Prueba de oposición - Algoritmos 2016

Brian Bokser

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

2 de noviembre de 2016

Introducción

- Materia : *Algoritmos y Estructuras de Datos II*

- Práctica : *Dividir y Conquistar*

Contexto

- Tienen claros los conceptos de complejidad, vistos en la teórica.
- Conocen el teorema maestro, y pueden usar alguno de sus casos para determinar la complejidad de una recursión.
- Los alumnos ya tuvieron la teórica de Divide and Conquer.
- Conocen las ideas clásicas del tema, ahora tienen que ejercitarlas.

Enunciado

Enunciado

Suponga que se tiene un método potencia que, dada un matriz cuadrada A de orden 4×4 y un número n, computa la matriz A^n.

Dada una matriz cuadrada A de este orden, y un número natural n que es potencia de 2, desarrollar, utilizando la técnica de dividir y conquistar, y el método potencia, un algoritmo que permita calcular:

$$A^1 + A^2 + \ldots + A^n$$

Calcule el número de veces que el algoritmo propuesto aplica el método potencia. Si no es estrictamente menor que O(n), resuelva el ejercicio nuevamente.

Motivación

- Este ejercicio me parece interesante porque:
  1. Mostramos que Divide and Conquer es un tema amplio y hay mas que “arboles y arreglos”.
  2. Del ejercicio surgen otras ideas y ejercicios relacionados.
- En el ejercicio vamos a trabajar:
  1. Formas de pensar los ejercicios Divide and Conquer.
  2. Conceptos de Complejidad.
  3. Teorema Maestro.

¡Al ataque!

Idea:

¡Al ataque!

Idea: algoritmo ingenuo.

¡Al ataque!

Idea: algoritmo ingenuo. Problema: $O(n)$ llamados a potencia.

¡Al ataque!

Idea: algoritmo ingenuo. Problema: $O(n)$ llamados a potencia.

¿Como resolver un problema de Divide and Conquer?

¡Al ataque!

Idea: algoritmo ingenuo. Problema: $O(n)$ llamados a potencia.

¿Como resolver un problema de Divide and Conquer?

Idea importante

¡Queremos algun subproblema que nos sirva para dar una solución al problema original!

Buscando el subproblema ganador

Queremos encontrar alguna propiedad que nos ayude a dar esta formulación en base a subproblemas.

Buscando el subproblema ganador

Queremos encontrar alguna propiedad que nos ayude a dar esta formulación en base a subproblemas.

Idea:

$$(A^{n/2+1} + A^{n/2+2} + \ldots + A^{n}) = (A^{1} + A^{2} + \ldots + A^{n/2}) \times A^{n/2}$$

Buscando el subproblema ganador

Queremos encontrar alguna propiedad que nos ayude a dar esta formulación en base a subproblemas.

Idea:

$$(A^{n/2+1} + A^{n/2+2} + \ldots + A^{n}) = (A^{1} + A^{2} + \ldots + A^{n/2}) \times A^{n/2}$$

$$A^{1} + \ldots + A^{n} = (A^{1} + \ldots + A^{n/2}) + (A^{1} + \ldots + A^{n/2}) \times A^{n/2}$$

Buscando el subproblema ganador

Queremos encontrar alguna propiedad que nos ayude a dar esta formulación en base a subproblemas.

Idea:

$$(A^{n/2+1} + A^{n/2+2} + \ldots + A^{n}) = (A^{1} + A^{2} + \ldots + A^{n/2}) \times A^{n/2}$$

$$A^{1} + \ldots + A^{n} = \color{red}{(A^{1} + \ldots + A^{n/2})} + (A^{1} + \ldots + A^{n/2}) \times A^{n/2}$$

¡Subproblema!

¿Estamos resolviendo el problema?

Importante: estamos haciendo menos operaciones de potencia. ¿Pero cuantas?

¿Estamos resolviendo el problema?

