

Lenguajes y Autómatas I

Juan Pablo Rosas Baldazo

Reporte de proyecto 2

Alfonso Guerrero Contreras

de control16480211

07/10/2018

Tabla de contenido

Validar formato de un correo electrónico mediante método propio	4
Descripción	4
Pruebas	5
Pruebas de longitud de la cadena	5
Pruebas de cantidad de iteraciones	6
Validar formato de correo electrónico mediante una expresión regular.	7
Descripción	7
Pruebas	8
Pruebas de longitud de cadena	8
Pruebas de cantidad de iteraciones	9
Comparación de resultados de ambos verificadores de correo electrónico	10
Longitud de cadena	10
Numero de iteraciones	10
Algoritmo KMP	11
Descripción	11
Conclusión	12

Introducción

Las expresiones regulares son útiles para reconocer los patrones dentro de cadenas de texto.

En este documento se verán tres programas, todos con similitudes, ya que trabajan con cadenas de caracteres.

Dos son para reconocer si una dirección de correo electrónico dada esta estructurada de forma correcta. Uno de los programas será enteramente propio, mientras que el otro implementará el uso de una expresión regular para verificar correos. Incluye una prueba para verificar cual es más eficaz.

El último programa será una implementación del Algoritmo KMP(Knuth-Morris-Pratt) y probar que funciona.

Validar formato de un correo electrónico mediante método propio

Descripción

Primero se ingresa el correo que se va a verificar mediante un input que guardaremos en una variable, para después mandarla a la función que lo verificara

```
correo = input("Inserta La direccion de correo electronico")  
verificarCorreo(correo)
```

La verificación en si es muy simple, primero verificamos que todos los caracteres son en letras minúsculas

```
if correo.islower():
```

de lo contrario es una dirección de correo invalido

```
else:  
    print("correo falso")
```

a continuación, verificamos que exista una arroba (@)

```
if "@" in correo:
```

si no existe el correo es invalido

```
else:  
    print("correo falso")
```

si existe separamos la cadena con el @ como marca y las guardamos en variables

```
usuario, dominio = correo.split("@")
```

a continuación, revisamos que en la cadena que contiene el dominio exista un punto

```
if "." in dominio:
```

si existe entonces el correo esta estructurado de forma correcta, de lo contrario es un correo no valido

```
if "." in dominio:  
    print ("Correo valido")  
else:  
    print ("Correo falso")
```

Pruebas

El equipo en que se harán las pruebas es un laptop HP Pavilon x360

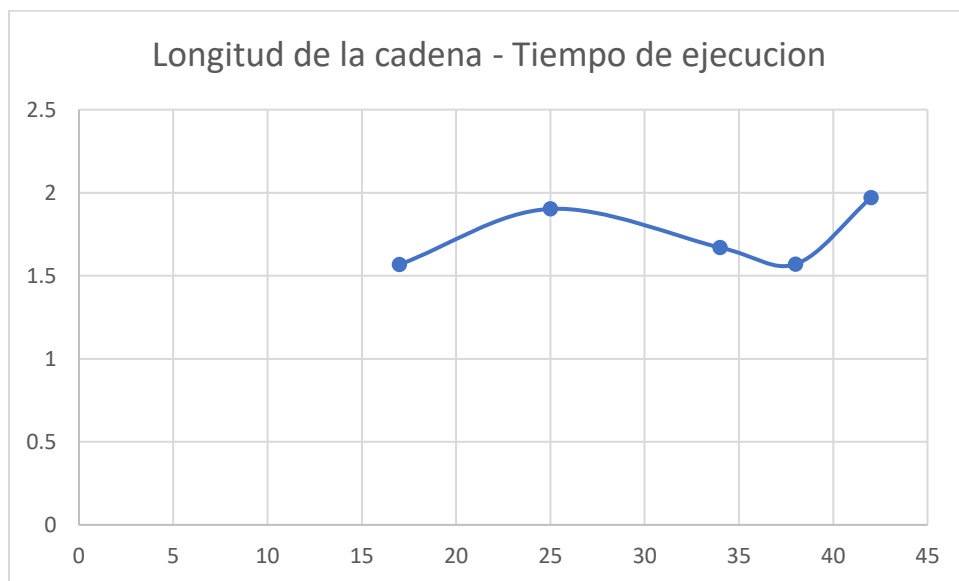
- Procesador AMD A8-6410 APU 2.00GHz
- Graficos Rendon R5 Graphics
- Windows 10 Home de 64 bits
- 4 GB de memoria RAM

Para probar el tiempo de ejecución de la aplicación, usare diferente longitud en las cadenas.

Pruebas de longitud de la cadena

Las pruebas de longitud.

