MC458 — Projeto e Análise de Algoritmos I

C.C. de Souza C.N. da Silva O. Lee

Antes de mais nada...

- Uma versão anterior deste conjunto de slides foi preparada por Cid Carvalho de Souza e Cândida Nunes da Silva para uma instância anterior desta disciplina.
- O que vocês tem em mãos é uma versão modificada preparada para atender a meus gostos.
- Nunca é demais enfatizar que o material é apenas um guia e não deve ser usado como única fonte de estudo. Para isso consultem a bibliografia (em especial o CLR ou CLRS).

Orlando Lee

Agradecimentos (Cid e Cândida)

- Várias pessoas contribuíram direta ou indiretamente com a preparação deste material.
- Algumas destas pessoas cederam gentilmente seus arquivos digitais enquanto outras cederam gentilmente o seu tempo fazendo correções e dando sugestões.
- Uma lista destes "colaboradores" (em ordem alfabética) é dada abaixo:
  - Célia Picinin de Mello
  - José Coelho de Pina
  - Orlando Lee
  - Paulo Feofiloff
  - Pedro Rezende
  - Ricardo Dahab
  - Zanoni Dias

Recorrências

Resolução de Recorrências

Resolução de Recorrências

- Relações de recorrência expressam a complexidade de algoritmos recursivos como, por exemplo, os algoritmos de divisão e conquista.

Resolução de Recorrências

- Relações de recorrência expressam a complexidade de algoritmos recursivos como, por exemplo, os algoritmos de divisão e conquista.
- É preciso saber resolver as recorrências para que possamos efetivamente determinar a complexidade dos algoritmos recursivos.

Mergesort

```
MergeSort(A, p, r)
1   se p < r
2      então q ← ⌊(p + r)/2⌋
3            MergeSort(A, p, q)
4            MergeSort(A, q + 1, r)
5            Intercala(A, p, q, r)
```

Mergesort

```
MergeSort(A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3             MergeSort(A, p, q)
4             MergeSort(A, q + 1, r)
5             Intercala(A, p, q, r)
```

Qual é a complexidade de MergeSort?

Mergesort

```
MergeSort(A, p, r)
1   se p < r
2      então q ← ⌊(p + r)/2⌋
3            MergeSort(A, p, q)
4            MergeSort(A, q + 1, r)
5            Intercala(A, p, q, r)
```

Qual é a complexidade de MergeSort?

Seja $T(n) :=$ o consumo de tempo máximo (pior caso) em função de $n = r - p + 1$

Complexidade do Mergesort

```
MergeSort(A, p, r)
1   se p < r
2      então q ← ⌊(p + r)/2⌋
3            MergeSort(A, p, q)
4            MergeSort(A, q + 1, r)
5            Intercala(A, p, q, r)
```

linha	consumo de tempo
1	?
2	?
3	?
4	?
5	?

Complexidade do Mergesort

```
MergeSort(A, p, r)
1   se p < r
2      então q ← ⌊(p + r)/2⌋
3            MergeSort(A, p, q)
4            MergeSort(A, q + 1, r)
5            Intercala(A, p, q, r)
```

linha	consumo de tempo
1	b_0
2	b_1
3	$T(\lceil n/2 \rceil)$
4	$T(\lfloor n/2 \rfloor)$
5	an

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + an + (b_0 + b_1)$$

Resolução de recorrências

Resolução de recorrências

- Queremos resolver a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + an + b \qquad \text{para } n \geq 2. \end{aligned}$$

Resolução de recorrências

- Queremos resolver a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + an + b \qquad \text{para } n \geq 2. \end{aligned}$$

- Resolver uma recorrência significa encontrar uma fórmula fechada para $T(n)$.

Resolução de recorrências

- Queremos resolver a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + an + b \qquad \text{para } n \geq 2. \end{aligned}$$

- Resolver uma recorrência significa encontrar uma fórmula fechada para $T(n)$.
- Não é necessário achar uma solução exata.

Resolução de recorrências

- Queremos resolver a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + an + b \qquad \text{para } n \geq 2. \end{aligned}$$

- Resolver uma recorrência significa encontrar uma fórmula fechada para $T(n)$.

- Não é necessário achar uma solução exata.
  Basta encontrar uma função $f(n)$ tal que
  $T(n) \in \Theta(f(n))$.

