Universidad Nacional de Jujuy

Facultad de Ingeniería

Cátedra:

“Desarrollo Sistemático de Programas”

Cartilla Teórica

Profesor:

Ing. Carlos Alberto Afranllie

Año 2021

"Quien pregunta parece ignorante durante un minuto. Quien no pregunta permanece ignorante el resto de su vida".

Proverbio Chino

"Lo que debemos aprender a hacer lo aprendemos haciéndolo".

Aristóteles, Ethica Nicomachea II (325 A.C.)

El presente documento ha sido elaborado originalmente como apoyo a la asignatura *"Desarrollo Sistemático de Programas"* de la carrera Ingeniería Informática de la Facultad de Ingeniería de la Universidad Nacional de Jujuy.

Este documento engloba la mayor parte de la materia e incluye ejemplos resueltos y algunos ejercicios que serán desarrollados en clases.

El apunte ha sido concebido para ser leído en forma secuencial, pero también para ser de fácil consulta para verificar algún tema específico.

No se pretende que este apunte sustituya a la bibliografía de la asignatura ni a las clases teóricas y prácticas, sino que sirva más bien como complemento a las notas que el alumno debe tomar en clases.

Asimismo, no debe considerarse un documento definitivo y exento de errores, si bien ha sido elaborado con detenimiento y revisado exhaustivamente.

Ing. Carlos A. Afranllie

UNIDAD 1 – ALGORITMOS: INTRODUCCIÓN Y REPASO

1- Historia de los Algoritmos

La palabra algoritmo procede del matemático Árabe Mohamed Ibn Al Juarismi (Al'Khwarizmi), el cual escribió sobre los años 800 y 825 su obra Quitad Al Mugabala, donde se recogía el sistema de numeración hindú y el concepto del cero. Este texto se perdió, pero su versión latina, Algoritmi de Numero Indorum, sí se conoce. Fibonacci, tradujo la obra al latín y la llamó: Algoritmi Dicit.

El trabajo de Al'Khwarizmi permitió preservar y difundir el conocimiento de los griegos (con la notable excepción del trabajo de Diofanto) e indios, pilares de nuestra civilización. Rescató de los griegos la rigurosidad y de los indios la simplicidad (en vez de una larga demostración, usar un diagrama junto a la palabra Mira). Sus libros son intuitivos y prácticos y su principal contribución fue simplificar las matemáticas a un nivel entendible por no expertos. En particular muestran las ventajas de usar el sistema decimal indio, un atrevimiento para su época, dado lo tradicional de la cultura árabe.

La exposición clara de cómo calcular de una manera sistemática a través de algoritmos diseñados para ser usados con algún tipo de dispositivo mecánico similar a un ábaco, más que con lápiz y papel, muestra la intuición y el poder de abstracción de Al'Khwarizmi. Hasta se preocupaba de reducir el número de operaciones necesarias en cada cálculo. Por esta razón, aunque no haya sido él el inventor del primer algoritmo, merece que este concepto esté asociado a su nombre.

Los babilonios que habitaron en la antigua Mesopotamia, empleaban unas pequeñas bolas hechas de semillas o pequeñas piedras, a manera de "cuentas" y que eran agrupadas en carriles de caña. Más aún, en 1.800 A.C. un matemático babilónico inventó los algoritmos que le permitieron resolver problemas de cálculo numérico.

En 1850 A.C., un algoritmo de multiplicación similar al de expansión binaria es usado por los egipcios.

La teoría de las ciencias de la computación trata cualquier objeto computacional para el cual se puede crear un buen modelo. La investigación en modelos formales de computación se inició en los 30's y 40's por Turing, Post, Kleene, Church y otros. En los 50's y 60's los lenguajes de programación, compiladores y sistemas operativos estaban en desarrollo, por lo tanto, se convirtieron tanto en el sujeto como la base para la mayoría del trabajo teórico.

