Instituto Politecnico Nacional

ESCOM “ESCUELA SUPERIOR DE CÓMPUTO”

TEORÍA COMPUTACIONAL

PROYECTO: MÁQUINA DE TURING (CONVERTIDOR DECIMAL-BINARIO)

PROFA: Luz María Sánchez García

ALUMMNOS: Rojas Alvarado Luis Enrique

Rodriguez Hernández Aldo Hassan

Trejo Rivera Oscar Gerardo

GRUPO: 2CM11

Introducción:

El proyecto que desarrollaremos realizara una simulación de la máquina de Turing programada en C con el objetivo de aplicarla a un convertidor de decimal a binario.

Desarrollo:

La máquina de Turing es un dispositivo que manipula símbolos sobre una tira de cinta de acuerdo a una tabla de reglas. A pesar de su simplicidad, una máquina de Turing puede ser adaptada para simular la lógica de cualquier algoritmo de computador y es particularmente útil en la explicación de las funciones de una CPU dentro de un computador.

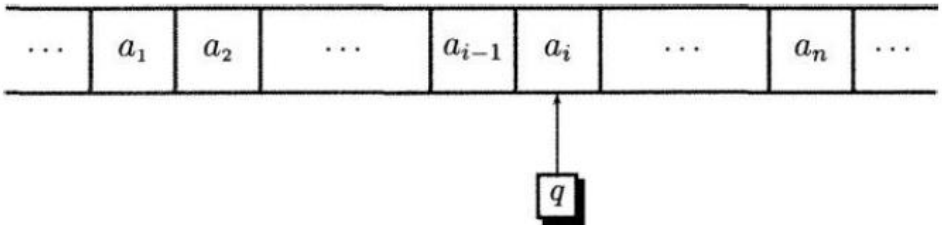
Originalmente fue definida por el matemático inglés Alan Turing como una «máquina automática» en 1936, en la revista Proceedings of the London Mathematical Society, La máquina de Turing no está diseñada como una tecnología de computación práctica, sino como un dispositivo hipotético que representa una máquina de computación. Las máquinas de Turing ayudan a los científicos a entender los límites del cálculo mecánico.

Una máquina de Turing MT= es una séptupla $MT = (Q, q0, F, \Sigma, \Gamma, ă, \delta)$, donde:

- Q es el conjunto finito de estados internos.
- $q0 \in Q$ es el estado inicial.
- $F\ f = \varphi$ es el conjunto finito de estados de aceptación, donde $F \subseteq Q$. Σ es el alfabeto de entrada.
- Γ es el alfabeto de cinta Tal que $\Sigma \subseteq \Gamma$.
- $ă \in \Gamma$ es el símbolo *blanco* tal que $ă\ f \in \Sigma$.
- δ es la función de transición, donde $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{\leftarrow, -, \rightarrow\}$, es decir, recibe un estado y un símbolo de la cinta para devolver otro estado, otro símbolo y una dirección de movimiento.
- La MT procesará cadenas $w \in \Sigma*$, donde w se colocará en la cinta al principio del cómputo. La cinta es en esencia infinita en ambas direcciones. La MT comenzará con el primer símbolo de w estando en el estado q0. Las demás casillas de la cinta contienen el símbolo blanco ă.

Descripcion

Es una expresión de la forma a1a2 • • • ai−1qai • • • an, donde a1, . . . , an ∈ Γ y q ∈ Q. Significa que estamos en el estado q escaneando el símbolo ai. Se supone que las casillas a la izquierda de a1 y a la derecha de an contienen el símbolo blanco ă.



La descripción inicial es *q*0*w*, donde *w* es la cadena de entrada. *w* puede ser colocada en cualquier parte de la cinta, pues esta es infinita.

Definimos a un **paso computacional** como el paso de una descripción instantánea a otra por medio de una transición definida por *δ*. Se denota como *u*1*qu*2 € *v*1*pv*2, donde *u*1, *u*2, *v*1, *v*2 ∈ Γ∗ y *p, q* ∈ *Q*. Esto es equivalente a decir que existe una transición *δ*(*q, a*) = (*p, b, d*).

Por último, la notación *u*1*qu*2 €∗ *v*1*pv*2 significa que *M* puede pasar de la descripción instantánea *u*1*qu*2 a *v*1*pv*2 en cero o más pasos computacionales.

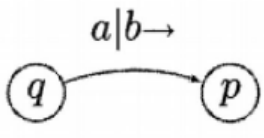
Lenguaje aceptado por una MT

Una cadena de entrada *w* es aceptada por *M* si el cómputo que inicia desde la descripción *q*0*w* termina en una descripción *w*1*pw*2, donde *p* ∈ *F* y *M* se detiene completamente. Por lo tanto, podemos definir el lenguaje aceptado por *M* como:
L(*M*) = {*w* ∈ Σ∗ | *qow* €∗ *w*1*pw*2, *p* ∈ *F, w*1, *w*2 ∈ Γ∗, *M* se detiene completamente en la descripción *w*1*pw*2}.

Vemos que, a diferencia de los autómatas que hemos estado desarrollando antes, una cadena no tiene forzosamente que ser leída en su totalidad para que sea aceptada, solo se requiere que la maquina se detenga completamente en algún momento en algún estado de aceptación. Para ello (y para simplificar) no se permitirán transiciones *δ*(*p, a*) cuando *p* ∈ *F* .

Notación en grafos

La función de transición δ de M se puede representar como un dígrafo etiquetado. Así, la transición
δ(q, a) = (p, b, →) se puede representar como:



Planteamiento del problema

Implementación de la MT

Tendremos una estructura Terna, la cual será simplemente el resultado de alguna función de transición. Contendrá el estado destino, el símbolo a ser escrito y la dirección de movimiento (-1 izquierda, 0 estacionario, 1 derecha).

La otra estructura será Turing Machine, la cual contendrá los siguientes atributos:
int numeroEstados: representara el conjunto de estados de entrada Q, donde asumiremos que

$Q = \{q0, q1, \ldots, qn-1\}$.

vector<char>alfabetoCinta: representara el alfabeto de cinta Γ.

int inicial: el estado inicial q0.

char blank: el s´ımbolo blanco ă.

vector<int>finales: el conjunto de estados finales F.

vector<map<char, terna>>: la función de transición δ. El primer argumento de Δ, que es q, estará representado por la posición en el arreglo. El segundo argumento, a, serán las claves de la tabla hash perteneciente al arreglo en la posición q. De esta forma logramos de almacenar de forma eficiente todas las transiciones.

Convertidor decimal a binario

Sea $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ y $\Gamma = \Sigma \cup \{ă\}$. Nuestra MT M recibirá una cadena $w \in \Sigma*$, es decir, la representación en base 10 de cualquier número natural. Al final del cómputo, M será capaz de dejar en la cinta la representación en binario de w, apuntando a su primer bit.

Recordemos el algoritmo usual de conversión a binario. Sea $n \in N$ el número a convertir. Sea $i \leftarrow 0$ y $q0 \leftarrow n$, entonces ejecutemos los siguientes pasos:

1. Si qi = 0 ya terminamos y vamos al paso 4.

2. Si qi f= 0, obtenemos mediante el algoritmo de la división qi+1, $ri \in Z$ tales que $qi = 2qi+1 + ri$, donde $0 \leq ri \leq 1$. Es decir, dividimos qi entre 2, guardamos el cociente en qi+1 y el residuo en ri.

3. Hacemos $i \leftarrow i + 1$ y vamos al paso 1.

4. La representación en binario de n será la concatenación de todas las ri desde la última hasta la primera, es decir, $n = (ri-1ri-2 \bullet \bullet \bullet r0)2$.

