ESTRUCTURA DE DATOS 2
Código ST0247

Taller en Sala 9
Programación Dinámica

Objetivos: **1.** Usar programación dinámica para resolver un problema apropiado. **2.** Traducir problemas del mundo real a soluciones algorítmicas.

Consideraciones: Lean y verifiquen las consideraciones de entrega,

Trabajo en Parejas

Mañana, plazo de entrega

Docente entrega plantilla de código en GitHub

Sí .cpp, .py o .java

No .zip, .txt, html o .doc

Alumnos entregan código sin comprimir GitHub

En la carpeta Github del curso, hay un código iniciado y un código de pruebas (tests) que pueden explorar para solucionar los ejercicios

Estructura del documento: **a)** Datos de *vida real*, **b)** I*ntroducción* a un problema, **c)** Problema a resolver, **d)** Ayudas. Identifiquen esos elementos así:

a)

b)

c)

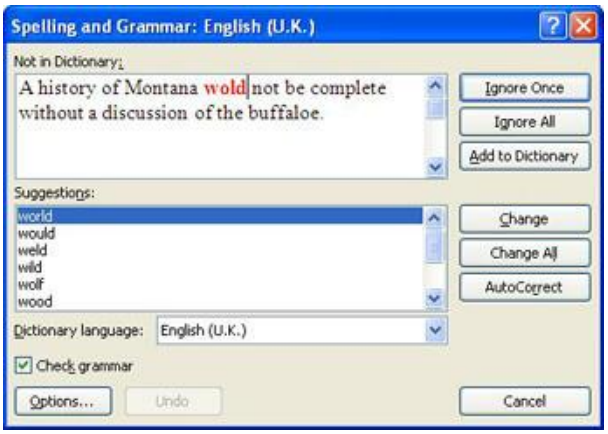
d)

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

En la vida real, la distancia de *Levenshtein* se utiliza para algoritmos de reconocimiento óptico de caracteres, es decir, pasar de imagen a texto. También se utiliza en correctores de ortografía, como el que trae *Microsoft Word* y el de los teclados de los celulares, al igual que en procesamiento del lenguaje natural como lo hace *Siri* de *Apple*

Ejercicios a resolver

1. El procesador de textos *Microsoft Word* tiene un corrector de ortografía. Este propone palabras similares a la que se escribe sí existen en el diccionario que trae Word. Una métrica de similitud entre palabras es la distancia de *Levenshtein*.

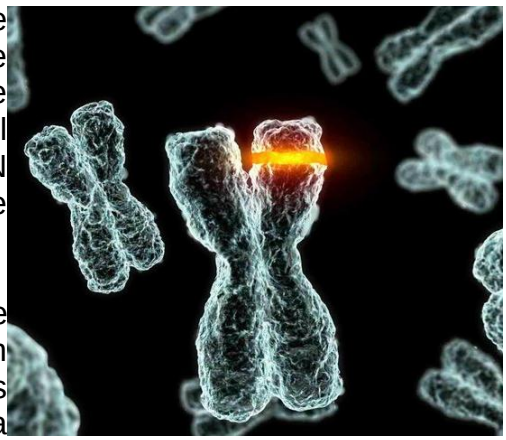


Un problema con esta distancia es que calcularla usando *backtracking* tiene una complejidad muy alta que haría muy lento al corrector de ortografía. Una solución es optimizar el algoritmo con programación dinámica.

Dadas dos cadenas de caracteres *a* y *b*, determine la distancia *Levenshtein* que hay entre ellas, es decir, la cantidad mínima de operaciones (insertar, remover o cambiar una letra) que se necesitan para transformar una en la otra utilizando programación dinámica. Considera que se asigna el mismo peso a una substitución, una inserción y un borrado.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

[Ejercicio Opcional] El genoma humano fue decodificado en su totalidad en el 2003. Desde ese entonces, una de las grandes líneas de investigación en Bioinformática es encontrar, en el genoma humano, las secuencias de ADN responsables de la aparición de diferentes tipos de cáncer.



Las cadenas de ADN son, simplemente, cadenas de caracteres. El problema de encontrar un patrón cancerígeno en el ADN de una persona, en algunos casos, puede verse como el problema de la subsecuencia común más larga.

