Project Euler

Solutions

David Esparza Alba

Contents

80 - Square root digital expansion

It is well known that if the square root of a natural number is not an integer, then it is irrational. The decimal expansion of such square roots is infinite without any repeating pattern at all.

The square root of two is 1.41421356237309504880..., and the digital sum of the first one hundred decimal digits is 475.

For the first one hundred natural numbers, find the total of the digital sums of the first one hundred decimal digits for all the irrational square roots.

Solution

To solve this problem we used the method of *Square Root by Subtraction* described by Jarvis [1]. This method allows us to find the square root of an integer n.

The algorithm 1 receives an integer n and returns the first one hundred decimal digits of $\sqrt{n}$. The value of p is set according of the number of decimal digits we want to calculate.

Algorithm 1 $squareRoot(n)$

$a = 5n$
$b = 5$
$p = 10^{101}$
while $b < p$ **do**
  if $a \geq b$ **then**
    $a \leftarrow a - b$
    $b \leftarrow b + 10$
  else
    $a \leftarrow a \times 100$
    $b \leftarrow \lfloor b/10 \rfloor \times 100 + 5$
  end if
end while
$b \leftarrow \lfloor b/100 \rfloor$
return b

91 - Right triangles with integer coordinates

The points $P(x_1, y_1)$ and $Q(x_2, y_2)$ are plotted at integer co-ordinates and are joined to the origin, $O(0, 0)$, to form ΔOPQ.

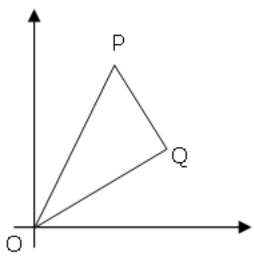

Figure 1

There are exactly fourteen triangles containing a right angle that can be formed when each co-ordinate lies between 0 and 2 inclusive; that is, $0 \leq x1, y1, x2, y2 \leq 2$.

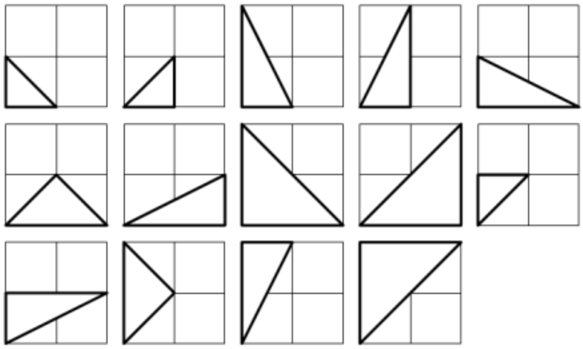

Figure 2

Given that $0 \leq x1, y1, x2, y2 \leq 50$, how many right triangles can be formed?

Solution

Since the coordinates are integer numbers in the range of $[0, 50]$, a brute force approach will run in $O(n^4)$, with $n = 50$ we are talking around six millions of operations, which looks reasonable to run in less than a minute. Now, in order to improve a little bit the running time we will do the following:

1. Place the first point (x_1, y_1) in the grid. This will take $O(n^2)$ to go trough the whole grid.

2. For the other point (x_2, y_2) iterate trough all values in the $y-axis$ and using a binary search obtain the $x - coordinate$, this part will run in $O(n \log n)$. Avoid to count the points where $y_2 = y_1$ and those where $x_2 = x_1$. To identify if we have found a valid coordinate we must check that the following statement is true (cosine law).

$$x_1^2 + y_1^2 + (x_2 - x_1)^2 + (y_2 - y1)^2 - x_2^2 - y_2^2 = 0$$

3. After counting the triangles f in step 2, we just need add $3n^2$, which is the number of right triangles with sides parallel to the x and y axis.

The running time of this solution is $O(n^3 \log n)$, which is around 7×10^5 operations.

94 - Almost equilateral triangles

It is easily proved that no equilateral triangle exists with integral length sides and integral area. However, the almost equilateral triangle 5-5-6 has an area of 12 square units.

We shall define an almost equilateral triangle to be a triangle for which two sides are equal and the third differs by no more than one unit.

Find the sum of the perimeters of all almost equilateral triangles with integral side lengths and area and whose perimeters do not exceed one billion (10000000000).