Importante: estamos haciendo menos operaciones de potencia. ¿Pero cuantas?

Escribamos un pseudocódigo y despues medimos la complejidad en términos de la función potencia.

```
function SUMA_GEOMETRICA(A, n)
    if n == 1 then
        return A                                        ▷ A = A^1
    end if
    subsuma ← suma_geometrica(A, n/2)                   ▷ (A^1 + . . . + A^{n/2})
    pot ← potencia(A, n/2)                              ▷ A^{n/2}
    return subsuma + subsuma × pot
end function
```

Pseudocódigo

function SUMA_GEOMETRICA(A, n)
    if n == 1 **then**
        return A ▷ A = A^1
    end if
    $subsuma \leftarrow suma\_geometrica(A, n/2)$ ▷ $(A^1 + \ldots + A^{n/2})$
    $pot \leftarrow potencia(A, n/2)$ ▷ $A^{n/2}$
    return $subsuma + subsuma \times pot$
end function

¿Complejidad (en términos de llamados a potencia)?:

Pseudocódigo

```
function SUMA_GEOMETRICA(A, n)
    if n == 1 then
        return A                                    ▷ A = A^1
    end if
    subsuma ← suma_geometrica(A, n/2)               ▷ (A^1 + . . . + A^{n/2})
    pot ← potencia(A, n/2)                          ▷ A^{n/2}
    return subsuma + subsuma × pot
end function
```

¿Complejidad (en términos de llamados a potencia)?:

$$P(n) = \begin{cases} 0 & n = 1 \\ P(\frac{n}{2}) + 1 & n > 1 \end{cases}$$

$$P(n) = \begin{cases} 0 & n = 1 \\ P(\frac{n}{2}) + 1 & n > 1 \end{cases}$$

¿Podemos usar el Teorema Maestro? Veamos sus casos:

El Teorema Maestro

$$P(n) = \begin{cases} 0 & n = 1 \\ P(\frac{n}{2}) + 1 & n > 1 \end{cases}$$

¿Podemos usar el Teorema Maestro? Veamos sus casos:

$P(n) = a \times P(n/b) + f(n)$

1. $f(n) \in \mathcal{O}(\text{n}^{\,log_b a - \epsilon})$ para algun $\epsilon > 0 \implies P(n) = \Theta(\text{n}^{\,log_b a})$
2. $f(n) \in \Theta(\text{n}^{\,log_b a}) \implies P(n) = \Theta(\text{n}^{\,log_b a} \lg \text{n})$
3. $f(n) \in \Omega(\text{n}^{\,log_b a + \epsilon})$ para algun $\epsilon > 0$, $af(n/b) \leq cf(n)$ para algún c < 1 y todo n $\geq n_0 \implies P(n) = \Theta(f(n))$

El Teorema Maestro

$$P(n) = \begin{cases} 0 & n = 1 \\ P(\frac{n}{2}) + 1 & n > 1 \end{cases}$$

¿Podemos usar el Teorema Maestro? Veamos sus casos:

$P(n) = a \times P(n/b) + f(n)$

2. $f(n) \in \Theta(\mathrm{n}^{\,log_b a}) \implies P(n) = \Theta(\mathrm{n}^{\,log_b a}\ \lg \mathrm{n})$

b = 2, a = 1, $log_b a = 0$, f(n) = 1 $\in \Theta(1) \implies$
Estamos en el segundo caso. $P(n) = \Theta(\lg \mathrm{n})$

Ejercicio Adicional

Si tomamos la multiplicación de matrices como $\mathcal{O}(1)$
¿Como es la complejidad en función de operaciones elementales?

Ejercicio Adicional

Si tomamos la multiplicación de matrices como $\mathcal{O}(1)$
¿Como es la complejidad en función de operaciones elementales?

¿Como varía segun las implementaciones de potencia?

¿Preguntas?

¡Muchas gracias!