Prueba de oposición - Algoritmos 2016

Brian Bokser

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

17 de octubre de 2016

Contenidos

Introducción

Materia : *Algoritmos y Estructuras de Datos II*

Práctica : *Séptima práctica - Dividir y Conquistar*

Contexto

Los alumnos ya tuvieron la teórica de Divide and Conquer.

Conocen las ideas clásicas, ahora tienen que ejercitarlas.

Conocen el teorema maestro, y pueden usar alguno de sus casos para determinar la complejidad de una recursión.

Presentación

- Primera parte
  - Especificación con Tipos Abstractos de Datos
  - Demostración de propiedades
  - Diseño: invariante de representación y función de abstracción
- Segunda Parte
  - Complejidad Algorítmica
  - Diseño: elección de estructuras de datos
  - Ordenamiento
  - **Dividir y Conquistar**

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 orden 4×4 y un número natural n que es potencia de 2 (i.e., $n = 2 \times k$ para algun $k \geq 1$) , 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

¡Al ataque!

Primer idea:

¡Al ataque!

Primer idea: Buscar la fórmula de serie geométrica.

¡Al ataque!

Primer idea: Buscar la fórmula de serie geométrica. No parece servir para matrices

¡Al ataque!

Primer idea: Buscar la fórmula de serie geométrica. No parece servir para matrices

Segunda idea:

¡Al ataque!

Primer idea: Buscar la fórmula de serie geométrica. No parece servir para matrices

Segunda idea: algoritmo ingenuo.

¡Al ataque!

Primer idea: Buscar la fórmula de serie geométrica. No parece servir para matrices

Segunda idea: algoritmo ingenuo. $O(n)$ llamados a potencia.

¡Al ataque!

Primer idea: Buscar la fórmula de serie geométrica. No parece servir para matrices

Segunda idea: algoritmo ingenuo. $O(n)$ llamados a potencia.

Buscamos algo sublineal...

¡Al ataque!

Primer idea: Buscar la fórmula de serie geométrica. No parece servir para matrices

Segunda idea: algoritmo ingenuo. $O(n)$ llamados a potencia.

Buscamos algo sublineal...

¿Como resolver un problema de Divide and Conquer?

¡Al ataque!

Primer idea: Buscar la fórmula de serie geométrica. No parece servir para matrices

Segunda idea: algoritmo ingenuo. $O(n)$ llamados a potencia.

Buscamos algo sublineal...

¿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:

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

Buscando el subproblema ganador

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

Idea:

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

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

Buscando el subproblema ganador

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

Idea:

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

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

¡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.

Pseudocódigo

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

Pseudocódigo

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

¿Complejidad (en función de operaciones potencia)?:

Pseudocódigo

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

¿Complejidad (en función de operaciones potencia)?:

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

El Teorema Maestro

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

¿Podemos usar el Teorema Maestro? Veamos sus casos:

El Teorema Maestro

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

¿Podemos usar el Teorema Maestro? Veamos sus casos:

$P(n) = aT(n/b) + f(n)$

1. $f(n) = \mathcal{O}(\text{n}^{\log_b a - \epsilon})$ para algun $\epsilon > 0 \implies P(n) = \Theta(\text{n}^{\log_b a})$
2. $f(n) = \Theta(\text{n}^{\log_b a}) \implies P(n) = \Theta(\text{n}^{\log_b a} \text{ lg n})$
3. $f(n) = \Omega(\text{n}^{\log_b a + \epsilon})$ 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

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

¿Podemos usar el Teorema Maestro? Veamos sus casos:

$P(n) = aT(n/b) + f(n)$

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

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

Ejercicio Adicional

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

Ejercicio Adicional

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

¿Como varía segun las implementaciones de potencia?

Ejercicio Adicional

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

¿Como varía segun las implementaciones de potencia?

Para pensar:

Enunciado

Ejercicio 3 de la práctica: Encuentre un algoritmo para calcular a^b en $\mathcal{O}(lg\ b)$. Piense cómo reutilizar los resultados ya calculados

¿Preguntas?