Resolução de recorrências

Veremos os seguintes métodos para resolução de recorrências:

Resolução de recorrências

Veremos os seguintes métodos para resolução de recorrências:

- substituição
- árvore de recorrência
- Teorema Master

Algumas relações

Antes de vermos as técnicas, vamos relembrar alguns fatos que são úteis quando lidamos com chão, teto e logaritmos:

- $\lceil \frac{n}{2} \rceil + \lfloor \frac{n}{2} \rfloor = n$ para todo inteiro $n \geq 0$,
- $\lfloor x \rfloor \leq x$ para todo número real x,
- $\left\lfloor \frac{\lfloor x/a \rfloor}{b} \right\rfloor = \lfloor \frac{x}{ab} \rfloor$ para quaisquer número real $x \geq 0$ e inteiros $a, b > 0$,
- $\left\lceil \frac{\lceil x/a \rceil}{b} \right\rceil = \lceil \frac{x}{ab} \rceil$ para quaisquer número real $x \geq 0$ e inteiros $a, b > 0$,
- $\lceil \frac{n}{2} \rceil \leq \frac{n}{2} + 1$ (ou $\lceil \frac{n}{2} \rceil \leq n$),
- $\log(xy) = \log x + \log y$ (qualquer base),
- $\log(\frac{x}{y}) = \log x - \log y$ (qualquer base),
- $\lg(\lfloor \frac{n}{2} \rfloor) \leq \lg n - 1$ e
- $a^{\log_b n} = n^{\log_b a}$.

Observação: $\log_b n = \frac{\log_a n}{\log_a b}$. Isto implica que $\log_b n \in \Theta(\log_a n)$ para quaisquer $a, b > 0$.

Método da substituição

Método da substituição

- Idéia básica: "adivinhe" qual é a solução e prove por indução que ela funciona!

Método da substituição

- Idéia básica: "adivinhe" qual é a solução e prove por indução que ela funciona!
- Método poderoso mas nem sempre aplicável (obviamente).

Método da substituição

- Idéia básica: "adivinhe" qual é a solução e prove por indução que ela funciona!
- Método poderoso mas nem sempre aplicável (obviamente).
- Com prática e experiência fica mais fácil de usar!

Exemplo

Considere a recorrência:

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \qquad \text{para } n \geq 2. \end{aligned}$$

Exemplo

Considere a recorrência:

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \qquad \text{para } n \geq 2. \end{aligned}$$

Chuto que $T(n) \in O(n \lg n)$.

Exemplo

Considere a recorrência:

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \qquad \text{para } n \geq 2. \end{aligned}$$

Chuto que $T(n) \in O(n \lg n)$.

Mais precisamente, chuto que $T(n) \leq 3n \lg n$.

(Lembre-se que $\lg n = \log_2 n$.)

Exemplo

$$
\begin{aligned}
T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \\
&\leq 3 \left\lceil \frac{n}{2} \right\rceil \lg \left\lceil \frac{n}{2} \right\rceil + 3 \left\lfloor \frac{n}{2} \right\rfloor \lg \left\lfloor \frac{n}{2} \right\rfloor + n \\
&\leq 3 \left\lceil \frac{n}{2} \right\rceil \lg n + 3 \left\lfloor \frac{n}{2} \right\rfloor (\lg n - 1) + n \\
&= 3 \left(\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor \right) \lg n - 3 \left\lfloor \frac{n}{2} \right\rfloor + n \\
&= 3n \lg n - 3 \left\lfloor \frac{n}{2} \right\rfloor + n \\
&\leq 3n \lg n.
\end{aligned}
$$

Exemplo

- Mas espere um pouco!

Exemplo

- Mas espere um pouco!
- $T(1) = 1$ e $3.1. \lg 1 = 0$ e a base da indução não funciona!

Exemplo

- Mas espere um pouco!
- $T(1) = 1$ e $3.1. \lg 1 = 0$ e a base da indução não funciona!
- Certo, mas lembre-se da definição da classe $O(\ )$.

Exemplo

- Mas espere um pouco!
- $T(1) = 1$ e $3.1. \lg 1 = 0$ e a base da indução não funciona!
- Certo, mas lembre-se da definição da classe $O(\ )$.

  Só preciso provar que $T(n) \leq 3n \lg n$ para $n \geq n_0$ onde n_0 é alguma constante.

Exemplo

- Mas espere um pouco!
- $T(1) = 1$ e $3.1.\lg 1 = 0$ e a base da indução não funciona!
- Certo, mas lembre-se da definição da classe $O(\ )$.

  Só preciso provar que $T(n) \leq 3n \lg n$ para $n \geq n_0$ onde n_0 é alguma constante.

  Vamos tentar com $n_0 = 2$. Neste caso, para poder fazer o passo de indução é necessário que $\lfloor n/2 \rfloor, \lceil n/2 \rceil \geq 2$, ou seja, $n \geq 4$. Assim, usamos como base os valores $n = 2, 3$:

$$T(2) = T(1) + T(1) + 2 = 4 \leq 3.2.\lg 2,$$
$$T(3) = T(1) + T(2) + 3 = 1 + 4 + 3 = 8 \leq 3.3.\lg 3.$$

Exemplo

- Certo, funcionou para $T(1) = 1$.

Exemplo

- Certo, funcionou para $T(1) = 1$.
- Mas e se por exemplo $T(1) = 8$?

Exemplo

- Certo, funcionou para $T(1) = 1$.
- Mas e se por exemplo $T(1) = 8$?

  Então $T(2) = T(1) + T(1) + 2 = 8 + 8 + 2 = 18$ e $3.2. \lg 2 = 6$.
  Não deu certo...

Exemplo

- Certo, funcionou para $T(1) = 1$.
- Mas e se por exemplo $T(1) = 8$?

  Então $T(2) = T(1) + T(1) + 2 = 8 + 8 + 2 = 18$ e $3.2. \lg 2 = 6$.
  Não deu certo...
- Certo, mas aí basta escolher uma constante maior. Provamos que $T(n) \leq 10n \lg n$ como antes e para esta escolha:

$$T(2) = 18 \leq 10.2. \lg 2$$
$$T(3) = T(1) + T(2) + 3 = 8 + 18 + 3 = 29 \leq 10.3. \lg 3$$

Exemplo

- Certo, funcionou para $T(1) = 1$.
- Mas e se por exemplo $T(1) = 8$?

  Então $T(2) = T(1) + T(1) + 2 = 8 + 8 + 2 = 18$ e $3.2. \lg 2 = 6$.
  Não deu certo...
- Certo, mas aí basta escolher uma constante maior. Provamos que $T(n) \leq 10n \lg n$ como antes e para esta escolha:

$$T(2) = 18 \leq 10.2. \lg 2$$
$$T(3) = T(1) + T(2) + 3 = 8 + 18 + 3 = 29 \leq 10.3. \lg 3$$

- De modo geral, se o passo de indução funciona, é possível escolher c e n_0 de modo conveniente!

Como achar as constantes?

- Tudo bem. Dá até para chutar que $T(n)$ pertence a classe $O(n \lg n)$.

Como achar as constantes?

- Tudo bem. Dá até para chutar que $T(n)$ pertence a classe $O(n \lg n)$.
- Mas como descobrir que $T(n) \leq 3n \lg n$? Como achar a constante 3?

Como achar as constantes?

- Tudo bem. Dá até para chutar que $T(n)$ pertence a classe $O(n \lg n)$.
- Mas como descobrir que $T(n) \leq 3n \lg n$? Como achar a constante 3?
- Eis um método simples: prove por indução que $T(n) \leq cn \lg n$ para $n \geq n_0$ onde c e n_0 são constantes a serem determinadas para que as contas funcionem.

Descobrindo c

$$
\begin{aligned}
T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \\
&\leq c \left\lceil \frac{n}{2} \right\rceil \lg \left\lceil \frac{n}{2} \right\rceil + c \left\lfloor \frac{n}{2} \right\rfloor \lg \left\lfloor \frac{n}{2} \right\rfloor + n \\
&\leq c \left\lceil \frac{n}{2} \right\rceil \lg n + c \left\lfloor \frac{n}{2} \right\rfloor (\lg n - 1) + n \\
&= c \left(\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor \right) \lg n - c \left\lfloor \frac{n}{2} \right\rfloor + n \\
&= cn \lg n - c \left\lfloor \frac{n}{2} \right\rfloor + n \\
&\leq cn \lg n.
\end{aligned}
$$

Para garantir a última desigualdade basta que $-c\lfloor n/2 \rfloor + n \leq 0$.
Tomando $c \geq 3$ funciona.

Completando o exemplo

Mostramos que a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \qquad \text{para } n \geq 2. \end{aligned}$$

satisfaz $T(n) \in O(n \lg n)$.

Completando o exemplo

Mostramos que a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \qquad \text{para } n \geq 2. \end{aligned}$$

satisfaz $T(n) \in O(n \lg n)$.

Mas quem garante que $T(n)$ não é "menor"?

Completando o exemplo

Mostramos que a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \qquad \text{para } n \geq 2. \end{aligned}$$

satisfaz $T(n) \in O(n \lg n)$.

