

## Capítulo

### Proceso de Resolución de Problemas

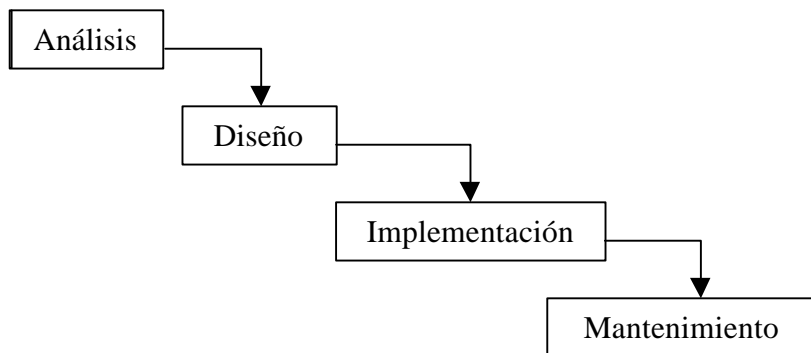
---

Comúnmente nos enfrentamos a problemas o situaciones que deben ser resueltos y pueden ser cosas tan sencillas como decidir la mejor forma de llegar hasta nuestro sitio de trabajo o realizar las compras para la comida, o quizá algunas más complicadas como desarrollar un brazo mecánico que soporte cierto peso y se mueva de cierta manera o idear un plan que nos facilite alguna tarea susceptible de ser sistematizada.

Todas estas situaciones tienen algo en común, necesitan una solución o un conjunto de pasos que permita llegar hasta la meta propuesta. Comúnmente se llama algoritmo a aquello que nos permite pasar de un estado inicial "A" a un estado final "B", justo lo que deseamos. Sin embargo, existe mucho más involucrado en el momento de resolver un problema que tan sólo el conjunto de instrucciones necesarias para ello.

Al plantear la solución de un problema, existen muchos aspectos que deben tenerse en cuenta: El problema mismo, las personas involucradas, las herramientas a disposición, la solución, la presentación de la misma, etc. Todo este proceso ha sido ampliamente estudiado por la ingeniería de software y suele dividirse en 4 etapas bien definidas que también pueden ser aplicadas al problema biológico, objeto de este libro.

El proceso de resolución de un problema es el siguiente:



Se denomina un proceso en cascada porque cada etapa permite el desarrollo de la siguiente, siendo las más importantes las dos primeras ya que incorporan toda la información sobre el problema y guían la solución que se desarrolla mediante el programa. La última etapa, la etapa de mantenimiento es quizá la más costosa, puesto que, infortunadamente, es al utilizar un programa cuándo se descubren la mayoría de las fallas del mismo pero, ¿por qué ocurre esto?, ¿por qué no se previenen las posibles fallas que más adelante se puedan presentar?. Todo es debido a la falta de planeación. Y ya lo hemos dicho, si las primeras etapas fallan o no son bien estructuradas, el resto del proceso no resultará como se espera y un análisis y diseño pobres se notarán durante la utilización de la solución implementada.

Debido a su importancia, en este capítulo presentaremos algunas nociones sobre el análisis y, ante todo, sobre el diseño de un algoritmo. Para este punto se presentarán varias posibilidades que pueden ser utilizadas a gusto de quien este desarrollando el algoritmo, presentando algunas ciertas ventajas sobre las otras.

Pero antes pasar al análisis y diseño de algoritmos, debemos hacer algunas aclaraciones acerca del término, ¿qué es un algoritmo y cómo cabe este concepto dentro de la biocomputación? Empezaremos nuestro tema de resolución de problemas justo por este punto.

## 1. ¿Qué es un algoritmo?

De una manera sencilla y como más adelante será definido podemos decir que un algoritmo es una secuencia ordenada de pasos que persiguen la consecución de un fin, de un objetivo bien claro y específico. Pero podemos también decir que un algoritmo es un procedimiento bien definido, que toma algún valor cualquiera y arroja como resultado otro valor. Pero ante nada un algoritmo no es más que una herramienta que proporciona ayuda para resolver problemas con un mínimo de esfuerzo. El algoritmo debe por tanto estar implementado sobre alguna plataforma que le permita interactuar con el mundo real y ser de utilidad, por lo tanto tiene en sí mismo requerimientos. Estos requerimientos en términos computacionales se pueden definir con nombre propio de una manera general: memoria RAM, capacidad de disco duro, lenguaje de programación, ancho de banda para transmisión de datos, eficiencia (tiempo que toma y recursos que consume) y tiempo de desarrollo son algunos de los parámetros que se deben considerar.

