
Certifique-se de que você tentou fazer por um tempo antes de ver as próximas páginas!

2. Suponha que $T(1) = 1$ e $T(2) = 1$. Resolva as seguintes recorrências com notação $O$ com o resultado mais justo possível (quando não especificado, use o método mais adequado):

(a) $T(n) = T(\frac{n}{3}) + n^2$ (método de iteração)

$$T(n) = T\left(\frac{n}{3}\right) + n^2$$
$$T(n) = T\left(\frac{n}{9}\right) + \left(\frac{n}{3}\right)^2 + n^2 = T\left(\frac{n}{9}\right) + \frac{1}{9}n^2 + n^2$$
$$T(n) = T\left(\frac{n}{27}\right) + \left(\frac{n}{9}\right)^2 + \frac{1}{9}n^2 + n^2 = T\left(\frac{n}{27}\right) + \frac{1}{81}n^2 + \frac{1}{9}n^2 + n^2$$
$$\vdots$$
$$T(n) = T\left(\frac{n}{3^i}\right) + \sum_{j=0}^{i-1}\left(\frac{1}{9^j}\cdot n^2\right)$$

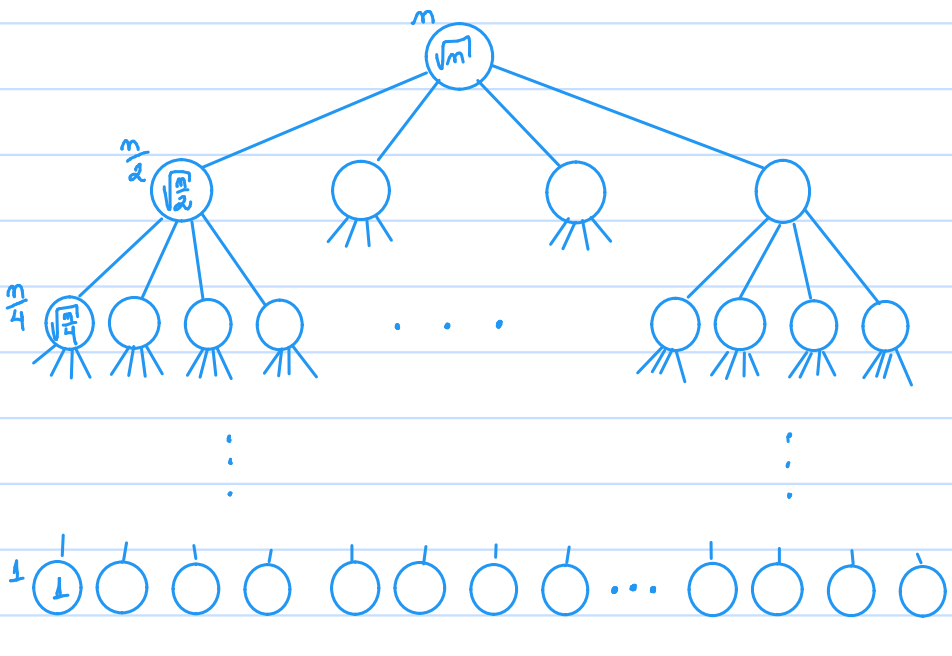
Como $\frac{n}{3^i} = 1$ quando $i = \log_3 n$, então

$$T(n) = T(1) + \sum_{j=0}^{\log_3 n}\left(\frac{1}{9^j}\cdot n^2\right) = 1 + n^2\left(\frac{1 - \left(\frac{1}{9}\right)^{(\log_3 n + 1)}}{1 - \frac{1}{9}}\right)$$
$$= 1 + \frac{9}{8}n^2\left(1 - \frac{1}{9}\cdot n^{\log_3 1/9}\right)$$
$$= 1 + \frac{9}{8}n^2\left(1 - \frac{1}{9}\cdot n^{(\log_3 1 - \log_3 9)}\right)$$
$$= 1 + \frac{9}{8}n^2\left(1 - \frac{1}{9}n^{0-2}\right)$$
$$= 1 + \frac{9}{8}n^2\left(1 - \frac{1}{9}\cdot\frac{1}{n^2}\right) = 1 + \frac{9}{8}n^2 - \frac{1}{8} = \Theta(n^2)$$