El poder de las computadoras en este período estaba limitado por procesadores lentos y por pequeñas cantidades de memoria. Así, se desarrollaron teorías (modelos, algoritmos y análisis) para hacer un uso eficiente de ellas. Esto dio origen al desarrollo del área que ahora se conoce como "Algoritmos y Estructuras de Datos". Al mismo tiempo se hicieron estudios para comprender la complejidad inherente en la solución de algunos problemas. Esto dio origen a lo que se conoce como la jerarquía de problemas computacionales y al área de "Complejidad Computacional".

2- Definición de algoritmo

El concepto intuitivo de algoritmo, lo tenemos prácticamente todos: Un algoritmo es una serie finita de pasos para resolver un problema.

Hay que hacer énfasis en dos aspectos para que un algoritmo exista:

1. El número de pasos debe ser finito. De esta manera el algoritmo debe terminar en un tiempo finito con la solución del problema.
2. El algoritmo debe ser capaz de determinar la solución del problema.

De este modo, podemos definir algoritmo como un "conjunto de reglas operacionales inherentes a un cómputo". Se trata de un método sistemático, susceptible de ser realizado mecánicamente, para resolver un problema dado.

Sería un error creer que los algoritmos son exclusivos de la informática. También son algoritmos los que aprendemos en la escuela para multiplicar y dividir números de varias cifras. De hecho, el algoritmo más famoso de la historia se remonta a la antigüedad: se trata del algoritmo de Euclides para calcular el máximo común divisor.

Siempre que se desee resolver un problema hay que plantearse qué algoritmo utilizar. La respuesta a esta cuestión puede depender de numerosos factores, a saber, el tamaño del problema, el modo en que está planteado y el tipo y la potencia del equipo disponible para su resolución.

Según la RAE (Real Academia Española): ***"conjunto ordenado y finito de operaciones que permite hallar la solución de un problema"***

Otra definicióin: ***"Un algoritmo es un conjunto de pasos claramente definidos que a partir de una cierta entrada (input) produce una determinada salida (output)".***

Los algoritmos, como indica su definición oficial, son una serie de pasos que permiten obtener la solución a un problema.

El lenguaje algorítmico es aquel que implementa una solución teórica a un problema indicando las operaciones a realizar y el orden en el que se deben efectuarse. Por ejemplo en el caso de que nos encontremos en casa con un foco quemado en una lámpara, un posible algoritmo sería:

(1) Comprobar si hay focos de repuesto
(2) En el caso de que las haya, sustituir el foco quemado anterior por el nuevo
(3) Si no hay focos de repuesto, comprar en un negocio y sustituir el foco quemado por el nuevo.

Los algoritmos son la base de la programación de computadoras, ya que los programas de computadora se puede entender que son algoritmos escritos en un código especial entendible por una computadora.

Lo malo del diseño de algoritmos está en que no podemos escribir lo que deseemos, el lenguaje a utilizar no debe dejar posibilidad de duda, debe recoger todas las posibilidades. Por lo que los tres pasos anteriores pueden ser mucho más largos:

[1] Comprobar si hay focos de repuesto
    (1.1) Abrir el cajón de los focos
    (1.2) Observar si hay focos
[2] Si hay focos:
    (2.1) Tomar el foco
    (2.2) Agarrar una silla
    (2.3) Subirse a la silla
    (2.4) Sacar el foco quemado
    (2.5) Poner el foco nuevo en la lámpara
[3] Si no hay focos
(3.1) Abrir la puerta
(3.2) Ir al negocio....

Cómo se observa en un algoritmo las instrucciones pueden ser más largas de lo que parecen, por lo que hay que determinar qué instrucciones se pueden utilizar y qué instrucciones no se pueden utilizar. En el caso de los algoritmos preparados para la computadora, se pueden utilizar sólo instrucciones muy concretas.

3- Características de los algoritmos

1. *Entrada:* definir lo que necesita el algoritmo
2. *Salida:* definir lo que produce.
3. *No ambiguo:* explícito, siempre sabe qué comando ejecutar.
4. *Finito:* El algoritmo termina en un número finito de pasos.

