Documentación

Mayra Oropeza Condori - 51575

Proyecto final de la materia de Programación Funcional

Universidad Privada Boliviana
Cochabamba, diciembre 2023

Funcionamiento del programa

A continuación, los requerimientos y especificaciones para que el script funcione correctamente.

- Se necesitan las librerías Data.List, Data.Array y Data.Ord. De no estar instaladas en su entorno de ejecución, por favor agréguelas con el siguiente comando:

```
cabal update
cabal install array list ord
```

- Para ejecutar el programa, ejecute la función “main”

Con estas dos acciones, se espera el correcto funcionamiento del proyecto.

Sobre el proyecto

El presente proyecto trata de la aplicación del algoritmo Smith-Waterman para el alineamiento de secuencias biológicas. En base a dicho algoritmo, el programa pide dos secuencias de caracteres, o cadenas de tipo String, y devuelve su matriz de puntuación, además del correspondiente resultado de alineamiento entre estas dos cadenas.

El programa consta de tres partes principales:

- Inicialización, donde el usuario introduce las dos cadenas de caracteres a comparar y se inicializan las variables.
- Llenado de matriz, donde se asignan valores en la matriz de puntuación en función del algoritmo y lo valores de puntuación y penalización asignados por defecto.
- Traceback, donde se construye la secuencia de caracteres resultantes de la alineación en función la puntuación en la matriz.

En las siguientes secciones se explicará a detalle el propósito de cada función y variable dentro del código.

Stage 1: Inicialización

Librerías utilizadas

- `Data.Array`: Utilizada para trabajar con arreglos, en este caso, matrices bidimensionales para representar la matriz de puntuación y la matriz de direcciones en el algoritmo de Smith-Waterman.
- `Data.List (intercalate, maximumBy)`: Utilizada para funciones relacionadas con listas.

    - `intercalate`: Toma una lista de listas y una lista "pegamento", y concatena las listas de la lista original utilizando la lista "pegamento" como separador entre cada par de listas. En el código, se utiliza para formatear la matriz de puntuación al imprimir las filas, separando los elementos con un espacio en blanco.
    - `maximumBy`: Toma una función de comparación y una lista, y devuelve el máximo elemento de la lista según la función de comparación proporcionada. En el código, se utiliza para encontrar la posición con el puntaje máximo en la matriz de puntuación durante el proceso de traceback.
- `Data.Ord (comparing)`: Proporciona funciones relacionadas con tipos ordenables, y en este caso, se utiliza “comparing” para facilitar la comparación de elementos, específicamente junto a “maximumBy” para comparar elementos en función de la puntuación en la matriz de puntuación durante el proceso de traceback.

Variables principales

- `s1` y `s2`: Cadenas de entrada que representan las dos secuencias que se comparan.
- `m` y `n`: Longitudes de las cadenas `s1` y `s2`, respectivamente.
- `bounds`: Tupla que representa los límites de las matrices.
- `scoreMatrix`: Matriz que almacena los puntajes acumulativos del algoritmo de Smith-Waterman.
- `traceback`: Matriz que almacena las direcciones para el traceback en el algoritmo de Smith-Waterman.
- `matchScore`, `mismatchScore`, `gapScore`: Constantes que representan los puntajes para coincidencias, no coincidencias y penalizaciones, respectivamente.
- `alignedS1` y `alignedS2`: Cadenas alineadas resultantes del algoritmo de alineación.

Funciones

- `smithWaterman`: Función principal que realiza el algoritmo de Smith-Waterman para comparar dos secuencias.
- `score`: Calcula el puntaje en una posición específica de la matriz de puntuación.
- `trace`: Determina la dirección en una posición específica de la matriz de direcciones.
- `delta`: Calcula el puntaje para un par de caracteres en las secuencias.
- `tracebackAlignment`: Realiza el traceback para obtener los alineamientos a partir de las matrices de puntuación y direcciones.
- `go`: Función auxiliar para realizar el traceback.

- `maxIndices`: Encuentra las coordenadas de la posición con el puntaje máximo en la matriz de puntuación.
- `print2DScoreMatrix`: Imprime la matriz de puntuación.
- `main`: Programa principal que solicita las secuencias de entrada, realiza el algoritmo y muestra los resultados.

Stage 2: Llenado de la matriz

El código implementa el algoritmo de Smith-Waterman para la alineación local de secuencias, como las de ADN o proteínas. Esto se realiza con el proceso de llenado de la matriz de puntuación, llamando a la función `smithWaterman` y las funciones auxiliares `score` y `trace`.

La matriz de puntuación (`scoreMatrix`) y la matriz de rastreo (`traceback`) se inicializan con las dimensiones apropiadas según las longitudes de las secuencias de entrada `s1` y `s2`. Las coordenadas de la matriz se encuentran en el rango definido por `bounds`.

1. **Puntuación del empate/match (función score):**

   Se calcula la puntuación de la posición `(i, j)` en la matriz de puntuación. Si `i` o `j` son cero, la puntuación es cero, ya que estamos tratando con la alineación local. De lo contrario, se calcula la puntuación tomando el máximo entre cero y tres posibles opciones:

   - Puntuación diagonal (match/mismatch) + Puntuación en la posición `(i-1, j-1)`.
   - Puntuación arriba + Puntuación en la posición `(i-1, j)`.
   - Puntuación a la izquierda + Puntuación en la posición `(i, j-1)`.

