Fundamentos de programación

0.8 Estructuras de datos y algoritmos

0.8.4 Definición de algoritmo y órdenes de complejidad

¿Qué es un Algoritmo?
Un algoritmo es un conjunto de instrucciones ordenadas que, seguidas paso a paso, realizan una tarea específica. Ejemplos de tareas realizadas por algoritmos incluyen:

- Ordenar una lista.
- Recorrer un mapa.
- Encontrar el camino más rápido entre dos puntos.

Clasificación de los Algoritmos

Orden de Complejidad
La eficiencia de un algoritmo se mide a través del "orden de complejidad", que indica cuántas instrucciones se deben ejecutar según el tamaño de los datos de entrada. Los tipos más comunes de orden de complejidad son:

Orden Constante (O(1))
El número de instrucciones es fijo y no depende del tamaño de los datos. Es el más eficiente.

- Ejemplo: Acceder a un elemento específico de un array.

Orden Lineal (O(n))
El número de instrucciones crece linealmente con el tamaño de los datos.

- Ejemplo: Recorrer un array y mostrar cada elemento.

Orden Cuadrático (O(n^2))
El número de instrucciones crece cuadráticamente con el tamaño de los datos.

- Ejemplo: Iterar sobre cada par de elementos en un array anidado.

Orden Logarítmico (O(log n))
El crecimiento del número de instrucciones es logarítmico con respecto al tamaño de los datos. Este orden es más eficiente que el lineal y se busca en muchos algoritmos.

- Ejemplo: Búsqueda binaria en un array ordenado.

Algoritmos Comunes

Algoritmos de Búsqueda

- **Búsqueda Lineal**: Recorre cada elemento de la lista hasta encontrar el objetivo.
- **Búsqueda Binaria**: Divide repetidamente la lista ordenada en mitades para encontrar el objetivo.

Algoritmos de Ordenación

- **Ordenación por Burbuja**: Compara y intercambia elementos adyacentes para ordenar la lista.
- **Ordenación por Inserción**: Construye la lista ordenada insertando elementos uno por uno en su posición correcta.
- **Ordenación Rápida (Quicksort)**: Divide y conquista, elige un pivote y ordena los elementos alrededor del pivote.

0.8.4 Definición de algoritmo y órdenes de complejidad

Algoritmos de Caminos Mínimos

- **Dijkstra**: Encuentra el camino más corto desde un nodo origen a todos los demás nodos en un grafo con pesos no negativos.
- **Floyd-Warshall**: Encuentra los caminos más cortos entre todos los pares de nodos en un grafo.

Ejemplos en Código

A continuación, se presentan ejemplos básicos de algunos algoritmos en pseudocódigo.

Búsqueda Lineal

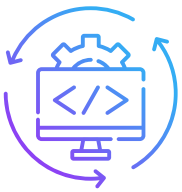
```
función búsquedaLineal(lista, objetivo)
  para cada elemento en lista
    si elemento == objetivo
      retornar verdadero
  retornar falso
```

Búsqueda Binaria

```
función búsquedaBinaria(listaOrdenada, objetivo)
  inicio = 0
  fin = tamaño de lista - 1
  mientras inicio <= fin
    medio = (inicio + fin) / 2
    si listaOrdenada[medio] == objetivo
      retornar verdadero
    sino si listaOrdenada[medio] < objetivo
      inicio = medio + 1
    sino
      fin = medio - 1
  retornar falso
```

Ordenación por Burbuja

```
función ordenaciónBurbuja(lista)
  n = tamaño de lista
  para i desde 0 hasta n-1
    para j desde 0 hasta n-i-1
      si lista[j] > lista[j+1]
        intercambiar lista[j] y lista[j+1]
```



0.8.4 Definición de algoritmo y órdenes de complejidad

Dijkstra

```
función dijkstra(grafo, nodoOrigen)
  distancia = lista de infinito para cada nodo
  distancia[nodoOrigen] = 0
  colaPrioridad = cola de prioridad con nodoOrigen
  mientras colaPrioridad no esté vacía
    nodoActual = extraerMinimo(colaPrioridad)
    para cada vecino de nodoActual
      si distancia a vecino > distancia a nodoActual + peso de arista
        actualizar distancia a vecino
        insertar o actualizar vecino en colaPrioridad
  retornar distancia
```