Proyecto Local Alignment by Leon Davis

Este proyecto implementa un alineamiento local de secuencias con manejo de scores ceros y alineamientos óptimos.

Requisitos

- .NET SDK 7.0 o superior
- Git

Cómo ejecutar el proyecto

1. Clonar el repositorio:

```
git clone https://github.com/LeonDavisCoropuna/LocalAlignment.git
cd LocalAlignment
```

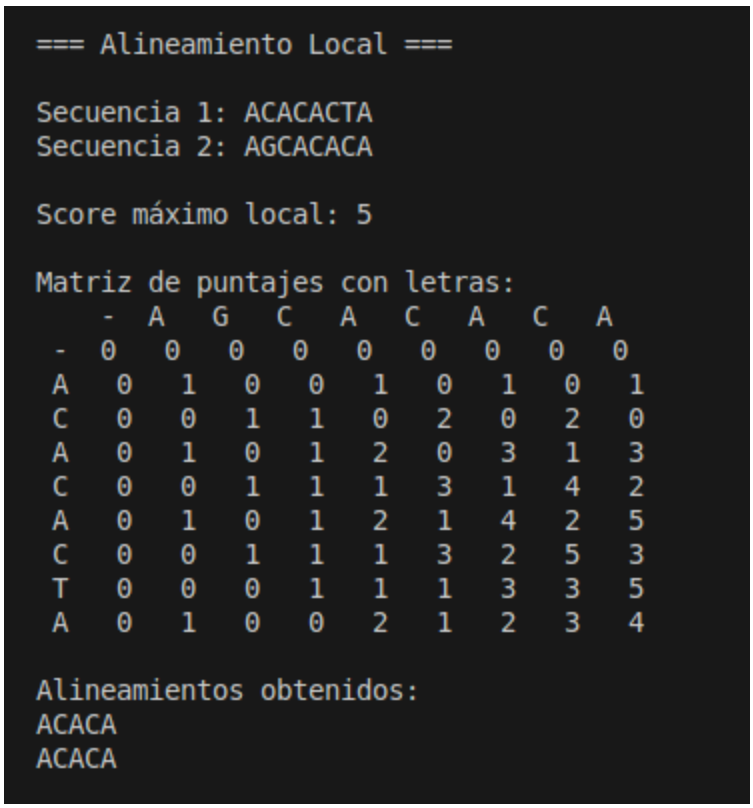
2. Compilar el proyecto:

```
dotnet build
```

3. Ejecutar el proyecto:

```
dotnet run --project GlobalAlignment
```

La salida es por consola indicando el tiempo de ejecución, el score y los alineamineot óptimos, ademas tambien se produce un .txt de salida donde se almacena la matriz de alineamineto global (por defecto resultado.txt), la estructura del .txt es la siguiente:



Cómo ejecutar los tests

Para correr los tests unitarios con xUnit:

```
dotnet test
```

Implementación

El código implementa un algoritmo de **alineamiento local** para dos secuencias de caracteres, utilizando una matriz de puntajes basada en la técnica de programación dinámica. Esta técnica es comúnmente utilizada en bioinformática para encontrar regiones similares entre dos secuencias, como ADN, ARN o proteínas.

Clases principales

- **ScoringCell**: representa cada celda de la matriz de puntajes, contiene la posición `(Row, Column)`, el puntaje acumulado `Score` y un enlace `Antecedent` a la celda precedente que permitió obtener ese puntaje (para reconstruir el alineamiento).
- **AlignmentResult**: estructura para almacenar el resultado del alineamiento, con el puntaje máximo (`Score`) y una lista de alineamientos (`Alignments`), donde cada alineamiento es una tupla con las secuencias alineadas.

Función `Similarity`

Define la regla de puntuación entre dos caracteres:

- Si alguno es un gap (`null`), penaliza con -2.
- Si los caracteres coinciden, suma +1.
- Si no coinciden, suma 0.

Función principal `Align`

1. **Inicialización**:
   Crea una matriz de `ScoringCell` con tamaño `(rows+1) x (cols+1)` inicializada con cero.

2. **Llenado de matriz (DP)**:
   Para cada celda `[i,j]` , calcula:

   - `scoreDiag` : puntaje diagonal (match/mismatch)
   - `scoreUp` : puntaje de eliminar un carácter en la primera secuencia (gap en la segunda)
   - `scoreLeft` : puntaje de eliminar un carácter en la segunda secuencia (gap en la primera)

   Selecciona el máximo entre estos tres y cero (porque el alineamiento local no permite puntajes negativos).
   Actualiza la celda con el puntaje máximo y guarda el antecedente que permitió ese puntaje.

