Módulo 8 – Algoritmos y estructuras de datos en bioinformática

Curso de Introducción a Estructuras de Datos y Algoritmos

4 de septiembre de 2025

Contents

1 Módulo 8 – Algoritmos y Estructuras de Datos: El Corazón Computacional de la Bioinformática

1.1 Motivación: De la Biología a los Terabytes

La biología del siglo XXI ha experimentado una transformación radical. Hemos pasado de estudiar organismos uno a uno a secuenciar genomas completos en cuestión de horas. Esta revolución ha generado un **diluvio de datos** de una escala sin precedentes. El genoma humano, por ejemplo, es una secuencia de 3.200 millones de caracteres. Un solo experimento de expresión génica puede generar una matriz con cientos de miles de puntos de datos.

La Filosofía Central: En este nuevo paradigma, los datos biológicos son demasiado vastos para la intuición humana. Un biólogo no puede "leer" un genoma para encontrar un gen, de la misma manera que un bibliotecario no puede leer todos los libros de una biblioteca para encontrar una frase. La **bioinformática** nace de esta necesidad: es la

disciplina que desarrolla los métodos computacionales para almacenar, organizar, analizar e interpretar esta inmensa cantidad de información.

Este módulo es la culminación de todo el curso. Aquí veremos cómo los conceptos que hemos estudiado —desde la humilde búsqueda lineal hasta los complejos recorridos de grafos— no son meros ejercicios teóricos, sino las **herramientas indispensables** que permiten hacer descubrimientos científicos. Sin algoritmos eficientes, la genómica moderna, la medicina personalizada y el diseño de fármacos serían, sencillamente, imposibles.

1.2 Procesamiento de Secuencias Biológicas: El Lenguaje de la Vida

La tarea más fundamental en bioinformática es el análisis de secuencias de ADN, ARN y proteínas. La eficiencia aquí no es un lujo, es una necesidad.

1.2.1 Búsqueda de Patrones (Motivos)

- **Problema Real**: No se trata solo de encontrar una subsecuencia. Buscamos "señales" con significado biológico.
  - **Ejemplo 1**: Localizar todas las ocurrencias del codón de inicio `ATG` para predecir dónde comienzan los genes.
  - **Ejemplo 2**: Encontrar secuencias cortas y específicas (ej. `TATAAT`, la "caja TATA") donde las proteínas reguladoras se unen al ADN para iniciar la transcripción.
  - **Ejemplo 3**: Identificar repeticiones de microsatélites (ej. `CAGCAGCAG...`), cuya expansión está asociada a enfermedades como la de Huntington.
- **Estrategias y Estructuras**:
  - **Tablas Hash de *k-mers***:
    - **Idea**: En lugar de escanear el genoma de 3.000 millones de bases cada vez que buscamos un patrón, lo pre-procesamos una sola vez. Se desliza una ventana de tamaño fijo k (ej. k=25) a lo largo de todo el genoma y se almacena cada *k-mer* y su(s) posición(es) en una tabla hash.
    - **Caso Práctico**: Para encontrar dónde se une una proteína que reconoce la secuencia `GATTACA`, en lugar de un escaneo de $O(n)$, simplemente calculamos el hash de `GATTACA` y consultamos la tabla en tiempo $O(1)$ para obtener una lista de todas sus localizaciones. ¡La diferencia entre horas de cómputo y una fracción de segundo!
  - **Árboles de Sufijos y Tries**:
    - **Filosofía**: Son la solución "definitiva" para la búsqueda de patrones. Un árbol de sufijos es una estructura de datos que contiene **todos los sufijos**

de una cadena de una manera comprimida.

* **Analogía**: Imagina tener un índice de un libro que no solo te dice en qué página aparece cada palabra, sino también cada frase, cada párrafo y cada combinación de letras posible. Eso es un árbol de sufijos para una secuencia. Permite responder preguntas complejas como "¿cuál es la subsecuencia más larga que se repite en este genoma?" en tiempo proporcional a la longitud del patrón, no del genoma.