También recordemos como dividir entre 2 a mano: supongamos que queremos hallar cociente y residuo de dividir n ∈ N entre 2. Si escribimos n = a0a1 • • • ak, es decir, digito por digito de izquierda a derecha, podemos ejecutar el siguiente algoritmo: sea i ← 0, q ← n, r ← 0, entonces ejecutemos los pasos:

1. Si i > k ya acabamos y vamos al paso 4. 2.Si i ≤ k:

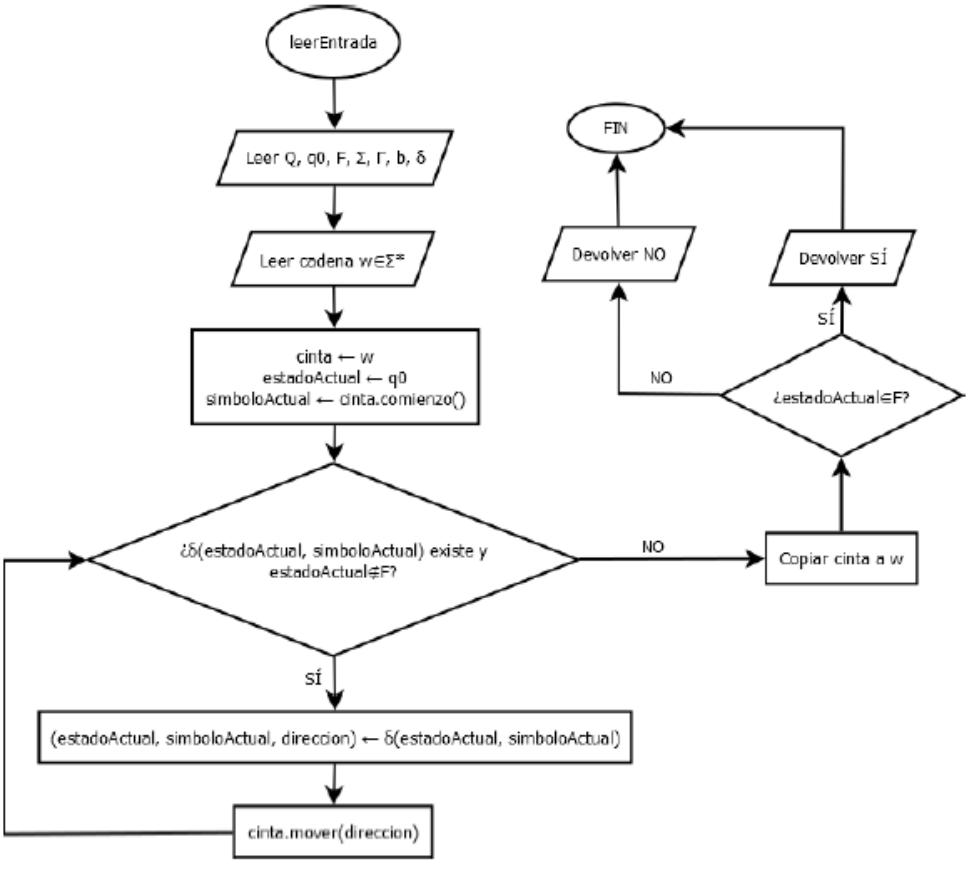
Si ai es par, hacemos ai
ai
← 2 + 5r y r ← 0.

Si ai es impar, hacemos ai
← ai − 1 + 5r y r ← 1.

3. Hacemos i ← i + 1 y vamos al paso 1. 4. El cociente será q y el residuo r.

Diagrama de flujo de la Maquina de Turing

Con ayuda del siguiente diagrama nos guamos para hacer funcionar la máquina de Turing de manera más general tomando en cuenta que la variable cinta es una lista que esta doblemente enlazada y esta se autorredimensiona con símbolos blancos cuanto sea necesario, incluso se copiara el contenido de la cinta w independientemente de si fue aceptada o no:



Convertidor decimal a binario

La séptupla que definirá a nuestra MT es:

M = ({$q0$, $q1$, $q2$, $q3$, $q4$, $q5$, $q6$, $q7$, $q8$}, $q0$, {$q8$}, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ă}, ă, δ)

En todo momento habrá un símbolo blanco 'a' separando el número en decimal y el número en binario. Al principio solo estará presente el número en decimal, y al final solo el binario.

