INFORME DE INGENIERIA

LABORATORIO UNIDAD 1

PRIMES NUMBERS GENERATOR

Presentado por:

Marisol Giraldo Cobo

Presentado al Profesor:

Andrés Aristizábal

Universidad ICESI

Santiago de Cali, febrero 16 del 2020

**INFORME DE INGENIERÍA**

**CONTEXTO DEL PROBLEMA**

En la actualidad la criptografía es una herramienta utilizada para definir algoritmos de encriptación que mantengan la triada CID: Confidencialidad, Integridad y Disponibilidad de los datos. Para garantizar lo anterior, se propone acudir a la matemática aprovechando las propiedades de los números primos, de tal manera que los algoritmos de encriptación utilicen esta tarea básica.

**PASO 1: IDENTIFICACIÓN DEL PROBLEMA**

**IDENTIFICACIÓN DE NECESIDADES Y SINTOMAS:**

* Ingresar un número máximo o tope (n) para la búsqueda de los números primos.
* Seleccionar que algoritmo se quiere utilizar para la generación de los números primos.
* Encontrar los números primeros menores que un número dado.
* Generar como salida una matriz (de 1 hasta n)
* Pintar de color verde los números primos y de rojo los que no son números primos
* Mostrar en tiempo real el proceso que se utiliza para encontrar los números primos

**DEFINICIÓN DEL PROBLEMA**

Se requiere una aplicación grafica que permita encontrar los números primeros menores o iguales a un n dado, mediante la implementación de tres algoritmos.

**PASO 2: RECOPILACIÓN DE INFORMACIÓN**

***Algoritmo***

Es una secuencia de pasos bien definidos que buscan resolver un problema computacional.

***Algoritmo Iterativo***

Son algoritmos que se caracterizan por ejecutarse mediante ciclos, es decir, aquellos que llegan a un resultado a través de una iteración mediante un ciclo definido o indefinido.

***Eficiencia***

Medida del uso de los recursos computacionales requeridos por la ejecución de un algoritmo en función del tamaño de las entradas.

***Complejidad temporal***

Función que describe el comportamiento (en tiempo) de un algoritmo conforme se incrementa el tamaño de la entrada.

***Complejidad espacial***

Uso de recursos de espacio que requiere un algoritmo en cuestión.

***Notación asintótica***

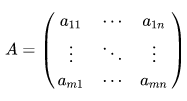
Son aquellas notaciones utilizadas para describir el tiempo de ejecución asintótico de un algoritmo. Se conoce la notación O para el peor caso, Ω para el mejor caso y θ para el caso promedio.

|  |  |
| --- | --- |
| **Complejidad** | **Terminología** |
| O(l) | Complejidad constante |
| O(log n) | Complejidad logarítmica |
| O(n) | Complejidad lineal |
| O(n log n) | Complejidad n log n |
| O() | Complejidad polinómica |
| O() | Complejidad exponencial |
| O(n!) | Complejidad factorial |

***Matriz***

Se puede definir una matriz, como un conjunto de elementos (números) ordenados en filas y columnas. Para designar una matriz se emplean letras mayúsculas. Cada uno de los elementos de la matriz (**aij)** tiene dos subíndices. El primero **i**indica la fila a la que pertenece y el segundo**j** la columna.

Esta es una matriz de **m** filas y ***n*** columnas, es decir, de **dimensión** ***m x n***.  Esta matriz también se puede representar de la forma siguiente:  **A = (aij)m x n.** Si el número de filas y de columnas es igual (**m = n**), entonces se dice que la matriz es de **orden n**.

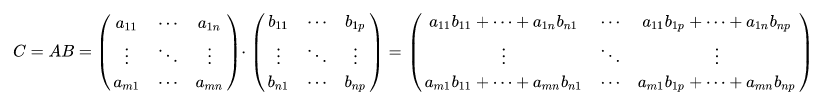


Sean *A*, *B* matrices y k escalar, la multiplicación de matrices se da de la siguiente manera:

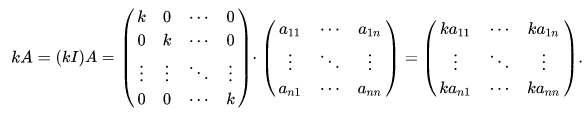
***Multiplicación de Matrices***

Dos matrices A y B son multiplicables si el **número de columnas de A** coincide con el **número de filas de B**. **Mm x n x Mn x p= M m x p**

El elemento**cij** de la matriz producto se obtiene **multiplicando** cada elemento de la **fila i**de la matriz A por cada elemento de la **columna j**de la matriz B y **sumándolos**.

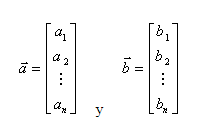


***Multiplicación de Matrices por un Escalar***



***Multiplicación Producto Punto***

Sea:



dos vectores del mismo número de componentes.

 El producto escalar de  http://oceanologia.ens.uabc.mx/~matematicas/algebralineal/II%20Matrices/promat_files/image006.gif  y  http://oceanologia.ens.uabc.mx/~matematicas/algebralineal/II%20Matrices/promat_files/image008.gif , denotado  http://oceanologia.ens.uabc.mx/~matematicas/algebralineal/II%20Matrices/promat_files/image010.gif , se define por:



***Modelo RAM***

Es el modelo que se usa de manera implícita en la práctica para evaluar la complejidad de los algoritmos.

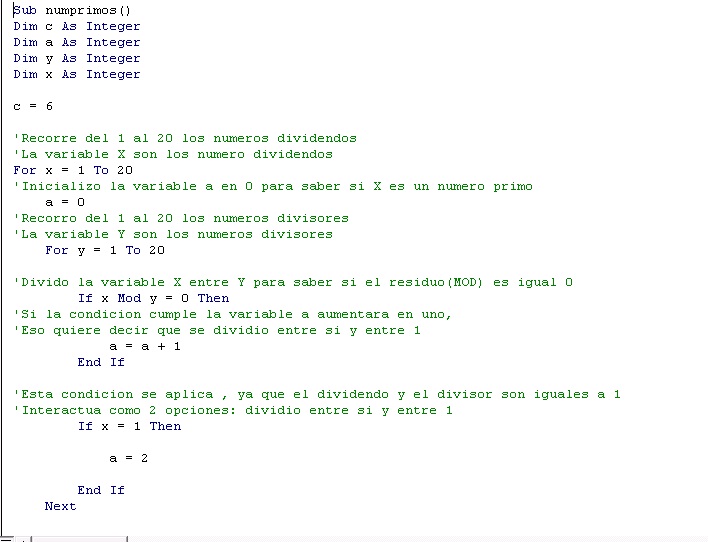
***Numero Primo***

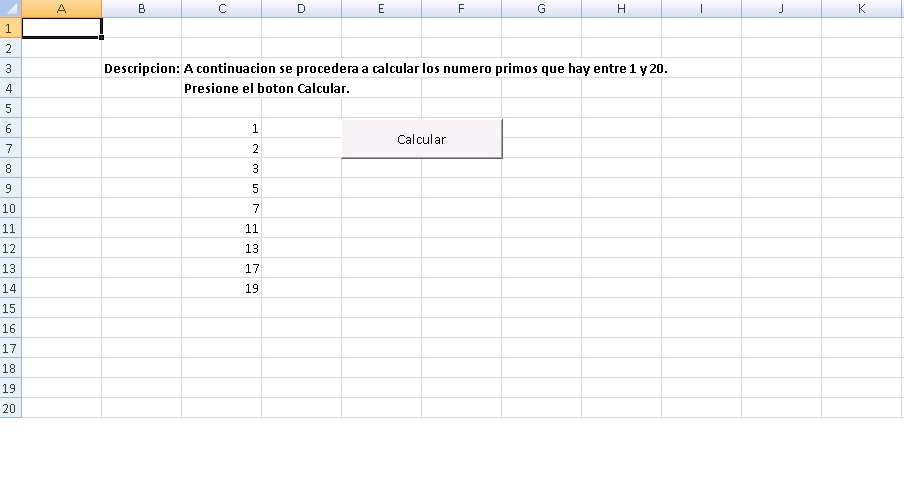
La definición correcta es la de un número natural que solo **puede dividirse por uno y por sí mismo.**

**PASO 3. BÚSQUEDA DE SOLUCIONES CREATIVAS**

**ALTERNATIVA 1: GENERAR NUMERO PRIMOS A TRAVES DE UNA MACRO EN EXCEL**

Excel incluye funciones de hoja de cálculo integradas, las cuales son posibles de crear de manera personalizada con el fin de satisfacer las necesidades de cálculo de cada usuario.





