

Ejercicio 4

¿Cómo podría modificar **Búsqueda Binaria** (algoritmo 1.4) para eliminar trabajo innecesario si se tiene la certeza de que K está en el arreglo? Dibuja un árbol de decisión para el algoritmo modificado con $n = 7$. Efectúe análisis de comportamiento promedio y de peor caso. (Para el promedio, puede suponerse que $n = 2^k - 1$ para alguna k .)

Análisis

Árbol de Decisión (n=7)

Para $n = 7$, los índices son $\{0, 1, 2, 3, 4, 5, 6\}$.

El algoritmo divide siempre el rango. En esta versión modificada, el algoritmo siempre baja hasta que el intervalo es de tamaño 1 (las hojas del árbol).

*Nota: En el gráfico, los nodos representan el índice **mid** calculado.*

Representación textual del Árbol (Traza del algoritmo):

El algoritmo siempre compara con **mid**. Si $arr[mid] < K$, va a la derecha (R). Si $arr[mid] \geq K$, va a la izquierda (L).

- **Raíz:** Compara índice 3.
 - **(Izquierda - Indices 0-2):** Compara índice 1.
 - **(Izq - Ind 0):** Hoja Final (Índice 0).
 - **(Der - Ind 2):** Hoja Final (Índice 2).
 - **(Derecha - Indices 4-6):** Compara índice 5.
 - **(Izq - Ind 4):** Hoja Final (Índice 4).
 - **(Der - Ind 6):** Hoja Final (Índice 6).

Observación clave: A diferencia de la búsqueda binaria estándar que puede detenerse en la raíz (si busca el 40), este algoritmo modificado baja hasta que $low == high$. Si busca el 40 (índice 3), hará: $3 \rightarrow izq$ (incluye 3) $\rightarrow \dots \rightarrow$ finaliza en 3.

Análisis Matemático

Aquí está la explicación paso a paso de las operaciones y el comportamiento.

A. Extracción de Datos y Definiciones

1. **Tamaño del problema (n):** El ejercicio define que $n = 2^k - 1$ para algún entero k .
 - Esto es importante porque $2^k - 1$ representa un **Árbol Binario Lleno** (Full Binary Tree). Todos los niveles están completos.
 - Ejemplo $n = 7$: $7 = 2^3 - 1$, por lo tanto $k = 3$.
2. **Operación Básica:** La comparación $arr[mid] < K$.
3. **Objetivo:** Reducir comparaciones.

- Algoritmo Estándar: 2 comparaciones por iteración ($<$, $>$).
- Algoritmo Modificado: 1 comparación por iteración.

B. Análisis de Peor Caso (T_{worst})

En el algoritmo modificado, el bucle $while(low < high)$ **no se detiene** si encuentra el elemento a la mitad. Continúa reduciendo el rango hasta que low es igual a $high$.

1. **Iteraciones:** Como dividimos el arreglo en 2 en cada paso, el número de veces que podemos dividir n hasta llegar a 1 es el logaritmo en base 2.
2. **La fórmula:** Dado que $n = 2^k - 1$, el número de elementos es tal que la altura del árbol es exactamente k .
 - Matemáticamente: $n \approx 2^k \Rightarrow k \approx \log_2(n)$.
3. Exactamente: La altura es $\lceil \log_2(n + 1) \rceil$
4. **Costo:** Hacemos **1 comparación** por nivel.
5. **Conclusión:** El algoritmo *siempre* recorre todo el camino desde la raíz hasta el nivel más bajo (hoja) para asegurar que $low == high$.

$$T_{worst}(n) = k = \lceil \log_2(n + 1) \rceil \text{ comparaciones.}$$

Para $n = 7$, $T_{worst} = 3$ comparaciones.