**Búsqueda Lineal**

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

Para este algoritmo tenemos como operación básica la comparación de los valores del arreglo con el número a buscar.

**Mejor Caso**

El mejor caso se da cuando el valor a buscar se encuentra en la primera posición del arreglo:

**Peor caso**

Esta pasa cuando el valor es el último en el arreglo o no está dentro del arreglo:

**Caso Medio**

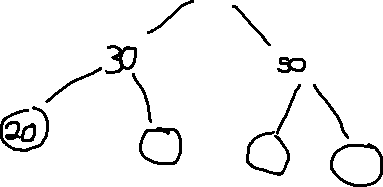
Para el caso medio existen varias posibilidades, tantas como grande sea el arreglo, pues la probabilidad de que el elemento este en la posición 1, 2, 3 o 10 es la misma. Como todos los casos tiene la misma probabilidad:

Para este caso tenemos una sumatoria que indica la posición donde puede estar el arreglo, desde la primera hasta la n, esto dividido entre la probabilidad, por lo tanto, tenemos:

**ÁRBOL BINARIO DE BÚSQUEDA**

Para realizar este análisis sabemos que en un árbol binario de búsqueda se van haciendo comparaciones entre los datos del árbol y el dato buscado, en el subárbol izquierdo tenemos valores menores al valor de la raíz y en el subárbol derecho valores mayores que la raíz.

Analicemos mejor con un ejemplo gráfico.



**MEJOR CASO:**

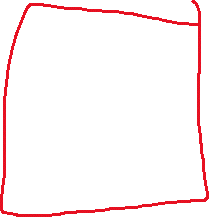
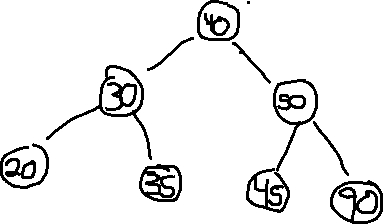
Para el mejor caso es muy sencillo, pues se dará cuando el elemento a buscar este en la raíz, en nuestro ejemplo si el número a buscar es el 40, esta en la raíz, de esta forma solo se hace una comparación:

**PEOR CASO**

Tenemos un primero peor caso si el arreglo con el que se crea el árbol es un arreglo ordenado, pues si está el arreglo ordenado, el árbol se vuelve una lista y se harán n comparaciones:

El segundo peor caso se da con el árbol balanceado como el de nuestro ejemplo, el peor caso dependerá de la altura o profundidad del árbol, sabemos que se hacen comparaciones con los nodos del árbol, si el dato a buscar es mayor que el del nodo se va al subárbol derecho y si es menor al izquierdo, reduciendo a la mitad el numero de nodos a comparar.

Si buscamos el 20 en nuestro ejemplo tenemos:



En lugar de buscar en todos los 6 nodos, solo se hace en la mitad izquierda por que es menor a la raíz, posteriormente en lugar de buscar en dos nodos, solo busca en uno por que es menor a 30. De esta forma vemos que se van haciendo divisiones entre dos, de esta forma la función complejidad temporal para el peor caso será:

**CASO MEDIO**

Para el caso medio tenemos que la búsqueda dentro del árbol se va a ir dividendo en dos, reduciendo así la n en dos:

De esta forma, resulta la siguiente formula donde i es el numero de divisiones que se van haciendo en el árbol:

Sabemos que el peor caso es log2n, de esta forma es el máximo de operaciones básicas que podemos tener, es decir, cuantas divisiones hacemos, esto nos sirve para obtener la siguiente sumatoria:

Sabiendo que todos los casos son igual de probables tenemos la siguiente función de complejidad temporal para el caso medio: