UANL
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FIME
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

Proyecto: Algoritmo de búsqueda basado en knuth-morris- pratt y boyer-moore

Nombre	***Matricula***	***Carrera***
Orlando Adiel Gonzalez Rocha	1889497	ITS
Daniel Alejandro Chavez Tello	1809882	ITS
Ricardo Antonio Garcia Pedroza	1938334	ITS
Sebastián Morales Leyva	1797510	ITS

Equipo Lila

TEORIA DE LA INFORM.Y METODOS DE CODIFICACION

Manual Técnico

Descripción del proyecto

El programa de este proyecto consiste en la busqueda de un patron de caracteres dentro de un texto definido por el usuario, ya sea introducido de forma manual en el programa, o por medio de un archivo de texto.
La busqueda del patrón en el texto se realizará por medio de un algoritmo que utiliza los métodos de Knuth-Morris-Pratt y Boyer-Moore, con el objetivo de completar la busqueda en el menor número de instancias posibles.
Adicionalmente, se pide ejecutar ambos métodos de forma individual para comparar los resultados obtenidos entre las tres ejecuciones.

Marco Teórico

Problemas de busqueda: Es una de las tareas fundamentales de la programación teórica, ya que, aunque la busqueda de textos es una de los más simples, es de los problemas más importantes de la programación.
La importancia de este problema puede verse en la amplia área de aplicaciones que esta tiene, como lo son, editores de texto, busqueda de texto en línea, obtención de informacion, etc.

Metodo de Knuth-Morris-Pratt: Este algoritmo realiza una busqueda de relaciones hacia adelante, entre los caracteres dentro del patrón, con los caracteres dentro del texto. Genera una tabla de cambios que se utiliza para definir el movimiento del patron con respecto al texto en base al número de combinaciones exitosas de la iteración anterior.

Método de Boyer-Moore: Este algoritmo realiza una busqueda de relaciones hacia atrás, es decir, el primer carácter que se compara con el texto, es el último del patron, en caso de obtener una relación el siguiente carácter en comparase es el penúltimo, y en ese orden. La tabla de cambios elige el cambio en base al último carácter del texto que se comparó. A diferencia de la tabla del algoritmo KMP, esta tabla se inicializa de manera diferente, y por lo tanto, sus valores son diferentes.

Métodos combinados: El algoritmo combinado de busqueda de texto está basado en dos algoritmos, Knuth-Morris-Pratt y Boyer-Moore. La importancia de usar ambos algoritmos al mismo tiempo, es para que, en caso de no obtener un resultado en una iteración, se elija la cantidad de caracteres que va a avanzar la posicion del patron con respecto al texto en base al mayor número de cambio propuesto por las tablas de cambios de cada algoritmo.

Requerimientos

- Leer un texto y un patrón que se quiera buscar dentro del texto, se debe de dar la opción de elegir como ingresar el texto, si por medio de un archivo de texto, o por medio de introducción manual en el programa.
- Hacer uso de los métodos descritos en el archivo.
- Realizar una ejecucion haciendo uso del método de Knuth-Morris-Pratt.
- Realizar una ejecucion haciendo uso del método de Boyer-Moore.
- Realizar una ejecucion usando ambos métodos combinados.
- En caso de obtener encontrar un resultado positivo, indicar el número de iteraciones que se llevaron a cabo y la posicion del patrón en el texto.