Solution

Two sides of the triangle are equal and the other differs by no more than one unit. One option is having lengths: $k, k, k+1$. Using the Heron's formula to obtain the triangle's area we have:

$$\begin{aligned} s &= \frac{k+k+k+1}{2} \\ &= \frac{3k+1}{2} \end{aligned}$$

and the triangle's area is given by

$$\begin{aligned} A_1 &= \sqrt{s(s-k)(s-k)(s-k-1)} \\ &= \sqrt{\left(\frac{3k+1}{2}\right)\left(\frac{k+1}{2}\right)\left(\frac{k+1}{2}\right)\left(\frac{k-1}{2}\right)} \\ &= \frac{(k+1)\sqrt{3k^2-2k-1}}{4} \end{aligned}$$

For lengths $k, k, k-1$ and following the same process we obtain that the area is

$$A_2 = \frac{(k-1)\sqrt{3k^2+2k-1}}{4}$$

Basically we are looking for values of k such as $3k^2 \pm 2k - 1$ is a square number, and that $(k \mp 1)(3k^2 \pm 2k - 1)$ is divisible by 4.

After generating the fist values of k I noticed that they are of the form $4i+1$, for some positive integers values of i. The upper bound for k is $k \leq M$, where $M = (10^9 - 1)/3$. So i started iterating trough all values of the form $4i+1$, for $i = 1, \ldots, M/4 - 1$ and checked if $3k^2 \pm 2k - 1$ was a square number, and if it was I added $3k \mp 1$ (the perimeter) to the result. The program ran in about 1.4 seconds in `C++`.

95 - Amicable chains

The proper divisors of a number are all the divisors excluding the number itself. For example, the proper divisors of 28 are 1, 2, 4, 7, and 14. As the sum of these divisors is equal to 28, we call it a perfect number.

Interestingly the sum of the proper divisors of 220 is 284 and the sum of the proper divisors of 284 is 220, forming a chain of two numbers. For this reason, 220 and 284 are called an amicable pair.

Perhaps less well known are longer chains. For example, starting with 12496, we form a chain of five numbers:

$$12496 \to 14288 \to 15472 \to 14536 \to 14264 (\to 12496 \to ...)$$

Since this chain returns to its starting point, it is called an amicable chain.

Find the smallest member of the longest amicable chain with no element exceeding one million.

Solution

For a number n the sum of the divisors less or equal than n is given by

$$S(n) = \prod_{i=1}^{m} \frac{p_i^{k_i+1} - 1}{p_i - 1}$$

where

$$n = \prod_{i=1}^{m} p_i^{k_i}$$

To generate the primes up to 10^6 quickly we can use the *Sieve of Eratosthenes.*

The idea is to mark the elements in a sequence and if we found a number that is already marked we will be able to obtain the length of the sequence and the starting/ending point of the cycle. We used an array L and a counter to obtain the length of a cycle. For the example mentioned in the description we will have

$$\begin{aligned}
L_{12496} &= 1 \\
L_{14288} &= 2 \\
L_{15472} &= 3 \\
L_{14536} &= 4 \\
L_{14264} &= 5
\end{aligned}$$

The next value is 12496 which is already marked, and the value of the counter is 6, so the length of the sequence is $6 - L_{12496} = 5$. Also we can store in another array X the starting/ending point of the cycle in each element of the chain, this is easily done using a recursive function.

$$\begin{aligned}
X_{12496} &= 12496 \\
X_{14288} &= 12496 \\
X_{15472} &= 12496 \\
X_{14536} &= 12496 \\
X_{14264} &= 12496
\end{aligned}$$

Sometimes we can start in a number that will led us to an already found cycle, just avoid counting the numbers found before entering the cycle.

Finally we just need to look for the number k which $k = X_k$, and have the greatest L_k. Start moving trough that cycle to find the smallest element.

98 - Anagramic squares

By replacing each of the letters in the word CARE with 1, 2, 9, and 6 respectively, we form a square number: $1296 = 36^2$. What is remarkable is that, by using the same digital substitutions, the anagram, RACE, also forms a square number: $9216 = 96^2$. We shall call CARE (and RACE) a square anagram word pair and specify further that leading zeroes are not permitted, neither may a different letter have the same digital value as another letter.

Using `words.txt` (right click and 'Save Link/Target As...'), a 16K text file containing nearly two-thousand common English words, find all the square anagram word pairs (a palindromic word is NOT considered to be an anagram of itself).

What is the largest square number formed by any member of such a pair?

NOTE: All anagrams formed must be contained in the given text file.

Solution

The longest word in the file has 14 letters, but looking for the anagram pairs I found out that the longest word with an anagram pair has 10 letters. So we only need to iterate through numbers less than 10^5 and store their squares (only the ones that has anagrams). For that I used a matrix, and in row 1 I stored the 1-digit squares, in row 2 the 2-digit squares, and so on.

To find words that are anagram pairs I used a map with the sorted word as key, because if we sort two anagrams we get the same string, and they will have the same key. The same approach is done to find the anagram squares, sort the digits and convert the resulting number into a string, with the leading zeros included, and use that string as key of a map, that help us to identify anagram squares.

I stored all the anagram pairs in a vector of pairs. For each pair $< A, B >$ I obtained the length of the words and started searching in the matrix in the corresponding row. For each number I checked if that number and word A were a valid match, for that I used two arrays, in one I was marking the digits that were already being used, and in the other array I stored the digit assigned to each letter. Doing that I could check that two letters didn't have the same digit assigned and also that one letter didn't have two different digits assigned.

Once we know the numeric value of each letter we build a new number using word B of the pair, and search for that number in the same row of the matrix, if the numbers are sorted we can use a binary search. If the number is found, then that pair is an *anagram word pair*. We repeat the process for each pair of words and report the largest square found. This approach run in 0.4 seconds approximately.

100 - Arranged probability

If a box contains twenty-one coloured discs, composed of fifteen blue discs and six red discs, and two discs were taken at random, it can be seen that the probability of taking two blue discs, $P(BB) = (15/21)(14/20) = 1/2$.

The next such arrangement, for which there is exactly 50% chance of taking two blue discs at random, is a box containing eighty-five blue discs and thirty-five red discs.

By finding the first arrangement to contain over $10^{12} = 1000000000000$ discs in total, determine the number of blue discs that the box would contain.

Solution

Be x the number of blue discs, and y the total number of discs. We are looking values of x and y such as

$$\begin{aligned}\left(\frac{x}{y}\right)\left(\frac{x-1}{y-1}\right) &= \frac{1}{2}\\ 2x^2 - 2x &= y^2 - y\\ 2x^2 - 2x - y^2 + y &= 0\end{aligned}$$

A Diophantine equation is a polynomial equation with integer solutions only. A linear Diophantine equation is

$$ax + by = c.$$

Which can be solved using the *Extended Euclidean* algorithm. For our case we have a quadratic Diophantine equation, and according to Mario Alpern's explanation [2], an equation of the form

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

has the following solutions:

$$\begin{aligned} x_{n+1} &= Px_n + Qy_n + K \\ y_{n+1} &= Rx_n + Sy_n + L \end{aligned}$$

For our specific equation the values of P, Q, K, R, S, L are:

$$\begin{aligned} P &= 3 \\ Q &= 2 \\ K &= -2 \\ R &= 4 \\ S &= 3 \\ L &= -3 \end{aligned}$$

The explanation of why these values is detailed in Mario Alpern's site.

Coming back to our problem. Using the formulas mentioned above, we only need to iterate though the different value of x and y until y exceeds 10^{12}.

101 - Optimum polynomial

If we are presented with the first k terms of a sequence it is impossible to say with certainty the value of the next term, as there are infinitely many polynomial functions that can model the sequence.

As an example, let us consider the sequence of cube numbers. This is defined by the generating function, $u_n = n^3 : 1, 8, 27, 64, 125, 216, \cdots$

Suppose we were only given the first two terms of this sequence. Working on the principle that "simple is best" we should assume a linear relationship and predict the next term to be 15 (common difference 7). Even if we were presented with the first three terms, by the same principle of simplicity, a quadratic relationship should be assumed.

We shall define $OP(k, n)$ to be the n^{th} term of the optimum polynomial generating function for the first k terms of a sequence. It should be clear that $OP(k, n)$ will accurately generate the terms of the sequence for $n \leq k$, and potentially the first incorrect term (FIT) will be $OP(k, k+1)$; in which case we shall call it a bad $OP(BOP)$.