2. Suponha que $T(1) = 1$ e $T(2) = 1$. Resolva as seguintes recorrências com notação $O$ com o resultado mais justo possível (quando não especificado, use o método mais adequado):

(e) $T(n) = 4T(\frac{n}{2}) + \sqrt{n}$ (árvore de recursão e método mestre)

ÁRVORE DE RECURSÃO




| NÍVEL | QTD PROBL. | TAMANH. PROB. | TEMPO PROBL. |
|---|---|---|---|
| 0 | 1 | $n$ | $\sqrt{n}$ |
| 1 | 4 | $\frac{n}{2}$ | $\sqrt{\frac{n}{2}}$ |
| 2 | 16 | $\frac{n}{4}$ | $\sqrt{\frac{n}{4}}$ |
| $i$ | $4^i$ | $\frac{n}{2^i}$ | $\sqrt{\frac{n}{2^i}}$ |
| $l$ | $4^l$ | 1 | $\sqrt{\frac{n}{2^l}}$ |

Note que $\frac{n}{2^i} = 1$ quando $i = \log_2 n$. Então $l = \log_2 n$.
Assim, o tempo gasto na árvore é

$$T(n) = \sum_{i=0}^{\log_2 n} 4^i \cdot \sqrt{\frac{n}{2^i}} = \sum_{i=0}^{\log_2 n} \frac{4^i}{\sqrt{2}^i}\sqrt{n} = \sqrt{n}\sum_{i=0}^{\log_2 n}\left(\frac{4}{\sqrt{2}}\right)^i$$

$$= \sqrt{n}\left(\frac{\left(\frac{4}{\sqrt{2}}\right)^{\log_2 n} - 1}{\left(\frac{4}{\sqrt{2}}\right) - 1}\right) = \sqrt{n}\left(\frac{n^{\log_2(4/\sqrt{2})} - 1}{\frac{4}{\sqrt{2}} - \frac{\sqrt{2}}{\sqrt{2}}}\right)$$

$$= \frac{\sqrt{2}}{4-\sqrt{2}}\sqrt{n}\left(n^{\log_2 4 - \log_2 \sqrt{2}} - 1\right)$$

$$= \frac{\sqrt{2}}{4-\sqrt{2}}\sqrt{n}\left(n^{\log_2 4 - \log_2 2^{1/2}}\right) - \frac{\sqrt{2}}{4-\sqrt{2}}\sqrt{n}$$

$$= \frac{\sqrt{2}}{4-\sqrt{2}} n^{1/2} \cdot n^{2-\frac{1}{2}} - \frac{\sqrt{2}}{4-\sqrt{2}}\sqrt{n} = \frac{\sqrt{2}}{4-\sqrt{2}} n^2 - \frac{\sqrt{2}}{4-\sqrt{2}}\sqrt{n}$$

que é $\Theta(n^2)$. Logo, $T(n) = 4T(\frac{n}{2}) + \sqrt{n}$ é $\Theta(n^2)$.

(e) $T(n) = 4T(\frac{n}{2}) + \sqrt{n}$ (árvore de recursão e método mestre)

MÉTODO MESTRE

Aqui temos $a=4$, $b=2$ e $k=\frac{1}{2}$.
Como $b^k = \sqrt{2}$ e $4 > \sqrt{2}$, caímos no caso 1, que nos diz que $T(n) = \Theta(n^{\log_b a})$ ou seja, $T(n) = \Theta(n^{\log_2 4}) = \Theta(n^2)$.

note como o resultado do mestre confirma que minhos contas anteriores deram certo.