Cadena	longitud	Resultado
alfonso@gmail.com	17	1.567763090133667
alfonsoguerrero@gmail.com	25	1.9024724960327148
alfonsoguerrerocontreras@gmail.com	34	1.669835090637207
Alfonsoguerrerocontreras1234@gmail.com	38	1.5697710514068604
Alfonsoguerrerocontreras12345678@gmail.com	42	1.9694550037384033

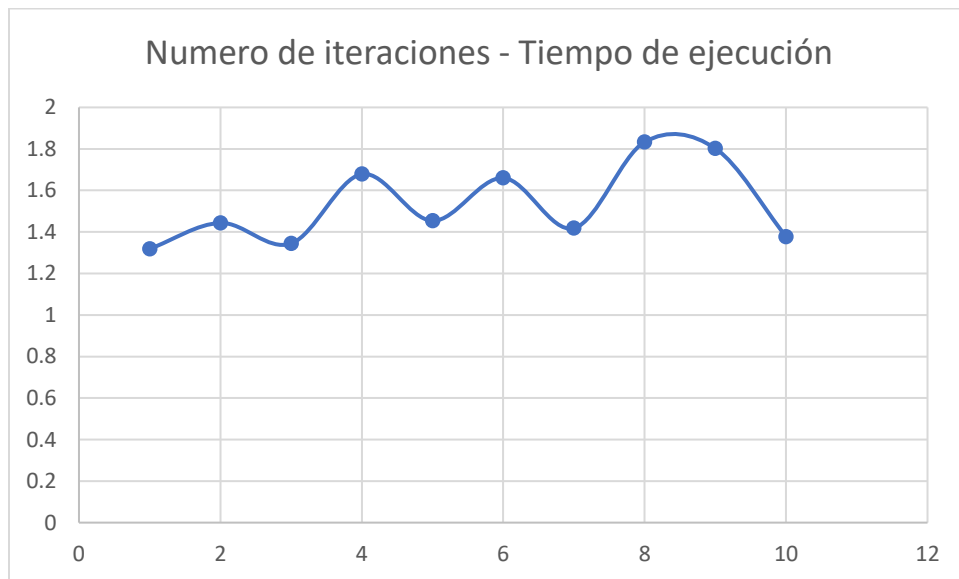


Como podemos observar, la longitud de la cadena no es relevante, ya que la variación en segundos es muy baja.

Pruebas de cantidad de iteraciones

Esta vez tomaremos uno de los casos anteriores y lo repetiremos 10 veces para observar las diferencias entre el tiempo de ejecución con los mismos parámetros.

Cadena	longitud	Iteración	Resultado
alfonso@gmail.com	17	1	1.3185153007507324
alfonso@gmail.com	17	2	1.4434807300567627
alfonso@gmail.com	17	3	1.3442604541778564
alfonso@gmail.com	17	4	1.6786532402038574
alfonso@gmail.com	17	5	1.4536638259887695
alfonso@gmail.com	17	6	1.6602065563201904
alfonso@gmail.com	17	7	1.4181671142578125
alfonso@gmail.com	17	8	1.8332490921020508
alfonso@gmail.com	17	9	1.8020215034484863
alfonso@gmail.com	17	10	1.3776590824127197



- En esta prueba podemos observar que, aun cuando los parámetros son iguales, el procesador los ejecutara a diferente velocidad dependiendo de que este haciendo en ese momento.

Validar formato de correo electrónico mediante una expresión regular.

Descripción

Primero se ingresa el correo que se va a verificar mediante un input que guardaremos en una variable, para después mandarla a la función que lo verificara

```
correo = input("Inserta La direccion de correo electronico")  
verificarCorreo(correo)
```

en validar correo tenemos la expresión regular que nos permitirá revisar si la estructura del correo es adecuada

```
'^[a-z0-9_\-\.\.])+@[a-z0-9_\-\.\.]+\.[a-z]{2,15}$'
```

Verificamos si la cadena enviada cuenta con estos requisitos colocándola en un if, de ser así entonces es una dirección de correo válida

```
if re.match('^[a-z0-9_\-\.\.])+@[a-z0-9_\-\.\.]+\.[a-z]{2,15}$', correo.lower()):  
    print ("Correo correcto")
```

de lo contrario es una dirección de correo electrónico no válida

```
else:  
    print ("Correo incorrecto")
```

Pruebas

El equipo en que se harán las pruebas es un laptop HP Pavilon x360

- Procesador AMD A8-6410 APU 2.00GHz
- Graficos Rendon R5 Graphics
- Windows 10 Home de 64 bits
- 4 GB de memoria RAM

