Reporte

Carlos Tonatihu Barrera Pérez

9 de septiembre de 2016

Índice

1. Alfabeto

1.1. Descripción del problema

El objetivo de esta practica es el generar las potencias del alfabeto binario $\sum = \{0, 1\}$ desde $k = 0$ hasta un k seleccionado con un máximo de $k = 1000$, para después guardar en un archivo todas las cadenas que se pudieron formar bajo estas condiciones, es decir:

$$\sum^{+} = \sum^{0} \cup \sum^{1} \cup \sum^{2} \cup \cdots \cup \sum^{1000}$$

Es importante señalar que este conjunto solo es un subconjunto de $\sum^{*}$ que representa todas las cadenas que se pueden formar con este alfabeto binario. El programa cuenta con modo manual (el usuario ingresa un k) y automático (genera su propio k).

1.2. Código

El código del programa fue realizado en C.

1.3. Pruebas

Las pruebas están divididas en modo automático y manual, en ambos dada una k se generan todas las cadenas de longitud 1 hasta k. Modo automático. Modo manual.

2. Números primos

2.1. Descripción del problema

Desarrollar un programa que encuentre todos los números primos en el intervalo $0 \leq n \leq 1000$ imprimirlos en pantalla junto con su representación en binario ademas de contar la cantidad de ceros y unos en dicho numero, finalmente guarda los números primos en su forma binaria en un archivo txt. Cuenta con modo manual (el usuario ingresa un numero n) y automático (el programa utiliza un n aleatorio).

2.2. Código

El código del programa fue realizado en Python 3.5.

2.3. Pruebas

Las pruebas están divididas en modo automático y manual. Modo automático. Modo manual.

3. AFD Palabras con terminación 'ere'

3.1. Descripción del problema

Desarrollar un autómata finito determinista capaz de encontrar las palabras con terminación 'ere' ya sea leyendo un archivo txt o en una linea de texto que el usuario ingresa, y que dichas palabras se muestren en pantalla y en el caso del archivo de texto imprimir la linea y el numero de palabra (por linea) en el que fue encontrada dicha palabra. Es importante señalar que todo aquello que no es un símbolo del alfabeto ingles, $\sum = \{a, b, ..., z, A, B, ..., Z\}$, es tomado como un espacio. Además, debe tener una opción para visualizar el siguiente diagrama. Representación del autómata en un diagrama de transiciones.

3.2. Código

El código fue realizado en Python 3.5.

3.3. Pruebas

Pruebas de las opciones del menú. Modo automático. Modo manual. Diagrama.

4. AFD Paridad en números binarios

4.1. Descripción del problema

Diseñar y programar un autómata finito determinista que acepte el lenguaje:

$$L = \{w \mid w \text{ tiene un número par de ceros y un numero par de unos}\}[1]$$

Es decir, los números binarios de entrada se generan de manera automática (cadena de longitud $n \mid 1 \leq n \leq 1000$) o manual y después se imprime si es una cadena valida o no y en ambos casos imprimir su historia. Ademas, mostrar el siguiente diagrama. Representación del autómata en un diagrama de transiciones. [1]

4.2. Código

El código fue realizado en Python 3.5.

4.3. Pruebas

Modo automático. Modo manual. Diagrama.

5. Protocolo de transmisión

5.1. Descripción del problema

Desarrollar un programa que genere 50 cadenas de 32 caracteres que sean guardadas en un archivo, para después ser evaluadas por un autómata, en este caso el autómata de paridad binaria y guardar las cadenas binarias en otro archivo, siguiendo el siguiente diagrama. Representación del autómata en un diagrama de transiciones. [2]

5.2. Código

El código fue realizado en Python 3.5.

5.3. Pruebas

Modo automático. Diagrama.

6. AFND Números binarios con terminación '01'

6.1. Descripción del problema

Desarrollar un autómata finito no determinista, que acepte todas y sólo las cadenas formadas por ceros y unos que terminan en 01. Asimismo, imprimir la tabla de transiciones (historia) y que la entrada de cadenas sea de forma manual o automática, la cadena automática debe de tener una longitud $n \mid 1 \leq n \leq 1000$. Y que contenga la opción de mostrar el siguiente diagrama. Diagrama de transiciones del autómata. [1]

6.2. Código

El código fue realizado en Python 3.5.

6.3. Pruebas

Modo automático. Modo manual. Diagrama.

Referencias

[1] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introducción a La Teoría De Autómatas, Lenguajes Y Computación*. Addison-Wesley, 2007.

[2] J. D. Ullman, "Finite automata." http://infolab.stanford.edu/~ullman/ialc/spr10/slides/fa1.pdf, 2010. Accessed: 2016-09-10.