UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Proyecto 3:

Implementación de Máquinas de Turing para encriptar y desencriptar un mensaje mediante el cifrado Caesar

Documentación

José Prince, 22087
Fabiola Contreras, 22787
Maria Jose Villafuerte, 22129

Alan Gerardo Reyes Figueroa
Teoría de la computación
Guatemala, 2024

Índice

I. Diseño de la aplicación

Como lenguaje de programación para el autómata, se decidió utilizar Java. Esto es debido a que al ser un lenguaje de programación diseñado para POO esto nos permite tener una mejor organización en el proyecto. Otra ventaja de usar Java es que es un lenguaje fuertemente tipado, lo que hace que se puedan detectar errores con más facilidad.

Proceso paso a paso:

Main

1. **Entrada del usuario**:

   Se solicita al usuario que ingrese el texto a encriptar/desencriptar en el formato: [k]#texto

   - Donde k es la clave del cifrado y texto es el contenido a procesar.
   - El usuario selecciona si desea encriptar o desencriptar el texto.
2. **Carga de configuración de la máquina de Turing**:
   - Se lee el archivo encrypt.json o desencrypt.json, que contiene la configuración de la máquina, incluyendo los estados, el alfabeto, y las transiciones (delta).
3. **Creación del encriptador**:
   - Se inicializa una instancia de la clase Caesar_Cipher, que configura las cintas de la máquina basándose en la clave k y el texto de entrada.
4. **Proceso de encriptación**:
   - Se invoca el método derivation, que realiza las transiciones según las reglas de la máquina de Turing, retornando el texto encriptado/desencriptado.
5. **Impresión del resultado**

Caesar_Cipher

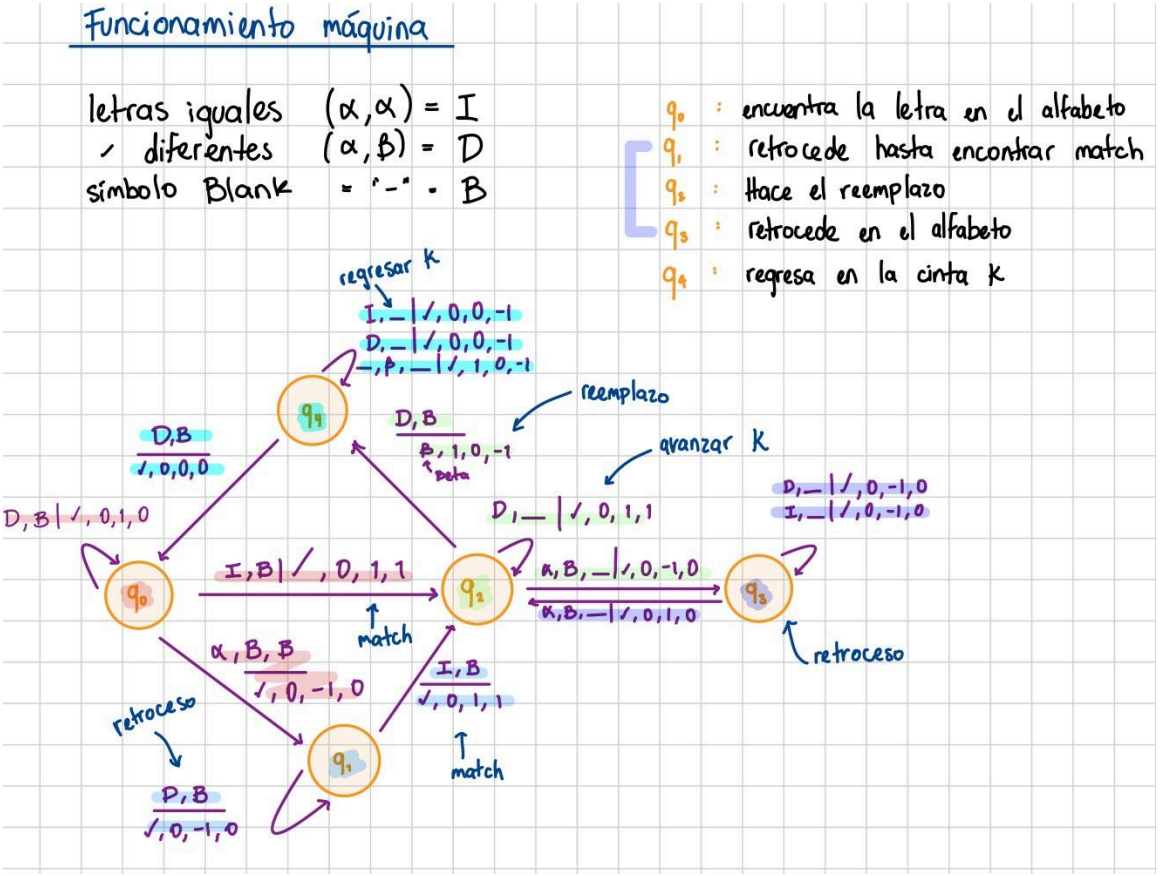
1. **Inicialización de variables**:
   - Se divide la entrada (input) en dos partes:
     - key: Número entero que representa la clave de cifrado.
     - input: Texto a procesar (convertido a minúsculas y sin espacios adicionales).
   - Se llama a tapesInitialization para configurar las tres cintas necesarias:
     - inputTape: Contiene el texto de entrada y símbolos blank (-).
     - alphabetTape: Contiene el alfabeto extendido y blank.
     - kTape: Representa la clave de cifrado.