2. Suponha que $T(1) = 1$ e $T(2) = 1$. Resolva as seguintes recorrências com notação $O$ com o resultado mais justo possível (quando não especificado, use o método mais adequado):

(l) $T(n) = 2T(\frac{3n}{4}) + n$

melhor opção

Usando o método mestre: $a = 2$, $b = \frac{4}{3}$ e $k = 1$.
Como $b^k = \frac{4}{3}$ e $2 > \frac{4}{3}$, vale o caso 1 do teorema e portanto $T(n)$ é $\Theta(n^{\log_{4/3} 2})$.

Usando o método da iteração:

$$T(n) = 2T\left(\frac{3n}{4}\right) + n$$
$$T(n) = 2\left[2T\left(\frac{9n}{16}\right) + \frac{3n}{4}\right] + n = 4T\left(\frac{9n}{16}\right) + \frac{3n}{2} + n$$
$$T(n) = 4\left[2T\left(\frac{27n}{64}\right) + \frac{9n}{16}\right] + \frac{3n}{2} + n = 8T\left(\frac{27n}{64}\right) + \frac{9n}{4} + \frac{3n}{2} + n$$
$$\vdots$$
$$T(n) = 2^i T\left(\left(\frac{3}{4}\right)^i n\right) + \sum_{j=0}^{i-1} \left(\frac{3}{2}\right)^j n$$

Como $\left(\frac{3}{4}\right)^i n = 1$ se $i = \log_{4/3} n$:

$$T(n) = 2^{\log_{4/3} n} + \sum_{j=0}^{\log_{4/3} n} \left(\frac{3}{2}\right)^j n = n^{\log_{4/3} 2} + n\left[\frac{\left(\frac{3}{2}\right)^{(\log_{4/3} n + 1)} - 1}{3/2 - 1}\right]$$
$$= n^{\log_{4/3}^2} + 2n\left(\frac{3}{2} \cdot n^{\log_{4/3} 3/2} - 1\right)$$
$$= n^{\log_{4/3}^2} + 3n^{\log_{4/3} 4/3 + \log_{4/3} 3/2} - 2n$$
$$= n^{\log_{4/3}^2} + 3n^{\log_{4/3} 4/3 \cdot 3/2} - 2n = n^{\log_{4/3}^2} + 3n^{\log_{4/3} 2} - 2n$$
$$= 4n^{\log_{4/3}^2} - 2n = \Theta\left(n^{\log_{4/3}^2}\right).$$

2. Suponha que $T(1) = 1$ e $T(2) = 1$. Resolva as seguintes recorrências com notação $O$ com o resultado mais justo possível (quando não especificado, use o método mais adequado):

(m) $T(n) = T(\frac{2n}{3}) + 2T(\frac{n}{3}) + 1$ (método de substituição)

Para usar substituição, precisamos de um chute.
Seja $T'(n) = 3T'(\frac{2n}{3}) + 1$ e note que $T(n) \leq T'(n)$.
Pelo método mestre, $T'(n) = \Theta(n^{\log_{3/2} 3})$, pois $a=3$, $b=\frac{3}{2}$, $k=0$ e $3 > (\frac{3}{2})^0 = 1$.
Note ainda que $\log_{3/2} 3 = \frac{\log_2 3}{\log_2 3/2} = \frac{\log_2 3}{\log_2 3 - \log_2 2} = \frac{\log_2 3}{\log_2 3 - 1}$.
Como $1 = \log_2 2 < \log_2 3 < \log_2 4 = 2$, $\frac{\log_2 3}{\log_2 3 - 1} < 2$.

Poderia chutar $O(n^2)$ sem problemas

Nosso chute então é que $T(n)$ é $O(n^{3/2})$.
Vamos provar por indução que $T(n) \leq cn^{3/2} + d$, para constantes $c$ e $d$ que acharemos depois.

