



Instituto Politécnico Nacional

# INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

## PRÁCTICA 4: BUSCADOR DE PALABRAS

*Teoría de la Computación*

Autor:  
Hernández Vergara Eduardo

Junio 2022

# Índice

<b>1. Introducción</b>	<b>2</b>
1.1. Objetivo . . . . .	2
1.2. Automatas finitos (AFD) . . . . .	2
1.3. Automata Finito No Determinista (AFN) . . . . .	2
1.3.1. AFN para búsqueda de texto . . . . .	2
<b>2. Solución</b>	<b>4</b>
2.1. Correr el Programa . . . . .	4
2.2. Grafos . . . . .	7
<b>3. Código</b>	<b>10</b>
3.1. El automata . . . . .	10
3.2. Realizacion del Grafo por medio de ReactJS . . . . .	23

# 1. Introducción

## 1.1. Objetivo

En la práctica desarrollada tuvimos que diseñar un buscador por medio del AFD, claro que primero tenemos que convertir el AFN a el AFD.

## 1.2. Automatas finitos (AFD)

Un autómata finito (AF) o máquina de estado finito es un modelo computacional que realiza cálculos en forma automática sobre una entrada para producir una salida. Este modelo está conformado por un alfabeto, un conjunto de estados finito, una función de transición, un estado inicial y un conjunto de estados finales. Su funcionamiento se basa en una función de transición, que recibe a partir de un estado inicial una cadena de caracteres pertenecientes al alfabeto (la entrada), y que va leyendo dicha cadena a medida que el autómata se desplaza de un estado a otro, para finalmente detenerse en un estado final o de aceptación, que representa la salida. La finalidad de los autómatas finitos es la de reconocer lenguajes regulares, que corresponden a los lenguajes formales más simples según la Jerarquía de Chomsky.

## 1.3. Automata Finito No Determinista (AFN)

Un autómata finito no determinista (AFN) tiene la capacidad de estar en varios estados a la vez. Esta capacidad a menudo se expresa como la posibilidad de que el autómata conjeture algo acerca de su entrada. Por ejemplo, cuando el autómata se utiliza para buscar determinadas secuencias de caracteres (por ejemplo, palabras clave) dentro de una cadena de texto larga, resulta útil conjeturar que estamos al principio de una de estas cadenas y utilizar una secuencia de estados únicamente para comprobar la aparición de la cadena, carácter por carácter.

### 1.3.1. AFN para búsqueda de texto

Supongamos que tenemos un conjunto de palabras, que denominaremos palabras clave, y deseamos hallar las apariciones de cualquiera de estas palabras. En aplicaciones de este tipo, una forma útil de proceder consiste en diseñar un autómata finito no determinista que indique, mediante un estado de aceptación, que ha encontrado una de las palabras clave. El texto de un documento se introduce carácter a carácter en este AFN, el cual reconoce a continuación las apariciones de las palabras clave en dicho texto. Existe una forma simple para que un AFN reconozca un conjunto de palabras clave.

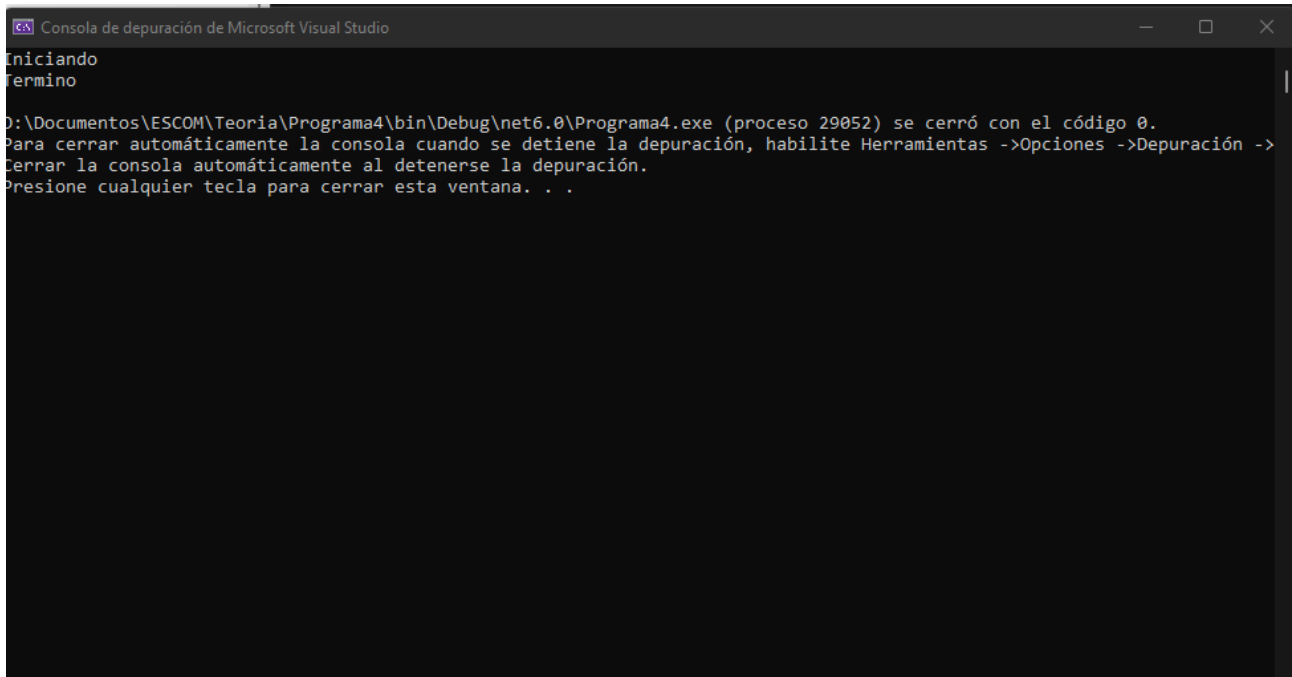
1. Hay un estado inicial con una transición a sí mismo para cada uno de los símbolos de entrada, por ejemplo, todos los caracteres ASCII imprimibles si estamos examinando texto. Intuitivamente, el estado inicial representa una conjetura de que todavía no hemos detectado una de las palabras clave, incluso aunque hayamos encontrado algunas de las letras de una de esas palabras.
2. Para cada palabra clave  $a_1 a_2 \dots a_k$ , existen  $k$  estados, por ejemplo,  $q_1, q_2, \dots, q_k$ . Existe una transición desde el estado inicial a  $q_1$  para el símbolo  $a_1$ , una transición desde

$q_1$  a  $q_2$  para el símbolo  $a_2$ , etc. El estado  $q_k$  es un estado de aceptación e indica que se ha encontrado la palabra clave  $a_1a_2 \dots a_k$ .

## 2. Solución

Para la implementación de la solución se uso C#, con .Net 6.0, con el IDE Visual Studio 2022, como veremos a continuación con las capturas de resultado, veremos como salio.

