**Grado de complejidad**

**Grado de complejidad 1**

|  |  |  |
| --- | --- | --- |
| Grado de complejidad | Número de instrucciones | Función |
| F(n) = 4  O(n) = 1 | 1  1  1 | m: = 3  b: = 35  Resul: = b/m |
|  | 1  3  3 | m: = a;  if b<m then m: = b;  if c<m then m: = c; |
| F(n) = 4 | 4 | If (a<=b) and (a<=c) then m:=a; |
|  |  |  |

**Algoritmos de ordenamiento**

**Burbuja:** Empuja el dato más grande hasta el final.

**Selección:** Busca el dato que le interesa (Más pequeño) y lo coloca de primero.

**Inserción:** Se va acomodando cada dato en la posición que va.

**Shell Sort:** Parecido al de inserción.

(Los algoritmos de ordenamiento vistos anteriormente son los menos eficientes para organizar grandes volúmenes de datos)

**Radix Sort:**

**Bucket Sort:**

**Heap Sort:**

**Merge Sort:** Identifica paquetes más pequeños para organizarlos 10, 20, 30

**Quicksort:** Se debe identificar un centro para ordenar los mayores a él y los menores a él

(Algoritmo más rápido de ordenamiento)

**Características**

**Divide y vencerás: (Método) (Técnica)**

* El problema puede dividirse en partes iguales.

**Programación dinámica: (Método) (Técnica)**

* Se puede resolver cuando se repiten operaciones con los mismos datos.

**Backtracking: (Método) (Árboles)**

* Cada nodo del árbol representa un estado, tener un valor.
* La respuesta que encuentre es la mejor
* Procedimiento:

1. Estado inicio.

2. Estado final.

3. Paso de estados.

4. Restricciones.

**Algoritmos Heurísticos: (Definición)**

* Usan reglas de sentido común basados en la experiencia. Ejemplo: para salir de un laberinto lo recorre pegado al lado derecho.
* Puede (o no) encontrar una solución.
* Pero… puede encontrar unas pésimas soluciones o no encontrar ninguna.
  + Es una función matemática que cuando entra a un nodo arroja un número.
  + El número final da 0
  + A medida que transcurre el valor tiende a 0

**Algoritmos avaros y métodos voraces: (Método) (Árboles)**

* Escojo una **heurística** quees una opción que yo asumo ser el más prometedor y sigo por ahí hasta encontrar el resultado
* Procedimiento:

1. Un candidato.

2. Tener un conjunto de decisiones (Candidatos ya escogidos).

3. Una función que me indique si un conjunto puede ser una solución.

4. Una función que determina si un conjunto es completable, si añadiendo ese

conjunto es posible alcanzar la solución al problema, suponiendo que esta exista

5. Una función de selección que escoge el candidato más prometedor.

6. Una función objetivo a maximizar o minimizar.

**Métodos aproximados: (Método) (Árboles)**

* Los algoritmos aproximados son algoritmos heurísticos que:
  + Siempre encuentran una solución.
  + No siempre existe un algoritmo aproximado para uno heurístico.
  + El error máximo está acotado por:
    - Valor absoluto máximo entre la solución obtenida menos la óptima.
    - Valor máximo de la razón entre la solución óptima y la obtenida.

**Métodos maestros: (Método para calcular la complejidad)**

* Para hallar la complejidad de métodos recursivos.

**Hashing: ( )**

* Como hacer para almacenar información y su búsqueda sea más eficiente.
* Almacena el número 100 en la casilla número 100 del arreglo, 101 en la posición 101…
* Coge el número, hace una operación y coloca el número en el arreglo, no recorre el arreglo para colocarlo.

**Ventajas**

* Se puede trabajar con las llaves, la traducción es transparente
* Las llaves son independientes del espacio de direcciones
* No se requiere almacenamiento adicional para índices
* O(1)

**Desventajas**

* No pueden usarse registros de longitud variable
* La información no está ordenada
* No permite llaves repetidas
* Sólo permite acceso por una sola llave
* O(huy!) si hay muchas colisiones

**Grafos ( )**

**Dijsktra:** Es un algoritmo en el que encuentra el camino más corto entre un modo A y un nodo B (no necesita recorrer todos los nodos)

* Recorre la estructura en anchura

**Floyd y Warshall:**