Proyecto 2530 – Estructuras de Datos

Oscar Samuel Pinilla Alvira[1] Samuel Andres Rey Marquez[1] and Juan Manuel Rivera Pabón[1]

[1] Pontificia Universidad Javeriana, Bogotá D.C, Colombia
samuelareym@javeriana.edu.co
jmanuelriverap@javeriana.edu.co
Pinillaa.osamuel@javeriana.co

Abstract. This paper incldes all the stages of the Data Structures class, a little explanation of the context, and all the development of the DNA reading, writing, compressing system

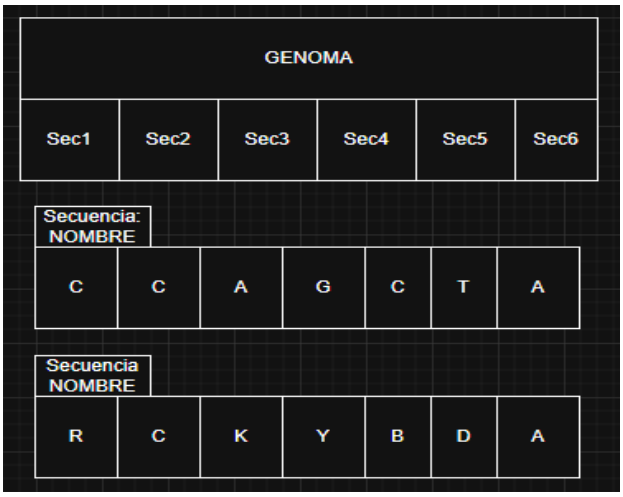
Keywords: Código genético, secuencias de bases, gestión de datos.

1 Descripción del problema

En el campo de la biología y la genética, se ha logrado reflejar la información que caracteriza a los organismos vivos, con los conceptos de genoma, código genético y bases nitrogenadas.

El genoma se define como toda la información genética de un ser vivo, es decir todo su ADN y ARN. Se compone de una lista de secuencias, llamadas código genético. Las secuencias están compuestas de letras que corresponden a bases nitrogenadas, cada una con una representación específica: (A) Adenina, (C) Citosina, (T) Timina, (G) Guanina para el caso del ADN y (A) Adenina, (C) Citosina, (T) Timina, (U) Uracilo para el caso del ARN.
Para cuestiones de estudios, la necesidad de herramientas que permitan la gestión, que faciliten el análisis y modificación de códigos genéticos se ha hecho cada vez más evidente. Los archivos FASTA son archivos con formato de texto plano que han surgido como una alternativa para manejar la información más fácilmente, que permite almacenar un genoma, incluyendo sus distintas secuencias de bases nitrogenadas, junto a una descripción para identificarlo. Lo que se requiere es la implementación de un sistema que permita modificar, interpretar y realizar análisis a partir de la lectura y escritura de archivos FASTA

- Representación gráfica para facilitar la interpretación del problema:



- Las 4 bases nitrogenadas principales, se mencionaron en el texto inicial, sin embargo, se han generado una serie de letras para representar casos especiales, por ejemplo, cuando no se tiene certeza de cual es la base que ocupa un espacio en la secuencia. Las letras usadas son:

  K: G/T
  S: G/C
  Y: T/C
  M: A/C
  W: A/T
  R: G/A
  B: G/T/C
  D: G/A/T
  H: A/C/T
  V: G/C/A
  -: gap of indeterminate length

- Los archivos fasta tienen un formato específico, a partir del siguiente ejemplo, se caracteriza el archivo:

```
>P01013 GENE X PROTEIN (OVALBUMIN-RELATED)
QIKDLLVSSSTDLDTTLVLVNAIYFKGMWKTAFNAEDTREMPFHVTKQESKPVQMMCMNNSFNVATLPAE
KMKILELPFASGDLSMLVLLPDEVSDLERIEKTINFEKLTEWTNPNTMEKRRVKVYLPQMKIEEKYNLTS
VLMALGMTDLFIPSANLTGISSAESLKISQAVHGAFMELSEDGIEMAGSTGVIEDIKHSPESEQFRADHP
FLFLIKHNPTNTIVYFGRYWSP
```

  - La que contiene la descripción de la secuencia, inicia con el símbolo de “mayor que” (>)
  - Las líneas que tienen un ajuste de texto justificado, con una longitud recomendada de 80 caracteres [1]
  - No se permiten líneas en blanco dentro del archivo
  - Las líneas en minúscula se aceptan, pero deben convertirse a mayúscula al leerse [1]

2 Componente 1: Organización de la Información

Para el componente 1, se implementarán los siguientes comandos:

- Cargar <Nombre_archivo>
- Listar_Secuencias
- Histograma <Descripción secuencia>
- Es_subsecuencia <subsecuencia>
- Enmascarar <subsecuenia>
- Guardar <Nombre_Archivo>

En conjunto, el objetivo de estos comandos es obtener información del archivo FASTA para dar una salida al usuario por pantalla dependiendo de la operación que seleccione. Su prioridad es brindar al usuario información por pantalla

2.1 Descripción de los Comandos

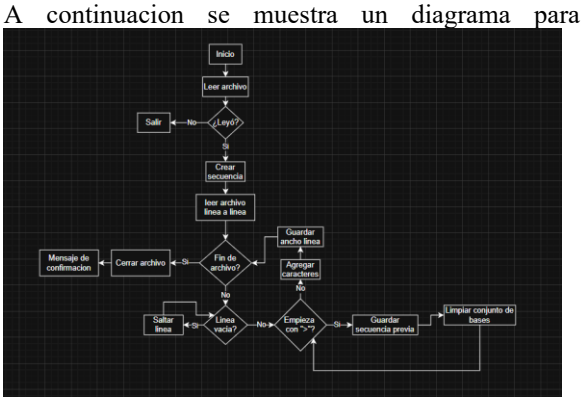
No incluyan código en esta sección; utilicen solo lenguaje natural para describir el proceso y las condiciones.

- Cargar:
    - Condiciones:
        - Debe ingresarse como parámetro el nombre del archivo con extensión .fa
        - Al almacenar información en un contenedor, este no puede contener información de 2 genomas diferentes
    - Proceso:
        - El usuario ingresa desde la consola el comando cargar acompañado del nombre de un archivo.
        - Se accede al archivo. En caso de error la salida será: (*Archivo erróneo): El archivo no se encuentra o no puede leerse.*
        - Se inicia a recorrer el archivo en un bucle, línea por línea hasta que se llegue al final del archivo
            - Por cada símbolo > se almacena el valor que lo acompaña como descripción
            - Por cada línea del archivo, se extrae cada una de las letras representado en las bases
            - Si en alguna línea se vuelve a encontrar ‘>’ se coloca un espacio vacío, indicando el inicio de una nueva secuencia, y se almacena un nuevo titulo
            - Al finalizar el recorrido, cuando se llega al final del archivo, si todavía hay una secuencia en proceso, esta también se guarda en memoria para no perderla.
            - Finalmente, el sistema informa el resultado. Si el archivo no contenía ninguna secuencia válida, se indica que no se encontraron secuencias. Si se cargó una sola secuencia, se informa que se cargó correctamente una secuencia. Y si se cargaron varias, se informa la cantidad total de secuencias cargadas desde el archivo.

Si n es 1 La salida es: *n secuencia (s) cargadas desde el archivo*

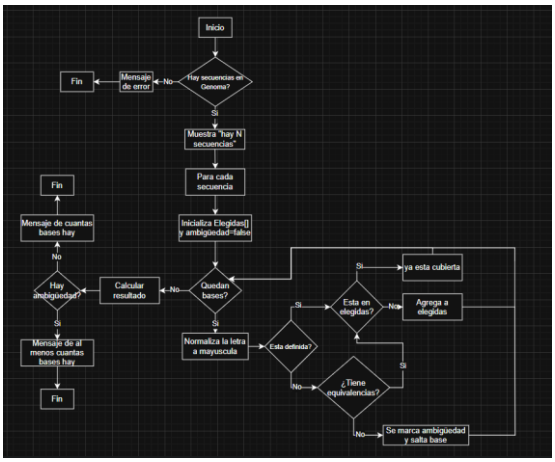
Si n>1 la salida es: *n secuencias cargadas desde el archivo*

A continuacion se muestra un diagrama para entender el flujo de este comando:



- Listar_secuencias:
  - Condiciones:
    - Debe haber al menos una secuencia cargada en el genoma; si no, se muestra: “No hay secuencias cargadas en memoria.”
    - Las letras se normalizan a mayúsculas antes de procesar.
  - Proceso:
    - Mostrar cuántas secuencias hay en memoria.
    - Para **cada secuencia**:
      - Crear un conjunto vacío “elegidas” y un indicador booleano
      - Comprobar si la letra está definida o no
        - Si la letra está definida actualiza contador para tener la base.
        - Si la letra No esta definida se buscan sus equivalentes y actualiza el indicador
      - Se comprueba si la el indicador esta en true
        - Si la bandera esta en true se comprueba que alguna de las opciones no este en el arreglo.
          - si esta no se cambia nada.
          - Si no lo esta se agrega 1 posible base
        - Si la bandera no esta en true se sigue
      - Se cuentan las secuencias con un contador
      - Se verifica el estado del indicador
        - Si es true se imprime “al menos”
        - Si es false se imprime “son”

A continuacion se muestra un diagrama para entender el flujo de este comando:

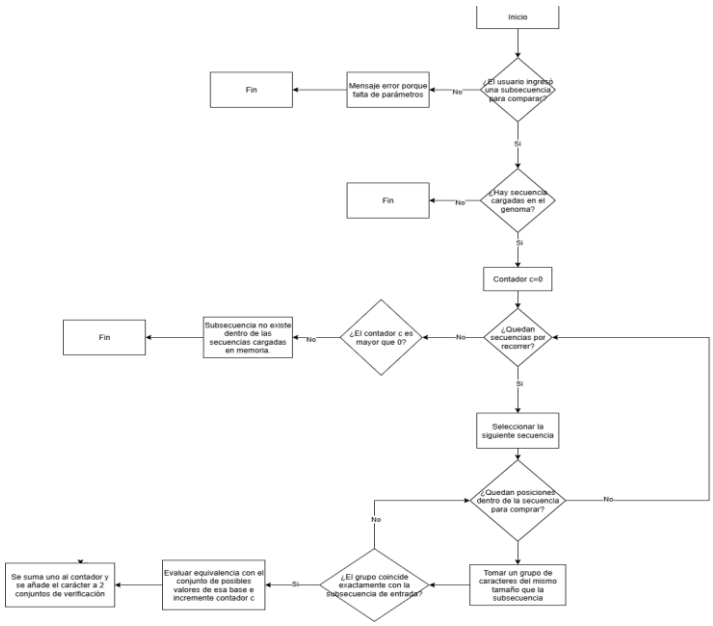


- Histograma:
  - Condiciones:
    - Se debe recibir una cadena de caracteres para comparar con la descripción
  - Proceso:
    - Se recorren todas las secuencias almacenadas en el genoma
    - Si no coincide la descripción con la cadena de entrada, la salida será: (la secuencia no existe*): Secuencia inválida*
    - Se compara la descripción de las secuencias con el valor de entrada
    - Se almacena la información de la secuencia coincidente
    - Se selecciona un valor de la tabla FASTA, y se recorren las bases de la secuencia coincidente
    - Se suma e imprime la suma por cada valor de la tabla FASTA, en el formato:
      - A: frecuencia_A
      - C: frecuencia_C

A continuacion se muestra un diagrama para entender el flujo de este comando:

- Es_subsecuencia
    - Condiciones: Debe ingresarse una subsecuencia de letras para comparar
    - Proceso:
        - Se verifica que la lista de secuencias en el genoma no esté vacía. En dicho caso la salida es: *(No hay secuencias cargadas): No hay subsecuencias cargadas en memoria*
        - Se crean 2 iteradores para recorrer la lista de secuencias y las secuencias internas
        - Se recorren las secuencias internas hasta el final de cada una, y del conjunto de secuencias completo, con un bucle
            - Se conforman grupos de caracteres del tamaño de la subsecuencia de entrada
            - Se evalúa si el grupo conformado es igual al valor de entrada, en dado caso se suma 1 a un contador c
            - Se compara si una base no definida conforma una subsecuencia igual a la entrada, recorriendo con un iterador, el conjunto de conjuntos de que incluyen el significado de las bases indefinidas.
        - Las salidas finales son:
            - Si el contador = 0: *La subsecuencia no existe dentro de las secuencias cargadas en memoria*
            - Si el contador > 0: *La subsecuencia se encuentra c veces dentro de las secuencias guardadas en memoria*

A continuacion se muestra un diagrama para entender el flujo de este comando:

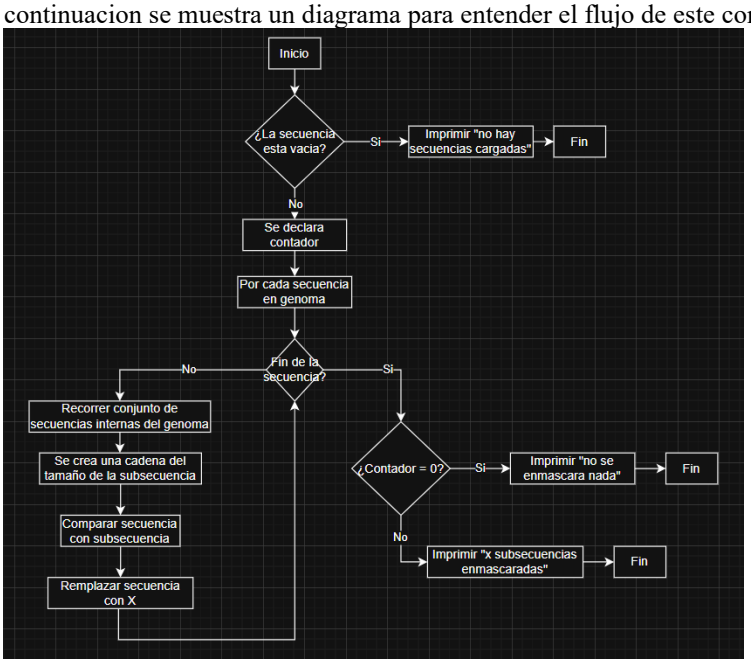


- Enmascarar
    - Condiciones: Debe ingresarse una subsecuencia de letras para comparar
    - Proceso:

- Se verifica que la lista de secuencias en el genoma no esté vacía. En dicho caso la salida es: *(No hay secuencias cargadas): No hay subsecuencias cargadas en memoria*
- Se crean dos iteradores para recorrer la lista de secuencias y cada secuencia interna.
- Se recorren todas las secuencias internas hasta el final de cada una y de la lista completa, mediante un bucle:
    - Con dos bucles anidados, se recorre cada posición y las siguientes n posiciones correspondientes al tamaño de la subsecuencia de entrada.
    - Se conforman grupos de caracteres del tamaño de la subsecuencia.
    - Se evalúa si el grupo conformado es igual a la subsecuencia de entrada; si coincide, se suma 1 a un contador **c.**
    - Inmediatamente después, se inicia un bucle no anidado que recorre esas posiciones y reemplaza los caracteres por 'X'.

- Si el contador c = 0: la subsecuencia no existe en las secuencias cargadas, por lo que **no se realiza ningún reemplazo**.
- Si el contador c > 0: la subsecuencia aparece c veces en las secuencias y **cada ocurrencia queda enmascarada con 'X' por cada letra de la subsecuencia**.

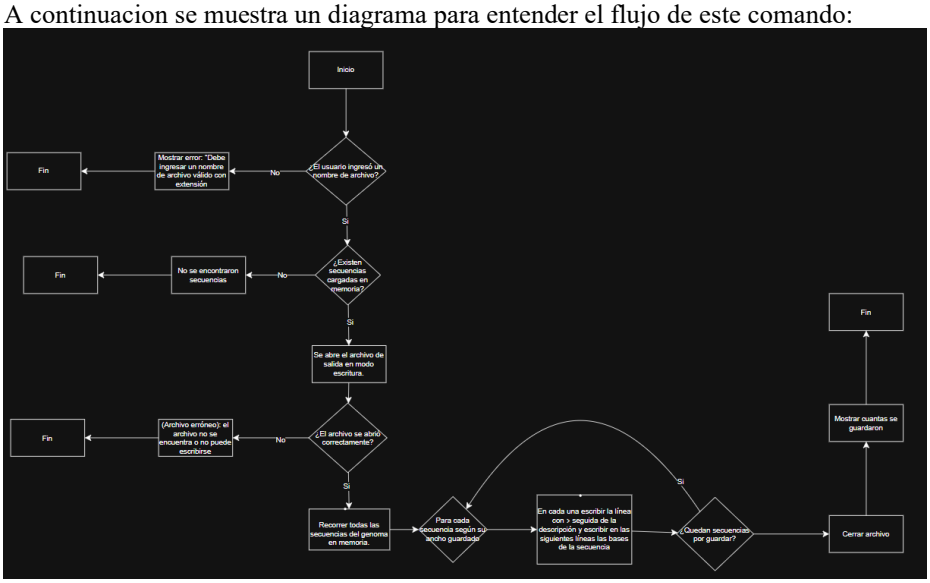
A continuacion se muestra un diagrama para entender el flujo de este comando:



- Guardar
    - Condiciones:
        - Debe ingresarse como parámetro el nombre del archivo con extensión.fa
        - No puede coexistir información de dos genomas distintos: antes de cargar, se debe vaciar el contenedor actual.

