
RECORRÊNCIAS - Aula 3 (Cormen, cap. 4)

Na aula anterior, nós concluímos que $h(t)$ limita superiormente o custo de uma chamada mergesort(V,i,f), onde

$$
\begin{aligned}
t(i,f) &= f - i + 1, && \text{se } i <= f,\\
t(i,f) &= 0, && \text{em caso contrário,}
\end{aligned}
$$

$$
\begin{aligned}
h(t) &= a + h(\lceil t/2\rceil) + h(\lfloor t/2\rfloor) + b*t, && \text{se } t >= k,\\
h(t) &= d, && \text{se } 0 <= t < k,
\end{aligned}
$$

e a,b,d,k são naturais positivos grandes o suficiente.

1. EXERCÍCIO: mostre que $h(t) >= 0$, $\forall t >= 0$.

O LEMA ABAIXO TRÁS UMA NOVIDADE TÉCNICA: como a primeira das equações de recorrência de "h", acima, somente se aplica a valores de "t" maiores ou iguais a "k", nós tratamos todos os valores de "m" a "k-1" na base indutiva, o que pode ser feito de uma maneira bastante simples: basta exigir que "c" seja maior ou igual a h(i) (ou algum termo relacionado), para todo "i" de "m" a "k-1" (ou algum intervalo relacionado). Observe que, como "k" é um natural positivo, então o conjunto dos naturais menores que "k" é finito, e portanto a exigência acima sobre "c" é perfeitamente plausível.

2. LEMA: $h(t) = O(n^2)$.

==

PROVA:

Temos que mostrar que existem um real "c" e um natural "m" positivos tais que, $\forall n >= m$, $0 <= h(n) <= c*n^2$.

Sejam então:

$$
\begin{aligned}
m &= 1,\\
m' &= \text{máx}\{m, k-1\},\\
c &= \text{máx}\ \{\ b,\ a + \text{máx}\ \{\ h(i) : m <= i <= m'\ \}\ \}.
\end{aligned}
$$

Nós mostraremos agora que, $\forall n >= m$, $0 <= h(n) <= c*n^2$. Pelo exercício anterior, segue que $0 <= h(n)$ para todo $n >= m$. Resta, então, mostrar que $h(n) <= c*n^2$ para todo $n >= m$, e nós o faremos por indução forte em "n":

* TESE: $\forall n >= m$, $h(n) <= c*n^2 - a$.

* BASE ($m <= n <= m'$): Temos:

$$
\begin{aligned}
h(n) &<= c - a && \text{---> pois } c >= a + h(n).\\
&<= c*n^2 - a, \text{ CQD.} && \text{---> pois } n^2 >= 1, \text{ já que } n >= m >= 1.
\end{aligned}
$$

* HIPÓTESE DE INDUÇÃO (H.I.): $\forall\ m <= l < n$, $h(l) <= c*l^2 - a$.

* PASSO ($n > m'$): Temos:

$$
\begin{aligned}
h(n) &= a + h(\lceil n/2\rceil) + h(\lfloor n/2\rfloor) + b*n && \text{---> pois } n >= k\ (n > m' >= k-1).\\
&<= a + c*\lceil n/2\rceil^2 - a + c*\lfloor n/2\rfloor^2 - a + b*n && \text{---> H.I. } (1 <= \lceil n/2\rceil,\lfloor n/2\rfloor\\
& && \text{---> } < n, \text{ pois } n > m' >= 1).\\
&= c*(\lceil n/2\rceil^2 + \lfloor n/2\rfloor^2) + b*n - a\\
&= c*(n^2 - 2*\lceil n/2\rceil*\lfloor n/2\rfloor) + b*n - a && \text{---> pois } x^2 + y^2 = (x+y)^2 - 2xy.\\
&= c*n^2 - a + b*n - 2*c*\lceil n/2\rceil*\lfloor n/2\rfloor.\\
&<= c*n^2 - a, && \text{---> veja abaixo.}
\end{aligned}
$$

onde a última inequação segue porque b*n - 2*c*⌈n/2⌉*⌊n/2⌋ <= 0, o que pode ser mostrado em cada um dos casos possíveis, que são os seguintes:

* CASO 1: n = 2j, para algum natural j > 0. Nesse caso, temos:

```
2*c*⌈n/2⌉*⌊n/2⌋  = 2*c*⌈j⌉*⌊j⌋
                 = 2*c*j*j
                >= c*2j     ---> pois j >= 1
                >= b*2j     ---> pois c >= b
                 = b*n, CQD.
```

* CASO 2: n = 2j + 1, para algum natural j > 0. Nesse caso, temos:

```
2*c*⌈n/2⌉*⌊n/2⌋  = 2*c*⌈j + 1/2⌉*⌊j + 1/2⌋
                 = 2*c*(j+1)*j
                 = c*(2j + 2)*j
                >= c*(2j + 2)     ---> pois j >= 1.
                >= c*(2j + 1)
                >= b*(2j + 1)     ---> pois c >= b.
                 = b*n, CQD.
```

Pela indução acima, segue portanto que, ∀n >= m, h(n) <= $c*n^2$ - a <= $c*n^2$, CQD.

