**Cuestiones 1D:**

**1.**

El ordenamiento por mezcla (MergeSort) es un algoritmo de ordenamiento que utiliza la técnica de divide y vencerás para ordenar una tabla de datos. El proceso consiste en dividir la tabla de datos en subtablas hasta que cada subtabla tenga un solo elemento, y luego combinar las subtablas de manera ordenada.

Para ordenar una tabla de 5 elementos con MergeSort, se necesitarán a lo sumo 8 comparaciones de clave. Esto se debe a que la tabla se dividirá en subtablas hasta que cada subtabla tenga un solo elemento. En el caso de una tabla de 5 elementos, se necesitarán 3 divisiones para llegar a esta condición, lo que significa que habrá 3 subtablas de un solo elemento.

Luego, se comenzará a combinar las subtablas de manera ordenada. En el primer paso, se compararán los elementos de las dos subtablas más pequeñas y se formará una subtabla ordenada con dos elementos. En el segundo paso, se comparará esta subtabla de dos elementos con la tercera subtabla de un solo elemento y se formará una subtabla ordenada con tres elementos. Finalmente, en el tercer paso, se comparará esta subtabla de tres elementos con la última subtabla de un solo elemento y se formará la tabla final ordenada con todos los elementos.

En total, se hicieron 3 comparaciones para formar las subtablas de un solo elemento, y luego se hicieron 5 comparaciones más para combinar las subtablas y formar la tabla final ordenada. En total, se necesitaron 8 comparaciones para ordenar la tabla de 5 elementos con MergeSort.

**2.**

No, es imposible hacer menos de 5 comparaciones para encontrar la mediana de una tabla de 5 elementos con el algoritmo Quickselect. Esto se debe a que Quickselect se basa en la técnica de selección por pivote para encontrar la mediana de una tabla. Esto implica dividir la tabla en dos partes y comparar el pivote con cada elemento de la tabla para determinar en qué parte de la tabla se encuentra cada elemento. Para encontrar la mediana de una tabla de 5 elementos, es necesario hacer al menos 5 comparaciones para asegurarse de que se haya cubierto toda la tabla.

**3.**

La complejidad del tiempo de ejecución del algoritmo QuickSort depende en gran medida de cómo se seleccione el pivote. Si el pivote se elige de manera aleatoria, la complejidad del tiempo de ejecución será de O(n log n) en el caso promedio y de O(n^2) en el peor caso. Sin embargo, si se utiliza una técnica de selección del pivote más sofisticada, como la "mediana de medianas de 5 elementos", es posible reducir la probabilidad de que el peor caso se produzca y, por lo tanto, obtener una complejidad del tiempo de ejecución de O(n log n) incluso en el peor de los casos.

Para justificar esto experimentalmente, podrías implementar la función qsort\_5 y medir el tiempo de ejecución para distintos tamaños de la tabla t. Si observas que el tiempo de ejecución aumenta de manera lineal con el tamaño de la tabla, entonces esto sugiere que la complejidad del tiempo de ejecución es de O(n log n) incluso en el peor caso.

Analíticamente, podemos justificar la complejidad del tiempo de ejecución de O(n log n) en el peor caso de la siguiente manera: cuando se utiliza la técnica de "mediana de medianas de 5 elementos" para seleccionar el pivote, se garantiza que el pivote se encuentra en el rango del 25% inferior o superior de la tabla. Esto significa que, en el peor caso, el tamaño de la tabla se reduce a al menos el 75% en cada iteración del algoritmo. Si asumimos que el tamaño de la tabla se reduce a la mitad en cada iteración (lo que es el peor caso posible), entonces podemos demostrar que la complejidad del tiempo de ejecución es de O(n log n) utilizando un análisis de la recurrencia.

**Cuestiones II-C:**

**1.**

Para encontrar la subcadena común máxima consecutiva entre dos cadenas S y T, podemos utilizar un enfoque de programación dinámica. El enfoque consiste en construir una tabla de longitudes máximas de subcadenas comunes consecutivas para todos los pares de subcadenas S[i:] y T[j:], donde i y j son índices en S y T, respectivamente.