- Proceso:
  - Recepción del comando: el usuario ingresa Cargar <Nombre_archivo.fa>
  - Apertura y verificación: se intenta abrir el archivo. Si falla, la salida es: (Archivo erróneo): El archivo no se encuentra o no puede leerse.
  - se limpia el Genoma para evitar mezclar genomas (un único genoma en memoria).
  - Se recorre el archivo línea por línea hasta el fin del archivo.
  - Para cada linea se revisa si la línea está vacía, esto se considera formato inválido
  - Si la línea comienza con >, se toma el texto posterior como descripción de la nueva secuencia.
  - Si ya se venía construyendo una secuencia, se guarda la anterior y se inicia una nueva.
  - Si aparece > nuevamente más adelante, se empieza otra secuencia
  - Si la linea no empieza con > Se convierte la línea completa a mayúsculas.
  - Se extraen y agregan las bases carácter por carácter a la secuencia en curso
  - si al llegar al fin del archivo hay una secuencia en construcción, se guarda para no perderla.
  - Si no se encontró ninguna secuencia válida: “No se encontraron secuencias”
  - Si n = 1: “1 secuencia cargada desde el archivo”.
  - Si n > 1: “n secuencias cargadas desde el archivo”-

A continuacion se muestra un diagrama para entender el flujo de este comando:



2.2 Diseño

Descripción de los Tipos Abstractos de Datos (TADs):

TAD: Secuencia

Descripción/propósito: Su función es identificar y distinguir un grupo de bases nitrogenadas ordenadas, que representan una parte de un código genético

Estado:

- Description (name): Cadena de caracteres, se encarga de darle una identificación para distinguir la secuencia
- Bases_nitrogenadas (bases): Conjunto de caracteres, representa un contenedor para todas las letras que contiene una secuencia

Interfaz:

- getName(): cadena de caracteres, retorna el nombre o descripción de la cadena
  - Precondiciones: N.A
  - Postcondiciones: Una cadena que da acceso al identificador de la secuencia

- setName(cadena de caracteres): no retorna, asigna nombre o descripción de la cadena
  - Precondiciones: N.A
  - Postcondiciones: La secuencia ahora tiene asignado un identificador
- getBases(): conjunto de caracteres, retorna un conjunto de caracteres
  - Precondiciones: N.A
  - Postcondiciones: Un conjunto que muestra las bases de la secuencia

- setBases(conjunto de caracteres): no retorna, asigna un conjunto de caracteres a la secuencia
  - Precondiciones: N.A
  - Postcondiciones: La secuencia ahora tiene asignado un grupo de caracteres

- getLineWidth(): numerico, retorna el tamaño de una linea de la secuencia
  - Precondiciones: N.A
  - Postcondiciones: Un dato numerico que indica cuantas bases tiene una linea de archivo fasta

- setLineWitdth(numerico): no retorna, asigna la cantidad de bases que tiene una linea de archivo fasta
  - Precondiciones: N.A
  - Postcondiciones: La secuencia ahora tiene asignado un ancho de línea
- Equivalencias (caracter): conjunto de caracteres, retorna un conjunto de caracteres correspondiente a la equivalencia de caracteres no definidos en el codigo fasta
  - Precondiciones: Debe recibirse un caracter
  - Postcondiciones: Un conjunto de caracteres que permiten conocer las bases equivalentes al caracter de entrada

- contarSubsecuencia(cadena de caracteres): Numérico, retorna la cantidad de veces que una subsecuencia está presente en la cadena
  - Precondiciones: Debe recibirse una cadena de caracteres y deben existir subsecuencias en memoria

    - Postcondiciones: Un valor numérico que indica las repeticiones de un conjunto de caracteres específicos en la secuencia que está en memoria
- Enmascarar (cadena de caracteres): Numérico, retorna la cantidad de veces que una subsecuencia está presente en la cadena, y reemplaza en lugar de las subsecuencias con 'X'
    - Precondiciones: Debe recibirse una cadena de caracteres y deben existir subsecuencias en memoria
    - Postcondiciones: Un valor numérico que indica las repeticiones de un conjunto de caracteres específicos en la secuencia que está en memoria

- contarCaracter(caracter): Numérico, retorna la cantidad de veces que un carácter está presente en la secuencia
    - Precondiciones: NA
    - Postcondiciones: Un valor numérico que indica las repeticiones de un carácter específico en la secuencia que está en memoria

TAD: Genoma
Descripción/propósito: Su función es identificar y distinguir un grupo de secuencias, que representan y almacenan toda la información de un código FASTA
Estado:
- secuencias(sqnces): Conjunto de TAD secuencias, representa un contenedor para todas las secuencias que contenga un archivo FASTA

Interfaz:
- GetSqnces(): Conjunto de TAD secuencias, extrae las secuencias pertenecientes al genoma
    - Precondiciones: N.A
    - Postcondiciones: Se obtiene una lista de secuencias correspondientes al genoma

- SetSqnces(Conjunto de TAD secuencias): No retorna, asigna un conjunto de secuencias dentro del genoma
    - Precondiciones: N.A
    - Postcondiciones: El genoma tiene asignado un conjunto de secuencias

- Es_subsecuencia(cadena de caracteres): no retorna, muestra por pantalla si un dato de entrada pertenece a alguna secuencia almacenada en el sistema
    - Precondiciones: Debe recibirse una cadena de caracteres
    - Postcondiciones: N.A
- Enmascarar (cadena de caracteres): no retorna, reemplaza una subsecuencia por X en cualquiera de las secuencias
    - Precondiciones: Debe recibirse una cadena de caracteres
    - Postcondiciones: En las secuencias se reemplazan los valores coincidentes por cadenas conformadas de 'X'

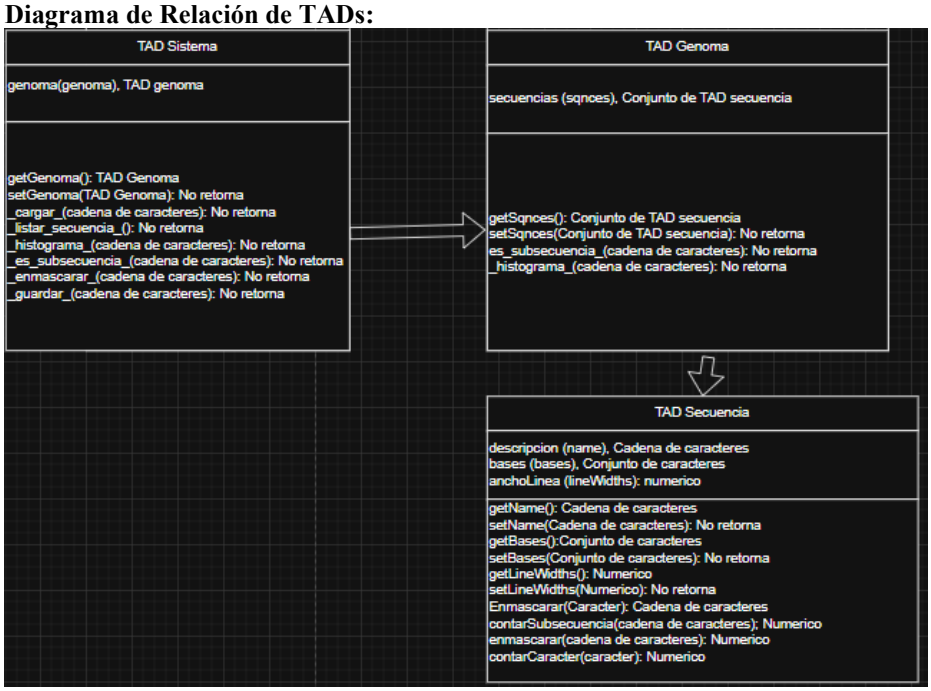
TAD: Sistema
Descripción/propósito: Su función es gestionar y administrar los archivos FASTA, así como emplear su información para mostrarla por pantalla
Estado:

Genoma (genoma): TAD Genoma, Representa el contenedor donde se almacenan la información del archivo FASTA
Interfaz:

- getGenoma (): TAD Genoma, obtiene el genoma en el que se cargan las secuencias del archivo FASTA
  - Precondiciones: N.A
  - Postcondiciones: Una copia del Genoma se almacena todas las secuencias de un archivo.FA
- setGenoma (TAD Genoma): modifica el genoma en el que se cargan las secuencias del archivo FASTA
  - Precondiciones: N.A
  - Postcondiciones: Un nuevo genoma en el sistema con la modificación de la lista de secuencias

- cargarArchivo (cadena de caracteres): no retorna, carga en el sistema el contenido del archivo fasta, separando la descripción y la secuencia de bases:
  - Precondiciones: N.A
  - Postcondiciones: En una instancia de Genoma se almacena toda la información de un archivo.FA
- listarSecuencias(): no retorna, muestra por pantalla la cantidad de listas almacenadas en memoria, indicando su cantidad de bases y si están completas o no
  - Precondiciones: N.A
  - Postcondiciones: N.A
- Histograma (cadena de caracteres): muestra por pantalla, usando una forma de histograma horizontal, la cantidad de elementos de la tabla de código fasta que se encuentran en determinada secuencia
  - Precondiciones: N.A
  - Postcondiciones: N.A
- Es_subsecuencia (cadena de caracteres): no retorna, muestra por pantalla si un dato de entrada pertenece a alguna secuencia almacenada en el sistema
  - Precondiciones: Debe recibirse una cadena de caracteres
  - Postcondiciones: N.A
- Enmascarar (cadena de caracteres): no retorna, reemplaza una subsecuencia por X en cualquiera de las secuencias
  - Precondiciones: Debe recibirse una cadena de caracteres
  - Postcondiciones: En las secuencias se reemplazan los valores coincidentes por cadenas conformadas de ‘X’
- Guardar (cadena de caracteres): no retorna, escribe dentro de un archivo FASTA el contenido del genoma con sus secuencias modificadas
  - Precondiciones: Debe respetarse el formato FASTA y la justificación de renglones
  - Postcondiciones: Se genera un archivo de extensión .fa, con el contenido de las secuencias modificado por el sistema

Diagrama de Relación de TADs:



El TAD sistema se encarga de los comandos que traer información con el usuario, tales como cargar, guardar e impresiones informativas como histograma y listar secuencias. El sistema contiene un único genoma. El TAD Genoma es el encargado de recibir almacenar la información de los archivos fasta, incluyendo en él una lista de secuencias (de lo que está compuesto dicha clase de archiva). Por último, el TAD secuencias, contiene una cadena de caracteres correspondiente a la descripción de una secuencia, y un conjunto de caracteres donde se almacenan todas las bases asociadas a la descrpción

2.3 Plan de Pruebas

Casos de prueba:

- No hay subsecuencias cargadas
- La subsecuencia de entrada no existe
- La subsecuencia se repite una sola vez
- La subsecuencia es igual a la secuencia completa
- La subsecuencia corresponde a un único carácter
- Hay varias subsecuencias en todas las secuencias
- Hay 2 subsecuencias sobrelapadas, en varias partes de la cadena

Criterios de prueba:

- El mensaje de salida es el esperado según el caso
- La cantidad de subsecuencias encontradas es correcta
- Las subsecuencias son reemplazadas por x
- Las secuencias mantienen su integridad si no corresponden a la subsecuencia

Metodología de pruebas (a excepción del caso vacío):

1. Se introduce un archivo de extensión fa que contiene 3 secuencias dentro de un genoma (comprobación de cargar)
2. Se llama a la función histograma para cada secuencia cargada
3. Se solicita la subsecuencia a enmascarar por pantalla
4. Se detalla el número de subsecuencias enmascaradas según el mensaje de salida
5. Se llama nuevamente al comando histograma para ver los cambios en las cantidades de caracteres
6. Se guardan las secuencias enmascaradas en un archivo.fa

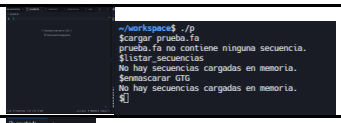
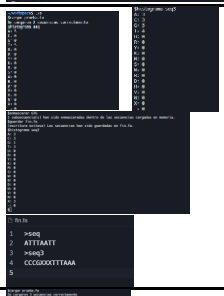
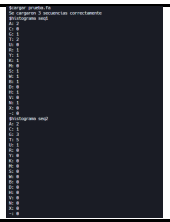
Ejecución de Pruebas

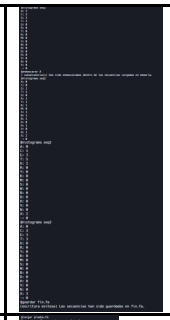
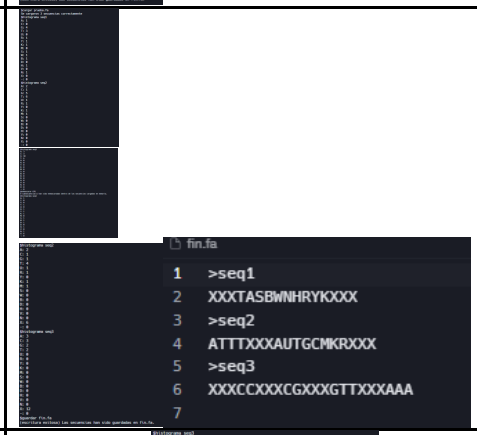
Documentación:

Plan de pruebas:			
Descripción de caso	Valores de entrada	Resultado esperado	Resultado obtenido
No se cargaron subsecuencias en memoria	Secuencias en memoria: N.A Subsecuencia: GTG	"No hay subsecuencias cargadas en memoria"	"No hay subsecuencias cargadas en memoria"
La subsecuencia de entrada no existe	Secuencias en memoria: >seq1 ATTTGT AATT >seq2 CCCTGTTAAA Subsecuencia: GTG	La subsecuencia dada no existe dentro de las secuencias cargadas en memoria	La subsecuencia dada no existe dentro de las secuencias cargadas en memoria
La subsecuencia se repite exactamente 1 vez	Secuencias en memoria: >seq ATTTAATT >seq3 CCCGGTGTTTAAA Subsecuencia: GTG	1 subsecuencias han sido enmascaradas dentro de las secuencias cargadas en memoria.	1 subsecuencias han sido enmascaradas dentro de las secuencias cargadas en memoria.
Una subsecuencia igual a una secuencia	Secuencias en memoria: >seq1 ATGTASBWNHRYK >seq2 ATTTGTGAUTGC >seq3 CCCGGGTTTAAA Subsecuencia:	1 subsecuencias han sido enmascaradas dentro de las secuencias cargadas en memoria.	1 subsecuencias han sido enmascaradas dentro de las secuencias cargadas en memoria.

	ATTT**GTG**AUTGC		
La subsecuencia es una base	Secuencias en memoria: >seq1 ATGTASBWNHRYK >seq2 ATTTGTGAUTGC >seq3 CCCGGGTTTAAA Subsecuencia: A	7 subsecuencias han sido enmascaradas dentro de las secuencias cargadas en memoria.	7 subsecuencias han sido enmascaradas dentro de las secuencias cargadas en memoria.
La subsecuencia se encuentra varias veces separada	Secuencias en memoria: >seq1 GTGTASBWNHRYK GTG >seq2 ATTTGTGAUTGCMK RGTG >seq3 GTGCCGTGCGGTGG TTGTGAAA Subsecuencia: GTG	8 subsecuencia(s) han sido enmascaradas dentro de las secuencias cargadas en memoria.	8 subsecuencia(s) han sido enmascaradas dentro de las secuencias cargadas en memoria.
La subsecuencia se encuentra varias veces sobrelapada	Secuencia en memoria >seq1 ATGTASBWNHRYK GTGTGTTAA >seq2 ATTTGTGAUTGCGT GCGTAGTGTGTGTG TGAATT >seq3 CCCGGGTGTTAAA Subsecuencia: GTG	7 subsecuencia(s) han sido enmascaradas dentro de las secuencias cargadas en memoria.	7 subsecuencia(s) han sido enmascaradas dentro de las secuencias cargadas en memoria.

Caso	Evidencia

No se cargaron subsecuencias en memoria	
La subsecuencia de entrada no existe	
La subsecuencia se repite exactamente 1 vez	
Una subsecuencia igual a una secuencia	
La subsecuencia es una base	

	
La subsecuencia se encuentra varias veces separada	
La subsecuencia se encuentra varias veces sobrelapada	

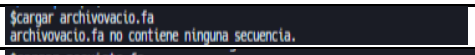
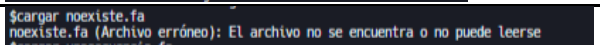
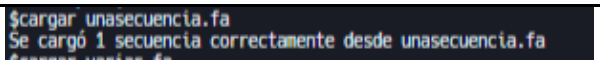
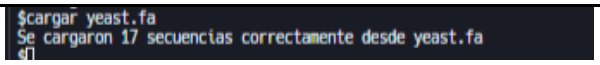
2.4 Correcciones componente 1

No existieron errores de funcionalidad, sin embargo, el plan de pruebas del componente no fue claro dentro del documento, por lo cual a continuación se presenta nuevamente, explicando el plan de pruebas de cada uno de los comandos en el componente

Plan de pruebas:

- **Cargar:**

Plan de pruebas:			
Descripción de caso	Valores de entrada	Resultado esperado	Resultado obtenido
Se cargó un archivo sin secuencias	Archivo vacío: ArchivoVacio.fa	archivovacio.fa no contiene ninguna secuencia.	archivovacio.fa no contiene ninguna secuencia.
Se cargó un archivo inexistente	Archivo que no existe: Noexiste.fa	noexiste.fa (Archivo erróneo): El archivo no se encuentra o no puede leerse	noexiste.fa (Archivo erróneo): El archivo no se encuentra o no puede leerse
Se carga un archivo con una secuencia	Archivo con una secuencia unasecuencia.fa	Se cargó 1 secuencia correctamente desde unasecuencia.fa	Se cargó 1 secuencia correctamente desde unasecuencia.fa
Se carga un archivo con 15 subsecuencias	Archivo con 15 subsecuencias: varias.fa	Se cargaron 15 secuencias correctamente desde varias.fa	Se cargaron 15 secuencias correctamente desde varias.fa
Se carga un archivo con 17 subsecuencias	Archivo con 17 subsecuencias: yeast.fa	Se cargaron 17 secuencias correctamente desde yeast.fa	Se cargaron 17 secuencias correctamente desde yeast.fa