As a basis, if we were only given the first term of sequence, it would be most sensible to assume constancy; that is, for $n \geq 2, OP(1, n) = u_1$.

Hence we obtain the following OPs for the cubic sequence:

$$\begin{aligned}
&OP(1,n) = 1 \;\; (1, 1, 1, 1, \ldots)\\
&OP(2,n) = 7n - 6 \;\; (1, 8, 15, \ldots)\\
&OP(3,n) = 6n^2 - 11n + 6 \;\; (1, 8, 27, 58, \ldots)\\
&OP(4,n) = n^3 \;\; (1, 8, 27, 64, 125, \ldots)
\end{aligned}$$

Clearly no BOPs exist for $k \geq 4$.

By considering the sum of FITs generated by the BOPs (indicated in red above), we obtain 1 + 15 + 58 = 74.

Consider the following tenth degree polynomial generating function:

$$u_n = 1 - n + n^2 - n^3 + n^4 - n^5 + n^6 - n^7 + n^8 - n^9 + n^{10}$$

Find the sum of FITs for the BOPs.

Solution

We are asked to obtain a polynomial model of degree $k-1$ ($1 \leq k \leq 10$) that fits the first k elements of u_n ($u_1, u_2, \ldots, u_k$) exactly, and then evaluate the polynomial model using the element $k+1$, and add the result to the answer.

Starting from the fact that if we have $n+1$ points, a polynomial of degree n will cross all points. In practice that is called *over-fitting*, and happens when the model is to complex to fit the data, and when we evaluate the model with other points, is possible that we get a larger error.

Then for k points, the polynomial model of degree $k-1$ we have to obtain has the following form:

$$OP(k,n) = \theta_0 + \theta_1 n_1 + \theta_2 n^2 + \cdots + \theta_{k-1} n^{k-1}$$

We need to find the values of θ, and then evaluate $OP(k,n)$ in $k+1$.

The equation above can be seen as a *linear model* of the form:

$$OP(k,n) = \theta_0 f_0(n) + \theta_1 f_1(n) + \theta_2 f_2(n) + \cdots + \theta_{k-1} f_{k-1}(n)$$

So we can use the *Least Squares* method to obtain the value of θ and get

$$\theta = (X^T X)^{-1} X^T y$$

where y are the values of u_n, and X is a matrix with element $X_{i,j}$ representing $f_j(i)$.

There are programming languages that have functions that estimates the value of θ. In my case I used `R`, which have the function `lm` that estimates the value of θ, and `predict`, which evaluates a model in a given point.

104 - Pandigital Fibonacci ends

The Fibonacci sequence is defined by the recurrence relation:

$$F_n = F_{n-1} + F_{n-2}, \text{where } F_1 = 1 \text{ and } F_2 = 1.$$

It turns out that F_{541}, which contains 113 digits, is the first Fibonacci number for which the last nine digits are 1-9 pandigital (contain all the digits 1 to 9, but not necessarily in order). And F_{2749}, which contains 575 digits, is the first Fibonacci number for which the first nine digits are 1-9 pandigital.

Given that F_k is the first Fibonacci number for which the first nine digits AND the last nine digits are 1-9 pandigital, find k.

Solution

Obtaining the last 10 digits of a Fibonacci number is not a problem, in fact you can store the last 10 digits only and ignore the rest, but what about the first 10 digits? Well for this case we used the *Binet's Formula* which states that

$$F_n = \frac{\phi^n - (1-\phi)^n}{\sqrt{5}}$$

where ϕ is the golden ratio and is defined by

$$\phi = \frac{1+\sqrt{5}}{2} \approx 1.6180339887\ldots$$

Since the value of $(1-\phi)^n$ is getting smaller as n increases we can redefine the equation as

$$F_n = \left[\frac{\phi^n}{\sqrt{5}} \right]$$

What we did was to generate some Fibonacci numbers, after that only kept track of the first 10 and the last 10 digits. The first k digits of a number n can be obtained with the following formula:

$$10^{\log n - k + 1}$$

Then

$$\begin{aligned} \log F_n &= \log \frac{\phi^n}{\sqrt{5}} \\ &= n \log \phi - 0.5 \log 5 \end{aligned}$$

Finally, once we have the first 10 digits and the last 10 digits we only need to check if both are pandigital, which can be easily verified, and this way we avoid the use of big numbers and all the carry operations.