2. **Inicialización de cintas (tapesInitialization)**
    - Se crean listas (cintas) que contienen el contenido inicial de cada una:
        - **inputTape**: Se llena con delimitadores y los caracteres del texto de entrada.
        - **alphabetTape**: Contiene el alfabeto seguido por delimitadores.
        - **kTape**: Representa la posición relativa basada en k. Tiene espacios en blanco como representantes de la clave k.

3. **Derivación de la cadena (derivation)**
    1. **Configuración inicial**:
        - Define el estado inicial de la máquina y las posiciones de los punteros para cada cinta.
        - Obtiene los caracteres actuales desde las cintas.
    2. **Ejecuta transiciones**:
        - Mientras el estado no sea el de aceptación **y** la cinta de entrada no contenga -, se ejecutan transiciones mediante el método delta_transitions.
        - Cambiando estados y modificando las cintas según lo configurado en la máquina.
    3. **Construcción del resultado**:
        - Al final del bucle, se convierte el contenido de inputTape en un solo texto concatenado, eliminando los delimitadores (-).

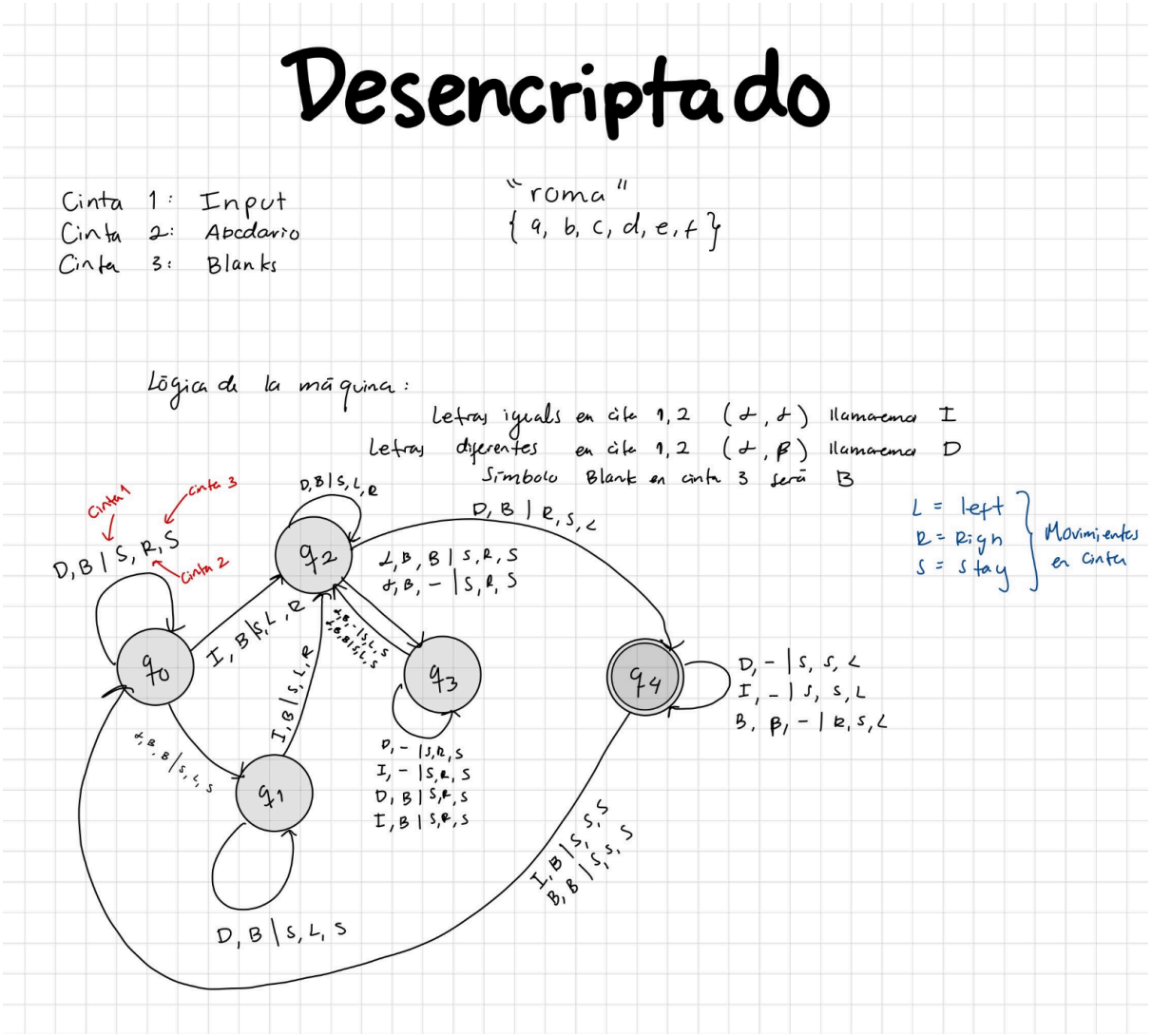
4. **Uso de transiciones (delta_transitions)**
    1. Calcula la nueva configuración de la máquina basándose en:
        a. El estado actual.
        b. Los caracteres en las posiciones actuales de las cintas.
    2. Genera una clave (keyValue) para buscar las transiciones en el mapa delta.
    3. Realiza las siguientes modificaciones:
        a. Actualiza los caracteres en las cintas.
        b. Cambia la dirección de los punteros basándose en las reglas (S, R, L).
        c. Modifica el estado actual.

Diseño de la Máquina de Turing (encriptación):




```
{
