MC458 — Projeto e Análise de Algoritmos I

C.C. de Souza C.N. da Silva O. Lee P.J. de Rezende

1º. Semestre de 2018

Divisão e Conquista

Projeto de Algoritmos por Divisão e Conquista

- **Dividir para conquistar:** uma tática de guerra aplicada ao projeto de algoritmos.
- Um algoritmo de divisão e conquista é aquele que resolve o problema desejado combinando as soluções parciais de (um ou mais) subproblemas, obtidas recursivamente.
- É mais um paradigma de projeto de algoritmos baseado no princípio da indução.
- Informalmente, podemos dizer que o paradigma incremental representa o projeto de algoritmos por indução fraca, enquanto o paradigma de divisão e conquista representa o projeto por indução forte.
- É natural, portanto, demonstrar a corretude de algoritmos de divisão e conquista por indução.

Algoritmo Genérico

DivisaoConquista(x)

▷ **Entrada:** A instância x
▷ **Saída:** Solução y do problema em questão para x
1. **se** x é suficientemente pequeno **então**
   ▷ *Solucao*(x) algoritmo para pequenas instâncias
2. **retorne** *Solucao*(x)
3. **senão**
   ▷ divisão
4. decomponha x em instâncias menores $x_1, x_2, \cdots, x_k$
5. **para** i **de** 1 **até** k **faça** $y_i := \mathit{DivisaoConquista}(x_i)$
   ▷ conquista
6. combine as soluções y_i para obter a solução y de x.
7. **retorne**(y)

Projeto por Divisão e Conquista - Exemplo 1 Exponenciação

Problema:

Calcular a^n, para todo real a e inteiro $n \geq 0$.

Primeira solução, por indução fraca:

- **Caso base:** $n = 0$; $a^0 = 1$.
- **Hipótese de indução:** *Suponha que, para qualquer inteiro $n > 0$ e real a, sei calcular a^{n-1}.*
- **Passo da indução:** Queremos provar que conseguimos calcular a^n, para $n > 0$. Por hipótese de indução, sei calcular a^{n-1}. Então, calculo a^n multiplicando a^{n-1} por a.

Exemplo 1 - Solução 1 - Algoritmo

Exponenciacao(a, n)

▷ **Entrada:** A base a e o expoente n.
▷ **Saída:** O valor de a^n.
1. **se** $n = 0$ **então**
2. **retorne**(1) {caso base}
3. **senão**
4. $an' := Exponenciacao(a, n-1)$
5. $an := an' * a$
6. **retorne**(an)

Exemplo 1 - Solução 1 - Complexidade

Seja $T(n)$ o número de operações executadas pelo algoritmo para calcular a^n.

Então a relação de recorrência deste algoritmo é:

$$T(n) = \begin{cases} c_1, & n = 0 \\ T(n-1) + c_2, & n > 0, \end{cases}$$

onde c_1 e c_2 representam, respectivamente, o tempo (constante) executado na atribuição da base e multiplicação do passo.

Sabemos que:

$$T(n) = c_1 + \sum_{i=1}^{n} c_2 = c_1 + nc_2 = \Theta(n).$$

Este algoritmo é linear no tamanho da entrada ?

Exemplo 1 - Solução 2 - Divisão e Conquista

Vamos agora projetar um algoritmo para o problema usando indução forte de forma a obter um algoritmo de divisão e conquista.

Segunda solução, por indução forte:

- **Caso base:** $n = 0$; $a^0 = 1$.
- **Hipótese de indução:** *Suponha que, para qualquer inteiro $n > 0$ e real a, sei calcular a^k, para todo $k < n$.*
- **Passo da indução:** Queremos provar que conseguimos calcular a^n, para $n > 0$. Por hipótese de indução sei calcular $a^{\lfloor \frac{n}{2} \rfloor}$. Então, calculo a^n da seguinte forma:

$$a^n = \begin{cases} \left(a^{\lfloor \frac{n}{2} \rfloor}\right)^2, & \text{se } n \text{ par;} \\ a \cdot \left(a^{\lfloor \frac{n}{2} \rfloor}\right)^2, & \text{se } n \text{ ímpar.} \end{cases}$$

Exemplo 1 - Solução 2 - Algoritmo

ExponenciacaoDC(a, n)

▷ **Entrada:** A base a e o expoente n.
▷ **Saída:** O valor de a^n.

```
1.  se n = 0 então
2.      retorne(1) {caso base}
3.  senão
        ▷ divisão
4.      an' := ExponenciacaoDC(a, n div 2)
        ▷ conquista
5.      an := an' * an'
6.      se (n mod 2) = 1
7.          an := an * a
8.  retorne(an)
```


Exemplo 1 - Solução 2 - Complexidade

- Seja $T(n)$ o número de operações executadas pelo algoritmo de divisão e conquista para calcular a^n.
- Então a relação de recorrência deste algoritmo é:

$$T(n) = \begin{cases} c_1, & n = 0 \\ T(\lfloor \frac{n}{2} \rfloor) + c_2, & n > 0, \end{cases}$$

- Sabemos que $T(n) \in \Theta(\log n)$.