206 - Concealed Square

Find the unique positive integer whose square has the form 1_2_3_4_5_6_7_8_9_0, where each "'_' is a single digit.

Solution

I'm not very happy with my solution here, but here it is. The first thing to notice is that we are looking for 9 digits between $[0, 9]$, making 10^9 possible combinations. Now, the number we are looking must have 10 digits and has the form:

$$d_0 d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8 d_9.$$

We know that d_9^2 must ends with 0. So the only choice is that

$$d_9 = 0$$

Also we know that

$$d_8^2 + d_7 d_9 = 9$$

Replacing d_9 with 0 we obtain

$$d_8^2 = 9$$

We have two solutions here, d_8 can be 3, or 7. What I did was to try both numbers, and give d_7 different values.

With the values of d_7, d_8, and d_9, we can find the the other seven digits by brute force in a reasonable amount of time.

323 - Bitwise-OR operations on random integers

Let $y_0, y_1, y_2, \ldots$ be a sequence of random unsigned 32 bit integers (i.e. $0 \leq y_i < 2^{32}$, every value equally likely).

For the sequence x_i the following recursion is given:

$$x0 = 0 \text{ and}$$
$$x_i = x_{i-1}|y_{i-1}, \text{ for } i > 0.(\ | \text{ is the bitwise-OR operator})$$

It can be seen that eventually there will be an index N such that $xi = 2^{32} - 1$ (a bit-pattern of all ones) for all $i \geq N$.

Find the expected value of N. Give your answer rounded to 10 digits after the decimal point.

Solution

We can see this as a game where the player wins when all 32 bits are 1. Define $s^{(t)}$ as the number of bits set to 1 after t games, and lets call $P_i^{(t)}$ the probability of having i 1's after t games. Initially we have that $P_i^{(1)}$ follows a binomial distribution

$$P_i^{(1)} = P(s^{(1)} = i) = \binom{32}{i}(0.5)^i(0.5)^{32-i}$$

since all bits are equally likely, the probability for a bit of having a value of 0 is 0.5.

Is important to notice that there will be at least one attempt, so $P(N = 1) = 1$.

Let's assume we did not win in the first attempt. All the bits set to 1 in the first attempt will remain as 1 (because of the bitwise-OR), and there is a chance that more bits are set to 1 in the second attempt, but the amount of 1's wont decrease.

We need to calculate the conditional probability of getting i ones, given that until the last attempt we had j ones. We will call this probability Q_{ij} $(i \geq j)$.

To calculate the value of Q_{ij} we know from the last attempt that j from the 32 bits are 1, and the other $32 - j$ bits are 0's. First we are going to place $i - j$ new bits to 1 (bits that were 0 after the last attempt). This follows a binomial distribution and we will call it q_a and is given by

$$q_a = \binom{32-j}{k}(0.5)^k(0.5)^{32-j-k},$$

where $k = i - j$.

We still need to deal with the bits that were 1 after the previous attempt. Well those bits don't matter if they are 0 or 1 in the next attempt, at the end they will remain with the value of 1. Be q_b the probability of getting at most j 1's in the next attempt in those bits. Its value is defined by

$$q_b = \sum_{h=1}^{j}\binom{j}{h}(0.5)^h(0.5)^{j-h}.$$

Then the value of Qij is defined as

$$Q_{ij} = q_a q_b$$

Once we know each value of Q_{ij}, the new value of $P_i^{(t+1)}$ is obtained by the formula of total probability:

$$
\begin{aligned}
P_i^{(t+1)} &= P(s^{(t+1)} = i) \\
&= P(s^{(t+1)} = i | s^{(t)} = 0) P(s^{(t)} = 0) + \cdots + P(s^{(t+1)} = i | s^{(t)} = i) P(s^{(t)} = i) \\
&= \sum_{j=0}^{i} P(s^{(t+1)} = i | s^{(t)} = j) P(s^{(t)} = j) \\
&= \sum_{j=0}^{i} Q_{ij} P_j^{(t)}
\end{aligned}
$$

