Best, worst & average case analysis

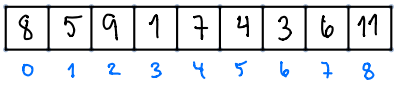
- El mejor de los casos (*best case*) se refiere a una situación de los datos que genera una ejecución de un algoritmo con la menor complejidad posible.
- El peor de los casos (*worst case*) se refiere a una situación de los datos que genera una ejecución de un algoritmo con la mayor complejidad posible.
- El caso promedio (*average case*) se refiere a una situación de los datos que no genera ningún patrón que aporte ventajas o desventajas. Es considerado como la situación típica de ejecución de un algoritmo.

⚠ En la mayoría de los algoritmos el caso promedio tiene la misma complejidad que el peor de los casos, es por esta razón que usualmente se analiza este último.

Ejemplo 1. Linear search o búsqueda lineal

La búsqueda lineal es un método para encontrar un valor objetivo dentro de una lista. Esta consiste en la evaluación secuencial de cada elemento de la lista para encontrar el valor objetivo, o bien, para determinar que este no está presente en la lista.

Supongamos que tenemos una lista de datos:



Best case

El valor objetivo se encuentra en la posición 0 de la lista → $b(n) = 1$

Worst case

El valor objetivo se encuentra en la última posición de la lista, o bien, no se encuentra en la lista → $w(n) = n$

Average case

Consideramos el número de comparaciones en cada uno de los casos posibles y dividimos el total de comparaciones por el número de casos posibles. Es decir:

$$A(n) = \frac{1 + 2 + 3 + \ldots + n}{n} = \frac{n(n+1)}{2n} = \frac{n+1}{2}$$

En el mejor de los casos, el algoritmo de búsqueda lineal es $b(n) = O(1)$, $b(n) = \Omega(1)$, o bien, $b(n) = \Theta(1)$

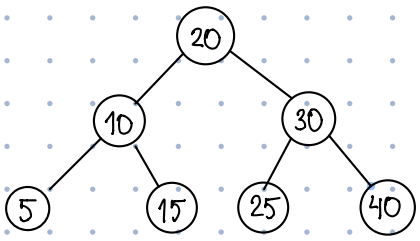
En el peor de los casos, el algoritmo de búsqueda lineal es $w(n) = O(n)$, $w(n) = \Omega(n)$, o bien, $w(n) = \Theta(n)$

En el caso promedio, el algoritmo de búsqueda lineal es $a(n) = O(n)$, $a(n) = \Omega(n)$, o bien, $a(n) = \Theta(n)$

⚠ Como podemos observar, las funciones de tiempo de los casos peor y promedio, aunque son distintas, pertenecen a la misma clase de funciones. Esto será así en la mayoría de los algoritmos que se estudian. Es por esta razón que, en general, se analiza el *worst case* ya que la operatoria involucrada con dicho análisis es considerablemente menor.

Ejemplo 2. Binary search tree

Un árbol binario que cumple con que el subárbol izquierdo de cualquier nodo (si no está vacío) contiene valores menores que el valor de dicho nodo, y el subárbol derecho (si no está vacío) contiene valores mayores que el valor de dicho nodo.



Best case

El valor objetivo coincide con la raíz del árbol → $b(n) = 1$

Worst case

El valor objetivo es uno de los nodos hoja. Entonces, para encontrarlo habría que recorrer todos los niveles del árbol.

El número de niveles en un árbol como el que se muestra puede obtenerse de la siguiente manera:

$$2^0 + 2^1 + 2^2 + 2^3 + \dots + 2^{K-1} = 2^K - 1 \quad , K \geq 1$$

Para mostrar la validez de esta fórmula, usamos el principio de inducción matemática:

Paso base: $K = 1$

$$2^{1-1} = 1 = 2^1 - 1$$

Paso inductivo: Asumimos $\underbrace{2^0 + 2^1 + 2^2 + 2^3 + \dots + 2^{K-1}}_{K\text{-niveles}} = 2^K - 1$

Probamos el caso $K+1$:

$$\underbrace{2^0 + 2^1 + 2^2 + 2^3 + \dots + 2^{K-1}}_{\text{hipótesis inductiva}} + 2^K = \underbrace{2^K - 1}_{} + 2^K$$

$$= 2 \cdot 2^K - 1$$

$$= 2^{K+1} - 1$$

Si le llamamos al número de datos n, tenemos entonces la siguiente relación: $n = 2^K - 1$

Finalmente, despejamos para k (el número de niveles / comparaciones):

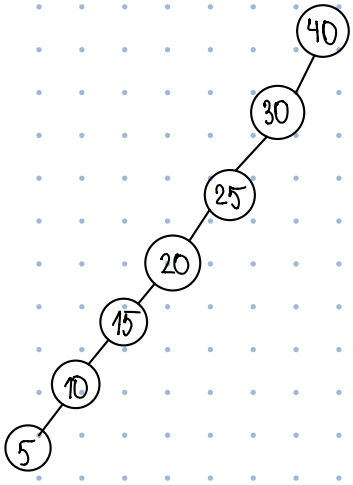
$$n = 2^K - 1 \longrightarrow 2^K = n+1 \longrightarrow K = \log_2(n+1)$$

La función de tiempo $w(n) = \log_2(n+1)$.

En el mejor de los casos, el algoritmo *binary search tree* es $b(n) = O(1)$.

En el peor de los casos, el algoritmo *binary search tree* es $w(n) = O(\log n)$.

⚠️ La estructura del árbol podría ser distinta, como vemos a continuación:



En el mejor de los casos, el algoritmo *binary search tree* es $b(n) = O(1)$.

En el peor de los casos, el algoritmo *binary search tree* es $w(n) = O(n)$

El algoritmo *binary search tree* es:

En el mejor de los casos, $b(n) = O(1)$.

En el peor de los casos mínimo $w(n) = O(\log n)$ y máximo $w(n) = O(n)$.