UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Redes de Ingeniería

http://revistas.udistrital.edu.co/ojs/index.php/redes/index
http://dx.doi.org/10.14483/udistrital.jour.redes.2016.2.a01

REDES DE INGENIERÍA
[Rompiendo las barreras del conocimiento]

ARTÍCULO DE INVESTIGACIÓN CIENTÍFICA

Algoritmo recursivo funcional para verificación y monitoreo de la densidad de números primos

A recursive algorithm for verifying and monitoring density of prime numbers using Functional Programming

Omar Iván Trejos Buriticá[1]

Para citar este artículo: Trejos O.I. (2016). Algoritmo recursivo funcional para verificación y monitoreo de la densidad de números primos. *Revista Redes de Ingeniería*. 7(2), 116-126. Doi: 10.14483/udistrital.jour.redes.2016.2.a01

Recibido: 9-marzo-2016 / Aprobado: 25-julio-2016

Resumen

El presente artículo formula una solución recursiva, usando programación funcional, para la verificación y monitoreo de la densidad de los números primos que corresponde a la cantidad de números primos que se encuentran en un rango determinado. A partir de este planteamiento se hacen algunas reflexiones en torno a la relación entre matemática y programación de computadores, así como alrededor de las tendencias que se notan en la estimación progresiva de la densidad de los números primos cuando se hacen algunas modificaciones en los rangos de evaluación. El propósito de este artículo es poner a consideración de los lectores una solución simple y ágil en la solución del problema planteado, así como las reflexiones que de allí se derivan.

Palabras clave: algoritmo, densidad, números primos, programación de computadores, programación funcional, recursividad.

Abstract

This article presents a recursive and functional based form to solve the problem to verifying and monitoring density of prime numbers which means the proportional quantity of prime numbers in a specific range. From this approach you can find some thoughts around the relation between math and computer programming and the tendencies you can see in the progressive evaluation of prime numbers when you change the evaluation ranges. The proposal of this article is to show a simple and agile solution in the formulated problem and the thoughts around it.

Keywords: algorithm, computer programming, density, functional programming, prime numbers, recursion.

1. Ingeniero de Sistemas, Universidad Incca de Colombia; MSc. en Comunicación Educativa, Universidad Tecnológica de Colombia; PhD en Ciencias de la Educación, Universidad Tecnológica de Pereira; docente de planta, Universidad Tecnológica de Pereira. Correo electrónica: omartrejos@utp.edu.co

INTRODUCCIÓN

Un problema típico de las matemáticas lo constituye el análisis de la densidad de los números primos que consiste en el cálculo de la cantidad de números primos presentes en un rango definido por cotas o límites dentro de la recta de los números naturales. Este factor llamado densidad permite hacer algunas reflexiones en torno a las tendencias sobre las cuales se realizan análisis numéricos y matemáticos, que propenden por predecir comportamientos específicos de dichos números.

En relación con este problema y con sus posibles soluciones se han presentado algunas propuestas que resuelven problemas similares formulados por el mismo autor, en los cuales intenta abordar propuestas simples que sean claras para los estudiantes de primeros semestres de ingeniería de sistemas. De la misma manera, en la parte teórica, se acude a otros autores que han intentado relacionar las matemáticas con la programación de computadores en la búsqueda de resolver problemas que en otros tiempos han sido resueltos por métodos manuales y que en tiempos modernos se resuelven aprovechando las tecnologías computacionales y las facilidades que ellas proveen.

El problema a resolver podría resumirse en la construcción de un algoritmo simple y eficaz que permita verificar y monitorear la densidad de los números primos tanto en rangos definidos como en rangos variables. Establecer relaciones entre las matemáticas y la programación de computadores bajo el manto de un determinado paradigma de programación, desde la óptica de las soluciones simples, posibilita la incorporación de un conocimiento con significado para los estudiantes de primeros semestres de programación en Ingeniería de Sistemas y abre caminos para encontrar un sentido, desde el aprendizaje por descubrimiento, entre lo que están aprendiendo en las matemáticas y lo que intentan aplicar en la programación de computadores.

Lo novedoso del artículo es que presenta una solución simple que, al tiempo que permite monitorear y verificar la densidad de los números primos en rangos definidos en la recta de los números naturales, permite también hacer análisis de la tendencia en dicha densidad abriendo caminos al análisis numérico desde la óptica de la programación sobre la base de la inferencia y la predicción. Los límites para el correcto funcionamiento de esta propuesta de solución los establece la máquina (el computador), por esa razón se ha acudido al lenguaje de programación DrScheme, ya que es el que maneja rangos de números mucho más amplio que lenguajes más comerciales.

