Semana 2 - Ejercicios de grupo

Métodos algorítmicos en resolución de problemas II
Facultad de Informática - UCM

	Nombres y apellidos de los componentes del grupo que participan	**ID Juez**
1	Daniel Hernández Martínez	MAR246
2	Miguel Verdaguer Velázquez	MAR285
3	Jorge Arevalo Echevarria	MAR205
4	Jesus Martin Moraleda	MAR255

Instrucciones:

1. Para editar este documento, es necesario hacer una copia de él. Para ello:
    - Alguien del grupo inicia sesión con la cuenta de correo de la UCM (si no la ha iniciado ya) y accede a este documento.
    - Mediante la opción *Archivo* → *Hacer una copia*, hace una copia del documento en su propia unidad de *Google Drive*.
    - Abre esta copia y, mediante el botón *Compartir* (esquina superior derecha), introduce los correos de los demás miembros del grupo para que puedan participar en la edición de la copia.

2. La entrega se realiza a través del Campus Virtual. Para ello:
    - Alguien del grupo convierte este documento a PDF (dándole como nombre el número del grupo, 1.pdf, 2.pdf, etc…). Desde *Google Docs*, puede hacerse mediante la opción *Archivo* → *Descargar* → *Documento PDF*.
    - Esta misma persona sube el fichero PDF a la tarea correspondiente del *Campus Virtual*. Solo es necesario que uno de los componentes del grupo entregue el PDF.

3,5

Subsecuencia común más larga

El objetivo de hoy es resolver el **problema 7 Subsecuencia común más larga**, del juez automático. Ahí podéis encontrar el enunciado completo del problema.

Las subsecuencias de una secuencia son todas aquellas que se obtienen a base de seleccionar algunos caracteres de la secuencia (una cantidad entre 0 y la longitud de la secuencia) en el mismo orden en que se encuentran en la secuencia original. Dadas dos secuencias de caracteres se pretende encontrar una subsecuencia de ambas que sea lo más larga posible. Fijaos en que si hay varias subsecuencias comunes de igual longitud se puede devolver como resultado cualquiera de ellas.

Solución: (Plantead aquí la recurrencia que resuelve el problema y explicad las razones por las que es mejor resolver el problema utilizando programación dinámica que una implementación recursiva sin memoria.)

Subsecuencia (i,j) = Longitud maxima ... Rango de i y j

i,j son los índices de palabra1 y palabra2, de longitud n y m respectivamente.

3,5

Caso Recursivo (si i < n-1 y j < m -1):
subsecuencia(i,j) = subsecuencia(i-1,j-1) + 1 si palabra1[i] == palabra2[j]
max(subsecuencia(i-1,j), subsecuencia(i,j-1)) si palabra1[i] != palabra2[j]

Caso base:
subsecuencia(i,j) = 0 si i >= n-1 ó j >= m-1

Si los índices decrecen, el caso base está erróneo

Incoherente con lo de después

Con esto conseguiríamos la longitud, después se puede reconstruir la solución.

Es mejor resolverlo mediante programación dinámica porque se repiten subproblemas y así bajamos el coste considerablemente.

Llamada inicial ?

Solución: (Escribid aquí las explicaciones necesarias para contar de manera comprensible la implementación del algoritmo de programación dinámica. Explicad con detalle cómo se construye una de las subsecuencias comunes más largas y en qué parte de ese código se está eligiendo una de las posibles subsecuencias en caso de existir varias. Dad un ejemplo en que existan varias posibles soluciones e indicad cuál elegiría vuestro algoritmo.
Incluid el código y el coste justificado de las funciones que resuelven el problema. Extended el espacio al que haga falta.)

La implementación usa una matriz de longitud(palabra1)*longitud(palabra2) donde va guardando la máxima longitud de subsecuencia para cada par de índices de palabras. Esto lo hacemos mediante una función recursiva que comprueba si ya se ha guardado la solución para esa combinación de i y j y si no hace la recursión que hemos descrito antes. Después, en la función principal, recorremos la matriz, recuperando la solución de máxima longitud. Para ello usamos una función recursiva que recorre la matriz desde el (0,0) hacia derecha, abajo, o diagonal abajo-izquierda poniendo la letra correspondiente si en la posición a la que pasamos es menor o continua si es igual.

AQUÍ LA SOLUCIÓN A ESTE APARTADO

```
const int N = 500;
const int M = 500;


int subsecuencia(int i, int j, Matriz<int>& subs, string const& palabra1, string const& palabra2) {
    int& aux = subs[i][j];
    if (aux == -1) {
        if ((i > palabra1.size() - 1) || (j > palabra2.size() - 1))
            aux = 0;
        else if (palabra1[i] == palabra2[j])
            aux = subsecuencia(i + 1, j + 1, subs, palabra1, palabra2) + 1;
        else
            aux = max(subsecuencia(i + 1, j, subs, palabra1, palabra2), subsecuencia(i, j + 1, subs, palabra1,
palabra2));
    }
    return aux;

}


string reconstruir(string const& palabra, Matriz<int> const& subs, int i, int j) {
    int aux = subs[i][j];
    if (aux == 0 || aux == -1)
        return {};
    int a = i + 1, b = j;
    if (subs[i + 1][j + 1] > subs[a][b]) {
        a = i + 1;
        b = j + 1;
    }
    if (subs[i][j + 1] > subs[a][b]) {
        a = i;
        b = j + 1;
    }
    if (aux == subs[a][b])
        return reconstruir(palabra, subs, a, b);
    else
        return palabra[i] + reconstruir(palabra, subs, a, b);
}
```

2

incorrecto
Necesitas saber si los caracteres son iguales, con los longs no basta

La reconstrucción no es correcta

```
string resolver(string p1, string p2) {
   Matriz<int> subs(p1.size()+1, p2.size()+1, -1);
   subsecuencia(0, 0, subs, p1, p2);
   return reconstruir(p1, subs, 0, 0);
}
```

¿? O(n*m)

El coste será el min(n*n , m*m) (cuadrático) siendo n longitud de palabra1 y m longitud de palabra2, ya que el número de subproblemas distintos es de ese orden, y una vez que tenga la solución de este subproblema el coste será constante. El coste en espacio será la matriz de espacio n*m para las dos funciones.
El coste de reconstruir es de n siendo la longitud de la solución, y con la concatenación que lineal respecto a la solución también, se nos queda un coste final (n*n) cuadrático.

O(n+m) No es correcto

Resolved el problema completo del juez, que ahora debe ser ya muy sencillo.

Número de envío: s32021 WA