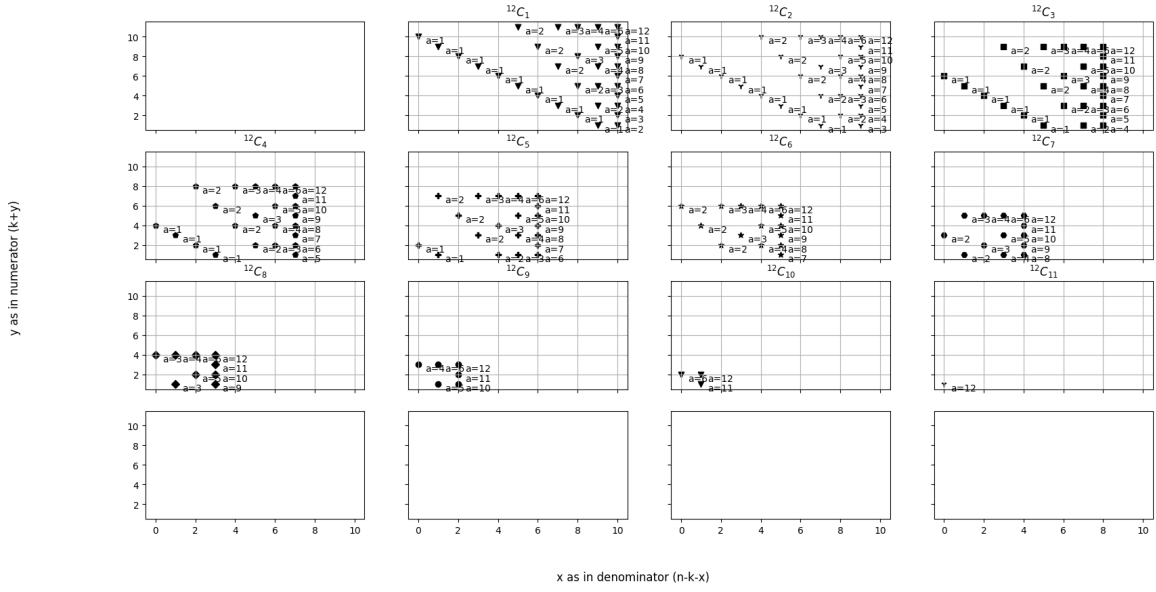


Figure 11: Plot of $^{12}C_k$

Graph for various nC_k , where n is odd

