THE BEAUTY OF PARTITION NUMBERS

Paawan Vala

January 2023

Contents

My attempt is to, First explain what the partition numbers are, Then by taking the recurssive approach trying to find the next term for the sequence ,and then term by term improving the method means generalizing it , and finally then explaining the correct method in which by the help of the terms we know how to find the next term of the sequence.

1 WHAT ARE PARTITION NUMBERS

By the name itself we can think that, it's something related to breaking and joining such as we can do to LEGO. If you think that than you are right because here we will do this process with numbers.

We can define partition numbers as follows : It's a way of writing an integer N as the sum of Positive integers.

1.1 AIM

We want to find the Total Number of Partitions for any Number N.

2 COUNTING THE PARTITIONS FOR A NUMBER

This can be generalized in two Methods.

METHOD 1:- In the first Method we will include the repeating Partition Ways.
For Example - 3+1 and 1+3 both will be counted.

METHOD 2:-In the Second method we will only include UNIQUE Partition ways.
Eg- 3+1 and 1+3 both will be treated as same so only one of them will be counted

3 METHOD 1

Let us say we want to write 0 in it's partition forms.
It can be written as 0 itself.
So we can say that 0 can be written in only 1 Way.

To write 1 in it's partition form , it can be written as 1 itself.
So 1 can also be written in only 1 way.

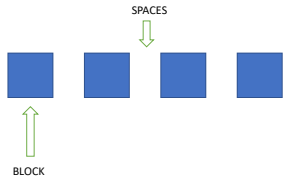
Like wise to write 4 in its partition form it can be written as follows:

4
3+1
1+3
2+2
2+1+1
1+1+2
1+2+1
1+1+1+1

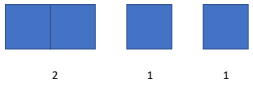
So there are total 8 partition ways of writing 4.

3.0.1 Generalizing The Problem

Let us assume our Numbers as Blocks. Where ONE block has the weightage of Number 1.
The Illustration is given below.



In the above Illustration we have 4 Blocks by arranging then we can form all the combination/partition of 4.
for Eg- 4 has total 8 Partitions, Given below are 2 partitions.



By observing this we can say that Total number of partitions for any number is dependent on the arrangement of SPACES.
So we can say that SPACES have two possible options (i) OPEN (ii) CLOSED , means they can either be OPEN or CLOSED.
And for a number N it has N Blocks so N-1 SPACES and every SPACES have 2 possibilities, So there are 2x2x...(N-1 times) possibilities.

Thus we can conclude that there are $2^{(N-1)}$ Possibilities of Partition for a number N.

4 METHOD :- 2

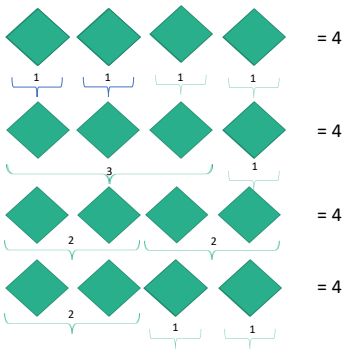
4.1 APPROACH 1

Let us say we want to write 0 in it's partition forms.
It can be written as 0 itself , So we can say that 0 can be written in only 1 Way.

To write 1 in it's partition form , it can be written as 1 itself , 1 can also be written in only 1 way.

To write 2 in it's Partition form. It can be written as 2 and 1+1. So 2 can be written in only 2 ways.

Similarly to write 4 in it's Partition ways, it is as follows.

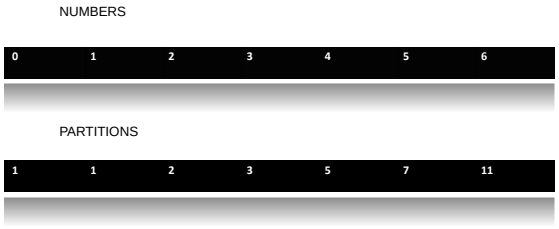


Also 4 can be written as 4 itself, So there are total 5 Ways of writing 4 in it's Partition form.

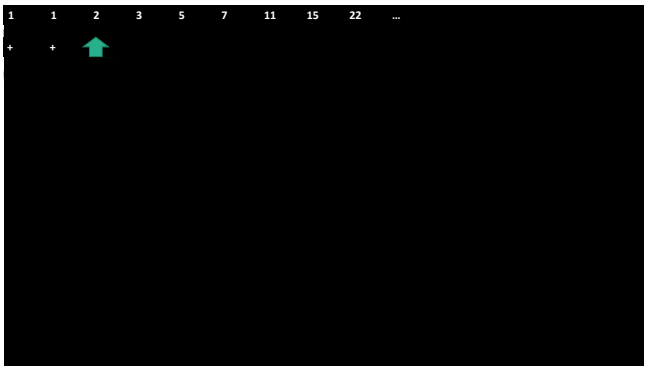
I hope by this way I am able to clarify the diffrence between Method 1 and Method 2.

4.1.1 Analyzing The Problem By Observation

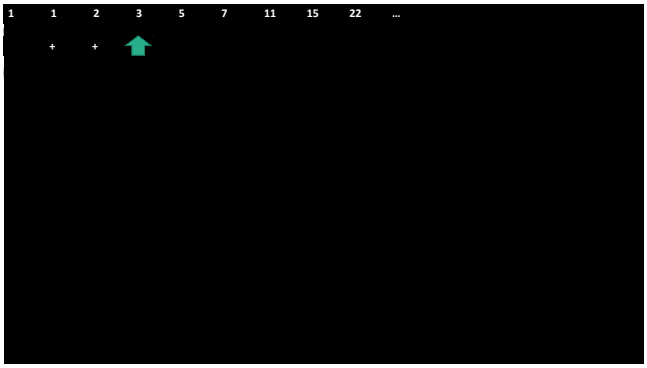
Now let us observe the series of Partition numbers and try to find some Patterns.