Los algoritmos biocomputacionales son variados, y dependen en muchos casos de los recursos de máquina con que se cuenta. La totalidad de los algoritmos presentados aquí tiene relación directa con el tratamiento de secuencias biológicas. Las secuencias son simples representaciones de estructuras, viene aquí la pregunta: ¿qué es una estructura? Las estructuras biológicas con las que se trata son de dos clases básicamente: proteicas y genómicas, ambas guardan un punto común: están compuestas de elementos, de unidades y relaciones entre ellos. Piense en estas estructuras biológicas de la misma manera en que considera un grafo, o una estructura tridimensional, una molécula cualquiera por ejemplo, tiene elementos que están entre sí interconectados, eliminemos los factores debidos a

la información que entre los elementos se tiene, y que nos queda... una representación de átomos y conexiones entre ellos. Tomemos en consideración otro caso que es característico para la biocomputación: estructuras de aminoácidos, péptidos o proteínas. En este caso no tomamos en cuenta la configuración espacial que la molécula pueda tener, nos vamos a concentrar simplemente en la representación más básica y no por ello la menos complicada, una proteína esta compuesta por aminoácidos, que para propósitos prácticos son representados por letras, por lo tanto una estructura proteica la podemos representar como una cadena de letras que a su vez representan aminoácidos. Hasta aquí tenemos entonces estructuras tridimensionales y estructuras de texto para representar molécula... cuando considerar una y cuando considerar la otra? Todo depende de la clase de análisis que deseemos llevar a cabo. Búsquedas sobre bases de datos pueden ser y de hecho son mejor entendidas usando estructuras de tipo texto como representación de la o las moléculas en estudio.

Consideremos ahora un caso que en biocomputación es típico, el encontrar un segmento en una secuencia larga (siempre las secuencias que se manejan en biocomputación son bastante largas). Una secuencia proteica entendida como una secuencia de caracteres esta formada por los 20 aminoácidos principales, en el caso del DNA solo tendríamos 4 caracteres, ATCG.

Es muy frecuente el que se requiera encontrar un patrón determinado dentro de una secuencia dada, supongamos

1st base (5' )	2nd base				3rd base (3' )
	U	C	A	G	
U	Phe	Ser	Tyr	Cys	U
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	STP	STP	A
	Leu	Ser	STP	Trp	G
C	Leu	Pro	His	Arg	U
	Leu	Pro	His	Arg	C
	Leu	Pro	Gln	Arg	A
	Leu	Pro	Gln	Arg	G
A	Ile	Thr	Asn	Ser	U
	Ile	Thr	Asn	Ser	C
	Ile	Thr	Lys	Arg	A
	MET	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	U
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val	Ala	Glu	Gly	G

que para un ejemplo la secuencia tiene un largo de 10000 residuos, podemos ahora pensar en varios algoritmos que podrían resolvernros el problema de encontrar un segmento de solamente 8 residuos dentro de nuestra secuencia madre de 10000. Denominemos para efectos prácticos como L y W a cada una de las secuencias a trabajar.

Considere el algoritmo siguiente para solucionar nuestro ejemplo específico:

Identifique el primer residuo en ambas secuencias  
 Compare en ambas secuencias el residuo en el que se encuentre y decida

- 2.1 CASO #1: ambos residuos son idénticos
  - 2.1.1 Identifique el segundo residuo i+1
  - 2.1.2 Compare, si es idéntico continúe comparando

$i+1$  hasta que llegue al último residuo en  $W$

2.1.3 Tabule el número de identidades siempre incrementándolo cuando se localice una nueva.

2.2 CASO #2: ambos residuos son diferentes

2.2.1 Siga al siguiente residuo y vaya al paso 2.

Consideremos otro caso que atañe a la biología molecular computacional, el cálculo de propiedades numéricas sobre secuencias. Pesemos por ejemplo que deseamos calcular el porcentaje de aminoácidos hidrofílicos e hidrofóbicos que nuestra secuencia en cuestión tiene, llevaríamos a cabo para esto sobre una secuencia de longitud  $l$  un número  $l$  de operaciones. Otro caso es el cálculo de regiones ricas en bases purínicas y regiones ricas en bases pirimídicas, aquí no trabajamos ya con aminoácidos si no lo hacemos con ATCG, y para una mejor interpretación de resultados no usaremos una tabla si no que haremos uso de una gráfica.

Un punto básico y crucial en un análisis de secuencias es el que a primera vista parecería más inmediato, comparación, búsqueda de similaridades. El concepto básico de similaridad tiene profundas raíces en el lenguaje mismo. Para el caso biológico existen dos puntos claves, uno de ellos es el concepto de funciones de distancia, el otro es el concepto de sitios motifs en proteínas. Secuencias cercanas pueden ser comparadas en términos de distancias (similaridad global) mientras que aquellas que compartan solamente unos pocos dominios pueden ser mejor comparadas en términos de motifs.