Estado q0

Este estado será el inicial de nuestra MT, el cual, comenzando al principio del número en decimal, eliminara los ceros a la izquierda, y en cuanto encuentre un digito distinto de 0, avanzara al estado q1 quedándose en ese digito. Pero si encuentra el símbolo blanco, asumiremos que ya no queda más por procesar e iremos al estado final q8.
De esta forma, tendremos las siguientes transiciones:

$\delta(q0, n) = (q1, n, -)$ si $1 \leq n \leq 9$
$(q8, ă, \rightarrow)$ si n = ă
q0, ă, →) si n = 0

Estado q1

Significa que dividiremos el dígito actual sin considerar un residuo de la división del dígito pasado. Si el dígito actual es par no generaremos residuo, si es impar si generaremos un residuo para la próxima división. Si ya llegamos al fin de la cadena, vamos al estado q3 para insertar un 0 al principio del resultado en binario, pues no llevamos residuo. De esta forma, tendremos las siguientes transiciones:

$$\delta(q_1, n) = \begin{cases} \left(q_1, \frac{n}{2}, \rightarrow\right) & \text{si } n \text{ es par} \\ \left(q_2, \frac{n-1}{2}, \rightarrow\right) & \text{si } n \text{ es impar} \\ (q_3, 0, -) & \text{si } n = \text{b} \end{cases}$$

Estado q_2

Significa que dividiremos el digito actual considerando un residuo de la división del digito pasado. Si el digito actual es par no generaremos residuo, si es impar si generaremos un residuo para la próxima división. Si ya llegamos al fin de la cadena, vamos al estado $q3$ para insertar un 1 al principio del resultado en binario, pues si llevamos residuo. De esta forma, tendremos las siguientes transiciones:

$$\delta(q_2, n) = \begin{cases} \left(q_1, \frac{n}{2} + 5, \rightarrow\right) & \text{si } n \text{ es par} \\ \left(q_2, \frac{n-1}{2} + 5, \rightarrow\right) & \text{si } n \text{ es impar} \\ (q_3, 1, -) & \text{si } n = \text{b} \end{cases}$$

Estado q3

Vamos a recorrer el resultado en binario que ya llevamos una posición a la derecha, puesto que los residuos se concatenan del último al primero. Si el bit actual es 0, iremos al estado q4 borrándolo; y de forma similar si el bit actual el 1, iremos al estado q5 borrándolo también, asegurándonos de siempre dejar un blanco entre la entrada y la salida.

$$\delta(q_3, n) = \begin{cases} (q_4, \text{b}, \rightarrow) & \text{si } n = 0 \\ (q_5, \text{b}, \rightarrow) & \text{si } n = 1 \end{cases}$$

Estado q_4

Significa que tenemos pendiente de escribir un 0. Si el bit actual es 0 nos movemos al estado *q*4, y si es 1 nos movemos al estado *q*5. Si ya estamos en el fin de la cadena binaria, simplemente escribimos el 0 pendiente y vamos al estado *q*6 moviéndonos hacia la izquierda para regresarnos.

$$\delta(q_4, n) = \begin{cases} (q_4, 0, \rightarrow) & \text{si } n = 0 \\ (q_5, 0, \rightarrow) & \text{si } n = 1 \\ (q_6, 0, \leftarrow) & \text{si } n = \text{b} \end{cases}$$

Estado q5

Significa que tenemos pendiente de escribir un 1. Si el bit actual es 0 nos movemos al estado q4, y si es 1 nos movemos al estado q5. Si ya estamos en el fin de la cadena binaria, simplemente escribimos el 1 pendiente y vamos al estado q6 moviéndonos hacia la izquierda para regresarnos.

$$\delta(q_5, n) = \begin{cases} (q_4, 1, \rightarrow) & \text{si } n = 0 \\ (q_5, 1, \rightarrow) & \text{si } n = 1 \\ (q_6, 1, \leftarrow) & \text{si } n = \text{b} \end{cases}$$

Estado q6

Significa que nos moveremos sobre la cadena binaria de derecha a izquierda hasta que encontremos el símbolo blanco que la separa de la cadena de entrada, con el fin de regresar a hacer más divisiones.