Diagrama de flujo

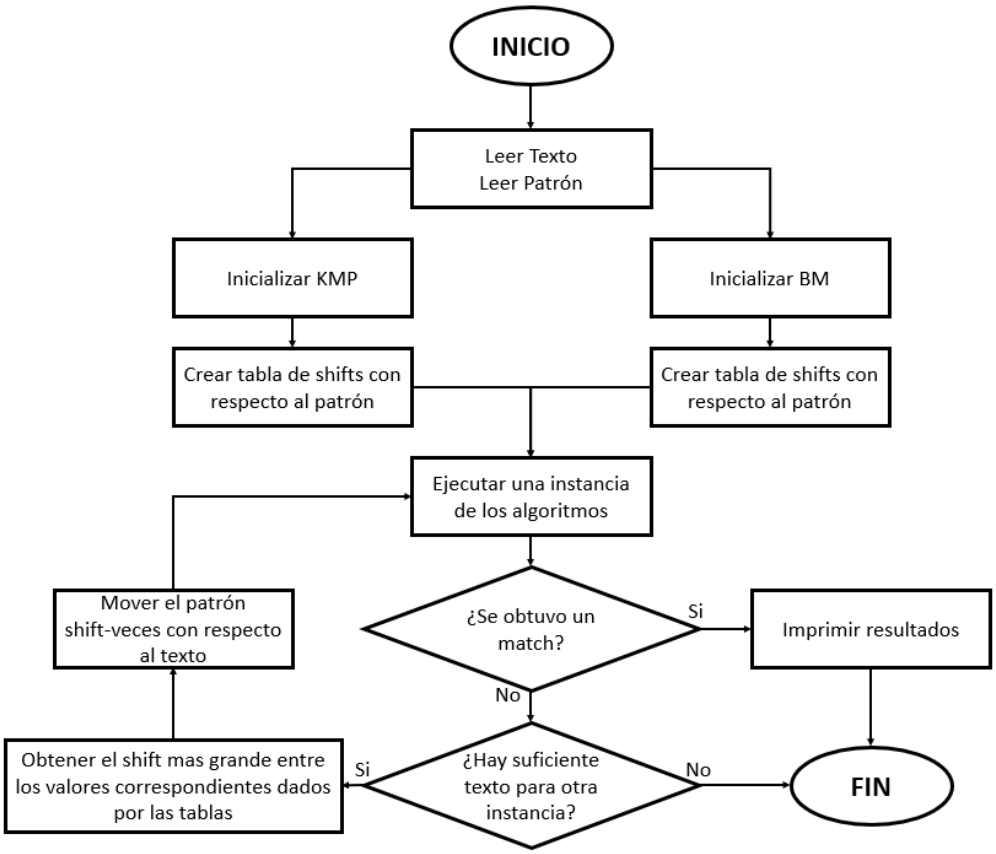
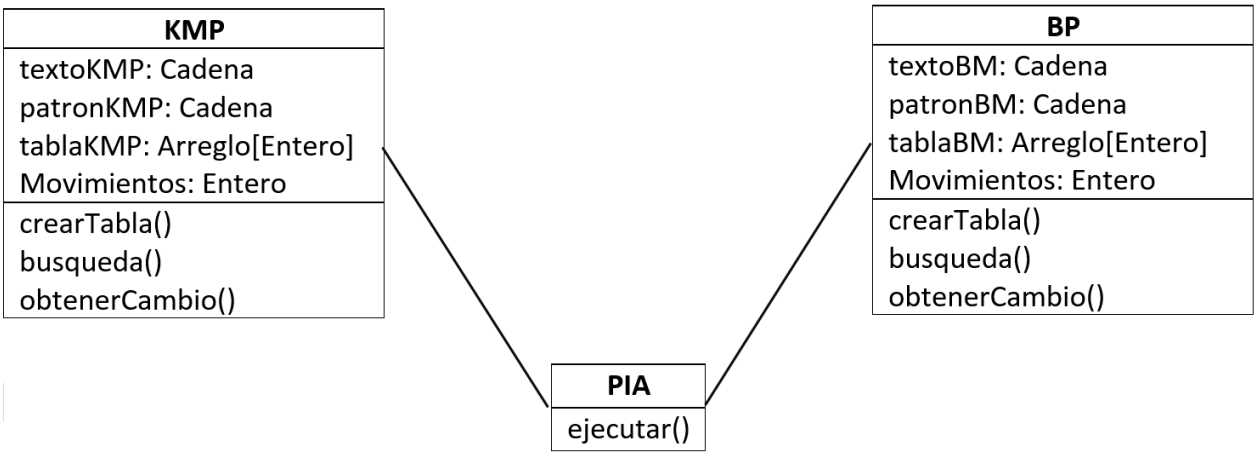


Diagrama de clases



Diccionario de datos

Clase PIA	
Nombre	**Tipo**
texto	Cadena
patron	Cadena
indice	Entero
iteraciones	Entero
Clase KMP	
Nombre	**Tipo**
textoKMP	Cadena
patronKMP	Cadena
tablaKMP	Arreglo[Entero]
movimientos	Entero
Clase BM	
Nombre	**Tipo**
textoBM	Cadena
patronBM	Cadena
tablaBM	Arreglo[Entero]
movimientos	Entero

Funcionalidad

- El programa al iniciar ejecuta el metodo principal de la clase PIA.

```
# Ejecucion del programa
PIA.ejecutar()
```

- El programa pregunta al usuario la forma en la que se va a ingresar el texto, si por medio de archivo o de forma manual
- El programa lee el texto/archivo ingresado por el usuario.
- El programa lee el patron ingresado por el usuario
- Se realizan las validaciones del texto y del patron, de completarse correctamente, se sigue con la ejecucion, en caso contrario, se interrumpe la ejecucion.

```
# Obtencion de datos
operador = input(
    '\n\nBienvenido! Selecciona la opcion  que quieres utilizar: \n\n   1.-Ingresar datos de Manera manual\n\n   2.-Leer un archivo txt  \n\n ')

# Obtencion del texto
if operador == "1":
    texto = input("\nIntroduzca el texto en minusculas -> ")
elif operador == "2":
    filename = input('\n  Escribe la direccion de tu archivo de texto :   ')
    with open(filename, "r") as file:
        texto = file.read().replace('\n', '')
        if texto == "":
            print("No se pudo leer el texto del archivo, revisa la direccion")
            return 0
else:
    "Seleccion no valida"
if texto == "":
    print("Texto invalido")
    return 0

# Obtencion del patron
patron = input("Introduzca el patron en minusculas -> ")
if patron == "":
    print("Patron invalido")
    return 0
```