    "Q": ["q0", "q1", "q2", "q3", "q4"],
    "sigma": ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z", " "],
    "gamma": ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z", " ", "-"],
    "q0": "q0",
    "F": ["q4"],
    "Blanc": "-",
    "delta": {
        "q0, alpha, beta, -": ["q0", "alpha", "beta", "-", "S", "R", "S"],
        "q0, alpha, alpha, -": ["q2", "alpha", "alpha", "-", "S", "R", "R"],
        "q0, alpha, -, -": ["q1", "alpha", "-", "-", "S", "L", "S"],
        "q1, alpha, beta, -": ["q1", "alpha", "beta", "-", "S", "L", "S"],
        "q1, alpha, alpha, -": ["q2", "alpha", "alpha", "-", "S", "R", "R"],
        "q2, alpha, beta,  ": ["q2", "alpha", "beta", " ", "S", "R", "R"],
        "q2, alpha, -,  ": ["q3", "alpha", "-", " ", "S", "L", "S"],
        "q2, alpha, -, -": ["q3", "alpha", "-", "-", "S", "L", "S"],
        "q2, alpha, beta, -": ["q4", "beta", "beta", "-", "R", "S", "L"],
        "q3, alpha, beta,  ": ["q3", "alpha", "beta", " ", "S", "L", "S"],
        "q3, alpha, alpha,  ": ["q3", "alpha", "beta", " ", "S", "L", "S"],
        "q3, alpha, -,  ": ["q2", "alpha", "-", " ", "S", "R", "S"],
        "q3, alpha, beta, -": ["q3", "alpha", "beta", "-", "S", "L", "S"],
        "q3, alpha, alpha, -": ["q3", "alpha", "beta", "-", "S", "L", "S"],
        "q3, alpha, -, -": ["q2", "alpha", "-", "-", "S", "R", "S"],
        "q4, alpha, beta,  ": ["q4", "alpha", "beta", " ", "S", "S", "L"],
        "q4, alpha, alpha,  ": ["q4", "alpha", "alpha", " ", "S", "S", "L"],
        "q4,  , beta,  ": ["q4", " ", "beta", " ", "R", "S", "L"],
        "q4, alpha, alpha, -": ["q0", "alpha", "alpha", "-", "S", "S", "S"],
        "q4, alpha, beta, -": ["q0", "alpha", "beta", "-", "S", "S", "S"]
    }
}
```

Diseño de la Máquina de Turing (desencriptado):




```
{
  "Q": ["q0", "q1", "q2", "q3", "q4"],
  "sigma": ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z", " "],
  "gamma": ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z", " ", "-"],
  "q0": "q0",
  "f": ["q4"],
  "Blanc": "-",
  "delta": {
    "q0, alpha, beta, -": ["q0", "alpha", "beta", "-", "S", "R", "S"],
    "q0, alpha, alpha, -": ["q2", "alpha", "alpha", "-", "S", "L", "R"],
    "q0, alpha, -, -": ["q1", "alpha", "-", "-", "S", "L", "S"],
    "q1, alpha, beta, -": ["q1", "alpha", "beta", "-", "S", "L", "S"],
    "q1, alpha, alpha, -": ["q2", "alpha", "alpha", "-", "S", "L", "R"],
    "q2, alpha, beta,  ": ["q2", "alpha", "beta", " ", "S", "L", "R"],
    "q2, alpha, -,  ": ["q3", "alpha", "-", " ", "S", "R", "S"],
    "q2, alpha, -, -": ["q3", "alpha", "-", "-", "S", "R", "S"],
    "q2, alpha, beta, -": ["q4", "beta", "beta", "-", "R", "S", "L"],
    "q3, alpha, beta,  ": ["q3", "alpha", "beta", " ", "S", "R", "S"],
    "q3, alpha, alpha,  ": ["q3", "alpha", "beta", " ", "S", "R", "S"],
    "q3, alpha, -,  ": ["q2", "alpha", "-", " ", "S", "L", "S"],
    "q3, alpha, beta, -": ["q3", "alpha", "beta", "-", "S", "R", "S"],
    "q3, alpha, alpha, -": ["q3", "alpha", "beta", "-", "S", "R", "S"],
    "q3, alpha, -, -": ["q2", "alpha", "-", "-", "S", "L", "S"],
    "q4, alpha, beta,  ": ["q4", "alpha", "beta", " ", "S", "S", "L"],
    "q4, alpha, beta, -": ["q0", "alpha", "beta", "-", "S", "S", "S"],
    "q4, alpha, alpha,  ": ["q4", "alpha", "beta", " ", "S", "S", "L"],
    "q4, alpha, alpha, -": ["q0", "alpha", "beta", "-", "S", "S", "S"],
    "q4,  , beta,  ": ["q4", " ", "beta", " ", "R", "S", "L"]
  }
}
```

II. Repositorio en Github

https://github.com/Jose-Prince/Caesar-Cipher-Turing-Machine

III. Discusión

1. Obstáculos

Durante el desarrollo del proyecto se pudieron encontrar diferentes errores en el desarrollo. Se encontraron errores como la ideación de la máquina de Turing y en la codificación del código en donde no se tuvo en cuenta algunas circunstancias al momento de ingresar los datos de la máquina. Aunque no fueron graves los errores si dieron problemas en el desarrollo del proyecto.

Primeramente la ideación, no llevó a tener la máquina de Turing más eficaz; debido a que se pensó en solo una idea para la máquina de Turing y esa fue la que se llevó a cabo. Con esta máquina se tuvo el problema que tenía varias cintas y tenía varios comportamientos entre la comparación de los caracteres de la primera y la segunda cinta. Esto hizo que si se calcula el número total de transiciones que llegaría a tener la máquina este sería fácilmente mayor a mil transiciones. Tanto hacer la creación de cada transición hubiera sido una tarea bastante tediosa, incluso si se hubieran declarado mediante un ciclo for o similar. Esto conllevo a tener una máquina de turing simplificada que terminó por tener solo 14 transiciones y que estas transiciones tienen caracteres codificados como “alpha” y “beta” de la primera y segunda cinta, esto para facilitar la escritura de las transiciones.

Para el código desarrollado no hubo tantas complicaciones pero sí se dieron problemas mínimos. Uno de esos pocos problemos fue el uso del espacio en el input debido a que el input en el programa se escribe con el siguiente formato: k # “cadena”, esto hizo que si se ingresaba el input con espacios, el código diera errores en el parsing de datos como lo era el número k o no encontraba la transición correcta. Estos errores conllevan en la creación de nuevas transiciones en la máquina de Turing que la hicieron más robusta.