El objetivo de este artículo consiste en construir un algoritmo simple y eficaz que permita verificar y monitorear la densidad de los números primos tanto en rangos definidos como en rangos variables y que permita realizar observaciones sobre la tendencia de dicha densidad. Este artículo es producto del proyecto de investigación "Análisis pedagógico, instrumental y conceptual de algunos paradigmas de programación como contenido de la asignatura Programación I del Programa Ingeniería de Sistemas y Computación", aprobado por la Vicerrectoría de Investigaciones, Innovación y Extensión de la Universidad Tecnológica de Pereira.

Para el desarrollo de este artículo, como producto de investigación, se utilizó el compilador del lenguaje DrScheme entorno DrRacket versión 5.1. A manera de hipótesis se podría afirmar, desde una óptica investigativa, que siempre es posible encontrar caminos lógicos simples para resolver problemas que la matemática ha formulado desde hace mucho tiempo y que las tecnologías computacionales modernas proveen velocidades suficientes como para que dichas soluciones sean óptimas en su funcionamiento así no lo sean en su rendimiento. Este artículo se ajusta al estándar internacional IMRYD, es decir, primero presenta una introducción acompañado de una teoría, luego plantea la metodología dividida en dos partes (descripción y aplicación), posteriormente plantea los resultados y sobre ellos abre un espacio de discusión para llegar a unas conclusiones y a unas referencias bibliográficas que le dan corpus investigativo.

APRENDIZAJE SIGNIFICATIVO

Se conoce como aprendizaje significativo el modelo pedagógico a través del cual se le concede significado al conocimiento que se adquiere en un proceso de aprendizaje para que este pueda ser aprendido, asimilado, apropiado y aplicado por un camino más llano. Esta teoría fue formulada por David Paul Ausubel [1] y según sus planteamientos, la base del aprendizaje se resume en el concepto de significado, es decir, en la relación entre lo conceptual y lo vivencial, la conexión entre lo que se plantea desde lo puramente teórico, lo que se vive y las posibles relaciones que se heredan de que dicho conocimiento tenga una representación práctica aplicativa en la vida cotidiana.

Según la teoría del aprendizaje significativo, el aprendizaje se basa en tres principios: el conocimiento previo, el nuevo conocimiento y la actitud del estudiante [1]. El conocimiento previo consiste en el conjunto de conceptos, teorías y planteamientos que se tienen al momento de iniciar un proceso de aprendizaje. El nuevo conocimiento está constituido por el conjunto de conceptos, teorías y planteamientos que, de una u otra forma, llegan bajo el rótulo "nuevo", es decir, que no se conocía, que no se sabía o que no se había aprendido.

La actitud del estudiante se fundamenta en la motivación y la capacidad que el estudiante desarrolle para establecer relaciones entre el conocimiento previo y el nuevo conocimiento. Estas relaciones pueden ser de complementación, suplantación, renovación o innovación [2], de tal forma que el nuevo conocimiento, a la luz de estas relaciones, le conceden significado a lo adquirido bien sea este el puro conocimiento previo, el nuevo conocimiento o la complementación de ambos.

Aprendizaje por descubrimiento

Esta es una vertiente de las teorías cognitivas formulada por el Jerome Seymour Bruner. De acuerdo a su teoría el aprendizaje involucra tres procesos casi simultáneos; de una parte, la adquisición de la información nueva, en segunda instancia el proceso de manipulación del conocimiento para adecuarlo a nuevas aplicaciones y la comprobación de la forma en que se ha manipulado la información y su evaluación en cuanto a la conveniencia frente a las tareas de aplicación [3].

De esta forma, adquisición, transformación y evaluación de una materia específica involucran el despliegue de las habilidades cognitivas de alto nivel, aunque al mismo tiempo se pueden intuir unas implicaciones de orden secundario: por un lado en el proceso de adquisición se ha de tener en cuenta lo que corresponde a la confiabilidad de las fuentes, de otra, en el proceso de transformación se debe considerar con especial atención la capacidad que se pueda tener para aplicar dicha transformación en relación con el conocimiento de forma que permita aplicarlo a otras tareas que pueden ser nuevas y en lo que compete a la evaluación se hace necesario tener unos elementos de juicio bastante sólidos que poder verificar si la transformación de la información es la apropiada desde el ángulo de las necesidades.

