Sieve of Eratosthenes - GeeksforGeeks

Source: https://www.geeksforgeeks.org/sieve-of-eratosthenes/

Courses Tutorials Practice Jobs DSA Tutorial Interview Questions Quizzes Must Do Advanced DSA System Design Aptitude Puzzles Interview Corner DSA Python Technical Scripter 2026 Explore DSA Fundamentals Logic Building Problems Analysis of Algorithms Data Structures Array Data Structure String in Data Structure Hashing in Data Structure Linked List Data Structure Stack Data Structure Queue Data Structure Tree Data Structure Graph Data Structure Trie Data Structure Algorithms Searching Algorithms Sorting Algorithms Introduction to Recursion Greedy Algorithms Tutorial Graph Algorithms Dynamic Programming or DP Bitwise Algorithms Advanced Segment Tree Binary Indexed Tree or Fenwick Tree Square Root (Sqrt) Decomposition Algorithm Binary Lifting Geometry Interview Preparation Interview Corner GfG160 Practice Problem GeeksforGeeks Practice - Leading Online Coding Platform Problem of The Day - Develop the Habit of Coding DSA Course 90% Refund Sieve of Eratosthenes Last Updated : 23 Jul, 2025 Given a number n , find all prime numbers less than or equal to n. Examples: Input: n = 10 Output: [2, 3, 5, 7] Explanation: The prime numbers up to 10 obtained by Sieve of Eratosthenes are [2, 3, 5, 7]. Input: n = 35 Output: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31] Explanation: The prime numbers up to 35 obtained by Sieve of Eratosthenes are [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31]. Try it on GfG Practice Table of Content [Naive Approach] - Using Loop O(n*sqrt(n)) Time and O(1) Space [Efficient Approach] - Sieve of Eratosthenes [Naive Approach] - Using Loop O(n*sqrt(n)) Time and O(1) Space The Naive Approach for finding all prime numbers from 1 to n involves checking each number individually to determine whether it is prime . Step By Step Implementations: Loop through all numbers i from 2 to n . For each i , check if it is divisible by any number from 2 to i - 1 . If it is divisible, then i is not prime. If it is not divisible by any number in that range, then i is prime. [Efficient Approach] - Sieve of Eratosthenes The Sieve of Eratosthenes efficiently finds all primes up to n by repeatedly marking multiples of each prime as non-prime, starting from 2. This avoids redundant checks and quickly filters out all composite numbers. Step By Step Implementations: Initialize a Boolean array p rime[0..n] and set all entries to true , except for 0 and 1 (which are not primes). Start from 2, the smallest prime number. For each number p from 2 up to √n : If p is marked as prime(true): Mark all multiples of p as not prime(false), starting from p * p (since smaller multiples have already been marked by smaller primes). After the loop ends, all the remaining true entries in p rime represent prime numbers. C++ #include <iostream> #include <vector> using namespace std ; vector < int > sieve (int n) { // creation of boolean array vector < bool > prime (n + 1 , true); for (int p = 2 ; p * p <= n ; p ++) { if (prime [p] == true) { // marking as false for (int i = p * p ; i <= n ; i += p) prime [i] = false ; } } vector < int > res ; for (int p = 2 ; p <= n ; p ++){ if (prime [p]){ res . push_back (p); } } return res ; } int main (){ int n = 35 ; vector < int > res = sieve (n); for (auto ele : res){ cout << ele << ' ' ; } return 0 ; } Java class GfG { static int [] sieve (int n) { // creation of boolean array boolean [] prime = new boolean [n + 1] ; for (int i = 0 ; i <= n ; i ++) { prime [i] = true ; } for (int p = 2 ; p * p <= n ; p ++) { if (prime [p]) { // marking as false for (int i = p * p ; i <= n ; i += p) prime [i] = false ; } } // Count number of primes int count = 0 ; for (int p = 2 ; p <= n ; p ++) { if (prime [p]) count ++ ; } // Store primes in an array int [] res = new int [count] ; int index = 0 ; for (int p = 2 ; p <= n ; p ++) { if (prime [p]) res [index ++] = p ; } return res ; } public static void main (String [] args) { int n = 35 ; int [] res = sieve (n); for (int ele : res) { System . out . print (ele + " "); } } } Python def sieve (n): #Create a boolean list to track prime status of numbers prime = [True] * (n + 1) p = 2 # Sieve of Eratosthenes algorithm while p * p <= n : if prime [p]: # Mark all multiples of p as non-prime for i in range (p * p , n + 1 , p): prime [i] = False p += 1 # Collect all prime numbers res = [] for p in range (2 , n + 1): if prime [p]: res . append (p) return res if __name__ == "__main__" : n = 35 res = sieve (n) for ele in res : print (ele , end = ' ') C# using System ; using System.Collections.Generic ; class GfG { // Function to return all prime numbers up to n static List < int > sieve (int n) { // Boolean array to mark primes bool [] prime = new bool [n + 1]; for (int i = 0 ; i <= n ; i ++) { prime [i] = true ; } // Sieve of Eratosthenes for (int p = 2 ; p * p <= n ; p ++) { if (prime [p]) { for (int i = p * p ; i <= n ; i += p) { prime [i] = false ; } } } // Store primes in list List < int > res = new List < int > (); for (int i = 2 ; i <= n ; i ++) { if (prime [i]) { res . Add (i); } } return res ; } static void Main () { int n = 35 ; List < int > res = sieve (n); foreach (int ele in res) { Console . Write (ele + " "); } } } JavaScript function sieve (n) { // Create a boolean array to mark primes let prime = new Array (n + 1). fill (true); // 0 and 1 are not prime prime [0] = prime [1] = false

; // Apply Sieve of Eratosthenes for (let p = 2 ; p * p <= n ; p ++) { if (prime [p]) { // Mark all multiples of p as not prime for (let i = p * p ; i <= n ; i += p) { prime [i] = false ; } } } // Collect all primes into result array let res = []; for (let p = 2 ; p <= n ; p ++) { if (prime [p]) { res . push (p); } } return res ; } // Driver code let n = 35 ; let res = sieve (n); console . log (res . join (' ')); Output 2 3 5 7 11 13 17 19 23 29 31 Time Complexity: O(n*log(log(n))). For each prime number, we mark its multiples, which takes around n/p steps. The total time is proportional to n*(1/2 + 1/3 + 1/5 + ...). This sum over primes grows slowly and is approximately O(n*log(log(n))) making the algorithm very efficient. Auxiliary Space: O(n) Related articles How is the time complexity of Sieve of Eratosthenes is n*log(log(n))? Segmented Sieve . Sieve of Eratosthenes in O(n) time complexity Sieve of Sundaram Sieve of Atkin Comment Article Tags: Article Tags: Dynamic Programming Mathematical DSA Qualcomm VMWare MAQ Software GE sieve Prime Number number-theory + 6 More