Algoritmia y programación

Definición de algoritmo. Representación y descubrimiento de algoritmos. Eficiencia y corrección.

Definición de algoritmo

Conjunto ordenado de pasos no ambiguos y ejecutables que definen un proceso que termina.

- Los pasos deben tener una estructura bien establecida en términos del orden en que se ejecutarán.
- Durante la ejecución de un algoritmo, la información de estado del proceso debe ser suficiente para determinar unívoca y completamente las acciones necesarias en cada paso.
- Los pasos deben ser ejecutables en el sentido de que deben ser factibles. Por ejemplo “listar los enteros positivos” no sería algo factible.
- La ejecución de un algoritmo debe conducir a un final.

Descubrimiento de algoritmos

- El desarrollo de un programa consiste de dos actividades: descubrir un algoritmo y representarlo como un programa.
- Descubrir un algoritmo para resolver un problema equivale encontrar un método para resolver ese problema.
- Entender cómo descubrir algoritmos es entender el proceso de resolución de problemas.
- La capacidad para resolver problemas es una habilidad artística que se debe desarrollar más que una ciencia precisa que debe aprenderse.

Descubrimiento de algoritmos

Como evidencia de la naturaleza imprecisa y artística de la resolución de problemas, las fases presentadas por el matemático George Pólya en 1945 para resolver problemas permanecen como las bases para el desarrollo de esta habilidad:

1. Entender el problema
2. Concebir un plan para resolverlo
3. Llevar a cabo el plan
4. Evaluar la precisión de la solución y su potencial como herramienta para resolver otros problemas

Descubrimiento de algoritmos

Traducción de las fases de Pólya al desarrollo de programas:

1. Entender el problema
2. Idear cómo un procedimiento algorítmico podría resolver el problema
3. Formular el algoritmo y representarlo como programa
4. Evaluar la precisión del programa y su potencial como herramienta para resolver otros problemas

Estas fases no necesariamente se completan en secuencia. Muchas veces se comienza por la fase 2 sin haber comprendido el problema completamente y por prueba y error se gana comprensión hasta alcanzar un procedimiento exitoso.

Generalidad de algoritmos

- Generalmente, el desarrollo de un programa no es el proceso de resolver una instancia particular de un problema sino encontrar un algoritmo general que pueda usarse para resolver todas las instancias del problema.
- Para encontrar una solución general a un problema se debe considerar todos los escenarios posibles o variantes del problema y encontrar los principios generales que permitan solucionar cada variante.
- Por ejemplo, si se deseara calcular la diferencia entre dos horas de un mismo día con precisión de minutos y segundos, el algoritmo debe contemplar casos de diferencia nula, o de diferencias de sólo segundos, de sólo minutos y segundos, o de horas, minutos y segundos, incluyendo casos de diferencia nula en segundos o minutos.

Estrategias para resolver problemas

Hay varias aproximaciones generales para descubrir soluciones a problemas:

- **Trabajar el problema hacia atrás**: desarrollo basado en datos de prueba. Si el problema se piensa como encontrar una forma de producir resultados particulares a partir de ciertos datos de entrada, se puede encontrar una solución pensando en el proceso inverso.
- **Por analogía**: pensar en problemas relacionados fáciles de resolver o que ya se han resuelto y tratar de aplicar sus soluciones al problema actual.
- **Por refinamientos sucesivos**: descomponer el problema en subproblemas más fáciles de resolver que el problema completo. Este enfoque también se denomina **descendente** *(top-down)*, en el sentido de que progresa de lo general a lo particular. El enfoque opuesto, **ascendente** *(bottom-up)* muchas veces complementa al anterior en soluciones creativas.

Representación de algoritmos

- La representación de un algoritmo puede ser problemática cuando se trata de comunicar el algoritmo para que se pueda ejecutar, sobre todo en el nivel de detalle con el que se describe.
- La informática establece un conjunto bien definido de bloques de construcción para representar algoritmos, llamados **primitivas**.
- El uso de primitivas elimina cualquier problema de ambigüedad y establece un nivel de detalle uniforme.
- Un conjunto de primitivas junto con las reglas para combinarlas para representar ideas más complejas constituye un **lenguaje de programación**.

Primitivas

- De definición de funciones
  - Determinar nombre y código de subprogramas con nombres formales de sus argumentos (parámetros).
- De entrada/salida

  Leer del teclado *(standard input)* o de un archivo, o escribir en pantalla *(standard output)* o en un archivo valores asociados a variables o resultados de una expresión o cómputo.
- De almacenamiento de resultados

  Asignación del resultado de un cómputo a una variable.
- De control
  - Condicional: selección de una de dos actividades dependiendo de la verdad o falsedad de una condición, o de dos o más actividades dependiendo del condiciones determinadas.
  - Iterativo: repetición de un conjunto de pasos condicionada por la verdad o falsedad de una condición.
  - De transferencia-retorno: invocación de subprogramas usando parámetros reales (variables o expresiones).

Eficiencia y corrección de algoritmos

- La **eficiencia** de un algoritmo o programa es una característica que sólo puede evaluarse en relación con otro algoritmo alternativo que resuelva el mismo problema, y se mide en términos de la cantidad de instrucciones que debe ejecutar cada uno (especialmente considerando las repeticiones de ciclos) con los mismos datos de prueba. Es más eficiente el algoritmo que obtenga un resultado correcto ejecutando la menor cantidad de instrucciones.
- La **corrección** de un algoritmo se evalúa comprobando que para los datos de prueba que se hayan diseñado, su ejecución devuelva los resultados esperados.

Caso de Ejemplo para Solución Algorítmica

Diferencia entre dos horas de un mismo día

Diseño de datos de prueba para un horario inicial hi:mi:si y otro final hf:mf:sf tales que hi:mi:si < hf:mf:sf

- mi > mf
  - si > sf: 9:15:30 – 8:30:45; (9-8-1):(60-30+15-1):(60-45+30)
  - si ≤ sf: 9:15:30 – 8:30:30; (9-8-1):(60-30+15):(30-30)
- mi < mf
  - si > sf: 9:30:30 – 8:15:45; (9-8):(30-15-1):(60-45+30)
  - si ≤ sf: 9:30:45 – 8:15:30; (9-8):(30-15):(45-30)
- mi = mf
  - si > sf: 9:30:30 – 8:30:45; (9-8-1):(59):(60-45+30)
  - si ≤ sf: 9:30:45 – 8:30:30; (9-8):(30-30):(45-30)