En cuanto lo encontremos, vamos al estado q7.

$$\delta(q_6, n) = \begin{cases} (q_6, n, \leftarrow) & \text{si } 0 \leq n \leq 1 \\ (q_7, \text{b}, \leftarrow) & \text{si } n = \text{b} \end{cases}$$

Estado q7

Significa que nos moveremos sobre la cadena de entrada de derecha a izquierda hasta que encontremos un símbolo blanco, que nos indicara que ya estamos una posición antes del comienzo de la cadena de entrada, por lo que nos moveremos una posición hacia la derecha en cuanto esto pase y regresaremos al estado de inicio q0.

$$\delta(q_7, n) = \begin{cases} (q_7, n, \leftarrow) & \text{si } 0 \leq n \leq 9 \\ (q_0, \text{b}, \rightarrow) & \text{si } n = \text{b} \end{cases}$$

Estado q8

Significa que ya acabamos de convertir la cadena de entrada decimal a binario. No hay más transiciones que añadir.

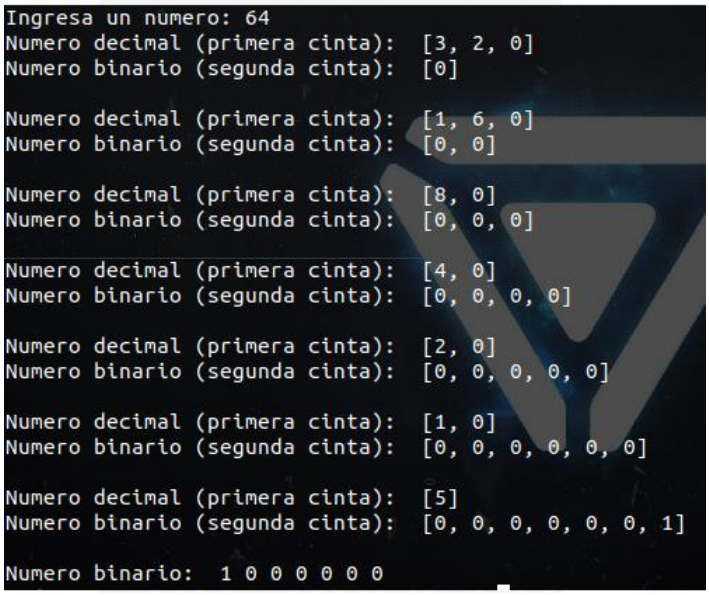
Implementación de la solución

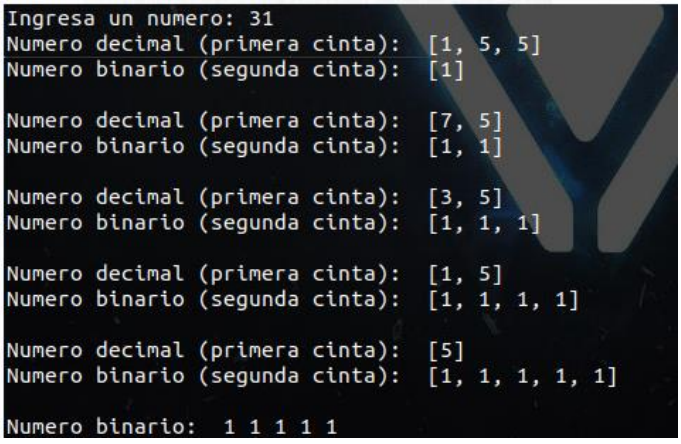
```
oddNumbers = [1,3,5,7,9]
evenNumbers = [0,2,4,6,8]
decNum = []
binNum = []
counter = 0
keepGoing = 1
def isOddNumber(num):
        return oddNumbers.count(num)
def isEvenNumber(num):
        return evenNumbers.count(num)
def addCarry(num,carry):
        if carry == 0:
                num = num
        elif carry == 5:
                if num == 0:
                        num = 5
                elif num == 1:
                        num = 6
                elif num == 2:
                        num = 7
                elif num == 3:
                        num = 8
                elif num == 4:
                        num = 9
        return num
def halving(num):
```