Let us see the series in a form of some Recurrence function.
Where our Base cases are, For number 0 we have 1 Partitons and For number 1 we have 1 Partitons.

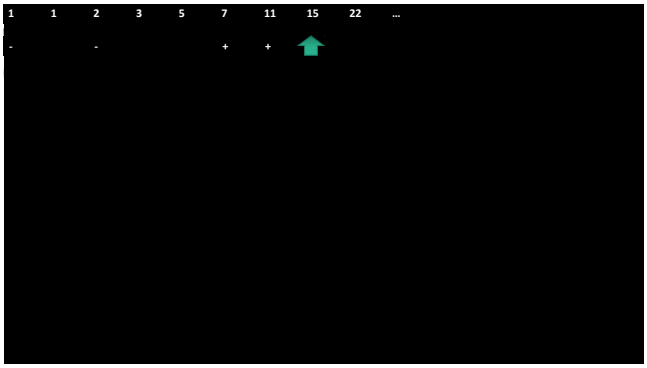


Now for number 2 the partitions are 2 which be represented as Partitions of number 0 + Partitions of number 1.



Similarly for number 3 the partitions are 3 which can be represented as Partitions of number 1 + Partitions of number 2.

But this pattern breaks when we try for number 5 whose Partitions are 7 which can be represented as Partitions of number 4 + Partitions of number 3 - Partitions of number 0.

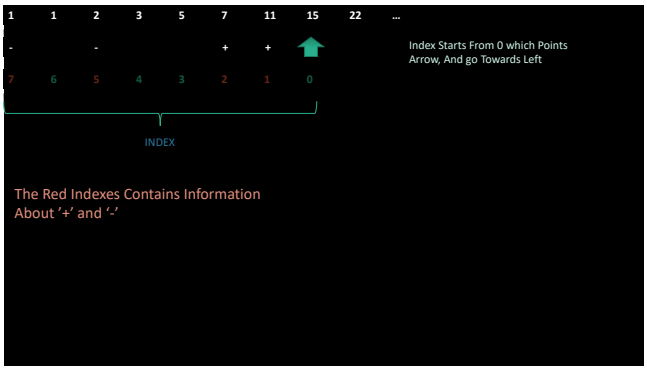


Similarly pattern breaks when we try for number 7 whose partitions are 15 which can be represented as Partitions of number 6 + Partitions of number 5 - Partitions of number 2 - Partitions of number 0.

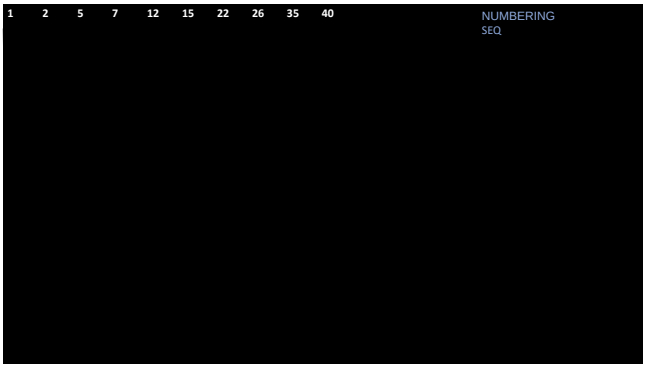
4.1.2 Approach For Solution

Ok now let us try to figure it out what is happening here, Let us say we are trying to find the Partitions for number 7.
As we have seen the partitions for number 7 we observe that first two terms are added and then another two terms are added by multiplying them by -1.
So it seems like this Pattern repeats such as {+,+,-,-,+,+,-,-,...,} as shown in figure for number 7.

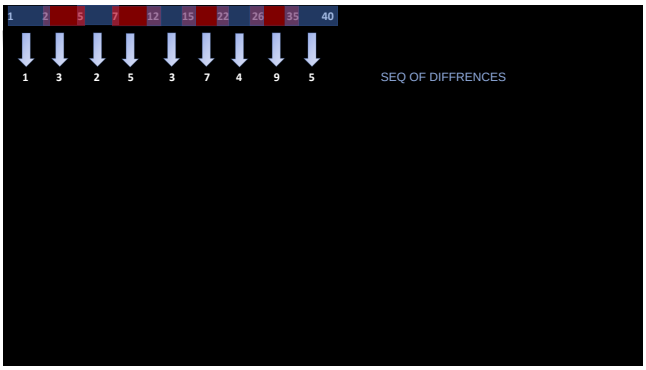
Now let us give Numbering (Index) to the place where this Addition and Subtraction will occur.
The Numbering is show in Below figure, Let us call this Sequence as "NUMBERING SEQUENCE"



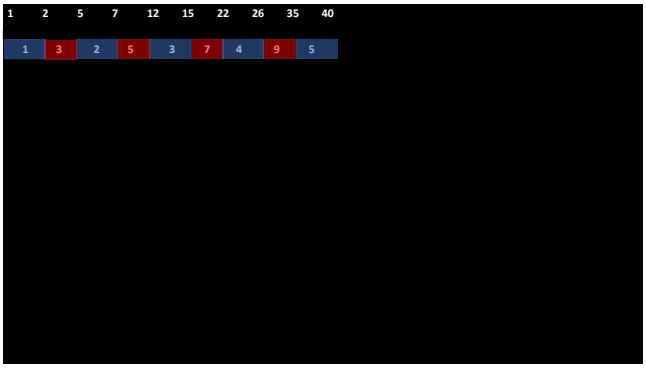
Let us try to Observe the "NUMBERING SEQUENCE".



