Counting Coprime

By Prajwal Bhagath

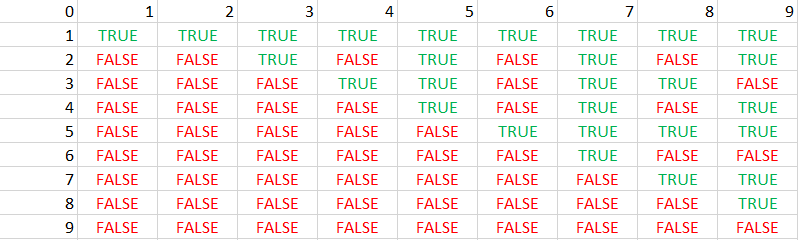
ID: 18039504

Coprime numbers are sets of two or more integers that have the greatest common divisor to be equal to one. The greatest common divisor or the GCD of two numbers is the largest number that divides them both. In this task we had to adapt the Sieve of Eratosthenes algorithm used in our week one tutorial and adapt it to find the sum of all coprime pairs within a set range between 0 and any positive integer.

The Sieve of Eratosthenes uses a one-dimensional vector of Boolean flags all set to true with an unsigned variable that holds the count of prime numbers, which starts from zero. In the function it uses a for loop going through each vector array one by one to check weather if it is true or false. If it is true, then it adds one to the count and sets the array space of the number multiplied by two to be false. When the loop comes the array of N\*2 it will not add a count since it is set to false.

In the algorithm used to count the Sum of coprime I used a one dimensional the same size as a 2D vector array by setting (X(Number of rows)\*X(Number of Columns)) all set to false. Although this is still a one-dimensional vector, we can still conceptualise it as a 2-dimensional vector. I then Set a counter variable starting at one. I then set two for loops one inside the other. The first for loop was used to navigate row by row and the second for loop was used navigate column by column. Inside the two for loops there is an if statement that calls the GCD function and If the function returns 1 then it sets that array located on the I’th row and j’th column from false to true and adds one to the count. Once the function comes out of the loop it returns the total count value.

The GCD function was adapted from Euclidean algorithm. This method will divide smaller number and will stop when it finds a remainder to equal zero. I put in 3 if statements within the function two of them were to return the other option if one of them was zero. The third if statement was to check if that the rows were not greater than the columns, because anything greater than or equal to the value of the column is automatically not a coprime (except for the exception of 1, hence why the counter started from one). If the row was less than the column then it will keep going through the loop until they get a modulus of zero, and then will return the greatest common divisor. Else if the row is greater than the column (i > j) it will return a value greater than 1, so it can automatically skip the if statement in the count\_coprimes function since (GCD(i,j) != 1).

To check my workings 

The grid above is a representation of my conceptualisation of the vector grid, with each column represented by j and each row represented by i. I manually found if a pair of numbers were coprime and changed the flag from false to true. I did this on a 9x9 grid and added up the pair of coprime numbers. Once I found all the numbers, I tested my program with each number and cross checked with the summed value (e.g. 9 = 28 coprime pairs, 8 = 22 coprime pairs, 7 = 18 coprime pairs etc.) and once I got every outcome correct I was confident.

My main function was adapted form the week one lab that we were provided with. How ever instead of having a constant variable I made the user input a positive integer. Using the number that has been inputted it calls the ‘count\_coprime’ function which returns the count of coprime pairs that exists between 0 and the entered value.

Referencing:

Euclidean algorithm: <https://www.geeksforgeeks.org/euclidean-algorithms-basic-and-extended/>