**ALTERNATIVA 2: GENERAR LOS NUMEROS PRIMOS MEDIANTE SU DEFINICIÒN FORMAL**

* Si un número NO es divisible por 2, entonces no es divisible por ninguno de sus múltiplos (por ningún número par).
* Si un número NO es divisible por 3, entonces NO es divisible por ninguno de sus múltiplos.
* No es necesario comprobar para 4 pues 4 es múltiplo de 2.
* Si un número NO es divisible por 5, entonces NO es divisible por ninguno de sus múltiplos.
* Siguiendo esta idea que es válida; de acuerdo con el estudio de la Teoría de Números, específicamente el álgebra modular, para comprobar que un número es primo BASTA con comprobar que no es divisible por ningún primo menor que ese número entre 2; pues un número sólo es candidato a ser divisible por otro si ese otro es menor que su mitad. Para cualquier “otro”, no tiene sentido comprobar divisibilidad porque no lo será. Excepto claro, si ese número mayor que la mitad es igual al que se quiere comprobar divisibilidad: Pues todo número se divide a sí mismo.

**ALTERNATIVA 3: SIEVE OF ERATOSTHENES**

Es una de las formas más eficientes de encontrar todos los números primos más pequeños que n cuando n es más pequeño que 10 millones.

El siguiente es el algoritmo para encontrar todos los números primos menores o iguales a un entero n dado por el método de Eratóstenes:

1. Cree una lista de enteros consecutivos de 2 a n: (2, 3, 4, ..., n).
2. Inicialmente, sea p igual a 2, el primer número primo.
3. A partir de p2, cuente en incrementos de p y marque cada uno de estos números mayor o igual que p2 en la lista. Estos números serán p (p + 1), p (p + 2), p (p + 3), etc.
4. Encuentre el primer número mayor que p en la lista que no está marcado. Si no hubo tal número, deténgase. De lo contrario, deje que p ahora sea igual a este número (que es el próximo primo) y repita desde el paso 3.

Cuando el algoritmo termina, todos los números en la lista que no están marcados son primos.

**PSEUDOCODIGO**

**Entrada:** un numero natural n

**Salida:** El conjunto de números primos anteriores a n (incluyendo n)

1. Escriba todos los números naturales desde 2 hasta n
2. **Para i** desde 2 hasta **haga lo siguiente**:

**Si** i no ha sido marcado

**entonces:**

**Para** j desde i hasta n / i **haga lo siguiente:**

Ponga una marca en i x j

1. **El resultado es**: Todos los números sin marca

**ALTERNATIVA** **4:** **SIEVE OF ATKIN**

Es un algoritmo moderno para encontrar todos los números primos hasta un número entero especificado. En comparación con el algoritmo Sieve of Eratóstenes, que marca múltiplos de números primos, realiza un trabajo preliminar y luego marca múltiplos de cuadrados de números primos.

**PSEUDOCODIGO**

// límite arbitrario de búsqueda

límite ← 1000000

// inicializar la criba

es\_primo(i) ← falso, i ∈ [5, límite]

// introducir candidatos a números primos:

// números naturales con un número impar de

// representaciones en determinadas formas cuadráticas

para (x, y) en [1, √límite] × [1, √límite]:

n ← 4x²+y²

si (n ≤ límite) ∧ (n mod 12 = 1 ∨ n mod 12 = 5):

es\_primo(n) ← ¬es\_primo(n)

n ← 3x²+y²

si (n ≤ límite) ∧ (n mod 12 = 7):

es\_primo(n) ← ¬es\_primo(n)

n ← 3x²-y²

si (x > y) ∧ (n ≤ límite) ∧ (n mod 12 = 11):

es\_primo(n) ← ¬es\_primo(n)