Más adelante dentro del desarrollo de este capítulo se explicará con más detalle la búsqueda de similaridades entre secuencias, sin embargo el concepto debe quedar claro desde el principio, dos secuencias pueden ser muy similares en términos químicos de función, pero esto no

implica que las dos mismas secuencias compartan altos puntajes de similitud cuando se usan para su comparación estructural, sin embargo cabe aclarar que aunque no es una regla general si sucede con gran frecuencia que una alta similitud en cuanto a función y estructura tridimensional se refiere desembocando en una similitud igualmente alta en comparaciones de estructuras textuales.

Para entender mejor el concepto de similitud tomemos un ejemplo trivial, pensemos que tenemos diferentes clases de computadoras, y queremos evaluar su grado de diferencia o similitud. Para empezar debemos primero que todo elegir una o varias características que nos permitan evaluar el grado de similitud o diferencia (capacidad en RAM, memoria CACHE, capacidad de disco duro, velocidad de procesador, tipo de procesador... etc.....) una vez hecha esta selección procederemos a una natural clasificación, agrupación dependiendo de la característica elegida, así podremos decir que todos los computadores que tengan 64 Megas en RAM pertenecen a una misma clasificación. Para el caso de los computadores el árbol que empezamos a armar apenas si empieza porque las variables a evaluar son muchas, un computador de x memoria en RAM no se puede inmediatamente clasificar como mejor al compararlo contra otro de menor capacidad pero con mayor velocidad de procesamiento de datos..... la evaluación del grado de discrepancia entre las unidades a comparar se torna entonces algo más compleja pero de cualquier forma manejable.

Al comparar secuencias proteicas encontramos dos problemas que son inherentes a las mismas secuencias, existen grandes similitudes entre secuencias y grandes discrepancias. Miembros de la misma familia de proteínas comparten básicamente los mismos componentes,

miembros de una misma superfamilia de proteínas comparten dominios comunes. Si se comparan proteínas homologas, o sea aquellas que están evolutivamente relacionadas podemos usar una medida numérica del grado de relación que unas con otras guardan, esto con el fin de clasificarlas, de catalogarlas. Sin embargo pensemos por un momento que ya no trabajamos con sistemas relacionados, si no que lo hacemos con sistemas altamente no relacionados..... en nuestro caso proteínas que poseen diferentes tipos de dominios..... entonces debemos ser en extremo cuidadosos con los parámetros de comparación, estos deberán ser altamente confiables porque de lo contrario se corre el riesgo de encontrar similitudes entre proteínas que no poseen elementos comunes.

Un concepto muy usado en textos y escritos de bioinformática es el de distancias. En este punto la idea que subyace tras distancias es la misma que plantea la geometría euclidiana. Lo que se calcula numéricamente es la distancia que existe entre un número determinado de puntos en un espacio. Para el caso de trabajar con estructuras textuales como representaciones de secuencias biológicas podemos construir funciones de distancias entre los caracteres que compongan la secuencia. Hagamos algunas consideraciones previas antes de continuar.

## 2. Análisis y Modelaje de Problemas

El análisis del problema es la etapa en la cual se define el problema y el alcance de la solución. Para esto se debe tener en cuenta no sólo la situación si no también su entorno, las personas que van a utilizar esta solución, el ambiente en el cual se puede desarrollar, etc. Es muy importante tener en cuenta estos aspectos desde el principio y para esto se utilizan diversos mecanismos que

son llamados de "elicitación de requerimientos".

Aunque esta etapa, tal cual se define y se describe más adelante, tal vez no sea muy aplicable al desarrollado de algoritmos biológicos tratados en forma aislada, si es muy importante en el momento de desarrollar proyectos más amplios o un sistema integrado que reúna varios de los algoritmos descritos en el presente libro. Cuando se desea desarrollar un sistema completo, la etapa de análisis, teniendo en cuenta todos los factores descritos, es indispensable.

En cuanto al modelaje, se puede considerar como el resultado formal del análisis aunque no es lo único que se obtiene y puede ser utilizada para el análisis de algoritmos desarrollados en forma aislada, dónde lo más importante es la solución y no el entorno. En el modelaje, nos referimos estrictamente al problema, sin tener en cuenta los otros aspectos ya mencionados; aspectos que deben ser consignado en un documento de análisis con un adjunto que especifique el modelaje de la situación tratada.

A continuación presentaremos algunos mecanismos de análisis y las nociones básicas en cuanto al modelaje de problemas.

## 2.1 Etapa de Análisis

En la etapa de análisis se reúnen todos los datos referentes al problema, el entorno y las personas para quienes se desarrollará la solución. Para esto existen diversas técnicas que serán brevemente mencionadas a continuación; si desea mayor información, puede consultar la bibliografía que se cita al final del capítulo.

