95

Problema 48

SOTILLO SOTILLO, CLARA (TAIS96)

ID envio	Usuario/a	Hora envío	Veredicto
63715	TAIS96	2022-11-23 10:58	AC
63653	TAIS96	2022-11-23 10:27	AC
63632	TAIS96	2022-11-23 10:14	WA

Fichero Source.cpp

```
*
* Indicad el nombre completo y usuario del juez de quienes habéis hecho esta solución:
* Estudiante 1:Carlos Gomez Lopez TAIS45
* Estudiante 2:Clara Sotillo Sotillo TAIS96
*
```

El coste de la funcion en el caso peor es O(m*n) siendo m la longitud de la primera palabra y n la longitud de la segunda.

↳ tanto en tiempo como en espacio adicional

-Recursion:

subsecuecia(p1,p2,i,j)= es la subsecuencia común de letras mas larga entre dos palabras, la palabra p1 con caracteres i, y la palabra p2 con caracteres j.

X[i..N) e Y[j..M)

```
                          {   subsecuencia(p1,p2,i,j) = 0
       si i == p1.length() == i || p2.length() == j
subsecuencia(p1,p2,i,j)   {   max(subsecuencia(p1,p2,i+1,j) ,subsecuencia(p1,p2,i,j+1))
       si p1[i]!=p2[j]
                          {   1+subsecuencia(p1,p2,i+1,j+1)
       si p1[i]==p2[j]
```

Tenemos un caso base, el cual tiene una de las dos palabras sin ningun caracter, en este caso no necesitamos leer las palabras.

```
subsecuencia(p1,p2,i,j) = 0     si i == p1.length() || p2.length() == j
```

Dentro de los casos recursivos podemos distinguir dos:

-Si el caracter i de la palabra p1 es distinto del caracter j de la palabra p2, entonces tenemos que comprobar la subsecuencia mas larga comprobando dos opciones, o la maxima subsecuencia a partir del siguiente caracter de la palabra p1 o la maxima subsecuencia a partir del siguiente caracter de la palabra p2

```
subsecuencia(p1,p2,i,j) = max(subsecuencia(p1,p2,i+1,j) ,subsecuencia(p1,p2,i,j+1))   si p1[i]!=p2[j]
```

-Si el caracter de la palabra p1 es igual al caracter j de la palabra p2, entonces incrementamos en 1 la funcion mas la recursion, la cual

obtienes la maxima subsecuencia a partir del siguiente caracter de la palabra p1 y p2

subsecuencia(p1,p2,i,j) = 1 + subsecuencia(p1,p2,i+1,j+1) si p1[i]==p2[j]

Implementamos la función de manera descendente rellenando la tabla de tamaño (N+1)(M+1), de abajo a arriba y de derecha a izquierda.

Primero creamos una matriz de int de tamaño n+1 * m+1 la cual rellenamos de forma recursiva a partir de la recursion descrita anteriormente
Una vez conocemos la longitud de la maxima subsecuencia, entonces pasamos a reconstruir dicha subsecuencia , aplicando esta recursion

reconstruir(p1,p2,i,j,m) { reconstruir(p1,p2,i,j) = "" si i == p1.length() == i || p2.length() == j
{ p1[i] + reconstruir(p1,p2,i + 1,j) si p1[i]==p2[j]
{ reconstruir(p1,p2, i + 1, j, m) si (m[i][j] == m[i + 1][j]
{ reconstruir(p1,p2, i, j + 1, m) si (m[i][j] != m[i + 1][j]

```
int subsecuencia(string const& p1, string const& p2, int i, int j, Matriz<int>& m) {

    if (m[i][j] != -1) {
        return m[i][j];
    }

    if (i == p1.length() || j == p2.length()) {
        m[i][j] = 0;
    }
    else if (p1[i] != p2[j]) {
        m[i][j] = max(subsecuencia(p1, p2, i + 1, j, m), subsecuencia(p1, p2, i, j + 1, m));
    }
    else {
        m[i][j] = 1 + subsecuencia(p1, p2, i + 1, j + 1, m);
    }

    return m[i][j];

}

string reconstruir(string const& p1,string const& p2, Matriz<int> const& m, int i, int j) {
    if (i == p1.length() || j == p2.length()) {
        return "";
    }
    if (p1[i] == p2[j])
        return p1[i] + reconstruir(p1,p2,m, i + 1, j + 1);
    else if (m[i][j] == m[i + 1][j])
        return reconstruir(p1,p2, m, i + 1, j);
```

```
    else
        return reconstruir(p1,p2,m, i, j + 1);
}

bool resuelveCaso() {

    // leemos la entrada
    string X, Y;
    cin >> X >> Y;

    if (!cin)
        return false;

    Matriz<int> m(X.length() + 1, Y.length() + 1, -1);

    int sol = subsecuencia(X, Y, 0, 0, m);



    cout << reconstruir(X,Y,m,0,0) << "\n";




    return true;
}
```