En la teoría del aprendizaje por descubrimiento se concede una importancia especial a la experimentación y, por tanto, al descubrimiento como ese fundamento a través del cual el estudiante construye sus propias estructuras cognitivas y las alimenta. Según Bruner "el ser humano aprende más fácilmente todo aquello que descubre" [4], y de esta manera consagra a la exploración, a la experimentación, al ensayo y al error como los elementos sustanciales que posibilitan el aprendizaje en tiempos modernos.

El descubrimiento es el camino para que la seducción permita la presencia de lo insólito [5].

El ser humano pareciera tener una predisposición temprana para lo inusual de manera que se tome distancia de lo habitual, para fijar la atención completamente en aquellos que, siendo insólito, capta los sentidos y captura toda nuestra atención. Bruner acierta cuando sostiene que en un contexto académico formativo si se aprovecha lo insólito como elemento instructivo dentro del proceso de aprendizaje y se le permite al estudiante que eso que empieza a conocer sea "descubierto", podrá asimilarlo, apropiarlo, aplicarlo y evaluarlo de una forma más fácil.

Aquello que se considera insólito involucra un sentido direccional hacia el conocimiento que no es controlable fácilmente, sobre todo cuando se vive en un mundo mediado por las nuevas tecnologías de la información y la comunicación que constituyen un faro sin rumbo que abre posibilidades a la mente humana. Bruner le concede a la palabra "descubrimiento" el estatus de sinónimo de la palabra "fascinación" y ahí radica la esencia de su teoría de aprendizaje.

Teoría de números

Como teoría de números se conoce la rama de las matemáticas que se ocupa de estudiar las propiedades de los números, especialmente de aquellos conocidos como números enteros y de los problemas derivados de su estudio [6]. Este estudio implica algunos problemas que bien podrían entenderse como "problemas no matemáticos". Por lo tanto, podría decirse que la teoría de números se ocupa de estudiar los problemas cuando se estudian los números enteros.

La teoría de números, por su misma naturaleza ocupa una posición privilegiada similar a la relación que se puede establecer entre las matemáticas y otras ciencias del conocimiento. En algún sentido la teoría de números se calificaba con el nombre de "aritmética" en tiempos pasados; sin embargo, en la actualidad esta definición se usa poco y los problemas que surgen producto del estudio de la teoría de números se refieren con su mismo nombre.

Los griegos descubrieron las leyes fundamentales de la aritmética, concebida como la base que subyace a todo lo que constituye la ciencia matemática. Fueron ellos quienes llegaron a plantear elementos tan definidos como la división euclidea, características de los números primos, el cálculo del máximo común divisor y el mínimo común múltiplo y otros planteamientos que aún siguen vigentes en nuestros días.

Lo que plantearon los griegos, dando inicio a la teoría de números, es que era posible definir la existencia de un lenguaje de los números que no significa que necesariamente ese lenguaje pudiera ser entendido plenamente. Un símil de nuestros tiempos podría ser el sistema de comunicación de los cetáceos que, si bien la ciencia ha definido claramente su existencia, no ha podido aún encontrar sus significados como en la lengua natural hablada de los seres humanos.

En su esencia, entonces, la teoría de números plantea las características y los elementos de juicio del análisis de los números y las relaciones tanto al interior de la matemática como con el entorno en el cual estos sirven de base para interpretar, modelar, analizar, intervenir y optimizar el mundo [7]. En su más simple concepción la teoría de números incluye unos campos plenamente definidos como son: la teoría elemental de los números (que se ocupa de estudiar los números enteros sin que se acuda a técnicas de otros campos de la matemática), la teoría analítica de los números (se basa en el cálculo y en el análisis complejo), la teoría aditiva de los números (se ocupa de los problemas de representación de los números), la teoría algebraica de los números (se ocupa de la expansión del concepto de número al algebra), la teoría geométrica de los números (se ocupa de la relación entre geometría y teoría de números), la teoría combinatoria de los

ISSN: 2248–762X • Vol. 7, No. 2 (julio-diciembre 2016). pp. 116-126

números (se ocupa de estudiar los números desde la óptica de las probabilidades y la combinatoria) y la teoría computacional de los números (se ocupa de estudiar los algoritmos que se consideran relevantes en la teoría de números).

