Entrega 3 Proyecto

Estructuras de datos

Grupo 2

Integrantes: Santiago Ibáñez Niño - David Santiago Calderón Idárraga

1. **Acta de evaluación.**
   La entrega anterior realizada (2), solo tuve un error el cual fue en la documentación, debido a que en el diagrama de TADs no se incluyó la relación que nodo tenía consigo mismo, error el cual fue corregido para esta entrega.
2. **Definición de TADs del sistema de archivos FASTA**
   a. TAD Secuencia

      Datos mínimos:

      - Nombresecuencia, Cadena de caracteres, Identifica la secuencia por ejemplo ">Full_SEQUENCE".
      - Lineas, Lista de caracteres, las bases que están en el archivo Fasta como: "CTCCGGTGAGAAATTTTGGGATGTATCAAATCACGGTCCTACTAC".
      - anchoLinea, número entero, la cantidad de columnas por línea usadas para justificar las líneas.
      - Completo, Booleano, Determina si la secuencia que se está completa o es una parte.

      Operaciones:

      - ObtenerNombre(): retorna el nombre/descripción de la secuencia.
      - ObtenerLongitud(): retorna la cantidad total de bases (sin contar "-").
      - EsCompleta(): indica si la secuencia está completa o no.
      - ContarBase(b): retorna cuántas veces aparece una base b en la secuencia.

   b. TAD Histograma

      Datos mínimos:

      - codigo, arreglo de caracteres, representan en letras las bases nitrogenadas del ADN.
      - Resultado, arreglo de enteros, representa la cantidad de caracteres o bases nitrogenadas que se pueden encontrar en una secuencia.

      Operaciones:

      - OrdenSeguntabla(): verifica el orden de las bases en una tabla de prioridad de bases dada por el enunciado.

   c. TAD Genoma

      Datos Mínimos:

      - Conjunto, Lista de secuencias, agrupa las secuencias de un genoma del archivo Fasta.
      - CantSecuencias, entero, contador de cuantas secuencias hay en el genoma.

Operaciones:

- ListarSecuencias(): Entra al conjunto de secuencias del genoma, y muestra por pantalla la cantidad de secuencias y cuantas bases contiene cada una de ellas.
- ObtenerSecuencias(nombre): Se encarga de hallar el histograma de la secuencia que recibe, teniendo en cuenta la tabla de ordenamiento.
- ExisteSubSecuencia(sub): Busca en el genoma, si una serie de bases ingresadas por el usuario existen en alguna de las secuencias, y si existen cuantas veces se repiten.
- EnmascararSubsecuencia(sub): Remplaza la subsecuencia proporcionada por el usuario por el carácter 'X'.
- Ruta_mas_corta(nomSec, i, j, x, y): Calcula la ruta mas corta entre las dos coordenadas recibidas siendo la coordena origen (i, j) y la coordenada de destino (x, y).
- Base_remota(i, j): Busca la base del mismo tipo mas lejana en la secuencia desde una coordenada de origen (x, y).

d. TAD Archivo

Datos Mínimos:

- Ruta, cadena de caracteres, indica la ruta y el nombre del archivo que se está trabajando.

Operaciones:

- leer(ruta): muestra nombre y número de x bases de cada secuencia cargada.
- guardarArchivo(genoma, nombreArchivoGuardado): Guarda en el archivo nombre_archivo las secuencias cargadas en memoria. Se debe teniendo en cuenta el ancho de línea(justificación).

e. TAD Menu

Datos Minimos:

- Comandospermitidos, arreglo de cadena de caracteres, contiene todos los comandos posibles y ejecutables por el programa para manipular los archivos Fasta.

Operaciones:

- ProcesarComando(linea, genoma): Determina que acción va a tomar el programa según lo que haya solicitado el usuario y sea válido.
- AyudaGeneral(): Muestra todos los comandos disponibles con sus respectivos parametros.
- AyudaComando(comando): Da información más específica de cada comando posible del sistema.
- ProcesarAyuda(parámetro): Determina si la ayuda solicitada fue general o especifica por un comando.

f. TAD Arbol Huffman

Datos Minimos:

- Comandospermitidos, arreglo de cadena de caracteres, contiene todos los comandos posibles y ejecutables por el programa para manipular los archivos Fasta.

