Divide y Vencerás

Problema 6 - Vectores unimodales

A. Herrera, A. Moya, I. Sevillano, J.L. Suarez

11 de abril de 2015

Introdución

- En esta presentación se proporciona una solución para el problem 6.
- El código, los resultados de las ejecuciones, las gráficas y los pdf asociados se puede encontrar en GitHub.

Explicación del problema

Definición de vector unimodal

Un vector v de números con n componentes, todas distintas, se dice unimofal si existe un índice p (que no es ni el primero ni el último) tal que:

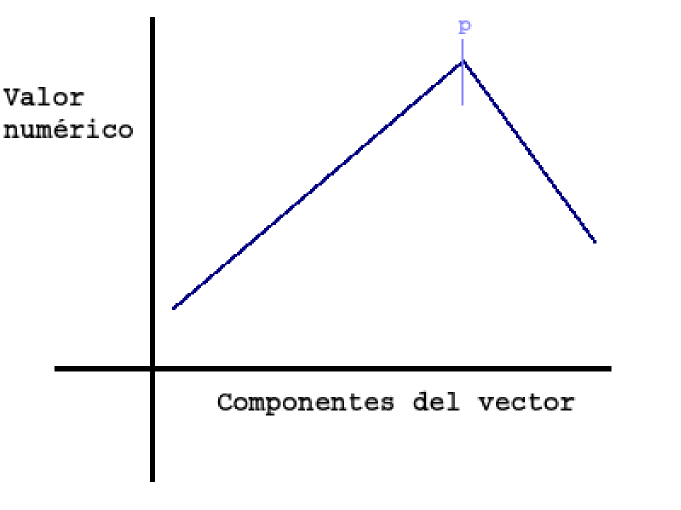
- A la izquierda de p los números están ordenados de forma creciente.
- A la derecha de p están ordenados de forma decreciente.

Matemáticamente:

- $\forall i, j \leq p$ con $i < j \Rightarrow v[i] < v[j]$
- $\forall i, j \geq p$ con $i < j \Rightarrow v[i] > v[j]$

Explicación del problema

Ejemplo de vector unimodal



Explicación del problema

Otro ejemplo de vector unimodal

Un ejemplo numérico de vector unimodal es el siguiente:

$$\mathrm{v} = [1\ 3\ 5\ 7\ 8\ 6\ 4]$$

En este caso $p = 4$ y $v[p] = 8$.

Explicación del problema

Problema

Diseñar un algoritmo que permita determinar p.

Algoritmos

Se proponen dos algoritmos:

- Algoritmo trivial de eficiencia $\theta(n)$.
- Algoritmo recursivo de eficiencia $\theta(\log n)$.

Observaciones

Proposición

La componente p es el máximo de v.

Demostración

Como v es unimodal:

- $v[p] \geq v[i] \ \forall i \leq p$
- $v[p] \geq v[i] \ \forall i \geq p$

Entonces: $v[p] \geq v[i] \ \forall i$

$\blacksquare$

Observaciones

Proposición

Un vector unimodal solo tiene un máximo local (en p)

Demostración

Sea $i \neq p$. Veamos que no es un máximo local:

- Si $i < p$, entonces $v[i] < v[i+1]$.
- Si $i > p$, entonces $v[i-1] > v[i]$.

∎

Observaciones

A. Herrera, A. Moya, I. Sevillano, J.L. Suarez

Proposición

Una componente i es p si, y solo si, $v[i-1] < v[i] > v[i+1]$.

Demostración

$i = p$ si, y solo si, i es un máximo local (solo hay uno que es p).
Esto se cumple si, y solo si, $v[i-1] < v[i] > v[i+1]$.

■

Algoritmo trivial

Búsqueda lineal del máximo local.

```
# Pseudocódigo del algoritmo trivial.
# v es el vector unimodal con n componentes.
for i in range(1, n-1):
    if v[i] > v[i+1]:
        p = i
        break
```

Algoritmo trivial

A. Herrera, A. Moya, I. Sevillano, J.L. Suarez

Ejemplo

$$\mathrm{v} = [1\ 3\ 5\ 7\ 8\ 6\ 4]$$

Se empieza en $v[1] = 3$, que no es mímimo local:

$$\mathrm{v} = [\text{-}\ \text{-}\ 5\ 7\ 8\ 6\ 4]$$

$v[2] = 5$ tampoco es máximo local. . .

