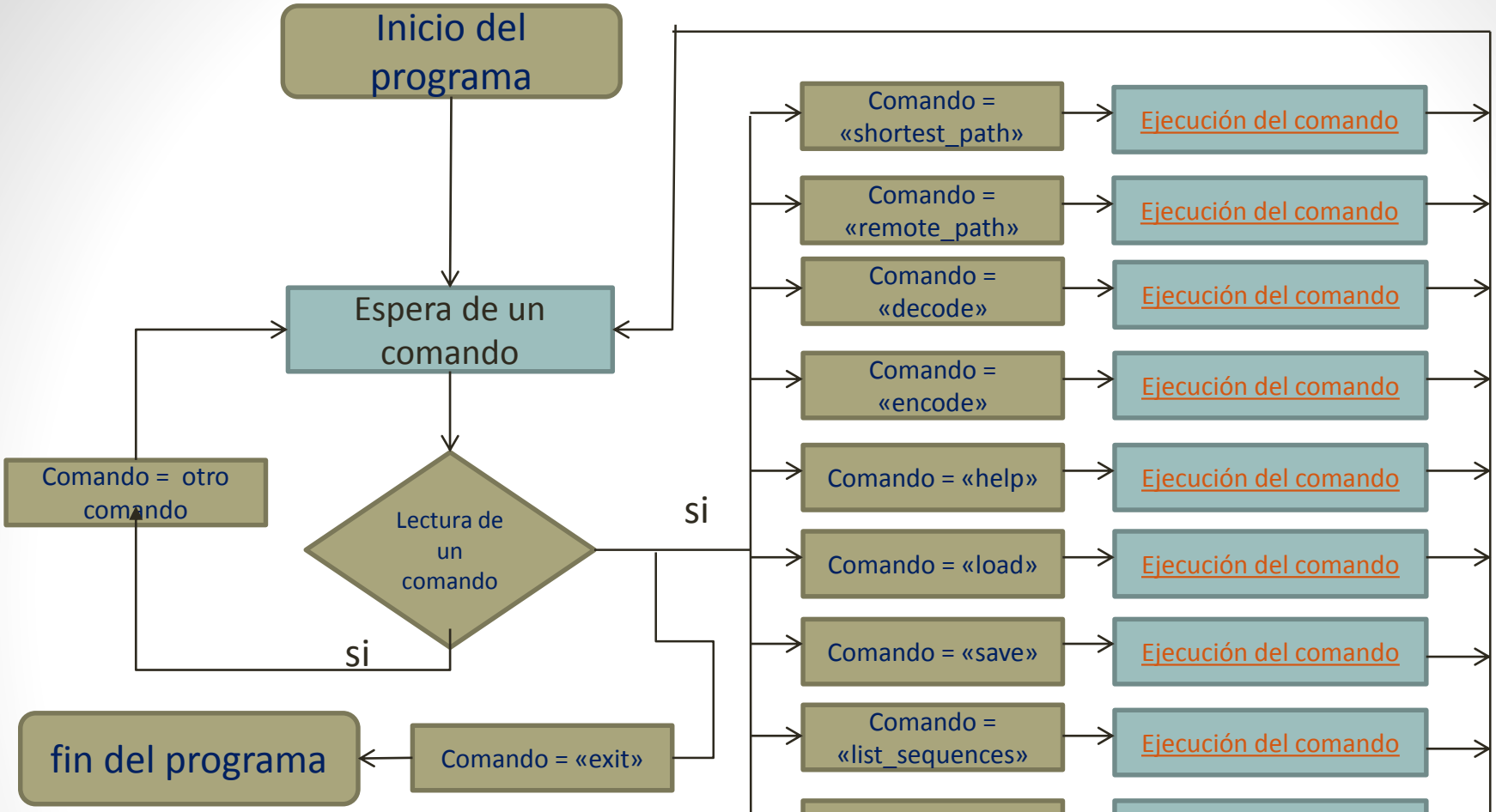


Esquemático del funcionamiento del proyecto (Entrega #3)

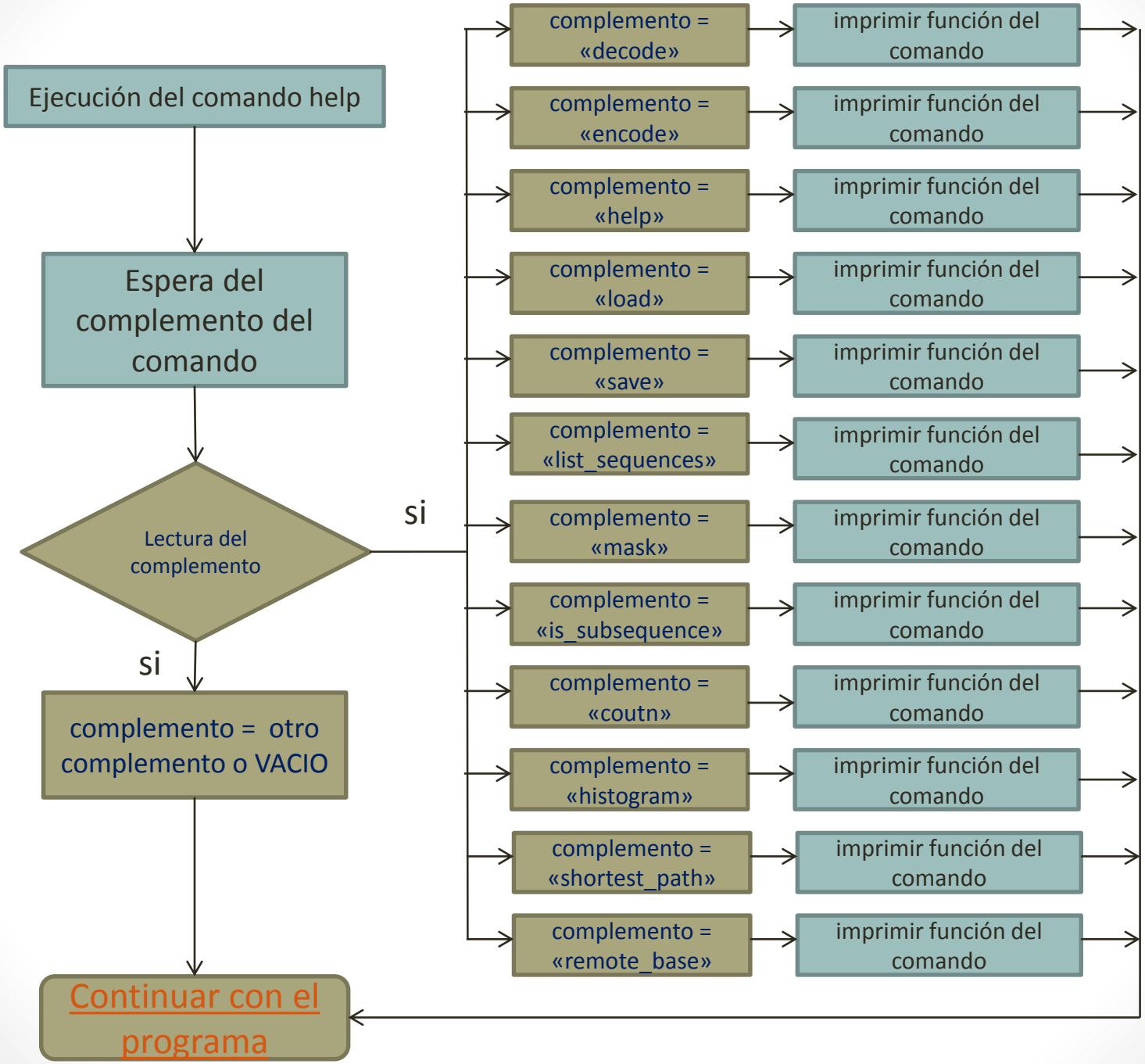
Juan Pablo Rodríguez
Luis Felipe Urdaneta