BASE: $n = 1$. Nesse caso, $T(1) = 1$ e $cn^{3/2} + d = c + d$.
Então a base vale se $c + d \geq 1$.

HIPÓTESE: $T(k) \leq ck^{3/2} + d \quad \forall\ 1 \leq k < n$.

PASSO: Seja $n > 1$ (e potência de 3).
Então $T(n) = T(\frac{2n}{3}) + 2T(\frac{n}{3}) + 1$.
Como $\frac{2n}{3} < n$, por HI vale $T(\frac{2n}{3}) \leq c(\frac{2n}{3})^{3/2} + d$
Como $\frac{n}{3} < n$, por HI vale $T(\frac{n}{3}) \leq c(\frac{n}{3})^{3/2} + d$
Então $T(n) = T(\frac{2n}{3}) + 2T(\frac{n}{3}) + 1$
$\leq c(\frac{2n}{3})^{3/2} + d + 2(c(\frac{n}{3})^{3/2}) + d + 1$
$= (\frac{2}{3})^{3/2} \cdot cn^{3/2} + 2(\frac{1}{3})^{3/2} cn^{3/2} + 2d + 1$
$= ((\frac{2}{3})^{3/2} + 2(\frac{1}{3})^{3/2}) cn^{3/2} + 2d + 1$
E o resultado vai seguir se $((\frac{2}{3})^{3/2} + 2(\frac{1}{3})^{3/2}) \leq 1$ e $2d + 1 \leq d$.

Note que $\left(\frac{2}{3}\right)^{3/2} + 2\left(\frac{1}{3}\right)^{3/2} = \sqrt{\left(\frac{2}{3}\right)^3} + 2\sqrt{\left(\frac{1}{3}\right)^3} = \sqrt{\frac{8}{27}} + 2\sqrt{\frac{1}{27}} = \frac{2\sqrt{2}}{3\sqrt{3}} + \frac{2}{3\sqrt{3}}$

$= \frac{2\sqrt{2}+2}{3\sqrt{3}}$. Se $\frac{2\sqrt{2}+2}{3\sqrt{3}} > 1$, então $2\sqrt{2}+2 > 3\sqrt{3}$ e $(2\sqrt{2}+2)^2 > (3\sqrt{3})^2$,

ou seja, $(2\sqrt{2})^2 + 2\cdot 2\sqrt{2}\cdot 2 + 4 > (3\sqrt{3})^2$, isto é,

$$8 + 8\sqrt{2} + 4 > 27$$
$$8\sqrt{2} > 15$$
$$\sqrt{2} > \frac{15}{8}$$
$$(\sqrt{2})^2 > \left(\frac{15}{8}\right)^2$$
$$2 > \frac{225}{64}$$
$$128 > 225$$

o que é impossível. Então temos que vale $\frac{2\sqrt{2}+2}{3\sqrt{3}} \leq 1$.

Para que $2d+1 \leq d$, basta que $d \leq -1$. Vamos tomar $d = -1$.
Para o caso base valer, $c + d \geq 1$, ou seja, $c - 1 \geq 1$ e $c \geq 2$.

Provamos, então, que $T(n) \leq 2n^{3/2} - 1$, ou seja, $T(n)$ é $O(n^{3/2})$.

CQD

Na próxima página, vamos mostrar um limitante superior mais justo, que é o próprio valor dado pelo mestre.

Vamos tentar provar que $T(n)$ é $O(n^{\log_{3/2}^3})$.
Por indução em $n$, vamos provar que $T(n) \leq c \cdot n^{\log_{3/2}^3} + d$

BASE: $n = 1$. $T(1) = 1$ e $c \cdot 1^{\log_{3/2}^3} + d = c + d$.
Para a base valer, basta que $c + d \geq 1$.

HI: $T(k) \leq c \cdot k^{\log_{3/2}^3} + d \quad \forall\, 1 \leq k < n$.