### 2.1. Correr el Programa



```
Consola de depuración de Microsoft Visual Studio
Iniciando
Termino
D:\Documentos\ESCOM\Teoria\Programa4\bin\Debug\net6.0\Programa4.exe (proceso 29052) se cerró con el código 0.
Para cerrar automáticamente la consola cuando se detiene la depuración, habilite Herramientas ->Opciones ->Depuración ->
Cerrar la consola automáticamente al detenerse la depuración.
Presione cualquier tecla para cerrar esta ventana. . .
```

Figura 1: Es lo que nos aparece en la consola

Como no se nos especifico poner algo a la entrada del programa, simplemente cuando lo corremos no necesitamos pasarle ningun parametro sino solo se ejecuta dandonos unos mensajes de iniciado y terminado En esta captura vemos lo que seria el Documento que usamos para probar que funcionara

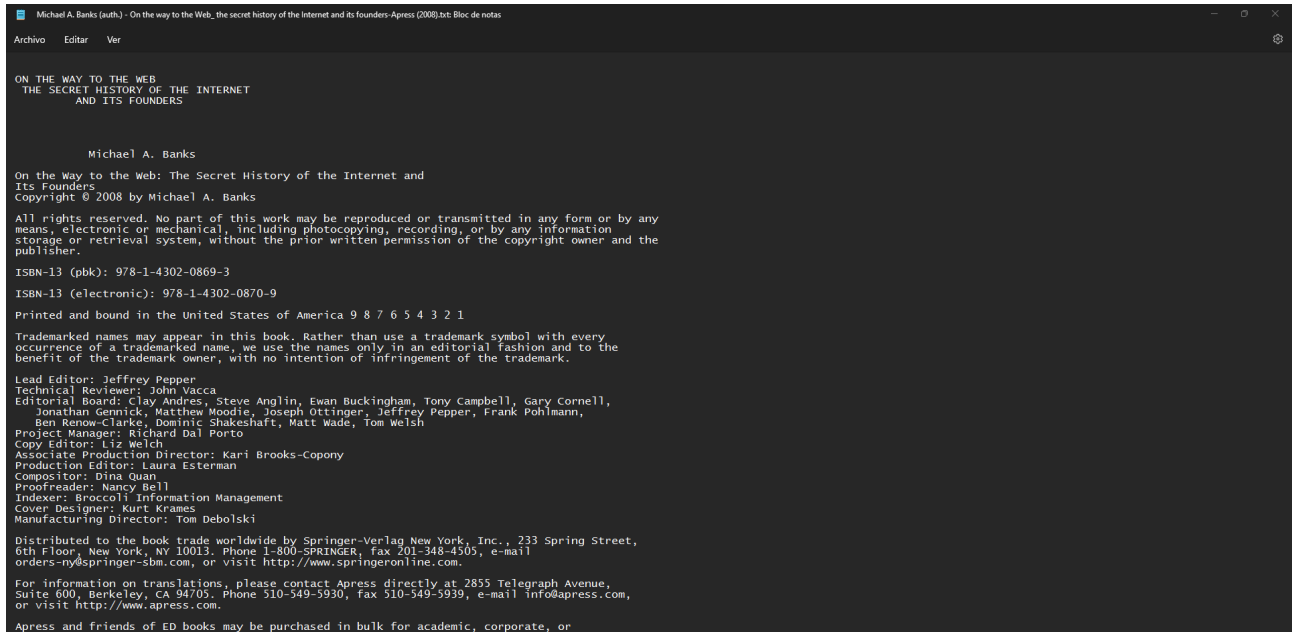


Figura 2: Es el documento de Prueba

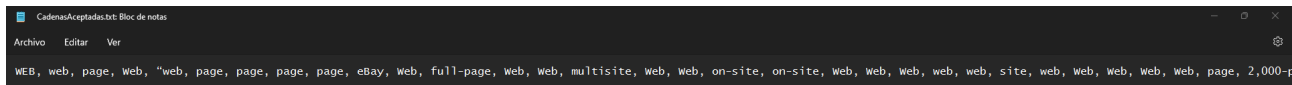


Figura 3: Cadenas Aceptadas



Figura 4: Cadenas Rechazadas

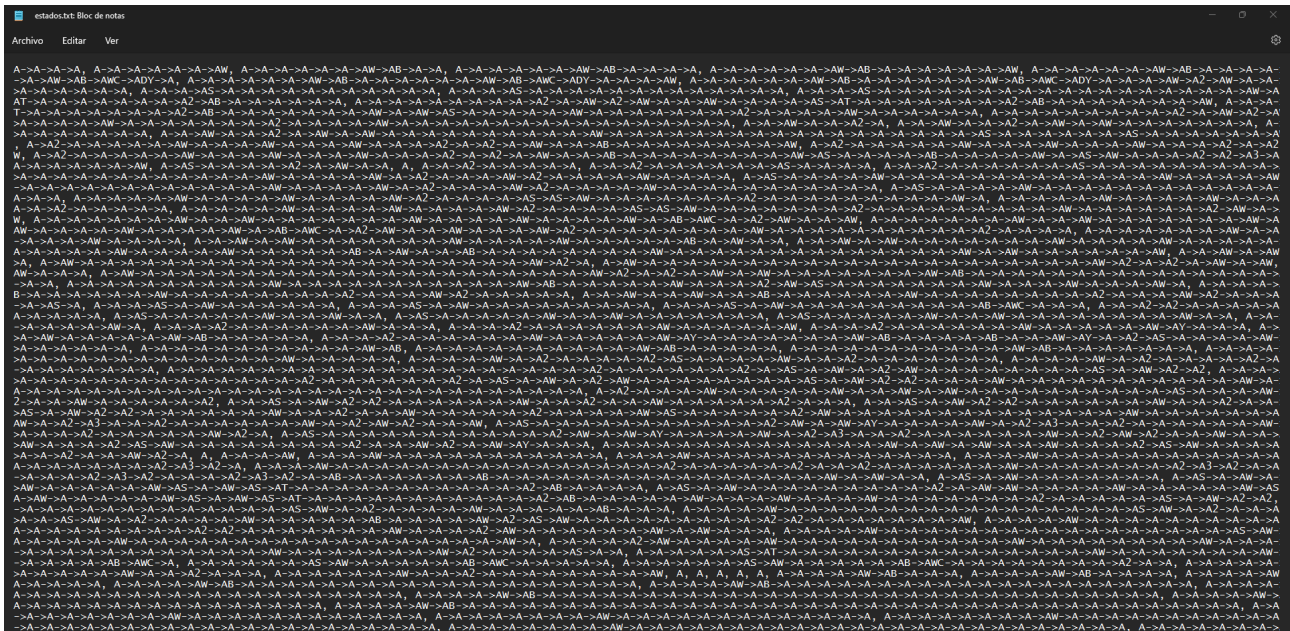


Figura 5: Estados del automata

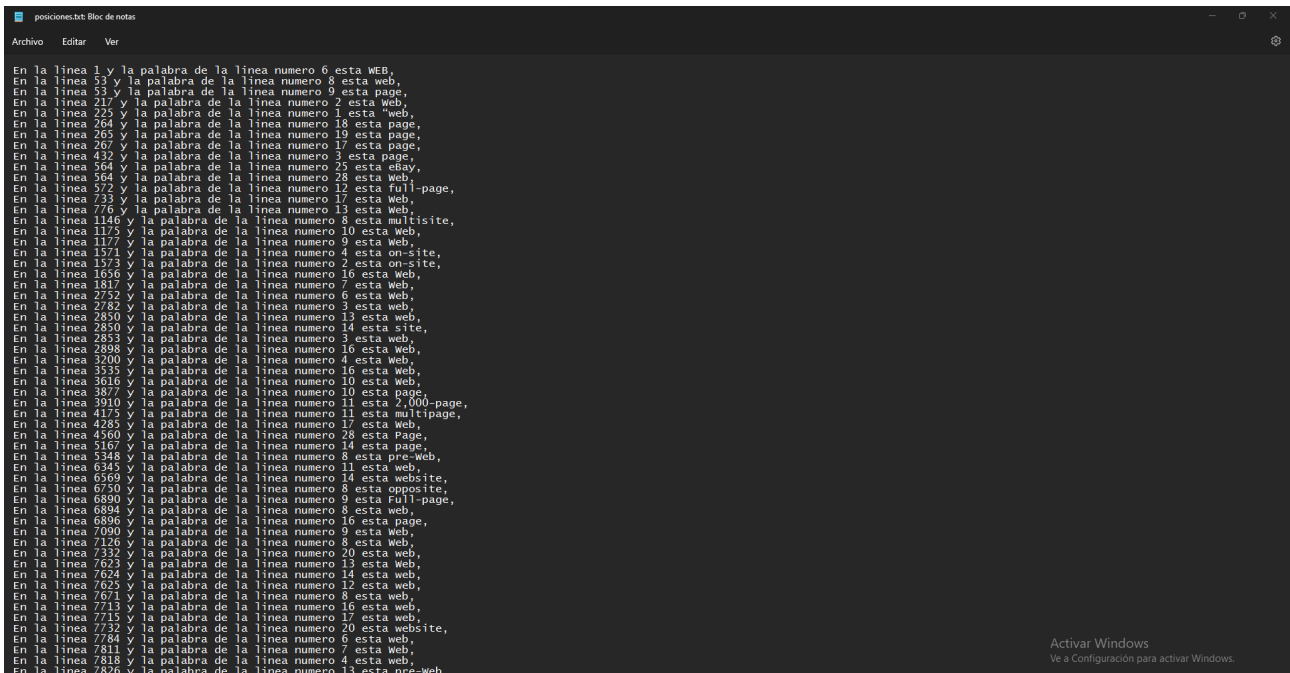
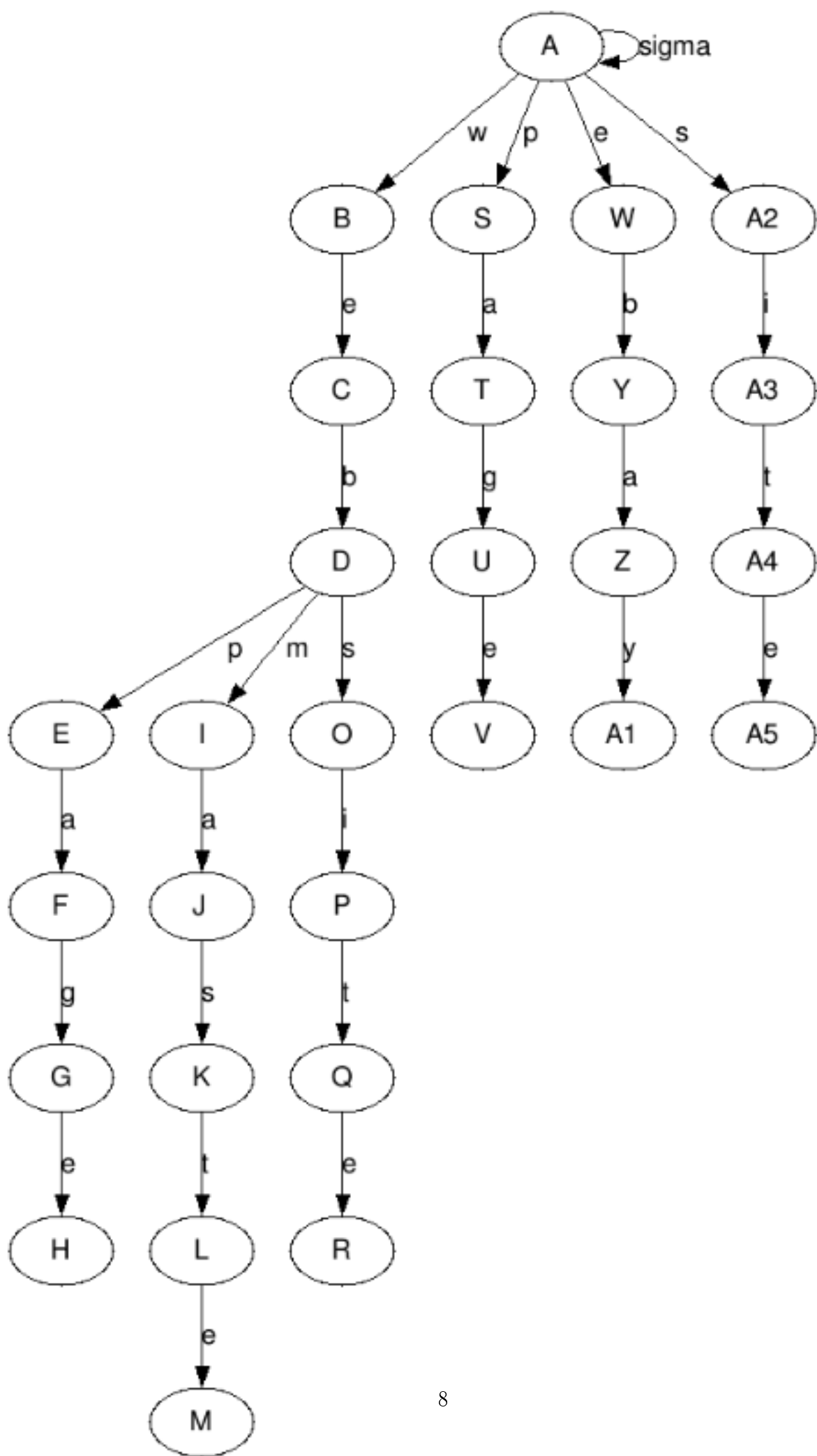