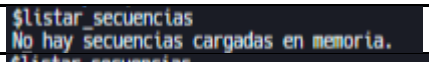
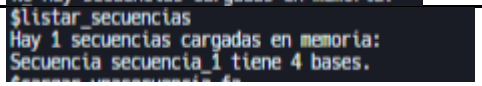
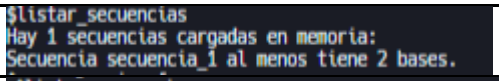
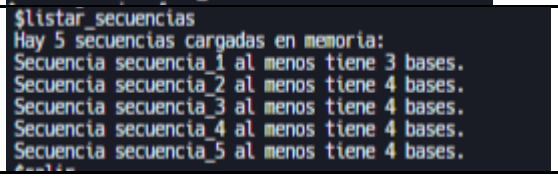
Caso	Evidencia
Se cargó un archivo sin secuencias	
Se cargó un archivo inexistente	
Se cargó un archivo con una secuencia	
Se cargó un archivo con 15 subsecuencias	
Se cargó un archivo con 17 subsecuencias	

- ***Listar Secuencias***

Plan de pruebas:			
Descripción de caso	Valores de entrada	Resultado esperado	Resultado obtenido

No se cargó ningun archivo	-	No hay subsecuencias guardadas en memoria	No hay subsecuencias guardadas en memoria
Se cargó una secuencia en memoria, cuyo contenido eran solo bases definidas	Unasecuencia.fa: >secuencia_1 ATGCTAGCTAGCTACGATC GATCGATCGATCGATCGAT CGATCGATCGATCGATCGATC GATCGATCGATCGATCGAT CGATCGATCGATCGATCGA TCGATCGATCGATCGATCGAT	Hay 1 secuencias cargadas en memoria: Secuencia secuencia_1 tiene 4 bases.	Hay 1 secuencias cargadas en memoria: Secuencia secuencia_1 tiene 4 bases.
Se carga una secuencia con bases ambiguas	Unasecuencia.fa: >secuencia_1 ATRMBYRKM	Hay 1 secuencias cargadas en memoria: Secuencia secuencia_1 al menos tiene 2 bases.	Hay 1 secuencias cargadas en memoria: Secuencia secuencia_1 al menos tiene 2 bases.
Se agregan 5 secuencias con bases ambiguas	>secuencia_1 ATRMBYRKMWSKDHVBN >secuencia_2 TGCARYWSNMBDHVKBN >secuencia_3 CGTAMRWSKNYBDHVBN >secuencia_4 ATGCYRWSKMBDHVNBN secuencia_5 TACGRMWSKNYBDHVBN	Hay 5 secuencias cargadas en memoria: Secuencia secuencia_1 al menos tiene 3 bases. Secuencia secuencia_2 al menos tiene 4 bases. Secuencia secuencia_3 al menos tiene 4 bases. Secuencia secuencia_4 al	Hay 5 secuencias cargadas en memoria: Secuencia secuencia_1 al menos tiene 3 bases. Secuencia secuencia_2 al menos tiene 4 bases.

		menos tiene 4 bases. Secuencia secuencia_5 al menos tiene 4 bases.	Secuencia secuencia_3 al menos tiene 4 bases. Secuencia secuencia_4 al menos tiene 4 bases. Secuencia secuencia_5 al menos tiene 4 bases.

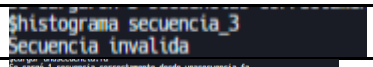
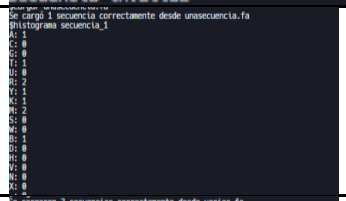
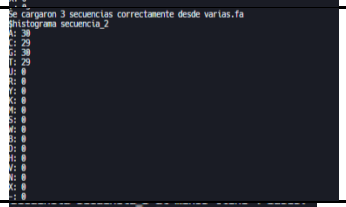
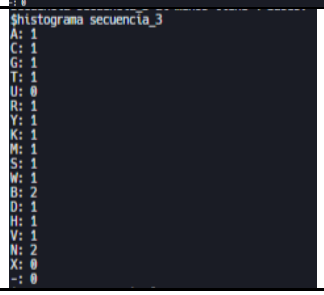
Caso	Evidencia
No se cargó ningun archivo	
Se cargó una secuencia en memoria, cuyo contenido eran solo bases definidas	
Se carga una secuencia con bases ambiguas	
Se agregan 5 secuencias con bases ambiguas	

- **HISTOGRAMA**

Plan de pruebas:			
Descripción de caso	Valores de entrada	Resultado esperado	Resultado obtenido
No existe la secuencia	En memoria: Unasecuencia.fa:	Secuencia invalida	Secuencia invalida

	>secuencia_1 ATRMBYRKM Por parametro: Secuencia_2		
Existe una secuencia en memoria, con varias bases ambiguas	En memoria: Unasecuencia.fa: >secuencia_1 ATRMBYRKM Por parametro: Secuencia_1	secuencia_1 A: 1 C: 0 G: 0 T: 1 U: 0 R: 2 Y: 1 K: 1 M: 2 S: 0 W: 0 B: 1 D: 0 H: 0 V: 0 N: 0 X: 0 -: 0	secuencia_1 A: 1 C: 0 G: 0 T: 1 U: 0 R: 2 Y: 1 K: 1 M: 2 S: 0 W: 0 B: 1 D: 0 H: 0 V: 0 N: 0 X: 0 -: 0
Se cargan varias secuencias, se hace el histograma de una secuencia con bases definidas	Variassecuencias.fa: >secuencia_1 ATGCTAGCTAGCTACGATCGAT CG ATCGATCGATCGATCGATCGAT CG ATCGATCGATC GATCGATCGATCGATCGATCGA TC GATCGATCGATCGATCGATCGA TC GATCGATCGAT >secuencia_2 GATCGATCGATCG ATCGATCGATCGATC GATCGATCGATCGATC GATCGATCGATCGAT CGATCGATCGATCGA TCGATCGATCGATC GATCGATCGATCGATCG ATCGATCGATCGA Parámetro: Secuencia 2	$histograma secuencia_2 A: 30 C: 29 G: 30 T: 29 U: 0 R: 0 Y: 0 K: 0 M: 0 S: 0 W: 0 B: 0 D: 0 H: 0 V: 0 N: 0 X: 0 -: 0	$histograma secuencia_2 A: 30 C: 29 G: 30 T: 29 U: 0 R: 0 Y: 0 K: 0 M: 0 S: 0 W: 0 B: 0 D: 0 H: 0 V: 0 N: 0 X: 0 -: 0
Se agregan 2 secuencias con con bases definidas y una con ambiguas, se pide el histograma de las ambiguas	>secuencia_1 ATGCTAGCTAGCTACGATCGAT CGATCGATCGATCGATCGATCG ATCGATCGATCGATC GATCGATCGATCGATCGATCGA TCGATCGATCGATCGATCGATC GATCGATCGATCGAT >secuencia_2 GATCGATCGATCGATCGATCGA TCGATCGATCGATCGATCGATC GATCGATCGATCGAT CGATCGATCGATCGATCGATCG ATCGATCGATCGATCGATCGAT	$histograma secuencia_3 A: 1 C: 1 G: 1 T: 1 U: 0 R: 1 Y: 1 K: 1 M: 1 S: 1 W: 1 B: 2 D: 1 H: 1 V: 1 N: 2 X: 0 -: 0	$histograma secuencia_3 A: 1 C: 1 G: 1 T: 1 U: 0 R: 1 Y: 1 K: 1 M: 1 S: 1 W: 1 B: 2 D: 1 H: 1 V: 1 N: 2 X: 0 -: 0

	CGATCGATCGATCGA >secuencia_3 CGTAMRWSKNYBDHVBN Parametro: secuencia 3		

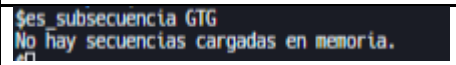
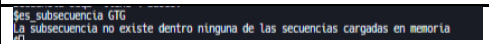
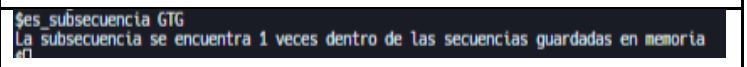
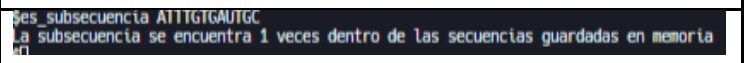
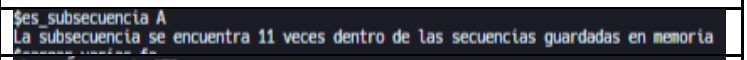
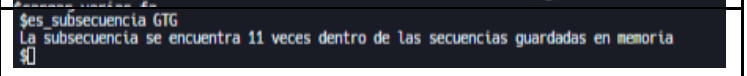
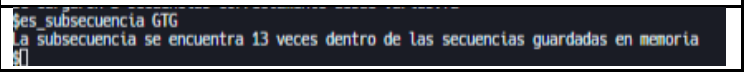
Caso	Evidencia
No existe la secuencia	
Existe una secuencia en memoria, con varias bases ambiguas	
Se cargan varias secuencias, se hace el histograma de una secuencia con bases definidas	
Se agregan 2 secuencias con bases definidas y una con ambiguas, se pide el histograma de las ambiguas	

- **Es subsecuencia**

Plan de pruebas:			
Descripción de caso	Valores de entrada	Resultado esperado	Resultado obtenido
No se cargaron subsecuencias en memoria	Secuencias en memoria:N.A Subsecuencia: GTG	"No hay subsecuencias cargadas en memoria"	"No hay subsecuencias cargadas en memoria"
La subsecuencia de	Secuencias en memoria: >seq1	La subsecuencia	La subsecuencia dada no

entrada no existe	ATTTGT AATT >seq2 CCCTGTTAAA Subsecuencia: GTG	dada no existe dentro de las secuencias cargadas en memoria	existe dentro de las secuencias cargadas en memoria
La subsecuencia se repite exactamente 1 vez	Secuencias en memoria: >seq ATTTAATT >seq3 CCCGGTGTTTAAA Subsecuencia: GTG	La subsecuencia se encuentra 1 veces dentro de las secuencias guardadas en memoria	La subsecuencia se encuentra 1 veces dentro de las secuencias guardadas en memoria
Una subsecuencia igual a una secuencia	Secuencias en memoria: >seq1 ATGTASBWNHRYK >seq2 ATTTGTGAUTGC >seq3 CCCGGGTTTAAA Subsecuencia: ATTT**GTG**AUTGC	La subsecuencia se encuentra 1 veces dentro de las secuencias guardadas en memoria	La subsecuencia se encuentra 1 veces dentro de las secuencias guardadas en memoria
La subsecuencia es una base, contando sus equivalencias en bases ambiguas	Secuencias en memoria: >seq1 ATGTASBWNHRYK >seq2 ATTTGTGAUTGC >seq3 CCCGGGTTTAAA Subsecuencia: A: Teniendo en cuenta que W, N y H cuentan como A	La subsecuencia se encuentra 11 veces en las secuencias guardadas en memoria	La subsecuencia se encuentra 11 veces en las secuencias guardadas en memoria
La subsecuencia se encuentra varias veces separada (con bases ambiguas)	Secuencias en memoria: >seq1 GTGTASBWNHRYKGTG >seq2 ATTTGTGAUTGCMKRGTG >seq3 GTGCCGTGCGGTGGTTGTGAAA Subsecuencia: GTG	La subsecuencia se encuentra 11 veces en las secuencias guardadas en memoria	La subsecuencia se encuentra 11 veces en las secuencias guardadas en memoria

La subsecuencia se encuentra varias veces sobrelapada	Secuencia en memoria >seq1 ATGTASBWNHRYKGTGTGTTAA >seq2 ATTTGTGAUTGCGTGCGTAGTGTGTGTGTGAATT >seq3 CCCGGGTGTTAAA Subsecuencia: GTG	La subsecuencia se encuentra 13 veces en las secuencias guardadas en memoria	La subsecuencia se encuentra 13 veces en las secuencias guardadas en memoria

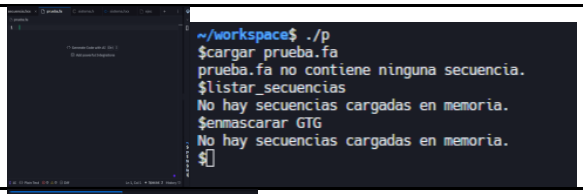
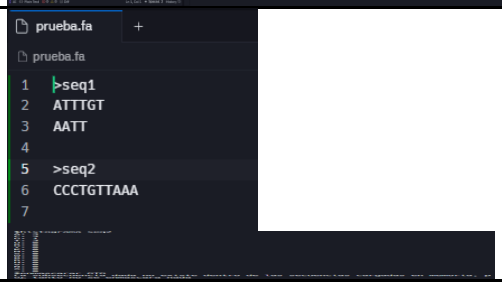
Caso	Evidencia
No se cargaron subsecuencias en memoria	
La subsecuencia de entrada no existe	
La subsecuencia se repite exactamente 1 vez	
Una subsecuencia igual a una secuencia	
La subsecuencia es una base	
La subsecuencia se encuentra varias veces separada (con bases ambiguas)	
La subsecuencia se encuentra varias veces sobrelapada	

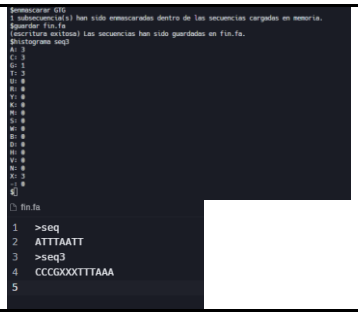
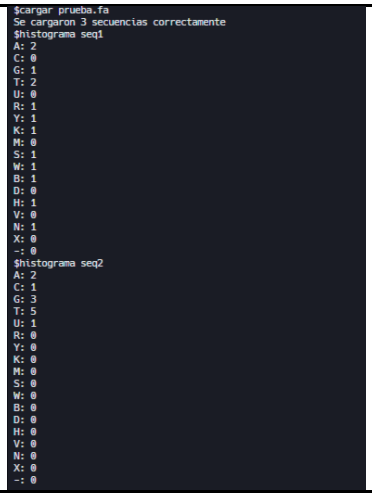
- **Enmascarar y Guardar archivo**

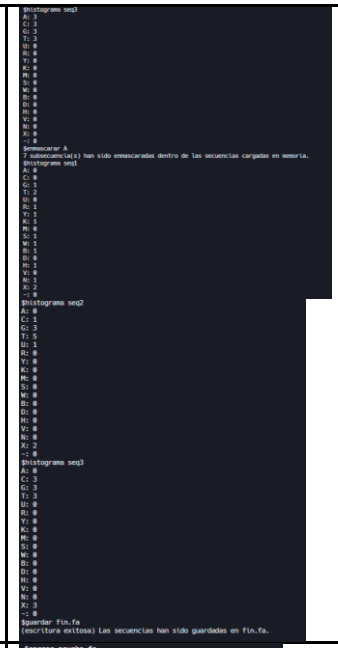
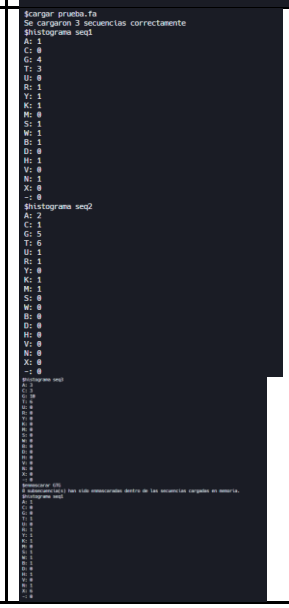
Plan de pruebas:			
Descripción de caso	Valores de entrada	Resultado esperado	Resultado obtenido

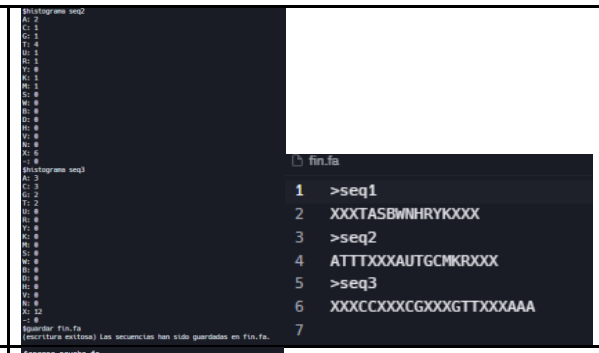
No se cargaron subsecuencias en memoria	Secuencias en memoria: N.A Subsecuencia: GTG	"No hay subsecuencias cargadas en memoria"	"No hay subsecuencias cargadas en memoria"
La subsecuencia de entrada no existe	Secuencias en memoria: >seq1 ATTTGT AATT >seq2 CCCTGTTAAA Subsecuencia: GTG	La subsecuencia dada no existe dentro de las secuencias cargadas en memoria	La subsecuencia dada no existe dentro de las secuencias cargadas en memoria
La subsecuencia se repite exactamente 1 vez	Secuencias en memoria: >seq ATTTAATT >seq3 CCCGGTGTTTAAA Subsecuencia: GTG	1 subsecuencias han sido enmascaradas dentro de las secuencias cargadas en memoria.	1 subsecuencias han sido enmascaradas dentro de las secuencias cargadas en memoria.
Una subsecuencia igual a una secuencia	Secuencias en memoria: >seq1 ATGTASBWNHRYK >seq2 ATTTGTGAUTGC >seq3 CCCGGGTTTAAA Subsecuencia: ATTT**GTG**AUTGC	1 subsecuencias han sido enmascaradas dentro de las secuencias cargadas en memoria.	1 subsecuencias han sido enmascaradas dentro de las secuencias cargadas en memoria.
La subsecuencia es una base	Secuencias en memoria: >seq1 ATGTASBWNHRYK >seq2 ATTTGTGAUTGC >seq3 CCCGGGTTTAAA Subsecuencia: A	7 subsecuencias han sido enmascaradas dentro de las secuencias cargadas en memoria.	7 subsecuencias han sido enmascaradas dentro de las secuencias cargadas en memoria.

La subsecuencia se encuentra varias veces separada	Secuencias en memoria: >seq1 GTGTASBWNHRYKGTG >seq2 ATTTGTGAUTGCMKRGTG >seq3 GTGCCGTGCGGTGGTTGTGAAA Subsecuencia: GTG	8 subsecuencia(s) han sido enmascaradas dentro de las secuencias cargadas en memoria.	8 subsecuencia(s) han sido enmascaradas dentro de las secuencias cargadas en memoria.
La subsecuencia se encuentra varias veces sobrelapada	Secuencia en memoria >seq1 ATGTASBWNHRYKGTGTGTTAA >seq2 ATTTGTGAUTGCGTGCGTAGTGTGTGTGTGAATT >seq3 CCCGGGTGTTAAA Subsecuencia: GTG	7 subsecuencia(s) han sido enmascaradas dentro de las secuencias cargadas en memoria.	7 subsecuencia(s) han sido enmascaradas dentro de las secuencias cargadas en memoria.

Caso	Evidencia
No se cargaron subsecuencias en memoria	
La subsecuencia de entrada no existe	
La subsecuencia se repite exactamente 1 vez	

	
Una subsecuencia igual a una secuencia	
La subsecuencia es una base	

	
La subsecuencia se encuentra varias veces separada	

	
La subsecuencia se encuentra varias veces sobre-lapada	

3 Componente 2: Compresión y descompresión de archivos FASTA

Para el componente 1, se implementarán los siguientes comandos:

- Codificar
- Decodificar

  En conjunto, el objetivo de estos comandos es guardar y acceder de forma más eficiente toda la información de un genoma, incluyendo todas sus secuencias y sus respectivas bases constitutivas. De forma tal que se puede consultar y almacenar la información en archivos binarios, con cada base (definida o ambigua) siendo codificada y decodificada usando un árbol de Huffman

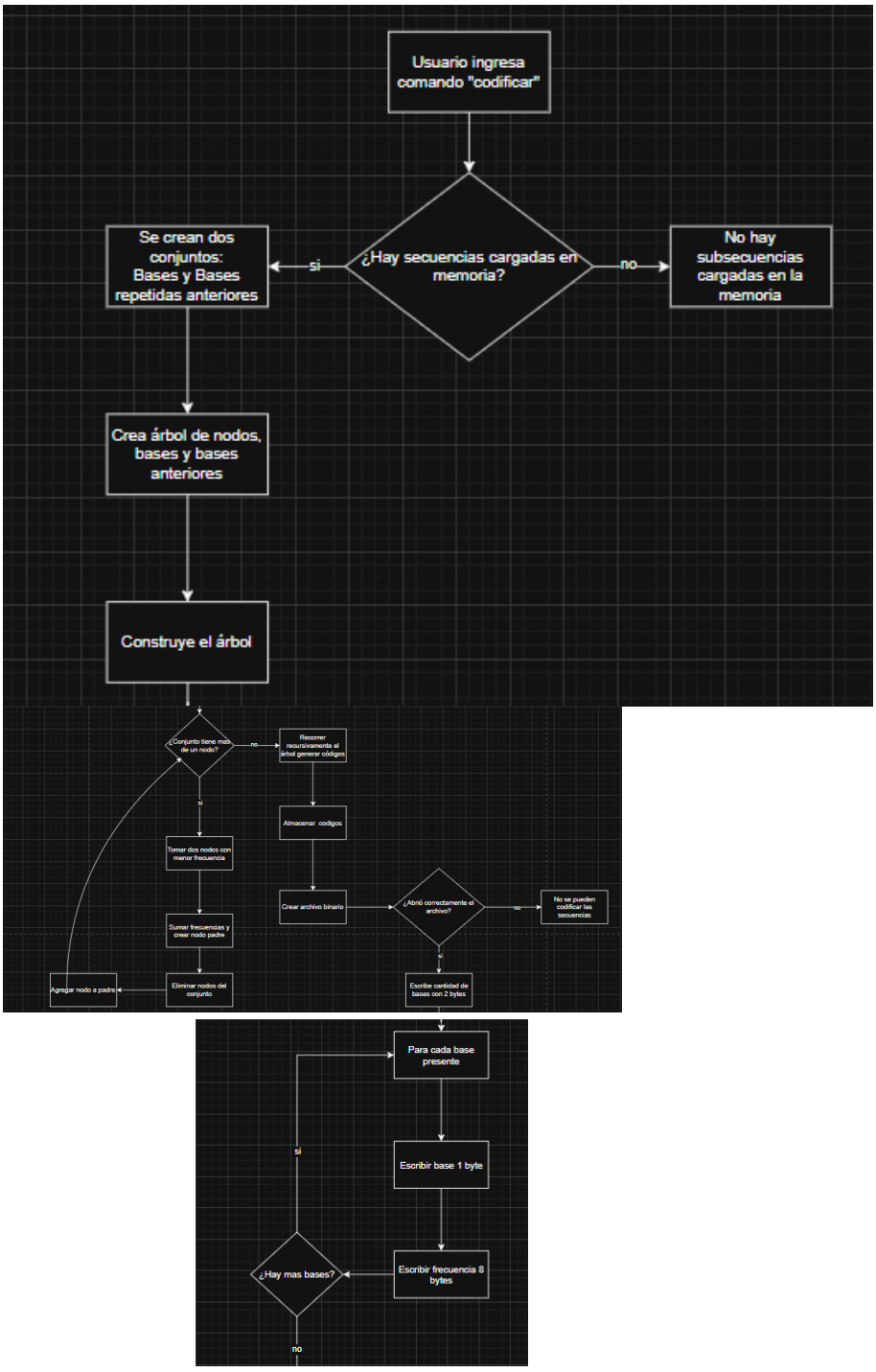
3.1 Descripción de los Comandos

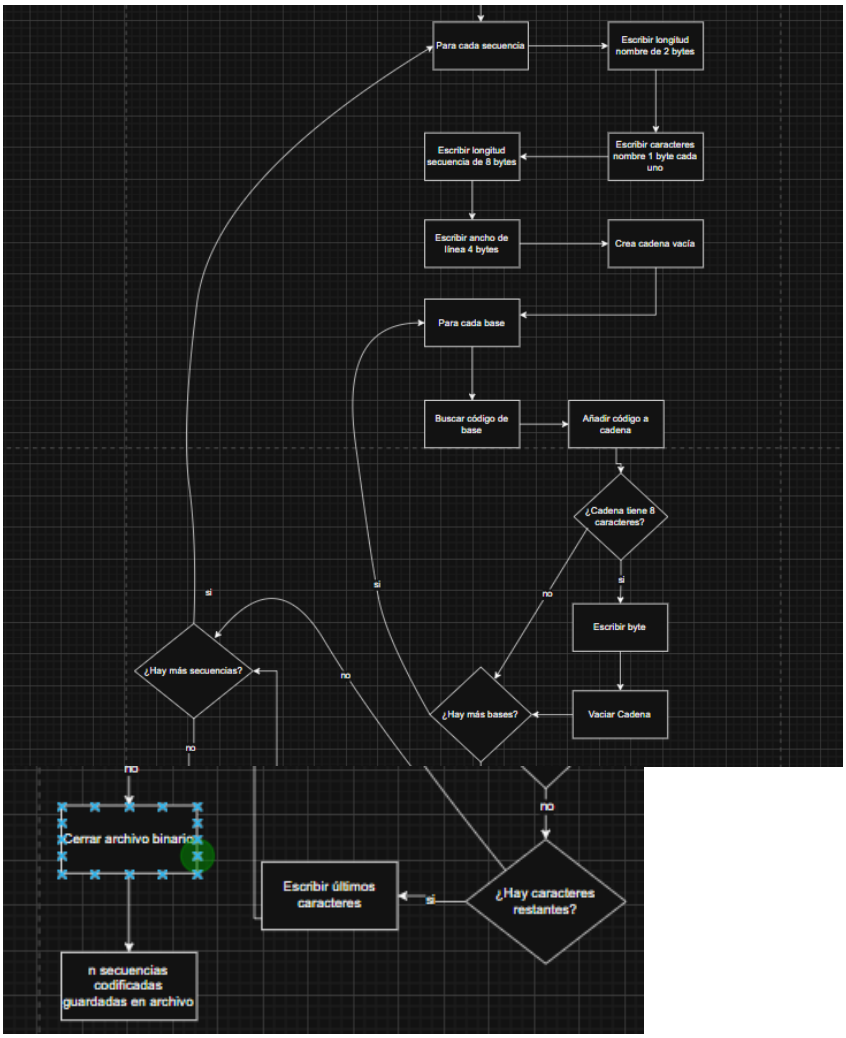
- Codificar:
  - Condiciones:
    - Debe ingresarse como parámetro el nombre del archivo donde se almacenará la información, un archivo en formato fabin: Archivo.fabin
  - Proceso:
    - El usuario ingresa desde la consola el comando “codificar” acompañado del nombre de un archivo de formato fabin
    - Se verifica que la memoria tenga secuencias cargadas, si no es el caso, la salida es *“No hay subsecuencias cargadas en memoria”*
    - El sistema ingresa a codificar el genoma
      - Se crean 2 conjuntos en paralelo, un conjunto que contiene las bases dentro de las secuencias, y otro que contiene la cantidad de veces que se repiten las bases del conjunto anterior. Se construyen de tal forma sus posiciones están sincronizadas
      - A partir de los conjuntos, se genera un árbol de Huffman
        - Se crea un conjunto de nodos, en el que se ingresan las bases con sus frecuencias
        - Se toman las menores frecuencias, se suman y se eliminan del conjunto, haciendo un nuevo nodo con la suma, que se vuelve padre de los nodos borrados. Este proceso se repite hasta que el conjunto de nodos solo contenga 1 elemento
      - Con el árbol de Huffman ya construido, se recorre recursivamente el árbol para generar los códigos de cada base, y se almacenan en un conjunto de cadenas de caracteres

- El sistema, una vez el genoma tiene los conjuntos de bases presentes, frecuencias y códigos, inicia con la escritura del archivo.fabin
- Se crea o accede al archivo en modo binario. En caso de error la salida será: *No se pueden codificar las secuencias cargadas en nombre_archivo.fabin*
- Se escribe la cantidad de bases presentes usando 2 bytes
- Se realiza un ciclo para escribir todas las bases presentes con un byte y sus respectivas frecuencias con 8 bytes
- Por cada secuencia:
    - Se escribe la cantidad de caracteres del nombre usando 2 bytes y cada letra del nombre usando un byte
    - Se escribe la longitud de la secuencia usando 8 bytes
    - Se escribe el ancho de línea de la secuencia usando 4 bytes
    - Crea una cadena de caracteres vacía
    - Busca cada base en el conjunto de bases presentes, obteniendo su código, que se añade a la cadena de caracteres vacía
    - Hace grupos de 8, y los escribe en archivo binario, para luego vaciar la cadena de caracteres y repetir el proceso hasta finalizar toda la secuencia
- Cierra el archivo binario

La salida es: *n secuencia codificada y guardada en el archivo nombre_archivo.fabin*

A continuacion se muestra un diagrama para entender el flujo de este comando:


Usuario ingresa comando "codificar"
¿Hay secuencias cargadas en memoria?
si
no
Se crean dos conjuntos: Bases y Bases repetidas anteriores
No hay subsecuencias cargadas en la memoria
Crea árbol de nodos, bases y bases anteriores
Construye el árbol
¿Conjunto tiene mas de un nodo?
no
si
Recorrer recursivamente el árbol generar códigos
Almacenar codigos
Tomar dos nodos con menor frecuencia
Sumar frecuencias y crear nodo padre
Eliminar nodos del conjunto
Agregar nodo a padre
Crear archivo binario
¿Abrió correctamente el archivo?
no
si
No se pueden codificar las secuencias
Escribe cantidad de basis con 2 bytes
Para cada base presente
si
Escribir base 1 byte
Escribir frecuencia 8 bytes
¿Hay mas bases?
no

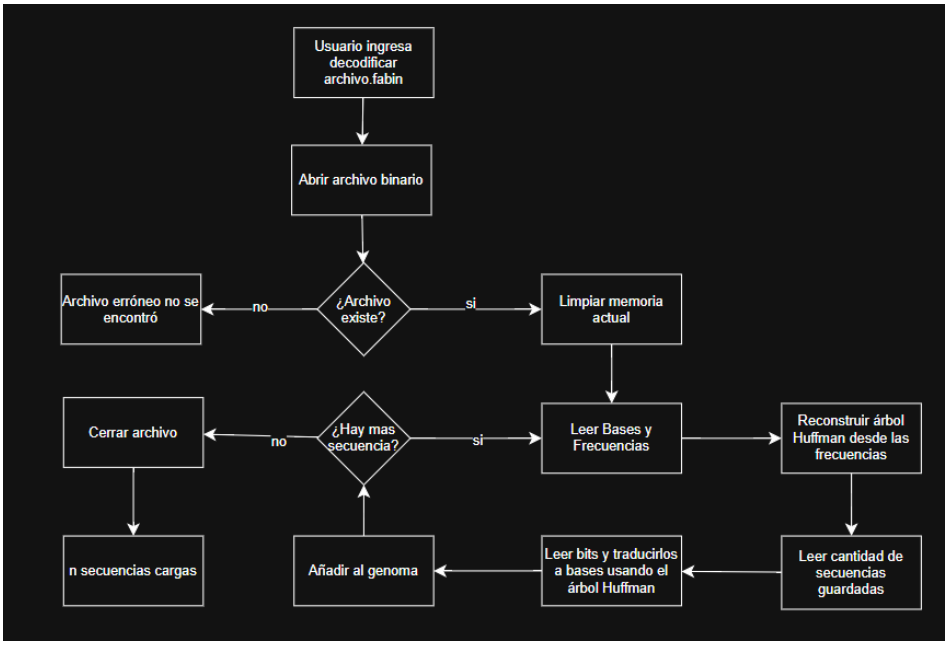


- Decodificar
  - Condiciones:
    - Que el archivo exista y se pueda leer.
    - Que las bases de la secuencia sean válidas
  - Proceso:
    - Vacía todo lo anterior que había en memoria
    - Abre el archivo binario.
    - Lee la cantidad de bases en la secuencias
    - Lee cada base y su frecuencia, almacenando las letras válidas y cuántas veces aparecieron en contenedores para cada uno
    - Reconstruye el árbol de Huffman con esa información.
    - Lee cuántas secuencias hay guardadas dentro del archivo.
    - Para cada secuencia:

- Lee su nombre.
- Lee cuántas bases tiene y el ancho de línea (para saber cada cuántos caracteres se hace un salto).
- Usa el árbol para traducir los ceros y unos en letras.
    - Si el árbol tiene varias, va leyendo los bits y recorriendo el árbol hasta encontrar cada letra.
- Guarda la secuencia reconstruida en memoria.

▪ Cierra el archivo y muestra un mensaje confirmando que las secuencias fueron decodificadas correctamente.

A continuación, se muestra un diagrama para entender el flujo de este comando:



3.2 Diseño

Descripción de los Tipos Abstractos de Datos (TADs):

Se añadieron 2 nuevos TAD: NodoHuffman y ArbolHuffman. Adicionalmente, se modificaron los TAD sistema y TAD genoma. En el sistema se agregaron 2 nuevas funciones: Una por código. Mientras en el genoma se añadieron 3 nuevos vectores para almacenar las bases presentes, las frecuencias respectivas y los códigos generados tras recorrer el árbol. El TAD secuencia NO fue modificado, por lo que no se incluye en la descripción de los TAD de este componente. A continuación se muestran los nuevos TAD, junto a los TAD modificados

NUEVOS TAD:

TAD: NodoHuffman
Descripción/propósito: Su función es constituir el árbol de Huffman, almacenando las bases del genoma junto a sus frecuencias
Estado:

- Símbolo (dato): Carácter, hace referencia a una base dentro del genoma. Default= ‘ ’
- Frecuencia (frecuencia): numérico, representa la cantidad de veces que se repite cierta base, o la suma de frecuencias de 2 bases
- Hijo Izquierdo(hijoIzq): Referencia a TAD NodoHuffman, representa el descendiente izquierdo del nodo
- Hijo Derecho(hijoDer): Referencia a TAD NodoHuffman, representa el descendiente derecho del nodo

Interfaz:

- NodoHuffman (): TAD NodoHuffman, Crea un NodoHuffman vacío, sin hijos
  - Precondiciones: N.A
  - Postcondiciones: N.A
- NodoHuffman (Carácter, numérico): TAD NodoHuffman, Crea un NodoHuffman con simbolo y frecuencia, sin hijos
  - Precondiciones: N.A
  - Postcondiciones: N.A
- NodoHuffman (Carácter, numérico, TAD NodoHuffman, TAD NodoHuffman): TAD NodoHuffman, Crea un NodoHuffman con símbolo, frecuencia e hijos
  - Precondiciones: N.A
  - Postcondiciones: N.A
- ObtenerDato (): Caracter, retorna el símbolo o base dentro del nodo
  - Precondiciones: N.A
  - Postcondiciones: N.A
- FijarDato (Carácter): No retorna, Asigna un símbolo o base al nodo
  - Precondiciones: N.A
  - Postcondiciones: N.A
- ObtenerFrecuencia (): Numérico, retorna la frecuencia dentro del nodo
  - Precondiciones: N.A
  - Postcondiciones: N.A
- FijarFrecuencia (Frecuencia): No retorna, Asigna una frecuencia al nodo
  - Precondiciones: N.A
  - Postcondiciones: N.A
- ObtenerHijoIzq (): Referencia a TAD NodoHuffman, Retorna el nodo asociado como hijo izquierdo
  - Precondiciones: Debe existir el hijo izquierdo
  - Postcondiciones: N.A
- FijarHijoIzq (TAD NodoHuffman): No retorna, Asigna un NodoHuffman como hijo izquierdo
  - Precondiciones: N.A
  - Postcondiciones: N.A
- ObtenerHijoDer (): Referencia a TAD NodoHuffman, Retorna el nodo asociado como hijo derecho

    - Precondiciones: Debe existir el hijo derecho
    - Postcondiciones: N.A
- FijarHijoDer (TAD NodoHuffman): No retorna, Asigna un NodoHuffman como hijo derecho
    - Precondiciones: N.A
    - Postcondiciones: N.A
- ObtenerHijoIzq (): Referencia a TAD NodoHuffman, Retorna el nodo asociado como hijo izquierdo
    - Precondiciones: Debe existir el hijo izquierdo
    - Postcondiciones: N.A
- EsHoja (): Booleano, verifica si el nodo tiene hijos
    - Precondiciones: N.A
    - Postcondiciones: Si el nodo tiene algún hijo, retorna falso, si no retorna verdadero