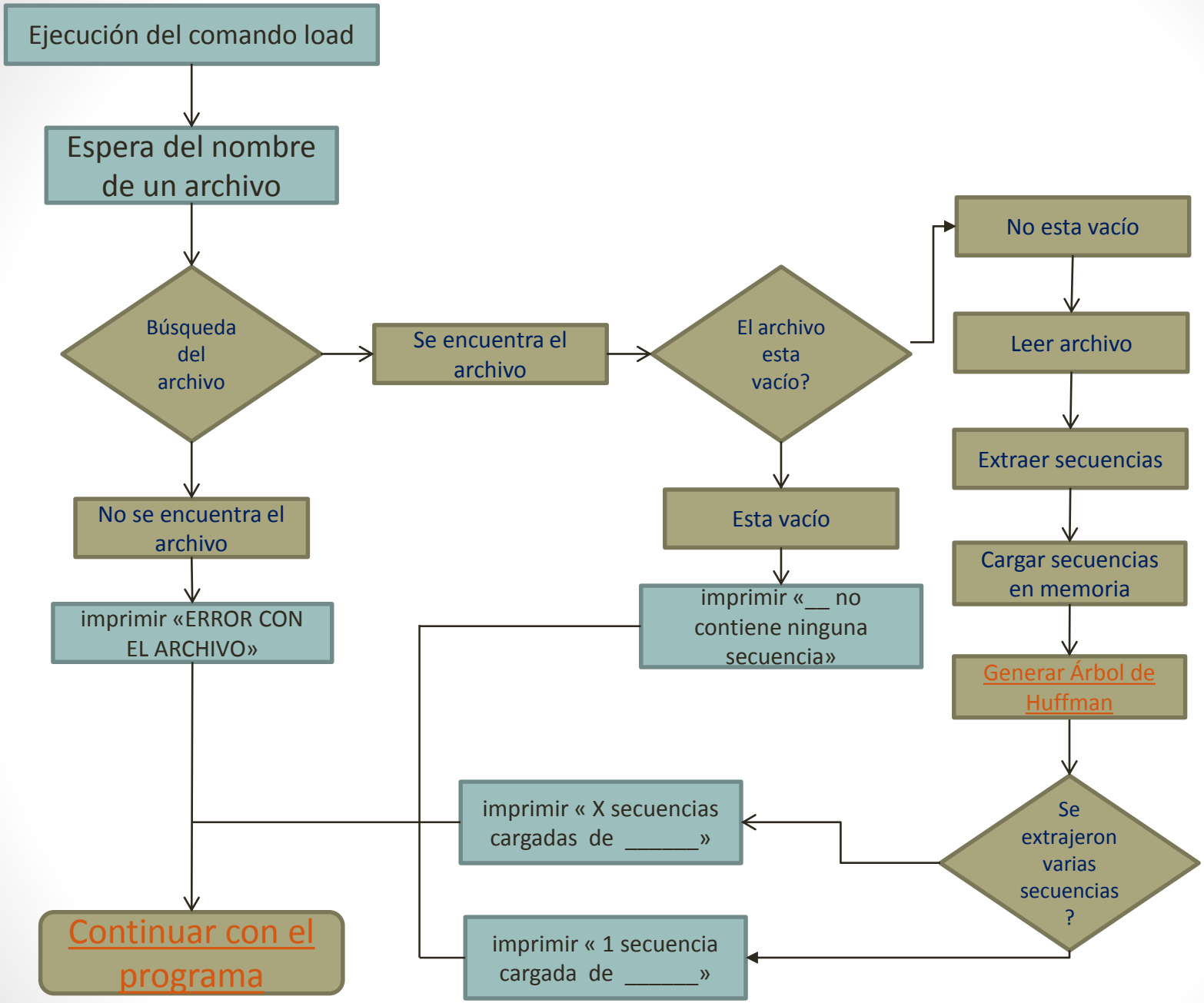
2015

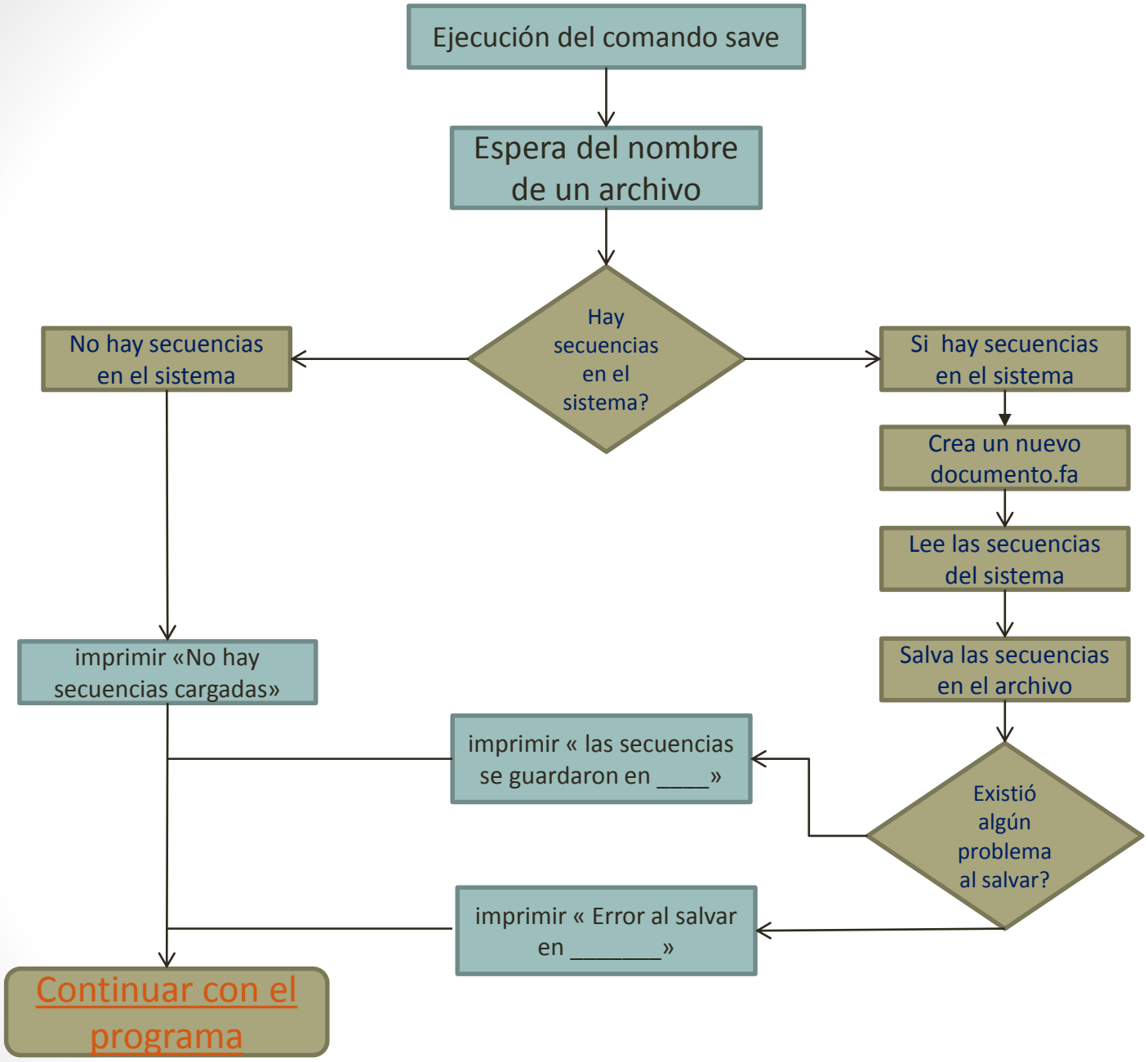
Nota: para navegar utilice los hipervínculos dentro de cada segmento.

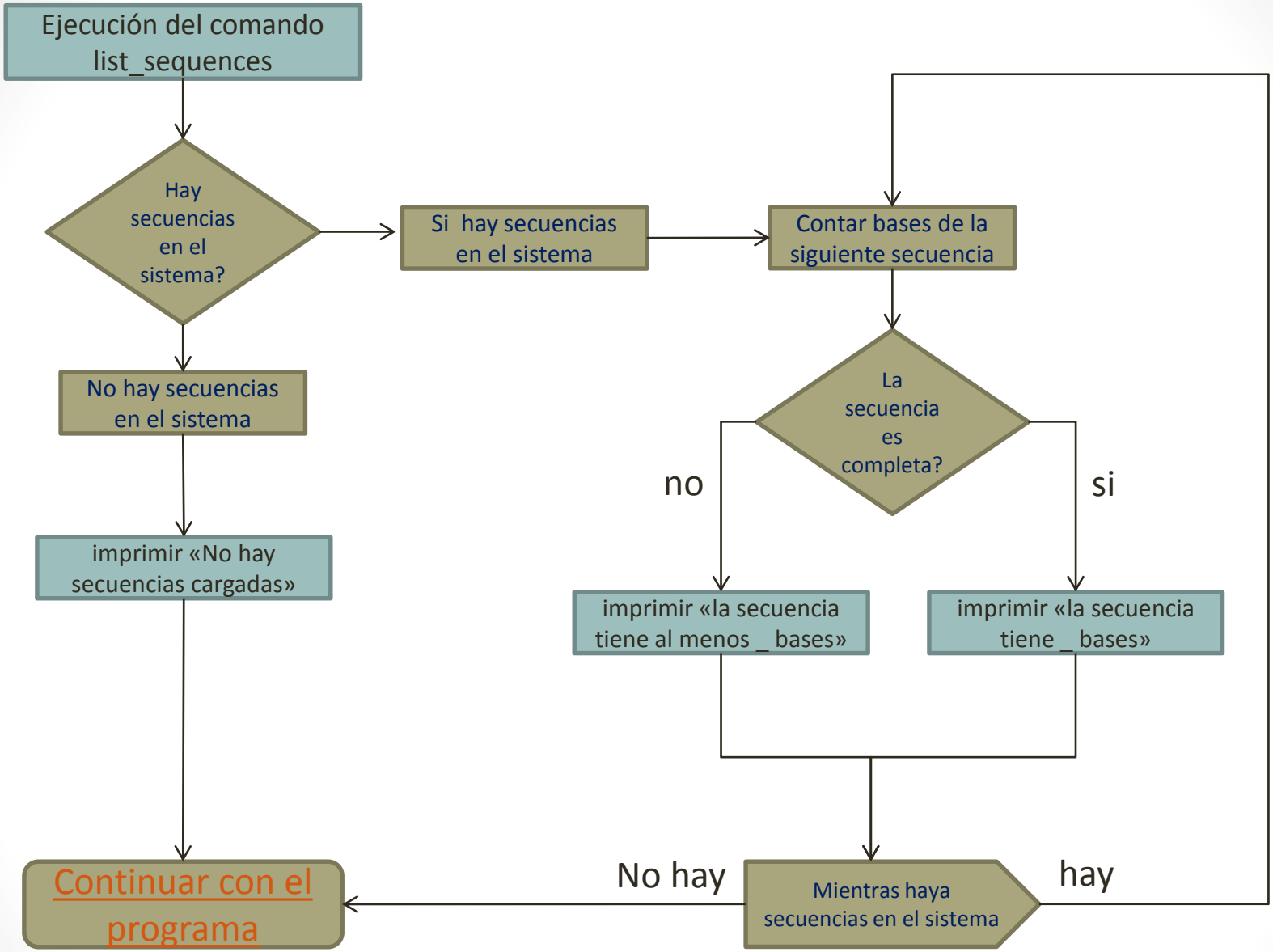
Como se forma el Árbol de Huffman.

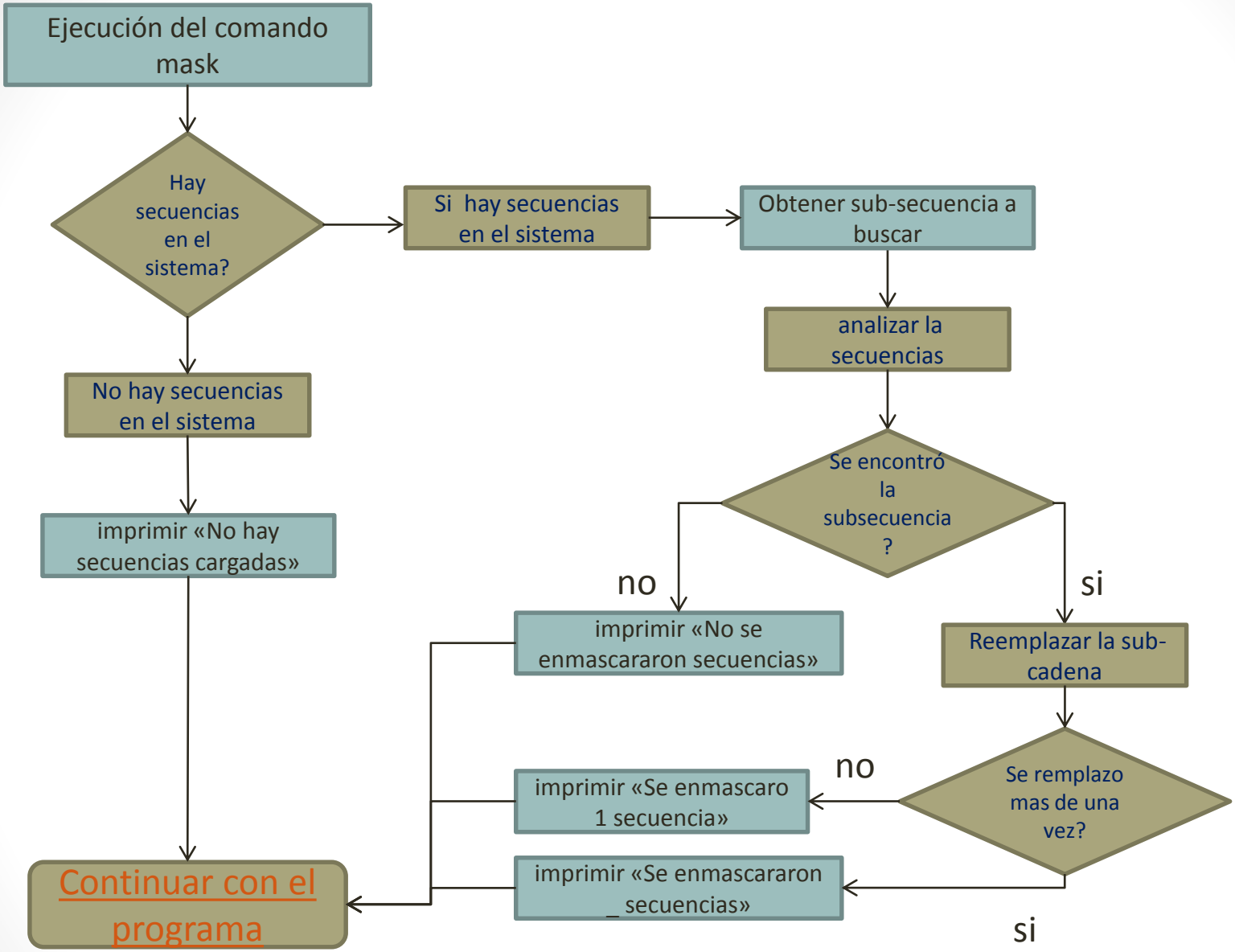
Como esta codificado el archivo binario.