Figura 6: Posicion de las cadenas aceptadas

## 2.2. Grafos

En la siguiente captura, podemos ver lo que seria el grafo NFA:





	w	e	b	p	a	g	m	s	i	t	y	r	z
A	AB	AW	A	AS	A	A	A	Aβ	A	A	A	A	A
AB	AB	ACW	A	AS	A	A	A	Aβ	A	A	A	A	A
AW	AB	AW	AY	AS	A	A	A	Aβ	A	A	A	A	A
AS	AB	AW	A	AS	AT	A	A	Aβ	A	A	A	A	A
Aβ	AB	AW	A	AS	A	A	A	Aβ	Aγ	A	A	A	A
ACW	AB	AW	ADY	AS	A	A	A	Aβ	A	A	A	A	A
AY	AB	AW	A	AS	AZ	A	A	Aβ	A	A	A	A	A
AT	AB	AW	A	AS	A	AU	A	Aβ	A	A	A	A	A
Aγ	AB	AW	A	AS	A	A	A	Aβ	A	Aφ	A	A	A
*ADY	AB	AW	A	AS	AZ	A	AI	Aβ	A	A	A	A	A
AZ	AB	AW	A	AS	A	A	A	Aβ	A	A	Aα	A	A
AU	AB	AWV	A	AS	A	A	A	Aβ	A	A	A	A	A
Aφ	AB	AWπ	A	AS	A	A	A	Aβ	A	A	A	A	A
AI	AB	AW	A	AS	AJ	A	A	Aβ	A	A	A	A	A
*Aα	AB	AW	A	AS	A	A	A	Aβ	A	A	A	A	A
*AWV	AB	AW	AY	AS	A	A	A	Aβ	A	A	A	A	A
*AWπ	AB	AW	AY	AS	A	A	A	Aβ	A	A	A	A	A
AJ	AB	AW	A	AS	A	A	A	Aβk	A	A	A	A	A
Aβk	AB	AW	A	AS	A	A	A	Aβ	A	Aγ	AL	A	A
AL	AB	AWM	A	AS	A	A	A	Aβ	A	A	A	A	A
AWM	AB	AW	AY	AS	A	A	A	Aβ	A	A	A	AN	A
*AN	AB	AW	A	AS	A	A	A	Aβ	A	A	A	A	A

Figura 8: Tabla de Transiciones



Figura 9: DFA

## 3. Código

### 3.1. El automata

Listing 1: Realizacion de la funcionalidad del automata C# .net 6.0

```
1 using System;
2 using System.Linq;
3
4 namespace Programa3
5 {
6     class Programa3
7     {
8         static void Main(string[] args)
9         {
10             //GenerarCadenas();
11             Console.WriteLine("Iniciando");
12             automata();
13         }
14         static void automata()
15         {
16             StreamReader lectura = new StreamReader("D:\\Documentos\\ESCOM
17             \\Teoria Computacional\\DatosP4\\todasCadenas.txt");
18             string linea;
19             int indiLinea = 1;
20             while ((linea = lectura.ReadLine()) != null)
21             {
22                 string[] vs = linea.Split(' ');
23                 string auto = "A", estado = "A";
24                 int indice = 1;
25                 foreach (string v in vs)
26                 {
27                     for (int c = 0; c < v.Length; c++)
28                     {
29                         if (auto.Equals("A") == true)
30                         {
31                             if (v[c].Equals('w') || v[c].Equals('W'))
32                             {
33                                 auto = "AB";
34                                 estado = estado + "->" + auto;
35                             }
36                             else if (v[c].Equals('e') || v[c].Equals('E'))
37                             {
38                                 auto = "AW";
39                                 estado = estado + "->" + auto;
40                             }
41                             else if (v[c].Equals('p') || v[c].Equals('P'))
42                             {
43                                 auto = "AS";
44                                 estado = estado + "->" + auto;
45                             }
46                             else if (v[c].Equals('s') || v[c].Equals('S'))
47                             {
48                                 auto = "A2";
49                                 estado = estado + "->" + auto;
50                             }
51                             else
52                             {
53                                 auto = "A";
54                             }
55                         }
56                     }
57                 }
58             }
59         }
60     }
61 }
```

```

53         estado = estado + "->" + auto;
54     }
55 }
56 else if (auto.Equals("AB") == true)
57 {
58     if (v[c].Equals('w') || v[c].Equals('W'))
59     {
60         auto = "AB";
61         estado = estado + "->" + auto;
62     }
63     else if (v[c].Equals('e') || v[c].Equals('E'))
64     {
65         auto = "AWC";
66         estado = estado + "->" + auto;
67     }
68     else if (v[c].Equals('p') || v[c].Equals('P'))
69     {
70         auto = "AS";
71         estado = estado + "->" + auto;
72     }
73     else if (v[c].Equals('s') || v[c].Equals('S'))
74     {
75         auto = "A2";
76         estado = estado + "->" + auto;
77     }
78     else
79     {
80         auto = "A";
81         estado = estado + "->" + auto;
82     }
83 }
84 else if (auto.Equals("AW") == true)
85 {
86     if (v[c].Equals('w') || v[c].Equals('W'))
87     {
88         auto = "AB";
89         estado = estado + "->" + auto;
90     }
91     else if (v[c].Equals('e') || v[c].Equals('E'))
92     {
93         auto = "AW";
94         estado = estado + "->" + auto;
95     }
96     else if (v[c].Equals('b') || v[c].Equals('B'))
97     {
98         auto = "AY";
99         estado = estado + "->" + auto;
100    }
101    else if (v[c].Equals('p') || v[c].Equals('P'))
102    {
103        auto = "AS";
104        estado = estado + "->" + auto;
105    }
106    else if (v[c].Equals('s') || v[c].Equals('S'))
107    {
108        auto = "A2";
109        estado = estado + "->" + auto;
110    }
111    else
112    {

```

```

113         auto = "A";
114         estado = estado + "->" + auto;
115     }
116 }
117 else if (auto.Equals("AS") == true)
118 {
119     if (v[c].Equals('w') || v[c].Equals('W'))
120     {
121         auto = "AB";
122         estado = estado + "->" + auto;
123     }
124     else if (v[c].Equals('e') || v[c].Equals('E'))
125     {
126         auto = "AW";
127         estado = estado + "->" + auto;
128     }
129     else if (v[c].Equals('p') || v[c].Equals('P'))
130     {
131         auto = "AS";
132         estado = estado + "->" + auto;
133     }
134     else if (v[c].Equals('a') || v[c].Equals('A'))
135     {
136         auto = "AT";
137         estado = estado + "->" + auto;
138     }
139     else if (v[c].Equals('s') || v[c].Equals('S'))
140     {
141         auto = "A2";
142         estado = estado + "->" + auto;
143     }
144     else
145     {
146         auto = "A";
147         estado = estado + "->" + auto;
148     }
149 }
150 else if (auto.Equals("A2") == true)
151 {
152     if (v[c].Equals('w') || v[c].Equals('W'))
153     {
154         auto = "AB";
155         estado = estado + "->" + auto;
156     }
157     else if (v[c].Equals('e') || v[c].Equals('E'))
158     {
159         auto = "AW";
160         estado = estado + "->" + auto;
161     }
162     else if (v[c].Equals('p') || v[c].Equals('P'))
163     {
164         auto = "AS";
165         estado = estado + "->" + auto;
166     }
167     else if (v[c].Equals('s') || v[c].Equals('S'))
168     {
169         auto = "A2";
170         estado = estado + "->" + auto;
171     }
172     else if (v[c].Equals('i') || v[c].Equals('I'))