Números primos

En matemáticas, y a la luz de la teoría de números, un número primo es un número, de los llamados naturales, que es mayor que 1 y que tiene solamente dos divisores exactos: el número 1 y el mismo número. De esta forma puede decirse que, teóricamente, los números primos son las antípodas conceptuales de los números compuestos que se definen como aquellos que tienen por lo menos un divisor natural distintos de ellos mismos y del número 1 [8].

Es de anotar que la propiedad de ser un número primo se conoce como primalidad y que en algunos textos la primalidad se refiere a primalidad impar dado que el único número que siendo par es también primo es el número 2. También vale le pena tener en cuenta que el número 1, por definición, no se considera como número primo. El estudio de los números primos es objeto de estudio de la teoría de números [9].

Los números primos parecieran ser números distribuidos aleatoriamente pero su distribución general se ajusta a leyes notoriamente definidas que no se conocen todavía completamente y que, con la ayuda de la tecnología computacional, se puede llegar a aproximaciones que en algún sentido describan dicha distribución.

Densidad de los números primos

En el libro VII de los *Elementos*, Euclides empieza con una relación de veintidós definiciones básicas, entre las cuales se presentan: un número par es aquel que es producto de dos números enteros iguales y un número impar es aquel que no tiene dicha propiedad. De la misma forma, Euclides establece que un número primo es aquel que no tiene más divisores enteros que la unidad y el mismo número. Si tiene divisores mayores que 1 y diferentes de sí mismo, Euclides lo definía como un número compuesto.

Euclides pudo demostrar algunos resultados fundamentales en relación con los números primos [10], como por ejemplo:

- Si un número primo *p* divide el valor resultado de un producto *ab*, entonces *p* divide al menos a uno de los dos números *a* o *b*.
- Cada número natural puede ser primo o puede ser expresado como el producto de los números primos de un modo único independientemente del orden en que dichos factores sean escritos.
- Los números primos son infinitos en cantidad.

El segundo de los planteamientos anteriores es conocido como el teorema fundamental de la aritmética. Los números primos, de acuerdo a lo que dice este planteamiento, son como la célula en los seres vivos tal que todo lo demás está construido a partir de ellos. Los números primos son la base para que todos los demás números naturales existan a partir de procesos multiplicativos. De esta manera cualquier número natural puede ser descompuesto en sus factores primos, es decir, en los números primos que (al multiplicarse) originan el número evaluado. Ahora bien, el tercer planteamiento llega a ser sorprendente para quien haya dedicado algo de su tiempo a escribir los números primos. No se puede desconocer que en la medida en que se avanza en los números naturales, el hallazgo de los números primos se hace cada vez más escaso, pero no resulta totalmente claro que se acaben en un límite determinado. En el rango de 1 a 100, existen 26 números primos sin embargo en el rango de 200000 a 200100 solo existen 9 números primos.

Uno de los métodos que posibilitan una idea precisa de cómo se distribuyen los números primos

ISSN: 2248–762X • Vol. 7, No. 2 (julio-diciembre 2016). pp. 116-126

consiste en prestar atención a lo que se llama la densidad de los números primos, la cual establece la proporción de los números primos en un rango determinado. Si se quiere obtener la densidad de números primos correspondientes a un número límite *N* se debe tomar la cantidad de números primos menores que *N* que se designa $\pi(N)$ y se divide entre *N*.

En el caso de que *N* sea 100, por ejemplo, la respuesta es 0.168 lo que indica que aproximadamente uno de cada seis números menores que 100 es un número primo. Para N=1.000.000 la proporción desciende hasta 0.078, es decir, cerca de uno de cada trece y para N=100.000.000 este factor de densidad es igual a 0.58 (aproximadamente 1 de cada 17). A medida que *N* crece esta reducción continúa. (Keith, 2002). Es de anotar que de que la proporción $\pi(N)/N$ tiende a disminuir, pareciera que los números primos no se extinguen del todo.

El concepto de función

La función es el núcleo lógico a partir del cual se construye un programa bajo la perspectiva "divide y vencerás". La función, en su definición algorítmica, está compuesta por un conjunto de instrucciones que tienen un nombre (en primera instancia único), que puede recibir argumentos (en programación funcional) o parámetros (en programación estructurada) y que pueden retornar o devolver un valor bien sea de manera automática como en la programación funcional o de manera explícita como en la programación estructurada que cristalice su razón de ser dentro de un contexto algorítmico [11].

La función simplifica el proceso de atomización de una solución, es decir, posibilita que un objetivo complejo sea dividido en pequeños y simples objetivos de manera que luego, al articularlos, se puede resolver el problema original a partir del enlace estratégico de las funciones. La función es como un pequeño programa que, a su vez, permite lograr un pequeño objetivo.

La función puede considerarse como la base lógica para abordar el paradigma funcional y capitalizarlo al máximo, para aprovechar las potencialidades de la programación estructurada y para simplificar el mundo a partir de la metodología orientada a objetos dado que la función constituye la base teórica para la construcción de los métodos en este paradigma.

Programación funcional

Se podría decir que la programación funcional es la arista de la programación declarativa que le concede mayor relevancia al concepto de función y que capitaliza sus características para que, en unión de la filosofía "divide and conquer" (divide y vencerás), posibilite la construcción de programas de gran simplicidad, fáciles de administrar en su código y mucho más fáciles de modificar cuando sea el caso [12].

La programación funcional concibe la programación como el diseño apropiado de unas funciones que logran objetivos específicos y que, enlazadas con otras funciones, permiten lograr un objetivo general que es el que justifica la realización del programa. A la luz de la programación funcional, todo gira alrededor del concepto de función y, por tanto, constituye su objeto de estudio más importante.
La programación funcional se basa en los postulados matemáticos del cálculo Lambda (λ calculus) propuesto por el matemático Alonzo Church, profesor de la Universidad de Princeton. Church estaba particularmente interesado en lo que la matemática abstracta proveía y especialmente en su aplicación a partir del poder computacional.

Este matemático se cuestionó qué pasaría si se dispusiera de máquinas con poder ilimitado y qué tipo de problemas se podría solucionar con ellas, en la búsqueda de respuestas a este interrogante. Church, hacia 1936, planteó un lenguaje abstracto al que llamó Cálculo Lambda en el cual usaba como mecanismo de cómputo la evaluación de

ISSN: 2248–762X • Vol. 7, No. 2 (julio-diciembre 2016). pp. 116-126

expresiones. En esos mismos tiempos, Alan Turing, otro matemático desarrolló una máquina abstracta con la cual intentaba resolver los mismos problemas que le preocupaban a Church.