Mas quem garante que $T(n)$ não é "menor"?

O melhor é mostrar que $T(n) \in \Theta(n \lg n)$.

Completando o exemplo

Mostramos que a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \qquad \text{para } n \geq 2. \end{aligned}$$

satisfaz $T(n) \in O(n \lg n)$.

Mas quem garante que $T(n)$ não é "menor"?

O melhor é mostrar que $T(n) \in \Theta(n \lg n)$.

Resta então mostrar que $T(n) \in \Omega(n \lg n)$.

A prova é similar. (Exercício!)

Como chutar?

Não há nenhuma receita genérica para adivinhar soluções de recorrências.
A experiência é o fator mais importante.

Como chutar?

Não há nenhuma receita genérica para adivinhar soluções de recorrências. A experiência é o fator mais importante.

Felizmente, há várias idéias que podem ajudar.

Como chutar?

Não há nenhuma receita genérica para adivinhar soluções de recorrências. A experiência é o fator mais importante.

Felizmente, há várias idéias que podem ajudar.

Considere a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(\lfloor n/2 \rfloor) + n \qquad \text{para } n \geq 2. \end{aligned}$$

Como chutar?

Não há nenhuma receita genérica para adivinhar soluções de recorrências. A experiência é o fator mais importante.

Felizmente, há várias idéias que podem ajudar.

Considere a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(\lfloor n/2 \rfloor) + n \qquad \text{para } n \geq 2. \end{aligned}$$

Ela é quase idêntica à anterior e podemos chutar que $T(n) \in \Theta(n \lg n)$.

Como chutar?

Não há nenhuma receita genérica para adivinhar soluções de recorrências.
A experiência é o fator mais importante.

Felizmente, há várias idéias que podem ajudar.

Considere a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(\lfloor n/2 \rfloor) + n \qquad \text{para } n \geq 2. \end{aligned}$$

Ela é quase idêntica à anterior e podemos chutar que $T(n) \in \Theta(n \lg n)$.

Chuto que existem c, n_0 tais que $T(n) \leq cn \lg n$ para $n \geq n_0$.

Como chutar?

$$
\begin{aligned}
T(n) &= 2T(\lfloor n/2 \rfloor) + n \\
&\leq 2c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor) + n \\
&\leq 2c(n/2)\lg(n/2) + n \\
&= cn\lg n - cn\lg 2 + n \\
&= cn\lg n - cn + n \\
&\leq cn\lg n,
\end{aligned}
$$

onde a última desigualdade vale se $c \geq 1$.

Como chutar?

Como chutar?

Considere agora a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(\lceil n/2 \rceil) + n \qquad \text{para } n \geq 2. \end{aligned}$$

Como chutar?

Considere agora a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(\lceil n/2 \rceil) + n \qquad \text{para } n \geq 2. \end{aligned}$$

Ela é quase idêntica à anterior e podemos chutar novamente que $T(n) \in \Theta(n \lg n)$. (Exercício!)

Observação: será necessário fortalecer a hipótese de indução (veja mais adiante).

Como chutar?

Considere agora a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(\lfloor n/2 \rfloor + 17) + n \qquad \text{para } n \geq 2. \end{aligned}$$

Como chutar?

Considere agora a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(\lfloor n/2 \rfloor + 17) + n \qquad \text{para } n \geq 2. \end{aligned}$$

Ela parece bem mais difícil por causa do "17" no lado direito.

Como chutar?

Considere agora a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(\lfloor n/2 \rfloor + 17) + n \qquad \text{para } n \geq 2. \end{aligned}$$

Ela parece bem mais difícil por causa do "17" no lado direito.

Intuitivamente, porém, isto não deveria afetar a solução. Para n **grande** a diferença entre $T(\lfloor n/2 \rfloor)$ e $T(\lfloor n/2 \rfloor + 17)$ não é tanta.

Como chutar?

Considere agora a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(\lfloor n/2 \rfloor + 17) + n \qquad \text{para } n \geq 2. \end{aligned}$$

Ela parece bem mais difícil por causa do "17" no lado direito.

Intuitivamente, porém, isto não deveria afetar a solução. Para n **grande** a diferença entre $T(\lfloor n/2 \rfloor)$ e $T(\lfloor n/2 \rfloor + 17)$ não é tanta.

Chuto então que $T(n) \in \Theta(n \lg n)$. (Exercício!)

Observação: será necessário fortalecer a hipótese de indução. Além disso, a prova requer também alguns truques...