```

```

173         {
174             auto = "A3";
175             estado = estado + "->" + auto;
176         }
177     else
178     {
179         auto = "A";
180         estado = estado + "->" + auto;
181     }
182 }
183 else if (auto.Equals("AWC") == true)
184 {
185     if (v[c].Equals('w') || v[c].Equals('W'))
186     {
187         auto = "AB";
188         estado = estado + "->" + auto;
189     }
190     else if (v[c].Equals('e') || v[c].Equals('E'))
191     {
192         auto = "AW";
193         estado = estado + "->" + auto;
194     }
195     else if (v[c].Equals('b') || v[c].Equals('B'))
196     {
197         auto = "ADY";
198         estado = estado + "->" + auto;
199     }
200     else if (v[c].Equals('p') || v[c].Equals('P'))
201     {
202         auto = "AS";
203         estado = estado + "->" + auto;
204     }
205     else if (v[c].Equals('s') || v[c].Equals('S'))
206     {
207         auto = "A2";
208         estado = estado + "->" + auto;
209     }
210     else
211     {
212         auto = "A";
213         estado = estado + "->" + auto;
214     }
215 }
216 else if (auto.Equals("AY") == true)
217 {
218     if (v[c].Equals('w') || v[c].Equals('W'))
219     {
220         auto = "AB";
221         estado = estado + "->" + auto;
222     }
223     else if (v[c].Equals('e') || v[c].Equals('E'))
224     {
225         auto = "AW";
226         estado = estado + "->" + auto;
227     }
228     else if (v[c].Equals('p') || v[c].Equals('P'))
229     {
230         auto = "AS";
231         estado = estado + "->" + auto;
232     }

```

```

233     else if (v[c].Equals('a') || v[c].Equals('A'))
234     {
235         auto = "AZ";
236         estado = estado + "->" + auto;
237     }
238     else if (v[c].Equals('s') || v[c].Equals('S'))
239     {
240         auto = "A2";
241         estado = estado + "->" + auto;
242     }
243     else
244     {
245         auto = "A";
246         estado = estado + "->" + auto;
247     }
248 }
249 else if (auto.Equals("AT") == true)
250 {
251     if (v[c].Equals('w') || v[c].Equals('W'))
252     {
253         auto = "AB";
254         estado = estado + "->" + auto;
255     }
256     else if (v[c].Equals('e') || v[c].Equals('E'))
257     {
258         auto = "AW";
259         estado = estado + "->" + auto;
260     }
261     else if (v[c].Equals('p') || v[c].Equals('P'))
262     {
263         auto = "AS";
264         estado = estado + "->" + auto;
265     }
266     else if (v[c].Equals('g') || v[c].Equals('G'))
267     {
268         auto = "AU";
269         estado = estado + "->" + auto;
270     }
271     else if (v[c].Equals('s') || v[c].Equals('S'))
272     {
273         auto = "A2";
274         estado = estado + "->" + auto;
275     }
276     else
277     {
278         auto = "A";
279         estado = estado + "->" + auto;
280     }
281 }
282 else if (auto.Equals("A3") == true)
283 {
284     if (v[c].Equals('w') || v[c].Equals('W'))
285     {
286         auto = "AB";
287         estado = estado + "->" + auto;
288     }
289     else if (v[c].Equals('e') || v[c].Equals('E'))
290     {
291         auto = "AW";
292         estado = estado + "->" + auto;

```

```

293     }
294     else if (v[c].Equals('p') || v[c].Equals('P'))
295     {
296         auto = "AS";
297         estado = estado + "->" + auto;
298     }
299     else if (v[c].Equals('s') || v[c].Equals('S'))
300     {
301         auto = "A2";
302         estado = estado + "->" + auto;
303     }
304     else if (v[c].Equals('t') || v[c].Equals('T'))
305     {
306         auto = "A4";
307         estado = estado + "->" + auto;
308     }
309     else
310     {
311         auto = "A";
312         estado = estado + "->" + auto;
313     }
314 }
315 else if (auto.Equals("ADY") == true)
316 {
317     if (v[c].Equals('w') || v[c].Equals('W'))
318     {
319         auto = "AB";
320         estado = estado + "->" + auto;
321     }
322     else if (v[c].Equals('e') || v[c].Equals('E'))
323     {
324         auto = "AW";
325         estado = estado + "->" + auto;
326     }
327     else if (v[c].Equals('p') || v[c].Equals('P'))
328     {
329         auto = "AS";
330         estado = estado + "->" + auto;
331     }
332     else if (v[c].Equals('a') || v[c].Equals('A'))
333     {
334         auto = "AZ";
335         estado = estado + "->" + auto;
336     }
337     else if (v[c].Equals('m') || v[c].Equals('M'))
338     {
339         auto = "AI";
340         estado = estado + "->" + auto;
341     }
342     else if (v[c].Equals('s') || v[c].Equals('S'))
343     {
344         auto = "A2";
345         estado = estado + "->" + auto;
346     }
347     else
348     {
349         auto = "A";
350         estado = estado + "->" + auto;
351     }
352 }

```



```

353     else if (auto.Equals("AZ") == true)
354     {
355         if (v[c].Equals('w') || v[c].Equals('W'))
356         {
357             auto = "AB";
358             estado = estado + "->" + auto;
359         }
360         else if (v[c].Equals('e') || v[c].Equals('E'))
361         {
362             auto = "AW";
363             estado = estado + "->" + auto;
364         }
365         else if (v[c].Equals('p') || v[c].Equals('P'))
366         {
367             auto = "AS";
368             estado = estado + "->" + auto;
369         }
370         else if (v[c].Equals('s') || v[c].Equals('S'))
371         {
372             auto = "A2";
373             estado = estado + "->" + auto;
374         }
375         else if (v[c].Equals('y') || v[c].Equals('Y'))
376         {
377             auto = "A1";
378             estado = estado + "->" + auto;
379         }
380         else
381         {
382             auto = "A";
383             estado = estado + "->" + auto;
384         }
385     }
386     else if (auto.Equals("AU") == true)
387     {
388         if (v[c].Equals('w') || v[c].Equals('W'))
389         {
390             auto = "AB";
391             estado = estado + "->" + auto;
392         }
393         else if (v[c].Equals('e') || v[c].Equals('E'))
394         {
395             auto = "AWV";
396             estado = estado + "->" + auto;
397         }
398         else if (v[c].Equals('p') || v[c].Equals('P'))
399         {
400             auto = "AS";
401             estado = estado + "->" + auto;
402         }
403         else if (v[c].Equals('s') || v[c].Equals('S'))
404         {
405             auto = "A2";
406             estado = estado + "->" + auto;
407         }
408         else
409         {
410             auto = "A";
411             estado = estado + "->" + auto;
412         }

