Entrega 0 Proyecto

Estructura de Datos 2530

Integrantes: Santiago Ibáñez Niño y David Santiago Calderón Idárraga

Objetivo del Proyecto

El objetivo principal de este proyecto es desarrollar un sistema interactivo en consola que permita al usuario realizar un manejo de archivos genéticos en formato FASTA, haciendo uso de los conceptos fundamentales de estructuras de datos. El sistema deberá ser capaz de cargar, listar, analizar, modificar, guardar, comprimir y descomprimir las secuencias genéticas almacenadas en dichos archivos

Para ello, el proyecto se estructura en tres componentes esenciales:

1. **Manejo básico de secuencias genéticas:** Esto la carga y visualización de secuencias, enmascaramiento y almacenamiento en disco de los cambios realizados.
2. **Compresión y descompresión de archivos FASTA:** Por medio del algoritmo de Huffman, que aprovecha estructuras jerárquicas (árboles binarios) para optimizar el almacenamiento, codificando y decodificando la información.
3. **Representación y análisis de relaciones entre bases utilizando grafos:** Las bases se modelan como nodos conectados con sus vecinos. Esto permitirá calcular rutas más cortas entre bases o encontrar la base más lejana de un mismo tipo, empleando diferencias de códigos ASCII.

Requerimientos del Proyectos

Se deben tener en cuenta los siguientes requerimientos con el fin de garantizar un correcto funcionamiento entre la interfaz y el usuario.

- Detectar y validar comandos ingresados por el usuario
- Verificar el formato de los datos de entrada, incluyendo la cantidad, los tipos de datos.
- Rechazar nombres o comandos que violen el formato establecido.
- Garantizar el sistema vuelva automáticamente a mostrar el indicador de '$' para recibir la siguiente instrucción o comando.
- La visualización de los archivos de salida sean ordenandos y respeten la estructura de secuencia genética.
- Establecer un comando de "ayuda" con el fin de generar una guia al usuario sobre que comandos o procesos puede realizar en un momento dado
- Se debe mostrar mensajes de error claros si un comando se ingresa de forma incorrecta. Estos mensajes se activarán en tres escenarios: cuando el comando tiene **parámetros faltantes**, cuando se introduce un **comando no establecido** o cuando se proveen **datos en exceso**. Cada mensaje debe guiar al usuario para que entienda qué salió mal.
- presentar en pantalla los mensajes de resultado (éxito o error) además de otros mensajes necesarios que permitan al usuario saber, por un lado, cuando terminó el comando su procesamiento, y por el otro lado, el resultado de ese procesamiento

Líneas de Comando

En el proyecto, la interacción fundamental del usuario con el programa se realiza a tra ves de líneas de comando, las cuales tienen un formato especifico de escritura para garantizar la validación de lo solicitado por el usuario. Por ejemplo, cada línea de comando tiene un indicar que está representado por el signo de dólar '$'. Tanto como el sistema analizará y ejecutará el comando dependerá de la correcta implementación de las instrucciones disponibles para el usuario. Esto garantiza un correcto funcionamiento del sistema. A continuación, se especificarán las reglas de formato del conjunto de comandos para una correcta interacción e implementación por parte del usuario:

- cargar
  - **Sintaxis:** $ cargar nombre_archivo
  - **Parámetros:** nombre_archivo
  - **Descripción detallada:** Cargar la información (secuencias) del archivo especificado.
  - **Ejemplos:**
    - ***Caso de exito:***
      $ cargar archivo_ejemplo.fa
      2 secuencias cargadas correctamente desde archivo_ejemplo.fa.
    - ***Caso de fracaso (archivo no encontrado):***
      $ cargararchivo_inexistente.fa
      Error: el archivo archivo_inexistente.fa no se encuentra o no puede leerse.
    - ***Caso de fracaso (archivo vacío):***
      $ cargar archivo_vacio.fa
      Error: el archivo archivo_vacio.fa no contiene ninguna secuencia.

- listar_secuencias
  - **Sintaxis:** $ listar_secuencias
  - **Parámetros:** *No aplica*
  - **Descripción detallada:** Muestra por pantalla las secuencias en memoria, mostrando la información necesaria de la secuencia (nombre, numero de bases).
  - **Ejemplos:**
    - ***Caso de éxito:***
      $ listar_secuencias
      Secuencia Incomplete_sequence contiene al menos 378 bases.
    - ***Caso de fracaso (sin secuencias cargadas):***
      $ listar_secuencias
      Error: No hay secuencias cargadas en memoria.