// eliminar compuestos mediante la criba

para n en [5, √límite]:

si es\_primo(n):

// n es primo, se omiten los múltiplos de su cuadrado, esto es

// suficiente porque los números compuestos que están

// en esta lista no pueden ser [[número libre de cuadrados|libres de cuadrados]]

es\_primo(k) ← falso, k ∈ {n², 2n², 3n², ..., limit}

escribir 2, 3

para n en [5, límite]:

si es\_prime(n): escribir n

**PASO 4: TRANSICIÓN DE LA FORMULACIÓN DE IDEAS A LOS DISEÑOS PRELIMINARES**

**ALTERNATIVA 1: GENERAR NUMERO PRIMOS A TRAVES DE UNA MACRO EN EXCEL**

Cuando se ejecuta una función o subrutina que requiere una modificación de los datos, una importación masiva… o un esfuerzo muy grande por parte de Excel, si la máquina no es muy potente, puede que no carque, que se bloquee o que incluso no llegue a procesar toda la información. Esto se debe a **problemas de memoria o lentitud en el refresco de datos cuando se ejecuta una macro compleja.**

**ALTERNATIVA 2: GENERAR LOS NUMEROS PRIMOS MEDIANTE SU DEFINICIÒN FORMAL**

Este algoritmo es de un gran costo computacional, sobre todo si se quiere hacer sobre números muy grandes.

**ALTERNATIVA 3: SIEVE OF ERATOSTHENES**

Este algoritmo es de un menor costo computacional. La complejidad del tiempo de ejecución de Sieve of Eratosthenes es O (n \* log (log (n))).

**ALTERNATIVA** **4:** **SIEVE OF ATKIN**

Tiene una mejor complejidad asintótica teórica con Complejidad de (N / (log log N))

**PASO 5: EVALUACIÓN Y SELECCIÓN DE LA MEJOR SOLUCIÓN**

**CRITERIO 1: MEDICIÓN**

**[1]:** Precisa

**[2]:** Aproximada

**[3]:** Inexacta

**CRITERIO 2: COMPLEJIDAD**

**[1]:** Constante

**[2]:** Logarítmica

**[3]:** Lineal

**[4]:** n log n

**[5]:** Polinómica

**[6]:** Exponencial

**[7]:** Factorial

**[8]:** n/p

**CRITERIO 3: COMPLETITUD**

**[1]:** Todas las soluciones.

**[2]:** Algunas soluciones.

**[3]:** 1 o ninguna solución.

**CRITERIO 4: FACILIDAD EN IMPLEMENTACIÒN ALGORITMICA**

**[1]:** Compatible con las operaciones aritméticas básicas de un equipo moderno.

**[2]:** No compatible completamente con las operaciones básicas de un equipo de cómputo moderno.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Criterio 1** | **Criterio 2** | **Criterio 3** | **Criterio 4** | **Total** |
| **Alternativa 1** |  |  |  |  |  |
| **Alternativa 2** |  |  |  |  |  |
| **Alternativa 3** |  |  |  |  |  |
| **Alternativa 4** |  |  |  |  |  |

De acuerdo con la tabla anterior, la mejor solución es la **ALTERNATIVA** ,

**PASO 6: PREPARACIÓN DE INFORMES Y ESPECIFICACIONES**

|  |  |
| --- | --- |
|  | Laboratorio I  Algoritmos y Estructuras de Datos  Requerimientos Funcionales |
| Nombre del Proyecto: | Primes Numbers |
| Presentado por: | Marisol Giraldo Cobo – Código: A00246380 |
| Presentado a: | Andrés Aristizábal |
| Fecha: | Febrero 16 del 2020 |

|  |  |
| --- | --- |
| Nombre | R1 – Generar una interfaz de usuario |
| Resumen | Permite mediante un entorno de usuario ingresar un numero n entero como tope y generar a partir de uno de los algoritmos escogidos los números primos detrás de ese n incluyéndose. |
| Entradas | |
| - n: entero | |
| Resultados | |
| Una matriz de dimensiones rows\*cols aproximadamente cuadrada con los números primos pintados en color ROJO. | |