Truques e sutilezas

Algumas vezes adivinhamos corretamente a solução de uma recorrência, mas as contas aparentemente não funcionam! Em geral, o que é necessário é fortalecer a hipótese de indução.

Truques e sutilezas

Algumas vezes adivinhamos corretamente a solução de uma recorrência, mas as contas aparentemente não funcionam! Em geral, o que é necessário é fortalecer a hipótese de indução.

Considere a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 1 \quad \text{para } n \geq 2. \end{aligned}$$

Truques e sutilezas

Algumas vezes adivinhamos corretamente a solução de uma recorrência, mas as contas aparentemente não funcionam! Em geral, o que é necessário é fortalecer a hipótese de indução.

Considere a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 1 \quad \text{para } n \geq 2. \end{aligned}$$

Chutamos que $T(n) \in O(n)$ e tentamos mostrar que $T(n) \leq cn$ para alguma constante $c > 0$.

Truques e sutilezas

Algumas vezes adivinhamos corretamente a solução de uma recorrência, mas as contas aparentemente não funcionam! Em geral, o que é necessário é fortalecer a hipótese de indução.

Considere a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 1 \quad \text{para } n \geq 2. \end{aligned}$$

Chutamos que $T(n) \in O(n)$ e tentamos mostrar que $T(n) \leq cn$ para alguma constante $c > 0$.

$$\begin{aligned} T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 1 \\ &\leq c\lceil n/2 \rceil + c\lfloor n/2 \rfloor + 1 \\ &= cn + 1. \end{aligned}$$

(Humm, falhou...)

Truques e sutilezas

Algumas vezes adivinhamos corretamente a solução de uma recorrência, mas as contas aparentemente não funcionam! Em geral, o que é necessário é fortalecer a hipótese de indução.

Considere a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 1 \quad \text{para } n \geq 2. \end{aligned}$$

Chutamos que $T(n) \in O(n)$ e tentamos mostrar que $T(n) \leq cn$ para alguma constante $c > 0$.

$$\begin{aligned} T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 1 \\ &\leq c\lceil n/2 \rceil + c\lfloor n/2 \rfloor + 1 \\ &= cn + 1. \end{aligned}$$

(Humm, falhou...)

E agora? Será que erramos o chute? Será que $T(n) \notin O(n)$?

Truques e sutilezas

Na verdade, adivinhamos corretamente. Para provar isso, é preciso usar uma hipótese de indução mais forte.

Truques e sutilezas

Na verdade, adivinhamos corretamente. Para provar isso, é preciso usar uma hipótese de indução mais forte.

Vamos mostrar que $T(n) \leq cn - b$ onde $b > 0$ é uma constante.

Truques e sutilezas

Na verdade, adivinhamos corretamente. Para provar isso, é preciso usar uma hipótese de indução mais forte.

Vamos mostrar que $T(n) \leq cn - b$ onde $b > 0$ é uma constante.

$$\begin{aligned} T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 1 \\ &\leq c\lceil n/2 \rceil - b + c\lfloor n/2 \rfloor - b + 1 \\ &= cn - 2b + 1 \\ &\leq cn - b \end{aligned}$$

onde a última desigualdade vale se $b \geq 1$.

Truques e sutilezas

Considere a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + 1 \quad \text{para } n \geq 2. \end{aligned}$$

Truques e sutilezas

Considere a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + 1 \quad \text{para } n \geq 2. \end{aligned}$$

Chutamos que $T(n) \in O(\lg n)$ e tentamos mostrar que que $T(n) \leq c \lg n$ para alguma constante $c > 0$.

Truques e sutilezas

Considere a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + 1 \quad \text{para } n \geq 2. \end{aligned}$$

Chutamos que $T(n) \in O(\lg n)$ e tentamos mostrar que que $T(n) \leq c \lg n$ para alguma constante $c > 0$.

$$\begin{aligned} T(n) &= T(\lceil n/2 \rceil) + 1 \\ &\leq c \lg(\lceil n/2 \rceil) + 1 \\ &\leq c \lg(n/2 + 1) + 1 \\ &= c \lg((n+2)/2) + 1 \\ &= c \lg(n+2) - c \lg 2 + 1 \\ &= c \lg(n+2) - c + 1. \end{aligned}$$

(Humm, falhou. . .)

Truques e sutilezas

Tudo que conseguimos foi mostrar que $T(n) \leq c \lg(n+2)$.

Truques e sutilezas

Tudo que conseguimos foi mostrar que $T(n) \leq c \lg(n+2)$.