2. **Dirección del rastreo (función trace):**

   Se determina la dirección del rastreo en la matriz (`Diag`, `Up`, `LeftDir`, o `Stop`) en función de la puntuación de las opciones anteriores. Si la puntuación máxima se encuentra en la posición diagonal, la dirección es `Diag`; si es arriba, la dirección es `Up`; si es a la izquierda, la dirección es `LeftDir`; de lo contrario, la dirección es `Stop`.

3. **Delta (función delta):**

   Calcula la puntuación de igualdad o desigualdad entre dos elementos en las secuencias `s1` y `s2`. Si los elementos son iguales, la puntuación es `matchScore`; de lo contrario, la puntuación es `mismatchScore`.

4. **Recorrido y llenado de la matriz:**

   Se encuentra en las definiciones de `smithWaterman`, `score`, y `trace`. Donde se recorre la matriz de puntuación comenzando desde `(1, 1)` y se llena utilizando la lógica descrita en

los pasos anteriores para cada posición. Cada celda de la matriz almacena la puntuación máxima alcanzada hasta esa posición.

Este proceso se realiza para todas las posiciones de la matriz de puntuación, y al final, la matriz y la matriz de rastreo están completamente llenas y listas para su uso en el proceso de traceback.

Stage 3: Traceback

Este proceso se realiza en la función `tracebackAlignment`**,** donde se inicia el proceso de traceback desde la posición `(maxIndices scoreMatrix)`, que es la posición que tiene la puntuación máxima en la matriz de puntuación `scoreMatrix`.

1. **Recorrido y construcción de las secuencias alineadas (función go):**

   Se realiza un recorrido en reversa desde la posición de puntuación máxima hasta llegar a la posición `(0, 0)`. En cada paso, se determina la dirección de rastreo desde la matriz de rastreo (`tb`) y se agrega el carácter correspondiente a las secuencias alineadas (`as1` y `as2`).

   - Si la dirección es `Diag`, se agrega el carácter correspondiente de ambas secuencias (`s1` e `s2`) a las secuencias alineadas.
   - Si la dirección es `Up`, se agrega el carácter de `s1` y un guion (`'-'`) a las secuencias alineadas.
   - Si la dirección es `LeftDir`, se agrega un guion (`'-'`) y el carácter de `s2` a las secuencias alineadas.
   - Si la dirección es `Stop`, el proceso de traceback se detiene.

2. **MaxIndices (función maxIndices):**

   Determina las coordenadas de la posición con la puntuación máxima en la matriz de puntuación (`scoreMatrix`). Utiliza la función `maximumBy` para encontrar la coordenada con el valor máximo de puntuación.

3. **Retorno de resultados:**

   Retorna las secuencias alineadas `(as1, as2)` resultantes del proceso de traceback.

Este proceso reconstruye las secuencias originales alineadas a partir de la información almacenada en la matriz de rastreo, comenzando desde la posición con la puntuación máxima en la matriz de puntuación y siguiendo las direcciones indicadas por la matriz de rastreo hasta llegar a la posición `(0, 0)`. Las secuencias alineadas resultantes se utilizan para mostrar el resultado final del algoritmo de Smith-Waterman.

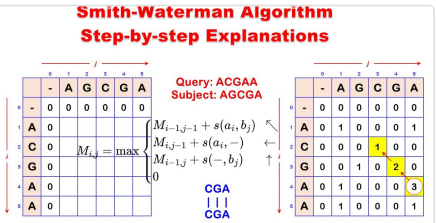
Bibliografía

- Smith-Waterman en Python: https://github.com/slavianap/Smith-Waterman-Algorithm
- Wikipedia: Smith–Waterman_algorithm
- Sobre el algoritmo y Haskell
  - Bio-Alignment-QAlign
  - Bio-Alignment-SAlign
- Youtube:

Smith-Waterman Algorithm — Step-by-step Explanations

Smith-Waterman algorithm is a dynamic programming approach used to solve pairwise local sequence alignment. It involves computing a two-dimensional (2D) matrix using the match, mismatch, and gap penalty

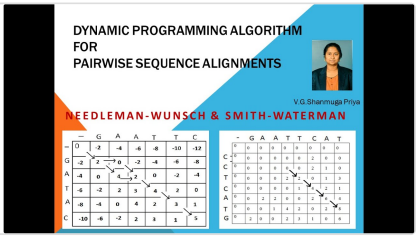
https://www.youtube.com/watch?v=te_csPu5lmM



Dynamic Programming Algorithm for Pairwise Sequence Alignment

In this video, Dynamic Programming algorithms, Needleman–Wunsch algorithm for Global Alignment and Smith–Waterman algorithm for Local Alignment are explained with examples. Here, creating the matrix, formulae

https://www.youtube.com/watch?v=DmBsdH6OfUs



Local Sequence Alignment & Smith-Waterman || Algorithm and Example

Local Sequence Alignment & Smith-Waterman || Algorithm and Example

In this video, we have discussed how to solve the local sequence alignment in

https://www.youtube.com/watch?v=lu9ScxSejSE