5. *Correcto:* Hace lo que se supone que debe hacer. La solución es correcta
6. *Efectividad:* Cada instrucción se completa en tiempo finito. Cada instrucción debe ser lo suficientemente básica como para que en principio pueda ser ejecutada por cualquier persona usando papel y lápiz.
7. *General: Debe ser lo suficientemente general como para contemplar todos los casos de entrada.*

Así podemos, decir que un **Algoritmo es un conjunto finito de instrucciones precisas para resolver un problema.**

Un **algoritmo** es un método o proceso seguido para resolver un problema. Si el problema es visto como una función, entonces el algoritmo toma una entrada y la transforma en la salida.

Un **problema** es una función o asociación de entradas con salidas. Un problema puede tener muchos algoritmos.

Por tanto, un **algoritmo** es un procedimiento para resolver un problema cuyos pasos son concretos y no ambiguos. El algoritmo debe ser correcto, de longitud finita y debe terminar para todas las entradas.

3.1 Características que deben de cumplir los algoritmos obligatoriamente

- Un algoritmo debe resolver el problema para el que fue formulado: Lógicamente no sirve un algoritmo que no resuelve ese problema. En el caso de los programadores, a veces crean algoritmos que resuelven problemas diferentes al planteado.

- Los algoritmos son independientes de la computadora: Los algoritmos se escriben para poder ser utilizados en cualquier máquina.

- Los algoritmos deben de ser precisos: Los resultados de los cálculos deben de ser exactos, de manera rigurosa. No es válido un algoritmo que sólo aproxime la solución.

- Los algoritmos deben de ser finitos: Deben de finalizar en algún momento. No es un algoritmo válido aquel que produce situaciones en las que el algoritmo no termina.

- Los algoritmos deben de poder repetirse: Deben de permitir su ejecución las veces que haga falta. No son válidos los que tras ejecutarse una vez, ya no pueden volver a hacerlo por la razón que sea.

3.2 Características aconsejables para los algoritmos

- Validez: Un algoritmo es válido si carece de errores. Un algoritmo puede resolver el problema para el que se planteó y sin embargo no ser válido debido a que posee errores

- Eficiencia: Un algoritmo es eficiente si obtiene la solución al problema en poco tiempo. No lo es si es lento en obtener el resultado.

- Óptimo: Un algoritmo es óptimo si es el más eficiente posible y no contiene errores. La búsqueda de este algoritmo es el objetivo prioritario del programador. No siempre podemos garantizar que el algoritmo hallado es el óptimo, a veces sí.

4- Elementos que conforman un algoritmo

- Entrada: Los datos iniciales que posee el algoritmo antes de ejecutarse.
- Proceso: Acciones que lleva a cabo el algoritmo.
- Salida: Datos que obtiene finalmente el algoritmo.

5- Fases en la creación de algoritmos

Hay tres fases en la elaboración de un algoritmo:

1. Análisis: En esta se determina cuál es exactamente el problema a resolver. Qué datos forman la entrada del algoritmo y cuáles deberán obtenerse como salida.
2. Diseño: Elaboración del algoritmo
3. Prueba: Comprobación del resultado. Se observa si el algoritmo obtiene la salida esperada para todas las entradas.

6- Validación de un algoritmo

Un algoritmo es correcto si el resultado que produce siempre resuelve un determinado problema a partir de una entrada válida. Demostrar ya sea en forma rigurosa o intuitiva que un algoritmo es correcto es el primer paso indispensable en el análisis de un algoritmo.

A esta fase del análisis de algoritmos se la conoce como validación del algoritmo. Los algoritmos que no son correctos a veces pueden ser útiles si, por ejemplo, producen una respuesta aproximada a un problema particularmente difícil en forma eficiente.

Importante: *El primer paso en todo análisis de un algoritmo es validarlo, es decir, demostrar que el algoritmo es correcto.*

7- Algoritmos determinísticos y no determinísticos

Un algoritmo es determinístico si la respuesta que produce se puede conocer a partir de los datos de entrada.

Un algoritmo es no-determinístico cuando no es determinístico. Que un algoritmo sea o no sea determinístico no aporta dato alguno sobre la corrección del algoritmo.

El estudio de los algoritmos se puede dividir en dos grandes categorías: el análisis de algoritmos y el diseño de algoritmos.