|  |  |
| --- | --- |
| Nombre | R2 – Generar números primos por divisibilidad |
| Resumen | Generar los números primos a partir de un n dado aplicando el primer algoritmo que utiliza la definición de divisibilidad de numero primo. |
| Entradas | |
| - n: entero | |
| Resultados | |
| Una matriz de dimensiones rows\*cols aproximadamente cuadrada con los números primos pintados en color ROJO y calculados por el primer algoritmo que tiene en cuenta la generación de primos a partir de la definición de divisibilidad. | |

|  |  |
| --- | --- |
| Nombre | **R3 – Generar números primos por Sieve Of Eratosthenes** |
| Resumen | Generar los números primos a partir de un n dado aplicando el segundo algoritmo que utiliza la definición de Sieve Of Eratosthenes |
| Entradas | |
| - n: entero | |
| Resultados | |
| Una matriz de dimensiones rows\*cols aproximadamente cuadrada con los números primos pintados en color ROJO y calculados por el segundo algoritmo que tiene en cuenta la generación de primos a partir de Sieve Of Eratosthenes | |
| Nombre | **R3 – Generar números primos por Sieve Of ATKIN** |
| Resumen | Generar los números primos a partir de un n dado aplicando el tercer algoritmo que utiliza la definición de Sieve Of Atkin |
| Entradas | |
| - n: entero | |
| Resultados | |
| Una matriz de dimensiones rows\*cols aproximadamente cuadrada con los números primos pintados en color ROJO y calculados por el tercer algoritmo que tiene en cuenta la generación de primos a partir de la definición de Sieve Of Atkin | |

|  |  |
| --- | --- |
|  | Proyecto Final  Algoritmos y Estructuras de Datos  Pruebas Unitarias Automáticas |
| Nombre del Proyecto: | Primes Numbers |
| Presentado por: | Marisol Giraldo Cobo – Código: A00246380 |
| Presentado a: | Andrés Aristizábal |
| Fecha: | Febrero 16 del 2020 |

**DISEÑO CASOS DE PRUEBA**

|  |  |  |  |
| --- | --- | --- | --- |
| **PRUEBA 1** | | | |
| **Objetivo:** Generar los números primos para un n=17 | | | |
| **Clase** | **Método** | **Entrada** | **Resultados** |
| PrimeGenerator | firstAlgorithm() | 1, 2, 3, 4, 5, 6,7, 8, 9, 10, 11, 12,  13, 14, 15, 16, 17 | 2, 3, 5, 7, 11, 13, 17 |

|  |  |  |  |
| --- | --- | --- | --- |
| **PRUEBA 2** | | | |
| **Objetivo:** Generar los números primos para un n=22 | | | |
| **Clase** | **Método** | **Entrada** | **Resultados** |
| PrimeGenerator | firstAlgorithm() | 1, 2, 3, 4, 5, 6,7, 8, 9, 10, 11, 12,  13, 14, 15, 16, 17,18,19,20,21,22 | 2, 3, 5, 7, 11, 13, 17, 19. |

|  |  |  |  |
| --- | --- | --- | --- |
| **PRUEBA 3** | | | |
| **Objetivo:** Generar los números primos para un n=150 | | | |
| **Clase** | **Método** | **Entrada** | **Resultados** |
| PrimeGenerator | firstAlgorithm() | 1, 2, 3, 4, 5, 6,7, 8, 9, 10, 11, 12,  13, 14, 15, 16, 17, 18, ….. 150 | 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. |

|  |  |  |  |
| --- | --- | --- | --- |
| **PRUEBA 1** | | | |
| **Objetivo:** Generar los números primos para un n=17 | | | |
| **Clase** | **Método** | **Entrada** | **Resultados** |
| PrimeGenerator | secondAlgorithm() | 1, 2, 3, 4, 5, 6,7, 8, 9, 10, 11, 12,  13, 14, 15, 16, 17 | 2, 3, 5, 7, 11, 13, 17 |