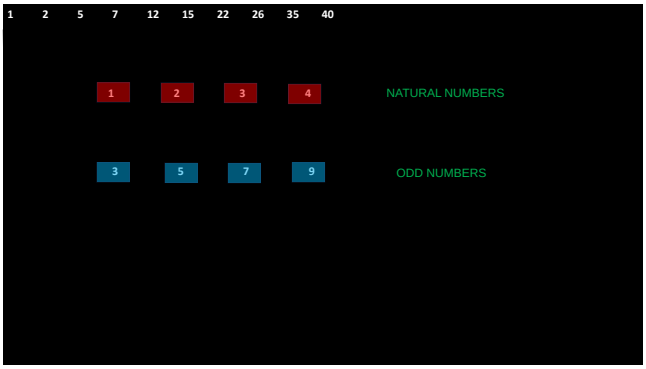
Let's find the Diffrence of Consecutive Terms in the Sequence. It seems as follows.



The Sequence formed Thus by calculating the diffrence of consecutive terms, Let us call it as "SEQ OF DIFFRENCES".
By Observing the SEQ OF DIFFRENCES can you try to find the N^{th} Term of "SEQ OF DIFFRENCES". Let me show you there's a hidden pattern Shown in below Figure.



In above figure, The Blue Highlighted seems a entire Sequence, and the red Highlighted seems a entire Sequence. By observing carefully we can tell that the Blue Highlighted Sequence contains ODD numbers and the red Highlighted Sequence contains NATURAL numbers.



Thus, in this SEQ OF DIFFRENCES x^{th} term can be easily find as follows.
(NOTE:- Indexing of SEQ OF DIFFRENCES starts from '0')
Let us have a function f: $\mathbb{W} \to \mathbb{N}$

$$f(x) = \begin{cases} \frac{x}{2} + 1, & \text{when x is EVEN} \\ x + 2, & \text{when x is ODD} \end{cases} \quad (1)$$

This Function $f(x)$ gives the Diffrence between two consecutive term of "NUMBERING SEQUENCE" sequence, That is the "SEQ OF DIFFRENCES" Sequence.
Thus by using the "SEQ OF DIFFRENCES" we can easily find x^{th} term of "NUMBERING SEQUENCE" as follows.

Now to Finally Generate x^{th} term of "NUMBERING SEQUENCE".

Let us have a function g: $\mathbb{N} \to \mathbb{N}$

Where $g(0)=1$ is defined.

And g(x) = g(x-1) + f(x-1) {x starts from 1}.
This Function $g(x)$ gives us the x^{th} Term of "NUMBERING SEQUENCE".
Thus in this way every term of "NUMBERING SEQUENCE" can be Generated.

If we know the "NUMBERING SEQUENCE" then you can find any term of "PARTITION SEQUENCE".
Because the "NUMBERING SEQUENCE" tells us which terms are to be ADDED and which terms need to be SUBTRACTED to find the N^{th} term of "PARTITION SEQUENCE" by using the knowing N-1 terms of "PARTITION SEQUENCE".

Thus in this way we can find any term of the PARTITION SEQUENCE.

4.1.3 ALGORITHM For Approach 1

For PARTITION SEQUENCE Our Base Case Is P[0] = 1 And P[1] = 1.
(Where P[k] Denotes K[th] Element of The Partition Sequence) We Will Start K from 2 And Apply The Algorithm And Increase K by 1 And Repeat The Algorithm Untill It reaches 'N' Where Our Aim Is To Find P[N].

1. To Find P[K] First We Will Find D[K-1] . D[K] Can Be Find With The Use Of Function F(x) as Follows.
Where D[K] Maps With f(x).
Where f(x) = (x/2)+1 , When x is even
And f(x) = x+2 , When x is odd
(D[k] Denoted The SEQ OF DIFFRENCES)

2. To Find N[k] We Will Use D[k-1] And N[K-1] Which Can Be Used In g(x) Function Introduced In The Video Before.
Where D[K] Maps With f(x) And N[K] Maps With g(x).
Where g(0)=1 is defined.
g(x) = g(x-1) + f(x-1) x starts from 1
(N[k] Denoting The NUMBERING SEQUENCE)

4.1.4 Computer Code To Generate PARTITION SEQUENCE

The C++ code for PARTITION NUMBERS is given Below.

```
    #include <iostream>
using namespace std;
int main()
```

```
{
    cout << "Enter N smaller than or equal to 400" << endl;
    // enter the user_input till index 400 because more then that
    ↪  computation limit is reached.
    int user_input;
    cin >> user_input;
    long long int SEQ_OF_DIFFRENCES[50]; // Here I have created
    ↪  the sequence of diffrences
    for (long long int i = 0; i < 39; i++)
    {
        if (i % 2 == 0)
            SEQ_OF_DIFFRENCES[i] = (i / 2) + 1;
        else
            SEQ_OF_DIFFRENCES[i] = i + 2;
    }

    long long int NUMBERING_SEQUENCE[50]; // the
    ↪  NUMBERING_SEQUENCE array contains the position of + and -
    ↪  for the sequence
    NUMBERING_SEQUENCE[0] = 1;
    for (long long int i = 1; i < 40; i++)
    {
        NUMBERING_SEQUENCE[i] = NUMBERING_SEQUENCE[i - 1] +
        ↪  SEQ_OF_DIFFRENCES[i - 1];
    }

    long long int PARTITION_SEQUENCE[NUMBERING_SEQUENCE[39] + 1];
    ↪  // the PARTITION_SEQUENCE array conatins the sequence of
    ↪  partition numbers
    PARTITION_SEQUENCE[0] = 1;
    PARTITION_SEQUENCE[1] = 1;
    long long int j;
    ↪  // "j" finds which term of PARTITION_SEQUENCE will be
    ↪  added and which term of PARTITION_SEQUENCE will be
    ↪  subtracted
    long long int sum;
    ↪  // "sum" gives n^th term of PARTITION_SEQUENCE every time
    ↪  the loops run
    for (long long int i = 2; i < NUMBERING_SEQUENCE[35] + 1;
    ↪  i++) // Here the FOR loop calculates every term From '0'
    ↪  to 'user_input'
    {
        sum = 0;
        j = 0;
        while (i - NUMBERING_SEQUENCE[j] >= 0)
        {
```

```
            if (j % 4 == 0 || j % 4 == 1)
                sum = sum + PARTITION_SEQUENCE[i -
                ↪  NUMBERING_SEQUENCE[j]];
            else if (j % 4 == 2 || j % 4 == 3)
                sum = sum - PARTITION_SEQUENCE[i -
                ↪  NUMBERING_SEQUENCE[j]];
            j++;
        }
        PARTITION_SEQUENCE[i] = sum;
    }

    cout << "For The Number " << user_input << " The Total
    ↪  Partitions Are " << PARTITION_SEQUENCE[user_input] <<
    ↪  endl;

    return 0;
}
```