2. Recomendaciones

Para la creación de futuros proyectos similares a este, se recomienda que se siga trabajando en un lenguaje tipado. Esto facilita el proceso de desarrollo porque se tiene un mejor control en el tipado de las variables. Como segunda recomendación, sería bueno que se haga un diseño con mayor cuidado para la máquina de Turing debido a que la primera idea no siempre es la mejor y es mejor si la máquina se diseña entre varias personas, para tener una mejor perspectiva y diseñar una máquina eficiente y que no termine con tantas transiciones, evitando que se hagan trucos raros para el funcionamiento de la máquina de Turing.

IV. Ejemplos y pruebas realizadas

Encriptado

```
@ParameterizedTest
@CsvSource({
        "3#hola mundo, krod pxqgr",
        "5#java test, ofaf yjxy",
        "1#cipher, djqifs",
        "7#prueba, wyblih",
        "3 # ROMA NO FUE CONSTRUIDA EN UN DIA, URPD QR IXH FRQVWUXLGD HQ XQ GLD"
})
void testCompleteFlow(String input, String expected) {
    Caesar_Cipher cipher = new Caesar_Cipher(machine, input);
    String result = cipher.derivation(machine.getQ0(), machine.getF().get(index:0), machine.getDelta());

    assertNotNull(result, "El texto encriptado no debería ser nulo.");
    assertEquals(expected.toLowerCase(), result,
            "El resultado del cifrado debería ser correcto para la clave proporcionada.");
}
```

```
e-invocation:#5]
%TESTS  11,testCompleteFlow(com.mycompany.app.EncriptTest)

%TESTE  11,testCompleteFlow(com.mycompany.app.EncriptTest)
%TESTS  6,testTransitions(com.mycompany.app.EncriptTest)
%TESTE  6,testTransitions(com.mycompany.app.EncriptTest)

%RUNTIME577
```

Test Runner for Java
- testTransitions()
- [1] 3#hola mundo, krod pxqgr
- [2] 5#java test, ofaf yjxy
- [3] 1#cipher, djqifs
- [4] 7#prueba, wyblih
- [5] 3 # ROMA NO FUE CONSTRUIDA EN UN DIA, URPD QR IXH F...

Desencriptado

```
@ParameterizedTest
@CsvSource({
        "3#krod pxqgr, hola mundo",
        "5#ofaf yjxy, java test",
        "1#djqifs, cipher",
        "7#wyblih, prueba",
        "3 # URPD QR IXH FRQVWUXLGD HQ XQ GLD, ROMA NO FUE CONSTRUIDA EN UN DIA"
})
void testCompleteFlow(String input, String expected) {
    Caesar_Cipher cipher = new Caesar_Cipher(machine, input);
    String result = cipher.derivation(machine.getQ0(), machine.getF().get(index:0), machine.getDelta());

    assertNotNull(result, "El texto desencriptado no debería ser nulo.");
    assertEquals(expected.toLowerCase(), result,
            "El resultado del cifrado debería ser correcto para la clave proporcionada.");
}
}
```

```
%TESTE  11,testCompleteFlow(com.mycompany.app.DesencriptTest)

%TESTS  6,testTransitions(com.mycompany.app.DesencriptTest)

%TESTE  6,testTransitions(com.mycompany.app.DesencriptTest)

%RUNTIME483
```

Test Runner for Java
- testTransitions()
- [1] 3#krod pxqgr, hola mundo
- [2] 5#ofaf yjxy, java test
- [3] 1#djqifs, cipher
- [4] 7#wyblih, prueba
- [5] 3 # URPD QR IXH FRQVWUXLGD HQ XQ GLD, ROMA NO FUE...

V. Referencias utilizadas

- Tanenbaum, A. S., & Austin, T. (2012). Structured Computer Organization (6th ed.). Pearson.
- Patterson, D. A., & Hennessy, J. L. (2014). Computer Organization and Design (5th ed.). Morgan Kaufmann.
- Wegener, I. (2005). *Complexity Theory: Exploring the Limits of Efficient Algorithms*. Springer.
- Du, D. Z., & Ko, K. I. (2014). *Theory of Computational Complexity*. John Wiley & Sons.