PASSO: Sabemos que $T(n) = T(\frac{2n}{3}) + 2T(\frac{n}{3}) + 1$
Por HI, $T(\frac{2n}{3}) \leq c \cdot (\frac{2n}{3})^{\log_{3/2}^3} + d$ e $T(\frac{n}{3}) \leq c \cdot (\frac{n}{3})^{\log_{3/2}^3} + d$.
Então

$$T(n) \leq c \cdot (\tfrac{2n}{3})^{\log_{3/2}^3} + d + 2 \cdot c \cdot (\tfrac{n}{3})^{\log_{3/2}^3} + d + 1$$

$$= c \cdot \left(\frac{2}{3}\right)^{\log_{3/2}^3} \cdot n^{\log_{3/2}^3} + 2 \cdot c \cdot \left(\frac{1}{3}\right)^{\log_{3/2}^3} \cdot n^{\log_{3/2}^3} + 2d + 1$$

$$= c n^{\log_{3/2}^3} \left( \left(\frac{2}{3}\right)^{\log_{3/2}^3} + 2 \cdot \left(\frac{1}{3}\right)^{\log_{3/2}^3} \right) + 2d + 1$$

$$\overset{\star}{\leq} c n^{\log_{3/2}^3} \left( \left(\frac{2}{3}\right)^{3} + 2 \cdot \left(\frac{1}{3}\right)^{3} \right) + 2d + 1$$

$$= c n^{\log_{3/2}^3} \left( \frac{8}{27} + \frac{2}{27} \right) + 2d + 1$$

$$\overset{\star\star}{\leq} c n^{\log_{3/2}^3} + 2d + 1$$

$$\overset{\star\star\star}{\leq} c n^{\log_{3/2}^3} + d$$

Onde:
$\star$ vale porque $\log_{3/2}^3 < \log_{3/2}^{\frac{27}{8}} = \log_{3/2}^{(3/2)^3} = 3$

$\star\star$ vale porque $10/27 < 1$

$\star\star\star$ vale quando $2d + 1 \leq d$, o que ocorre se $d \leq -1$.

Para que $c + d \geq 1$, podemos tomar $d = -1$ e $c = 2$.

Assim, provamos que $T(n) \leq 2 \cdot n^{\log_{3/2}^3} - 1$, que é $O(n^{\log_{3/2}^3})$.

QED

2. Suponha que $T(1) = 1$ e $T(2) = 1$. Resolva as seguintes recorrências com notação $O$ com o resultado mais justo possível (quando não especificado, use o método mais adequado):

(r) $T(n) = T(\sqrt{n}) + 1$ (faça uma mudança de variáveis: defina $m = \log n$, escreva uma recorrência $S(m)$ equivalente a $T(n)$, resolva $S(m)$ e use esse resultado para encontrar $T(n)$)

Seja $m = \log n$. Então $n = 2^m$.
Assim, $T(2^m) = T(2^{m/2}) + 1$.

Defina $S(m) = T(2^m)$. Então
$S(m) = S(\frac{m}{2}) + 1$.
Pelo método mestre, $S(m)$ é $\Theta(\log m)$, já que $a=1$, $b=2$, $k=0$ e $1 = a = b^k = 2^0 = 1$.

Voltando as variáveis, $T(n)$ é $\Theta(\log \log n)$.

CQD

3. Seja $A[1..n]$ um vetor qualquer de inteiros de tamanho $n$ e seja $k$ um inteiro qualquer. Resolva de forma recursiva o problema de verificar se existem posições $i$ e $j$ tais que $A[i] + A[j] = k$.