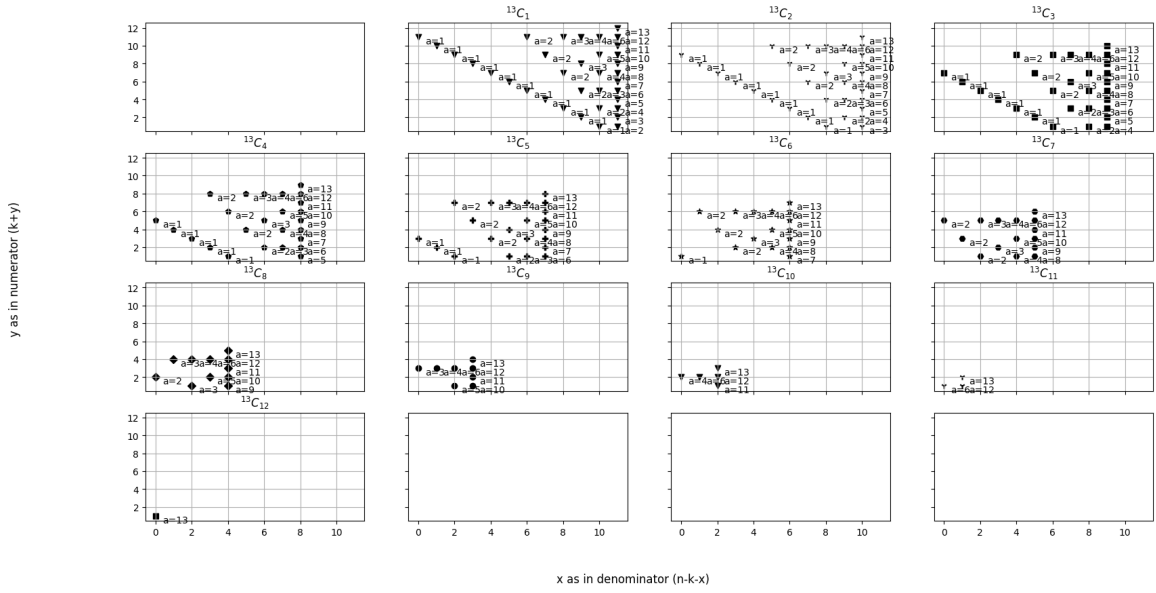


Figure 12: Plot of $^{13}C_k$

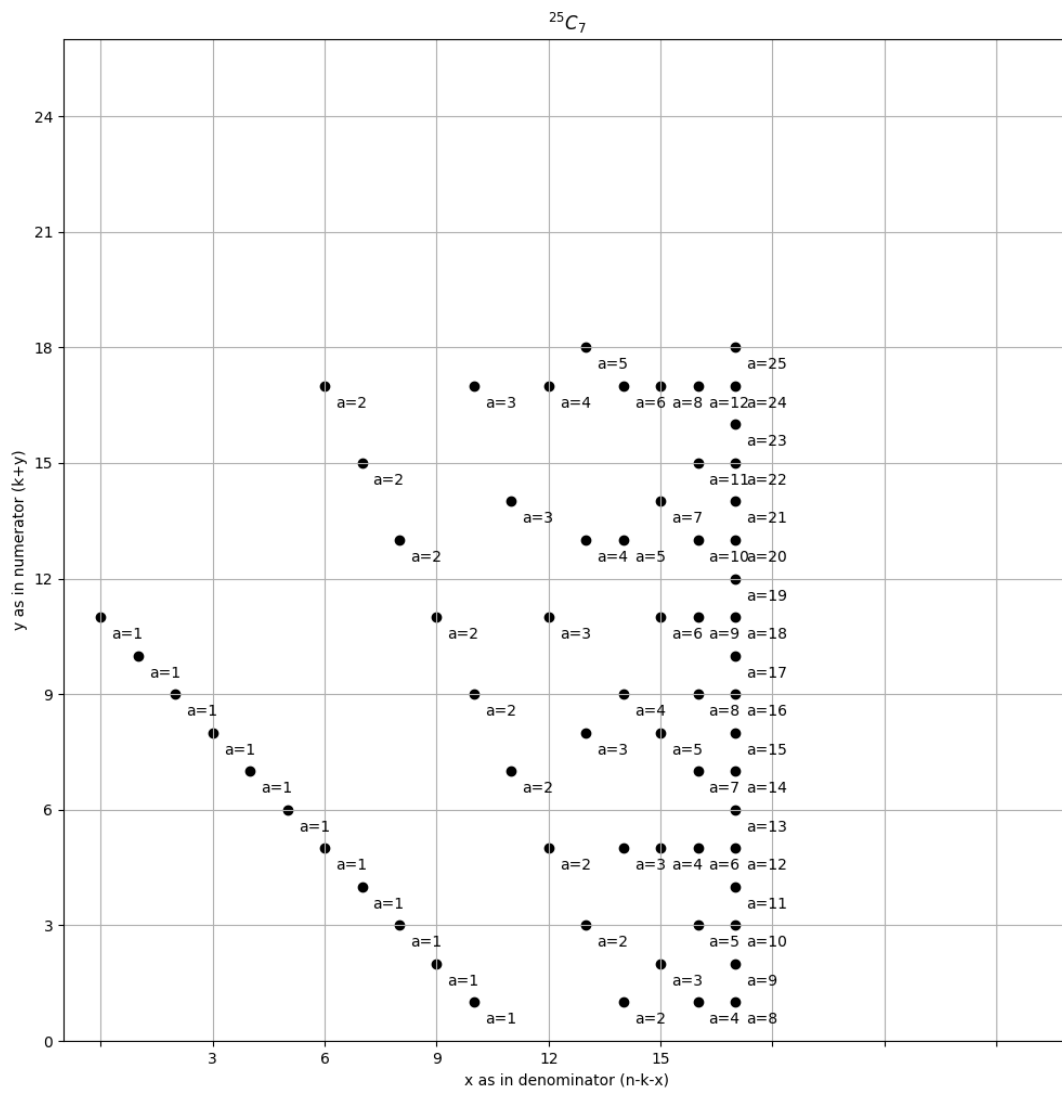


Figure 13: Distribution of x, y and a in the graph of ${}^{25}C_7$

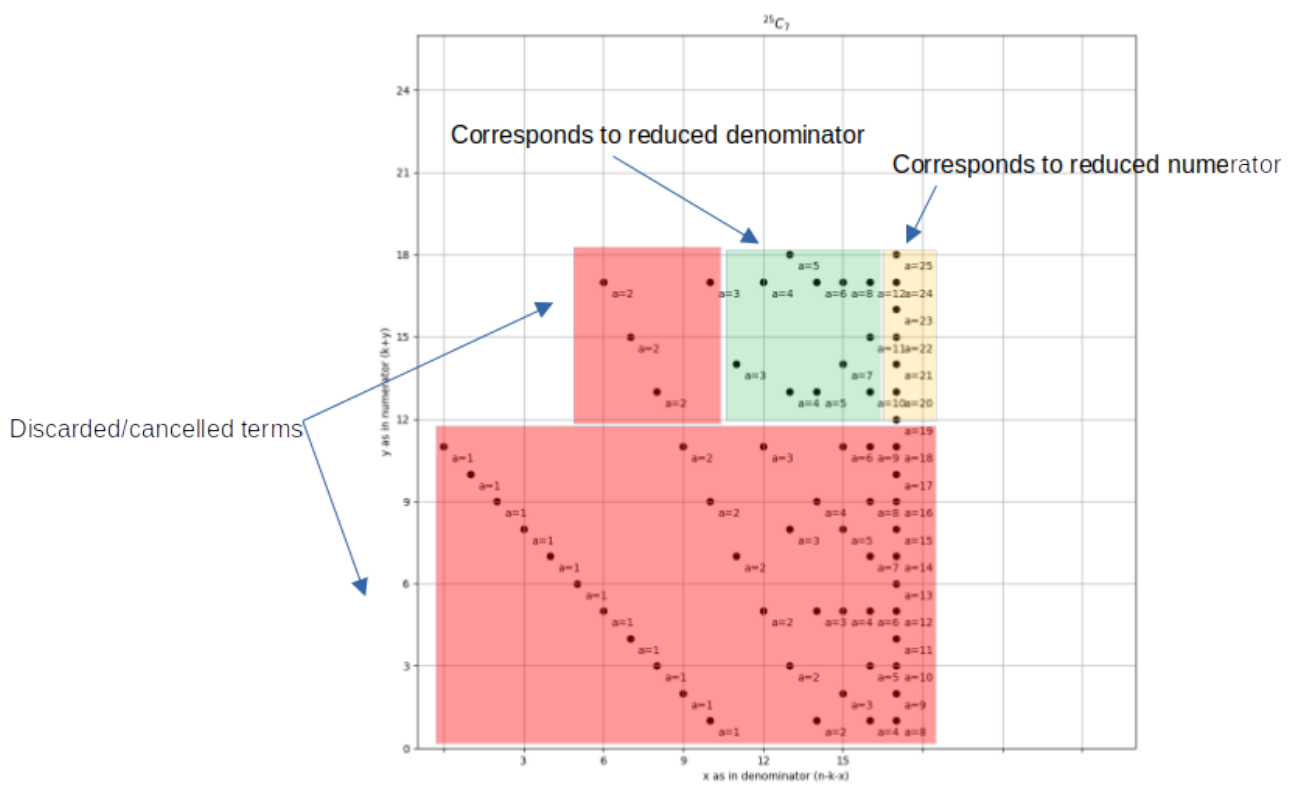


Figure 14: Showing set of numerator and denominator