Reporte automata de pila

Murga Dionicio Miguel Angel

Enero 2021

1 Problema

Elaborar un programa implementando con un autómata de pila para reconocer el lenguaje libre de contexto $0^n1^n|n >= 1$.

Adicionalmente, el programa debe de contar con las siguientes características:

1. La cadena puede ingresar por el usuario o automáticamente.
2. Mandar a un archivo y en pantalla la evaluación del autómata a través de descripciones instantáneas.
3. Animar el autómata de pila.
4. La longitud máxima será manejar cadenas de 100,000 caracteres

2 ¿Cómo ejecutar el programa?

Para poder ejectuar el programa es necesario contar con Node.js y npm en sus últimas versiones en el sistema, además de un navegador en donde se plasmará el html(El programa usa html 5 así que es necesario un navegador que lo soporte). Si ya se tienen los requerimientos entonces nos metemos a la carpeta donde está el index.js, es decir, al primer nivel de la carpeta y ejecutar el siguiente código en la terminal: 'npm run dev'. Esto para poder correr el servidor el cuál será el encargado de hacer el proceso.
Ya que se esté corriendo el servidor dirá el puerto donde se estará corriendo (tiene por defecto el puerto 3000 aunque puede ser cambiado) e ingresamos a la siguiente url: 'http://localhost:3000'.

3 Solución

Para empezar a resolver el programa es necesario crear una pila. En JavaScript las pilas las creamos como clases, en donde en el contructor tenemos 2 variables que guardarán los valores. La primera variable es un arreglo el cuál guardará todos los datos y la otra variable es una variable que contiene el número de índice del último valor agregado.

```
class Stack{
    constructor(){
        this.items = [];
        this.top = 0;
    };

    push(item){
        this.top++;
        this.items[this.top] = item;
    }
```

```
    pop(){
        let deletedItem;
        if(this.top){
            deletedItem = this.items[this.top];
            delete this.items[this.top];
            this.top--;
            return deletedItem;
        }
    }

    print(){
        let dataString = "";
        for(let i = 0; i<this.top; i++){
            dataString+= this.items[i];
        }
        console.log(dataString);
        return dataString;
    }

    getSize(){
        return this.top;
    }
}

module.exports = Stack;
```

En una página html obtenemos la cadena del usuario o generamos una.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <%- include('partials/cdn.html') -%>
    <title>Stack automata</title>
</head>

<body style="background-color:#1F1C18">
    <div id="input">
        <form action="http://localhost:3000/get_data" method="POST">
            <h4 style="color: white;">Porfavor inserta una cadena de 0 y 1</h4>
            <input type="text" name="dinput" id="idInput">
```

```
            <button type="submit">Aceptar</button>
            <button type="button" onclick="randomString()">Auto generar</button>
        </form>
    </div>
    <div id="stackAnimation" style="color: white;">
        <h1>
            Stack animation
        </h1>
    </div>
    <script>
        function randomString(){
            let randomNumber= Math.floor(Math.random() * 100001);
            let stringToSet = "";

            for(let i = 0; i<randomNumber; i++){
                stringToSet+="0";
            }
            for(let i = 0; i<randomNumber; i++){
                stringToSet+="1";
            }
            document.getElementById('idInput').value = stringToSet;
        }
    </script>
</body>

</html>
```

Ahora pasamos a la funcionalidad principal del programa, ya que obtiene los datos va obtener la cantidad de 0 para poder meter las X en la pila y de 1 para poder sacarlas. Ya que se obtuvo ese valor ese valor se hace un ciclo para poder obtener la pila, y posteriormente se dedica a meter las X a la pila de una en una si el valor correspondiente es 0 y saca las X si el valor es 1.

