Capítulo 7 DIVIDE Y VENCERÁS

Nombre: Jean Carlos Iñiguez

* **BÚSQUEDA BINARIA:**

**Objetivo:**

Dado un arreglo ordenado T[1 .. n] en orden no decreciente y un elemento x, se desea encontrar el índice i tal que:

T[i - 1] < x ≤ T[i]

**Convención:**  
Se asume que T[0] = -∞ y T[n+1] = +∞ (aunque estos valores no están en el arreglo, ayudan en la definición formal del problema).

Enfoque clásico: Búsqueda Secuencial

**Código:**

función secuencial(T[1 .. n], x)

para i ← 1 hasta n hacer

si T[i] ≥ x entonces

devolver i

devolver n + 1

**Análisis:**

* Peor caso: O(n)
* Mejor caso: O(1)
* Promedio: (n + 1)/2 comparaciones → también O(n)
* **Búsqueda Binaria:** mejora con reducción a mitades

En lugar de revisar uno por uno, se compara con el elemento del medio:

1. Sea k = ⌊n / 2⌋
2. Si x ≤ T[k], buscar en la mitad izquierda T[1 .. k]
3. Si x > T[k], buscar en la mitad derecha T[k+1 .. n]

Este proceso se repite recursivamente, dividiendo el tamaño en dos en cada paso.

**Idea principal:**

Es una técnica de reducción, ya que reduce el problema de tamaño n a uno de tamaño n/2.

**Complejidad temporal:**

* Como en cada paso se reduce el problema a la mitad, su complejidad es logarítmica:
  + Peor caso: O(log n)
  + Mejor caso: O(1)
  + Promedio: O(log n)

**Aplicación práctica:**

Este es el algoritmo que usamos de forma intuitiva al buscar una palabra en un diccionario: abrimos por la mitad, comparamos, y decidimos si mirar a la izquierda o a la derecha.

* **BÚSQUEDA DE LA MEDIANA:**

**¿Qué es la mediana?**

Dada una matriz de enteros T[1 .. n], la mediana es el elemento que estaría en la posición central si ordenáramos el arreglo.

* Si n es impar y los elementos son distintos, es el elemento que tiene la misma cantidad de números menores y mayores que él.
* Ejemplo:  
  T = [3, 1, 4, 1, 5, 9, 2, 6, 5]  
  Ordenado → [1, 1, 2, 3, 4, 5, 5, 6, 9] → Mediana = 4

Formalmente, la mediana es el ⌊n / 2⌋-ésimo elemento más pequeño.

**Problema de selección**

Generalización de la búsqueda de la mediana:

* **Input:** Un arreglo T[1 .. n] y un entero s tal que 1 ≤ s ≤ n
* **Objetivo:** Encontrar el s-ésimo elemento más pequeño

Este problema se conoce como el problema de selección.

**Algoritmo ingenuo**

1. Ordenar el arreglo T → O(n log n)
2. Devolver el elemento en la posición s

Este enfoque funciona, pero no es óptimo. Se busca algo más rápido que ordenar todo el arreglo.

**Solución eficiente:** Algoritmo de selección basado en la mediana

Supongamos que tenemos una función llamada mediana(T) que nos devuelve la mediana de T. Podemos usar esa mediana como pivote, al estilo del algoritmo quicksort.

**Algoritmo** pivoteBis(T, p; k, l)

* Reorganiza T en 3 secciones:
  + T[1 .. k] → Elementos menores que p
  + T[k+1 .. l-1] → Elementos iguales a p
  + T[l .. n] → Elementos mayores que p

**Lógica del algoritmo de selección:**

1. Usar p = mediana(T) como pivote.
2. Particionar el arreglo con pivoteBis.
3. Comparar s con las posiciones k y l:
   * Si k < s < l → El elemento deseado es p (hemos terminado)
   * Si s ≤ k → Buscar en la submatriz T[1 .. k]
   * Si s ≥ l → Buscar en la submatriz T[l .. n] y ajustar el índice como s ← s - l + 1

Esto garantiza que en cada paso se descarta al menos la mitad del arreglo → reducción efectiva, como en la búsqueda binaria.

**Complejidad esperada:**

Con una buena elección del pivote (por ejemplo, usando la "mediana de medianas"), el algoritmo puede alcanzar O(n) en tiempo en el peor caso, lo cual es óptimo.

**Implementación iterativa**

* En lugar de usar recursión, se puede usar un bucle con dos índices i y j, manteniendo la propiedad:

T[1 .. i-1] < T[i .. j] < T[j+1 .. n]

El s-ésimo elemento más pequeño siempre estará dentro del segmento [i .. j].