MÉTODOS

Descripción

Para la construcción de este programa se acudió al lenguaje de programación Scheme entorno DrRacket versión 5.1. El algoritmo solución [13], ha sido ejecutado en computadores portátiles convencionales y computadores de escritorio comerciales y los resultados han tenido los mismos resultados tanto en lo numérico como en el rendimiento y en la eficiencia. Para efectos de entender el código que a continuación se presenta (Figura 1) debe anotarse que los signos ";;" indican el inicio de un comentario.

```
;;===================================================
==============
;; D E N S I D A D   D E   L O S   N U M E R O S   P R I M O S
;; CONTEO DE PRIMOS EN RANGOS REGULARES CON UN TOPE
MAXIMO
;; Este programa muestra cuantos números existen en un rango
;; determinado para lo cual se necesita recibir como argumentos
;; el valor inicial, el valor final del primer rango, el
;; tope final de todo el proceso y el tope parcial de cada rango
;;===================================================
==============

;; Función que determina si un valor es múltiplo de otro
;; ****************************************************
(define (divisor a b)       ;; Definición función divisor
(if ( = (remainder a b) 0)  ;; Si un valor divide exactam. al otro
   1                    ;; entonces retorne Verdadero
   0                    ;; sino retorne Falso
))                        ;; Fin condicional - Fin función divisor

;; Función que cuenta la cantidad de divisores de un n
;; (rango 2, sqrt n)
;; ****************************************************
(define (cuentadivisores num div)           ;; Función cuentadivisores
             ;; Si se llegó al tope de evaluación
  (if (>= div (floor ( / (sqrt num) 1)))
    ;; Entonces llamar a función divisor con argumentos
    ;; num y la parte entera de la raiz cuadrada de num
    (divisor num (floor ( / (sqrt num) 1)))
    ;; Sino sumar lo que devuelva la función divisor con
    ;; argumentos num y div con la misma función cuentadivisores
    ;; enviándole num y div+1
   ( + (divisor num div)
      (cuentadivisores num (+ div 1)))
  ))  ;; Fin condicional - Fin función cuentadivisores

;; Función que determina si un valor es primo
;; ******************************************
(define (esprimo n)               ;; Definición función esprimo
;; Si el valor n evaluado no tiene divisores exactos
;; en el rango 2 - raiz cuadrada de n
(if (= (cuentadivisores n 2) 0)
    1                   ;; Entonces retorne un valor Verdadero
    0                   ;; Sino retorne un valor Falso
))                                ;; Fin condicional - Fin función esprimo

;; Función que cuenta los primos de un rango
;; ****************************************
(define (cuentaprimos li ls)      ;; Definición función cuentaprimos
  (if (= li ls)             ;; si el líminf es igual al límsup
     (esprimo li)           ;; entonces evaluar si el liminf es primo
 ;; sino sumar lo que devuelva la función esprimo con argum limsup
 ;; mas lo que retorne la misma función cuentaprimos
  (+ (esprimo ls)
    (cuentaprimos li (- ls 1)))))

;; Contador de primos con limites definidos
;; ****************************************
(define (cp li ls)                ;; definición función cp
  (if (= li 1)                    ;; Si el lím inf es igual a 1
;; entonces sume 2 mas lo que retorne la función
;; cuentaprimos con argumentos 2 y lim sup
  (+ 2 (cuentaprimos 2 ls))
;; Sino llame a la función cuentaprimos con
;; argumentos lim inf y lim sup
  (cuentaprimos li ls)
  ))                              ;; Fin condicional - Fin función cp
```

```
;; Tabla de densidad de primos (cada n numeros)
;; ********************************************
(define (fila li ls np)  ;; Definición función fila
;;(newline)
  (display li)          ;; Muestre el lim inf
  (display ", ")        ;; Muestre separador
  (display ls)          ;; Muestre el lím sup
  (display ", ")        ;; Muestre separador
  (display np)          ;; Muestre número de primos en el rango
  (display ", ")        ;; Muestre separados
  ;; Almacene el factor de densidad de los números primos
  (set! np (/ np (- ls li)))
  ;; Muestre separador
  ;; Muestre el factor densidad de primos
  (display (* np 1.0))          ;; Muestre el valor en formato real
  (newline)             ;; Pase a la siguente línea
  0                     ;; Retorno Falso
  )                             ;; Fin Función fila

;; Función que pone un título
;; *************************
(define (titulo)        ;; Definición función titulo
  ;; Despliegue de títulos
  (display "(LimInf, LimSup, NumPrim, Densidad)")
  (newline)                     ;; Pase a la siguiente línea
  )                     ;; Fin función titulo

;; Función que selecciona los datos de cada fila
;; ********************************************
(define (proceso li ls tope paso)          ;; Definición función
proceso
  (if (= li tope)               ;; Si líminf es igual a tope
    0                   ;; entonces retorne valor Falso
;; Sino suma lo que retorne la función fila
;; mas lo que retorne la función proceso
  (+ (fila li (+ li paso) (cp li (+ li paso)))
    (proceso ls (+ ls paso) tope paso))
  ))                            ;; Fin condicional - Fin función proceso

;; Función que genera la tabla
;; **************************
(define (tabla li ls tope paso)   ;; Fin función tabla
  (newline)                     ;; Pase a la siguiente línea
  (titulo)                              ;; Mostrar título
;; Iniciar proceso con argumentos lím inf, lím sup, tope y paso
;; li (límite inferior), ls (límite superior), tope (valor hasta
;; donde se llegará con la evaluación), paso (avance para evaluación)
(proceso li ls tope paso)
)                                       ;; Fin función tabla
```

Figura 1. Densidad de los números primos, conteo de primos en rangos regulares con un tope máximo.

Aplicación

El esquema funcional del programa presentado se muestra en la Figura 2. Como se puede notar, el esquema funcional diagrama 1 presenta la relación entre las funciones y, de esa forma, tener un panorama general claro del programa en su conjunto. La función inicial es la función tabla, esta llama a la función título y luego llama a la función recursiva proceso que, a su vez, llama a la función fila.

La función proceso llama a la función *cp* que a su vez llama a la función recursiva cuentaprimos, esta llama a la función esprimo quien a su vez llama a la función recursiva cuentadivisores la cual, por último, llama a la función divisor. Es claro que la gran característica de este programa es que se acude a la recursividad nivel III para facilitar su concepción algorítmica. La recursividad nivel III es la que se presenta cuando una función recursiva acude a una o más funciones recursivas para lograr su objetivo.