3. **Identificación del puntaje máximo**:
   Guarda la celda con el puntaje más alto, que será el final del mejor alineamiento local.

4. **Reconstrucción del alineamiento**:
   Desde la celda con puntaje máximo, sigue la cadena de `Antecedent` hacia atrás mientras el puntaje sea mayor que cero, insertando caracteres o gaps según el movimiento:

   - Diagonal: ambos caracteres coinciden (o mismatch).
   - Arriba: gap en la segunda secuencia.
   - Izquierda: gap en la primera secuencia.

5. **Salida y guardado**:
   Construye un reporte con:

   - Las secuencias originales.
   - El puntaje máximo.
   - La matriz con puntajes y letras.
   - El alineamiento obtenido.

   Este reporte se escribe en un archivo y se muestra por consola.

6. **Resultado**:
   Devuelve un objeto `AlignmentResult` con el puntaje máximo y la lista con el único alineamiento encontrado.

Función auxiliar `GetMatrixWithLetters`

Genera una representación en texto de la matriz de puntajes, con las letras de las secuencias en la fila superior y en la columna izquierda para facilitar la lectura y análisis del alineamiento.

Resumen

Este código realiza un alineamiento local clásico (similar al algoritmo de Smith-Waterman) con:

- Penalización fija por gaps (-2).
- Puntaje simple de match (+1) y mismatch (0).
- Reconstrucción del mejor alineamiento local.

- Reporte detallado en archivo y consola.

Es ideal para comparar dos secuencias y encontrar la región con mayor similitud local, útil en análisis biológicos y aplicaciones de comparación de strings.

Tests

Este bloque de código contiene pruebas unitarias usando el framework **xUnit** para validar el correcto funcionamiento del método `Align` de la clase `LocalAligner`.

Setup

- La variable `outputFile` indica la ruta del archivo donde se guarda el reporte del alineamiento (`test_output.txt`).
- En el constructor `LocalAlignerTests()`, se elimina el archivo `test_output.txt` antes de cada prueba para asegurar que no haya datos residuales entre tests.

Test 1: `Align_SimpleSequences_ReturnsCorrectScoreAndAlignment`

- **Objetivo:** Verificar que para dos secuencias con similitud parcial, el alineamiento retorne un puntaje positivo y un alineamiento válido.
- **Entradas:**
  - `seq1 = "ACACACTA"`
  - `seq2 = "AGCACACA"`
- **Validaciones:**
  - El puntaje (`Score`) debe ser mayor que 0.
  - Se debe obtener exactamente un alineamiento.
  - Los alineamientos deben tener la misma longitud.
  - La primera secuencia alineada debe contener la letra `'A'` (verifica que la secuencia original está reflejada).
  - La segunda secuencia alineada debe no estar vacía (puede contener caracteres similares o gaps).
  - El archivo de salida debe existir.
  - El contenido del archivo debe incluir la frase `"Score máximo local"` y la secuencia alineada `aligned1`.

Test 2: `Align_SequencesWithNoSimilarity_ReturnsZeroScore`

- **Objetivo:** Validar que cuando las secuencias no tienen similitud alguna, el puntaje máximo sea 0 y el alineamiento refleje esta condición.
- **Entradas:**
  - `seq1 = "AAAAAA"`
  - `seq2 = "GGGGGG"`
- **Validaciones:**
  - El puntaje debe ser exactamente 0.
  - Se debe obtener un alineamiento (aunque puede ser vacío o con gaps).
  - Si el alineamiento tiene longitud, ambas secuencias alineadas deben ser del mismo tamaño.
  - El archivo de salida debe existir.

- El contenido del archivo debe contener `"Score máximo local: 0"`.

Ejemplo de salida en consola al ejecutar los tests con Xunit

resultado.txt

```
=== Alineamiento Local ===

Secuencia 1: ACACACTA
Secuencia 2: AGCACACA

Score máximo local: 5

Matriz de puntajes con letras:
   -  A  G  C  A  C  A  C  A
-  0  0  0  0  0  0  0  0  0
A  0  1  0  0  1  0  1  0  1
C  0  0  1  1  0  2  0  2  0
A  0  1  0  1  2  0  3  1  3
C  0  0  1  1  1  3  1  4  2
A  0  1  0  1  2  1  4  2  5
C  0  0  1  1  1  3  2  5  3
T  0  0  0  1  1  1  3  3  5
A  0  1  0  0  2  1  2  3  4

Alineamientos obtenidos:
ACACA
ACACA

```