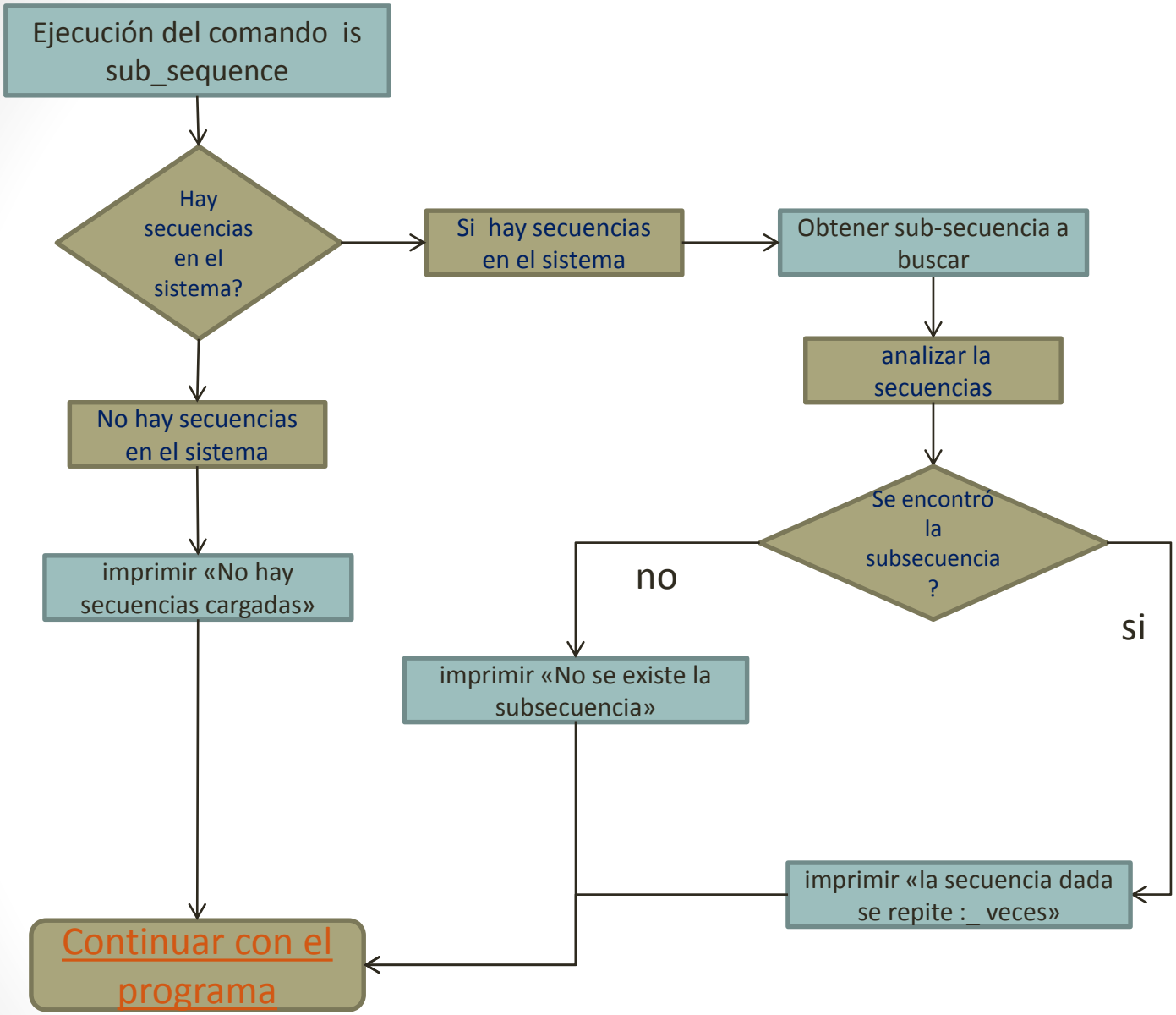

Ejecución del comando help
Espera del complemento del comando
Lectura del complemento
si
si
complemento = otro complemento o VACIO
Continuar con el programa
complemento = «decode»
imprimir función del comando
complemento = «encode»
imprimir función del comando
complemento = «help»
imprimir función del comando
complemento = «load»
imprimir función del comando
complemento = «save»
imprimir función del comando
complemento = «list_sequences»
imprimir función del comando
complemento = «mask»
imprimir función del comando
complemento = «is_subsequence»
imprimir función del comando
complemento = «coutn»
imprimir función del comando
complemento = «histogram»
imprimir función del comando
complemento = «shortest_path»
imprimir función del comando
complemento = «remote_base»
imprimir función del comando

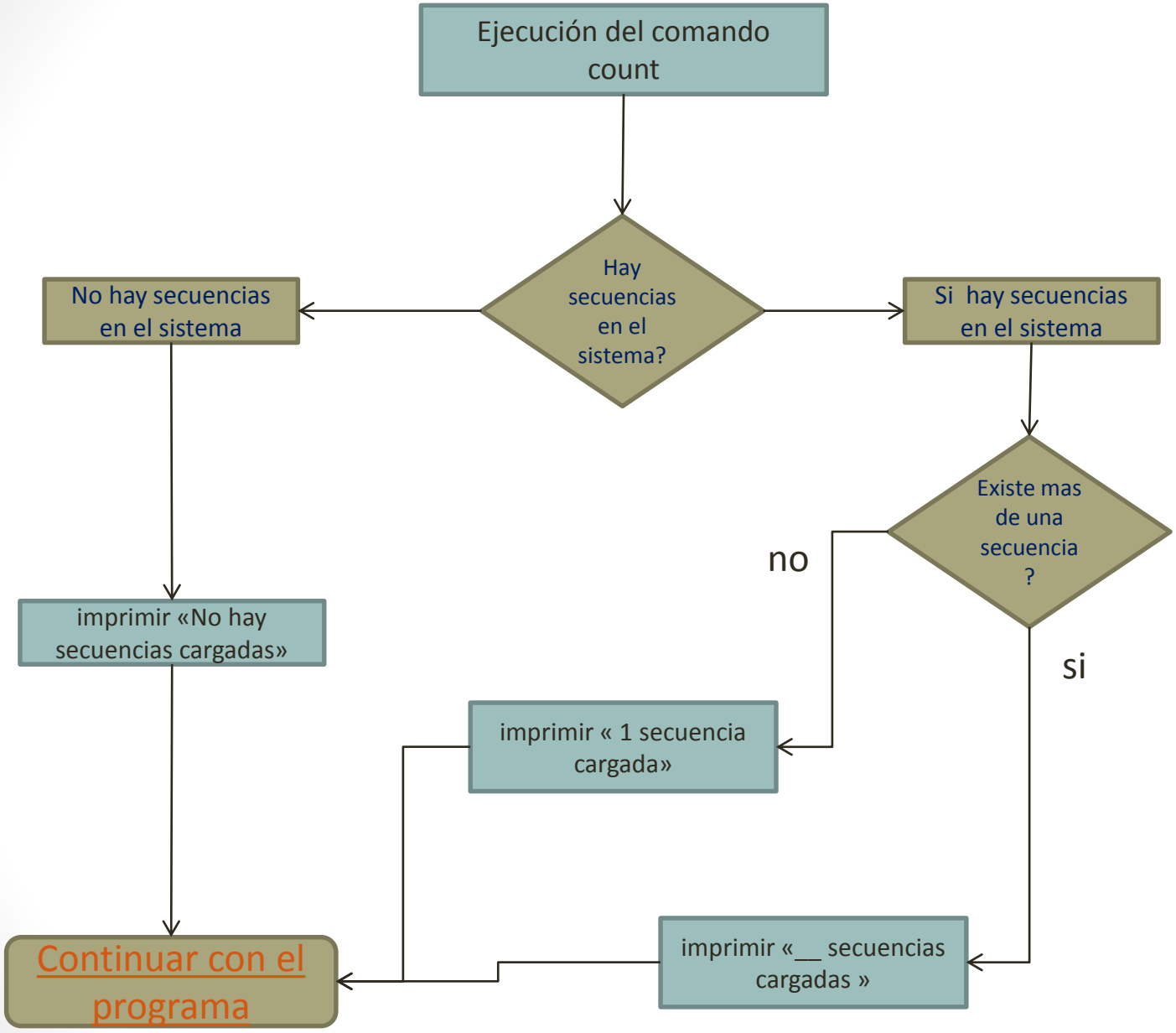

Ejecución del comando load
Espera del nombre de un archivo
Búsqueda del archivo
Se encuentra el archivo
El archivo esta vacío?
No esta vacío
Leer archivo
Extraer secuencias
Cargar secuencias en memoria
Generar Árbol de Huffman
Se extrajeron varias secuencias ?
No se encuentra el archivo
imprimir «ERROR CON EL ARCHIVO»
Esta vacío
imprimir «__ no contiene ninguna secuencia»
imprimir « X secuencias cargadas de _____»
imprimir « 1 secuencia cargada de _____»
Continuar con el programa