$$\mathrm{v} = [\text{-}\ \text{-}\ \text{-}\ 7\ 8\ 6\ 4]$$

$v[3] = 7$ tampoco es máximo local. . .

$$\mathrm{v} = [\text{-}\ \text{-}\ \text{-}\ \text{-}\ 8\ 6\ 4]$$

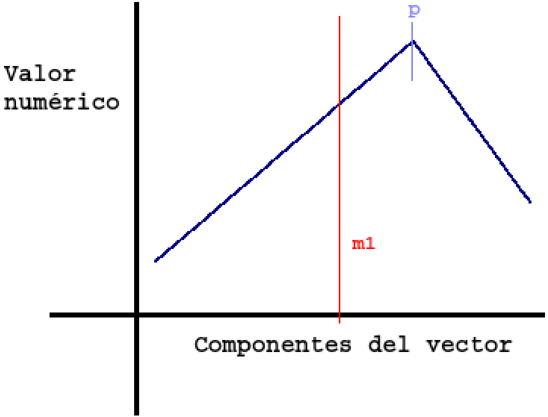
¡$v[4] = 8$ sí es máximo local! **Solución:** $p = 4$

Algoritmo recursivo

A. Herrera, A. Moya, I. Sevillano, J.L. Suarez

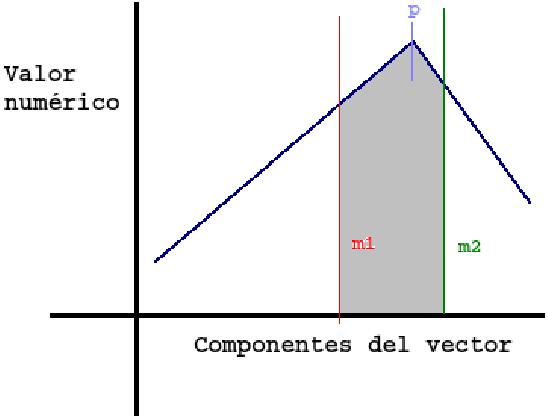
- Queremos deshacernos de la búsqueda lineal.
- Problema similar: Búsqueda en un vector. Si está ordenado: **Búsqueda binaria**.
- Empezamos por la componente mitad. Si es un máximo local hemos terminado. **¿Y si no es máximo local?**

Algoritmo recursivo



$$v[m1-1] < v[m1] < v[m1+1] \Rightarrow m2 < p$$

Algoritmo recursivo



$$v[m2-1] > v[m2] > v[m2+1] \Rightarrow p < m2$$

Algoritmo recursivo

A. Herrera, A. Moya, I. Sevillano, J.L. Suarez

```
# Parametros:
#  - v       : Vector unimodal.
#  - inicio  : Posición de inicio.
#  - final   : Posición siguiente a la última.
def obtenerP(v, inicio, final):
  mitad = (inicio + final) / 2
  if mitad > inicio:
    if v[mitad-1] < v[mitad] and
       v[mitad] > v[mitad+1]:
      return mitad
    else if v[mitad-1] < v[mitad]:
      return obtenerP(v, mitad+1, final)
    else:
      return obtenerP(v, inicio, mitad)
  else:
    return inicio
```

Algoritmo recursivo

Ejemplo

$$v = [1\ 3\ 5\ 7\ 8\ 6\ 4]$$

Se empieza en $v[3] = 7$, que es menor que $v[4] = 8$.

$$v = [-\ -\ -\ -\ 8\ 6\ 4]$$

Se empieza en $v[5] = 6$, que es menor que $v[4] = 8$.

$$v = [-\ -\ -\ -\ 8\ -\ -]$$

¡$v[4] = 8$ es máximo local! **Solución:** $p = 4$

- **Menos iteraciones que el trivial.**

Algoritmo recursivo

Proposición

El algoritmo recursivo es $\theta(\log n)$ en el peor caso.

Demostración

En el peor caso, $T(n) = T(\frac{n}{2}) + \theta(1)$. Tomando $n = 2^k$:

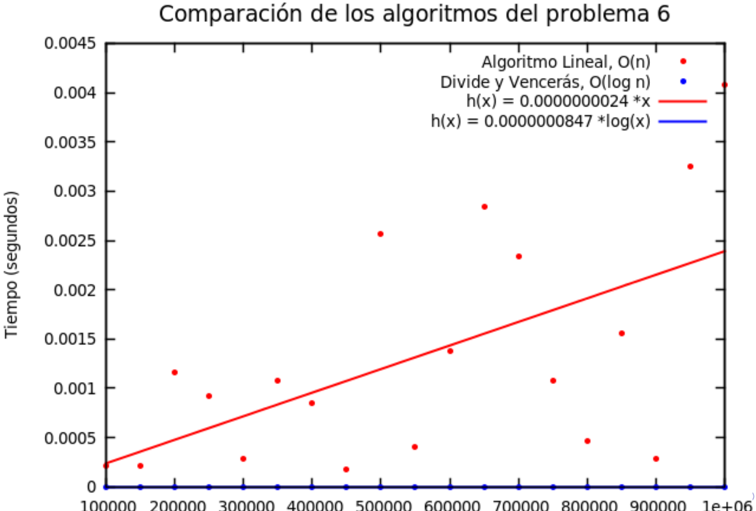
$$T(n) = T(2^k) = T(2^{k-1}) + \theta(1) = \ldots = T(1) + (k-1)\theta(1)$$

$$T(n) \in \theta(k) = \theta(\log_2 n)$$

$\blacksquare$

Análisis empírico. Eficiciencia híbrida

- Una única ejecución por tamaño del vector



Análisis empírico. Eficiciencia híbrida

Problema

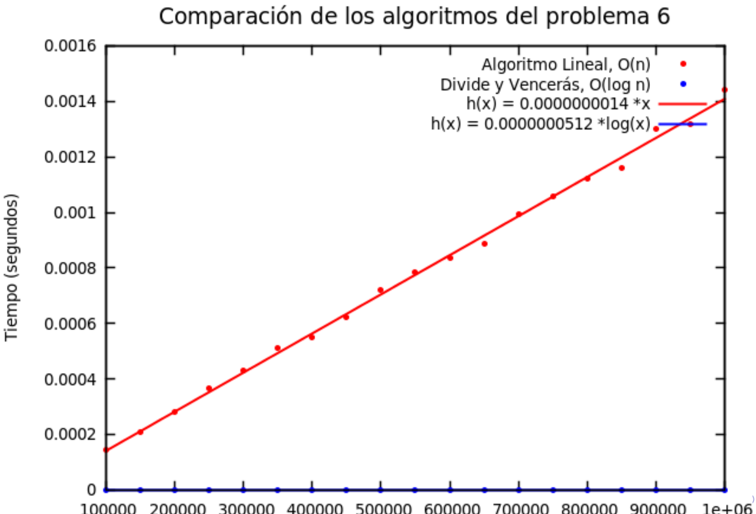
En el mejor caso ambos algoritmos son $\theta(1)$.

Solución

Hacer la media de varias ejecuciones.

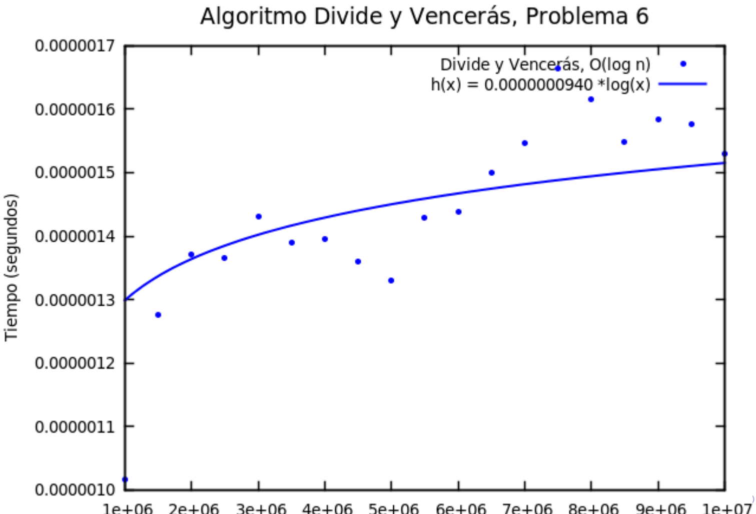
Análisis empírico. Eficiciencia híbrida

- Media de 1000 ejecuciones



Análisis empírico. Eficiciencia híbrida

- Media de 1000 ejecuciones



Fin de la presentación

A. Herrera, A. Moya, I. Sevillano, J.L. Suarez

¡Gracias por su atención!