```

```

413 }
414 else if (auto.Equals("A4") == true)
415 {
416     if (v[c].Equals('w') || v[c].Equals('W'))
417     {
418         auto = "AB";
419         estado = estado + "->" + auto;
420     }
421     else if (v[c].Equals('e') || v[c].Equals('E'))
422     {
423         auto = "AW5";
424         estado = estado + "->" + auto;
425     }
426     else if (v[c].Equals('p') || v[c].Equals('P'))
427     {
428         auto = "AS";
429         estado = estado + "->" + auto;
430     }
431     else if (v[c].Equals('s') || v[c].Equals('S'))
432     {
433         auto = "A2";
434         estado = estado + "->" + auto;
435     }
436     else
437     {
438         auto = "A";
439         estado = estado + "->" + auto;
440     }
441 }
442 else if (auto.Equals("AI") == true)
443 {
444     if (v[c].Equals('w') || v[c].Equals('W'))
445     {
446         auto = "AB";
447         estado = estado + "->" + auto;
448     }
449     else if (v[c].Equals('e') || v[c].Equals('E'))
450     {
451         auto = "AW";
452         estado = estado + "->" + auto;
453     }
454     else if (v[c].Equals('p') || v[c].Equals('P'))
455     {
456         auto = "AS";
457         estado = estado + "->" + auto;
458     }
459     else if (v[c].Equals('a') || v[c].Equals('A'))
460     {
461         auto = "AJ";
462         estado = estado + "->" + auto;
463     }
464
465     else if (v[c].Equals('s') || v[c].Equals('S'))
466     {
467         auto = "A2";
468         estado = estado + "->" + auto;
469     }
470     else
471     {
472         auto = "A";

```

```

473         estado = estado + "->" + auto;
474     }
475 }
476 else if (auto.Equals("A1") == true)
477 {
478     if (v[c].Equals('w') || v[c].Equals('W'))
479     {
480         auto = "AB";
481         estado = estado + "->" + auto;
482     }
483     else if (v[c].Equals('e') || v[c].Equals('E'))
484     {
485         auto = "AW";
486         estado = estado + "->" + auto;
487     }
488     else if (v[c].Equals('p') || v[c].Equals('P'))
489     {
490         auto = "AS";
491         estado = estado + "->" + auto;
492     }
493     else if (v[c].Equals('s') || v[c].Equals('S'))
494     {
495         auto = "A2";
496         estado = estado + "->" + auto;
497     }
498     else
499     {
500         auto = "A";
501         estado = estado + "->" + auto;
502     }
503 }
504 else if (auto.Equals("AWV") == true)
505 {
506     if (v[c].Equals('w') || v[c].Equals('W'))
507     {
508         auto = "AB";
509         estado = estado + "->" + auto;
510     }
511     else if (v[c].Equals('e') || v[c].Equals('E'))
512     {
513         auto = "AW";
514         estado = estado + "->" + auto;
515     }
516     else if (v[c].Equals('b') || v[c].Equals('B'))
517     {
518         auto = "AY";
519         estado = estado + "->" + auto;
520     }
521     else if (v[c].Equals('p') || v[c].Equals('P'))
522     {
523         auto = "AS";
524         estado = estado + "->" + auto;
525     }
526     else if (v[c].Equals('a') || v[c].Equals('A'))
527     {
528         auto = "AY";
529         estado = estado + "->" + auto;
530     }
531     else if (v[c].Equals('s') || v[c].Equals('S'))
532     {

```

```

533         auto = "A2";
534         estado = estado + "->" + auto;
535     }
536     else
537     {
538         auto = "A";
539         estado = estado + "->" + auto;
540     }
541 }
542 else if (auto.Equals("AW5") == true)
543 {
544     if (v[c].Equals('w') || v[c].Equals('W'))
545     {
546         auto = "AB";
547         estado = estado + "->" + auto;
548     }
549     else if (v[c].Equals('e') || v[c].Equals('E'))
550     {
551         auto = "AW";
552         estado = estado + "->" + auto;
553     }
554     else if (v[c].Equals('b') || v[c].Equals('B'))
555     {
556         auto = "AY";
557         estado = estado + "->" + auto;
558     }
559     else if (v[c].Equals('p') || v[c].Equals('P'))
560     {
561         auto = "AS";
562         estado = estado + "->" + auto;
563     }
564     else if (v[c].Equals('s') || v[c].Equals('S'))
565     {
566         auto = "A2";
567         estado = estado + "->" + auto;
568     }
569     else
570     {
571         auto = "A";
572         estado = estado + "->" + auto;
573     }
574 }
575 else if (auto.Equals("AJ") == true)
576 {
577     if (v[c].Equals('w') || v[c].Equals('W'))
578     {
579         auto = "AB";
580         estado = estado + "->" + auto;
581     }
582     else if (v[c].Equals('e') || v[c].Equals('E'))
583     {
584         auto = "AW";
585         estado = estado + "->" + auto;
586     }
587     else if (v[c].Equals('p') || v[c].Equals('P'))
588     {
589         auto = "AS";
590         estado = estado + "->" + auto;
591     }
592     else if (v[c].Equals('s') || v[c].Equals('S'))

```

```

593     {
594         auto = "A2K";
595         estado = estado + "->" + auto;
596     }
597     else
598     {
599         auto = "A";
600         estado = estado + "->" + auto;
601     }
602 }
603 else if (auto.Equals("A2K") == true)
604 {
605     if (v[c].Equals('w') || v[c].Equals('W'))
606     {
607         auto = "AB";
608         estado = estado + "->" + auto;
609     }
610     else if (v[c].Equals('e') || v[c].Equals('E'))
611     {
612         auto = "AW";
613         estado = estado + "->" + auto;
614     }
615     else if (v[c].Equals('p') || v[c].Equals('P'))
616     {
617         auto = "AS";
618         estado = estado + "->" + auto;
619     }
620     else if (v[c].Equals('s') || v[c].Equals('S'))
621     {
622         auto = "A2";
623         estado = estado + "->" + auto;
624     }
625     else if (v[c].Equals('i') || v[c].Equals('I'))
626     {
627         auto = "A3";
628         estado = estado + "->" + auto;
629     }
630     else if (v[c].Equals('t') || v[c].Equals('T'))
631     {
632         auto = "AL";
633         estado = estado + "->" + auto;
634     }
635     else
636     {
637         auto = "A";
638         estado = estado + "->" + auto;
639     }
640 }
641 else if (auto.Equals("AL") == true)
642 {
643     if (v[c].Equals('w') || v[c].Equals('W'))
644     {
645         auto = "AB";
646         estado = estado + "->" + auto;
647     }
648     else if (v[c].Equals('e') || v[c].Equals('E'))
649     {
650         auto = "AWM";
651         estado = estado + "->" + auto;
652     }