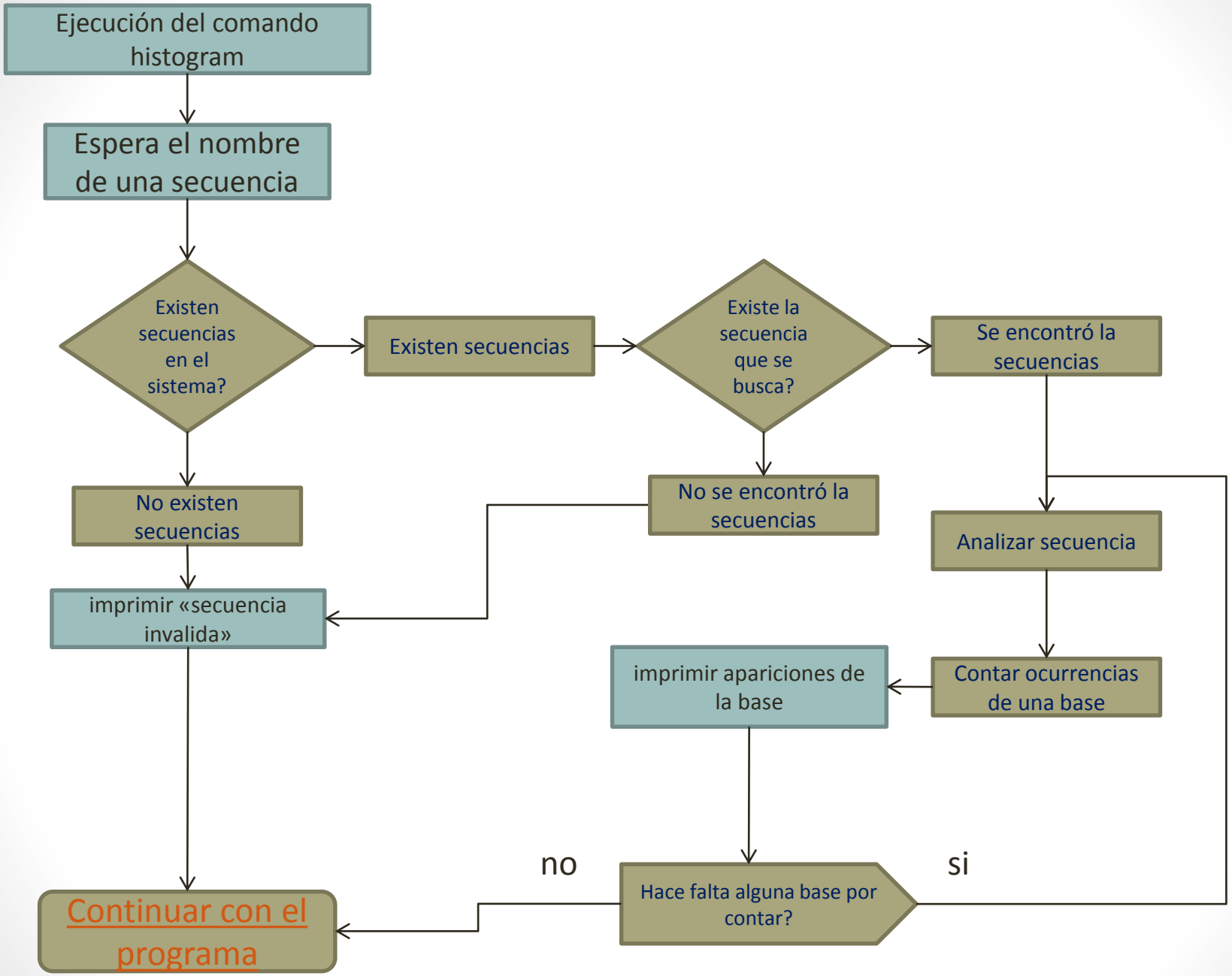

Ejecución del comando save
Espera del nombre de un archivo
Hay secuencias en el sistema?
No hay secuencias en el sistema
Si hay secuencias en el sistema
Crea un nuevo documento.fa
Lee las secuencias del sistema
Salva las secuencias en el archivo
Existió algún problema al salvar?
imprimir « las secuencias se guardaron en ____»
imprimir « Error al salvar en _____»
imprimir «No hay secuencias cargadas»
Continuar con el programa

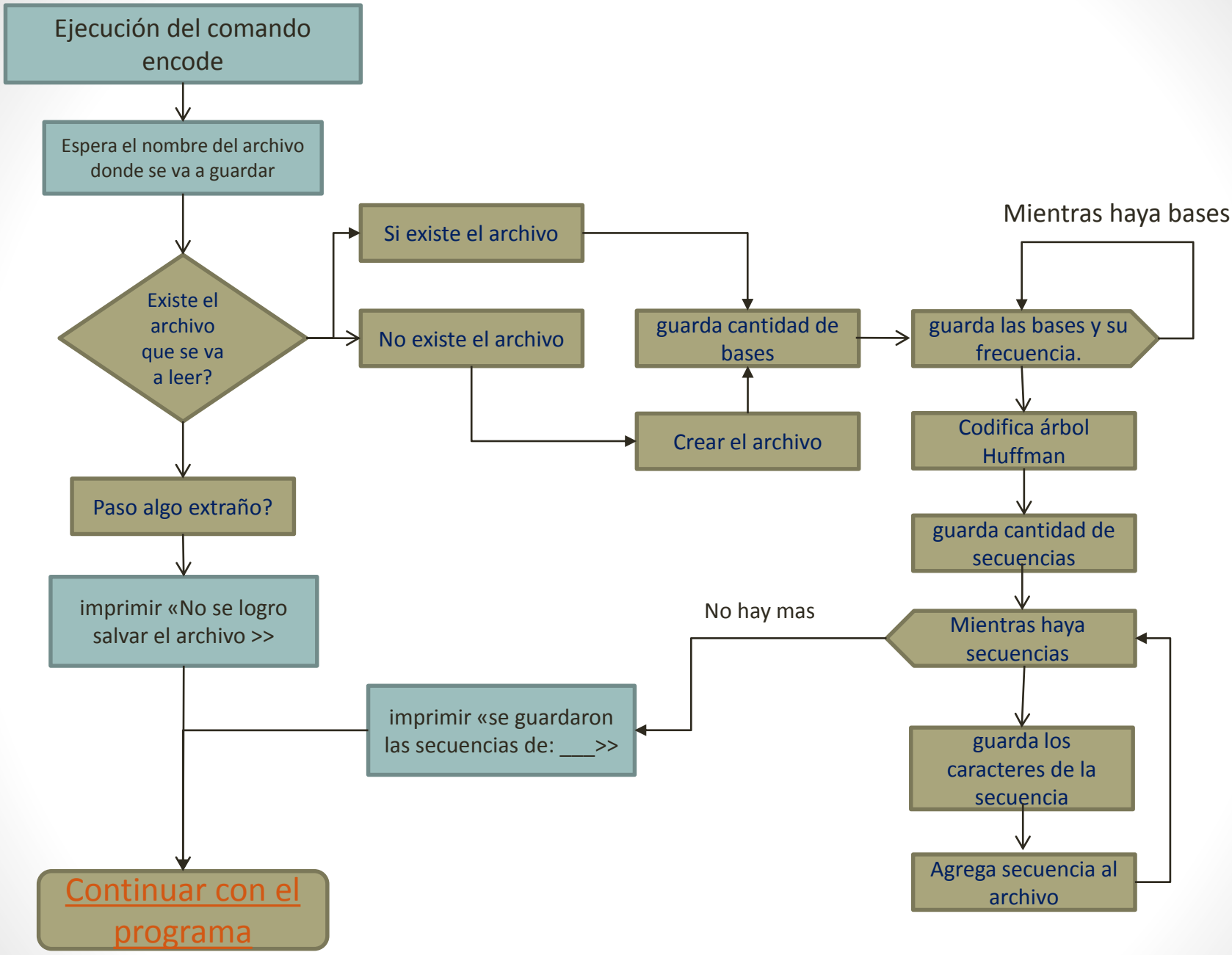

Ejecución del comando list_sequences
Hay secuencias en el sistema?
Si hay secuencias en el sistema
Contar bases de la siguiente secuencia
No hay secuencias en el sistema
La secuencia es completa?
no
si
imprimir «No hay secuencias cargadas»
imprimir «la secuencia tiene al menos _ bases»
imprimir «la secuencia tiene _ bases»
Continuar con el programa
No hay
Mientras haya secuencias en el sistema
hay

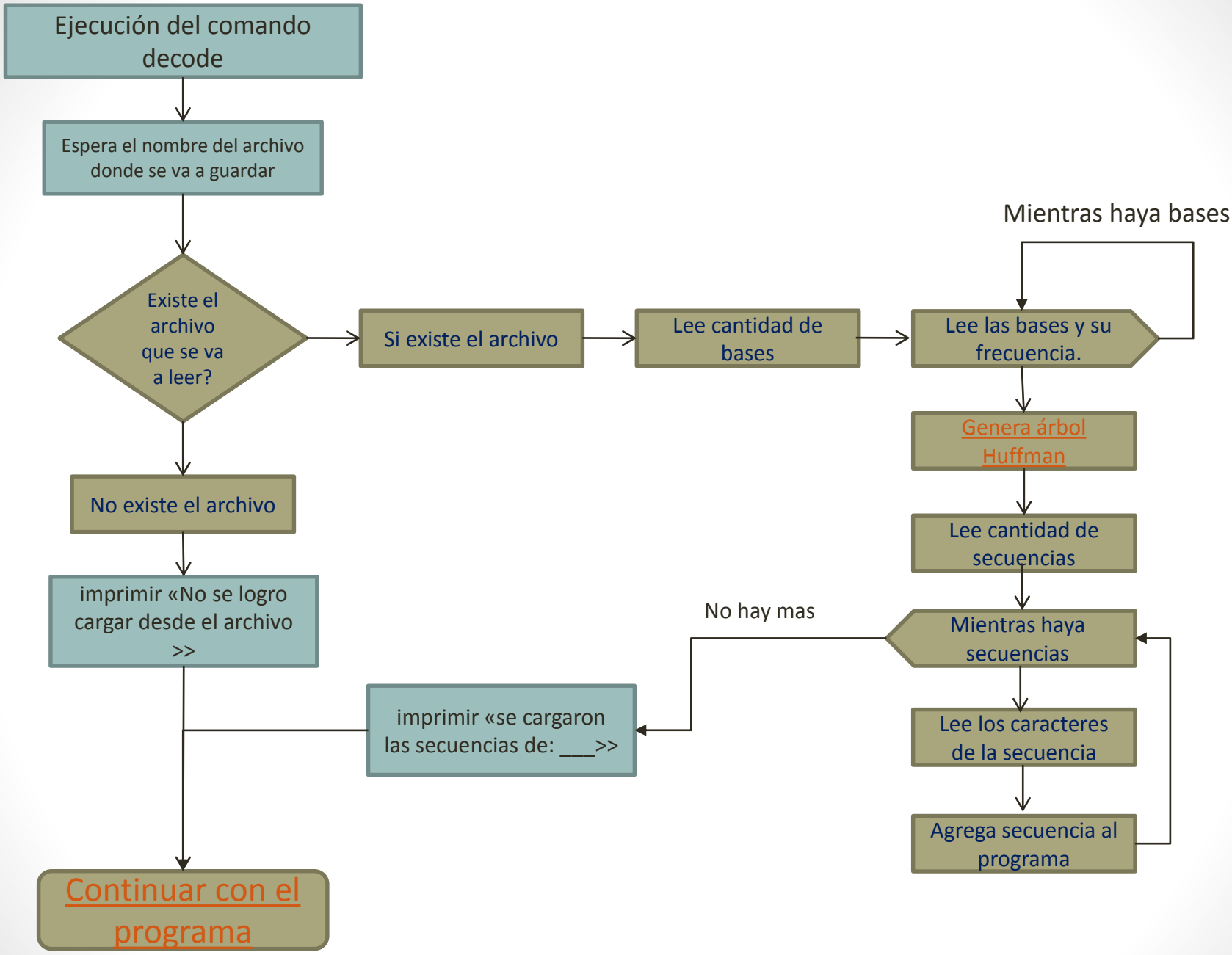

Ejecución del comando mask
Hay secuencias en el sistema?
Si hay secuencias en el sistema
Obtener sub-secuencia a buscar
analizar la secuencias
Se encontró la subsecuencia ?
no
si
No hay secuencias en el sistema
imprimir «No hay secuencias cargadas»
imprimir «No se enmascararon secuencias»
Reemplazar la sub-cadena
Se remplazo mas de una vez?
no
imprimir «Se enmascaro 1 secuencia»
imprimir «Se enmascararon _ secuencias»
si
Continuar con el programa