==

RESOLVENDO PEQUENOS ENTRAVES NA INDUÇÃO - MUDAR A BASE DO LOGARITMO

Considere o seguinte algoritmo:

```
===========================================================================
Algoritmo: primeiro_curioso
Entrada: um vetor V[1..n] e índices "i" e "f" tais que 1 <= i <= f <= n.
Saída: um elemento do vetor.
---------------------------------------------------------------------------
1. SE i = f
2. | RETORNE V[i].
3. m := i + ⌊(f-i)/2⌋
4. RETORNE primeiro_curioso(V,i,m).
===========================================================================
```

Observe que f(t) limita superiormente o tempo de uma chamada primeiro_curioso(V,i,f), onde

```
t(i,f) = f -i +1, se i <= f,
t(i,f) = 0,       em caso contrário,

f(t) = a + f(⌈t/2⌉), se t > 1,
f(t) = b,            se t = 1,
```

e "a" e "b" são naturais positivos grandes o suficiente.

3. EXERCÍCIO: mostre que, ∀n >= 1, f(n) >= 0.

4. EXERCÍCIO: mostre que, para todo real x >= 0, ⌈x⌉ <= ⌊x⌋ + 1.

Neste ponto, nós gostaríamos de provar que f(n) = O(lg n), utilizando indução no cerne do argumento. No passo indutivo, porém, logo após aplicar a hipótese de indução, obtém-se f(n) <= a + c*lg(⌈n/2⌉), e se deseja concluir que f(n) <= c*lg(n). Pela nossa experiência anterior, isso seria fácil se, ao invés de lg(⌈n/2⌉), tivéssemos lg(⌊n/2⌋), pois lg(⌊n/2⌋) <= lg(n/2). Observe, porém, que lg(⌈n/2⌉) <= lg(⌊n/2⌋ + 1) <= lg(n/2 + 1), e que, sendo "n" grande o suficiente, é possível argumentar que 1 <= n/10, ou 1 <= n/50, etc. Logo, nós concluímos que lg(n/2 + 1) <= lg(n/k) = lg(n) - lg(k), onde 1 < k < 2. O restante do argumento é então imediato.

Observe que a argumentação acima exige que "n" seja suficientemente grande no passo indutivo, mas, como vimos na prova do lema 2 acima, isso não é um problema: basta aumentar o conjunto dos valores de "n" tratados na base da indução.

```
5. LEMA: f(n) = O(lg n).
   ----
================================================================================
PROVA:

Temos que mostrar que existem um real "c" e um natural "m" positivos tais que,
∀n >= m, 0 <= f(n) <= c*lg n.

Observe primeiramente que, como "lg" é crescente e

  1 = 51/51 < 100/51 < 100/50 = 2,

então 0 = lg(1) < lg(100/51) < lg 2 = 1.

Sejam então m = 2 e c = máx{ a/lg(100/51), máx { f(i) : 2 <= i <= 99 } }.

Nós mostraremos então que, ∀n >= m, 0 <= f(n) <= c*lg n. O exercício 3 implica
que, ∀n >= m, 0 <= f(n). Logo, resta apenas mostrar que, ∀n >= m,
f(n) <= c*lg(n), e nós o faremos por indução forte:

  * TESE: ∀ n >= m, f(n) <= c*lg n.

  * BASE (2 <= n <= 99): Temos:

      f(n) <= c              ---> pois c >= máx { f(i) : 2 <= i <= 99 } >= f(n).
           <= c*lg n, CQD.   ---> pois 1 <= lg n, já que 2 <= n.

  * HIPÓTESE DE INDUÇÃO (H.I.): ∀ 2 <= l < n, f(l) <= c*lg l.

  * PASSO (n >= 100): Temos:

      f(n)  = a + f(⌈n/2⌉)              ---> pois n > 1
           <= a + c*lg(⌈n/2⌉)           ---> pela H.I. (2 <= 50 <= ⌈n/2⌉ < n).
           <= a + c*lg(⌊n/2⌋ + 1)       ---> pois "lg" é crescente.
           <= a + c*lg(n/2 + 1)         ---> pois "lg" é crescente.
           <= a + c*lg(n/2 + n/100)     ---> pois "lg" é crescente e n >= 100.
            = a + c*lg(n*51/100)
            = a + c*lg(n/(100/51))
            = a + c*(lg(n) - lg(100/51))
            = c*lg(n) + a - c*lg(100/51)
           <= c*lg(n), CQD.             ---> pois c*lg(100/51) >= a.

================================================================================
```