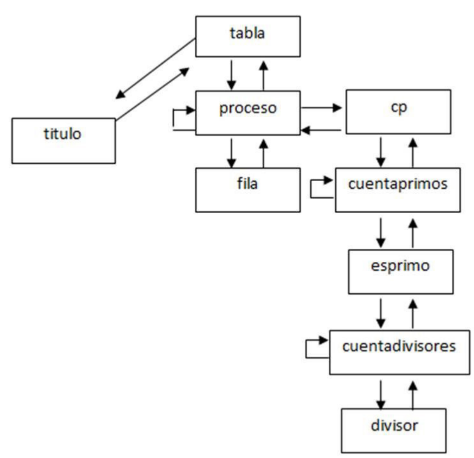

Figura 2. Esquema Funcional.

ISSN: 2248–762X • Vol. 7, No. 2 (julio-diciembre 2016). pp. 116-126

RESULTADOS

Cuando se ejecuta este programa se obtienen los resultados que indica la teoría alrededor del factor de densidad de números primos el cual, dicho sea de paso, es constante dentro de un mismo rango. Tal como lo indica el fundamento teórico de la densidad de los números primos los rangos a evaluar son (1, 10), (1, 100), (1, 1000), (1, 10000) y el rango (1, 100000). La ejecución del programa en entorno DrRacket genera los resultados de la tabla 1.

Tabla 1.

```
> (tabla 1 10 10 10)
(LimInf, LimSup, NumPrim, Densidad)
1, 11, 5, 0.5
0
> (tabla 1 100 100 100)
(LimInf, LimSup, NumPrim, Densidad)
1, 101, 26, 0.26
0
> (tabla 1 1000 1000 1000)
(LimInf, LimSup, NumPrim, Densidad)
1, 1001, 168, 0.168
0
> (tabla 1 10000 10000 10000)
(LimInf, LimSup, NumPrim, Densidad)
1, 10001, 1229, 0.1229
0
> (tabla 1 100000 100000 100000)
(LimInf, LimSup, NumPrim, Densidad)
1, 100001, 9592, 0.09592
0
```

Este mismo programa posibilita usarlo de otra forma. Supongamos que quisiéramos analizar la densidad de los números primos por rangos, por ejemplo, que se quisiera saber cuántos números primos hay entre 1000 y 10000 analizando por rangos temporales de 1000 números, es decir, evaluar la densidad entre 1000 y 2000, entre 2000 y 3000, entre 3000 y 4000 y así sucesivamente hasta llegar al rango comprendido entre 9000 y 10000. Su ejecución genera los resultados de la tabla 2.

Tabla 2.

```
> (tabla 1000 2000 10000 1000)
(LimInf, LimSup, NumPrim, Densidad)
1000, 2000, 135, 0.135
2000, 3000, 127, 0.127
3000, 4000, 120, 0.12
4000, 5000, 119, 0.119
5000, 6000, 114, 0.114
6000, 7000, 117, 0.117
7000, 8000, 107, 0.107
8000, 9000, 110, 0.11
9000, 10000, 112, 0.112
0
```

Ahora bien, si se quisiera evaluar la densidad en el rango de 100000 a 500000 por rangos de 100000, los resultados se muestran en la tabla 3.

Tabla 3.

```
> (tabla 100000 200000 500000 100000)
(LimInf, LimSup, NumPrim, Densidad)
100000, 200000, 8392, 0.08392
200000, 300000, 8013, 0.08013
300000, 400000, 7863, 0.07863
400000, 500000, 7678, 0.07678
```

DISCUSIÓN

Primeramente, se puede notar que la recursividad posibilita la construcción de algoritmos sencillos, fáciles de concebir y muy fáciles de entender para abordar problemas que, como este, son tradicionales en las matemáticas. De una parte, la tecnología computacional posibilita encontrar resultados en rangos mucho más amplios que los que convencionalmente se han encontrado. De otra parte, el paradigma funcional permite que dichos algoritmos se puedan basar en el concepto de función que, debidamente enlazado tal como se observa en la figura 1, simplifica las pruebas parciales a las funciones, simplifica el objetivo y facilita la reutilización de funciones ya construidas.

Llama la atención de que, tal como lo tenía previsto Euclides, en la medida en que se amplía el rango de evaluación para conocer la densidad de los números primos o en la medida en que se analiza dicha densidad por rangos continuos, pareciera que la densidad tendiera a disminuir. Si bien Euclides, Riemann, Fermat y otros matemáticos encontraron demostraciones para indicar que el conjunto de números primos es infinito y que la descomposición de los números enteros en sus factores primos así lo demuestra, queda la inquietud acerca de la búsqueda de números primos de orden superior (aquellos que tienen más de 1000 dígitos) y la densidad asociada dentro de su rango respectivo de ubicación dentro de los números naturales.