Note que se existe um tal par em $A[1..n]$, então há duas opções: (1) esse par está em $A[1..n-1]$
(2) um dos elementos do par é o $A[n]$ e o elemento $K - A[n]$ existe em $A[1..n-1]$. (*)
Se nada disso acontecer, então $A[1..n]$ não tem um tal par.
É claro que essa abordagem só funciona se $n > 1$
Segue o pseudocódigo.

```
SOMAK (A, n, K)
1   se n ≤ 1
2       devolve (-1,-1)
3   (i,j) ← SOMAK (A, n-1, K)
4   se (i,j) ≠ (-1,-1)
5       devolve (i,j)
6   i ← BUSCALINEAR (A, n-1, K-A[n])
7   se i ≠ -1
8       devolve (i,n)
9   devolve (-1,-1).
```

Para provar sua corretude, precisamos provar por indução em $n$ que $P(n)$ = "SOMAK$(A, n, K)$ corretamente detecta se existem índices $(i,j)$ tais que $A[i] + A[j] = K$" é verdade.

BASE: $n \leq 1$. Se o vetor tem 0 ou 1 elementos, claramente os índices não existem. E nesse caso, SOMAK devolve $(-1,-1)$, de fato. Logo, $P(0)$ e $P(1)$ valem.

HIPÓTESE: $P(q)$ vale, para $1 \leq q < n$, isto é, SOMAK funciona sobre vetores de tamanho menor que $n$.

PASSO: Seja $n > 1$. O algoritmo começa executando a linha 3, onde chama SOMAK$(A, n-1, k)$. Por HI, essa chamada funciona, isto é, se $A[1..n-1]$ tem dois elementos que somam $k$, seus índices são devolvidos. Nesse caso, tais elementos estão em $A[1..n]$. O algoritmo então devolve esses índices, como esperado.
Se os índices não existem em $A[1..n-1]$, a chamada recursiva devolveu $(-1,-1)$. Faz-se então uma chamada à BUSCALINEAR para procurar por $k-A[n]$ em $A[1..n-1]$. Caso exista, devolvemos os índices $(i,n)$, em que $A[i]=k-A[n]$, pois sabemos que BUSCALINEAR funciona.
Caso não exista, então não há possibilidade de $A[1..n]$ conter dois elementos que somam $k$ (conforme discussão ★).
Assim, $P(n)$ vale. CQD

7. O *Quicksort* é outro algoritmo de divisão e conquista para o problema da ordenação. É muito usado na prática, pois é fácil de implementar, tem tempo esperado de execução $\Theta(n \log n)$ e as constantes escondidas pela notação assintótica são pequenas.

Solução completa no meu livro!

8. No ensino fundamental, aprendemos um algoritmo para multiplicar dois números "grandes". Esse algoritmo tem tempo de execução $O(n^2)$ quando os números têm $n$ dígitos[1].

   Prove a corretude e analise seu tempo de execução do algoritmo de Karatsuba. *Dica:* primeiro, imagine como os números estão sendo representados (afinal, se fossem duas variáveis inteiras bastaria fazer $x \times y$ para resolver o problema).

Solução completa no meu livro!

10. Suponha que um vetor (não necessariamente ordenado) $A[1..n]$ contém todos os números em $\{1, 2, \ldots, n-1\}$, ou seja, um dos números está aparecendo duas vezes. Dê um algoritmo de divisão e conquista que determina qual é o número que ocorre duas vezes em $A$.

ATENÇÃO! Ordenar o vetor e procurar o número não é uma solução permitida pelo enunciado (apesar de resolver o problema).

Note que se um número aparece duas vezes em $A[1..n]$:
(i) aparece duas vezes em $A[1..\frac{n}{2}]$ ou
(ii) aparece duas vezes em $A[\frac{n}{2}+1..n]$ ou
(iii) tem uma cópia em $A[1..\frac{n}{2}]$ e outra em $A[\frac{n}{2}+1..n]$

Isso nos dá o seguinte algoritmo de divisão e conquista:

```
ENCONTRAREPETIDO (A, esq, dir)
1   se esq ≥ dir
2       devolve -1
3   meio = ⌊(esq + dir)/2⌋
4   i = ENCONTRAREPETIDO (A, esq, meio)
5   se i ≠ -1
6       devolve i
7   i = ENCONTRAREPETIDO (A, meio+1, dir)
8   se i ≠ -1
9       devolve i
10  para i = esq até meio
11      para j = meio+1 até dir
12          se A[i] == A[j]
13              devolve A[i]
14  devolve -1
```

Para a corretude, precisamos mostrar que

$P(n)$ = "ENCONTRAREPETIDO $(A, esq, dir)$, com $n = dir - esq + 1$, devolve o elemento repetido em $A[esq..dir]$, se este existir, e devolve $-1$, caso contrário."

vale para todo $n$.

