Metodología de trabajo en arreglos ordenados: Búsqueda

Búsqueda binaria

La búsqueda binaria la utilizaremos cuando el arreglo o vector se encuentra **ORDENADO**. Debemos aprovechar esta forma de distribución de los datos, por que si hacemos la búsqueda de un valor de manera secuencial se nos hará "ineficiente", pero con este método en el peor caso hará N comparaciones para encontrar el dato buscado. *Esto reduce el tiempo de búsqueda de manera considerable, disminuyendo exponencialmente el número de iteraciones innecesarias que haríamos de forma lineal.*

La estrategia es **"partir"** a la mitad el arreglo, analizar si el elemento del medio es el valor que buscamos, de no serlo puede darse que sea MENOR y se descarta la mitad derecha, o será MAYOR y descartamos la mitad izquierda.

Paso a paso haremos una y otra vez este mismo proceso redefiniendo límite derecho o izquierdo de un subarreglo que resulta de la aplicación del mismo criterio. El algoritmo finaliza cuando lo encuentra o cuando no se puede continuar redefiniendo un subarreglo (el índice inicial supera al índice final).

Ej:

La función retorna 1 si lo encontró o en su defecto 0.

Parámetros:

validos: representa cantidad de valores cargados en el arreglo.

X: es el dato que deseamos buscar, podemos trabajar con cualquier tipo de dato (en este caso es entero)

arreglo: es nuestro conjunto de datos, puede ser de cualquier tipo de dato primitivo.

Variables locales: (serán todas de tipo entero ya que sólo guardan **índices**)

medio: suma del primer y último índice y división entera por 2

enc: variable que utilizaremos para dar aviso en la búsqueda, se utiliza como booleana donde 1 es verdadero y 0 es falso

pri: índice inicial donde desarrollo la búsqueda

ult: último índice donde desarrollo la búsqueda

```
1   int busquedaBinaria(int arreglo[], int validos, int X)
2   {
3       int medio, pri, ult, enc;
4       enc = 0; ///Asumo que no lo encontré
5       pri = 0; ///primer índice será el 0
6       ult = validos-1;  ///inicializo con el último índice válido
    de mi arreglo
7       medio = (pri + ult) / 2;. ///Partimos el arreglo a la mitad

8       while ( (pri < ult) && (X != arreglo[medio]) ) ///si
    todavía índice pri no se cruzó con ult y no encontré X
9       {
10          if (X < arreglo[medio] ) ///el valor buscado es menor
11          {
12              ult = medio - 1;///descarto la mitad derecha del
    arreglo
13          }
14          else ///el valor buscado es mayor
15          {
16              pri = medio + 1; ///descarto la mitad izquierda
17          }
18          medio=(pri+ult) / 2;  ///vuelvo a partir a la mitad
19      }
20      if (X == arreglo[medio]) ///si lo encontré cambio enc a
    "verdadero"
21          enc = 1;
22      return enc;
23  }
```

Podríamos hacer una mejora de esta función y en vez de retornar un 1 o 0 para saber si lo encontramos, directamente retornar el índice dónde se encuentra dicho valor.