The probability of keep playing (probability of losing) is given by $1 - P_{32}$. As we continue playing that value will approach to zero. If we continuously add that value we will find the value of $E(N)$. Then

$$
\begin{aligned}
E_1 &= 1 \\
E_i &= E_{i-1} + (1 - P_{32}^{(i)}) \text{ for } i \geq 2
\end{aligned}
$$

345 - Matrix Sum

We define the Matrix Sum of a matrix as the maximum sum of matrix elements with each element being the only one in his row and column. For example, the Matrix Sum of the matrix below equals 3315 ($= 863 + 383 + 343 + 959 + 767$):

7	53	183	439	**863**
497	**383**	563	79	973
287	63	**343**	169	583
627	343	773	**959**	943
767	473	103	699	303

Figure 3: Caption

Find the Matrix Sum of:

7	53	183	439	863	497	383	563	79	973	287	63	343	169	583
627	343	773	959	943	767	473	103	699	303	957	703	583	639	913
447	283	463	29	23	487	463	993	119	883	327	493	423	159	743
217	623	3	399	853	407	103	983	89	463	290	516	212	462	350
960	376	682	962	300	780	486	502	912	800	250	346	172	812	350
870	456	192	162	593	473	915	45	989	873	823	965	425	329	803
973	965	905	919	133	673	665	235	509	613	673	815	165	992	326
322	148	972	962	286	255	941	541	265	323	925	281	601	95	973
445	721	11	525	473	65	511	164	138	672	18	428	154	448	848
414	456	310	312	798	104	566	520	302	248	694	976	430	392	198
184	829	373	181	631	101	969	613	840	740	778	458	284	760	390
821	461	843	513	17	901	711	993	293	157	274	94	192	156	574
34	124	4	878	450	476	712	914	838	669	875	299	823	329	699
815	559	813	459	522	788	168	586	966	232	308	833	251	631	107
813	883	451	509	615	77	281	613	459	205	380	274	302	35	805

Figure 4: Caption

Solution

This problem can be seen as a Minimum Bipartite Matching Problem. From one side we have the rows, and in the other side we have the columns, and we must find the minimum 1-1 matching. In other words, for each row we must assign only one column, and the sum of all matches must be minimum.

For this problem we can use *Hungarian Algorithm*, which if my memory is correct runs in $O(n^4)$, where n is the size of the matrix. In [3] there is a great explanation of the algorithm.

357 - Prime generating integers

Consider the divisors of 30 : $1, 2, 3, 5, 6, 10, 15, 30$. It can be seen that for every divisor d of 30, $d + 30/d$ is prime
.

Find the sum of all positive integers n not exceeding 100000000 such that for every divisor d of n, $d + n/d$ is prime.

Solution

The sequences for the first ten numbers are the following:

$$
\begin{aligned}
2 &: 3, 3 \\
3 &: 4, 4 \\
4 &: 5, 4, 5 \\
5 &: 6, 6 \\
6 &: 7, 5, 5, 7 \\
7 &: 8, 8 \\
8 &: 9, 6, 6, 9 \\
9 &: 10, 6, 10 \\
10 &: 11, 7, 7, 11
\end{aligned}
$$

Is easy to notice that the first element in the sequence is $n + 1$, then $n + 1$ must be prime. Then the integers we are looking are even, since $n + 1$ must be prime, and all prime numbers except 2 are odd.

So far we know that we are looking for even numbers, and since all even numbers have 2 as a divisor, then $n/2+2$ must be prime as well. Then we are looking even numbers such as, $n+1$ is prime and $n/2+2$ is prime.

I found out (by try and error) that these integers, except 2 and 6 ,are of the form of $10+20a, 18+20b$, and $22+20c$, for $a, b, c \geq 0$. Now, not every number is valid, so we must check if all divisors d of those numbers meet that $d+n/d$ is prime. From sequences above we can notice the sequences are mirrored from half, then we only need to search for divisors up to $\sqrt{n}$, which will run a lot faster.

381 - (prime-k) factorial

For a prime p let $S(p) = \left(\sum_{k=1}^{5}(p-k)!\right) \bmod p$.