Projeto por Divisão e Conquista - Exemplo 2 Busca Binária

Problema:

Dado um vetor ordenado A com n números reais e um real x, determinar a posição $1 \leq i \leq n$ tal que $A[i] = x$, ou que não existe tal i.

- O projeto de um algoritmo para este problema usando indução fraca, nos leva a um algoritmo incremental de complexidade de pior caso $\Theta(n)$. Pense em como seria a indução !
- Se utilizarmos indução forte para projetar o algoritmo, podemos obter um algoritmo de divisão e conquista que nos leva ao algoritmo de busca binária. Pense na indução !
- Como o vetor está ordenado, conseguimos determinar, com apenas uma comparação, que *metade* das posições do vetor não pode conter o valor x.

Exemplo 2 - Algoritmo

BuscaBinaria(A, e, d, x)

▷ **Entrada:** Vetor A, delimitadores e e d do subvetor e x.
▷ **Saída:** Índice $1 \leq i \leq n$ tal que $A[i] = x$ ou $i = 0$.

```
 1.  se e = d então se A[e] = x então retorne(e)
 2.          senão retorne(0)
 3.  senão
 4.      i := (e + d) div 2
 5.      se A[i] = x retorne(i)
 6.      senão se A[i] > x
 7.          i := BuscaBinaria(A, e, i - 1, x)
 8.          senão {A[i] < x}
 9.          i := BuscaBinaria(A, i + 1, d, x)
10.  retorne(i)
```

Exemplo 2 - Complexidade

- O número de operações $T(n)$ executadas na busca binária no pior caso é:

$$T(n) = \begin{cases} c_1, & n = 1 \\ T(\lceil \frac{n}{2} \rceil) + c_2, & n > 1, \end{cases}$$

- Já vimos que $T(n) \in \Theta(\log n)$.
- O algoritmo de busca binária (divisão e conquista) tem complexidade de pior caso $\Theta(\log n)$, que é assintoticamente melhor que o algoritmo de busca linear (incremental).
- E se o vetor não estivesse ordenado, qual paradigma nos levaria a um algoritmo assintoticamente melhor ?

Projeto por Divisão e Conquista - Exemplo 3 - Máximo e Mínimo

Problema:

Dado um conjunto S de $n \geq 2$ números reais, determinar o maior e o menor elemento de S.

- Um algoritmo incremental (projetado através de uma indução fraca) para esse problema faz $2n - 3$ comparações: fazemos uma comparação no caso base e duas no passo.
- Será que um algoritmo de divisão e conquista (projetado através de uma indução forte) seria melhor ?
- Um possível algoritmo de divisão e conquista seria:
  - Divida S em dois subconjuntos de mesmo tamanho S_1 e S_2 e solucione os subproblemas.
  - O máximo de S é o máximo dos máximos de S_1 e S_2 e o mínimo de S é o mínimo dos mínimos de S_1 e S_2.

Exemplo 3 - Complexidade

- Qual o número de comparações $T(n)$ efetuado por este algoritmo?

$$T(n) = \begin{cases} 1, & n = 2 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + 2, & n > 2, \end{cases}$$

- Supondo que n é uma potência de 2, temos:

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

- Neste caso, podemos provar que $T(n) = \frac{3}{2}n - 2$ usando o método da substituição (indução !).

Exemplo 3 - Complexidade

- **Caso Base:** $T(2) = 1 = 3 - 2$.
- **Hipótese de Indução:** Suponha, para $n = 2^{k-1}$, $k \geq 2$, que $T(n) = \frac{3}{2}n - 2$.
- **Passo de Indução:** Queremos provar para $n = 2^k$, $k \geq 2$, que $T(n) = \frac{3}{2}n - 2$.

$$\begin{aligned} T(n) &= 2T(\tfrac{n}{2}) + 2 \\ &= 2(\tfrac{3}{4}n - 2) + 2 \text{ (por h. i.)} \\ &= \tfrac{3}{2}n - 2. \end{aligned}$$

- Exercício: $T(n) = \frac{3}{2}n - 2$ quando n não é potência de 2? Por quê?

Exemplo 3 - Complexidade

- Assintoticamente, os dois algoritmos para este problema são equivalentes, ambos $\Theta(n)$.
- No entanto, o algoritmo de divisão e conquista permite que menos comparações sejam feitas. A estrutura hierárquica de comparações no retorno da recursão evita comparações desnecessárias.

Obs: mas daria ainda para projetarmos um algoritmo incremental de complexidade dada pela relação de recorrência $T(n) = T(n-2) + 3$, com $T(2) = 1$ e $T(1) = 0$, cuja solução é $T(n) = 3n/2 - 2$. Como seria uma prova por indução de onde adviria tal algoritmo?