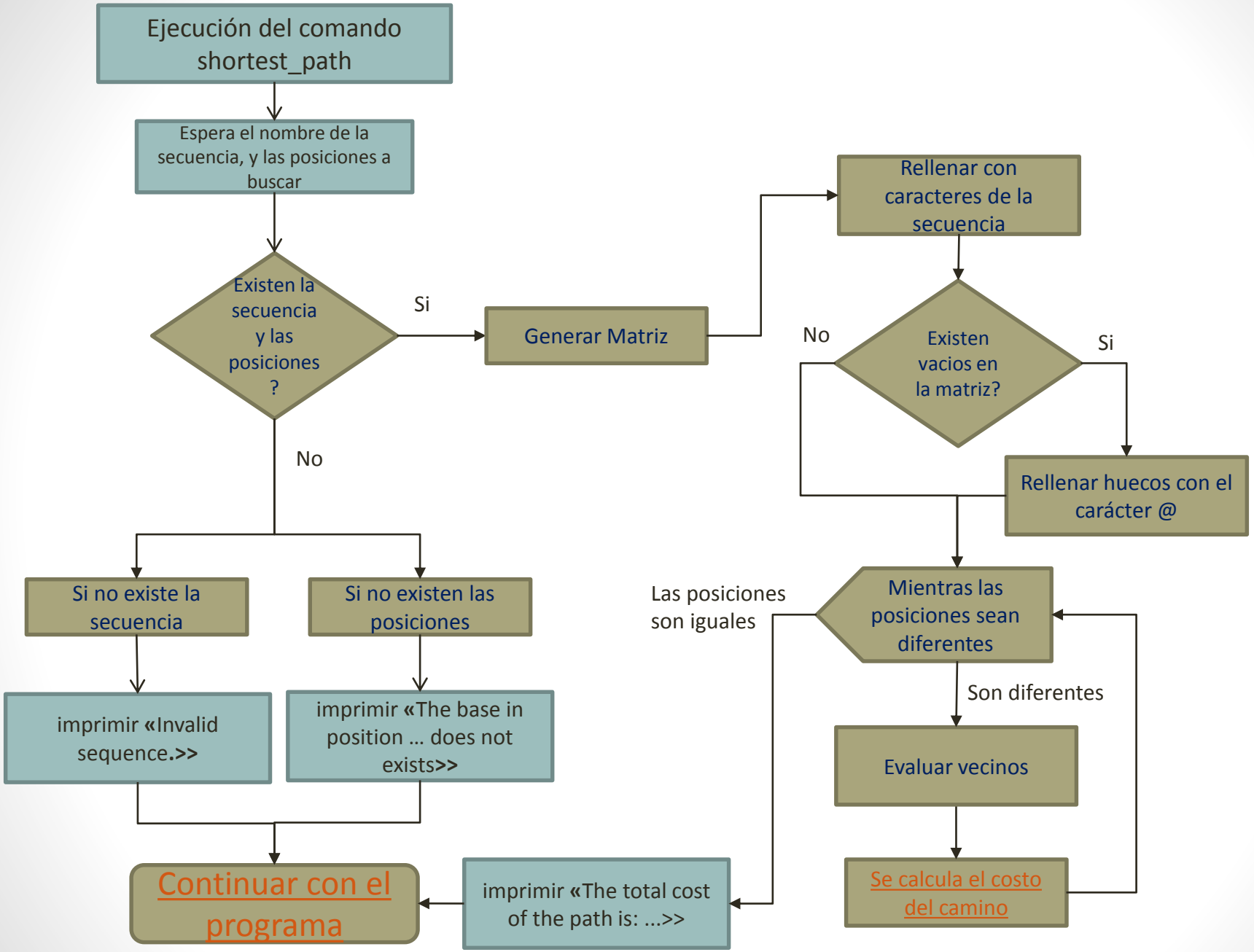

Ejecución del comando is sub_sequence
Hay secuencias en el sistema?
Si hay secuencias en el sistema
Obtener sub-secuencia a buscar
analizar la secuencias
Se encontró la subsecuencia ?
no
si
No hay secuencias en el sistema
imprimir «No hay secuencias cargadas»
imprimir «No se existe la subsecuencia»
imprimir «la secuencia dada se repite :_ veces»
Continuar con el programa

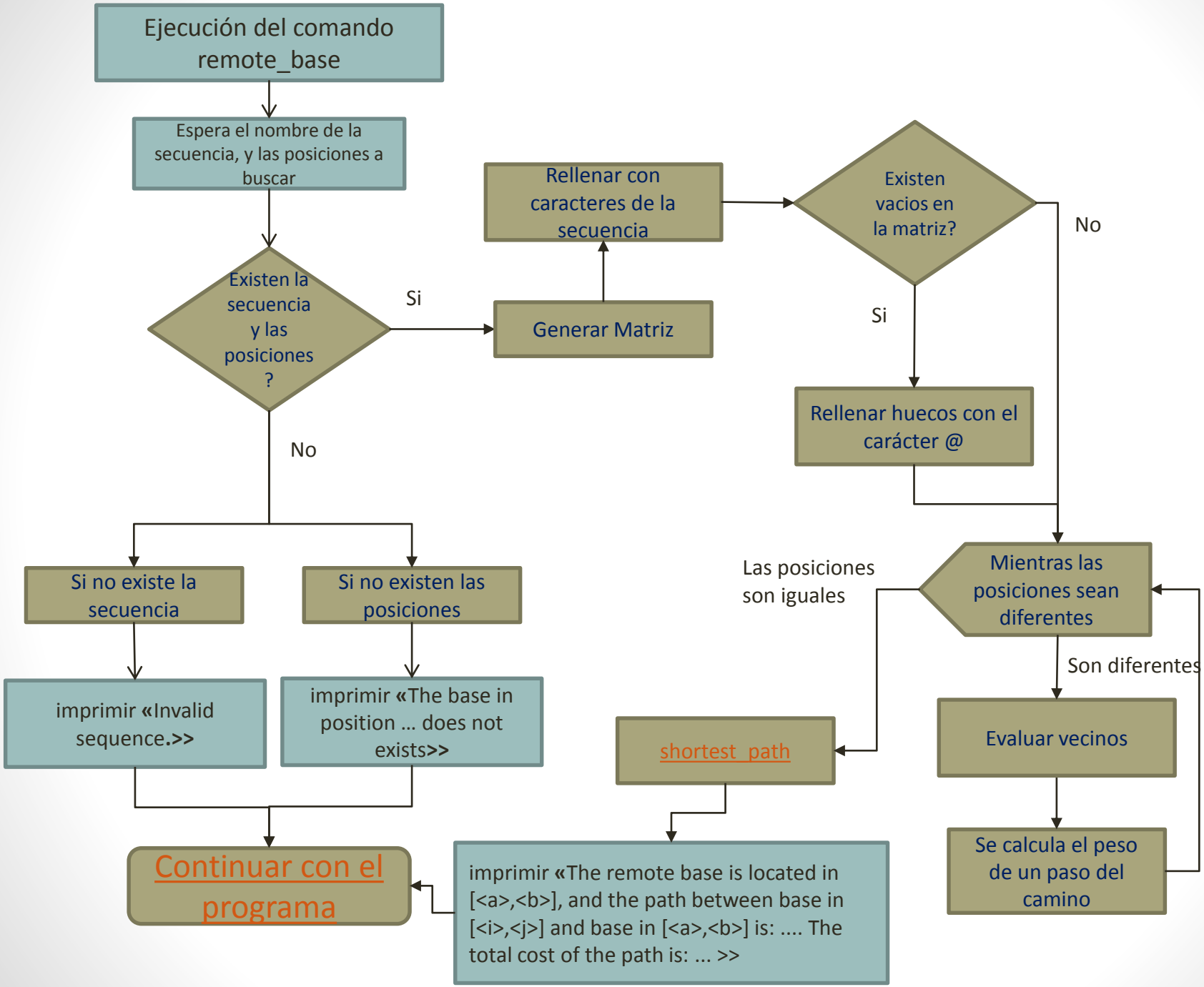

Ejecución del comando count
Hay secuencias en el sistema?
No hay secuencias en el sistema
Si hay secuencias en el sistema
Existe mas de una secuencia ?
no
si
imprimir «No hay secuencias cargadas»
imprimir « 1 secuencia cargada»
imprimir «__ secuencias cargadas »
Continuar con el programa


Ejecución del comando histogram
Espera el nombre de una secuencia
Existen secuencias en el sistema?
Existen secuencias
Existe la secuencia que se busca?
Se encontró la secuencias
No existen secuencias
No se encontró la secuencias
Analizar secuencia
imprimir «secuencia invalida»
imprimir apariciones de la base
Contar ocurrencias de una base
no
si
Hace falta alguna base por contar?
Continuar con el programa


Ejecución del comando encode
Espera el nombre del archivo donde se va a guardar
Existe el archivo que se va a leer?
Si existe el archivo
No existe el archivo
Crear el archivo
guarda cantidad de bases
Mientras haya bases
guarda las bases y su frecuencia.
Codifica árbol Huffman
guarda cantidad de secuencias
Mientras haya secuencias
No hay mas
guarda los caracteres de la secuencia
Agrega secuencia al archivo
Paso algo extraño?
imprimir «No se logro salvar el archivo >>
imprimir «se guardaron las secuencias de: ___>>
Continuar con el programa


Ejecución del comando decode
Espera el nombre del archivo donde se va a guardar
Existe el archivo que se va a leer?
Si existe el archivo
Lee cantidad de bases
Mientras haya bases
Lee las bases y su frecuencia.
Genera árbol Huffman
Lee cantidad de secuencias
Mientras haya secuencias
No hay mas
Lee los caracteres de la secuencia
Agrega secuencia al programa
No existe el archivo
imprimir «No se logro cargar desde el archivo >>
imprimir «se cargaron las secuencias de: ___>>
Continuar con el programa


Ejecución del comando shortest_path
Espera el nombre de la secuencia, y las posiciones a buscar
Existen la secuencia y las posiciones ?
Si
Generar Matriz
Rellenar con caracteres de la secuencia
No
Existen vacios en la matriz?
Si
Rellenar huecos con el carácter @
No
Si no existe la secuencia
Si no existen las posiciones
imprimir «Invalid sequence.>>
imprimir «The base in position … does not exists>>
Las posiciones son iguales
Mientras las posiciones sean diferentes
Son diferentes
Evaluar vecinos
Se calcula el costo del camino
imprimir «The total cost of the path is: ...>>
Continuar con el programa

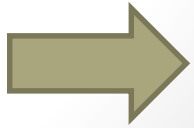

Ejecución del comando remote_base
Espera el nombre de la secuencia, y las posiciones a buscar
Existen la secuencia y las posiciones ?
Si
Generar Matriz
Rellenar con caracteres de la secuencia
Existen vacios en la matriz?
No
Si
Rellenar huecos con el carácter @
No
Si no existe la secuencia
Si no existen las posiciones
imprimir «Invalid sequence.>>
imprimir «The base in position … does not exists>>
Las posiciones son iguales
Mientras las posiciones sean diferentes
Son diferentes
shortest_path
Evaluar vecinos
Se calcula el peso de un paso del camino
Continuar con el programa
imprimir «The remote base is located in [<a>,<b>], and the path between base in [<i>,<j>] and base in [<a>,<b>] is: .... The total cost of the path is: ... >>

¿Como se calcula el costo del camino?