For example, if $p = 7$,

$$\begin{aligned}\sum_{k=1}^{5}(p-k)! &= (7-1)! + (7-2)! + (7-3)! + (7-4)! + (7-5)! \\ &= 6! + 5! + 4! + 3! + 2! \\ &= 720 + 120 + 24 + 6 + 2 \\ &= 872.\end{aligned}$$

As $872 \bmod 7 = 4$, $S(7) = 4$.

It can be verified that $\sum_{p=5}^{99} S(p) = 480$.

Find $\sum_{p=5}^{10^8} S(p)$.

Solution

By doing simple math and using modular arithmetic we get the $S(p)$ is defined by

$$\begin{aligned}S(p) &= ((p-5)! \bmod p) \times 9 \\ &= \left(\left(\frac{(p-1)!}{(p-1)(p-2)(p-3)(p-4)}\right) \bmod p\right) \times 9\end{aligned}$$

According to Wilson's Theorem [4],

$$(n-1)! \bmod n = -1 \bmod n$$

When n is prime.

Then, we only need to get rid of the denominator by using modular arithmetic. More precisely by using the inverse multiplicative. Suppose we have the following operation

$$\frac{a}{b} \bmod m$$

we can rewrite it as

$$(a \bmod m)\left(\frac{1}{b} \bmod m\right)$$

The inverse multiplicative of b is a number k, such that $(bk) \bmod m = 1$. Getting back to our problem, we are looking for an integer k such that

$$k(p-1)(p-2)(p-3)(p-4) \bmod p = 1$$

Using modular arithmetic we have that

$$\begin{aligned}(k(-1)(-2)(-3)(-4)) \bmod p = 1 \\ 24k \bmod p = 1\end{aligned}$$

Then

$$\begin{aligned}S(p) &= \left(\frac{(p-1)!}{24} \bmod p\right) \times 9 \\ &= \left((p-1)!\frac{3}{8}\right) \bmod p \\ &= (-1 \bmod p)(3k \bmod p) \\ &= (-3 \bmod p)(k \bmod p) \\ &= (p-3)(k \bmod p)\end{aligned}$$

where $8k \bmod p = 1$

The value of k can be found using the *Extended Euclidean Algorithm*, which finds the inverse multiplicative of a number $a \bmod n$, only when a and m are coprimes. Since for our problem m is prime, we can use it.

The *Extended Euclidean Algorithm* find the values of x and y that solve the following equation:

$$ax + by = 1$$

Using $a = 8$ and $b = p$ we have

$$8x + py = 1$$

Obtaining the modulus in both sides of the equation we get

$$\begin{aligned}(8x + py) \bmod p &= 1\\ 8x \bmod p &= 1\end{aligned}$$

which is what we are looking for. So basically we must calculate the value of x using the *Extended Euclidean Algorithm* and add the value of p to avoid negative values. Then

$$(x, y) = \text{ExtendedEuclidean}(8, p)$$

and

$$k = (x + p) \bmod p$$

Finally the value of $S(p)$ is given by

$$S(p) = ((p - 3) \times k) \bmod p$$

429 - Sum of squares of unitary divisors

A unitary divisor d of a number n is a divisor of n that has the property $gcd(d, n/d) = 1$. The unitary divisors of $4! = 24$ are 1, 3, 8 and 24. The sum of their squares is $1^2 + 3^2 + 8^2 + 24^2 = 650$.

Let $S(n)$ represent the sum of the squares of the unitary divisors of n. Thus $S(4!) = 650$.

Find $S(100000000!)$ modulo 1000000009.