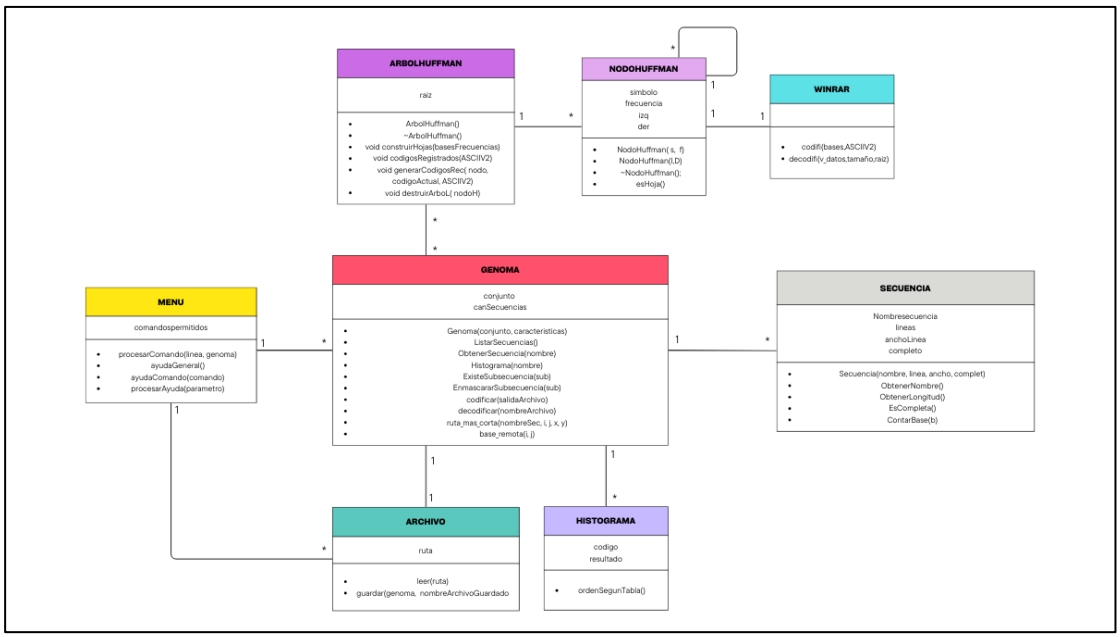
Operaciones:

- ProcesarComando(linea, genoma): Determina que acción va a tomar el programa según lo que haya solicitado el usuario y sea válido.

g. TAD Nodo Huffman

NodoHuffman *izq;
NodoHuffman(char s, int f);
NodoHuffman(I,D):
~NodoHuffman():
bool esHoja():

3. Diagrama de TADS



El diagrama de TADS muestra un sistema bien estructurado para la gestión y análisis de genomas, donde cada clase cumple un papel específico y se relaciona de forma coherente con las demás. El menú funciona como interfaz de interacción, el genoma concentra la lógica principal de manejo de secuencias, mientras que la clase secuencia abstrae la información genética en unidades manejables. Además, el archivo asegura la persistencia de datos y el histograma organiza los resultados de los análisis. En conjunto, el diseño refleja un enfoque modular y claro que facilita la reutilización, el mantenimiento y la comprensión del sistema.

4. Plan de Pruebas – Comando base_remota

Objetivo

Verificar que el comando base_remota localice correctamente, dentro de la matriz de una secuencia, la base que sea idéntica (mismo carácter) a la base de origen y cuyo camino mínimo desde el origen tenga el costo más alto posible, mostrando sus coordenadas, la ruta y el costo total.

Casos de Prueba

Caso 1: Secuencia no encontrada

i. **Entrada:** base_remota SecuenciaInexistente 0 0
ii. **Condición inicial:** Se han cargado secuencias, pero ninguna se llama "SecuenciaInexistente".
iii. **Salida esperada:** "La secuencia SecuenciaInexistente no existe."
iv. **Criterio de Éxito:** El sistema identifica que el nombre no corresponde a ninguna secuencia en memoria y muestra el error sin colapsar.

Caso 2: Posición de base inválida (Fuera de rango)

i. **Entrada:** base_remota Full_SEQUENCE -1 500
ii. **Condición inicial:** Existe la secuencia Full_SEQUENCE pero las coordenadas exceden las dimensiones de la matriz (filas o columnas).
iii. **Salida esperada:** "La base en la posición [-1,500] no existe."
iv. **Criterio de Éxito:** El sistema valida los límites de la matriz antes de iniciar el algoritmo de búsqueda.

Caso 3: No existen otras bases idénticas alcanzables

i. **Entrada:** base_remota SecuenciaUnica 0 0
ii. **Condición inicial:** La secuencia cargada tiene una base en [0,0] (ej. 'Z') que no se repite en ninguna otra parte de la secuencia, o las otras ocurrencias están aisladas (no hay camino).
iii. **Salida esperada:** "No se encontró ninguna base remota alcanzable idéntica a la original."
iv. **Criterio de éxito:** El sistema recorre la matriz, no encuentra otro carácter igual con distancia válida y notifica al usuario.