terms ${}^{25}C_7$

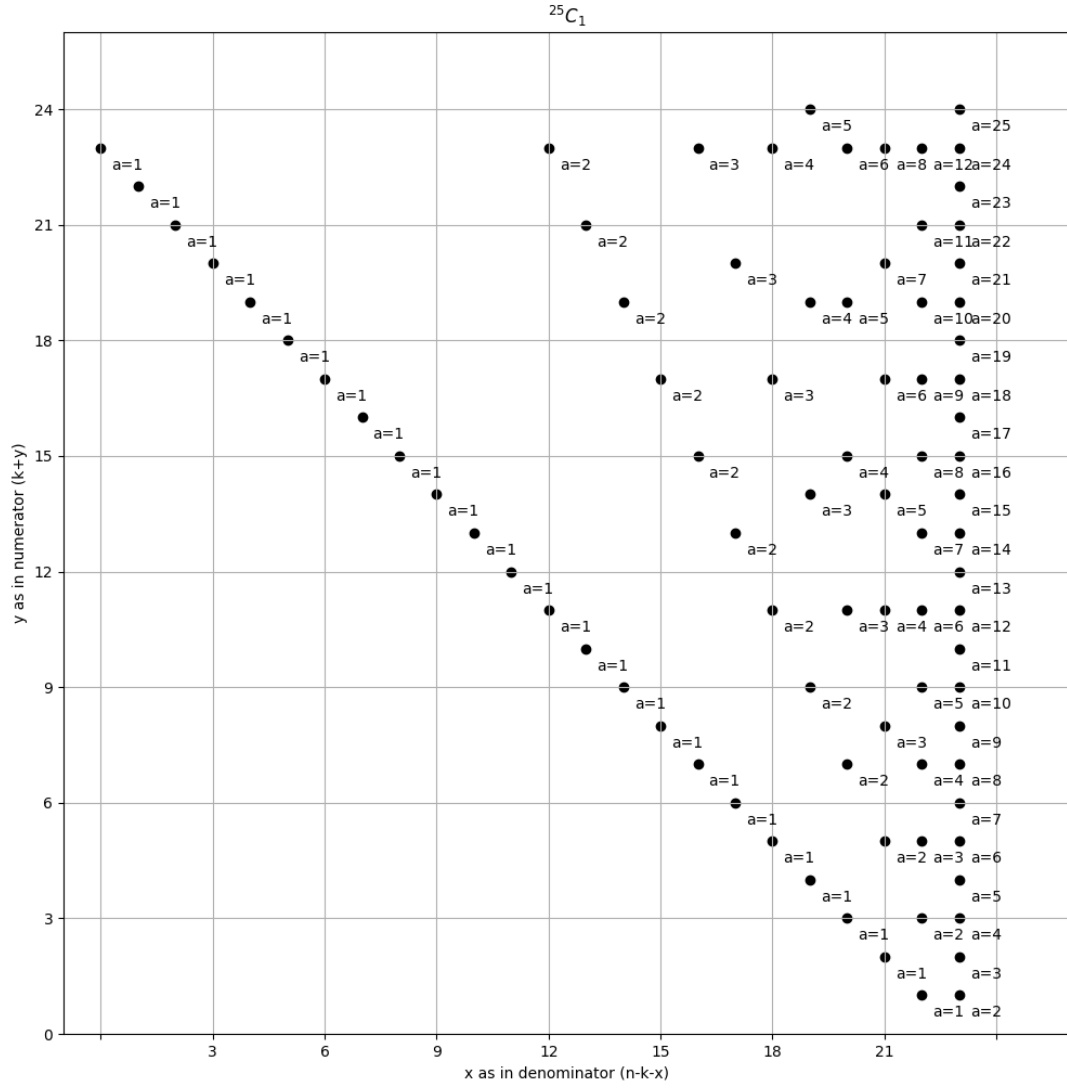


Figure 15: Distribution of x, y and a in the graph of ${}^{25}C_1$