Two Test Cases Are Given Below.

TEST CASE 1:-

```
Enter N smaller than or equal to 400
125
For The Number 125 The Total Partitions Are 3163127352
```

TEST CASE 2:-

```
Enter N smaller than or equal to 400
379
For The Number 379 The Total Partitions Are 1812356499739472950
```

Lets Discuss Another Method For Finding Partitions For Any Number.

4.2 APPROACH 2

Here Our Approach Will Be To Make A Formula Which Can Generate The NUMBERING SEQUENCE.
Here I Will Solve The NUMBERING SEQUENCE By Generating A Formula, And Finally State A Algorithm For APPROACH 2.
From Approach 1, Lets Revisit the NUMBERING SEQUENCE.

IMAGE 4.1

The Blue Boxes In Figure Above Shows Us The Even Indexing Terms, And red Boxes Shows Us The Odd Indexing Terms.
Let Us First observe The Even Indexed Terms.

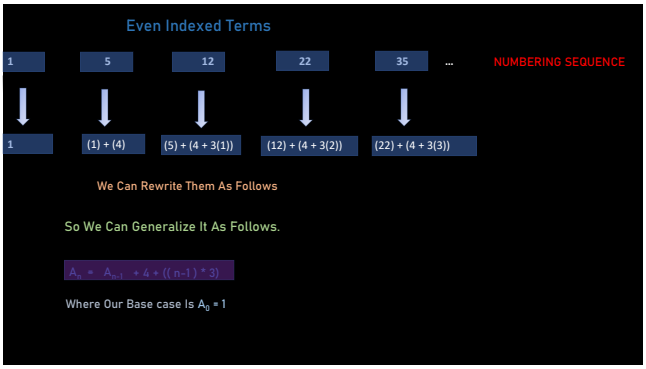


IMAGE 4.2

4.2.1 Solving Recurssion Relation For EVEN Indexing

As Shown In Figure We Have Our Recursion Relation For Even Terms As Follows.
$A_n = A_{n-1} + 4 + 3(n-1)$

To Solve This Recursion Function, This is a linear Non-Homogeneous Recurrence Relation.
First We will solve The Homogeneous Recursion Part Of It.
$A_n = A_{n-1}$ EQ-1
Where $A_n = x^n$
Let The Solution Of The Form $A_n{}^{(h)} = \mathrm{a}(x_1)$
Where a is Constant and x_1 is The Root of x.
Substituiting $A_n = x^n$ We get
$\therefore \mathrm{x}^n = x^{n-1}$
$\therefore \mathrm{x}^n - x^{n-1} = 0$

$\therefore x = 1$
Therefore The Homogeneous Solution Is Of The Form a

Now Solving For The Non-Homogeneous Part.
Let The Solution Of The Form $A_n{}^{(p)}$ = x(bx+c)
Substituiting $A_n{}^{(p)}$ = n(bx+c) in $A_n = A_{n-1} + 4 + 3(n-1)$ We get.
$\therefore x(bx + c) = (x-1)(b(x-1) + c) + 3x + 1$
$\therefore$ bx^2 + cx = (x-1)(bx-b+c) + 3x+1
By Simplifying We Get
$\therefore$ bx^2 + cx = b(x^2) + (3+c+2b)x + (b-c+1)
By Comparing The Coefficients Of x^2 and x we get.
b = 3/2 and c = 5/2
Thus The Overall Solution Of The Recurrence Relation is Of The Form
$A_n{}^{(h)}$ + $A_n{}^{(p)}$
So Our Solution Is $\frac{3x^2+5x}{2}$ + a
We have our Base Case $a_0 = 1$, By Using it we get a = 1
Thus our Final Solution is $\frac{3x^2+5x}{2}$ + 1