- histograma
  - **Sintaxis:** $ histograma descripcion_secuencia
  - **Parámetros:** descripcion_secuencia
  - **Descripción detallada:** Muestra por pantalla la frecuencia de aparición de cada código en la secuencia indicada.

- **Ejemplos:**
  - ***Caso de éxito:***
    $ histograma Full_SEQUENCE
    A: 20 C: 25 G: 22 T: 18
  - ***Caso de fracaso (secuencia inexistente):***
    $ histograma Secuencia_No_Existente
    Error: Secuencia inválida.

- es_subsecuencia
  - **Sintaxis:** $ es_subsecuencia subsecuencia
  - **Parámetros:** subsecuencia
  - **Descripción detallada:** Alerta al usuario si la subsecuencia, y la cantidad de apariciones en memoria.
  - **Ejemplos:**
    - ***Caso de éxito:***
      $ es_subsecuencia CTCCG
      La subsecuencia dada se repite 5 veces dentro de las secuencias cargadas en memoria.
    - ***Caso de fracaso (subsecuencia no encontrada):***
      $ es_subsecuencia CTCCG
      La subsecuencia dada no existe dentro de las secuencias cargadas en memoria.
    - ***Caso de fracaso (sin secuencias cargadas):***
      $ es_subsecuencia CTCCG
      No hay secuencias cargadas en memoria

- enmascarar
  - **Sintaxis:** $ enmascarar subsecuencia
  - **Parámetros:** subsecuencia
  - **Descripción detallada:** Cada subsecuencia en la memoria es remplazada por ‘X’.
  - **Ejemplos:**
    - ***Caso de éxito:***
      $ enmascarar TCCCTTC
      3 subsecuencias han sido enmascaradas dentro de las secuencias cargadas en memoria.
    - ***Caso de fracaso (subsecuencia no encontrada):***
      $ enmascarar TCCCTTC
      No hay secuencias cargadas en memoria
    - ***Caso de fracaso (subsecuencia no encontrada):***
      $ enmascarar XYZ
      La subsecuencia dada no existe dentro de las secuencias cargadas en memoria, por tanto, no se enmascara nada.

- guardar
    - **Sintaxis:** $ guardar nombre_archivo
    - **Parámetros:** nombre_archivo
    - **Descripción detallada:** guarda todas las secuencias del archivo Fasta en su debido formato.
    - **Ejemplos:**
        - ***Caso de éxito:***
          $guardar genomas_modificado.fa
          Las secuencias han sido guardadas en genomas_modificado.fa.
        - ***Caso de fracaso (sin secuencias cargadas):***
          $ guardar nuevo_archivo.fa
          No hay secuencias cargadas en memoria.
        - ***Caso de fracaso (problemas de escritura):***
          $guardar archivo.fa
          Error guardando en /ruta/protegida/archivo.fa.

- salir
    - **Sintaxis:** $ salir
    - **Parámetros:** *No aplica*
    - **Descripción detallada:** El usuario ya no realizara alguna otra accion sobre el sistema.
    - **Ejemplos:**
        - ***Caso de éxito:***
          $salir (El programa finaliza sin mostrar una salida en pantalla).

- codificar
    - **Sintaxis:** $ codificar nombre_archivo.fabin
    - **Parámetros:** nombre_archivo.fabin
    - **Descripción detallada:** Utilizando el algoritmo de huffman, el sistema codifica y guarda en un archivo binario las secuencias.
    - **Ejemplos:**
        - ***Caso de exito:***
          $codificar genomas.fabin
          Secuencias codificadas y almacenadas en genomas.fabin.
        - ***Caso de fracaso (sin secuencias cargadas):***
          $ codificar genomas_vacios.fabin
          No hay secuencias cargadas en memoria.
        - ***Caso de fracaso (problemas de escritura):***
          codificar genoma.fabin
          No se pueden guardar las secuencias cargadas en genoma.fabin.