BASE: $n \leq 1$. Então $dir - esq + 1 \leq 1$, ou seja, $dir \leq esq$. Nesse caso, o algoritmo devolve $-1$, que é o esperado quando o vetor tem 0 ou 1 elementos.

HIPÓTESE: $P(k)$ vale para $1 \leq k < n$ (o algoritmo funciona sobre vetores de tamanho menor que $n$).

PASSO: $n > 1$. Então $dir - esq + 1 > 1$, ou seja, $dir > esq$. Nesse caso, o algoritmo calcula $meio = \lfloor (esq + dir)/2 \rfloor$ e chama ENCONTRAREPETIDO $(A, esq, meio)$. Note que

$$\begin{aligned} meio - esq + 1 &= \lfloor (esq + dir)/2 \rfloor - esq + 1 \\ &\leq (esq + dir)/2 - esq + 1 \\ &= \frac{dir}{2} - \frac{esq}{2} + \frac{2}{2} = \frac{n+1}{2} \end{aligned}$$

e $\frac{n+1}{2} < n$ quando $n > 1$. Como $meio - esq + 1 < n$ é o tamanho do vetor na chamada da linha 4, por HI essa chamada corretamente detecta se existe elemento repetido em $A[esq..meio]$. Se existe, $i \neq 1$ e nosso algoritmo devolve $i$. Se não existe, chamamos ENCONTRAREPETIDO$(A, meio+1, dir)$. Note que

$$\begin{aligned} dir - (meio+1) + 1 &= dir - \lfloor (esq + dir)/2 \rfloor \\ &\leq dir - (esq + dir)/2 + 1 \\ &= \frac{dir}{2} - \frac{esq}{2} + \frac{2}{2} = \frac{n+1}{2}. \end{aligned}$$

Novamente, por HI, essa chamada corretamente detecta se existe elemento repetido em $A[meio+1..dir]$. Se existe, $i \neq 1$ e nosso algoritmo devolve $i$.

(CONTINUA)

Agora sabemos que não há elemento repetido em A[esq.. meio] nem em A[meio+1.. dir]. Então, se houver elemento repetido em A[esq.. dir], um deles está em A[esq.. meio] e o outro está em A[meio+1.. dir]. O laço das linhas 10 a 13 encontrará isso, caso exista, e o algoritmo devolverá o elemento. Se não houver um elemento em A[esq.. meio] igual a outro em A[meio+1.. dir], então A[esq.. dir] não tem elemento repetido e o algoritmo devolve -1 (linha 14), como esperado.
Logo, ENCONTRAREPETIDO (A, esq, dir) detecta corretamente se existe repetição em A[esq.. dir] e P(n) vale.

CQD

O tempo $T(n)$ de execução de ENCONTRAREPETIDO é dado por $T(1) = 1$ e $T(n) \leq 2T(\frac{n}{2}) + O(n^2)$, pois há no máximo duas chamadas recursivas sobre metade do vetor e o laço da linha 10 leva tempo $O(n^2)$, já que para $n/2$ valores de i, o laço interno executa no máx. $n/2$ vezes, para os valores de j, realizando um teste de tempo constante.

Pelo método mestre, $T'(n) = 2T'(\frac{n}{2}) + n^2$ tem tempo $\Theta(n^2)$, pois $a=2, b=2, k=2$ e $a = 2 < 2^2 = b^k$.

Então $T(n)$ é $O(n^2)$.