8- Algoritmos y programas

Debemos distinguir algoritmo de programa, un algoritmo es independiente del lenguaje en el cual se programa, de la máquina en la cual se implemente y de otras restricciones que hacen a la puesta en operación del algoritmo.

Un **programa** es una instanciación de un algoritmo en un lenguaje de programación.

Desde el punto de vista del estudio de los algoritmos los mismos pueden considerarse como entidades matemáticas abstractas independientes de restricciones tecnológicas.

9- Razones para estudiar los Algoritmos

Con el logro de computadores cada vez más rápidos se podría caer en la tentación de preguntarse si vale la pena preocuparse por aumentar la eficiencia de los algoritmos. ¿No sería más sencillo aguardar a la siguiente generación de computadores?. Vamos a demostrar que esto no es así.

Supongamos que se dispone, para resolver un problema dado, de un algoritmo que necesita un tiempo exponencial y que, en un cierto computador, una implementación del mismo emplea 10-4 x 2n segundos. Este programa podrá resolver un ejemplar de tamaño n=10 en una décima de segundo. Necesitará casi 10 minutos para resolver un ejemplar de tamaño 20. Un día entero no bastará para resolver uno de tamaño 30. En un año de cálculo ininterrumpido, a duras penas se resolverá uno de tamaño 38. Imaginemos que necesitamos resolver ejemplares más grandes y compramos para ello un computador cien veces más rápido. El mismo algoritmo conseguirá resolver ahora un ejemplar de tamaño n en sólo 10-6 2n segundos. ¡Qué decepción al constatar que, en un año, apenas se consigue resolver un ejemplar de tamaño 45!

En general, si en un tiempo dado se podía resolver un ejemplar de tamaño n, con el nuevo computador se resolverá uno de tamaño n+7 en ese mismo tiempo.

Imaginemos, en cambio, que investigamos en algorítmica y encontramos un algoritmo capaz de resolver el mismo problema en un tiempo cúbico. La implementación de este algoritmo en el computador inicial podría necesitar, por ejemplo, 10-2 n3 segundos. Ahora se podría resolver en un día un ejemplar de un tamaño superior a 200. Un año permitiría alcanzar casi el tamaño 1500.

Por tanto, el nuevo algoritmo no sólo permite una aceleración más espectacular que la compra de un equipo más rápido, sino que hace dicha compra más rentable.

Dado estos argumentos, podemos resumir las siguientes razones para justificar el estudiar los algoritmos:

1. Evitar reinventar la rueda.
   Para algunos problemas de programación ya existen buenos algoritmos para solucionarlos. Para esos algoritmos ya fueron analizadas sus propiedades. Por ejemplo, confianza en su correctitud y eficiencia.

2. Para ayudar cuando desarrollen sus propios algoritmos.
   No siempre existe un algoritmo desarrollado para resolver un problema. No existe regla general de creación de algoritmos. Muchos de los principios de proyecto de algoritmos ilustrados por algunos de los algoritmos que estudiaremos son importantes en todos los problemas de programación. El conocimiento de los algoritmos bien definidos provee una fuente de ideas que pueden ser aplicadas a nuevos algoritmos.

3. Ayudar a entender herramientas que usan algoritmos particulares. Por ejemplo, herramientas de compresión de datos:

   Pack usa Códigos Huffman.
   Compress usa LZW.
   Gzip usa Lempel-Ziv.

4. Útil conocer técnicas empleadas para resolver problemas de determinados tipos.

10- Formas de representación de Algoritmos

Existen diversas formas de representación de algoritmos, pero no hay un consenso con relación a cuál de ellas es mejor.

Algunas formas de representación de algoritmos tratan los problemas a un nivel lógico, abstrayéndose de detalles de implementación, muchas veces relacionados con un lenguaje de programación específico.

Por otro lado, existen formas de representación de algoritmos que poseen una mayor riqueza de detalles y muchas veces acaban por oscurecer la idea principal, el algoritmo, dificultando su entendimiento.

Dentro de las formas de representación de algoritmos más conocidas, sobresalen:

1. La descripción narrativa
2. El Flujograma convencional
3. El diagrama Chapin
4. El pseudocódigo, o también conocido como lenguaje estructurado