- decodificar

- **Sintaxis:** $ decodificar nombre_archivo.fabin
- **Parámetros:** nombre_archivo.fabin
- **Descripción detallada:** Decodificada el binario obtenido y recarga las secuencias en memoria.
- **Ejemplos:**
  - ***Caso de exito:***
    $ decodificar genomas.fabin
    Secuencias decodificadas desde genomas.fabin y cargadas en memoria.
  - ***Caso de fracaso (archivo no encontrado o corrupto):***
    $ decodificar archivo_invalido.fabin
    No se pueden cargar las secuencias desde archivo_invalido.fabin.

- ruta_mas_corta
  - **Sintaxis:** $ruta_mas_corta descripcion_secuencia i j x y
  - **Parámetros:** descripcion_secuencia, índices i,j,x,y
  - **Descripción detallada:** El sistema calcula la reutra de mejor costo entre los vertices de la matriz de secuencias.
  - **Ejemplos:**
    - ***Caso de exito:***
      $ruta_mas_corta Full_SEQUENCE 1 3 1 7
      Para la secuencia Full_SEQUENCE, la ruta más corta entre la base C en [1,3] y la base G en [1,7] es: C -> G -> G -> A -> A El costo total de la ruta es: 0.85
    - ***Caso de fracaso (secuencia inexistente):***
      $ ruta_mas_corta Secuencia_No_Existe 1 3 1 7
      La secuencia Secuencia_No_Existe no existe.
    - ***Caso de fracaso(posición inválida):***
      $ ruta_mas_corta Full_SEQUENCE 50 3 1 7
      La base en la posición [50,3] no existe.

- base_remota
  - **Sintaxis:** $ base_remota descripcion_secuencia i j
  - **Parámetros:** descripcion_secuencia, índices i,j
  - **Descripción detallada:** busca la ubicación de la base más lejana que sea idéntica a la base en una posición específica de la matriz de una secuencia. Luego, imprime la ubicación de esa base remota, incluyendo datos como el costo.
  - **Ejemplos:**
    - ***Caso de exito:***
      $ base_remota Full_SEQUENCE 1 3
      Para la secuencia Full_SEQUENCE, la base remota está ubicada en [3, 15], y la ruta entre la base en [1,3] y la base remota en [3,15] es: ...

El costo total de la ruta es: 1.2

- ***Caso de fracaso (secuencias inexistente):***
  $ base_remota Secuencia_No_Existe 1 3
  La secuencia Secuencia_No_Existe no existe.
- ***Caso de fracaso (posición inválida):***
  $base_remota Full_SEQUENCE 50 3
  La base en la posición [50,3] no existe.

- ayuda
  - **Sintaxis:** $ ayuda
  - **Parámetros:** *No aplica*
  - **Descripción detallada:** Muestra una lista de los comandos principales en el inicio del programa y su funcionamiento.
  - **Ejemplos:**

    - ***Caso de exitó:***
      $ ayuda
      Lista de comandos disponibles:
      cargar, listar_secuencias, histograma, es_subsecuencia, enmascarar, guardar, salir, codificar, decodificar, ruta_mas_corta, base_remota, ayuda.

- ayuda
  - **Sintaxis:** $ ayuda <comando>
  - **Parámetros:** *comando*
  - **Descripción detallada:** Muestra una guía sobre el uso y funcionamiento de el comando solicitado.
  - **Ejemplo:**
    - **Caso de éxito:**
      $ ayuda cargar
      Sintaxis: cargar nombre_archivo
      Parámetros: nombre_archivo
      Descripción: Carga las secuencias del archivo .fa especificado.

Conclusión

El proyecto describe la creación de un sistema interactivo en consola para gestionar archivos genéticos en formato FASTA, utilizando conceptos de estructuras de datos. El sistema permite cargar, listar, analizar y modificar secuencias genéticas, además de comprimir y descomprimir archivos usando el algoritmo de Huffman. También incorpora un análisis de relaciones entre bases genéticas utilizando grafos, lo que permite calcular la ruta más corta o encontrar la base más lejana.

Se ha establecido un conjunto claro de comandos y sus sintaxis para una interacción precisa con el usuario. La gestión de errores está bien definida, con mensajes específicos para escenarios como

comandos no establecidos, falta de parámetros o exceso de datos. Esto asegura que el sistema sea robusto y guíe al usuario de manera efectiva, evitando interrupciones inesperadas.