TAD: ArbolHuffman

Descripción/propósito: Su función es constituir el árbol de Huffman, almacenando las bases del genoma junto a sus frecuencias

Estado:

- Raiz(raíz): Referencia a NodoHuffman, representa el nodo de inicio del ArbolHuffman

Interfaz:

- ArbolHuffman (): TAD ArbolHuffman, Crea un ArbolHuffman, con la raíz nula
    - Precondiciones: N.A
    - Postcondiciones: N.A
- EsVacio (): Booleano, Evalúa si el árbol tiene un nodo raíz
    - Precondiciones: N.A
    - Postcondiciones: Si la raíz del árbol no existe, retorna falso.
- DatoRaiz (): Numérico, Retorna la suma total de frecuencias
    - Precondiciones: La raíz debe existir, no debe estar vacía
    - Postcondiciones: N.A
- ObtenerRaiz (): TAD NodoHuffman, Permite obtener la raíz del Arbol
    - Precondiciones: La raíz no debe ser null
    - Postcondiciones: N.A
- Altura(): Numérico, retorna la suma de las alturas, es decir, la altura del arbol
    - Precondiciones: N.A
    - Postcondiciones: N.A
- Altura (NodoHuffman): Numérico, retorna la altura del NodoHuffman que se recibe por parámetro
    - Precondiciones: N.A
    - Postcondiciones: N.A
- Tamano(): Numérico, retorna el tamaño del arbol
    - Precondiciones: N.A
    - Postcondiciones: N.A
- Tamano (NodoHuffman): Numérico, añade una unidad más al tamaño
    - Precondiciones: N.A
    - Postcondiciones: N.A
- Generar (Conjunto de caracteres, conjunto de enteros): No retorna, Conforma los nodos y los organiza en el árbol, hasta llegar a la raíz

- Precondiciones: Debe existir un conjunto de bases y un conjunto de sus frecuencias, que deben estar sincronizados
- Postcondiciones: Se genera un ArbolHuffman para codificar las bases

TAD MODIFICADOS

TAD: Genoma

Descripción/propósito: Su función es identificar y distinguir un grupo de secuencias, que representan y almacenan toda la información de un código FASTA

Estado:

- secuencias(sqnces): Conjunto de TAD secuencias, representa un contenedor para todas las secuencias que contenga un archivo FASTA

NUEVOS ATRIBUTOS

- Bases(basesPresentes): Conjunto de caracteres, representa un contenedor para las bases que están presentes en todas las secuencias del genoma
- Frecuencia (frecuencias): Conjunto de datos numéricos, representa un contenedor para las frecuencias presentes en todas las secuencias del genoma
- Codigos (codigos): Conjunto de cadenas de caracteres, representa un contenedor para los códigos generados a partir del arbol de Huffman

Interfaz:

- GetSqnces(): Conjunto de TAD secuencias, extrae las secuencias pertenecientes al genoma
  - Precondiciones: N.A
  - Postcondiciones: Se obtiene una lista de secuencias correspondientes al genoma

- SetSqnces(Conjunto de TAD secuencias): No retorna, asigna un conjunto de secuencias dentro del genoma
  - Precondiciones: N.A
  - Postcondiciones: El genoma tiene asignado un conjunto de secuencias

- Es_subsecuencia(cadena de caracteres): no retorna, muestra por pantalla si un dato de entrada pertenece a alguna secuencia almacenada en el sistema
  - Precondiciones: Debe recibirse una cadena de caracteres
  - Postcondiciones: N.A
- Enmascarar (cadena de caracteres): no retorna, reemplaza una subsecuencia por X en cualquiera de las secuencias
  - Precondiciones: Debe recibirse una cadena de caracteres
  - Postcondiciones: En las secuencias se reemplazan los valores coincidentes por cadenas conformadas de ‘X’

NUEVAS OPERACIONES

- GetBasesPresentes(): Conjunto de caracteres, extrae las bases presentes en todas las secuencias en memoria
    - Precondiciones: N.A
    - Postcondiciones: Se obtiene un conjunto de caracteres correspondientes a las bases existentes en la memoria

- SetBasesPresentes(Conjunto de caracteres): No retorna, asigna un conjunto de caracteres dentro del genoma
    - Precondiciones: N.A
    - Postcondiciones: El genoma tiene asignado un conjunto de caracteres

- GetFrecuencia(): Conjunto de datos numéricos, extrae las frecuencias de las bases presentes en todas las secuencias en memoria
    - Precondiciones: N.A
    - Postcondiciones: Se obtiene un conjunto de datos numéricos correspondientes a las repeticiones de bases en las secuencias existentes en la memoria

- SetFrecuencia(Conjunto de datos numéricos): No retorna, asigna un conjunto de datos numéricos dentro del genoma
    - Precondiciones: N.A
    - Postcondiciones: El genoma tiene asignado un conjunto de datos numéricos que indican las repeticiones de cada base presente en el mismo
- GetCodigos(): Conjunto de cadenas de caracteres, extrae los códigos correspondientes a las bases presentes en todas las secuencias en memoria
    - Precondiciones: N.A
    - Postcondiciones: Se obtiene un conjunto de cadenas de caracteres correspondientes a los códigos de las bases existentes en la memoria
- SetCodigos(Conjunto de caracteres): No retorna, asigna un conjunto de caracteres dentro del genoma
    - Precondiciones: N.A
    - Postcondiciones: El genoma tiene asignado un conjunto de caracteres que representan a las bases
- ObtenerCodigo(TAD NodoHuffman, cadena de caracteres): No retorna, Genera un código único para cada uno
    - Precondiciones: Debe haberse generado un Arbol de Huffman en el genoma
    - Postcondiciones: Se obtiene un conjunto de caracteres correspondientes a los códigos equivalentes a las bases existentes en la memoria

- Codificar(Conjunto de caracteres): No retorna, conforma todos los conjuntos necesarios para crear el árbol Huffman y generar los códigos de cada base
    - Precondiciones: N.A
    - Postcondiciones: El genoma tiene asignado un conjunto de caracteres con sus bases presentes, un conjunto de datos numéricos con las frecuencias de cada base y un conjunto de cadena de caracteres con los códigos de cada base

TAD: Sistema
Descripción/propósito: Su función es gestionar y administrar los archivos FASTA, así como emplear su información para mostrarla por pantalla
Estado:
Genoma (genoma): TAD Genoma, Representa el contenedor donde se almacenan la información del archivo FASTA
Interfaz:

- getGenoma (): TAD Genoma, obtiene el genoma en el que se cargan las secuencias del archivo FASTA
    - Precondiciones: N.A
    - Postcondiciones: Una copia del Genoma se almacena todas las secuencias de un archivo.FA
- setGenoma (TAD Genoma): modifica el genoma en el que se cargan las secuencias del archivo FASTA
    - Precondiciones: N.A
    - Postcondiciones: Un nuevo genoma en el sistema con la modificación de la lista de secuencias

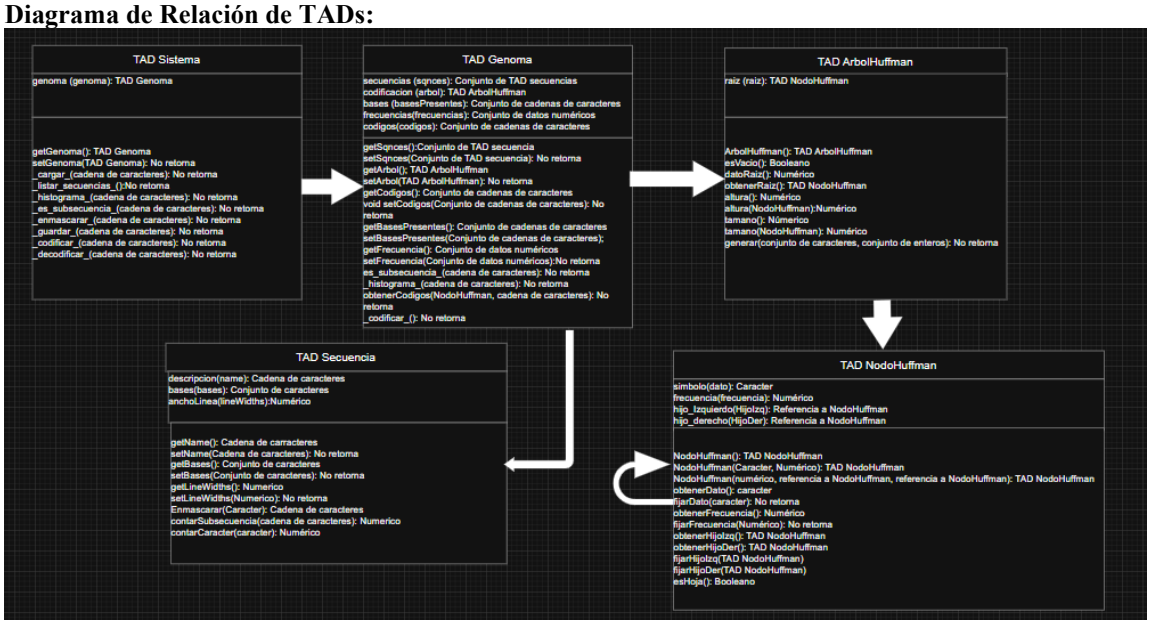
- cargarArchivo (cadena de caracteres): no retorna, carga en el sistema el contenido del archivo fasta, separando la descripción y la secuencia de bases:
    - Precondiciones: N.A
    - Postcondiciones: En una instancia de Genoma se almacena toda la información de un archivo.FA
- listarSecuencias(): no retorna, muestra por pantalla la cantidad de listas almacenadas en memoria, indicando su cantidad de bases y si están completas o no
    - Precondiciones: N.A
    - Postcondiciones: N.A
- Histograma (cadena de caracteres): muestra por pantalla, usando una forma de histograma horizontal, la cantidad de elementos de la tabla de código fasta que se encuentran en determinada secuencia
    - Precondiciones: N.A
    - Postcondiciones: N.A
- Es_subsecuencia (cadena de caracteres): no retorna, muestra por pantalla si un dato de entrada pertenece a alguna secuencia almacenada en el sistema
    - Precondiciones: Debe recibirse una cadena de caracteres
    - Postcondiciones: N.A
- Enmascarar (cadena de caracteres): no retorna, reemplaza una subsecuencia por X en cualquiera de las secuencias
    - Precondiciones: Debe recibirse una cadena de caracteres
    - Postcondiciones: En las secuencias se reemplazan los valores coincidentes por cadenas conformadas de ‘X’
- Guardar (cadena de caracteres): no retorna, escribe dentro de un archivo FASTA el contenido del genoma con sus secuencias modificadas
    - Precondiciones: Debe respetarse el formato FASTA y la justificación de renglones

- Postcondiciones: Se genera un archivo de extensión.fa, con el contenido de las secuencias modificado por el sistema

NUEVAS OPERACIONES

- Codificar (cadena de caracteres): no retorna, comprime dentro de un archivo .fabin toda la información cargada desde un archivo.fasta, ocupando menos espacio en memoria
  - Precondiciones: Debe existir un archivo en formato FASTA que no esté vacío
  - Postcondiciones: Se crea en disco en formato FABIN, que contiene la información del .FASTA
- Decodificar (cadena de caracteres): no retorna, comprime dentro de un archivo .FABIN toda la información cargada desde un archivo .FASTA, ocupando menos espacio en memoria
  - Precondiciones: Debe existir un archivo en formato .FABIN que no esté vacío
  - Postcondiciones: Se borra el contenido actual de la memoria, reemplazandolo por el contenido del archivo .FABIN

Diagrama de Relación de TADs:



Para este segundo componente el TAD sistema tiene ahora la responsabilidad de codificar y decodificar el genoma. El TAD genoma, adquiere nuevos datos, como lo son los conjuntos de caracteres de basesPresentes y códigos, el conjunto de datos numéricos frecuencias y la inclusión de un dato de tipo TAD ArbolHuffman. Además, que ahora es responsable de generar un código para cada base presente a partir de su árbol. Los 2 nuevos TAD hacen referencia a la herramienta que se usa para codificar las bases: El TAD ArbolHuffman, contiene una referencia al TAD NodoHuffman y se encarga de hacer operaciones básicas del árbol además de generar la estructura del árbol a partir de los conjuntos de BasesPresentes y frecuencias que también están dentro del genoma. El TAD NodoHuffman, se usa para conformar el ArbolHuffman, esta implementación

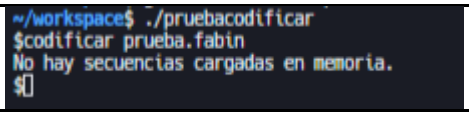
contiene tanto el dato que representa el símbolo (vacio por defecto), como la frecuencia (para todos los nodos internos y la raíz. Tiene 2 referencias a sí mismo, para manejar sus descendientes

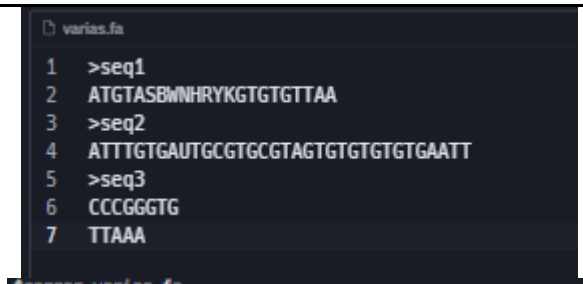
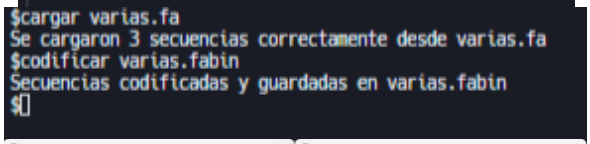
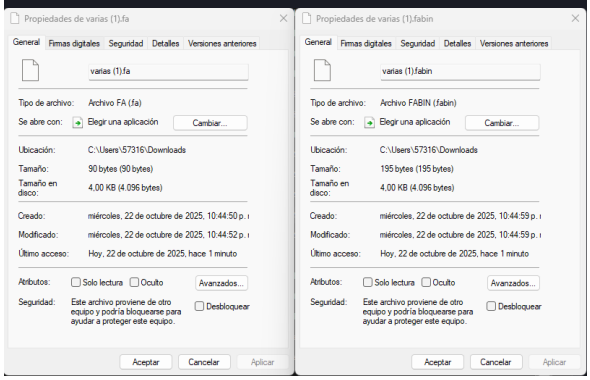
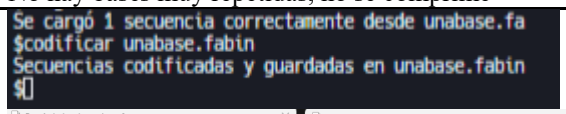
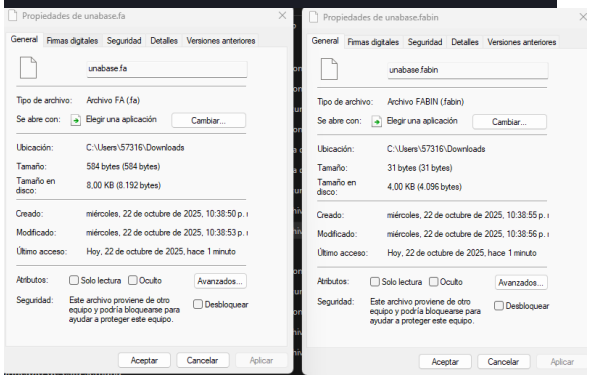
3.3 Plan de Pruebas

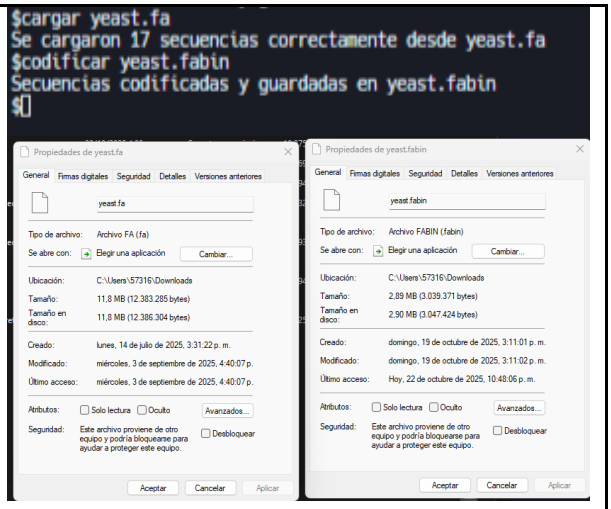
- **Codificar**

Plan de pruebas: CODIFICAR			
Descripción de caso	Valores de entrada	Resultado esperado	Resultado obtenido
No se cargaron secuencias en memoria	Secuencias en memoria: N.A	"No hay secuencias guardades en memoria"	"No hay secuencias guardades en memoria
Se codifica existosamente, un archivo con 7 lineas	Secuencias en memoria (del archivo varias.fa): >seq1 ATGTASBWNHRYKGTGTGTTAA >seq2 ATTTGTGAUTGCGTGCGTAGTGTGTGTGTGAATT >seq3 CCCGGGTGTTAAA	"Secuencias decodificadas en el archivo variassecuencias.fabin".	"Secuencias decodificadas en el archivo variassecuencias.fabin"
Se codifica exitosamente un archivo con una sola base	Secuencias en memoria (del archivo unabase.fa) >seq1 AAAAAAAAAAA AAAAAAA AAAAAAAAA AAAAAAAA AAAAAAAAAAAAAAA AAAAAA AAAAAAAAA AAAAAAAAAA AAAAAAAAAA AAAAAAAAAA AAAAAAAAAA AAAAAAAAAA AAAAA AAAAAAAAAA AAAAAAAAAA AAAAAAAAAA AAAAAAAAA AAAAAAAAA AAAAAAAAAA AAAA AAA	"Secuencias decodificadas en el archivo unabase.fabin.También se espera que el tamaño del .fabin sea significativamente menor por la repetición de bases	"Secuencias decodificadas en el archivo unabase.fabin". "Secuencias decodificadas en el archivo unabase.fabin.También se espera que el tamaño del .fabin sea significativamente menor por la repetición de bases

	AAAAAAAA AAAAAAA AAAAAAA AAAAAAAA AAAAAAAA AAAAAAAA AAAAAAAAA AAAAAAAAAA AAAAAA AAAAAAA AAAAAAAA AAAAAAA AAAAAAA AAAAAAA AAAAAAA A		
Se codifica un archivo con +17000 lineas (yeast.fa)	Secuencias en memoria: Las secuencias almacenadas en yeast.fa (archivo brindado por el profesor)	"Secuencias decodificadas en el archivo yeast.fabin".También se espera que el tamaño del .fabin sea significativamente menor por la repetición de bases	"Secuencias decodificadas en el archivo yeast.fabin".