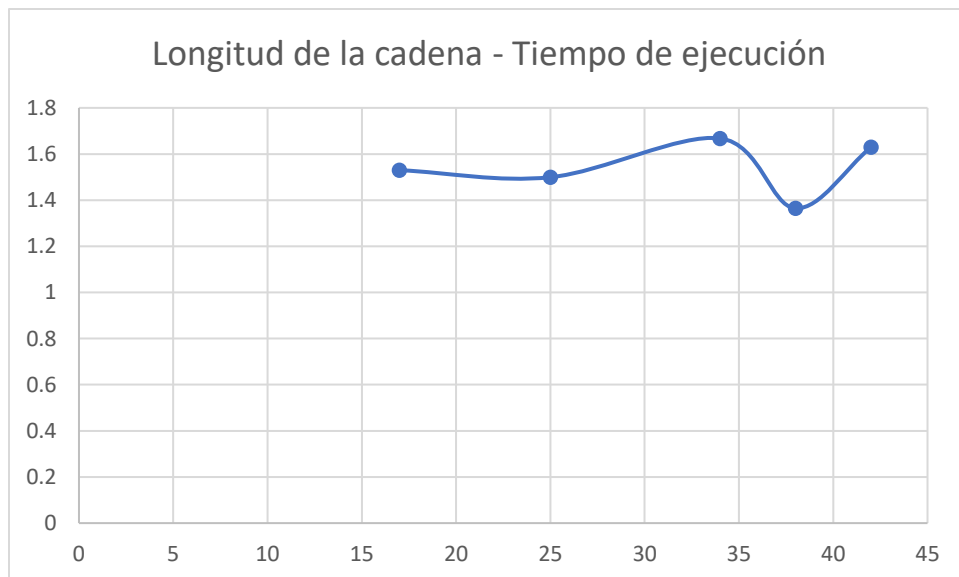
Para probar el tiempo de ejecución de la aplicación, usare diferente longitud en las cadenas.

Pruebas de longitud de cadena

Primero probaremos como reacciona el sistema cuando se le ingresan cada vez más datos.

Cadena	longitud	Resultado
alfonso@gmail.com	17	1.5301012992858887
alfonsoguerrero@gmail.com	25	1.4993832111358643
alfonsoguerrerocontreras@gmail.com	34	1.6678433418273926
Alfonsoguerrerocontreras1234@gmail.com	38	1.3646585941314697
Alfonsoguerrerocontreras12345678@gmail.com	42	1.629441499710083

- Al incrementar la longitud de la cadena (alfabeto), se nota un ligero incremento en el tiempo de ejecución.



Pruebas de cantidad de iteraciones

Esta vez tomaremos uno de los cazos anteriores y lo repetiremos 10 veces para observar las diferencias entre el tiempo de ejecución con los mismos parámetros.

Cadena	longitud	Iteración	Resultado
alfonso@gmail.com	17	1	1.4516689777374268
alfonso@gmail.com	17	2	1.0698623657226562
alfonso@gmail.com	17	3	1.3995728492736816
alfonso@gmail.com	17	4	0.9732472896575928
alfonso@gmail.com	17	5	1.052912712097168
alfonso@gmail.com	17	6	1.180114507675171
alfonso@gmail.com	17	7	1.1998951435089111
alfonso@gmail.com	17	8	1.0866975784301758
alfonso@gmail.com	17	9	0.995703935623169
alfonso@gmail.com	17	10	1.249755859375



- Al igual que en la prueba de iteraciones de la otra función, podemos observar que, aun cuando los parámetros son iguales, el procesador los ejecutara a diferente velocidad dependiendo de que este haciendo en ese momento.

Comparación de resultados de ambos verificadores de correo electrónico

Longitud de cadena

Lenguaje regular		Propio	
longitud	Resultado	longitud	Resultado
17	1.5301012992858887	17	1.567763090133667
25	1.4993832111358643	25	1.9024724960327148
34	1.6678433418273926	34	1.669835090637207
38	1.3646585941314697	38	1.5697710514068604
42	1.629441499710083	42	1.9694550037384033

Como podemos observar, los tiempos, aunque muy similares, son mayores en el programa que realice por mi cuenta que en el que utiliza el algoritmo regular.

Numero de iteraciones

Lenguaje regular		Propio	
Iteración	Resultado	Iteración	Resultado
1	1.4516689777374268	1	1.3185153007507324
2	1.0698623657226562	2	1.4434807300567627
3	1.3995728492736816	3	1.3442604541778564
4	0.9732472896575928	4	1.6786532402038574
5	1.052912712097168	5	1.4536638259887695
6	1.180114507675171	6	1.6602065563201904
7	1.1998951435089111	7	1.4181671142578125
8	1.0866975784301758	8	1.8332490921020508
9	0.995703935623169	9	1.8020215034484863
10	1.249755859375	10	1.3776590824127197

De nuevo podemos observar que el tiempo de ejecución al repetir la misma cadena múltiples veces es menor en la aplicación que usa un lenguaje regular.

Algoritmo KMP

Descripción

Conclusión

El uso de expresiones regulares vivió mucho más corto el código para la validación de correo electrónico. También redujo el tiempo de ejecución, incluso si solo fue por unos 0.1 segundos aproximadamente.

Trabajar con expresiones regulares ya diseñadas sin duda facilita las cosas, sin embargo, en programas simples, encontrar dicha expresión regular puede tomar mas tiempo que hacerlo sin estas.

En las aplicaciones complicadas y en las que debemos cuidar cada recurso que tenemos es donde brilla el uso de las expresiones regulares.

Fuentes

<https://docs.python.org/3/library/>

<https://www.lawebdelprogramador.com/codigo/Python/2040-Validar-cuenta-de-correo.html>