Asignatura	**Análisis de algoritmo**
Código	**1155404-A**
Tipo de asignatura:	**Obligatoria**
Profesores:	**MARCO A. ADARME JAIMES (madarme@ufps.edu.co)**

INFORMACIÓN DE LA ACTIVIDAD

Competencias a desarrollar:

- Ser capaz de utilizar una estrategia algorítmica para la búsqueda de patrones en archivos de texto plano
- Ser capaz de reconocer la importancia de los algoritmos de búsqueda simples de cadena.

Tipo de Actividad: Teórica () Práctica (X)

DESCRIPCIÓN

El *String Matching* es un problema clásico en ciencias de la computación ampliamente estudiado y con repercusiones actuales importantes en el desarrollo de software que realiza tratamiento de cadenas para búsquedas en grandes volúmenes de datos. En este contexto se distinguen dos definiciones: El "Patrón" y el "Texto", el primero comprende una o varias cadenas a buscar dentro de un "Texto"[1]. El término "Texto" se asocia a una cadena escrita en cualquier idioma y que es almacenado en memoria principal a través de una entrada por teclado o su almacenamiento se realiza en archivos de texto plano.

El fin de los algoritmos *String Matching* consiste en buscar coincidencias de patrones dentro de textos por lo general formateados en ASCII[2]. Algunos de sus principales usos comprenden los correctores ortográficos, los filtros de spam, análisis de logs para la detección de intrusos en servidores de red, motores de búsqueda, informática forense, sistemas de recuperación de información, entre otros.

Su tarea consiste en implementar alguno de los siguientes algoritmos String Matching:

A. Boyer Moore Horspool (ver vídeo)
B. Boyer Moore Horspool Sunday (ver vídeo)
C. Tuned Boyer-Moore
D. Karp Robin (ver)
E. Knuth-Morris-Pratt con función de fallo (ver)
F. Shift Or
G. Not So Naive algorithm

Protocolo de Desarrollo:

1. Su equipo debe estar conformado por máximo dos estudiantes.

2. Se debe crear un video de máximo 10 min donde explique de la forma más descriptiva posible el funcionamiento del algoritmo seleccionado. Todos los integrantes del equipo deben participar. Puede utilizar cualquier herramienta lúdica digital que apoye la explicación. Para este punto ES **OBLIGATORIO** utilizar las instalaciones del viveLab o del laboratorio de medios de nuestro programa (3 piso SA). **Debe publicarse en el canal "promedia ufps" de youtube.com.** Deben comunicarse con la Ing Janeth Chapeta al cel: 3162419189 y agendar turno para grabar sus vídeos (ver ejemplo de vídeo). Se recomienda para esto:
   a. Llevar las láminas de la exposición.
   b. Ir vestidos de camisa unicolor preferiblemente oscura.

   EL VIDEO ES SOBRE EL ALGORITMO Y NO DE LA APLICACIÓN PARA ESTO SE SUGIERE, TENER LÁMINAS QUE TENGAN LA SIGUIENTE INFORMACIÓN:
   1. AUTOR DEL MÉTODO
   2. AÑO DE CREACIÓN
   3. DEFINICIÓN DE PATRÓN
   4. EXPLICACIÓN DEL MÉTODO A TRAVÉS DE UN DIAGRAMA DE ACTIVIDAD (FLUJO).
   5. EJEMPLO PRÁCTICO
   6. VENTAJAS
   7. DESVENTAJAS

3. Implementación: Se debe crear una aplicación en JavaScript o tecnologías derivadas y ser publicadas en gitlab pages o github pages.
   a. Una vista(Frontend) que permita:
      i. Digitar un texto en textarea.
      ii. Digitar el patrón
      iii. Mostrar la cantidad de veces que el patrón está en el textarea.
      iv. Se debe crear una estrategia para mostrar paso a paso lo que el algoritmo de string matching realiza.
      v. Una página donde explique el algoritmo (colocar el vídeo que realizó de forma embebida).
      vi. Una página donde muestre el costo algorítmico de la solución, solo de la función que realiza la búsqueda. (Coloque el documento PDF embebido).
   b. Backend:
      i. Una Función con el Algoritmo de String Matching.
      ii. Clases utilitarias: Si el equipo lo considera puede implementar clases para el manejo de sus vistas.

   SI LA APLICACIÓN ES REALIZADA EN JAVA EN FORMA CONVENCIONAL TENDRÁ UNA CALIFICACIÓN MÁXIMA DE 3.0

4. Costo algorítmico: Se debe crear un documento que detalle la complejidad algorítmica en notación Big O del algoritmo seleccionado. **Tome en cuenta que sólo se analiza el método que calcula las incidencias del patrón en el texto. Para esto:**
    a. Cálculo del Tn
    b. Usar si la ecuación permite teorema maestro, expansión o método convencional de análisis.
5. **NO SE PUEDEN UTILIZAR MÉTODOS DE LA CLASE STRING O CLASES DERIVADAS. SOLO SE PERMITE UN ALMACENAMIENTO SIMPLE Y RECORRIDOS A NIVEL DE DATOS TIPO CHAR SOBRE VECTORES.**

Valores porcentuales de la evaluación:

Descripción	Valor
Aplicaciones(Fuentes)	30%
Video	20%
Costo y documento en pdf	10%
Prueba escrita	40%

Si no se presenta la aplicación el trabajo NO SE PODRÁ PRESENTAR LA PRUEBA ESCRITA.

Entregables:

1. Un documento PDF de nombre CODIGO1-CODIGO2.pdf donde código1 y código2 son sus códigos UFPS, que contiene:
    - Portada con los nombres de los integrantes y códigos
    - Costo algorítmico de la solución
    - (OPCIONAL). En caso de utilizar IA debe anexar al documento con declaración que usó IA de forma efectiva y basado en las normas que acá se describen(ver).
    - URL del proyecto en gitlab de forma pública.
    - URL en gitlabpage o githubpage. (**Si es en Java escritorio esté punto sobraría**).
    - URL del vídeo.

El documento debe estar en tamaño carta con párrafos justificados, márgenes de 2cm en todos los lados. Letra color negra Arial de 11pto, interlineado 1,15.

VER PLANTILLA DE DOCUMENTO

2. Un archivo .zip con las fuentes del proyecto, del tipo: CODIGO1-CODIGO2.zip.

BIBLIOGRAFÍA

[1] B. Commentz-Walter, "A string matching algorithm fast on the average," in *Automata, Languages and Programming*, 1979, pp. 118–132.

[2] K. Kapil, R. Soni, A. Vyas, and D. A. Sinhal, "Importance of String Matching in Real World Problems," vol. 3, pp. 2319–7242, 2014.

TODOS LOS TRABAJOS DEBEN CUMPLIR CON EL PROTOCOLO DE ENTREGA

Éxitos!!!