```

```

653     else if (v[c].Equals('p') || v[c].Equals('P'))
654     {
655         auto = "AS";
656         estado = estado + "->" + auto;
657     }
658     else if (v[c].Equals('s') || v[c].Equals('S'))
659     {
660         auto = "A2";
661         estado = estado + "->" + auto;
662     }
663     else
664     {
665         auto = "A";
666         estado = estado + "->" + auto;
667     }
668 }
669 else if (auto.Equals("AWM") == true)
670 {
671     if (v[c].Equals('w') || v[c].Equals('W'))
672     {
673         auto = "AB";
674         estado = estado + "->" + auto;
675     }
676     else if (v[c].Equals('e') || v[c].Equals('E'))
677     {
678         auto = "AW";
679         estado = estado + "->" + auto;
680     }
681     else if (v[c].Equals('b') || v[c].Equals('B'))
682     {
683         auto = "AY";
684         estado = estado + "->" + auto;
685     }
686     else if (v[c].Equals('p') || v[c].Equals('P'))
687     {
688         auto = "AS";
689         estado = estado + "->" + auto;
690     }
691     else if (v[c].Equals('s') || v[c].Equals('S'))
692     {
693         auto = "A2";
694         estado = estado + "->" + auto;
695     }
696     else if (v[c].Equals('r') || v[c].Equals('R'))
697     {
698         auto = "AN";
699         estado = estado + "->" + auto;
700     }
701     else
702     {
703         auto = "A";
704         estado = estado + "->" + auto;
705     }
706 }
707 else if (auto.Equals("AN") == true)
708 {
709     if (v[c].Equals('w') || v[c].Equals('W'))
710     {
711         auto = "AB";
712         estado = estado + "->" + auto;

```

```

713     }
714     else if (v[c].Equals('e') || v[c].Equals('E'))
715     {
716         auto = "AW";
717         estado = estado + "->" + auto;
718     }
719     else if (v[c].Equals('p') || v[c].Equals('P'))
720     {
721         auto = "AS";
722         estado = estado + "->" + auto;
723     }
724     else if (v[c].Equals('s') || v[c].Equals('S'))
725     {
726         auto = "A2";
727         estado = estado + "->" + auto;
728     }
729     else
730     {
731         auto = "A";
732         estado = estado + "->" + auto;
733     }
734     }
735 }
736
737 if (auto.Equals("ADY") || auto.Equals("A1") || auto.
Equals("AWV") || auto.Equals("AW5") || auto.Equals
("AN"))
738 {
739     File.AppendAllText("D:\\Documentos\\ESCOM\\Teoria
Computacional\\DatosP4\\posiciones.txt", $"En
la linea {indiLinea} y la palabra de la linea
numero {indice} esta {v}, \n");
740     File.AppendAllText("D:\\Documentos\\ESCOM\\Teoria
Computacional\\DatosP4\\CadenasAceptadas.txt",
v + ", ");
741     File.AppendAllText("D:\\Documentos\\ESCOM\\Teoria
Computacional\\DatosP4\\estados.txt", estado +
", ");
742 }
743 else
744 {
745
746     File.AppendAllText("D:\\Documentos\\ESCOM\\Teoria
Computacional\\DatosP4\\CadenasRechazadas.txt"
, v + ", ");
747     File.AppendAllText("D:\\Documentos\\ESCOM\\Teoria
Computacional\\DatosP4\\estados.txt", estado +
", ");
748
749     }
750     indice++;
751 }
752     indiLinea++;
753 }
754 lectura.Close();
755 Console.Beep(37, 5000);
756 Console.WriteLine("Termino");
757 }
758
759 }

```

## 3.2. Realizacion del Grafo por medio de ReactJS

Listing 2: Creacion del Grafo en js, index.js

```

1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5
6 const root = ReactDOM.createRoot(document.getElementById('root'));
7 root.render(
8   <React.StrictMode>
9     <App />
10  </React.StrictMode>
11 );

```

Listing 3: Creacion del Grafo en js, App.js

```

1 import './App.css';
2 import { Graphviz } from 'graphviz-react';
3 function App() {
4   return (
5     <div className="App">
6       <h1>Practica 4 grafo dibujado</h1>
7       <Graphviz
8         options={
9           {
10             rankdir: 'LR',
11             ranksep: '1',
12             nodesep: '1',
13             splines: 'polyline',
14             overlap: 'false',
15             height: 720,
16             width: 1280,
17             node: {
18               shape: 'circle',
19               style: 'filled',
20               fillcolor: '#006699',
21               fontname: 'Arial',
22               fontsize: '12',
23               fontcolor: 'white',
24               margin: '0.1',
25               width: '0.5',
26               height: '0.5',
27             },
28             edge: {
29               arrowhead: 'open',
30               arrowsize: '0.5',
31               fontname: 'Arial',
32               fontsize: '12',
33               fontcolor: 'black',
34               color: 'black',
35               style: 'solid',
36               dir: 'forward',
37             }