### 2.1.1 Entrevistas

Las entrevistas son reuniones que se realizan entre el analista y las personas involucradas, pueden realizarse en forma colectiva o personal y los resultados de cada entrevista deben ser documentados para facilitar la integración al final.

### 2.1.2 Encuestas

Este método consiste en realizar encuestas a los implicados en la situación problemática. Para el desarrollo de las encuestas se tienen charlas anteriores con personas clave, así se evita considerar aspectos que están fuera de contexto. Otro factor muy importante, es no limitar las respuestas ni conducir las mediante el enunciado de las preguntas.

### 2.1.3 Tormenta de Ideas

La tormenta de ideas es un método grupal. Consiste en reunir un grupo de personas a hablar sobre un tema y dejar que expresen todo lo que deseen. Se debe tener cuidado e impedir que alguien monopolice o guíe a los demás y evitar que algunos se queden sin participar.

Estas son sólo una pequeña muestra de las técnicas de elicitación de requerimientos, existen muchas más pero todas persiguen el mismo objetivo: Conocer el problema y su entorno.

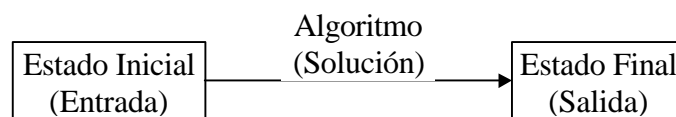
Pensando en los algoritmos biológicos, estos mecanismos tal vez no son muy adecuados ya que no se está tratando una situación problemática sino un problema específico y muy bien limitado; sin embargo, no está de más un conocimiento global en esta área.

Veremos ahora un análisis que se adecua más al tema que nos interesa, los algoritmos biológicos.

## 2.2 Modelaje Mediante Formalismos

La solución de problemas no es ajena a nuestra vida cotidiana, diariamente nos enfrentamos a situaciones que requieren decisiones y problemas que necesitan soluciones. Algunos de éstos están bien definidos y algunos otros no tanto; por ejemplo, el problema del tráfico o la inseguridad en una ciudad, son cuestiones difíciles de modelar porque no tienen límites claros y el estado actual y deseado no son precisos.

Por otra parte, están los problemas bien definidos, éstos pueden ser modelados mediante un **estado inicial**, un **estado final** y una **solución**. El estado inicial representa el punto de partida y los datos de entrada; el estado final se refiere a la meta o estado deseado al cual queremos llegar y, la solución es el conjunto de pasos que debemos seguir para transformar la situación inicial en la situación final. Este último paso es conocido como **algoritmo**, del cual ya hablamos al inicio de este capítulo. La descripción anterior se puede visualizar así:



Este tipo de problemas es el que se pueden abordar mediante un modelaje formal, lo cual se aplica bastante bien para los algoritmos biológicos que trata este libro, son problemas en los cuales se pueden definir con claridad las entradas, las salidas y los procesos involucrados. De nuevo, recomendamos el análisis íntegro para el desarrollo de un sistema completo, complejo e integrado.

### 2.2.1 ¿Cómo abordar un problema y plantear su solución?

Para resolver un problema, el primer paso es modelarlo. Para esto es necesario entender el problema, sus alcances y limitaciones; si el problema está bien definido, puede ser modelado mediante los siguientes pasos:

1. Identificar los datos de entrada distinguiendo los elementos constantes y los datos variables
2. Identificar los datos de salida o respuesta al problema planteado
3. Para cada dato determinar el conjunto al cual pertenece y sus características, incluyendo la forma de referirse a él dentro del modelo.
4. Plantear las condiciones iniciales y finales del problema (especificación)
5. Definir un plan de solución (conjunto de instrucciones o algoritmo)

Comúnmente, los tres primeros pasos son el modelaje propiamente dicho, el cuarto paso es la especificación junto con el quinto paso se conoce como refinamiento a pasos y se basa en la técnica "divide y vencerás", el problema no se trata como un todo, se divide en subproblemas más manejables. A continuación se verán en detalle cada uno de estos puntos, teniendo en cuenta que el primero, el modelaje, pertenece a la etapa de análisis y los demás, refinamiento a pasos, a la etapa de diseño de la solución.

### 2.2.2 Modelaje

Dentro del modelaje se identifican los datos iniciales y finales y se definen sus características, éstas son el objeto del mundo que representan, su identificador, su clase y su tipo.

#### 2.2.2.1 Datos de entrada, de salida e intermedios

La identificación de estos datos es muy importante para el posterior desarrollo del problema.

Los datos iniciales son aquellos que necesitamos para resolver la situación planteada pero que no conocemos, estos datos generalmente son consultados al usuario. Existe otro tipo de datos de entrada que son constantes del entorno, estos datos tienen un valor fijo que persiste a lo largo de toda la situación.

Los datos de salida son los que representan las respuestas que deseamos encontrar, son las preguntas que el problema nos plantea y que queremos contestar.

