**Explicar porque existe la complejidad logarítmica para la búsqueda binaria**

Se tiene una lista de n elementos, ordenado de izquierda a derecha

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 |  | 8 | … |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 10000000000000000 |
|  |  |  |  |  |  |  |  |  |  |  | n/2 |  |  |  |  |  |  |  |  | n |

1

Se debe realizar las siguientes operaciones (n: Longitud de lista)

1. Si n = 1 , se compara el valor de la posición 1 y termina la búsqueda
2. Caso contrario :
   1. Encontrar el posición central (n/2)
   2. Comparar si el valor en dicha posición con el valor buscado
      1. Si es menor : Repetir para la porción de la izquierda ( tamaño =n/2)
      2. Si es mayor : Repetir para la porción de la derecha (tamaño =n/2)

**El peor caso: no existe el valor buscado**

C operaciones para comprobar que n =1 y comparar el valor buscado

F operaciones para verificar el tamaño de lista, encontrar la posición central y elegir porción (derecha o izquierda)

C y F son constantes

T(n/2): es las operaciones que toma evaluar la búsqueda con la mitad de elementos ya que se aplica la búsqueda solo a la porción derecha o de la izquierda (descartamos la mitad de los elementos)

Aplicamos la búsqueda a T(n/2) que representa la mitad de la lista y se obtiene

Aplicamos la búsqueda a T(n/4) y se obtiene

Durante cada iteración la longitud de la lista evaluada se reduce a la mitad.

En una iteración k la longitud de la lista se reduce a 1 y la búsqueda termina

K es la cantidad de veces que aplique la búsqueda en el peor de los casos, despejamos k

La cantidad de operaciones de la búsqueda en el peor de los casos

Por lo tanto complejidad de la búsqueda seria

**Explicar la complejidad exponencial para el algoritmo de fibonacci de doble recursividad**

**Considerar que**

Reducimos la ecuación a

Dado que

El número de operaciones se duplica respecto a la anterior

En k iteraciones n-k =1 siendo k la iteración donde se calcula Fibonacci(1) que solo tiene un número de operaciones constante

k=n - 1

La cantidad de operaciones

Por lo tanto complejidad de la búsqueda seria