- Despues de que se obtiene la matriz de una secuencia solicitada, y ya se tiene identificada la ruta de un punto de la matriz a otro, se calcula el peso de la conexión entre nodo y nodo usando la siguiente formula:

$$C_{ij-xy} = \frac{1}{1 + |ASCII_{ij} - ASCII_{xy}|}$$

- Donde:
  - C es igual al peso de la conexión
  - I y J representan las coordenadas de un punto
  - X y Y representan las coordenadas de un punto
  - ASCII representa el valor del dato en las coordenadas según el código ASCII.



Ejemplo:

En el caso de esta tabla, queremos llegar de la posición [0,0] (amarillo) a la [1,1] (rojo):

A	C	T	G
C	G	T	C
A	A	C	G

A	C	T	G
C	G	T	C
A	A	C	G

Para este caso, como los vecinos de A son C, y ambos están En contacto con nuestra meta, cualquier camino entre los Dos da igual.

Nuestro camino en este caso seria: A->C->G

Y utilizando nuestra formula el peso de las conexiones Quedaría de la siguiente manera:

$$C_{[0,1]-[0,0]} = \frac{1}{1 + |ASCII_{[0,1]} - ASCII_{[0,0]}|} \longrightarrow C_{[0,1]-[0,0]} = \frac{1}{1 + |67_{[0,1]} - 65_{[0,0]}|} \longrightarrow C_{[0,1]-[0,0]} = \frac{1}{1 + |2|}$$

$$C_{[0,1]-[0,0]} = \frac{1}{3} \longrightarrow$$ Nuestra conexión de A a C tiene un peso de 0.33

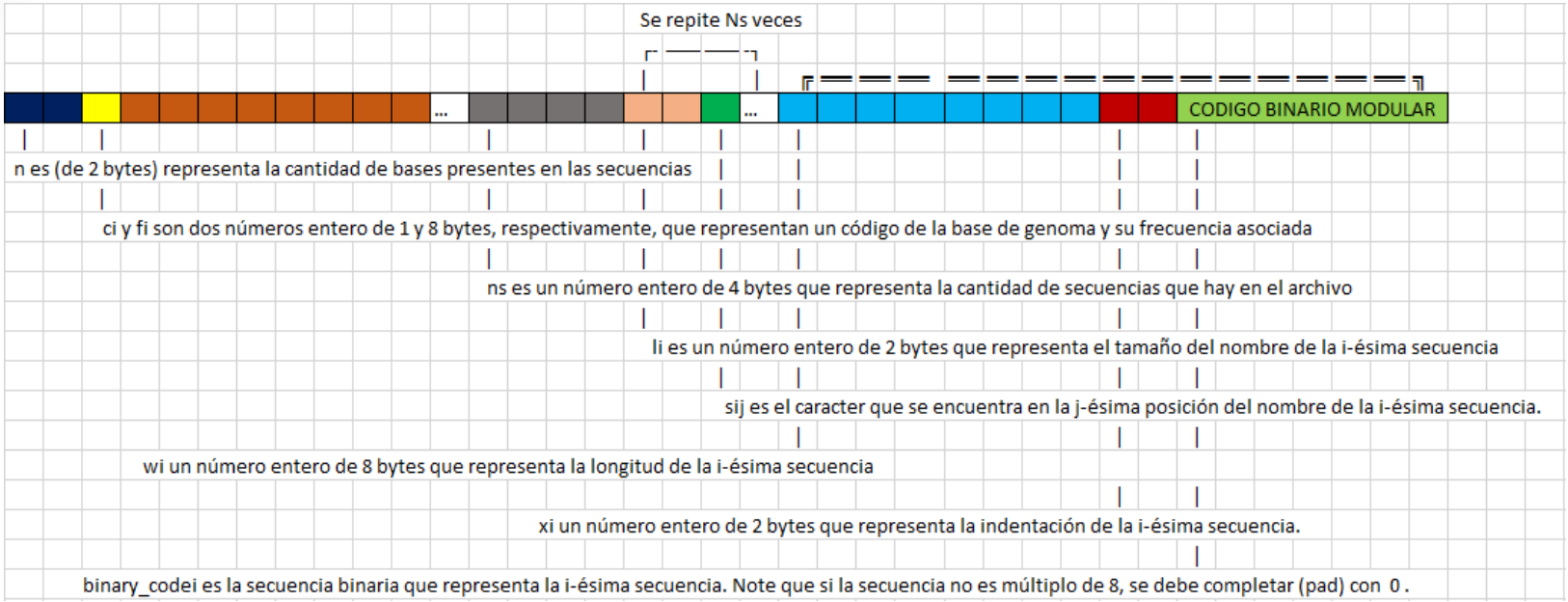
$$C_{[1,1]-[0,1]} = \frac{1}{1 + |ASCII_{[1,1]} - ASCII_{[0,1]}|} \longrightarrow C_{[1,1]-[0,1]} = \frac{1}{1 + |71_{[1,1]} - 67_{[0,1]}|} \longrightarrow C_{[1,1]-[0,1]} = \frac{1}{1 + |4|}$$

$$C_{[0,1]-[0,0]} = \frac{1}{5} \longrightarrow$$ Nuestra conexión de C a G tiene un peso de 0.20

Con lo que el peso de nuestro camino seria 0.33 + 0.20 = ***0,53***

Volver

Como esta codificado el archivo binario:

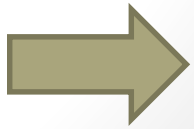


Nota: con código binario modular nos referimos a la secuencia de bits binaray_code, donde un modulo individual representa cada letra dentro de una secuencia

Volver

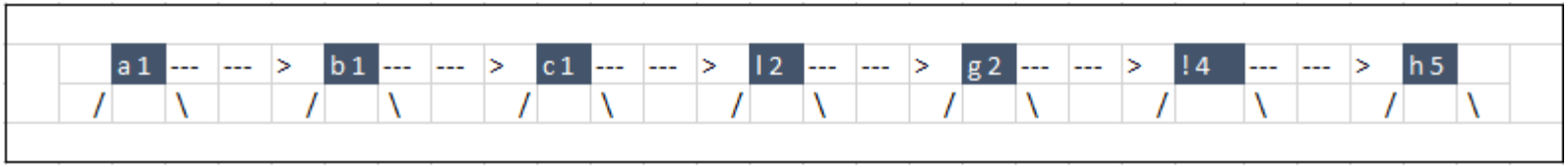
Como se genera el árbol de Huffman:

- Al momento de leer los datos, estos se agregan a un Map de carácter-frecuencia, donde se cuenta la cantidad total de todas las bases presentes en todas las secuencias del archivo que se lee.
- Por medio de una función, cada dato del Map se convierte en un NodoHuffman (remítase al documento adjunto del diseño del proyecto).
- Estos nodos se insertan en una Priority queue, que por medio de un comparator organiza todos los elementos al momento que se ingresa un nuevo nodo.
- A continuación se muestra como se forma el árbol a partir de esta priority queue, después de que todos los datos han sido insertados.

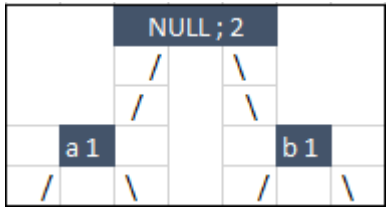


Generar el árbol a partir de la priority queue

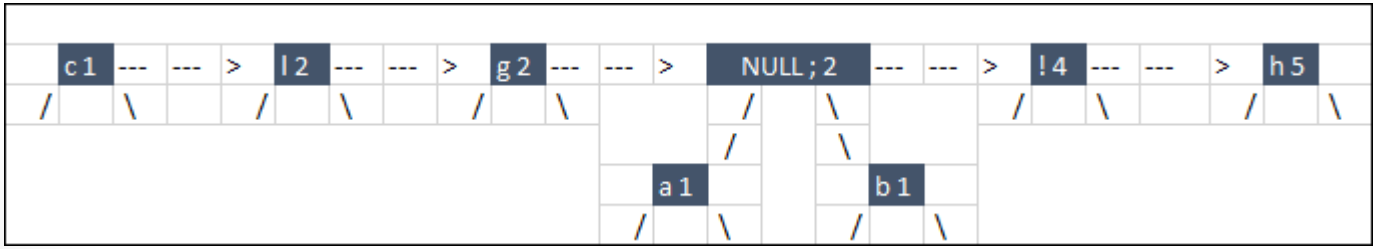
Tenemos nuestra priority queue ya ordenada, el elemento mas a la derecha es El tope de la cola



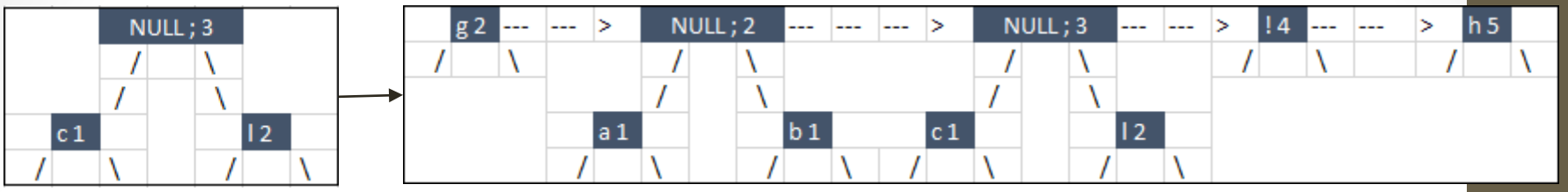
Los 2 primeros elementos de la cola se extraen y se les asigna un padre de carácter que nunca va a ser una base, para la explicación se usara un NULL para mostrar esta idea.
la frecuencia de este nodo padre es la suma de la frecuencia de sus 2 hijos:

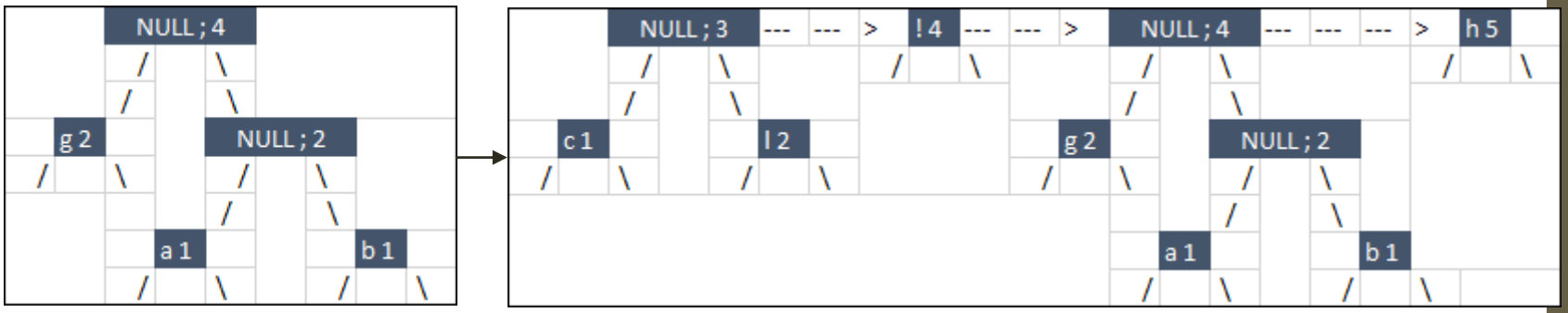


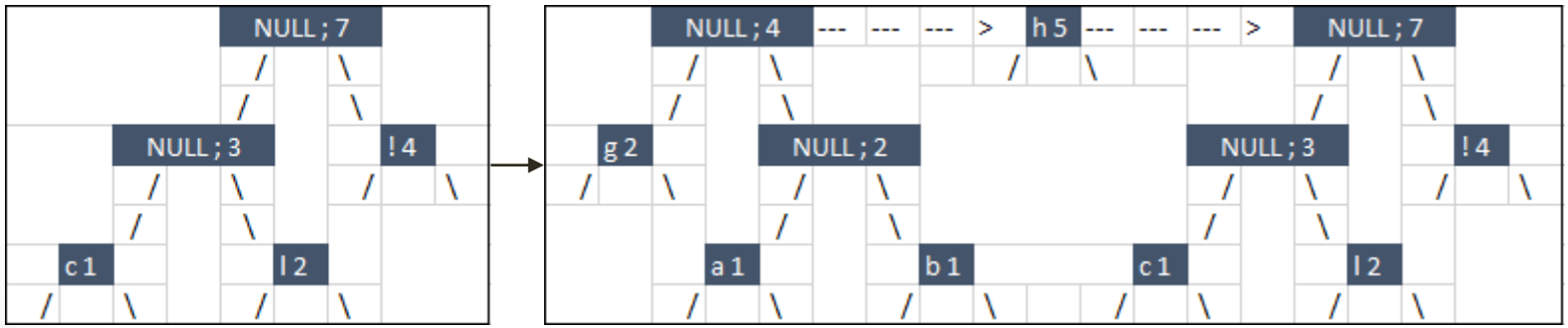
Este nuevo nodo se vuelve meter a la cola, y esta se ordena sola basado en la frecuencia:

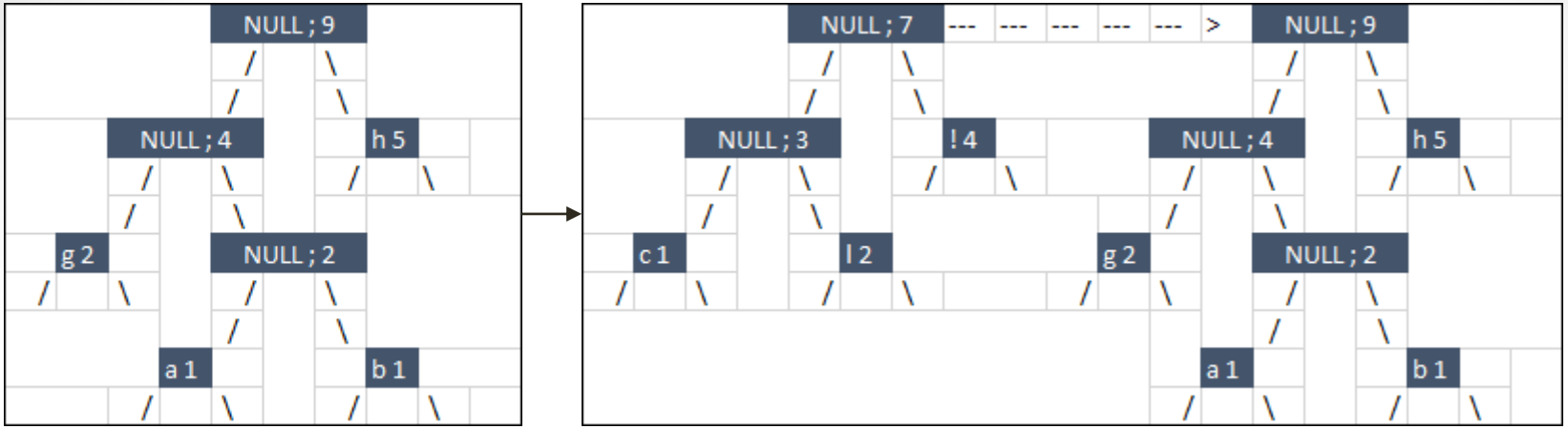


Se repite este proceso N veces:

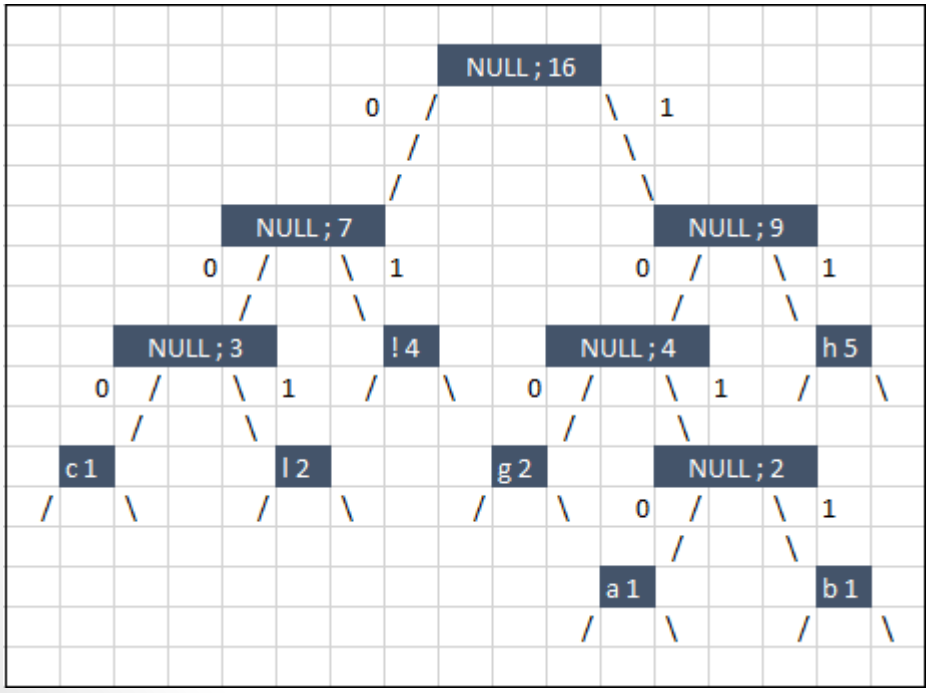








Cuando quedan solo 2 nodos, se crea un padre que se convierte en la raíz de mi ArbolHuffman.



Cada letra tiene un código:

A	1010
B	1011
C	000
!	01
G	100
H	11
L	001

Y un mensaje como

a	l	!	!	c

Quedaría codificado así:

1010|001|0
1|01|000|00

Volver

TAD Cadena

Conjunto mínimo de datos

- tipo, identifica el tipo de cadena que es.

- cadena, es lo que contiene la cadena.

- tam, es el tamaño de la cadena

- ident, es el tamaño de identación.

- complete, dice si esta completa o no la cadena

- bases, tiene las bases con su frecuencia en la cadena

Operaciones del objeto

- Cadena (tipo1,cadena1), Constructor de Cadena, le asigna el tipo y una cadena dada.

- getTipo(), Retorna el tipo de la cadena.

- getCadena(), Retorna la cadena.

- setTipo(tipo), Recibe un tipo por parámetro y cambia el que estaba antes.

- setCadena(cadena), Recibe una cadena por parámetro y cambia el que estaba antes.

- contarBases(), Cuenta las bases que tiene la cadena y retorna la cantidad.

- subCadenas(subCadena,bandera1), Encuentra una subCadena, si bandera1 es verdadero me cuenta las subCadenas que hay, si bandera1 es falso me enmascara esas subcadenas.

TAD CódigoGenetico

Conjunto mínimo de datos

- listaCadenas, vector de tipo cadena, contiene un conjunto de cadenas.

- basesTotales, mapa, contiene las bases de todas las secuencias con su correspondiente frecuencia.

- tree, ArbolHuffman, el arbol asociado a ese código genetico.

Operaciones del objeto

- CodigoGenetico(), Constructor de CodigoGenetico.

- getListaCadenas(), Retorna la lista de cadenas.

- contarSecuencias(), Retorna la cantidad de secuencias que tiene almacenadas.

- cargarDatos(nombreArchivo), Carga las cadenas desde un archivo y las almacena.

- listaSecuencias(), Muestra el tipo de las cadenas y además cuantas bases tiene.

- buscarCadena(descripción), Busca una cadena con su descripción y la retorna.

- mostrarHistograma(descripción), Muestra el histograma de una cadena dada su descripción.

- guardarDatos(nombreArchivo), Guarda los datos que tiene almacenados.

- subCadenas(subCadena,bandera), Retorna la cantidad de subCadenas que se tiene.

- generarArbol(), Genera el correspondiente árbol de HuffMan de las secuencias.

- encode(), Se encarga de codificar (siguiendo el formato) los datos que se almacenan en codigo genetico y posteriormente los guarda en un archivo binario.

- decode(), Se encarga de decodificar los datos que se almacenan en un archivo binario, con un formato establecido.

- shortest_path(), se encarga de encontrar el peso del camino entre dos posiciones

- remote_base(), se encarga de encontrar la base mas lejana y el peso del camino entre esos dos nodos.

