Goldbach conjecture:

1. Every even number > 2 = the sum of two prime numbers -> Binary Conjecture
2. First proposed by Christian Goldbach (Prussian mathematician) in 1742 in letter addressed to Leonhard Euler (Swiss mathematician)
   1. Originally claimed that “every number greater than two is an aggregate of three prime numbers” (1 was considered prime ) -> Ternary Conjecture
3. GB Conjecture published in *Meditationes Algebraicae* (Edward Waring – 1770)
4. 1973: Chen Jing Run (Chinese mathematician) proved that every sufficiently large even number is the sum of a prime and a number with at most two prime factors
5. Goldbach Partition: The pair of two prime numbers, p and q, where p + q = n and n being an even integer, is called a Goldbach partition of n

|  |  |  |  |
| --- | --- | --- | --- |
| Even Integer | 10 | 12 | 22 |
| Partitions | 3 + 7  5 + 5 | 5 + 7 | 3 + 19  5 + 17  11 + 11 |
| Total # Partitions | 2 | 1 | 3 |

* 1. Example Above shows the Even Integer, the individual Goldbach partitions pairs that make up the integer, and the total number of Goldbach Partitions

1. Source: <https://www.britannica.com/science/Goldbach-conjecture>, <https://uu.diva-portal.org/smash/get/diva2:1471524/FULLTEXT02.pdf>

Provided Python Version Analysis:

* Breakdown:
  + get\_primes(max\_number):
    - This function computes and returns all prime numbers up to max\_number.
    - It initializes an empty list primes.
    - It iterates through numbers from 2 to max\_number, checking each number's primality by dividing it by all the primes found so far.
    - If a number is prime, it's added to the list of primes.
    - Finally, it returns the list of prime numbers.
  + goldbach(value):
    - This function takes an even integer value and returns a list of pairs of prime numbers that sum up to value.
    - It initializes an empty list result.
    - It checks if value is even and greater than or equal to 4.
    - It generates a list of primes up to value.
    - It iterates through the list of primes and finds pairs of primes that sum up to value.
    - Each pair is added to the result list.
    - Finally, it returns the result list.
  + main():
    - This is the entry point of the program.
    - It first checks if command-line arguments are provided. If so, it reads data from files specified in the arguments; otherwise, it uses a default list of even numbers.
    - For each value in the data, it prints out the Goldbach pairs using the print\_goldbach() function.
  + print\_goldbach(value, prime\_list):
    - This function prints out the result of the Goldbach conjecture for a given value of n and its corresponding Goldbach pairs.
    - If there are no Goldbach pairs found for the value, it prints a message indicating so.
    - Otherwise, it prints the number of pairs found and each pair along with their sum.
  + readfile(filename):
    - This function reads data from a file and returns it as a list of integers.
    - It opens the file specified by filename, reads each line as an integer, and appends it to the data list.
    - Finally, it returns the data list.
  + Invocation of main():
    - The main() function is invoked at the end of the module, triggering the execution of the program.
* Inefficiency:
  + The inefficiency in the provided Python program lies in the get\_primes(max\_number) function, particularly in its method of determining primality. This function uses trial division to check for primality, which involves dividing the number by all previously found primes up to its square root.
  + While trial division is a valid method for determining primality, it becomes increasingly inefficient as the number of primes found grows. This is because the number of divisions required increases with the number of primes. As a result, the time complexity of the get\_primes function is approximately , where is the value of max\_number.
  + A more efficient approach for generating primes is to use the Sieve of Eratosthenes algorithm. The Sieve of Eratosthenes can generate prime numbers up to a given limit much faster than trial division by eliminating multiples of primes from a list of candidates.
  + By replacing the primality check in the get\_primes function with the Sieve of Eratosthenes algorithm, the program's overall efficiency can be significantly improved. This improvement would especially benefit scenarios where max\_number is relatively large.

Implementation of Goldbach Conjecture:

1. Input: Take an even integer n from the user. Ensure that the input is a positive even integer greater than 2. If the input is invalid, prompt the user to enter a valid input.
2. Generate Primes: Write a function to generate prime numbers up to a certain limit. Use methods like the Sieve of Eratosthenes (more efficient) or trial division to generate primes efficiently. Store these primes in a list or another suitable data structure.
3. Search for Goldbach Partition: Iterate through the list of prime numbers generated in step 2. For each prime , check if is also prime. If both and are prime, you have found a Goldbach partition for .
4. Output: If a Goldbach partition is found, print the pair of primes that sum up to . If no such partition is found, print a message indicating that the Goldbach conjecture is not yet proven for .