Caso	Evidencia
No se cargaron subsecuencias en memoria	

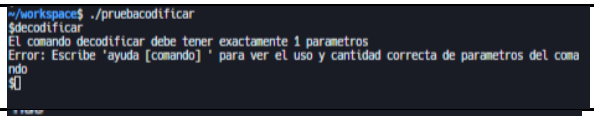
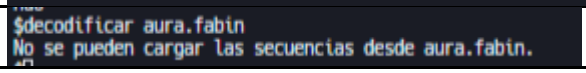
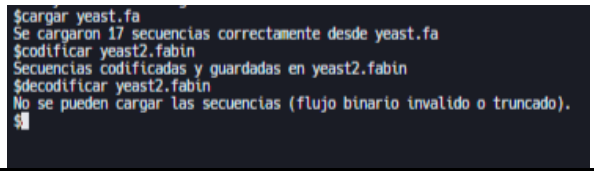
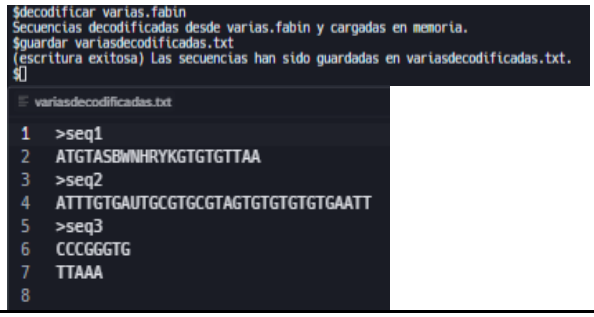
Se codifica existosamente, un archivo con 7 lineas y bases distintas	   No hay bases muy repetidas, no se comprime
Se codifica exitosamente un archivo con una sola base	  Se comprime teniendo en cuenta la repetición de la base
Se codifica un archivo con +17000 lineas Y solo 4 bases(yeast.fa)	

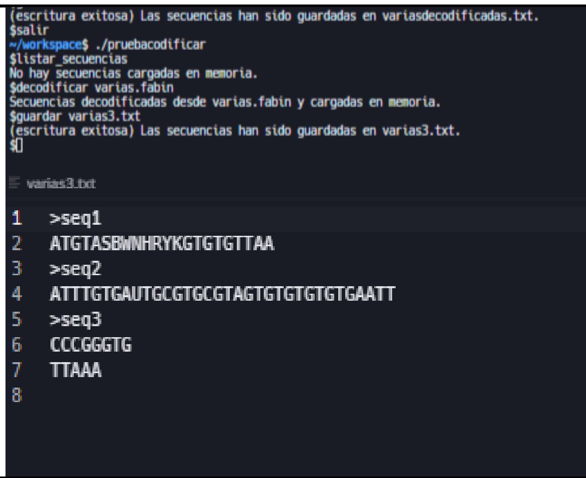
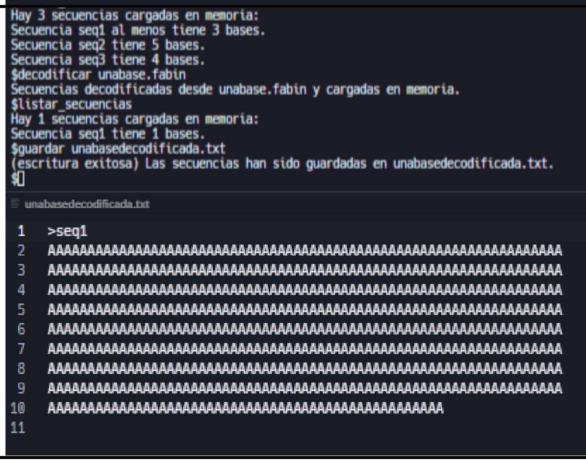
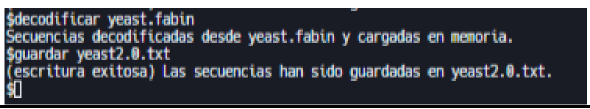
	

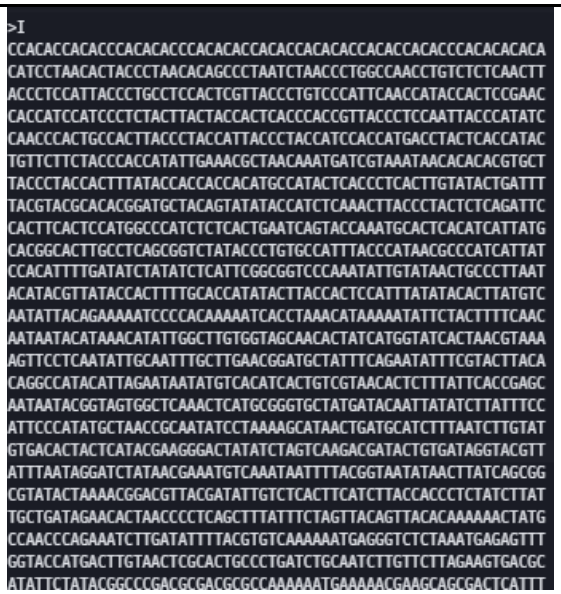
- **DECODIFICAR**

Plan de pruebas: Decodificar			
Descripción de caso	Valores de entrada	Resultado esperado	Resultado obtenido
No se cargó un archivo de formato.fabin	Secuencias en memoria: N.A		"No hay secuencias guardades en memoria
Se decodifica desde un archivo inexistente	Secuencias en memoria: Archivo que no existe: Aura.fabin		
Se decodifica existosamente, un archivo con 7 lineas	Secuencias en memoria (del archivo varias.fabin, proveniente de varias.fa) >seq1 ATGTASBWNHRYKGTGTGTTAA >seq2 ATTTGTGAUTGCGTGCGTAGTGTGTGTGTGAATT >seq3 CCCGGGTGTTAAA	"Secuencias decodificadas desde archivo varias.fabin".	"Secuencias decodificadas desde archivo varias.fabin"
Se decodifica luego de reiniciar la memoria	Secuencias en memoria: N.A Archivo a decodificar varias.fabin	"Secuencias decodificadas desde archivo varias.fabin".	"Secuencias decodificadas desde archivo varias.fabin".

Se decodifica un .fabin que contiene un genoma con una sola base. Teniendo la memoria ocupada	Secuencias en memoria: Secuencias en memoria (del archivo varias.fabin, proveniente de varias.fa) >seq1 ATGTASBWNHRYKGTGTGTTAA >seq2 ATTTGTGAUTGCGTGCGTAGTGTGTGTGTGAATT >seq3 CCCGGGTGTTAAA	Secuencias decodificadas desde unabase.fabin y cargadas en memoria.	Secuencias decodificadas desde unabase.fabin y cargadas en memoria.
Se decodifica un archivo con +17000 lineas (yeast.fa)	Secuencias en memoria: Las secuencias almacenadas en yeast.fa (archivo brindado por el profesor)	Secuencias decodificadas desde yeast.fabin y cargadas en memoria.	Secuencias decodificadas desde yeast.fabin y cargadas en memoria.

Caso	Evidencia
No se cargó un archivo de formato.fabin	
Se decodifica un archivo inexistente	
El archivo está corrupto (manualmente)	
Se decodifica existosamente, un archivo con 7 lineas	

Se decodifica tras reiniciar la memoria y el programa	
Se decodifica un archivo con una sola base	
Se decodifica un archivo con +17000 lineas (yeast.fa)	



3.4 Correcciones segunda entrega

Probando la función decodificar y escribiendo el archivo una vez descomprimido, se identificó que las secuencias no se estaban escribiendo completas, es decir, iniciaba de forma correcta, sin embargo, la secuencia, debido a su longitud, quedaba incompleta, de esta forma reduciendo la similitud entre el archivo original y el decodificado casi en un 100%.

Por esta razón, se modificó la función escribir, eliminando la parte que modificaba las bases a escribir, casteandolas a String e imponiendo un límite sobre ellas. Este fragmento se corrigió y reemplazó, colocando en su lugar una escritura que parte directamente desde el conjunto de caracteres de las bases, tal como se documenta a continuación

```
void Sistema::_guardar_(string nombre_archivo) {
    if (genoma.getSqnces().empty()) {
        cout << "(no hay secuencias cargadas) No hay secuencias cargadas en memoria." << endl;
        return;
    }

    ofstream archivo(nombre_archivo); // modo texto
    if (!archivo.is_open()) {
        cout << "(problemas en archivo) Error guardando en " << nombre_archivo << "." << endl;
        return;
    }
        for (size_t i = 0; i < genoma.getSqnces().size(); ++i) {
            Secuencia &s = genoma.getSqnces()[i];
            archivo << ">" << s.getName() << "\n";

            vector<char> &bases = s.getBases();
            size_t pos = 0;

            if (!s.getLineWidths().empty()) {
                for (size_t k = 0; k < s.getLineWidths().size() && pos < bases.size(); ++k) {
                    size_t ancho = s.getLineWidths()[k];
                    size_t take = min(ancho, bases.size() - pos);


                    for (size_t j = 0; j < take; ++j)
                        archivo << bases[pos + j];
                    archivo << "\n";
                    pos += take;
                }
                while (pos < bases.size()) {
                    size_t ancho = s.getLineWidths().back();
                    size_t take = min(ancho, bases.size() - pos);
                    for (size_t j = 0; j < take; ++j)
                        archivo << bases[pos + j];
                    archivo << "\n";
                    pos += take;
                }
            } else {

                size_t ancho = 80;
                while (pos < bases.size()) {
                    size_t take = min(ancho, bases.size() - pos);
                    for (size_t j = 0; j < take; ++j)
                        archivo << bases[pos + j];
                    archivo << "\n";
                    pos += take;
                }
            }
        }

        archivo.close();
        cout << "(escritura exitosa) Las secuencias han sido guardadas en "
             << nombre_archivo << "." << endl;
    }
}
```

Ahora en la función, se recorren todas las secuencias cargadas en el sistema y se carga en un vector auxiliar las bases de cada secuencia, y se va recorriendo las bases de la secuencia por medio de una variable que funciona como cursor. En el segundo ciclo for, se visualiza como se escribe en el archivo de salida, directamente desde el vector que contiene las bases, lo cual evita que la escritura se limite, posibilitando la escritura completa del archivo. Corrigiendo satisfactoriamente el funcionamiento del comando guardar.

4 Componente 3

Dentro del componente 3, se implementarán los comandos

- Ruta_mas_corta <descripcion_secuencia> <i> <j> <x> <y>
- Base_Remota <descripcion_secuencia> <i> <j>

En conjunto, el objetivo de este par de comandos es conocer información adicional sobre la distribución de las bases en las secuencias, mediante el uso e implementación de grafos. Aplicando algunos conceptos y teoría propia de estas estructuras, para verificar distancias entre las bases

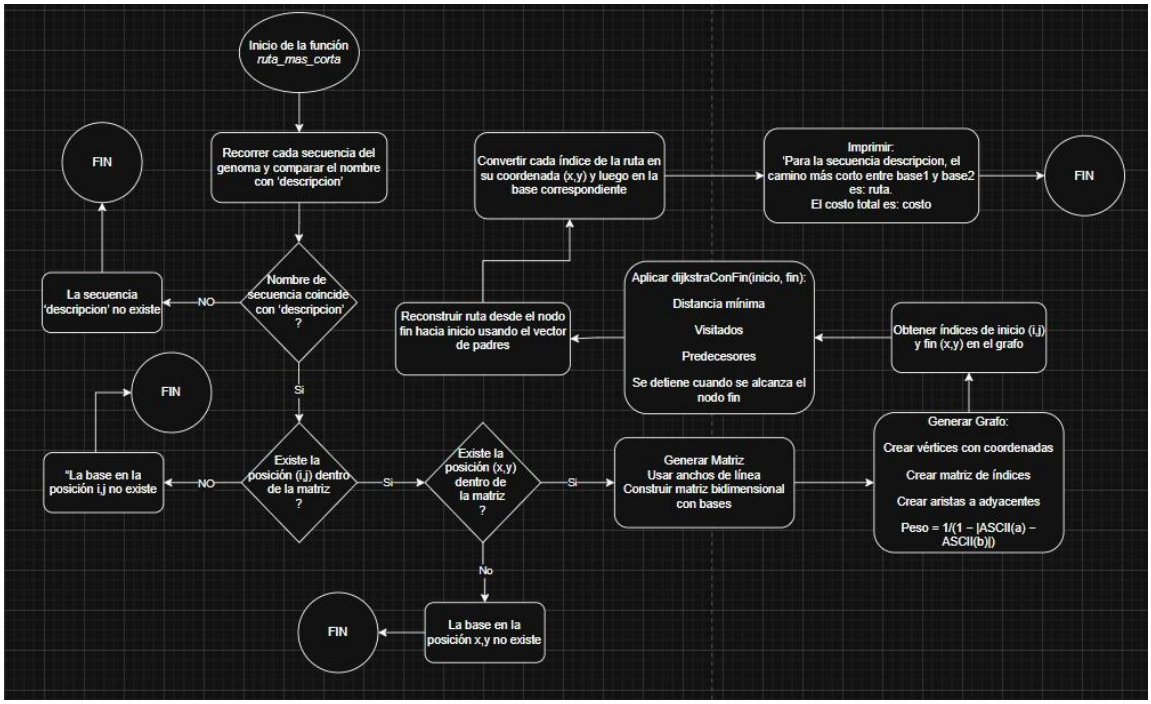
4.1 Descripción de los comandos

- Ruta más corta
  - Condiciones:
    - Existe una secuencia dentro del genoma en memoria, con la descripción coincidente
    - Las coordenadas (i,j) y (x,y) existen dentro de la matriz correspondiente a la secuencia
  - Proceso:
    - Se hace un ciclo para recorrer las secuencias presentes en el genoma
    - Se compara el nombre recibido con las descripciones de los genomas
    - Si no se encuentra una descripción coincidente, se obtiene la salida:
      La secuencia "descripcion" no existe

    - Si se encuentra una descripción coincidente:
      - Se evalúa si las coordenadas i,j existen. Si no es así, tiene la salida:
        "La base en la posicion i,j no existe"
      - Se evalúa si las coordenadas x,y existen. Si no es así, tiene la salida:
        "La base en la posición x,y no existe"
      - Se genera una matriz con las bases, teniendo en cuenta el ancho de linea y la cantidad de filas. Donde se arma incialmente un conjunto, que al llegar al ancho límite, dicho conjunto, pasa a ser parte del conjunto bidimensional resultado
      - Se crea el grafo, guardando en los vértices las coordenadas de la matriz con las bases, y adicionalmente, se crea una matriz o conjunto bidimensional de enteros, que almacena los índices del vector de vértices, para conocer el índice de cada coordenada dentro del grafo
      - Luego, usando la fórmula $\frac{1}{1-|ASCII\,(ij)-ASCII(XY)|}$, se calculan los pesos de las aristas y se agregan en la matriz de adyacencia, uniendo así los vértices con coordenadas adyacentes.
      - Una vez construido el grafo, se aplica un algoritmo de dijkstra modificado para obtener la ruta más corta entre los puntos:
      - Se crean 3 conjuntos auxiliares, uno para vértices visitados, uno para distancias y otro para predecesores.
      - Luego, se inicia un doble ciclo, que verifica vértice por vértice, cual no ha sido visitado y cuál es la distancia mínima. Al finalizar el ciclo interno, se marca el vértice como visitado y se evalúa si el índice del vértice visitado es el índice del vértice destino.
      - El detalle que se altera es este último, donde se evalúa si ya se llegó al índice final o destino
        - Si es el caso, termina el ciclo activo y se pasa a reconstruir la ruta a partir del conjunto de vértices de predecesores
        - Si no es el caso, se evalúa si la suma del peso, más alguna de las distancias del nodo actual es el menor valor y se almacena dentro del vector de distancias

- Una vez finalizados ciclos, se construye la ruta a partir del vector de padres, y se retorna un par de elementos: El conjunto de indices que representa la ruta y la distancia total recorrida en la misma

▪ La salida final es:
"Para la secuencia "descripcion" el camino mas corto entre "base1" y "base2" es: "Secuencia de bases". El costo total es de: "costo"