Finalmente, los datos intermedios son aquellos elementos que hacen parte de la solución y son necesarios para llegar a los datos finales.

Los datos de entrada y de salida están dados por el problema en tanto que los datos intermedios dependen de la solución que nosotros planteemos, existen muchas formas de resolver una situación y cada una tiene sus propios datos o variables intermedias.

#### 2.2.2.2 Objeto del mundo

El objeto del mundo es el elemento en el mundo real que estamos representando bien sea como un dato de entrada o de salida, típicamente es un pequeño comentario que nos ayuda a saber rápida y fácilmente a qué nos referimos.

#### 2.2.2.3 Identificador

El identificador es el nombre con el cual bautizamos los datos para referirnos a ellos en una forma cómoda y rápida. Un identificador puede ser una palabra o varias palabras unidas por medio del símbolo '\_', también puede contener otros símbolos que representen algo con respecto a ese dato, por ejemplo '?', '&', etc. Sin embargo, los identificadores no deben empezar con estos símbolos, el carácter inicial debe ser un '\_' o una letra.

Los identificadores deben ser claros y tener relación con el objeto del mundo que representan. Por ejemplo, si el objeto del mundo es una secuencia de aminoácidos, un buen identificador podría ser **sec\_aa** y un mal identificador sería **var1**.

#### 2.2.2.4 Clase

Puede ser variable o constante. Un dato es constante cuando su valor a lo largo de todo el problema y solución no cambia su valor, de otra forma en variable. Generalmente, se acostumbra utilizar identificadores en MAYÚSCULAS para las constantes y en minúsculas o combinación para las variables.

Por ejemplo, si el problema es encontrar la velocidad de un objeto que cae, la aceleración es un dato constante y el espacio que recorre es un dato variable ya que es necesario preguntarlo para poder solucionar el problema.

#### 2.2.2.5 Tipo y Valor

Los datos variables tienen un tipo en tanto que los datos constantes tienen un valor. En el caso de las constantes, el valor está dado por el problema mismo, en tanto que en el caso de las variables, el problema generalmente ofrece pistas de los tipos o son decididos por sentido común. Existen tres tipos básicos de datos los cuáles son:

##### 2.2.2.5.1 Enteros

El tipo entero es el conjunto **Z** con todas sus propiedades y operaciones como son la suma, resta, división, módulo y multiplicación. La división es un caso especial ya que al dividir dos enteros se obtiene un entero y se desprecia la parte decimal:

$$7 \overline{)4} \\ 1$$

### 2.2.2.5.2 Reales

Corresponde al conjunto **R** con propiedades y operaciones, sin embargo, se debe tener cuidado con los decimales, el número 11 es un entero en tanto que el 11.0 es un real. Nuevamente, la división sigue algunas reglas especiales:

$$\begin{array}{r} 11.00 \overline{)4.00} \\ \underline{2.75} \end{array} \quad \begin{array}{r} 11.00 \overline{)4} \\ \underline{2} \end{array} \quad \begin{array}{r} 11 \overline{)4.00} \\ \underline{2} \end{array}$$

### 2.2.2.5.3 Caracteres

Corresponden a los símbolos sencillos e individuales que pueden ser escritos, letras, números y otros: 'e', 'Y', '3', '+', '%', etc. Van encerrados entre comillas sencillas. Vale la pena notar que el carácter '4' es diferente al número 4.

### 2.2.2.5.4 Otros

Existen otros tipos más elaborados y que no se consideran básicos pero que son muy útiles al momento de desarrollar algoritmos biológicos; dentro de esos están:

1. Cadenas de caracteres: "Hola", "Centella", "Una frase", etc.
2. Secuencias o listas: <1, 1, 2, 3, 5, 8> (No tienen una longitud fija).
3. Multilistas: <<1,3,7,5,4,6>, <2,6,2,6,0>, <0,1>, <4,6,7,3,0>> Cada elemento es una secuencia independiente y están reunidas bajo una secuencia de agrupamiento.
4. Vectores: [1][2][4] (tienen una longitud fija).

5. Matrices: [3][7][9]

[4][5][7] (número de filas y columnas fijas).

### Ejemplo de Modelaje

A lo largo de este capítulo se trabajará un mismo ejercicio mostrando los pasos de su modelaje, especificación y refinamiento hasta llegar a una implementación en el lenguaje de programación C.

El ejercicio es el siguiente: Se desea saber si una secuencia de aminoácidos está contenida dentro de otra.