4.2.2 Solving Recurssion Relation For ODD Indexing

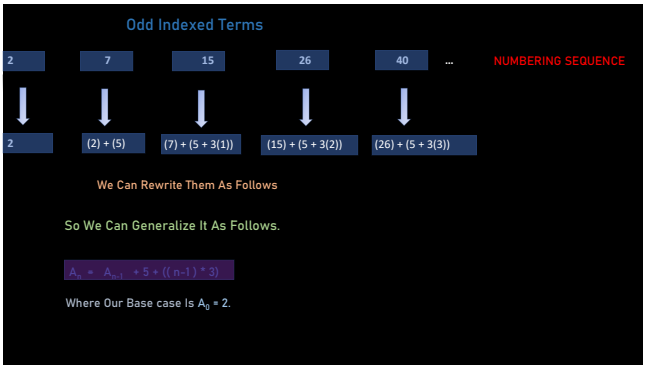


IMAGE 4.3

As Shown In Figure 4.3, We Have Our Recursion Relation For Even Terms As Follows.
$A_n = A_{n-1} + 5 + 3(n-1)$
To Solve This Recursion Function, This is a linear Non-Homogeneous Recurrence Relation.
First We will solve The Homogeneous Recursion Part Of It.
$A_n = A_{n-1}$ EQ-2
Where $A_n = x^n$
Let The Solution Of The Form $A_n{}^{(h)}$ = a(x_1)
Where a is Constant and x_1 is The Root of x.
Substituiting $A_n = x^n$ We get
$\therefore$ $x^n = x^{n-1}$
$\therefore$ $x^n - x^{n-1} = 0$

$\therefore x = 1$
Therefore The Homogeneous Solution Is Of The Form a

Now Solving For The Non-Homogeneous Part.
Let The Solution Of The Form ${A_n}^{(p)}$ = x(bx+c)
Substituiting ${A_n}^{(p)}$ = n(bx+c) in $A_n = A_{n-1} + 5 + 3(n-1)$ We get.
$\therefore x(bx + c) = (x - 1)(b(x - 1) + c) + 3x + 2$
$\therefore$ bx^2 + cx = (x-1)(bx-b+c) + 3x+2
By Simplifying We Get
$\therefore$ bx^2 + cx = b(x^2) + (3+c+2b)x + (b-c+2)
By Comparing The Coefficients Of x^2 and x we get.
b = 3/2 and c = 7/2
Thus The Overall Solution Of The Recurrence Relation is Of The Form
${A_n}^{(h)}$ + ${A_n}^{(p)}$
So Our Solution Is $\frac{3x^2+7x}{2}$ + a
We have our Base Case $a_0 = 1$, By Using it we get a = 2
Thus our Final Solution is $\frac{3x^2+7x}{2}$ + 2

Thus Now We Can Generate NUMBERING SEQUENCE For Both ODD And EVEN Indexing.

4.2.3 ALGORITHM For Approach 2

For PARTITION SEQUENCE Our Base Case Is P[0] = 1 And P[1] = 1 .(Where P[k] Denotes K [th] Element of The Partition Sequence) We Will Start K from 2 And Apply The Algorithm And Increase K by 1 And Repeat The Algorithm Until It reaches 'N' Where Our Aim Is To Find P[N].

1. To Find P[k] We Will Need N[k] (N[k] Denoting The NUMBERING SEQUENCE).
2. Even Terms Of N[k] Can be Found Using $\frac{3x^2+5x}{2}$ + 1 And Odd Terms of N[k] can Be Found Using $\frac{3x^2+7x}{2}$ + 2 (NOTE :- Where Input x = $\lfloor \frac{k}{2} \rfloor$)

Now You Can Find P[k]

4.2.4 C++ CODE For Approach 2

C++ Code Following Approach 2 Is Given Below.

```
   #include <iostream>
using namespace std;
int main()
```

```
{
     cout << "Enter N smaller than or equal to 400" << endl;
    // enter the user_input till index 400 because more then that
    ↪  LONG LONG INT cannot hold the number so it throws some
    ↪  garbage values .
    int user_input;
    cin >> user_input;
    long long int NUMBERING_SEQUENCE[50]; // the
    ↪  NUMBERING_SEQUENCE array contains the position of + and -
    ↪  for the sequence
    NUMBERING_SEQUENCE[0] = 1;
    for (long long int i = 1; i < 50; i++)
    {
        if(i%2==0)
        {
            int r = i/2;
            NUMBERING_SEQUENCE[i] = ((3*r*r)+(5*r)+2)/2;
        }
        else
        {
            int r = i/2;
            NUMBERING_SEQUENCE[i] = ((3*r*r)+(7*r)+4)/2;
        }
    }

    long long int PARTITION_SEQUENCE[NUMBERING_SEQUENCE[40] + 1];
    ↪  // the PARTITION_SEQUENCE array conatins the sequence of
    ↪  partition numbers
    PARTITION_SEQUENCE[0] = 1;
    PARTITION_SEQUENCE[1] = 1;
    long long int j;   // "j" finds which term of
    ↪  PARTITION_SEQUENCE will be added and which term of
    ↪  PARTITION_SEQUENCE will be subtracted
    long long int sum; // "sum" gives n^th term of
    ↪  PARTITION_SEQUENCE every time the loops run
    for (long long int i = 2; i < NUMBERING_SEQUENCE[39] + 1;
    ↪  i++)
    {
        sum = 0;
        j = 0;
        while (i - NUMBERING_SEQUENCE[j] >= 0)
        {
            if (j % 4 == 0 || j % 4 == 1)
                sum = sum + PARTITION_SEQUENCE[i -
                ↪  NUMBERING_SEQUENCE[j]];
            else if (j % 4 == 2 || j % 4 == 3)
```

```
                sum = sum - PARTITION_SEQUENCE[i -
                ↪  NUMBERING_SEQUENCE[j]];
            j++;
        }
        PARTITION_SEQUENCE[i] = sum;
    }
     cout << "For The Number " << user_input << " The Total
     ↪  Partitions Are " << PARTITION_SEQUENCE[user_input] <<
     ↪  endl;
    return 0;
}
```

Two Test Cases Are Given Below.

TEST CASE 1:-
Enter N smaller than or equal to 400
85
For The Number 85 The Total Partitions Are 30167357

TEST CASE 2:-
Enter N smaller than or equal to 400
327
For The Number 327 The Total Partitions Are 60105349839666544

5 CONNECTION BETWEEN PRIMES AND PARTITIONS

Here I will show you one amazing connection between PARTITION NUMBERS and PRIME NUMBERS.
We will use the NUMBERING SEQUENCE (For a Quick Recap it gives indexing of '+' and '-') we have defined earlier.