Uma outra maneira de provar o resultado acima é mudar a base do logaritmo na indução:

```
6. LEMA: f(n) = O(log n), "log n" denotando o logaritmo de "n" na base 100/51.
   ----
================================================================================
PROVA:
```

Temos que mostrar que existem um real "c" e um natural "m" positivos tais que, ∀n >= m, 0 <= f(n) <= c*log n.

Sejam então m = 2 e c = máx{ a, máx { f(i) : 2 <= i <= 99 } }.

Nós mostraremos então que, ∀n >= m, 0 <= f(n) <= c*log n. O exercício 3 implica que, ∀n >= m, 0 <= f(n). Logo, resta apenas mostrar que, ∀n >= m, f(n) <= c*log(n), e nós o faremos por indução forte:

```
  * TESE: ∀ n >= m, f(n) <= c*log n.

  * BASE (2 <= n <= 99): Temos:

      f(n) <= c              ---> pois c >= máx { f(i) : 2 <= i <= 99 } >= f(n).
           <= c*log n, CQD.  ---> pois 1 = log(100/51) <= log n,
                                  já que 100/51 < 2 <= n.

  * HIPÓTESE DE INDUÇÃO (H.I.): ∀ 2 <= l < n, f(l) <= c*log l.

  * PASSO (n >= 100): Temos:

      f(n)  = a + f(⌈n/2⌉)              ---> pois n > 1
           <= a + c*log(⌈n/2⌉)          ---> pela H.I. (2 <= 50 <= ⌈n/2⌉ < n).
           <= a + c*log(⌊n/2⌋ + 1)      ---> pois "log" é crescente.
           <= a + c*log(n/2 + 1)        ---> pois "log" é crescente.
           <= a + c*log(n/2 + n/100)    ---> pois "log" é crescente e n >= 100.
            = a + c*log(n*51/100)
            = a + c*log(n/(100/51))
            = a + c*(log(n) - log(100/51))
            = a + c*(log(n) - 1)
            = c*log(n) + a - c
           <= c*log(n), CQD.            ---> pois c >= a.

================================================================================
```

Observe que o lema acima prova apenas que f(n) = O(log n), "log n" denotando o logaritmo de "n" na base 100/51, ao passo que o lema 5 prova que f(n) = O(lg n) (base 2). Entretanto, é fácil ver que uma coisa implica a outra:

```
7. EXERCÍCIO: usando o lema 6, mostre que f(n) = O(lg n).
   ---------  (DICA: log_b(a) = log_c(a) / log_c(b).)
```

8. EXERCÍCIO: PROVE OU REFUTE: log_a(n) = θ(log_b(n)), ∀ a,b > 1 reais.

 TEOREMA MESTRE
----

9. TEOREMA (MESTRE): sejam a >= 1 e b > 1 números reais, "f" uma função de
   ---------------- naturais em reais, e "t" uma função de naturais em reais
   definida por uma equação de recorrência da forma

      t(n) = a*t(⌈n/b⌉) + f(n)
      ou
      t(n) = a*t(⌊n/b⌋) + f(n).

   Nesse caso, sendo L = log_b(a), "t" pode ser assintoticamente limitada assim:

   1. Se f(n) = O(n^(L-d)), para algum real d > 0, então t(n) = θ(n^L).

   2. Se   f(n) = θ(n^L * (lg n)^k   ), para algum natural k >= 0,
      então t(n) = θ(n^L * (lg n)^(k+1)).

   3. Se f(n) = Ω(n^(L+d)), para algum real d > 0, e se existe um real c < 1
      tal que a*f(n/b) <= c*f(n) para todo "n" grande o suficiente,
      então t(n) = θ(f(n)).

Observação: o caso 2 acima é uma generalização daquele que aparece na segunda
edição do livro-texto da disciplina. Essa versão mais geral aparece no capítulo
5 deste livro: http://ww3.algorithmdesign.net/ . O capítulo em questão é
publicamente acessível neste endereço:

   http://ww3.algorithmdesign.net/sample/ch05-patterns.pdf .