<b><u>Datos Constantes:</u></b>		
No hay		
<b><u>Datos de Entrada</u></b>		
<u>Objeto del Mundo</u>	<u>Identificador</u>	<u>Tipo</u>
Secuencia Principal	Sec_Ppal	Secuencia de Caracteres (String)
Secuencia Secundaria	Sec_Sec	Secuencia de Caracteres (String)
<b><u>Datos de Salida</u></b>		
<u>Objeto del Mundo</u>	<u>Identificador</u>	<u>Tipo</u>
Existencia de la secuencia secundaria dentro de la principal (1=Si, 0=No)	Existe?	Entero (int)

Hasta aquí trataremos la etapa de análisis, a continuación nos dedicaremos a la etapa de diseño, etapa en la cual se plantea la solución y las estructuras necesarias para el desarrollo del problema.

### 2.3 Etapa de Diseño

La etapa de diseño, al igual que la etapa de análisis, es una

de las más importantes en el proceso de resolución de problemas. Es aquí donde se plantean el tratamiento que se le dará a la información y la forma en que se estructurará, se deciden los pasos y la presentación de la solución, buscando siempre eficacia, eficiencia, corrección, robustez y facilidad para los usuarios.

Existen algunos tipos de diseño gráficos, formales, diagramas de flujo, etc. En este capítulo presentaremos algunos de ellos y, por supuesto, siguiendo el formalismo ya planteado en la sección anterior, incluiremos el diseño mediante formalismos.

### 2.3.1 Diagrama de Flujos

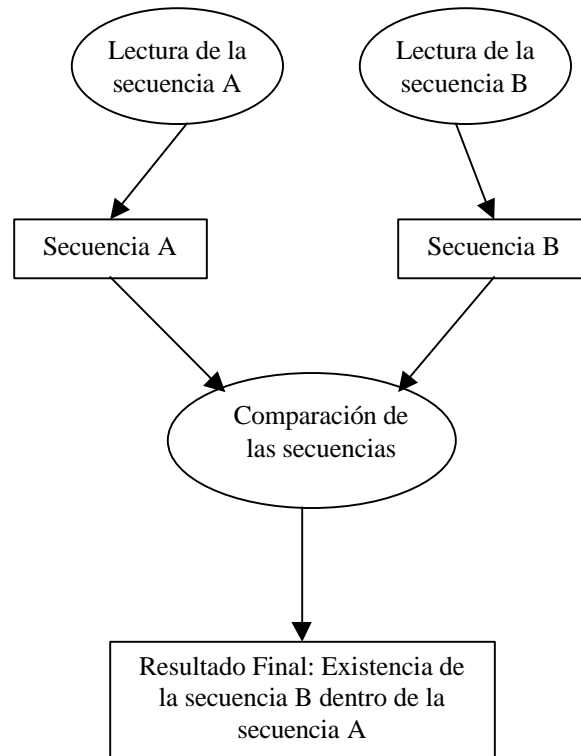
El diagrama de flujos es un método gráfico que presenta todos los pasos de la solución, el orden en el que deben ser realizados y los datos involucrados.

Pensemos, por ejemplo, en el caso que ya hemos considerado, la existencia de una secuencia B dentro de una secuencia A. El diagrama de flujos sería el siguiente: Como se puede observar, los datos se consignan en cajas cuadradas en tanto que los procesos se hallan dentro de cajas ovaladas. Esto se hace para facilitar la lectura del diagrama.

Vale la pena anotar que el diagrama principal puede ser refinado mediante diagramas que ilustren cada proceso, detallando la forma en la que se llevarán a cabo.

El éxito de este método radica en la claridad de los diagramas, para esto es muy importante identificar claramente los grandes procesos para luego ir desglosándolos poco a poco, no puede incluirse todo desde el principio ya que el diagrama resultaría demasiado extenso y complicado, lo cual dificultaría su lectura y

utilización.



### 2.3.2 Diseño Orientado por Objetos

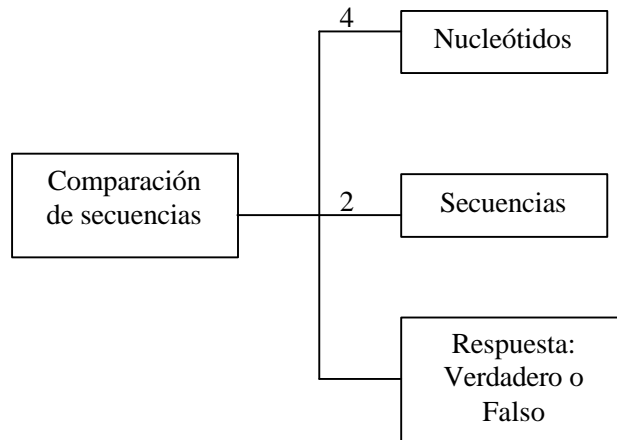
La técnica de orientación por objetos es cada vez más utilizada ya que los objetos buscan representar la realidad; cada objeto modelado debe existir en la vida real.