```



```

38     }
39 }
40 dot = {
41 digraph {
42     //Para A
43     A -> AB[label="w",weight="w"];
44     A -> AW[label="e",weight="e"];
45     A -> AS[label="p",weight="p"];
46     A -> A2[label="s",weight="s"];
47     A -> A[label="sigma, r, y, t, i, m, g, a, b",weight="sigma"];
48     //Para AB
49     AB -> AB[label="w",weight="w"];
50     AB -> ACW[label="e",weight="e"];
51     AB -> AS[label="p",weight="p"];
52     AB -> A2[label="s",weight="s"];
53     AB -> A[label="sigma, r, y, t, i, m, g, a, b",weight="sigma"];
54     //Para AW
55     AW -> AB[label="w",weight="w"];
56     AW -> AW[label="e",weight="e"];
57     AW -> AY[label="b",weight="b"];
58     AW -> AS[label="p",weight="p"];
59     AW -> A2[label="s",weight="s"];
60     AW -> A[label="sigma, r, y, t, i, m, g, a",weight="sigma"];
61     //Para AS
62     AS -> AB[label="w",weight="w"];
63     AS -> AW[label="e",weight="e"];
64     AS -> AS[label="p",weight="p"];
65     AS -> AT[label="a",weight="a"];
66     AS -> A2[label="s",weight="s"];
67     AS -> A[label="sigma, r, y, t, i, m, g, b",weight="sigma"];
68     //Para A2
69     A2 -> AB[label="w",weight="w"];
70     A2 -> AW[label="e",weight="e"];
71     A2 -> AS[label="p",weight="p"];
72     A2 -> A2[label="s",weight="s"];
73     A2 -> A3[label="i",weight="i"];
74     A2 -> A[label="sigma, r, y, t, m, g, a, b",weight="sigma"];
75     //Para ACW
76     ACW -> AB[label="w",weight="w"];
77     ACW -> AW[label="e",weight="e"];
78     ACW -> ADY[label="b",weight="b"];
79     ACW -> AS[label="p",weight="p"];
80     ACW -> A2[label="s",weight="s"];
81     ACW -> A[label="sigma, r, y, t, i, m, g, a",weight="sigma"];
82     //Para AY
83     AY -> AB[label="w",weight="w"];
84     AY -> AW[label="e",weight="e"];
85     AY -> AS[label="p",weight="p"];
86     AY -> AZ[label="a",weight="a"];
87     AY -> A2[label="s",weight="s"];
88     AY -> A[label="sigma, r, y, t, i, m, g, b",weight="sigma"];
89     //Para AT
90     AT -> AB[label="w",weight="w"];
91     AT -> AW[label="e",weight="e"];
92     AT -> AS[label="p",weight="p"];
93     AT -> AU[label="g",weight="g"];
94     AT -> A2[label="s",weight="s"];
95     AT -> A[label="sigma, r, y, t, i, m, a, b",weight="sigma"];
96     //Para A3
97     A3 -> AB[label="w",weight="w"];

```

```

98     A3 -> AW[label="e",weight="e"];
99     A3 -> AS[label="p",weight="p"];
100    A3 -> A2[label="s",weight="s"];
101    A3 -> A4[label="t",weight="t"];
102    A3 -> A[label="sigma, r, y, i, m, g, b",weight="sigma"];
103    //Para ADY
104    ADY -> AB[label="w",weight="w"];
105    ADY -> AW[label="e",weight="e"];
106    ADY -> AS[label="p",weight="p"];
107    ADY -> AZ[label="a",weight="a"];
108    ADY -> AI[label="m",weight="m"];
109    ADY -> A2[label="s",weight="s"];
110    ADY -> A[label="sigma, r, y, t, i, g, b",weight="sigma"];
111    //Para AZ
112    AZ -> AB[label="w",weight="w"];
113    AZ -> AW[label="e",weight="e"];
114    AZ -> A[label="b",weight="b"];
115    AZ -> AS[label="p",weight="p"];
116    AZ -> A2[label="s",weight="s"];
117    AZ -> A1[label="y",weight="y"];
118    AZ -> A[label="sigma, r, t, i, m, g, s, b",weight="sigma"];
119    //Para AU
120    AU -> AB[label="w",weight="w"];
121    AU -> AWV[label="e",weight="e"];
122    AU -> AS[label="p",weight="p"];
123    AU -> A2[label="s",weight="s"];
124    AU -> A[label="sigma, r, y, t, i, m, g, s, b",weight="sigma"];
125    //Para A4
126    A4 -> AB[label="w",weight="w"];
127    A4 -> AW5[label="e",weight="e"];
128    A4 -> AS[label="p",weight="p"];
129    A4 -> A2[label="s",weight="s"];
130    A4 -> A[label="sigma, r, y, t, i, m, g, a, b",weight="sigma"];
131    //Para AI
132    AI -> AB[label="w",weight="w"];
133    AI -> AW[label="e",weight="e"];
134    AI -> AS[label="p",weight="p"];
135    AI -> AJ[label="a",weight="a"];
136    AI -> A2[label="s",weight="s"];
137    AI -> A[label="sigma, r, y, t, i, m, g, b",weight="sigma"];
138    //Para A1
139    A1 -> AB[label="w",weight="w"];
140    A1 -> AW[label="e",weight="e"];
141    A1 -> AS[label="p",weight="p"];
142    A1 -> A2[label="s",weight="s"];
143    A1 -> A[label="sigma, r, y, t, i, m, g, s, b",weight="sigma"];
144    //Para AWV
145    AWV -> AB[label="w",weight="w"];
146    AWV -> AW[label="e",weight="e"];
147    AWV -> AY[label="b",weight="b"];
148    AWV -> AS[label="p",weight="p"];
149    AWV -> A2[label="s",weight="s"];
150    AWV -> A[label="sigma, r, y, t, i, m, g, a",weight="sigma"];
151    //Para AW5
152    AW5 -> AB[label="w",weight="w"];
153    AW5 -> AW[label="e",weight="e"];
154    AW5 -> AY[label="b",weight="b"];
155    AW5 -> AS[label="p",weight="p"];
156    AW5 -> A2[label="s",weight="s"];
157    AW5 -> A[label="sigma, r, y, t, i, m, g, a",weight="sigma"];

```

```

158 //Para AJ
159 AJ -> AB[label="w",weight="w"];
160 AJ -> AW[label="e",weight="e"];
161 AJ -> AS[label="p",weight="p"];
162 AJ -> A2K[label="s",weight="s"];
163 AJ -> A[label="sigma, r, y, t, i, m, g, a, b",weight="sigma"];
164 //Para A2K
165 A2K -> AB[label="w",weight="w"];
166 A2K -> AW[label="e",weight="e"];
167 A2K -> AS[label="p",weight="p"];
168 A2K -> A2[label="s",weight="s"];
169 A2K -> A3[label="t",weight="t"];
170 A2K -> AL[label="y",weight="y"];
171 A2K -> A[label="sigma, r, i, m, g, a, b",weight="sigma"];
172 //Para AL
173 AL -> AB[label="w",weight="w"];
174 AL -> AWM[label="e",weight="e"];
175 AL -> AS[label="p",weight="p"];
176 AL -> AZ[label="a",weight="a"];
177 AL -> A2[label="s",weight="s"];
178 AL -> A[label="sigma, r, y, t, i, m, g, b",weight="sigma"];
179 //Para AWM
180 AWM -> AB[label="w",weight="w"];
181 AWM -> AW[label="e",weight="e"];
182 AWM -> AY[label="b",weight="b"];
183 AWM -> AS[label="p",weight="p"];
184 AWM -> A2[label="s",weight="s"];
185 AWM -> AN[label="r",weight="r"];
186 AWM -> A[label="sigma, y, t, i, m, g, a",weight="sigma"];
187 //Para AN
188 AN -> AB[label="w",weight="w"];
189 AN -> AWM[label="e",weight="e"];
190 AN -> AS[label="p",weight="p"];
191 AN -> A2[label="s",weight="s"];
192 AN -> A[label="sigma, r, y, t, i, m, g, a, b",weight="sigma"];
193 }' } />
194
195 </div>
196 );
197 }
198
199 export default App;

```

---