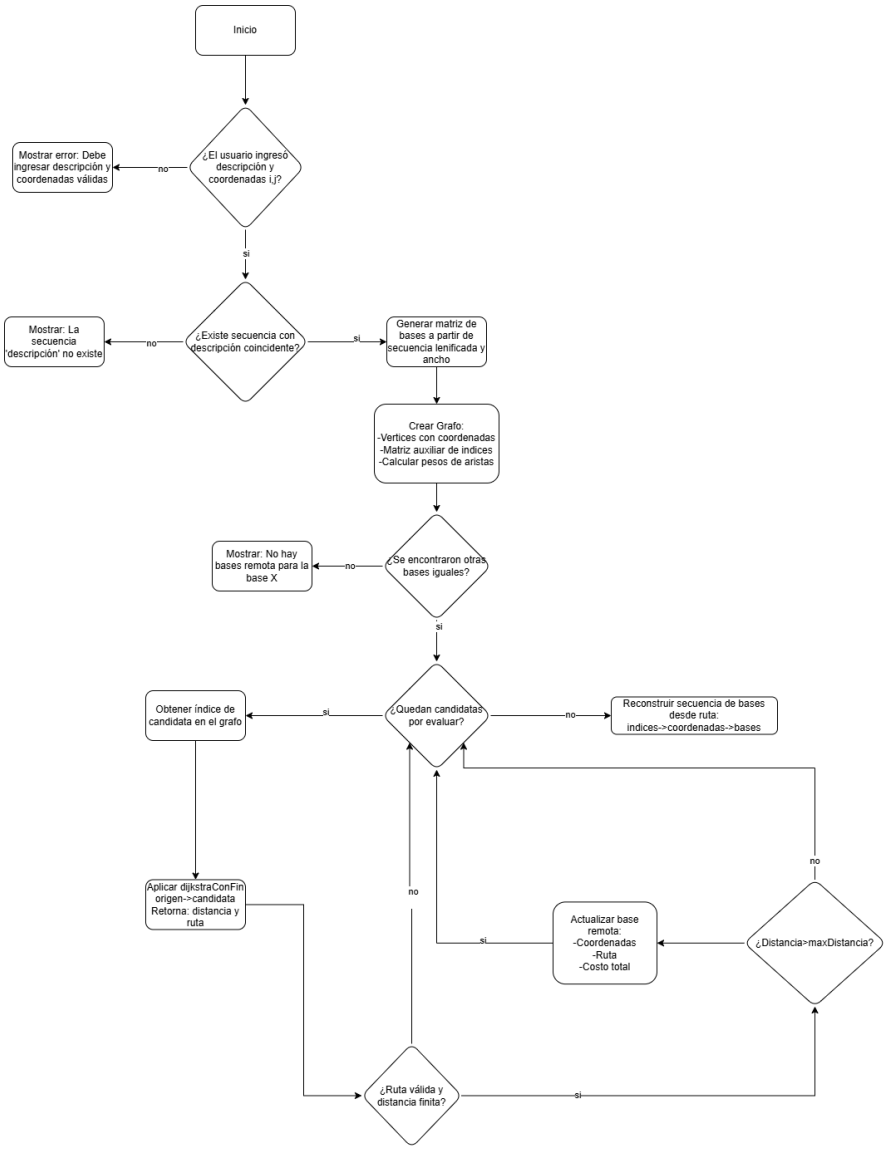


Se evidencia un diagrama de flujo del proceso anteriorente explicado

- Base_remota
  - Condiciones:
    - Existe una secuencia dentro del genoma en memoria, con la descripción coincidente.
    - Las coordenadas (i,j) existen dentro de la matriz correspondiente a la secuencia.
    - Dentro de la misma matriz, existe **al menos otra base igual** a la de la posición (i,j), ubicada en una posición diferente.
  - Proceso:
    - Se hace un ciclo para recorrer las secuencias presentes en el genoma. Se compara el nombre recibido con las descripciones de las secuencias. Si no se encuentra una descripción coincidente, se obtiene la salida: La secuencia "descripcion" no existe
    - Si se encuentra una descripción coincidente:
    - Se genera la matriz con las bases, a partir de la secuencia lineificada y del ancho de línea, construyendo un conjunto bidimensional donde cada celda contiene una base y sus coordenadas. Luego, se crea el grafo correspondiente, almacenando

cada coordenada como vértice y generando, además, una matriz auxiliar que contiene los índices de cada celda dentro del vector de vértices.
Se calculan los pesos de las aristas usando la fórmula $\frac{1}{1-|ASCII\ (ij)-ASCII(XY)|}$, conectando únicamente las posiciones adyacentes dentro de la matriz.

- Se verifica que las coordenadas (i,j) existan dentro de la matriz. Si no es así, la salida es:
  "La base en la posicion i,j no existe"
- Se identifica la base de origen ubicada en (i,j). A continuación, se recorre toda la matriz para buscar todas las demás posiciones que contengan la misma base, excluyendo la posición original (i,j).
  Si no se encuentra ninguna otra coincidencia, la salida es:
  "No hay base remota para la base X"
  donde X es la base de origen.
- Para cada posición candidata encontrada, se obtiene su índice dentro del grafo y se aplica el algoritmo dijkstraConFin, que retorna la distancia mínima entre el nodo de origen y el nodo destino, junto con la ruta de vértices recorridos.
  Si la ruta hallada no es válida o presenta una distancia infinita, dicha candidata se descarta.
  Si es válida y su distancia es mayor que cualquier distancia registrada hasta el momento, se actualiza como la nueva “base remota”, guardando tanto sus coordenadas como su ruta y su costo total.
- Una vez evaluadas todas las posibles posiciones, se reconstruye la ruta desde el vector de índices retornado por Dijkstra. Para cada índice en la ruta, se obtienen sus coordenadas dentro del grafo y se accede a la matriz de bases para recuperar la base correspondiente, formando así la secuencia de bases de la ruta final.
- La salida final es:
  "Para la secuencia 'descripcion', la base remota está ubicada en [x2,y2], y la ruta entre la base en [i,j] y la base remota es: SecuenciaDeBases. El costo total de la ruta es costo."



4.2 Diseño

Descripción de datos abstractos:

NUEVOS TADS:

TAD: Coordenada
Descripción: Es la representación de un par de datos numéricos ordenados, sirve para aplicar los grafos al contexto del robot de carga, debido a que cada vértice va a contener la coordenada donde se encuentra una caja.
Estado:

- Coordenada X (x): Conjunto de elementos, representa la lista de todos los vértices que componen el grafo
- Coordenada Y (y): Conjunto bidimensional de datos numéricos, representa las aristas o conexiones entre los vértices. Sus valores en cada posición representan los pesos de cada arista.

Interfaz:

- Coordenada (numérico, numérico): TAD Coordenada, instancia una coordenada, con 2 valores numéricos
  - Precondiciones: N.A
  - Postcondiciones: Se instancia una coordenada con un par de datos numéricos
- getX(): Numérico, permite el acceso al componente Y de la coordenada
  - Precondiciones: Debe existir una coordenada con valores numéricos
  - Postcondiciones: Se obtiene un dato numérico
- setX(Numérico): No retorna asigna el componente Y de la coordenada
  - Precondiciones: N.A
  - Postcondiciones: El componente Y de la coordenada tiene un nuevo valor asignado

TAD: Grafo

Descripción/Propósito: Es la representación de la estructura de un grafo, cuenta con vértices, aristas y la posibilidad de mencionar si es dirigido o no. Cuenta también con algunos recorridos implementados. Se construyó con el uso de plantillas, es decir que sus vértices pueden componerse de diferentes tipos de datos facilitando su reutilización en futuros contextos.
Estado:

- Vertices (vertices): Conjunto de elementos, representa la lista de todos los vértices que componen el grafo
- Aristas (matrizDeAdyacencia): Conjunto bidimensional de datos numéricos, representa las aristas o conexiones entre los vértices. Sus valores en cada posición representan los pesos de cada arista.
- Tipo (dirigido): Booleano, indica si el grafo es dirigido o no. Por defecto se crea NO dirigido

Interfaz:

- Grafo (): TAD Grafo, Instancia un grafo sin vértices, no dirigido
  - Precondiciones: N.A

    - Postcondiciones: Se crea un grafo no dirigido
- Grafo(booleano): TAD Grafo, instancia un gráfo sin vértices, con dirigido
    - Precondiciones: N.A
    - Postcondiciones: Se crea un grafo dirigido
- esDirigido(): Booleano, indica si el grafo es o no dirigido
    - Precondiciones: N.A
    - Postcondiciones: Se obtiene 1 (true), si el grafo es dirigido y 0 (false) para el caso contrario
- setDirigido(booleano): No retorna, cambia el modo del grafo, de dirigido o no dirigido
    - Precondiciones: N.A
    - Postcondiciones: El grafo cambia su estado de dirigido a NO dirigido o viceversa
- obtenerMatrizDeAdyacencia(): Conjunto bidimensional de datos numéricos, permite acceder a la representación de las aristas del grafo
    - Precondiciones: N.A
    - Postcondiciones: N.A
- obtenerVertices(): Conjunto de datos, permite acceder al conjunto de vértices presentes dentro del grafo
    - Precondiciones: N.A
    - Postcondiciones: N.A
- agregarVertice(Dato): No retorna, permite añadir un nuevo dato al grafo sin aristas que lo conecten
    - Precondiciones: N.A
    - Postcondiciones: Se agrega un nuevo elemento al conjunto de vértices del grafo, desconectado de todos los demás
- agregarArista(numérico, numérico, numérico): No retorna, permite crear una arista entre 2 vertices
    - Precondiciones: Debe existir al menos un vértice dentro del grafo
    - Postcondiciones: Se crea una nueva arista, cuyo peso se verá reflejado en la matriz de adyacencia
- preOrden(numérico): No retorna, realiza el recorrido del grafo en profundidad, desde el dato correspondiente al valor numérico de entrada y lo imprime en pantalla
    - Precondiciones: N.A
    - Postcondiciones: N.A
- recorridoVecindario (numérico): No retorna, realiza el recorrido del grafo en anchura, desde el dato correspondiente al valor numérico de entrada y lo imprime en pantalla
    - Precondiciones: N.A
    - Postcondiciones: N.A
- DijkstraConFin (numérico, numérico): Par de datos (Conjunto de datos numéricos, numérico), Se encarga de realizar el recubrimiento de costo mínimo, empleando el algoritmo

de dijkstra, calculando distancias, marcando vértices visitados y vértices padres, consiguiendo las rutas más cortas entre el vértice inicial y el final

- Precondiciones: El grafo debe estar inicializado, con vértices y aristas
- Postcondiciones: Se genera un par de datos, compuestos por la ruta más corta y su costo total

TADS Modificados:

TAD: Secuencia

Descripción/propósito: Su función es identificar y distinguir un grupo de bases nitrogenadas ordenadas, que representan una parte de un código genético
Estado:

- Description (name): Cadena de caracteres, se encarga de darle una identificación para distinguir la secuencia
- Bases_nitrogenadas (bases): Conjunto de caracteres, representa un contenedor para todas las letras que contiene una secuencia
- Matriz de bases (matriz): Conjunto bidimensional de caracteres, representa una matriz que conserva la forma de la secuencia, teniendo en cuenta el ancho de línea y la cantidad de líneas
- Grafo (Grafo): TAD Grafo, representa la estructura de datos generada a partir de la matriz de bases
- Matriz de indices (indices): Conjunto bidimensional de índices, representa una matriz que guarda los índices correspondientes a cada vértice del grafo propio de la secuencia

Interfaz:

- getName(): cadena de caracteres, retorna el nombre o descripción de la cadena
  - Precondiciones: N.A
  - Postcondiciones: Una cadena que da acceso al identificador de la secuencia

- setName(cadena de caracteres): no retorna, asigna nombre o descripción de la cadena
  - Precondiciones: N.A
  - Postcondiciones: La secuencia ahora tiene asignado un identificador
- getBases(): conjunto de caracteres, retorna un conjunto de caracteres
  - Precondiciones: N.A
  - Postcondiciones: Un conjunto que muestra las bases de la secuencia

- setBases(conjunto de caracteres): no retorna, asigna un conjunto de caracteres a la secuencia
  - Precondiciones: N.A
  - Postcondiciones: La secuencia ahora tiene asignado un grupo de caracteres

- getLineWidth(): numerico, retorna el tamaño de una linea de la secuencia
  - Precondiciones: N.A
  - Postcondiciones: Un dato numerico que indica cuantas bases tiene una linea de archivo fasta

- setLineWitdth(numerico): no retorna, asigna la cantidad de bases que tiene una linea de archivo fasta

    - Precondiciones: N.A
    - Postcondiciones: La secuencia ahora tiene asignado un ancho de línea
- Equivalencias (caracter): conjunto de caracteres, retorna un conjunto de caracteres correspondiente a la equivalencia de caracteres no definidos en el codigo fasta
    - Precondiciones: Debe recibirse un caracter
    - Postcondiciones: Un conjunto de caracteres que permiten conocer las bases equivalentes al caracter de entrada
- contarSubsecuencia(cadena de caracteres): Numérico, retorna la cantidad de veces que una subsecuencia está presente en la cadena
    - Precondiciones: Debe recibirse una cadena de caracteres y deben existir subsecuencias en memoria
    - Postcondiciones: Un valor numérico que indica las repeticiones de un conjunto de caracteres específicos en la secuencia que está en memoria
- Enmascarar (cadena de caracteres): Numérico, retorna la cantidad de veces que una subsecuencia está presente en la cadena, y reemplaza en lugar de las subsecuencias con 'X'
    - Precondiciones: Debe recibirse una cadena de caracteres y deben existir subsecuencias en memoria
    - Postcondiciones: Un valor numérico que indica las repeticiones de un conjunto de caracteres específicos en la secuencia que está en memoria

- contarCaracter(caracter): Numérico, retorna la cantidad de veces que un carácter está presente en la secuencia
    - Precondiciones: NA
    - Postcondiciones: Un valor numérico que indica las repeticiones de un carácter específico en la secuencia que está en memoria

NUEVAS FUNCIONES:

- generarMatriz(): No retorna, Genera una matriz con las bases propias de la secuencia, respetando la cantidad de líneas y su ancho
    - Precondiciones: La secuencia debe tener bases
    - Postcondiciones: Se genera un conjunto bidimensional de caracteres que representa en forma de matriz la secuencia completa

- generarGrafo(): No retorna, Genera un grafo con base en la matriz de bases, con cada vértice siendo la coordenada dentro de la matriz de bases, y estableciendo las aristas entre coordenadas adyacentes.
    - Precondiciones: La matriz de bases debe estar previamente establecida
    - Postcondiciones: Se establece el TAD grafo correspondiente a la seucnia, junto a su matriz de indices

- calcularDistancia(numérico, numérico): numérico, realiza el cálculo para determinar el peso de las
    - Precondiciones: N.A
    - Postcondiciones: Se genera un dato numérico.

- getMatriz(): conjunto bidimensional de caracteres, retorna la matriz correspondiente a la secuencia
    - Precondiciones: La función generarMatriz fue ejecutada anteriormente

    - Postcondiciones: Un conjunto que contiene la secuencia, almacenada en forma de matriz

- getIndices(): conjunto bidimensional de datos numéricos, retorna la matriz que almacena los índices correspondientes a cada vértice del grafo
    - Precondiciones: La función generarGrafo fue ejecutada anteriormente
    - Postcondiciones: Un conjunto bidimensional con números, paralelo al de la matriz de bases

- getGrafo(): TAD Grafo, retorna el grafo correspondiente a la matriz de bases, incluyendo sus vértices y matriz de adyacencia
    - Precondiciones: La función generarGrafo fue ejecutada anteriormente
    - Postcondiciones: Un dato numerico que indica cuantas bases tiene una linea de archivo fasta

TAD: Genoma

Descripción/propósito: Su función es identificar y distinguir un grupo de secuencias, que representan y almacenan toda la información de un código FASTA

Estado:

- secuencias(sqnces): Conjunto de TAD secuencias, representa un contenedor para todas las secuencias que contenga un archivo FASTA
- Bases(basesPresentes): Conjunto de caracteres, representa un contenedor para las bases que están presentes en todas las secuencias del genoma
- Frecuencia (frecuencias): Conjunto de datos numéricos, representa un contenedor para las frecuencias presentes en todas las secuencias del genoma
- Codigos (codigos): Conjunto de cadenas de caracteres, representa un contenedor para los códigos generados a partir del arbol de Huffman

Interfaz:

- GetSqnces(): Conjunto de TAD secuencias, extrae las secuencias pertenecientes al genoma
    - Precondiciones: N.A
    - Postcondiciones: Se obtiene una lista de secuencias correspondientes al genoma

- SetSqnces(Conjunto de TAD secuencias): No retorna, asigna un conjunto de secuencias dentro del genoma
    - Precondiciones: N.A
    - Postcondiciones: El genoma tiene asignado un conjunto de secuencias

- Es_subsecuencia(cadena de caracteres): no retorna, muestra por pantalla si un dato de entrada pertenece a alguna secuencia almacenada en el sistema
    - Precondiciones: Debe recibirse una cadena de caracteres
    - Postcondiciones: N.A
- Enmascarar (cadena de caracteres): no retorna, reemplaza una subsecuencia por X en cualquiera de las secuencias
    - Precondiciones: Debe recibirse una cadena de caracteres
    - Postcondiciones: En las secuencias se reemplazan los valores coincidentes por cadenas conformadas de ‘X’

- GetBasesPresentes(): Conjunto de caracteres, extrae las bases presentes en todas las secuencias en memoria
    - Precondiciones: N.A
    - Postcondiciones: Se obtiene un conjunto de caracteres correspondientes a las bases existentes en la memoria

- SetBasesPresentes(Conjunto de caracteres): No retorna, asigna un conjunto de caracteres dentro del genoma
    - Precondiciones: N.A
    - Postcondiciones: El genoma tiene asignado un conjunto de caracteres

- GetFrecuencia(): Conjunto de datos numéricos, extrae las frecuencias de las bases presentes en todas las secuencias en memoria
    - Precondiciones: N.A
    - Postcondiciones: Se obtiene un conjunto de datos numéricos correspondientes a las repeticiones de bases en las secuencias existentes en la memoria

- SetFrecuencia(Conjunto de datos numéricos): No retorna, asigna un conjunto de datos numéricos dentro del genoma
    - Precondiciones: N.A
    - Postcondiciones: El genoma tiene asignado un conjunto de datos numéricos que indican las repeticiones de cada base presente en el mismo
- GetCodigos(): Conjunto de cadenas de caracteres, extrae los códigos correspondientes a las bases presentes en todas las secuencias en memoria
    - Precondiciones: N.A
    - Postcondiciones: Se obtiene un conjunto de cadenas de caracteres correspondientes a los códigos de las bases existentes en la memoria
- SetCodigos(Conjunto de caracteres): No retorna, asigna un conjunto de caracteres dentro del genoma
    - Precondiciones: N.A
    - Postcondiciones: El genoma tiene asignado un conjunto de caracteres que representan a las bases
- ObtenerCodigo(TAD NodoHuffman, cadena de caracteres): No retorna, Genera un código único para cada uno
    - Precondiciones: Debe haberse generado un Arbol de Huffman en el genoma
    - Postcondiciones: Se obtiene un conjunto de caracteres correspondientes a los códigos equivalentes a las bases existentes en la memoria

