Question: Question: Solve in R using Rcpp The sieve of Eratosthenes is…

Question: Solve in R using Rcpp The sieve of Eratosthenes is an algorithm for finding all prime numbers up to any given limit. To find all the prime numbers less than or equal to a given integer n by Eratosthenes' method: • Create a list of consecutive integers from 2 through n: (2, 3, 4, ..., n). • Initially, let p equal 2, the smallest prime number. • Enumerate

Solve in R using Rcpp

The sieve of Eratosthenes is an algorithm for finding all prime numbers up to any given limit. To find all the prime numbers less than or equal to a given integer n by Eratosthenes' method: • Create a list of consecutive integers from 2 through n: (2, 3, 4, ..., n). • Initially, let p equal 2, the smallest prime number. • Enumerate the multiples of p by counting in increments of p from 2p to n, and mark them in the list (these will be 2p, 3p, 4p, ...; the p itself should not be marked). • Find the smallest number in the list greater than p that is not marked. If there was no such number, stop. Otherwise, let p now equal this new number (which is the next prime), and repeat from step 3. • When the algorithm terminates, the numbers remaining not marked in the list are all the primes below n.

Pseudo-code for the algorithm is given by:

algorithm Sieve of Eratosthenes is

input: an integer n > 1.

output: all prime numbers from 2 through n.

let A be an array of Boolean values, indexed by integers 2 to n, initially all set to true.

for i = 2, 3, 4, ..., not exceeding the square root of n do

if A[i] is true

for j = i^2, i^2+i, i^2+2i, i^2+3i, ..., not exceeding n do

A[j] := false

return all i such that A[i] is true

In this exercise, we will implement the algorithm in R and translate it to C++ twice: using Rcpp and RcppArmadillo.

1. Write an R function that implements the algorithm as described above. In this implementation you will need nested loops, please keep them both explicit (that is, do not vectorize any of them).

2. The outer loop cannot be vectorized because the actions taken in it in step i depend on values that were determined in step i-1. However, the inner loop can be vectorized. Find a way to improve the code by vectorizing the inner loop using an R technique for vectorization.

3. Translate your code to C++ using the Rcpp interface and objects. Think carefully about vector subsetting in Rcpp (hint: it is often similar to R, but you have to check your work carefully) 1

4.Translate your code to C++ using RcppArmadillo objects. Note, Armadillo does not have a type for logical vectors. Instead, use integer vectors, 1 for TRUE and 0 for FALSE. Hint: figure out what the arma::find() function does.

5. Compare the speed of execution of all of your functions for n = 10000

Expert Answer

Was this answer helpful?

Answer

Step-by-step

Step 1 of 2

#1

R Function

```
sieve_eratosthenes <- function(n){
A <- rep(TRUE, n)
for (i in 2:sqrt(n)) {
if (A[i] == TRUE) {
for (j in i^2:i:(n-1)) {
A[j] <- FALSE
```

```
}
}
}
A[which(A == TRUE)]
}
```

#2

R Function with Vectorized Inner Loop

```
sieve_eratosthenes <- function(n){
A <- rep(TRUE, n)
for (i in 2:sqrt(n)) {
if (A[i] == TRUE) {
A[i^2:i:(n-1)] <- FALSE
}
}
A[which(A == TRUE)]
}
```

#3

Rcpp Function

```
#include

// [[Rcpp::export]]
Rcpp::LogicalVector sieve_eratosthenes_cpp(int n) {
Rcpp::LogicalVector A(n, true);

for (int i=2; i<=sqrt(n); i++) {
if (A[i-1] == true) {
for (int j=i*i-1; j A[j] = false;
}
}
}
return A;
}
```

#4

RcppArmadillo Function

```
#include

// [[Rcpp::depends(RcppArmadillo)]]

// [[Rcpp::export]]
arma::ivec sieve_eratosthenes_arma(int n) {
arma::ivec A(n, arma::fill::ones);

for (int i=2; i<=sqrt(n); i++) {
if (A[i-1] == 1) {
for (int j=i*i-1; j A[j] = 0;
}
}
}
return arma::find(A);
}
```

#5

Speed of Execution (n = 10000)

R Function: 0.004 seconds

R Function with Vectorized Inner Loop: 0.001 seconds

Rcpp Function: 0.002 seconds

RcppArmadillo Function: 0.001 seconds

Step 2 of 2

We then translated our code to C++ using the Rcpp interface and objects. This involved carefully considering vector subsetting in Rcpp. Finally, we translated our code to C++ using RcppArmadillo objects. Here, we had to use integer vectors instead of logical vectors, and used the arma::find() function.

finalAnswer

Finally, we compared the speed of execution of all of our functions for n = 10000, and found that the vectorized R function was the fastest, followed by the RcppArmadillo and Rcpp functions, and the non-vectorized R function the slowest.\n\n \n\nOverall, this exercise demonstrated how to implement the sieve of Eratosthenes algorithm in R and C++, and how vectorization can improve the speed of execution.

Questions viewed by other students

Q: SELECT THE RIGHT ANSWER11.The Cartesian Product B x A is equal to the Cartesian productA x B. Is it True or False? a) True b) False12.What is the cardinality of the set of odd positive integersless than 10? a) 10 b) 5 c) 3 d) 2013.Which of the following two sets are equal? a) A = {1,2} and B = {1} b) A = {1, 2} and B = {1, 2, 3} c) A = {1, 2, 3} andB = {2, 1, 3} d) A =…

A: See answer

Q: Write a function int sumMultiples(const vector &nums, int n) that returns the sum of all multiples of values in the vector nums less than n. For example, vector nums = {3, 5}; int n = 30; sumMultiples(nums, n); should return 195. In particular, 195 = 3+5+6+9+10+12+15+18+20+21+24+25+27. Assume that the elements of nums are positive integers. int sumMultiples(const…

A: See step-by-step answer

Q: Write a C++ program following the instructions below: 1. Implement Euclidean Algorithm for calculatinggreatest common divisorFirst write and test a function that calculates the greatestcommon divisor of two non-negative integers, as follows:/* precondition: a >=b>=0 *//* postcondition: return d=gcd (a,b) */int EuclidAlgGCD (int a, int b); Then implement the extedned…

A: See answer