1.2.2 Comparación de Secuencias (Alineamiento)

- **Problema Real**: ¿Cómo cuantificamos la similitud entre dos genes? Esto es crucial para inferir funciones (si un gen desconocido es 80% similar a un gen conocido, probablemente tengan funciones parecidas) o para trazar relaciones evolutivas.
- **Algoritmos basados en Programación Dinámica**:
  - **La Idea**: El alineamiento se visualiza como encontrar el "mejor camino" a través de una matriz donde un eje es la secuencia A y el otro la secuencia B. Cada celda (i, j) de la matriz almacena la puntuación del mejor alineamiento posible entre el prefijo $A[1..i]$ y $B[1..j]$.
  - **Analogía**: Es como un juego de mesa donde puedes moverte en diagonal (match/mismatch), hacia abajo (gap en A) o a la derecha (gap en B). Cada movimiento tiene una puntuación. La programación dinámica garantiza que, para calcular la puntuación de una celda, ya hemos calculado las puntuaciones óptimas de las celdas de las que depende, evitando volver a calcular nada.
  - **Needleman-Wunsch vs. Smith-Waterman**: El primero encuentra el mejor alineamiento **global** (de principio a fin), ideal para comparar dos genes que se cree que están relacionados en toda su longitud. El segundo encuentra la mejor región de similitud **local**, perfecto para descubrir dominios funcionales compartidos dentro de proteínas más grandes.
- **Heurísticas como BLAST**:
  - **Motivación**: Un alineamiento con programación dinámica entre dos genomas es computacionalmente inviable. BLAST (*Basic Local Alignment Search Tool*) es una heurística ingeniosa que cambia la pregunta de "¿cuál es el mejor alineamiento?" a "¿existen regiones de alta similitud?".
  - **Funcionamiento**: BLAST utiliza **hashing de k-mers** para encontrar "semillas" (matches exactos muy cortos) entre las dos secuencias. Luego, intenta extender estas semillas en ambas direcciones para construir un alineamiento local más largo y significativo. Es un compromiso brillante: sacrifica la garantía de optimalidad por una velocidad miles de veces mayor.

1.3 Ordenación: Poniendo en Fila los Datos Biológicos

La ordenación es un paso de pre-procesamiento omnipresente. * **Caso Práctico 1: Análisis de datos de secuenciación (NGS)**. Un experimento produce cientos de millones de lecturas de ADN cortas (*reads*). Antes de mapearlas contra un genoma de referencia, es útil ordenarlas. ¿Por qué? Un lote de lecturas ordenadas lexicográficamente puede ser procesado de forma más eficiente por los algoritmos de mapeo debido a una mejor localidad de caché y patrones de acceso a memoria. * **Caso Práctico 2: Análisis de expresión génica**. Se tiene una matriz donde las filas son genes y las columnas son pacientes. Ordenar los genes por su nivel de expresión promedio permite identificar rápidamente los genes más y menos activos en una condición particular (ej. un tipo de cáncer).

- **Algoritmos aplicados**:
  - **Merge Sort Externo**: Cuando el archivo de lecturas de NGS ocupa 100 GB y solo tienes 16 GB de RAM, es imposible cargarlo todo en memoria. Merge Sort es el algoritmo natural aquí: lee un trozo del archivo que quepa en memoria, lo ordena con Quicksort, lo escribe en un archivo temporal. Repite esto hasta procesar todo el archivo. Finalmente, fusiona todos los archivos temporales ordenados en un único archivo final ordenado.
  - **Radix Sort**: Para ordenar secuencias de ADN, Radix Sort puede ser mucho más rápido que Quicksort. ¿Por qué? Porque el "alfabeto" es muy pequeño ({A, C, G, T}). Radix Sort ordena las secuencias basándose en el último carácter, luego el penúltimo, y así sucesivamente. Su complejidad $O(d \cdot n)$ (donde d es la longitud de la secuencia) supera a $O(n \log n)$ en este contexto específico.

1.4 Grafos: La Estructura de las Redes Biológicas

La biología está llena de redes, y los grafos son el lenguaje natural para describirlas.

