Divisão e Conquista

Fernando Lobo

Algoritmos e Estrutura de Dados

Divisão e Conquista

- Nesta aula vamos ver mais alguns exemplos de divisão e conquista.
- A técnica é simples, mas a sua aplicação requer prática e algum engenho.
- É útil vermos mais exemplos.

Divisão e Conquista

- Consiste em 3 passos:
  - Dividir o problema em um ou mais subproblemas.
  - Conquistar (i.e., resolver) cada subproblema recursivamente.
  - Combinar os resultados dos subproblemas para a obter a solução para o problema original.

Divisão e Conquista (cont.)

Tempo de execução é dado quase sempre por uma recorrência do tipo: $T(n) = D(n) + aT(n/b) + C(n)$.

- $D(n)$ é o tempo gasto a dividir o problema em subproblemas.
- a é o número de subproblemas.
- n/b é o tamanho de cada subproblema.
- $C(n)$ é o tempo gasto a combinar as soluções dos subproblemas.

Divisão e Conquista (cont.)

- Normalmente agregamos os tempos $D(n) + C(n)$ correspondentes ao tempo gasto pelo algoritmo em trabalho não recursivo, chamando a esse tempo $f(n)$.
- Obtemos assim a recorrência $T(n) = aT(n/b) + f(n)$, que é a recorrência do Teorema Mestre.

Exemplo 1: Busca Binária

- Dividir: verificar o elemento que está no meio de array.
- Conquistar: resolve recursivamente o mesmo problema para um dos subarrays (esq. ou dta.)
- Combinar: não faz nada.
- Recorrência é $T(n) = T(n/2) + \Theta(1)$.
- Utilizando o Teorema Mestre, $a = 1$, $b = 2$, $f(n) = c$ (em que c é uma constante). $n^{\log_b a} = n^{\log_2 1} = n^0 = 1 = \Theta(f(n))$. Estamos no caso 2 do Teorema Mestre. A complexidade é $T(n) = \Theta(\lg n)$.

Exemplo 2: Potência de um número

- Problema: calcular a^n , sendo n um número inteiro.
- Algoritmo naive: $\underbrace{a * a * \ldots * a}_{n \text{ vezes}} = \Theta(n)$.
- É possível fazer melhor assimptoticamente?
- Vamos ver que sim.

Exemplo 2: Potência de um número (cont.)

- Exemplo:
  $2^{16} = 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 = 65536$.
- Requer $16 - 1 = 15$ multiplicações com o algoritmos naive.
- Outro método: $2^{16} = (2^8)^2 = ((2^4)^2)^2 = (((2^2)^2)^2)^2 = 65536$.
  Requer apenas 4 multiplicações.
- No caso geral,

$$a^n = \begin{cases} a^{n/2} * a^{n/2} & \text{, se n par} \\ a^{(n-1)/2} * a^{(n-1)/2} * a & \text{, se n ímpar} \end{cases}$$

- Recorrência igual à da busca binária:

$$T(n) = T(n/2) + \Theta(1) = \Theta(\lg n)$$

.

Pseudocódigo

```
POWER(a, n)
    if n == 0
        return 1
    if n == 1
        return a
    if EVEN(n)
        tmp = POWER(a, n/2)
        return tmp * tmp
    else
        tmp = POWER(a, (n − 1)/2)
        return tmp * tmp * a
```


Exemplo 3: Multiplicação de inteiros

- Multiplicar 2 números inteiros, cada qual com n dígitos.
- Imaginemos que n era muito grande podendo "rebentar" com o limite máximo de inteiro na nossa linguagem de programação.
- Numa abordagem POO teríamos de implementar a nossa própria classe para "inteiros grandes" e incluir métodos para somar, multiplicar, etc.
  - Representação para um "inteiro grande": um array de dígitos.

Exemplo 3: Multiplicação de inteiros

- Solução óbvia: algoritmo da escola primária.
- Exemplo:

```
      2374
    * 1853
  ---------
      7122
     11870
    18992
+  2374
-----------
   4399022
```

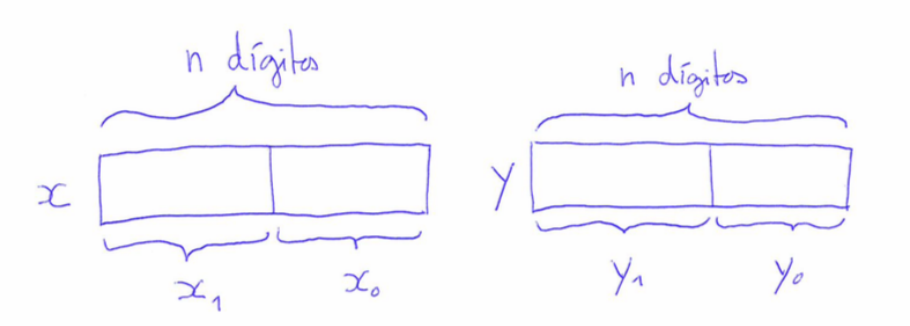

Exemplo 3: Multiplicação de inteiros

- No nosso exemplo $n = 4$.
- Tempo de execução do algoritmo em função de n: $\Theta(n^2)$.
- É possível fazer melhor?
- Vamos tentar com divisão e conquista.

