Tarea #5

Análisis de algoritmos

Un *algoritmo* es una secuencia de pasos lógica para encontrar la solución de un problema.

Todo algoritmo debe contar con las siguientes características: *preciso*, *definido* y *finito*. Por Preciso, entenderemos que cada paso del algoritmo tiene una relación con el anterior y el siguiente; un algoritmo es Definido, cuando se ejecuta más de una vez con los mismos datos y el resultado es el mismo; y Finito, indica que el algoritmo cuenta con una serie de pasos definidos o que tiene un fin.

Hablando de estructuras de datos podemos decir que los algoritmos según su función se dividen en:

- Algoritmos de ordenamiento y
- Algoritmos de búsqueda.

Un *algoritmo de ordenamiento*, es el que pone los elementos de una lista o vector en una secuencia (ascendente o descendente) diferente a la entrada, es decir, el resultado de salida debe ser una permutación (reordenamiento) de la entrada que satisfaga la relación de orden requerida.

Un *algoritmo de búsqueda*, es aquel que está diseñado para encontrar la solución de un problema boleano de existencia o no de un elemento en particular dentro de un conjunto finito de elementos (estructura de datos), es decir al finalizar el algoritmo este debe decir si el elemento en cuestión existe o no en ese conjunto, además, en caso de existir, el algoritmo podría proporcionar la localización del elemento dentro del conjunto.

Complejidad en espacio de memoria

Es la memoria que utiliza un programa para su ejecución. Lo que implica que la eficiencia en memoria de un algoritmo lo indica la cantidad de espacio requerido para ejecutarlo, es decir, el espacio memoria que ocupan todas las variables propias del algoritmo.

Esta se divide en Memoria Estática y Memoria Dinámica.

Memoria estática. Para calcularla se suma de memoria que ocupan las variables declaradas en el algoritmo.

Memoria dinámica. Su cálculo no es tan simple ya que depende de cada ejecución del algoritmo.

Ejemplo
Algoritmo de Búsqueda en Arboles

Función ---->busqueda_arboles (problema).
Devuelve ---->solución/fallo.
Inicializa ---->árbol de búsqueda con estado inicial.
Ciclo Hacer
Si ---->no hay candidatos para expandir.
Entonces ---->devolver fallo.
En Otro Caso ---->escoger nodo para expandir.
Si ---->el nodo es el objetivo.
Entonces ---->devolver solución.
En Otro Caso ---->expandir nodo

Resultados Obtenidos

Depth	Nodes	Time	Memory
0	1	1 milisecond	100 bytes
2	111	0.1 seconds	11 kilobytes
4	11111	1.1 seconds	1 megabytes
6	10^6	18 minutes	111 megabytes
8	10^8	31 hours	11 gigabytes
10	10^10	128 days	1 terabytes
12	10^12	35 years	111 terabytes

Eficiencia de algoritmo

Medida del uso de los recursos computacionales requeridos por la ejecución de un algoritmo en función del tamaño de las entradas.

T(n) Tiempo empleado para ejecutar el algoritmo con una entrada de tamaño n

Tipos de análisis

¿Cómo medimos el tiempo de ejecución de un algoritmo?

- Mejor caso En condiciones óptimas (no se usa por ser demasiado optimista).
- Peor caso En el peor escenario posible (nos permite acotar el tiempo de ejecución).
- Caso promedio Caso difícil de caracterizar en la práctica.
- Análisis probabilístico Asume una distribución de probabilidad sobre las posibles entradas.
- Análisis amortizado Tiempo medio de ejecución por operación sobre una secuencia de ejecuciones sucesivas.

Ejemplo:

- Algoritmo 1:
T(n) = 10^-4 2^n segundos
n = 38 datos
T(n) = 1 año
- Algoritmo 2:
T(n) = 10^-2 n^3 segundos
n = 1000 bits
T(n) = 1 año

¿Cuál es mejor? Se precisa un análisis asintótico

https://sites.google.com/site/estdatjiq/home/unidad-vii

http://iscestructuradedatosjesusolivas.blogspot.com/2008/09/complejidad-en-el-espacio.html

https://www.programacionfacil.com/estructura_datos_csharp/complejidad_espacio.html

http://elvex.ugr.es/decsai/algorithms/slides/2%20Eficiencia.pdf