Lets define a new sequence named "PRIME CHECKER", to Generate "PRIME CHECKER" we will define a counter box whose value starts from 1 and after every operation it increases by 1.
To define this "PRIME CHECKER", we will take use of NUMBERING SEQUENCE.
We will place the "NUMBERING SEQUENCE" below the Counter Box , and we will perform Operation on it Denoted by the NUMBERING SEQUENCE.

5.0.1 Step By Step Procedure

1. Compute Arithmetic operation using The COUNTER BOX, N-1 terms of the PRIME CHECKER Sequence and The Instruction Given by NUMBERING

SEQUENCE of '+' and '-' to generate the N^{th} Term of PRIME CHECKER.
2. As the N^{th} term is Assigned value, increase the COUNTER BOX by 1.
3. Shift the NUMBERING SEQUENCE to right by ONE unit.
4. Repeat all the Three Steps again.

5.0.2 Illustrating Using The Rules

The Initial case is given below.

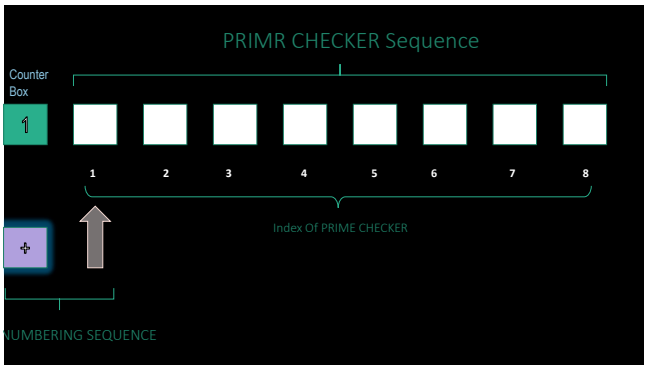


IMAGE 5.1

STEP 1:-
As shown in figure now we will perform 1+0 which will give us 1 in the first White box which Denotes First Term of PRIME CHECKER sequence.
STEP 2:-
After which COUNTER BOX increases by 1 and becomes 2.
The figure is given below.

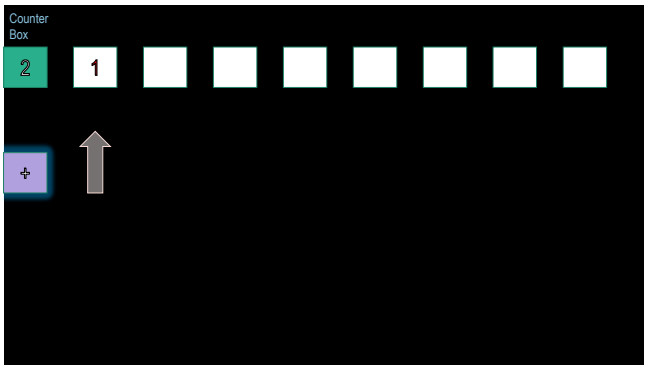


IMAGE 5.2

STEP 3:-
Now the "NUMBERING SEQUENCE" will shift one unit to right.

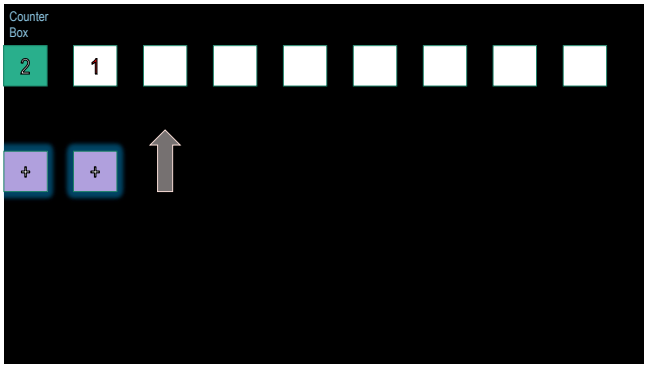


IMAGE 5.3

Repeating All three Steps for generating more Terms of PRIME CHECKER Sequence.
STEP 1:-
Now the operation 2+1 occurs which results in value 3 for the next box.
STEP 2:-
And the counter increases by 1 making it to value 3.

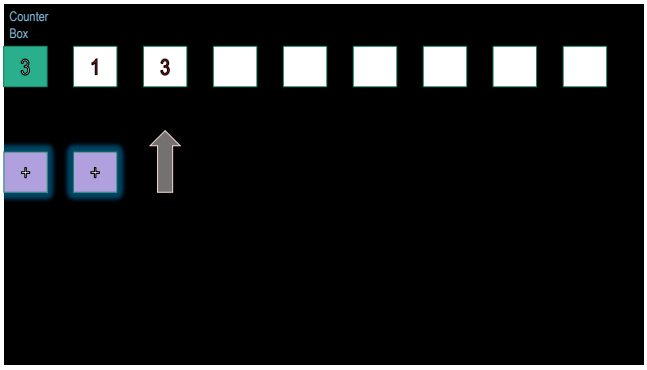


IMAGE 5.4

STEP 3:- Now the "NUMBERING SEQUENCE" will shift one unit to right.

Now this process goes on , I have illustrated it for some more terms , which is given below.