Este método primero identifica los objetos del mundo real, su descripción, su representación o forma de visualización comúnmente utilizada, sus propiedades, operaciones e interrelaciones con los demás y luego los representa mediante fichas, como se observa en la figura de la página siguiente.

Para cada objeto que surge a partir de la situación problemática, se desarrolla una ficha con todas estas características y el modelo final se consigna por separado

Objeto: Secuencia
Descripción: Secuencia de caracteres correspondientes a nucleótidos (A, T, C, G)
Representación: Sec = <e1, e2, e3, e4, ..., en>
Invariante: $\forall i \in \{1, 2, 3, \dots, n\}, e_i \in \{A, T, C, G\}$
Atributos: Nombre Conjunto de nucleótidos
Operaciones o Métodos: <ul style="list-style-type: none"> <li>- Creación</li> <li>- Adición de un nuevo nucleótido</li> <li>- Eliminación de un nucleótido</li> <li>- Etc.</li> </ul>

mediante un grafo de clases (una clase corresponde a una ficha). El grafo representa la situación completa.

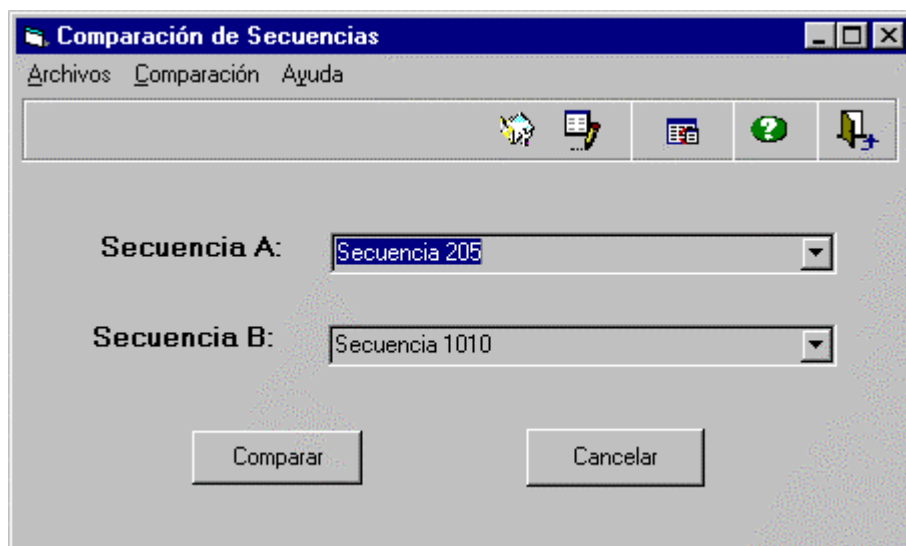


El grafo de clases presenta toda la información necesaria para el desarrollo del problema en cuanto a información se

refiere; incluye información referente a la cantidad de cada elemento en caso de que sea conocida.

### 2.3.3 Diseño Guiado por Prototipos

Este método es muy gráfico y se utiliza más que todo cuando se está desarrollando una solución para alguien más. Consiste en realizar muestras tentativas de la forma en la que se visualizaría la solución e ir cambiándola hasta llegar a una muestra satisfactoria. Es muy similar a la realización de una obra de arte, primero se realizan los bocetos y, finalmente, el producto final. No es muy aplicable para el desarrollo de algoritmos biológicos pero tal vez si para una aplicación que permita su fácil utilización. Se usa más que todo para mostrar las interfaces y dar una idea de los procesos que se manejarían cada interacción.



### 2.3.4 Diseño mediante formalismos

Como ya habíamos dicho en la sección de modelaje mediante formalismos, la etapa de diseño está compuesta por la especificación formal de los datos encontrados

durante el modelaje y la especificación de cada uno de los pasos que conforman la solución.

#### 2.3.4.1 Especificación

La especificación es la determinación de las condiciones iniciales (Precondiciones) y las condiciones finales (Poscondiciones). Estas condiciones están directamente relacionadas con los datos de entrada y salida y determinan sus rangos, valores iniciales y restricciones que deben cumplir.

En la precondición se definen las variables de entrada, su tipo, el rango de valores válidos y las limitaciones adicionales que imponga el problema mismo.

En la poscondición se plantea el estado inicial, indicando la naturaleza de los datos finales, el rango de valores válidos y las restricciones sobre éstos. De igual forma, se establecen las fórmulas que rigen a estos datos, si existen, y se puede hacer referencia a datos intermedios que sean necesarios para precisar el resultado.

Un punto muy importante que se debe tener en cuenta dentro de la especificación y, particularmente, dentro de la poscondición es el objetivo que se persigue. Se trata de plantear las condiciones y no los procesos; no es necesario decir cómo llegamos a la solución, sólo queremos dejar explícita la forma y las condiciones que cumple la respuesta a la que deseamos llegar.