- Codificar(Conjunto de caracteres): No retorna, conforma todos los conjuntos necesarios para crear el árbol Huffman y generar los códigos de cada base
    - Precondiciones: N.A
    - Postcondiciones: El genoma tiene asignado un conjunto de caracteres con sus bases presentes, un conjunto de datos numéricos con las frecuencias de cada base y un conjunto de cadena de caracteres con los códigos de cada base

TAD: Sistema
Descripción/propósito: Su función es gestionar y administrar los archivos FASTA, así como emplear su información para mostrarla por pantalla
Estado:
Genoma (genoma): TAD Genoma, Representa el contenedor donde se almacenan la información del archivo FASTA
Interfaz:

- getGenoma (): TAD Genoma, obtiene el genoma en el que se cargan las secuencias del archivo FASTA
    - Precondiciones: N.A
    - Postcondiciones: Una copia del Genoma se almacena todas las secuencias de un archivo.FA
- setGenoma (TAD Genoma): modifica el genoma en el que se cargan las secuencias del archivo FASTA
    - Precondiciones: N.A
    - Postcondiciones: Un nuevo genoma en el sistema con la modificación de la lista de secuencias

- cargarArchivo (cadena de caracteres): no retorna, carga en el sistema el contenido del archivo fasta, separando la descripción y la secuencia de bases:
    - Precondiciones: N.A
    - Postcondiciones: En una instancia de Genoma se almacena toda la información de un archivo.FA
- listarSecuencias(): no retorna, muestra por pantalla la cantidad de listas almacenadas en memoria, indicando su cantidad de bases y si están completas o no
    - Precondiciones: N.A
    - Postcondiciones: N.A
- Histograma (cadena de caracteres): muestra por pantalla, usando una forma de histograma horizontal, la cantidad de elementos de la tabla de código fasta que se encuentran en determinada secuencia
    - Precondiciones: N.A
    - Postcondiciones: N.A
- Es_subsecuencia (cadena de caracteres): no retorna, muestra por pantalla si un dato de entrada pertenece a alguna secuencia almacenada en el sistema
    - Precondiciones: Debe recibirse una cadena de caracteres
    - Postcondiciones: N.A
- Enmascarar (cadena de caracteres): no retorna, reemplaza una subsecuencia por X en cualquiera de las secuencias
    - Precondiciones: Debe recibirse una cadena de caracteres
    - Postcondiciones: En las secuencias se reemplazan los valores coincidentes por cadenas conformadas de ‘X’
- Guardar (cadena de caracteres): no retorna, escribe dentro de un archivo FASTA el contenido del genoma con sus secuencias modificadas
    - Precondiciones: Debe respetarse el formato FASTA y la justificación de renglones

    - Postcondiciones: Se genera un archivo de extensión.fa, con el contenido de las secuencias modificado por el sistema

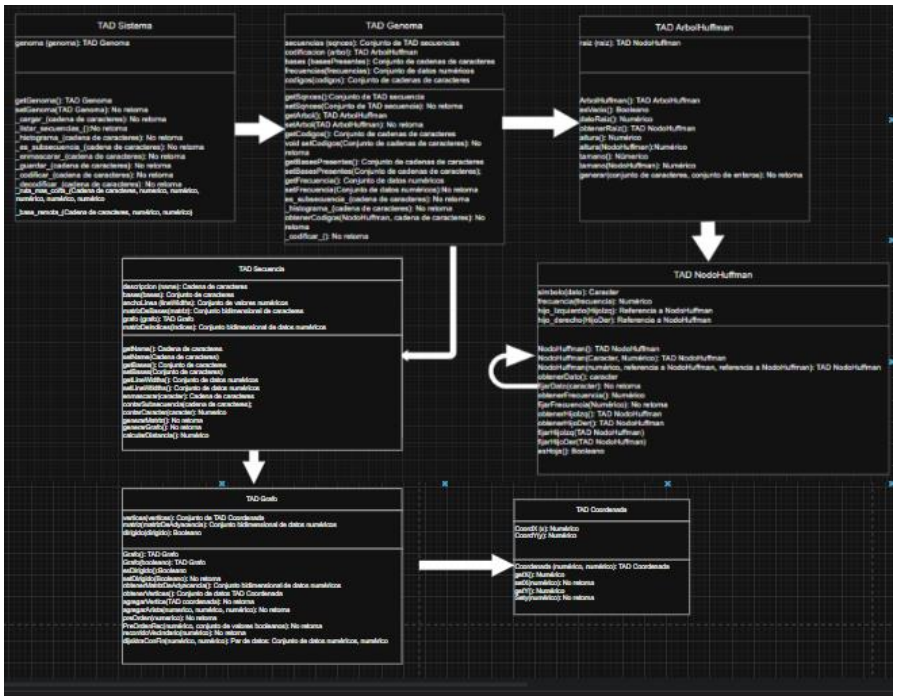
- Codificar (cadena de caracteres): no retorna, comprime dentro de un archivo .fabin toda la información cargada desde un archivo.fasta, ocupando menos espacio en memoria
    - Precondiciones: Debe existir un archivo en formato FASTA que no esté vacío
    - Postcondiciones: Se crea en disco en formato FABIN, que contiene la información del .FASTA
- Decodificar (cadena de caracteres): no retorna, comprime dentro de un archivo .FABIN toda la información cargada desde un archivo .FASTA, ocupando menos espacio en memoria
    - Precondiciones: Debe existir un archivo en formato .FABIN que no esté vacío
    - Postcondiciones: Se borra el contenido actual de la memoria, reemplazandolo por el contenido del archivo .FABIN

NUEVAS OPERACIONES

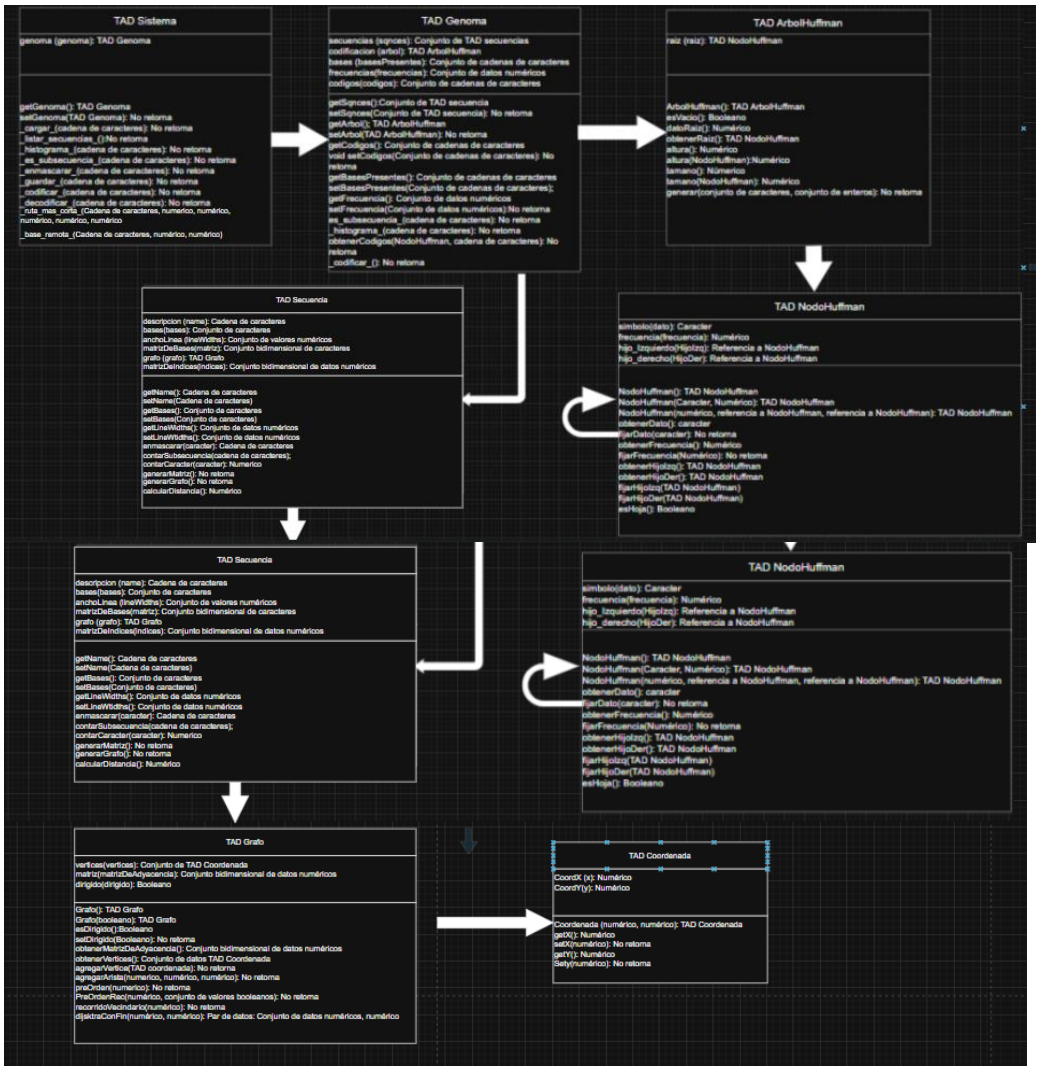
- Ruta_mas_corta (cadena de caracteres, numerico, numerico, numerico, numerico): No retorna, Calcula cual es la ruta más corta entre las coordenadas de 2 bases dadas, apoyándose de la matriz de bases, su respectivo grafo, su matriz de índices, aplicando un algoritmo de dijkstra modificado.
    - Precondiciones: Debe haber secuencias cargadas en memoria
    - Postcondiciones: Se imprime la ruta más corta entre 2 coordenadas de la matriz de bases, con su costo
- Base_remota (cadena de caracteres, numérico, numérico): No retorna, busca las bases que sean iguales a la correspondiente a las coordenadas, las almacena y calcula la ruta más larga entre ellas, seleccionando la ruta y la ubicación de la base más lejana
    - Precondiciones: Debe haber secuencias guardadas en memoria
    - Postcondiciones: Se imprime la ruta entre la distancia seleccionada y la base equivalente que más lejos se encuentra dentro de la secuencia, junto con su costo total

Diagrama de Relación de TADs:

Primero, una ilustración completa, para entender las relaciones entre TADs, luego, una vista con más acercamiento para detallar cada TAD



A continuación, se muestra el mismo diagrama, pero fragmentado para visualizarlo con más claridad:



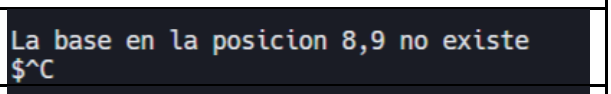
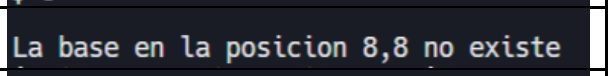
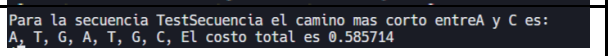
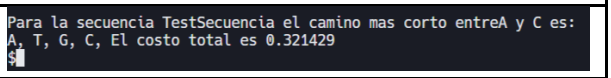
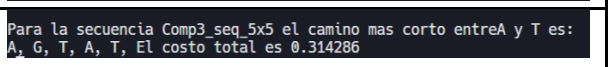
4.3 Plan de pruebas

A continuación, se verifica el correcto funcionamiento de las funciones implementadas en este componente, se plantearon casos variados en ambos casos:

- Ruta más corta:

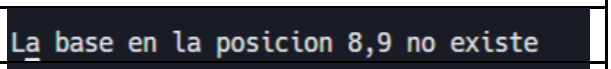
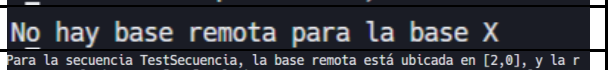
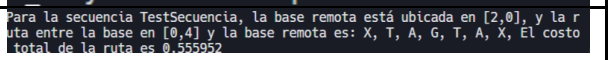
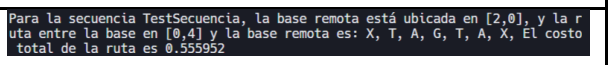
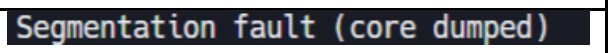
Plan de pruebas: ruta_mas_corta			
Descripción de caso	Valores de entrada	Resultado esperado	Resultado obtenido
La descripción de la secuencia no existe	Secuencias en memoria: varias.fa >TestSecuencia ACGTX TGCAX XATGC	No hay secuencias guardades en memoria	No hay secuencias guardades en memoria
Las coordenadas i j no existen	Secuencias en memoria:varias.fa >TestSecuencia ACGTX TGCAX XATGC (8,9)(1,3)	La base en la posicion 8,9 no existe	La base en la posicion 8,9 no existe
La coordenadas x y no existen	Secuencias en memoria: varias.fa >TestSecuencia ACGTX TGCAX XATGC (1,2)(8,8)	La base en la posicion 8,8 no existe	La base en la posicion 8,8 no existe
Se intenta sacar la distancia mas corta entre los extremos de una matriz rectangular	Secuencias en memoria: varias.fa >TestSecuencia ACGTX TGCAX XATGC (0,0) (2,4)	Para la secuencia TestSecuencia el camino mas corto entreA y C es: A, T, G, A, T, G, C, El costo total es 0.585714	Para la secuencia TestSecuencia el camino mas corto entreA y C es: A, T, G, A, T, G, C, El costo total es 0.585714
Se saca la ruta más corta entre 2 base adyacentes	Secuencias en memoria: varias.fa >TestSecuencia ACGTX TGCAX XATGC (0,0) (0,1)	Para la secuencia TestSecuencia el camino mas corto entreA y C es: A, T, G, C, El costo total es 0.321429	Para la secuencia TestSecuencia el camino mas corto entreA y C es: A, T, G, C, El costo total es 0.321429
Se saca una ruta aleatoria entre 2 bases de una matriz cuadrada	Secuencias en memoria: ev.fa Comp3_seq_5x5 ACGTA CTAGC	Para la secuencia Comp3_seq_5x5 el camino mas corto entreA y T	Para la secuencia Comp3_seq_5x5 el camino mas corto entreA y T

	GACTG TGTAC CCGTA	es: A, G, T, A, T, El costo total es 0.314286	es: A, G, T, A, T, El costo total es 0.314286

Caso	Evidencia
La descripción de la secuencia no existe	
Las coordenadas i j no existen	
La coordenadas x y no existen	
Se intenta sacar la distancia mas corta entre los extremos de una matriz rectangular	
Se saca la ruta más corta entre 2 base adyacentes	
Se saca una ruta aleatoria entre 2 bases de una matriz cuadrada	

Plan de pruebas: Base_remota			
Descripción de caso	Valores de entrada	Resultado esperado	Resultado obtenido
La descripción de la secuencia no existe	Secuencias en memoria: varias.fa >TestSecuencia ACGTX TGCAX XATGC	No hay secuencias guardades en memori	No hay secuencias guardades en memoria
Las coordenadas i j no existen	Secuencias en memoria:varias.fa >TestSecuencia ACGTX TGCAX XATGC (8,9)	La base en la posicion 8,9 no existe	La base en la posicion 8,9 no existe
No hay bases remotas	Secuencias en memoria: varias.fa >TestSecuencia ACGTX TGCAT AATGC	No hay base remota para la base X	No hay base remota para la base X

	(0,4)		
Se intenta sacar la distancia con una única base remota	Secuencias en memoria: varias.fa >TestSecuencia ATGTX TGCAX XATGC (0,4)	Para la secuencia TestSecuencia, la base remota está ubicada en [2,0], y la ruta entre la base en [0,4] y la base remota es: X, T, A, G, T, A, X, El costo total de la ruta es 0.555952	Para la secuencia TestSecuencia, la base remota está ubicada en [2,0], y la ruta entre la base en [0,4] y la base remota es: X, T, A, G, T, A, X, El costo total de la ruta es 0.555952
Se saca la base remota con varias coincidentes	Secuencias en memoria: varias.fa >TestSecuencia ACGTX TGCAX XATGC (0,4)	Para la secuencia TestSecuencia, la base remota está ubicada en [2,0], y la ruta entre la base en [0,4] y la base remota es: X, T, A, G, T, A, X, El costo total de la ruta es 0.555952	Para la secuencia TestSecuencia, la base remota está ubicada en [2,0], y la ruta entre la base en [0,4] y la base remota es: X, T, A, G, T, A, X, El costo total de la ruta es 0.555952
Se saca la distancia mas lejana entre 2 coincidentes de una matriz cuadrada	Secuencias en memoria:ev.fa Comp3_seq_5x5 ACGTA CTAGC GACTG TGTAC CCGTA (1,4)		Segmentation fault (core dumped)

Caso	Evidencia
La descripción de la secuencia no existe	
Las coordenadas i j no existen	
No hay bases remotas	
Se intenta sacar la distancia con una única base remota	
Se saca la base remota con varias coincidentes	
Se saca la distancia mas lejana entre 2 coincidentes de una matriz cuadrada	

Referencias

1. *FASTA format for nucleotide sequences*. (n.d.). https://www.ncbi.nlm.nih.gov/genbank/fastaformat/
2. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016).
3. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999).
4. Author, F.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010).
5. LNCS Homepage, http://www.springer.com/lncs, last accessed 2016/11/21.

Documentación uso de IA:

- Código para armar bytes a partir de strings:

- Código de función reserve para vectores:

Excelente pregunta, reysito 🧡
te lo explico clarito y con contexto de tu proyecto (donde manejas vectores grandes de bases y secuencias).

🧠 Qué hace `reserve()`

`reserve()` reserva espacio en memoria para un número determinado de elementos en un `vector`, sin cambiar su tamaño actual.

Por ejemplo:

```
cpp                                                    Copiar código

vector<int> v;
v.reserve(1000);
```

👉 Esto no mete 1000 elementos, pero le dice al vector:

> "Oye, vas a tener 1000 elementos, así que alístate y reserva espacio de una vez."

Así, cuando más adelante hagas:

```
cpp                                                    Copiar código

for (int i = 0; i < 1000; i++)
    v.push_back(i);
```

el vector ya tiene todo el espacio preparado, y no necesita volver a pedir memoria al sistema a cada `push_back()`.