

## ***Algoritmo Knuth–Morris–Pratt***

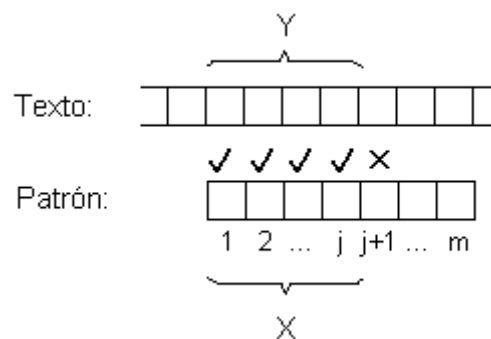
El algoritmo para búsqueda de cadenas Knuth–Morris–Pratt (KMP) busca la aparición de una palabra  $P$  dentro de una “cadena de texto” principal  $C$ , empleando la simple observación de que cuando no sucede una coincidencia, la palabra misma contiene suficiente información para determinar cuándo puede ocurrir la siguiente coincidencia, reexaminando caracteres previamente coincidentes.

El algoritmo fue inventado por Knuth y Pratt e independientemente por J. H. Morris en 1977, pero los tres lo publicaron en conjunto.

### ***El algoritmo KMP***

El objetivo de la tabla es no permitir al algoritmo hacer coincidir ningún carácter de  $S$  más de una vez. La observación clave acerca de la naturaleza de la búsqueda lineal que permite que esto suceda está dada en tener algunos segmentos de la secuencia principal chequeados con un segmento inicial del patrón. Con esto sabemos exactamente en qué lugares una nueva coincidencia potencial que podría continuar desde la posición actual podría comenzar previamente a la posición presente. En otras palabras, realizamos una “pre-búsqueda” del patrón mismo y compilamos una lista de todas las posibles posiciones a las que deberíamos retornar, eludiendo al máximo los caracteres inútiles mientras no sacrificamos ninguna coincidencia potencial por eso.

Suponga que se está comparando el patrón y el texto en una posición dada, cuando se encuentra una discrepancia.



Sea  $X$  la parte del patrón que calza con el texto, e  $Y$  la correspondiente parte del texto, y suponga que el largo de  $X$  es  $j$ . El algoritmo de fuerza bruta mueve el patrón una posición hacia la derecha, sin embargo, esto puede o no puede ser lo correcto en el sentido que los primeros  $j-1$  caracteres de  $X$  pueden o no pueden calzar los últimos  $j-1$  caracteres de  $Y$ .

### ***Validar formato de correo electrónico mediante una expresión regular.***

La expresión regular que se utiliza para validar un correo electrónico en Python es la siguiente: `('^[a-z0-9_\-\.\.]+@[a-z0-9_\-\.\.]+\.[a-z]{2,15}$', correo.lower())` ya que en ella contiene todos los caracteres y símbolos que son permitidos o que debe llevar un correo electrónico.

#### **Seudocódigo**

- ✓ Primero se declaró una variable en la cual se ingresará el correo electrónico que deseen verificar, ya sea con un correo electrónico ya establecido dentro del programa o ingresándolo desde el teclado con la función 'input'.
- ✓ Después se pone la condición en la cual se ingresará la expresión regular dada anteriormente y si esa condición se cumple, se imprime como resultado "Correo correcto".
- ✓ Se crea otra condición, la cual, imprimirá "Correo incorrecto" en dado caso de que la condición anterior no se cumpla.
- ✓ En el siguiente paso se calcula el tiempo de ejecución con la función que se nos proporcionó.

### ***Validar formato de un correo electrónico mediante método propio:***

En este programa primero que nada tuvimos que buscar información sobre los métodos que existen para poder separar nuestro código en dos con el fin de que nos vaya verificando por separado la existencia del carácter como el "@" y el "." que en un correo electrónico son los caracteres más importantes. Para este ejemplo encontramos que se podía utilizar el método Split el cual su función es que es capaz de separar una cadena extensa en partes menos extensas.

Y como ya habíamos mencionado el from time es más que nada para ver cuánto tiempo tarda nuestro programa ejecutándose el cual cada vez que se corra el programa puede que vaya tardando más o se corra mucho más rápido y también esto es dependiendo de la máquina de donde estés corriendo el programa ya que no todas son iguales.

### ***Cual fue el mejor y diferencias:***

El mejor programa fue el que encontramos en internet debido a que no es muy extenso y es mejor entendible ya que en el método propio se generaron más líneas de código y más revuelto.






### ***¿Para qué se realizó?***

Se realizó con el fin de revisar si alguna dirección de correo es una dirección válida o no.

Un correo será válido siempre y cuando se cumplan las especificaciones que se muestran en la expresión.

También se buscó que se comprendiera que hay distintas maneras de verificarlo mediante el uso de expresiones regulares dentro del código.

### ***En la máquina que se probó este código fue en la siguiente:***

-  Computadora marca dell
-  Procesador Intel® core i5
-  Memoria Ram de 8,00 GB
-  Sistema Operativo de 64 bits
-  Windows 10