Operaciones Adicionales

ingresarBases(), Retorna un vector con las bases ingresadas.

buscarBasePos(letra), Dada una letra busca en el vector la base . Si la encuentra retorna la posición, si no retorna -1.

help load(), Imprime en pantalla la ayuda del comando load.

help count(), Imprime en pantalla la ayuda del comando count.

help ist_sequences(), Imprime en pantalla la ayuda del comando list_sequences

help histogram(), Imprime en pantalla la ayuda del comando histogram.

help is_subsequence(), Imprime en pantalla la ayuda del comando is_subsequence().

help mask(), Imprime en pantalla la ayuda del comando mask.

help save(), Imprime en pantalla la ayuda del comando save.

help encode(), Imprime en pantalla la ayuda del comando encode.

help decode(), Imprime en pantalla la ayuda del comando decode.

help shortest_path(), Imprime en pantalla la ayuda del comando shortest_path.

help remote_path(), Imprime en pantalla la ayuda del comando remote_path.

cargar(código,nombreArchivo), Es la función que se llama cuando se recibe el comando load, recibe CodigoGenetico y el nombre del archivo para cargarlo.

contar(codigo), Es la función que se llama cuando se recibe el comando count, recibe CodigoGenetico para saber de cual código genético debe saber la longitud de sus cadenas.

listaSecuencias(código), Es la función que se llama cuando se recibe el comando list_sequences, recibe CodigoGenetico para mostrar las secuencias que pertenecen a ese CodigoGenetico.

subSecuencia(código,secuencia), Es la función que se llama cuando se recibe el comando is_subsequence, recibe CodigoGenetico y el nombre del tipo de la secuencia. Me muestra en pantalla si esa sub secuencia existe en una secuencia cargada.

mascara(código,secuencia), Es la función que se llama cuando se recibe el comando mask, recibe CodigoGenetico y el nombre del tipo de la secuencia. Muestra en pantalla la cantidad de cadenas que fueron enmascaradas.

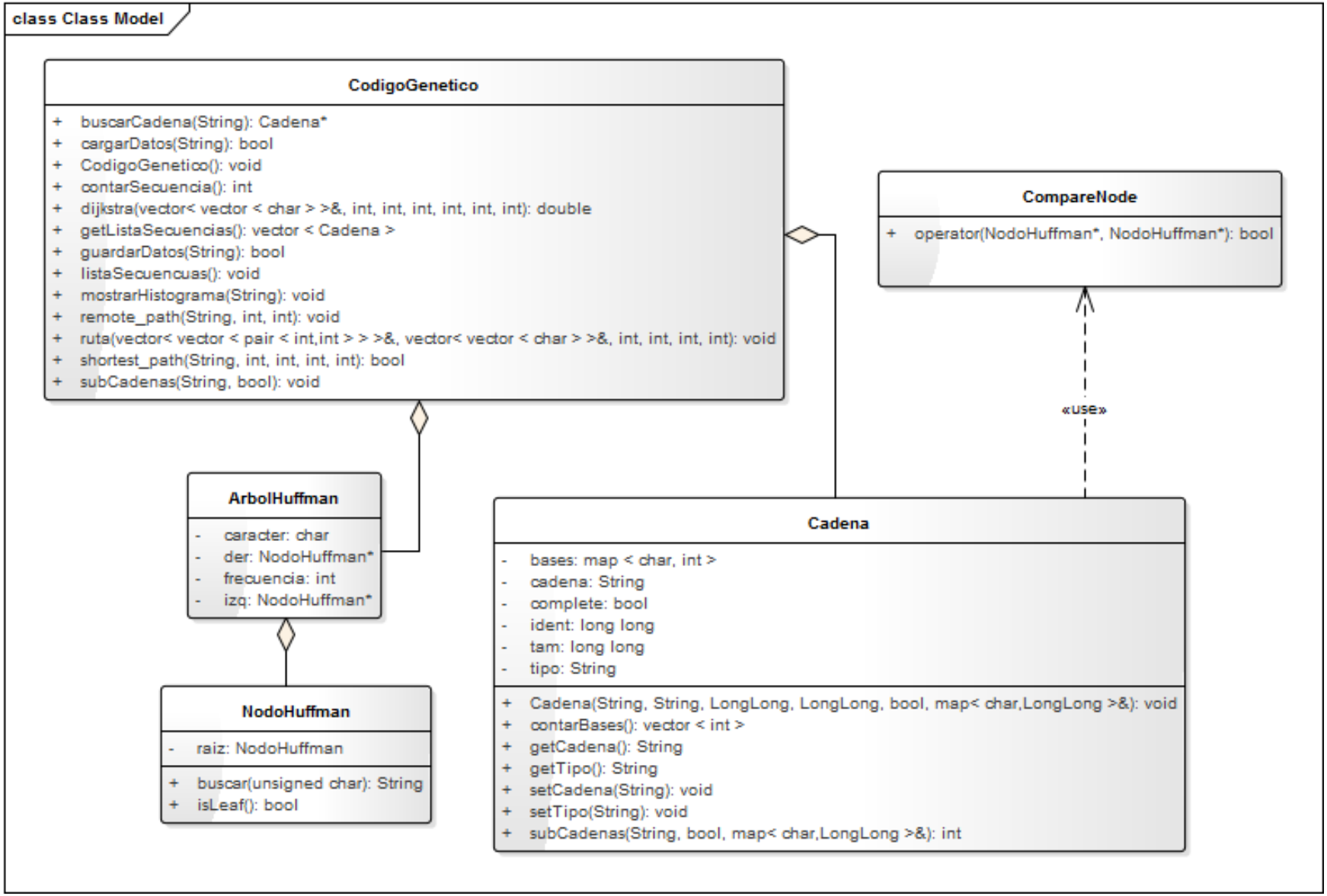
guardar(código,nombreArchivo), Es la función que se llama cuando se recibe el comando save, recibe CodigoGenetico y el nombre del archivo al cual se va a mandar los datos y posteriormente guardarlos.

encode(), Es la función que se llama cuando se recibe el comando encode, recibe CodigoGenetico y el nombre del archivo al cual se va a guardar la codificación.

decode(), Es la función que se llama cuando se recibe el comando decode, recibe CodigoGenetico y el nombre del archivo al cual se va a cargar la codificación.

shortest_path(), Es la función que se llama cuando se recibe el comando shortest_path, recibe CodigoGenetico, el nombre de la secuencia, las coordenadas del punto inicial y las coordenadas del punto final al cual se le va calcular el peso del camino mas corto entre esos 2 puntos.

remote_base(), Es la función que se llama cuando se recibe el comando remote base recibe CodigoGenetico, el nombre de la secuencia y las coordenadas del punto inicial al cual se le va calcular el peso del camino mas corto entre el punto inicial y la base igual mas lejana.


class Class Model
CodigoGenetico
+ buscarCadena(String): Cadena*
+ cargarDatos(String): bool
+ CodigoGenetico(): void
+ contarSecuencia(): int
+ dijkstra(vector< vector < char > >&, int, int, int, int, int, int): double
+ getListaSecuencias(): vector < Cadena >
+ guardarDatos(String): bool
+ listaSecuencuas(): void
+ mostrarHistograma(String): void
+ remote_path(String, int, int): void
+ ruta(vector< vector < pair < int,int > > >&, vector< vector < char > >&, int, int, int, int): void
+ shortest_path(String, int, int, int, int): bool
+ subCadenas(String, bool): void
CompareNode
+ operator(NodoHuffman*, NodoHuffman*): bool
«use»
ArbolHuffman
- caracter: char
- der: NodoHuffman*
- frecuencia: int
- izq: NodoHuffman*
NodoHuffman
- raiz: NodoHuffman
+ buscar(unsigned char): String
+ isLeaf(): bool
Cadena
- bases: map < char, int >
- cadena: String
- complete: bool
- ident: long long
- tam: long long
- tipo: String
+ Cadena(String, String, LongLong, LongLong, bool, map< char,LongLong >&): void
+ contarBases(): vector < int >
+ getCadena(): String
+ getTipo(): String
+ setCadena(String): void
+ setTipo(String): void
+ subCadenas(String, bool, map< char,LongLong >&): int

Diseño Del Proyecto – Change Log – Entrega #3

Comentarios Primera entrega:

Documento de diseño:

- El documento de diseño debería tener todo en un sólo archivo, para facilidad de lectura y que yo pueda ver a organización lógica de la concepción de la solución.

- En el diagrama de relación entre TADs sólo se incluye lo que es un TAD, es decir, que tiene datos Y operaciones asociadas. El main no es un TAD, clases auxiliares que solo provean métodos no son TADs.

Código fuente:

- El comando load no carga los archivos cuando le indico la extensión, con el nombre de archivo solo si lo hace.

- Hay que revisar la demora al listar las secuencias del archivo na_arms.fa, parece estar afectado por la longitud del archivo y de las secuencias.

- En el histograma me debe aparecer el conteo del – también.

- El comando guarda el archivo agregándole la extensión .fa, lo que hace que quede con doble extensión.

Correcciones Realizadas:

Con respecto al documento de diseño, se unieron todos los elementos solicitados (Esquemático, diagrama de clases y Diseño de los TAD), y se corrigieron en el diagrama de relación de los TAD la aparición de elementos innecesarios.

Con respecto al código fuente, se llegó a un acuerdo en el que el enunciado no era lo suficientemente claro con respecto al tema de las extensiones de los archivos, cosa que se corregirá en la segunda entrega. Se agregó el conteo del carácter “-” al histograma y se explicó que la demora al momento de cargar se debe a que se genera todo lo necesario para el programa en la ejecución del comando, para qué sea más rápida la entrega de la información al usuario.

Comentarios Segunda entrega:

Se califica sobre 5.0. Corrección entrega 1:

Documento de diseño:

- No se adjuntó acta de evaluación con los comentarios hechos a la entrega 1 y las soluciones propuestas.

- El documento de diseño debería incluir en un único documento el diseño textual, el diagrama de relación entre TADs y los esquemáticos.

- Además, el documento es incremental, eso quiere decir que debe estar presente también el diseño y esquemáticos de la entrega 1.

- El diagrama de relación entre TADs incluye código fuente (nombres de los tipos de datos). Comando decode: El comando funciona pero tiene dos pequeños problemas:

Código fuente:

- Para el archivo AligSeq85678.fa, la secuencia Frog no se carga completa, lo que se ve reflejado en el correspondiente histograma.

- Si hago dos decode en la misma sesión del programa, parece como si el segundo decode no borrara bien la memoria de lo ya cargado, porque queda con datos adicionales.

- Para la codificación de archivos grandes (na_arms.fa), se demora demasiado en hacer la decodificación, más de 5 minutos, cuando se hace dentro de la misma sesión (es decir, despues de haber cargado ya datos

Correcciones Realizadas:

Se creó y agrego esta acta de cambios del programa al documento de diseño.

Se le agrego al documento de diseño los comentarios y las correcciones de la primera entrega y se eliminaron todas las referencias de código fuente existentes en el diseño de los TADs. Se agregaron cosas al diagrama de clases.

Con respecto al código, se corrigió un bug en el programa que no leía correctamente los datos de la última secuencia cargada en archivos que contenían el carácter “-”. Como se demostró en una sesión en la oficina, los tiempos de codificación, decodificación y los datos que se quedan en el sistema despues de realizar varios decode, se debe probablemente a que en el computador en el que se corrió el programa se habían ejecutado previamente varios proyectos similares que no utilizaban la memoria de la mejor manera posible.