La evaluación de la densidad de los números primos en rangos contiguos significativos (que tengan 1000 o más de 1000 números naturales) indica que en la medida en que se avanza, se hace más escaso el hallazgo de dichos números primos. ¿Sería posible que se encuentre un rango a partir del cual ya no existan más números primos? La matemática a través de la descomposición de números primos dice que no, sin embargo, la tecnología computacional podría decir algo diferente en la medida en que algoritmos como el que se presenta en este artículo sean ejecutados en computadores de alto rendimiento sobre rangos de 1000000 dígitos o más.

CONCLUSIONES

Es posible encontrar soluciones a problemas que tradicionalmente la matemática ha planteado a partir de las tecnologías computacionales y de la algoritmia moderna. La programación funcional posibilita caminos para que estudiantes de los primeros semestres de Ingeniería de Sistemas puedan encontrar soluciones óptimas, fáciles de concebir y fáciles de mantener en relación con su concepción algorítmica. Muchas teorías que hasta el momento se han demostrado por mecanismos y formulaciones matemáticas, en tiempos modernos se pueden confirmar, rebatir, reforzar o controvertir a través de la tecnología computacional que la electrónica provee. Una de las aristas de utilización de la computación de alto rendimiento podría ser la ejecución de algoritmos que vayan en el sentido de las demostraciones que las matemáticas han presentado hasta el momento.

Las inferencias alrededor de la densidad de los números primos pueden llegar a ser controvertidas a través de los resultados que la tecnología computacional provee. Es posible encontrarle significado a problemas matemáticos a través de la utilización de la tecnología computacional y de la programación de computadores como punto de encuentro entre las ciencias de la computación y la misma matemática. Es posible aplicar los fundamentos del aprendizaje significativo para que los estudiantes de ingeniería se motiven, adquieran nuevos conocimientos y los relacionen con conocimientos previos de forma que asimilen, apropien y apliquen la programación de computadores en la solución de problemas de las matemáticas. Es posible encontrar caminos que le permitan al estudiante descubrir nuevas soluciones y nuevas verificaciones a soluciones existentes a problemas matemáticos a partir de la tecnología computacional moderna aplicando los fundamentos del aprendizaje por descubrimiento. Todo planteamiento matemático puede tener un equivalente algorítmico que puede ser revalidado con las tecnologías computacionales modernas.

REFERENCIAS

[1] P. Ausubel, *Sicología Educativa: Un punto de vista cognoscitivo*. México: Trillas. 1986.

[2] J. Medina, *Los 12 principios del cerebro*. Bogotá, D.C.,: Grupo Editorial Norma, 2010.

[3] J. Bruner, *Hacia una teoría de la instrucción*. Manuales Uteha, No. 373, México: Editorial Hispanoamericana, 1969.

[4] J. Bruner, *Actos de significado, Más allá de la revolución cognitiva*, Madrid: Alianza Editorial, 1991.

[5] G. Small, *El cerebro digital*, Barcelona: Editorial Urano, 2010.

[6] W. Mora, *Introducción a la teoría de números*. San José, Costa Rica: Escuela de Matemática. Instituto Tecnológico de Costa Rica, 2010.

[7] C. Ivorra, *Teoría de Números*, Valencia, España: Universidad Politécnica de Valencia, Editorial Sanz y Torres, 2010.

[8] R. Crandall, C. and Pomerance, *Prime Numbers, a computational perspective*. 2nd Ed. NY, USA: Springer Science+Bussiness Media, 2005.

[9] P. Montgomery, "A Survey of Modern Integer Factorization Algorithms". *Quaterly Journal*, vol. 7(4), pp. 337. 1994.

[10] K. Devlin, *El lenguaje de las matemáticas*. Barcelona. España: MaNon Troppo, Ediciones Robinson, 2002.

[11] O. Trejos, *La esencia de la lógica de programación*. Manizales: Centro Editorial Universidad de Caldas, 2000.

[12] P. Van Roy, *Concepts, techniques and models of Computer Programming*. Switzerland: Universidad Católica de Lovaine. Swedish Institute of Computer Science, 2003.

[13] O. Trejos, "Algoritmo de optimización para la detección de un número primo basado en programación funcional utilizando DrScheme", *Scientia et Technica*, vol. 17 (47), abril, 2011.