Exemplo 3: Divisão e Conquista (versão 1)

Ideia base:

- Dividir x ao meio, ficando com x_1 dígitos na parte esquerda (os mais significativos) e x_0 dígitos na parte direita (os menos significativos). Fazer a mesma coisa para y.



- $x = x_1 10^{n/2} + x_0$.
- $y = y_1 10^{n/2} + y_0$.

Exemplo 3: Divisão e Conquista (versão 1)

$$\begin{aligned} xy &= (x_1 10^{n/2} + x_0)(y_1 10^{n/2} + y_0) \\ &= x_1 y_1 10^n + (x_1 y_0 + x_0 y_1) 10^{n/2} + x_0 y_0 \end{aligned}$$

- Voltando ao nosso exemplo:
  - $x = 2374 \longrightarrow x_1 = 23,\ x_0 = 74$
  - $y = 1853 \longrightarrow y_1 = 18,\ y_0 = 53$

$$\begin{aligned} 2374 * 1853 &= (23 * 10^2 + 74)(18 * 10^2 + 53) \\ &= 23 * 18 * 10^4 + (23 * 53 + 74 * 18) * 10^2 + 74 * 53 \\ &= \ldots \\ &= 4399022 \end{aligned}$$

Exemplo 3: Divisão e Conquista (versão 1)

- As operações 10^n e $10^{n/2}$ não requerem multiplicações. Basta meter zeros à direita...
- Multiplicação de 2 inteiros de n dígitos dá origem a 4 multiplicações de 2 inteiros de $n/2$ dígitos + um nº constante de adições de inteiros de n dígitos.
- A adição requer $\Theta(n)$.
- Ou seja, $T(n) = 4T(n/2) + \Theta(n)$.
- Aplicando o Teorema Mestre: $a = 4$, $b = 2$.
  $n^{\log_b a} = n^{\log_2 4} = n^2$
- Estamos no caso 1: $T(n) = \Theta(n^2)$

Exemplo 3: Divisão e Conquista (versão 1)

- **Conclusão:** o algoritmo de divisão e conquista não é melhor que o algoritmo da escola primária. É apenas um algoritmo complicado que tem a mesma complexidade!

Exemplo 3: Divisão e Conquista (versão 2)

- Será possível usar apenas 3 chamadas recursivas, cada qual de tamanho $n/2$?
- Se fosse possível obteríamos:

$$T(n) = 3T(n/2) + \Theta(n) = \Theta(n^{\log_2 3}) \approx \Theta(n^{1.59})$$

- Vamos ver que tal é possível com uma observação engenhosa.

Exemplo 3: Divisão e Conquista (versão 2)

- Queremos obter:
  $xy = x_1y_1 10^n + (x_1y_0 + x_0y_1)10^{n/2} + x_0y_0$
- Truque:
  $(x_1 + x_0)(y_1 + y_0) = x_1y_1 + x_1y_0 + x_0y_1 + x_0y_0$
- A multiplicação acima tem as 4 multiplicações que pretendemos.
- Se também calcularmos x_1y_1 e x_0y_0 recursivamente, poderemos calcular xy com apenas 3 chamadas recursivas. Estas:
  - $(x_1 + x_0)(y_1 + y_0)$ // mult1
  - x_1y_1 // mult2
  - x_0y_0 // mult3

$xy = \text{mult}_2 * 10^n + (\text{mult}_1 - \text{mult}_2 - \text{mult}_3) * 10^{n/2} + \text{mult}_3$

Exemplo 3: Pseudocódigo

RECURSIVE-MULTIPLY(x, y, n)

if $n < 3$

return MULTIPLY(x, y, n) // base case

else

Dividir x ao meio e obter x_1 e x_0

Dividir y ao meio e obter y_1 e y_0

$mult1 =$ RECURSIVE-MULTIPLY$(x_1 + x_0, y_1 + y_0)$

$mult2 =$ RECURSIVE-MULTIPLY(x_1, y_1)

$mult3 =$ RECURSIVE-MULTIPLY(x_0, y_0)

return $mult2 * 10^n + (mult1 - mult2 - mult3) * 10^{n/2} + mult3$

Algumas notas

- Caso base poderia ter outra constante $\neq 3$.
- MULTIPLY chamado no caso base poderia ser o algoritmo da "escola primária".
- O pseudocódigo não está totalmente correcto. Deixei assim para se perceber a essência do algoritmo.
- Questões a resolver:
  - O que fazer se n ímpar?
  - Como generalizar para o caso de x e y terem um número diferente de dígitos?
- Podem pensar e tentar implementar isto como exercício.

Em resumo

- Algoritmo óbvio da escola primária requer $\Theta(n^2)$.
- Algoritmo de D&C requer $\Theta(n^{\log_2 3}) \approx \Theta(n^{1.59})$.

Mais exemplos

- Há muitos mais exemplos em que a técnica de D&C consegue melhorar a complexidade do algoritmo óbvio (brute-force).
  - Algoritmo de Strassen para multiplicação de matrizes.
    De $\Theta(n^3)$ para $\Theta(n^{\log_2 7}) \approx \Theta(n^{2.81})$.
  - Dado n pontos no plano, obter o par de pontos mais próximo.
    De $\Theta(n^2)$ para $\Theta(n \lg n)$.
  - ...
- Quase sempre à custa de uma solução engenhosa, como acabámos de ver para a multiplicação de inteiros.