Universidad Autónoma del Estado de Hidalgo

Licenciatura en Ciencias Computacionales

Reporte De Práctica No. 2

Alumno:

Sergio García León

Práctica 2. AFD y AFND

INSTRUCCIONES. Resuelve el siguiente reto.
https://omegaup.com/arena/problem/Simulacion-de-AFD/

El constructor inicializa el AFD con el número de estados, el estado inicial, el conjunto de estados de aceptación y el alfabeto.

```
public AFD(int numEstados, int estadoInicial, Set<Integer> estadosAceptacion, Set<Character> alfabeto) {
    this.numEstados = numEstados;
    this.estadoInicial = estadoInicial;
    this.estadosAceptacion = estadosAceptacion;
    this.alfabeto = alfabeto;
    this.transiciones = new HashMap<>();
}
```

Este método agrega una transición desde un estado con nombre desde, con un símbolo nombrado simbolo, hacia un estado nombrado hacia.

```
public void agregarTransicion(int desde, char simbolo, int hacia) {
    transiciones.putIfAbsent( key: desde, new HashMap<>());
    transiciones.get( key: desde).put( key: simbolo,  value: hacia);
}
```

Este método verifica si una cadena es aceptada por el AFD.

- Comienza en el estado inicial y lee cada símbolo de la cadena.
- Para cada símbolo, verifica si el símbolo pertenece al alfabeto y si existe una transición definida desde el estado actual.

```
public boolean aceptaCadena(String cadena) {
    int estadoActual = estadoInicial;
    for (char simbolo : cadena.toCharArray()) {
        if (!alfabeto.contains( o: simbolo)) {
            return false; // Símbolo no pertenece al alfabeto
        }
        if (!transiciones.containsKey( key: estadoActual) || !transiciones.get( key: estadoActual).containsKey( key: simbolo)) {
            return false; // No hay transición definida
        }
        estadoActual = transiciones.get( key: estadoActual).get( key: simbolo);
    }
    return estadosAceptacion.contains( o: estadoActual);
}
```

Se lee el alfabeto y se almacena en un conjunto alfabeto.

```
String[] simbolosAlfabeto = entrada.nextLine().split( regex: "\\s+");
Set<Character> alfabeto = new HashSet<>();
for (String simboloStr : simbolosAlfabeto) {
    if (!simboloStr.isEmpty()) {
        alfabeto.add( e: simboloStr.charAt( index: 0));
    }
}
```

Se leen los estados de aceptación y se guardan en el conjunto estadosAceptacion.

```
String[] estadosAceptacionStr = entrada.nextLine().split( regex: "\\s+");
Set<Integer> estadosAceptacion = new HashSet<>();
for (String estadoStr : estadosAceptacionStr) {
    if (!estadoStr.isEmpty()) {
        estadosAceptacion.add( e: Integer.parseInt( s: estadoStr));
    }
}
```

Se crea una instancia de la clase AFD con los datos leídos. Y despues se leen las transiciones del AFD, especificando para cada estado desde qué estado, con qué símbolo, y hacia qué estado se debe transitar.

```
// Crear el AFD
AFD afd = new AFD( numEstados: numes,  estadoInicial: q0, estadosAceptacion, alfabeto);

// Leer las transiciones
for (int i = 0; i < tra; i++) {
    int desde = entrada.nextInt();
    String simboloStr = entrada.next();
    char simbolo = simboloStr.charAt( index: 0);
    int hacia = entrada.nextInt();
    afd.agregarTransicion(desde, simbolo, hacia);
}

entrada.nextLine(); // Consumir el resto de la línea
```

- Para cada cadena ingresada, se utiliza el método aceptaCadena para determinar si es aceptada o rechazada.
- Se imprime ”ACEPTADA” si la cadena es aceptada, y ”RECHAZADA” si es rechazada.

```
// Verificar las cadenas
for (int i = 0; i < cad; i++) {
    String cadena = entrada.nextLine();
    if (afd.aceptaCadena(cadena)) {
        System.out.println( x: "ACEPTADA");
    } else {
        System.out.println( x: "RECHAZADA");
    }
}

entrada.close();
```

Este programa fue hecho con ayuda de IA para ser mas exactos de chat GPT, la primera imagen muestra como tenia el programa al leer el alfabeto y los estados de aceptación lo cual me causaba errores, así que le pregunte a la IA "¿Dónde se encuentra el error de mi programa?" dando como corrección el programa en la segunda imagen.

```
// Leer alfabeto
int numSimbolos = sc.nextInt();
Set<Character> alfabeto = new HashSet<>();
for (int i = 0; i < numSimbolos; i++) {
    alfabeto.add( e: sc.next().charAt(0));
}

// Leer estados de aceptación
Set<Integer> estadosAceptacion = new HashSet<>();
for (int i = 0; i < numEstadosAceptacion; i++) {
    estadosAceptacion.add( e: sc.nextInt());
}
```

```
// Leer el alfabeto
String[] simbolosAlfabeto = entrada.nextLine().split( regex: "\\s+");
Set<Character> alfabeto = new HashSet<>();
for (String simboloStr : simbolosAlfabeto) {
    if (!simboloStr.isEmpty()) {
        alfabeto.add( e: simboloStr.charAt( index: 0));
    }
}

// Leer los estados de aceptación
String[] estadosAceptacionStr = entrada.nextLine().split( regex: "\\s+");
Set<Integer> estadosAceptacion = new HashSet<>();
for (String estadoStr : estadosAceptacionStr) {
    if (!estadoStr.isEmpty()) {
        estadosAceptacion.add( e: Integer.parseInt( s: estadoStr));
    }
}
```