```
const { render } = require('ejs');
const express = require('express');
const router = express.Router();
const bodyParser = require('body-parser');
const Stack = require('../stack');
const fs = require('fs');
router.use(bodyParser.urlencoded({limit: '50mb', extended: true}));

router.get('/', (req,res)=>{
    res.render('index.html');
});
```

```
router.post('/get_data', (req,res)=>{
    let data = req.body.dinput;
    let contador0 = 0;
    let contador1 = 0;
    for(let q = 0; q < data.length; q++){
        if(data[q] == "0") contador0++;
        else contador1++;
    }
    if(contador0 == contador1){
        //Aplication
        fs.createWriteStream('output.txt');
        let substring;
        let state = "q";
        let xsubstring = "";
        let size=0;
        //fs.appendFileSync('output.txt', `d(q,${data},Z)->\n`);
        const stack = new Stack;
        for(let i = 0; i<data.length; i++){
            size = stack.getSize();
            console.log(size);
            for(let r = 0; r < size; r++){
                xsubstring += "X";
            }
            substring = data.substring(i,data.length);
            fs.appendFileSync('output.txt', `d(${state},${substring},${xsubstrin

            if(data[i] == 0){
                stack.push("X");
            } else {
                state="p";
                stack.pop();
            };

            substring = "";
            xsubstring="";
        }
        fs.appendFileSync('output.txt', `d(p,e,Z)->\n`);
        fs.appendFileSync('output.txt', `d(f,e,Z)`);
        let process = fs.readFileSync('output.txt');
        res.render('stackAnimation.html', {process: process});
    }else{
        res.render('stackAnimation.html', {process: "Cadena inv lida , no hay la
    }
});
```

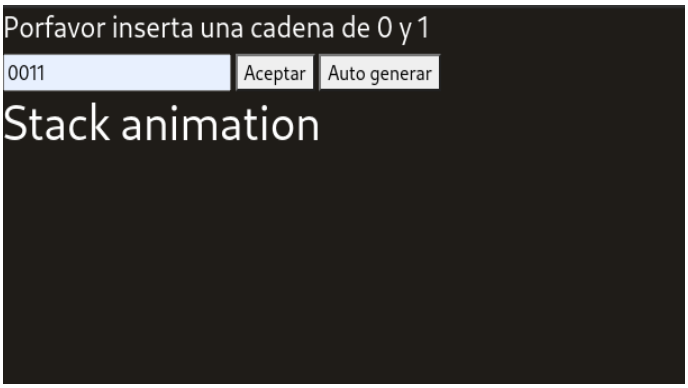
```
module.exports = router;
```

---

4 Ejemplo

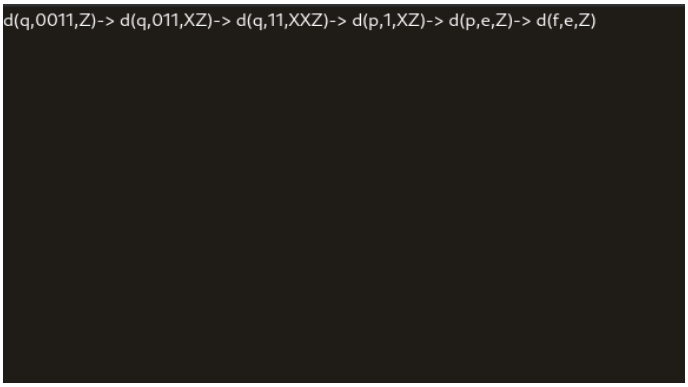
Si por ejemplo insertamos '0011' en el input nos generará el siguiente archivo:

Página donde se inserta el valor



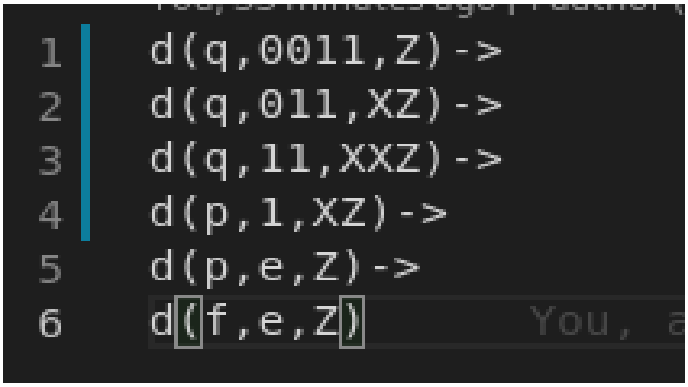
Ahora vemos cuál es la cadena final.

Página donde se muestra el proceso



Y por último veamos cómo queda el archivo:

Archivo que se generó



5 Comentarios

1. La animación de la pila no está incluida ya que no se terminó el código de la animación.