Ejemplo de Especificación

##### **Precondición**

{  
Sec\_Ppal <a.1, a.2, a.3, a.4, a5, ..., a.N>  
 $\forall (1 \leq I \leq N) a.I \in \{a, t, c, g\}$   
Sec\_Sec = <b.1, b.2, b.3, b.4, b.5, ..., b.M>

$$\forall (1 \leq I \leq N) b.I \in \{a, t, c, g\}$$

### **Poscondición**

$$\{$$
$$[ ( \text{Existe?} = 1 ) \wedge ( \text{Sec\_Sec} \subseteq \text{Sec\_Ppal} ) ] \vee$$
$$[ ( \text{Existe?} = 0 ) \wedge ( \text{Sec\_sec} \not\subseteq \text{Sec\_Ppal} ) ]$$
$$\}$$

#### 2.3.4.2 Refinamiento a Pasos

Una vez que se ha realizado el modelaje y la especificación, el conocimiento sobre el problema es el necesario para plantear la solución. Para esto se decide cómo resolver el problema, que pasos son necesarios y que datos adicionales, intermedios, se requieren. Los datos intermedios son definidos de la misma forma que los datos de entrada y salida en el modelaje y los pasos intermedios se determinan mediante condiciones o aserciones intermedias, similares a la precondition y poscondition.

Cada paso intermedio se representa con una aserción, en cada una se presentan las variables o datos intermedios que se están introduciendo, su tipo, rangos y restricciones y las fórmulas que cumplen.

El esquema general de una solución es la siguiente:

Pre - Q1 - Q2 - ... - Qn - Pos

Donde cada Qi es una aserción o paso intermedio, Pre es la precondition y Pos es la poscondition. Vale la pena notar que un par de aserciones consecutivas Q(i), Q(i+1) son la precondition y poscondition del paso o conjunto de instrucciones que me lleva de una a otra.

#### Ejemplo de Refinamiento a Pasos

Dentro del refinamiento a pasos existe la representación de pasos sencillos como instrucciones directas, pasos condicionales y pasos repetitivos. Para estos últimos se

utiliza como refinamiento el **invariante**, que indica los pasos que se repiten y bajo que condiciones se realiza la repetición. A continuación se describe el refinamiento para el ejemplo que venimos tratando:

### **Precondición**

#### **Q1**

```
{
  Long_Ppal  $\wedge$  Long_Sec  $\in$  Naturales
  Long_Ppal = longitud( Sec_Ppal)
  Long_Sec = longitud( Sec_Sec )
}
```

#### **Q2**

```
{
  [( Long_Sec > Long_Ppal )  $\wedge$  ( Existe? = 0 )]  $\vee$ 
  [( Long_Sec  $\leq$  Long_Ppal )  $\wedge$ 
```

#### **Inv1**

```
{
  actual_ppal  $\in$  Sec_Ppal
  rec_ppal = actual_ppal
  rec_sec = b.1
```

#### **Inv2**

```
{
  rec_ppal = rec_sec
  rec_ppal.K = siguiente(rec_ppal.(K-1))
  rec_sec.K = siguiente(rec_ppal.(K-1))
  K es la iteración actual dentro del ciclo
   $0 \leq K \leq$  Long_Sec
```

```
}
[(K=Long_Sec)  $\wedge$  (Existe? = 1 )]  $\vee$ 
[(K< Long_Sec)  $\wedge$  actual_ppal.I = actual_ppal.(I-1)
I es la iteración actual dentro del ciclo
 $0 \leq I \leq$  Long_Ppal]
```

```
}
]
```

}

### **Poscondición**

#### 2.3.4.3 Otros Métodos de Modelaje

El método anterior, y sobre todo la parte de especificación y refinamiento formal, es muy efectivo si uno desea realizar una corrección formal del algoritmo escrito, para esto existen teorías desarrolladas por Hoare, Knuth, entre otros que apoyan el desarrollo y verificación formal de los algoritmos. Sin embargo, estos métodos tan estrictos no son muy utilizados por el común de la gente, están un poco restringidos a quienes los disfrutan y a los matemáticos, quienes están familiarizados con demostraciones, formalismos y pruebas rigurosas.

Como ya se ha dicho anteriormente, la utilización de cierto formalismo a la hora de desarrollar algoritmos es importante para evitar posteriores inconvenientes durante el proceso descrito al inicio del capítulo. Sin embargo, tal vez lo más relevante es algo que está presente en todos los métodos presentados, la técnica de "divide y vencerás", inicialmente identificar los grandes procesos y luego analizarlos y diseñarlos como un todo, para posteriormente integrar cada una de las pequeñas soluciones que conforman la gran solución, la que desde el principio nos planteamos como meta. ¡La división de los problemas en subproblemas más pequeños es la mejor táctica!