Solution

First of all we must avoid calculating $n!$ directly. A better approach is to represent $n!$ as the product of its prime factors. For example, for 4! we have

$$4! = 2^3 \times 3^1.$$

Since we only care about those divisors with $gcd(d, n/d) = 1$, we must represent the number as the product of factors with exponent 1. For 4! we obtain

$$4! = 8^1 \times 3^1.$$

The result we are looking for is the sum of the products of all combinations of the square of those factors. For this specific case we have that

$$S(4!) = 1 + 8^2 + 3^2 + 8^2 3^2 = 650$$

To calculate that sum we can use the following trick. Suppose there 3 factors, f_1, f_2, f_3. The sum of the products of all combinations of those factors is given by:

$$(1 + f_1)(1 + f_2)(1 + f_3) = 1 + f_1 + f_2 + f_3 + f_1 f_2 + f_1 f_3 + f_2 f_3 + f_1 f_2 f_3.$$

For our problem, if we have m factors the solution would be given by

$$\begin{aligned} S(n!) &= \prod_{i=1}^{m} (1 + f_i^2) \\ &= \prod_{i=1}^{m} (1 + p_i^{2k_i}) \end{aligned}$$

To obtain the value of $p_i^{2k^i}$ we can use binary exponentiation, in that case we will be dealing only with sums and products, and we can make use of the properties of modular arithmetic to keep the result modulo 1000000009.

491 - Double pandigital number divisible by 11

We call a positive integer double pandigital if it uses all the digits 0 to 9 exactly twice (with no leading zero). For example, 40561817703823564929 is one such number.

How many double pandigital numbers are divisible by 11?

Solution

We can use some tricks to solve this problem. First of all, to know if a number is divisible by 11 we can sum all its digits in even positions and subtract the result by the sum of all digits in odd positions, if the result is divisible by 11, then the original number is also divisible by 11.

Knowing that we can generate only the digits in even positions using backtracking and keeping track of how many digits we have left. For that we can use an array X of 10 elements, with initially $X_i = 2$, for each $i = 0, \ldots, 9$, and every time we use the digit k we decrease the value of X_k by one.

When we have placed the ten digits in even positions we only need to calculate if the number is divisible by 11 and obtain the number of combinations to place the remaining digits in odd positions, this will be given by

$$\binom{10}{X_1, X_2, \ldots, X_9} = \frac{n!}{X_1!X_2!\cdots X_9!}$$

The result is then added to our answer.

So far we will obtain a correct answer, but it can take a while to run.To improve the running time we can use memoization to store the results we are calculating in order to avoid to compute them again. To accomplish that we can represent the array X as a base-3 number, and use a matrix C, so every time we are placing a digit in position k, we can store the result in C_{kl}, where l is the base-10 representation of X.

516 - 5-smooth totients

5-smooth numbers are numbers whose largest prime factor doesn't exceed 5. 5-smooth numbers are also called Hamming numbers. Let $S(L)$ be the sum of the numbers n not exceeding L such that Euler's totient function $\phi(n)$ is a Hamming number.

$S(100) = 3728$.

Find $S(10^{12})$. Give your answer modulo 2^{32}.

Solution

To solve this problem we do the following:

- Store all numbers of the form $2^a 3^b 5^c$ in an array H.
- Obtain all prime numbers of the form $2^a 3^b 5^c + 1$ greater than 5 and store them in an array T. Since a prime number p has $\phi(p) = p - 1$, then all elements in T comply with $\phi(2^a 3^b 5^c + 1) = 2^a 3^b 5^c$.
- Sort the elements in H and T.
- Add the sum of all elements in H to the answer.
- Multiply each element in H to all combinations of products of the elements in T and add those values to the answer. Since the ϕ function is multiplicative , then for a any prime p in T we have

$$\begin{aligned}
\phi(p2^a 3^b 5^c) &= \phi(p)\phi(2^a)\phi(3^b)\phi(5^c) \\
&= (2^x 3^y 5^z)(2^{a-1})(3^{b-1})(5^{c-1}) \\
&= 2^{x+a-1} 3^{y+b-1} 5^{z+c-1}
\end{aligned}$$

To this last part we can use backtracking, just be careful to not multiply the same prime more than once, since $\phi(nm) = \phi(n)\phi(m)$ only if $gcd(n, m) = 1$.

Bibliography

[1] Frazer Jarvis (2002) http://www.afjarvis.staff.shef.ac.uk/maths/jarvisspec02.pdf

[2] Mario Alpern https://www.alpertron.com.ar/CUAD.HTM

[3] Dénes Kőnig and Jenő Egerváry (1955) https://en.wikipedia.org/wiki/Hungarian_algorithm

[4] Ibn al-Haytham https://en.wikipedia.org/wiki/Wilson%27s_theorem

Index