- Crea una instancia de la clase KMP, enviando el texto y el patron como parámetros para su inicialización y creación de la tabla de cambios.
- Crea una instancia de la clase BM, enviando el texto y el patron como parámetros para su inicialización y creación de la tabla de cambios.

```
# Inicializar clases de los metodos / Creacion de tablas de los metodos
kmp = KMP(texto, patron)
bm = BM(texto, patron)
```

- Realiza la busqueda usando solamente el metodo Knuth-Morris-Pratt, mientras que aún haya texto disponible para realizar la busqueda.

```
# Ejecucion usando el metodo Knuth-Morris-Prath
indice = 0
iteraciones = 0
shiftsKMP = list()
# Mientras que el texto tenga espacio suficiente para realizar la busqueda
while len(texto) > (indice+len(patron)-1):
    # Incrementar valor de iteraciones
    iteraciones = iteraciones+1

    # Realizar una iteracion de busqueda de los metodos en el indice indicado
    resultadoKMP = kmp.busqueda(indice)

    # Si uno de los metodos encontro un match
    if resultadoKMP:
        # Agregar los datos obtenidos a la lista de resultados
        shiftsKMP.append(iteraciones)

        indice = indice+1
    else:
        # Incrementar el indice el numero de cambios definidos
        indice = indice+kmp.obtenerCambio()
```

- Realiza la busqueda usando solamente el metodo Boyer-Moore, mientras que aún haya texto disponible para realizar la busqueda.

```
# Ejecucion usando el metodo Booyer-Moore
indice = 0
iteraciones = 0
shiftsBM = list()
# Mientras que el texto tenga espacio suficiente para realizar la busqueda
while len(texto) > (indice+len(patron)-1):
    # Incrementar valor de iteraciones
    iteraciones = iteraciones+1

    # Realizar una iteracion de busqueda de los metodos en el indice indicado
    resultadoBM = bm.busqueda(indice)

    # Si uno de los metodos encontro un match
    if resultadoBM:
        # Agregar los datos obtenidos a la lista de resultados
        shiftsBM.append(iteraciones)

        indice = indice+1
    else:
        # Incrementar el indice el numero de cambios definidos
        indice = indice+bm.obtenerCambio()
```

- Realiza la busqueda ejecutando la busqueda por medio de ambos métodos mientras que aún haya texto disponible para realizar la busqueda, en caso de que una iteracion no encuentre una relación entre el texto y el patrón, obtener el mayor cambio entre los indicados por las tablas de KMP y BM, y realizar el movimiento del patron con respecto al texto de acuerdo al cambio indicado.

```
# Ejecucion usando los dos metodos combinados
indice = 0
iteraciones = 0
posicionesComb = list()
shiftsComb = list()
# Mientras que el texto tenga espacio suficiente para realizar la busqueda
while len(texto) > (indice+len(patron)-1):
    # Incrementar valor de iteraciones
    iteraciones = iteraciones+1

    # Realizar una iteracion de busqueda de los metodos en el indice indicado
    resultadoKMP = kmp.busqueda(indice)
    resultadoBM = bm.busqueda(indice)

    # Si uno de los metodos encontro un match
    if resultadoKMP  or  resultadoBM:
        # Agregar los datos obtenidos a la lista de resultados
        posicionesComb.append(indice+1)
        shiftsComb.append(iteraciones)

        indice = indice+1
    else:
        # Obtener el mayor cambio establecido por las tablas de los metodos
        cambio = 1
        if kmp.obtenerCambio() > bm.obtenerCambio():
            cambio = kmp.obtenerCambio()
        else:
            cambio = bm.obtenerCambio()

        # Incrementar el indice el numero de cambios definidos
        indice = indice+cambio
```

- Imprimir los resultados obtenidos.

```
# Impresion de resultados
print(f"\n\nNumero de concurrencias del patron en el texto: {len(posicionesComb)}\n")
if len(posicionesComb) != 0:
    print("\t\t\t\t\t\tShifts")
    print("\t\t\t\t", "-"*33)
    print("Palabra\t\tPosicion\t KMP\t\tBM\tCombinados")
    for i in range(len(posicionesComb)):
        print(f"{patron}[{i+1}]\t {posicionesComb[i]}\t\t {shiftsKMP[i]}\t\t{shiftsBM[i]}\t    {shiftsComb[i]}")
else:
    print(f"\n\nPatron '{patron}' no encontrado en el texto")
```