|  |  |  |  |
| --- | --- | --- | --- |
| **PRUEBA 2** | | | |
| **Objetivo:** Generar los números primos para un n=22 | | | |
| **Clase** | **Método** | **Entrada** | **Resultados** |
| PrimeGenerator | secondAlgorithm() | 1, 2, 3, 4, 5, 6,7, 8, 9, 10, 11, 12,  13, 14, 15, 16, 17,18,19,20,21,22 | 2, 3, 5, 7, 11, 13, 17, 19. |

|  |  |  |  |
| --- | --- | --- | --- |
| **PRUEBA 3** | | | |
| **Objetivo:** Generar los números primos para un n=150 | | | |
| **Clase** | **Método** | **Entrada** | **Resultados** |
| PrimeGenerator | secondAlgorithm() | 1, 2, 3, 4, 5, 6,7, 8, 9, 10, 11, 12,  13, 14, 15, 16, 17, 18, ….. 150 | 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. |

|  |  |  |  |
| --- | --- | --- | --- |
| **PRUEBA 1** | | | |
| **Objetivo:** Generar los números primos para un n=17 | | | |
| **Clase** | **Método** | **Entrada** | **Resultados** |
| PrimeGenerator | thirdAlgorithm() | 1, 2, 3, 4, 5, 6,7, 8, 9, 10, 11, 12,  13, 14, 15, 16, 17 | 2, 3, 5, 7, 11, 13, 17 |

|  |  |  |  |
| --- | --- | --- | --- |
| **PRUEBA 2** | | | |
| **Objetivo:** Generar los números primos para un n=22 | | | |
| **Clase** | **Método** | **Entrada** | **Resultados** |
| PrimeGenerator | thirdAlgorithm() | 1, 2, 3, 4, 5, 6,7, 8, 9, 10, 11, 12,  13, 14, 15, 16, 17,18,19,20,21,22 | 2, 3, 5, 7, 11, 13, 17, 19. |

|  |  |  |  |
| --- | --- | --- | --- |
| **PRUEBA 3** | | | |
| **Objetivo:** Generar los números primos para un n=150 | | | |
| **Clase** | **Método** | **Entrada** | **Resultados** |
| PrimeGenerator | thirdAlgorithm() | 1, 2, 3, 4, 5, 6,7, 8, 9, 10, 11, 12,  13, 14, 15, 16, 17, 18, ….. 150 | 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. |

**PASO 7: IMPLEMENTACIÓN DEL DISEÑO**

**A screenshot of a cell phone

Description automatically generated**

**BIBLIOGRAFIA**

**RESOLUCIÓN DE PROBLEMAS**

<https://www.cs.upc.edu/~bejar/ia/transpas/teoria-n/2-BH1-introduccion_busqueda-n.pdf>

**METODO DE INGENIERIA**

\*Resumen del capítulo 5 del libro Introduction to Engineering. Paul H. Wright. 3rd ed. Editorial

John Wiley & Sons, Inc. 2002.

**NUMEROS PRIMOS EN JAVA**

<http://lineadecodigo.com/java/numeros-primos-en-java/>

**SIEVE OF ERATOSTHENES**

<https://www.khanacademy.org/computing/computer-science/cryptography/comp-number-theory/v/sieve-of-eratosthenes-prime-adventure-part-4>

**SIEVE OF ERATOSTHENES**

<https://www.geeksforgeeks.org/sieve-of-eratosthenes/>

[**HOW DO I REDUCE THE SPACE COMPLEXITY IN SIEVE OF ERATOSTHENES TO GENERATE PRIME BETWEEN A AND B?**](https://stackoverflow.com/questions/12430495/how-do-i-reduce-the-space-complexity-in-sieve-of-eratosthenes-to-generate-prime)

<https://stackoverflow.com/questions/12430495/how-do-i-reduce-the-space-complexity-in-sieve-of-eratosthenes-to-generate-prime>

**PRIME NUMBERS - SIEVE OF ERATOSTHENES**

<https://www.youtube.com/watch?v=V08g_lkKj6Q>

**SIEVE OF ATKIN**

<https://www.geeksforgeeks.org/sieve-of-atkin/>

**SIEVE OF ATKIN – EXPLANATION AND JAVA EXAMPLE**

<https://stackoverflow.com/questions/10580159/sieve-of-atkin-explanation-and-java-example>