- **Redes de Interacción Proteína-Proteína (PPI)**:
  - **Modelo**: Los nodos son proteínas, las aristas indican que dos proteínas interactúan físicamente.
  - **Análisis**: Se pueden aplicar algoritmos de grafos para encontrar "módulos" o clústeres de proteínas densamente conectadas, que a menudo corresponden a complejos moleculares con una función biológica común. Un **DFS** puede encontrar todos los miembros de una ruta de señalización.
  - **Caso Práctico**: Descubrir que una proteína asociada a una enfermedad es un "hub" (un nodo con muchas conexiones) en la red puede identificarla como una diana terapéutica prometedora.
- **Grafos de Ensamblado de Genomas**:

- **El Problema**: Reconstruir un libro de 3.000 millones de letras a partir de millones de fragmentos de 150 letras que se solapan.
- **La Solución (Grafos de Bruijn)**: En lugar de tratar cada fragmento como un nodo (lo que sería muy complejo), los nodos son todos los *k-mers* (ej. subsecuencias de 30 letras) presentes en los fragmentos. Se dibuja una arista dirigida desde el k-mer A al k-mer B si se solapan en k-1 letras.
- **La Magia**: El problema biológico de "reconstruir la secuencia" se transforma en el problema clásico de la teoría de grafos de "encontrar un camino Euleriano" (un camino que visita cada arista exactamente una vez). ¡Un problema aparentemente intratable se convierte en uno bien estudiado!

1.5 Árboles: Jerarquías en Evolución y Datos

- **Árboles Filogenéticos**:
  - **Modelo**: Representan la historia evolutiva. Las hojas son las especies actuales y los nodos internos son los ancestros comunes. La longitud de las ramas puede representar el tiempo evolutivo o la cantidad de cambio genético.
  - **Construcción**: Se parte de una matriz de distancias (ej. calculada alineando un gen entre todas las especies). Algoritmos como **UPGMA** (un método de clustering jerárquico) agrupan iterativamente las dos especies más cercanas, creando un nuevo nodo ancestro, hasta que se forma el árbol completo.
- **BST y sus variantes en Bases de Datos**:
  - **Caso Práctico**: Bases de datos como Ensembl o GenBank almacenan información sobre millones de genes. Cuando un investigador busca "el gen BRCA1 en humanos", el sistema no puede permitirse una búsqueda lineal. Internamente, utiliza estructuras de datos avanzadas como los **Árboles B+** (una variante de BST optimizada para disco) para indexar los datos por nombre de gen, especie, posición cromosómica, etc., permitiendo recuperaciones casi instantáneas.

1.6 Conclusiones Finales: La Caja de Herramientas del Bioinformático

Este módulo demuestra que la bioinformática no es simplemente "aplicar programas a datos biológicos". Es el campo donde la teoría de la computación se encuentra con los desafíos más fundamentales de la biología.

- **La Eficiencia es Descubrimiento**: La diferencia entre un algoritmo $O(n^2)$ y uno $O(n \log n)$ puede ser la diferencia entre un análisis que tarda un siglo y uno que

tarda un día. Un algoritmo más rápido puede permitir explorar hipótesis que antes eran inabordables.

- **La Abstracción es Poder**: La capacidad de ver un problema de ensamblado de genomas como un camino en un grafo, o un problema de búsqueda de patrones como una consulta a una tabla hash, es la habilidad central de un bioinformático.
- **No hay "Bala de Plata"**: No existe un único algoritmo o estructura de datos que resuelva todo. La elección correcta depende críticamente de la escala de los datos, la pregunta biológica y las limitaciones de hardware. Un buen bioinformático conoce su caja de herramientas y sabe cuándo usar un martillo (Quicksort), un destornillador (BFS) o una llave inglesa de precisión (un árbol de sufijos).

1.7 Ejercicios de autoevaluación

1. Diseña un pseudocódigo para buscar un motivo de longitud 3 en una secuencia de ADN usando **tablas hash de k-mers** (apóyate en el Módulo 6).

2. ¿Qué ventajas tiene **radix sort** sobre quicksort para ordenar secuencias de nucleótidos? (revisa Módulo 7).

3. Explica cómo un **grafo de Bruijn** ayuda en el ensamblado de genomas (conecta con Módulo 5).

4. Construye un pequeño **árbol filogenético** a partir de tres secuencias ficticias.

5. El alineamiento de dos secuencias de 1 millón de bases con Needleman–Wunsch es inviable ($O(n^2)$). ¿Qué estrategias alternativas existen?

1.8 Referencias

- Jones, N. C., & Pevzner, P. A. *An Introduction to Bioinformatics Algorithms*. MIT Press.

- Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. *Biological Sequence Analysis*. Cambridge University Press.

- Compeau, P., & Pevzner, P. *Bioinformatics Algorithms: An Active Learning Approach*. Active Learning Publishers.