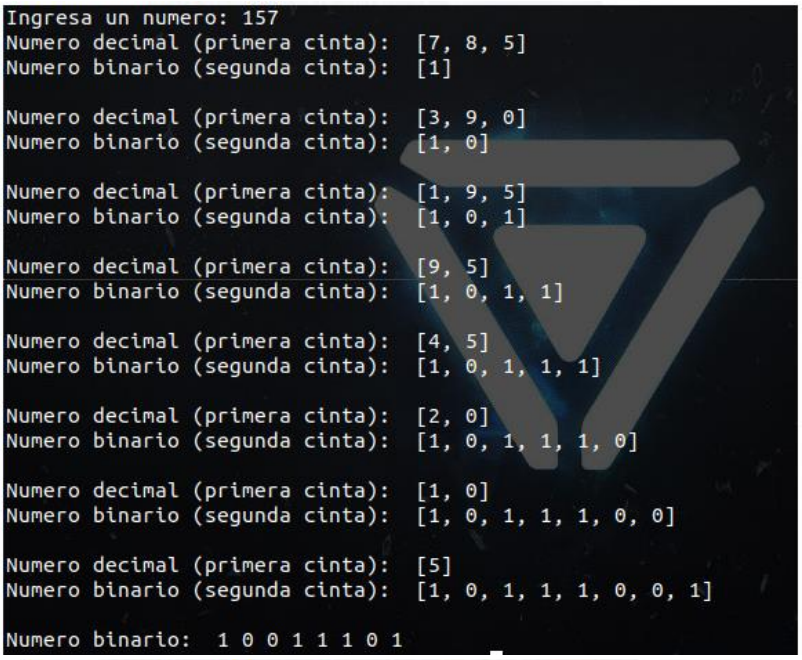
```
        if num == 0 or num == 1:
                return 0
        elif num == 2 or num == 3:
                return 1
        elif num == 4 or num == 5:
                return 2
        elif num == 6 or num == 7:
                return 3
        elif num == 8 or num == 9:
                return 4
def secondStep(numArray):
        for x in range(len(numArray)-2,-1,-1):
                if isOddNumber(numArray[x]):
                        numArray[x] = halving(numArray[x])
                        counter = 5
                        numArray[x+1] = addCarry(numArray[x+1],counter)
                elif isEvenNumber(numArray[x]):
                        numArray[x] = halving(numArray[x])
                        counter = 0
                        numArray[x+1] = addCarry(numArray[x+1],counter)

#Begin process
inp = raw_input("Ingresa un numero: ")
#First step
for x in inp:
        try:
                x = int(x)
                decNum.append(x)
        except ValueError:
                print("Eso no es un numero!")
                exit(0)
decNum.append(0)
#Second Step
while keepGoing:
        secondStep(decNum)
        if decNum[0] == 0:
                decNum.remove(0)
        if decNum[-1] == 0:
                binNum.append(0)
        elif decNum[-1] == 5:
                binNum.append(1)
```

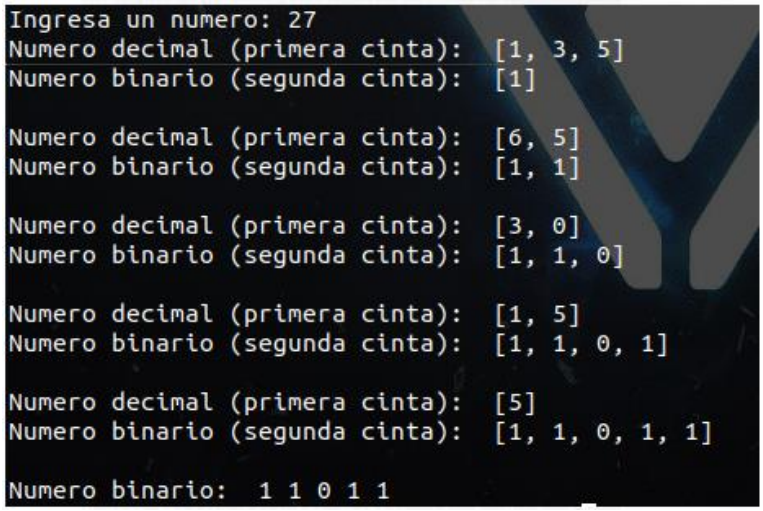
```
print ("Numero decimal (primera cinta): ", print(decNum))
print ("Numero binario (segunda cinta): ", print(binNum))
print ("")
decNum[-1] = 0
if len(decNum) == 1:
        keepGoing = 0
```

