Un dibujo en una pared

Descripción generada automáticamente con confianza mediaDesarrollo de un simulador de alineamiento múltiple secuencial para la evaluación de secuencias biológicas.

El propósito de este programa es obtener las mejores secuencias de caracteres en una matriz por medio de una puntuación y descarte de resultados no favorables.

El código utiliza una clase auxiliar llamada “**resultado**” que tiene como propiedades la matriz generada y su respectivo puntaje. Esto es importante ya que me facilitó el almacenamiento de las mejores matrices en funciones que detallaré a continuación.

La función “**initialize**” recibe como parámetros un arreglo bidimensional de caracteres y un arreglo de secuencias, utiliza la longitud del arreglo de secuencias para iterar sobre cada secuencia (sec1, sec2, sec3) haciendo uso de un ciclo fori, dentro de este ciclo se encuentra otro ciclo for en el cual se itera sobre cada carácter de la secuencia actual y se substrae el carácter en posición utilizando la función character at index (charAt(i)), se realiza la validación de carácter válido retornada por el llamado a la función “**generarGaps**”.

La función “**generarGaps**” hace uso de una instancia de la clase Random para definir la probabilidad de inserción de gaps en este caso establecida al 30% y se declara una variable booleana “hayCaracter” se utiliza un ciclo for para iterar sobre la fila del arreglo, utilizo random para seleccionar una posición aleatoria en el rango de la longitud de la fila y antes de ser insertada a la fila esta es recorrida una posición a la derecha, se realiza la validación de carácter válido asegurando que haya al menos un carácter válido (cualquier carácter con excepción de un gap o un char vacío es válido).

Función auxiliar “**compareChars**” obtiene como parámetros 3 caracteres en donde se válida si existe una coincidencia entre al menos 2 de estos en caso de que sí retorna True, de lo contrario los 3 son mismatches y retorna False. Esto para facilitar la lógica de puntuación de matrices.

Función “**puntuarMatriz**” recibe como parámetro un arreglo de caracteres, se declaran las variables de puntaje, coincidencia (match), incongruencia (mismatch), gap y comparaciones, las he utilizado para llevar un conteo de cada caso posible y una variable maxLength para evitar el caso en el que una matriz sea mayor en longitud que otra y compare nulls con caracteres válidos; hace uso de la función Math.max para obtener el máximo entre 2 valores. Se utiliza un ciclo for para iterar a través de las columnas hasta alcanzar la longitud máxima de las filas, en cada iteración se obtiene el carácter de las tres en misma posición de la actual columna y se hace uso de la función “**compareChars**” para determinar coincidencias o penalizaciones. Si el resultado de **compareChars** es True entonces se aumenta el contador matches y se suma 2 puntos a la matriz, si el resultado es False entonces se aumenta el contador de mismatch y se resta 1 punto a la matriz, en caso de que cualquiera de los 3 caracteres sea un gap se aumenta su contador y se resta 3 puntos a la matriz, esta validación sucede al final dado que puede haber una situación en la que el carácter de la primer fila sea un gap pero los caracteres de la segunda y tercer fila coincidan. Se imprime el conteo y puntaje de la matriz evaluada y se retorna el puntaje obtenido esto será necesario para evaluar cada matriz y conservar su puntaje.

Función “**evaluarMatrices**” de tipo ‘resultado’ recibe un arreglo de secuencias el cual es utilizado solo para obtener la longitud máxima de estas, se ajusta un nuevo arreglo de caracteres a las necesidades de estas secuencias, se inicializan las variables para almacenar la ‘mejor matriz’ le he establecido el valor más pequeño posible debido a que por la lógica de puntuación los resultados son usualmente negativos, utilizo un ciclo for establecido en 10 iteraciones (esto puede variar dependiendo al grado de mejora deseado ya que se obtendrán las 10 mejores matrices de todas estas iteraciones, si fuesen 100 iteraciones serian 100\*10=1000 matrices de las cuales se obtienen 10 mejores), en cada iteración se restablece la matriz y se inicializa nuevamente para obtener otra matriz aleatoria con el método “**initialize**”, esta matriz es posteriormente puntuada por el método “**puntuarMatriz**”, se compara el puntaje obtenido de esta con el puntaje máximo actual, si este es mayor el puntaje máximo se actualiza y la matriz en cuestión se almacena en la variable “**bestMatriz**”, retorna una nueva instancia de tipo resultado con la mejor matriz y su respectivo puntaje, para evitar visualizar puntajes negativos utilicé la clase Math.abs para convertir en positivo el valor, se pudiese multiplicar por -1 pero esta función valida internamente que el valor a convertir sea negativo.

Por último pero no menos importante, el método “**printMatriz**”, itera a través de la matriz e imprime en consola cada carácter de forma ordenada, esto permite la visualización de las mejores matrices para corroborar que el resultado es el esperado.

Finalmente en el método main es donde ingresé las ‘secuencias’ de prueba, después en un ciclo for establecido en 10 iteraciones he almacenado un llamado a la función **evaluarMatrices** que obtiene las secuencias declaradas poco antes(recordar que el método retorna una nueva instancia de resultado), y cada iteración es ingresada a una lista de objetos de tipo resultado llamada bestMatrices, después con un ciclo foreach (for each matriz in bestMatrices) se imprime en consola cada matriz incluyendo su índice actual y su puntaje.

En conclusión; estos métodos en conjunto satisfacen lo que a mi criterio son las necesidades del proyecto, sin embargo para realmente poder trabajar con secuencias biológicas como las proteínas o ADN es necesario realizar ajustes en las dimensiones en las que los arreglos operan, por lo que únicamente es funcional cuando se tiene establecida la dimensión de las secuencias a evaluar, me gustaría señalar que el planteamiento de este proyecto me resultó sumamente interesante, he aceptado el desafío y me encontré con funcionalidades nuevas como la reflexión de objetos en lenguaje C, validaciones ternarias y algoritmos como el Longest Common Subsequence, progressive alignment y global sequence alignment. Es de mi agrado expresar que este proyecto me ha parecido genial y bastante entretenido.