El problema de la subsecuencia común más larga es el siguiente: Dadas dos secuencias, encontrar la longitud de la secuencia más larga presente en ambas. Una subsecuencia es una secuencia que aparece en el mismo orden relativo, pero no necesariamente de forma contigua.

Implementen un algoritmo que calcule la longitud de la subsecuencia común más larga a dos cadenas de caracteres utilizando programación dinámica.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ayudas para resolver los Ejercicios

Ejercicio 1

Pista 1:

```
public static int levenshtein(String a, String b) {
// complete...
}
```

Ejemplo 1: Si tenemos las palabras "*carro*" y "*casa*" la distancia *Levenshtein* que hay entre ellas es 3:

1. Remover una letra: "carr"
2. Cambiar una letra: "casr"
3. Cambiar una letra: "casa"

Pista 2: Para observar la respuesta que entrega el algoritmo, utilice el simulador que se encuentra en **https://bit.ly/2tVtaoP**. Considera que se asigna el mismo peso a una inserción, un borrado y una substitución, es decir, configura así los parámetros del simulador: `indel=1`, substitution=`1` y `swap=1`

Pista 3: Nótese que las operaciones y su orden pueden ser diferentes, pero lo importante es que el número mínimo de operaciones para transformar "carro" en "casa" son 3.

Pista 4: Asuman que las cadenas dadas están ambas completamente en minúscula o mayúscula.

Pista 5: Solucionen el siguiente problema para tener una mayor seguridad que su implementación es correcta: **https://bit.ly/2KohOVI**

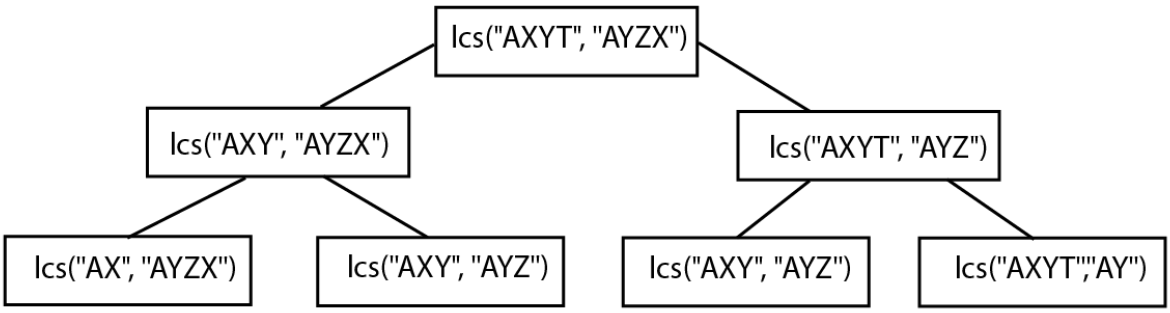
PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ejercicio 2

Ejemplo 1: “abc”, “abg”, “bdf”, “aeg” y “acefg” son subsecuencias de “abcdefg”. Entonces, para una cadena de longitud n existen 2^n posibles subsecuencias.

Ejemplo 2: Consideren los siguientes ejemplos para el problema:

Para “ABCDGH” y “AEDFHR” es “ADH” y su longitud es 3. Para “AGGTAB” y “GXTXAYB” es “GTAB” y su longitud es 4. Una forma de resolver este problema es usando *backtracking*, como un ejemplo, para las cadenas “AXYT” y “AYZX”, dada una función recursiva los que resuelve el problema, se obtendría el siguiente árbol (parcial) de recursión:



Pista 1: Usando *backtracking* para ese problema, el problema lcs(“AXY”, “AYZ”) se resuelve dos veces. Si dibujamos el árbol de recursión completo, veremos que aparecen más y más problemas repetidos, así como en el caso de serie de Fibonacci. Este problema se puede solucionar guardando la soluciones, que ya se han calculado para los subproblemas, en una tabla; es decir, usando programación dinámica.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Pista 2: Completen el siguiente método:

```
// Precondición: Ambas cadenas x, y son no vacías
public static int lcsdyn(String x, String y) {
 …
}
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

¿Alguna inquietud?

CONTACTO

Docente Mauricio Toro Bermúdez
Teléfono: (+57) (4) 261 95 00 Ext. 9473
Correo: mtorobe@eafit.edu.co
Oficina: 19- 627

Agenden una cita dando clic en la pestaña -*Semana*- de http://bit.ly/2gzVg10