Ubaid Ur Rehman

BSCS-5B

Reg. number: 00000123495

uurehman.bscs15seecs@seecs.edu.pk

Advanced Programming

Lab 5 Document (Sieve of Eratosthenes in Scala)

# Introduction

# This lab is about to implement Sieve of Eratosthenes in Scala

# Approach

Simple Functional programming approach.

Build .scala file and run it.

# Output:

Enter the number to which you want to find prime numbers: 1000

TreeSet(2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997)

SUCCESS: Completed without errors.

Input: 1000 Time taken: 141 ms

# Code:

/\*\*

\* Subject: Advanced Programming - Lab 5: Sieve of Eratosthenes

\*

\* Coded by: Ubaid ur Rehman

\* Email: uurehman.bscs15seecs@seecs.edu.pk

\* Date: October 19, 2017

\*/

import scala.annotation.tailrec // for @tailrec

import scala.collection.parallel.mutable // to make parSet mutable

import util.Random.nextInt

import io.StdIn.\_

import scala.compat.Platform

object AllPrime {

def main(args: Array[String]): Unit = {

print("Enter the number to which you want to find prime numbers: ")

val numProvided: Int = readInt()

val executionStart = Platform.currentTime

val setOfPrimeNumbers = sieveOfEratosthenes(numProvided)

val executionEnd = Platform.currentTime

println(setOfPrimeNumbers)

println("SUCCESS: Completed without errors.\nInput: " + numProvided + "\t\tTime taken: " + (executionEnd - executionStart) + " ms")

} // end main

// source: https://rosettacode.org/wiki/Sieve\_of\_Eratosthenes#Scala

def sieveOfEratosthenes(limit: Int) = {

val setOfPrimeNumbers: mutable.ParSet[Int] = mutable.ParSet.empty ++ (2 to limit) // mutable parallel set of prime numbers

val sqrtLimit = math.sqrt(limit).toInt // square root of limit in integer

// tail recursive function

@tailrec

def isPrime(numberGiven: Int): Unit = {

if (numberGiven <= sqrtLimit) {

/\* if numberGiven is in setOfPrimeNumbers, remove the series:

numberGiven^2, numberGiven^2 + numberGiven, numberGiven^2 + ^2numberGiven, numberGiven^2

+ 3numberGiven, ..., not exceeding limit.\*/

if (setOfPrimeNumbers contains numberGiven) setOfPrimeNumbers --= numberGiven \* numberGiven to limit by numberGiven

isPrime(numberGiven + 1) //recursive call for next integer

}

}

isPrime(2)

setOfPrimeNumbers

val sortedSetOfPrimeNumbers = collection.immutable.SortedSet[Int]() ++ setOfPrimeNumbers

sortedSetOfPrimeNumbers

}

}

Github link: https://github.com/uurehman/Sieve-of-Eratosthenes