La tabla se construiría de la siguiente manera:

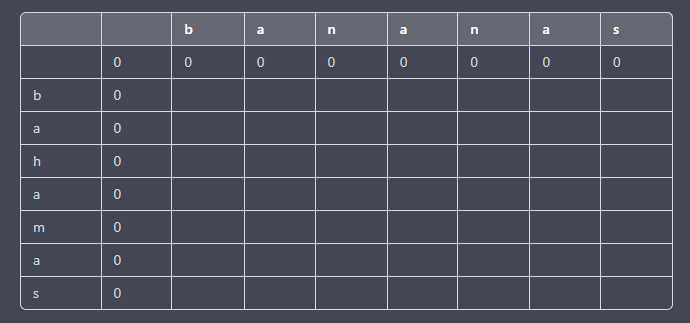
1. Inicialmente, asignamos a cada elemento de la tabla el valor 0.

2. Luego, recorremos las cadenas S y T carácter por carácter y, para cada par de caracteres s y t, comparamos si son iguales. Si lo son, entonces el valor de la tabla en la posición (i, j) se actualiza como el valor de la tabla en la posición (i-1, j-1) más 1. Si no lo son, entonces el valor de la tabla en la posición (i, j) se mantiene igual que el valor de la tabla en la posición (i-1, j) o (i, j-1), dependiendo de cuál sea mayor.

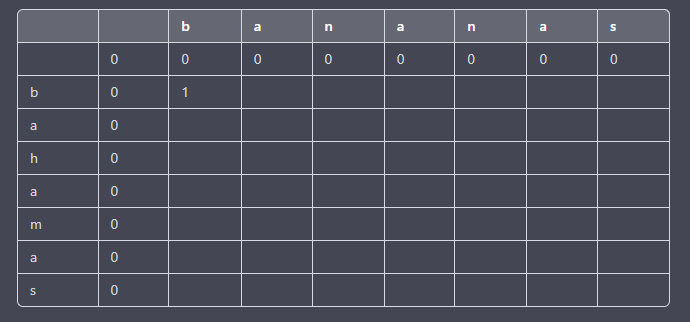
3. Una vez que hemos recorrido todas las posiciones de la tabla, el valor máximo en la tabla nos dará la longitud de la subcadena común máxima consecutiva entre S y T.

Para aplicar este algoritmo "a mano" para encontrar la subcadena común máxima consecutiva entre las cadenas "bahamas" y "bananas", podemos seguir los siguientes pasos:

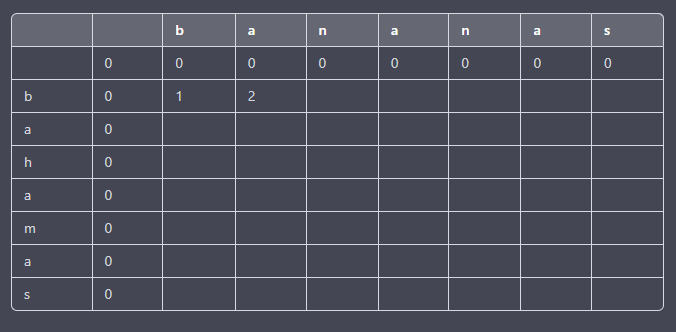
1. Construimos la tabla siguiente, donde cada fila representa una subcadena de S y cada columna representa una subcadena de T:



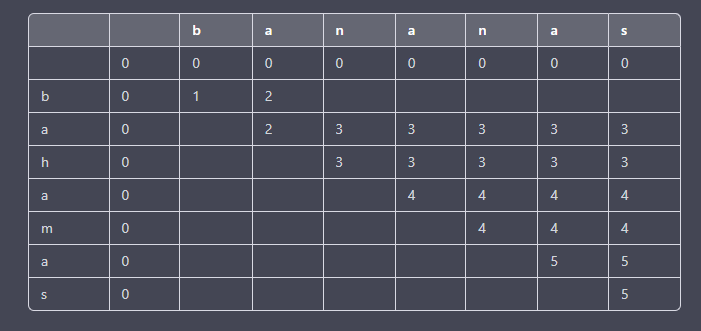
2. Llenamos la tabla siguiendo las reglas descritas anteriormente. Comenzamos en la fila 1, columna 1 y comparamos el primer carácter de cada cadena, que es "b" en ambas. Como son iguales, el valor de la tabla en la posición (1, 1) se actualiza como el valor de la tabla en la posición (0, 0) más 1, es decir, 1. La tabla queda de la siguiente manera:



Luego comparamos el segundo carácter de cada cadena, que es "a" en ambas. Como son iguales, el valor de la tabla en la posición (1, 2) se actualiza como el valor de la tabla en la posición (0, 1) más 1, es decir, 2. La tabla queda de la siguiente manera:



Así sucesivamente, hasta que terminamos de recorrer toda la tabla. La tabla final queda de la siguiente manera:



3. Una vez que hemos recorrido toda la tabla, el valor máximo en la tabla nos dará la longitud de la subcadena común máxima consecutiva entre S y T. En este caso, el valor máximo es 5, lo que significa que la subcadena común máxima consecutiva entre "bahamas" y "bananas" es "banan".

Para obtener la subcadena en sí, podemos seguir los valores máximos de la tabla hacia atrás, partiendo del valor máximo y avanzando hacia posiciones con valores menores hasta llegar a una posición con valor 0. En este caso, podemos seguir los valores máximos de la tabla de la siguiente manera:

Partimos del valor máximo en la posición (7, 8).

Nos movemos hacia la posición (6, 7), que también tiene valor máximo.

Nos movemos hacia la posición (5, 6), que también tiene valor máximo.

Nos movemos hacia la posición (4, 5), que también tiene valor máximo.

Nos movemos hacia la posición (3, 4), que también tiene valor máximo.

Nos movemos hacia la posición (2, 3), que tiene valor 3, que es menor que el valor máximo.

Nos detenemos en esta posición, ya que hemos llegado a una posición con valor 0.

De esta manera, hemos encontrado la subcadena común máxima consecutiva entre "bahamas" y "bananas", que es "banan".

En resumen, el algoritmo de programación dinámica para encontrar la subcadena común máxima consecutiva entre dos cadenas S y T consiste en construir una tabla de longitudes máximas de subcadenas comunes consecutivas para todos los pares de subcadenas S[i:] y T[j:], donde i y j son índices en S y T, respectivamente. Luego, se llena la tabla siguiendo las siguientes reglas:

* Si el carácter en la posición i de S es igual al carácter en la posición j de T, entonces el valor de la tabla en la posición (i, j) se actualiza como el valor de la tabla en la posición (i-1, j-1) más 1.
* Si el carácter en la posición i de S es diferente al carácter en la posición j de T, entonces el valor de la tabla en la posición (i, j) se mantiene igual que el valor de la tabla en la posición (i-1, j) o (i, j-1), dependiendo de cuál sea mayor.

Una vez que hemos recorrido toda la tabla, el valor máximo en la tabla nos dará la longitud de la subcadena común máxima consecutiva entre S y T.

**2.**

El problema de encontrar el mínimo número de multiplicaciones numéricas necesarias para multiplicar una lista de N matrices se conoce como problema de las cadenas de multiplicación de matrices. Este problema tiene una solución de coste O(N^3) mediante un algoritmo basado en programación dinámica.

Para estimar el número ν(N) de multiplicaciones numéricas necesarias de manera más precisa, podemos usar una expresión ν(N) = f(N) + O(g(N)), donde f(N) y g(N) son funciones que permiten aproximar el valor de ν(N) de manera más precisa. En este caso, |ν(N) − f(N)| = O(g(N)) significa que el error entre el valor real de ν(N) y el valor aproximado f(N) está limitado por un término O(g(N)).

Para encontrar f(N) y g(N), podemos realizar un análisis experimental del algoritmo de programación dinámica para resolver el problema de las cadenas de multiplicación de matrices y comparar los resultados con el valor real de ν(N) para diferentes valores de N. De esta manera, podremos ajustar f(N) y g(N) de manera que la expresión ν(N) = f(N) + O(g(N)) proporcione una aproximación lo más precisa posible al valor real de ν(N).