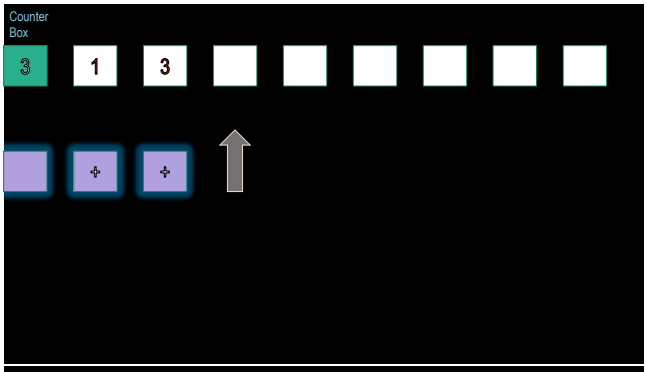


IMAGE 5.5

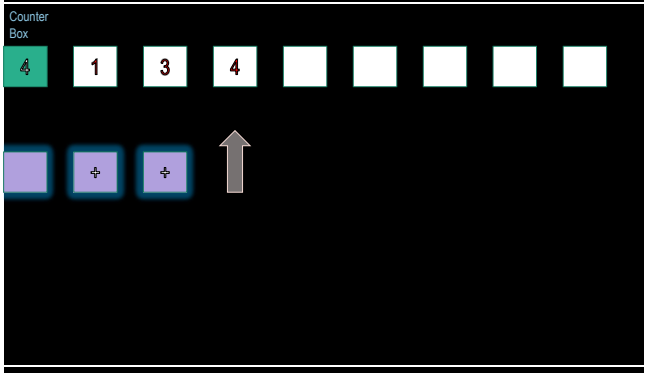


IMAGE 5.6

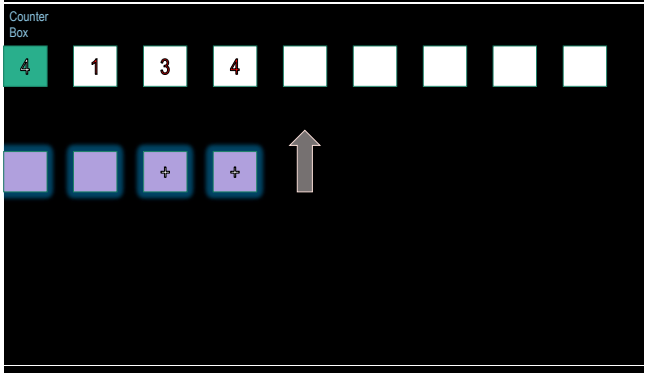


IMAGE 5.7

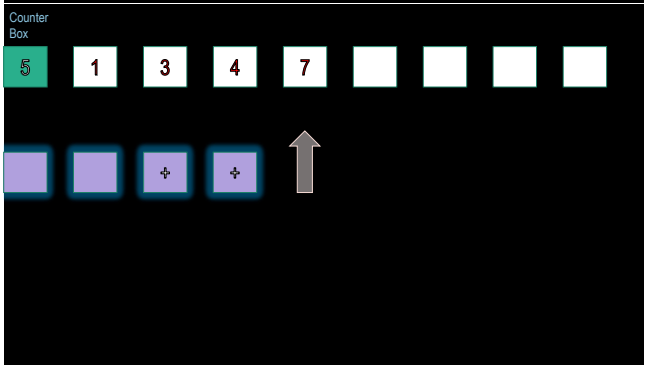


IMAGE 5.8

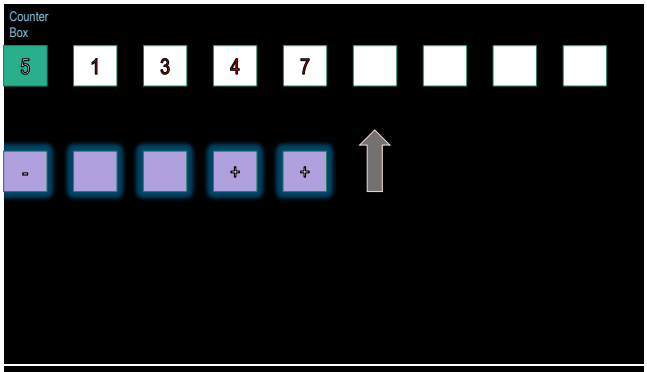


IMAGE 5.9

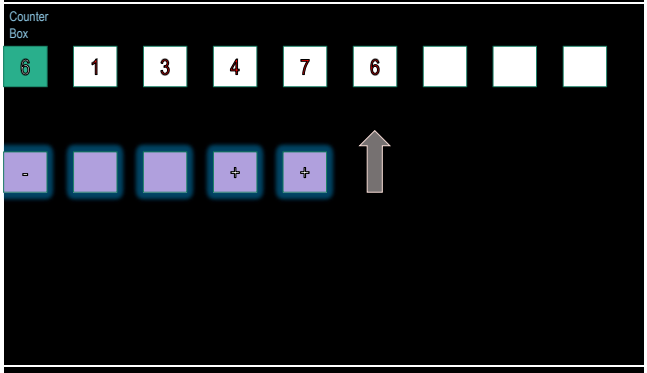


IMAGE 5.10

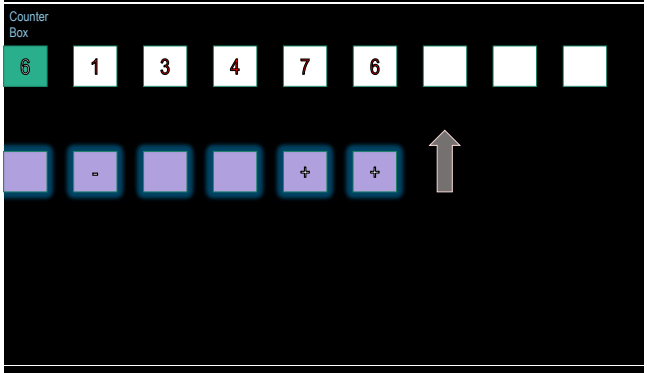


IMAGE 5.11

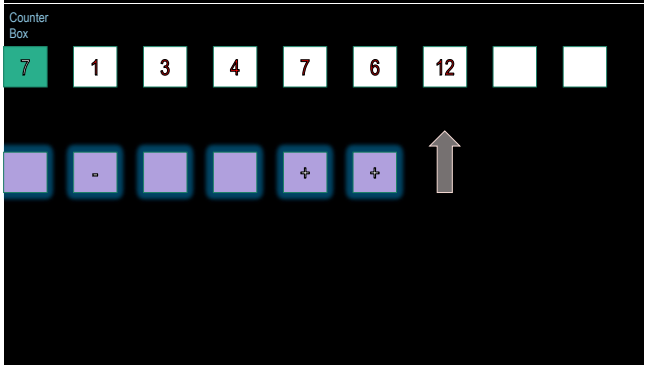


IMAGE 5.12

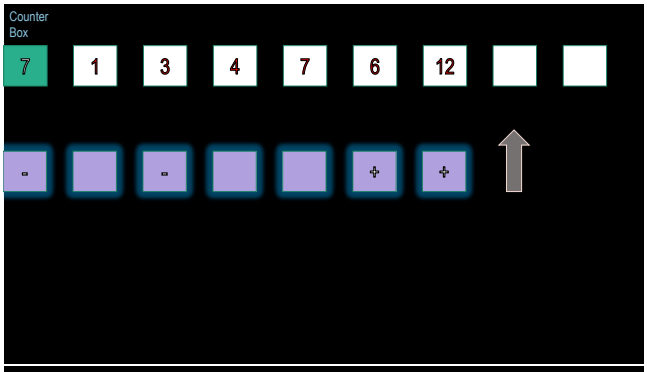


IMAGE 5.13

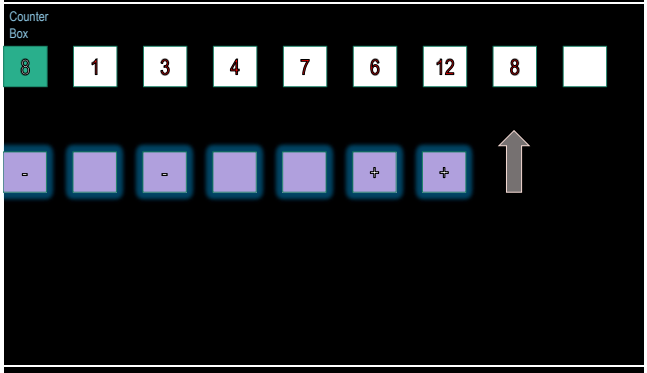


IMAGE 5.14

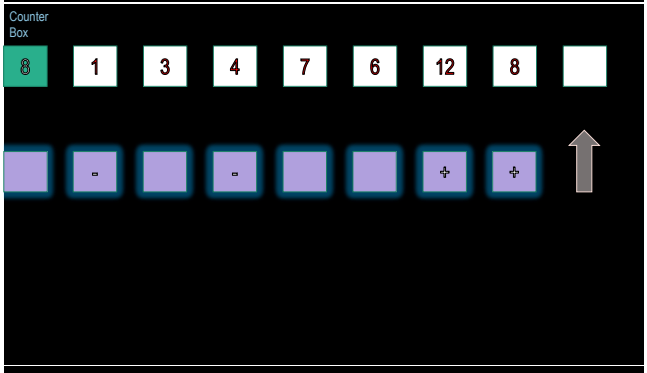


IMAGE 5.15

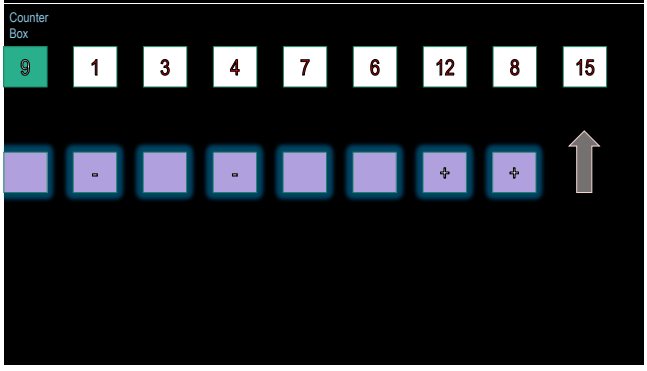


IMAGE 5.16

This process can be repeated as long as you want.
After seeing so many Illustrations, have you observed any pattern or any behaviour ?

5.0.3 Showing Relation Between Prime And Partitions

Okay Observe the Image 5.3 where the counter box has value 2 and the arithmetic operation happens is 2+1 which gives 3 , which is shown in Image 5.4.
Lets take one more case from Image 5.11, Where counter box is 6 and arithmetic operation is 7+6-1 which gives 12 as shown in Image 5.12.

When the Counter Box has value N, the result of N^{th} Term of PRIME CHECKER is the sum of all the factors of N

This Property is very useful, Let the counter box has Prime Number N, then it's factors are 1 and N itself, so here the output we will get is N+1 in the "PRIME CHECKER" series.

So By using the "PRIME CHECKER" series we can identify whelther the number is prime or not.
So for Eg see the Image 5.9, Where Counter Box is 5 which is prime so we expect the result should be 6, also you can see in Image 5.10 that the answer is 6.

The Sequence we created here "PRIME CHECKER" and the method we used is called EULER TWISTED MACHINE.

6 Refrences

1. Partition (number theory) - Wikipedia
2. Partition function (number theory) - Wikipedia
3. Integer Partitions by T. Geetha - Youtube Video