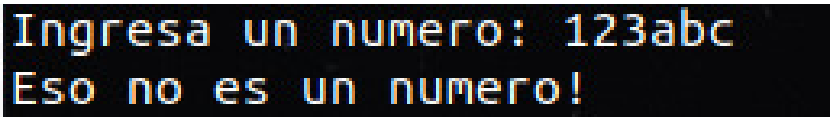
Funcionamiento











Conclusiones

Rojas Alvarado Luis Enrique

Para éste proyecto se utilizó como lenguaje de programación, de principio en C, ya que tenemos un conocimiento más amplio y se nos hacía más amigable la interfaz, pero debido a que en nuestro programa original, nosotros programábamos el algoritmo para convertir a binario tradicionalmente usado (la división entre dos, de la cadena, después el residuo lo tomas para volver a dividir y pones un 1 o un 0 dependiendo de qué residuo tengas) pero el problema es que nosotros escaneábamos ésta cadena para posteriormente dividirla y hacer operaciones con la cadena ingresada. Posteriormente nos dimos cuenta que una máquina de Turing

nunca hizo algún tipo de modificación a la cadena, ni la manipuló para llegar a un resultado puesto que en esos tiempos no lo podías computar del todo. Así que decidimos cambiar la lógica de programación, esta vez en Python ya que nos ahorramos muchas líneas de código y una ventaja es que ya puedes acceder a una pila o una lista-pila con solo llamar a una función. Básicamente se tuvieron que hacer muchos condicionales anidados para que se llegara a la conclusión de que es básicamente lo que la máquina de Turing hace, una validación a base de condicionales que describe lo que se está haciendo para llegar a la solución.

Trejo Rivera Oscar Gerardo

La máquina de Turing es una de las herramientas más importantes para la computación ya que esta nos dio las bases de la computadora como hoy la conocemos, además de que puede resolver muchos de los problemas que se plantean en ella y como tal también sentó las bases de lo que es un algoritmo, pues si recordamos un poco como un es su funcionamiento de que con lo que tenía la cinta era lo que hacía lo cual nos permitía hacer muchas actividades de manera computable, es por eso que esta máquina tiene un gran poder de funcionamiento computacional ya que en esa época podía hacer un sin fin de actividades sin importar cuantas cintas usáramos no cambiaba en mucho el funcionamiento pues la única limitante que tendría seria que describiéramos de forma muy específica el funcionamiento sea como el que deseamos y por lo que vemos en ese tiempo había muchas limitante que tenía la máquina pero con la implementación de la computadora actual muchas de estas limitaciones se corrigieron y además que a pesar de los grandes avances se sigue basando en la grandiosa máquina de Turing

Rodríguez Hernández Aldo Hassan

Se pudo complementar los conocimientos sobre la máquina de Turing al realizar este programa, la definición de la máquina de Turing se trata de una cinta infinita lo cual implica una gran cantidad de almacenamiento Tuvimos la oportunidad de adaptar el algoritmo de la conversión de decimal a binario, a una máquina de Turing determinista. De esta forma verificamos una vez más que la tesis de Church-Turing está es correcta. La Máquina de Turing es el autómata más poderoso, pues a pesar de agregarle elementos (más cintas, pistas o no determinismo) no aumenta ni disminuye su capacidad computacional. Las máquinas de Turing son muy impresionantes y lo que se vio con el proyecto, es que tienen un alcance increíble y en efecto, cualquier algoritmo computable tiene su equivalente en una máquina de Turing, asumiendo que la memoria y el tiempo de ejecución tienden a infinito.

BIBLIOGRAFÍA

1) http://maquinaturing.blogspot.com/p/funcionamiento-de-la-maquina-turing.html
2) https://books.google.com.mx/books?id=lse-ib-1JuwC