Caso 4: Base remota encontrada exitosamente

i. **Entrada:** base_remota Full_SEQUENCE 0 0
ii. **Condición inicial:** La secuencia tiene múltiples ocurrencias de la base ubicada en [0,0] distribuidas en la matriz.
iii. Salida esperada:

a. "Para la secuencia Full_SEQUENCE, la base remota está ubicada en [F, C]." (Donde F,C son coordenadas).
b. "La ruta entre la base en [0,0] y la base remota en [F,C] es: A, T, G, ... , A"
c. "El costo total de la ruta es: X.XX"

iv. **Criterio de éxito:** El sistema calcula correctamente la base del mismo tipo con el mayor costo de camino mínimo, imprime sus coordenadas correctas, la secuencia de pasos y el costo numérico.

Criterios de Éxito Globales

1. El sistema valida la existencia de la secuencia y la validez de las coordenadas antes de procesar.
2. La base destino encontrada es siempre del mismo tipo (letra) que la base origen.
3. La ruta mostrada corresponde efectivamente al camino de costo calculado usando Dijkstra.

5. **Diagrama de flujo comandos**

INICIO

↓

Mostrar menú de comandos disponibles

↓

Usuario ingresa comando

↓

| EVALUAR COMANDO |

↓

[$cargar archivo]

→ menu.cxx recibe el comando

→ Llama a Archivo::leerArchivo(nombreArchivo)

→ Archivo.cxx:

- Abre archivo FASTA

- Crea objetos Secuencia (id + cadena ADN)

- Envía cada Secuencia a Genoma::agregarSecuencia()

→ Genoma guarda todas las secuencias en memoria

[$listar_secuencias]

→ menu.cxx recibe el comando

→ Llama a Genoma::mostrarSecuencias()

→ Genoma:

- Recorre vector de secuencias

- Imprime el nombre y la cantidad de bases

[$histograma]

→ menu.cxx recibe el comando

→ Llama a Histograma::generar(genoma)

→ Histograma.cxx:

- Recorre todas las secuencias en Genoma

- Cuenta ocurrencias de A, C, G, T

- Calcula frecuencias

- Muestra histograma textual en consola

[$es_subsecuencia]

→ menu.cxx recibe el comando con parámetros

→ Llama a Secuencia::esSubsecuencia(otra)

→ Secuencia.cxx:

- Verifica si la cadena de "otra" está contenida en la secuencia

→ Muestra en consola la cantidad de veces que está contenida

[$enmascarar]

→ menu.cxx recibe el comando con parámetros

→ Llama a Secuencia::enmascarar

→ Busca la subsecuencia en las secuencias de Genoma

→ Secuencia.cxx:

- Sustituye caracteres de la región seleccionada por 'N'

- Actualiza la secuencia en Genoma

→ Confirmación en consola

[$ruta_mas_corta]

→ menu.cxx recibe el comando con parámetros (descripción, i, j, x, y)

→ Llama a Genoma::ruta_mas_corta(descripcion, i, j, x, y)

→ Genoma.cxx:

- Busca la secuencia en memoria por su nombre

- Convierte la lista de bases a una matriz (Secuencia::obtenerMatriz)

- Valida que las coordenadas [i,j] y [x,y] existan

- Ejecuta algoritmo de Dijkstra para hallar el camino de costo mínimo

→ Imprime en consola la ruta de bases y el costo total

[$base_remota]

→ menu.cxx recibe el comando con parámetros (descripción, i, j)

→ Llama a Genoma::base_remota(descripcion, i, j)

→ Genoma.cxx:

- Busca la secuencia en memoria y genera su matriz

- Ejecuta Dijkstra desde el origen [i,j] hacia todos los nodos

- Recorre la matriz de costos para encontrar la base del mismo tipo (letra) con la mayor distancia

→ Imprime en consola las coordenadas de la base remota, la ruta para llegar y el costo

[$guardar archivo]

→ menu.cxx recibe el comando

→ Llama a Archivo::guardarArchivo(nombreSec, genoma)

→ Archivo.cxx:

- Recorre todas las secuencias del Genoma

- Escribe en formato FASTA:

>id

secuencia

- Cierra archivo

→ Mensaje de confirmación en consola

[$ayuda]

→ menu.cxx recibe el comando

→ Recorre la lista de comandos permitidos

→ Imprime en consola la lista y descripción breve

[$salir]

→ menu.cxx recibe el comando

→ Finaliza el bucle principal

→ Libera memoria (si aplica)

→ Termina ejecución

→ FIN