Vamos tentar nos livrar do termo $+2$ fortalecendo a hipótese de indução.

Truques e sutilezas

Tudo que conseguimos foi mostrar que $T(n) \leq c \lg(n+2)$.

Vamos tentar nos livrar do termo $+2$ fortalecendo a hipótese de indução.

Vamos mostrar que $T(n) \leq c \lg(n-2)$ para alguma constante $c > 0$.

Truques e sutilezas

Tudo que conseguimos foi mostrar que $T(n) \leq c \lg(n+2)$.

Vamos tentar nos livrar do termo $+2$ fortalecendo a hipótese de indução.

Vamos mostrar que $T(n) \leq c \lg(n-2)$ para alguma constante $c > 0$.

$$\begin{aligned}
T(n) &= T(\lceil n/2 \rceil) + 1 \\
&\leq c \lg(\lceil n/2 \rceil - 2) + 1 \\
&\leq c \lg(n/2 + 1 - 2) + 1 \\
&= c \lg(n/2 - 1) + 1 \\
&= c \lg((n-2)/2) + 1 \\
&= c \lg(n-2) - c \lg 2 + 1 \\
&= c \lg(n-2) - c + 1 \\
&\leq c \lg(n-2)
\end{aligned}$$

onde a última desigualdade vale tomando $c \geq 1$.

Árvore de recorrência

Árvore de recorrência

- Não precisa adivinhar a resposta!

Árvore de recorrência

- Não precisa adivinhar a resposta!
- Permite visualizar o que acontece quando a recorrência é iterada.

Árvore de recorrência

- Não precisa adivinhar a resposta!
- Permite visualizar o que acontece quando a recorrência é iterada.
- É relativamente fácil organizar as contas.

Árvore de recorrência

- Não precisa adivinhar a resposta!
- Permite visualizar o que acontece quando a recorrência é iterada.
- É relativamente fácil organizar as contas.
- Útil para recorrências de algoritmos de divisão-e-conquista.

Árvore de recorrência

Considere a recorrência

$$\begin{aligned} T(n) &= \Theta(1) && \text{para } n = 1, 2, 3, \\ T(n) &= 3T(\lfloor n/4 \rfloor) + cn^2 && \text{para } n \geq 4, \end{aligned}$$

onde $c > 0$ é uma constante.

Árvore de recorrência

Considere a recorrência

$$\begin{aligned} T(n) &= \Theta(1) && \text{para } n = 1, 2, 3, \\ T(n) &= 3T(\lfloor n/4 \rfloor) + cn^2 && \text{para } n \geq 4, \end{aligned}$$

onde $c > 0$ é uma constante.

CLRS costuma usar a notação $T(n) = \Theta(1)$ para indicar que $T(n)$ é uma constante.

Árvore de recorrência

Simplificação

Vamos supor que a recorrência está definida apenas para potências de 4

$$\begin{aligned} T(n) &= \Theta(1) && \text{para } n = 1, \\ T(n) &= 3T(n/4) + cn^2 && \text{para } n = 4, 16, \ldots, 4^i, \ldots. \end{aligned}$$

Árvore de recorrência

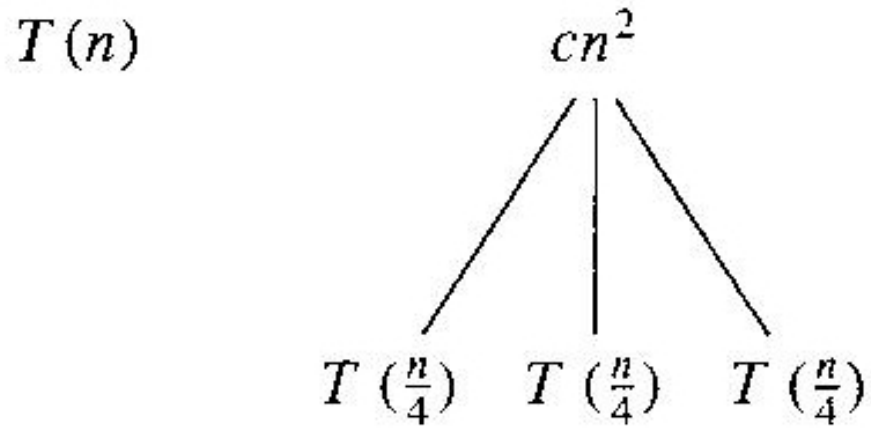
Simplificação

Vamos supor que a recorrência está definida apenas para potências de 4

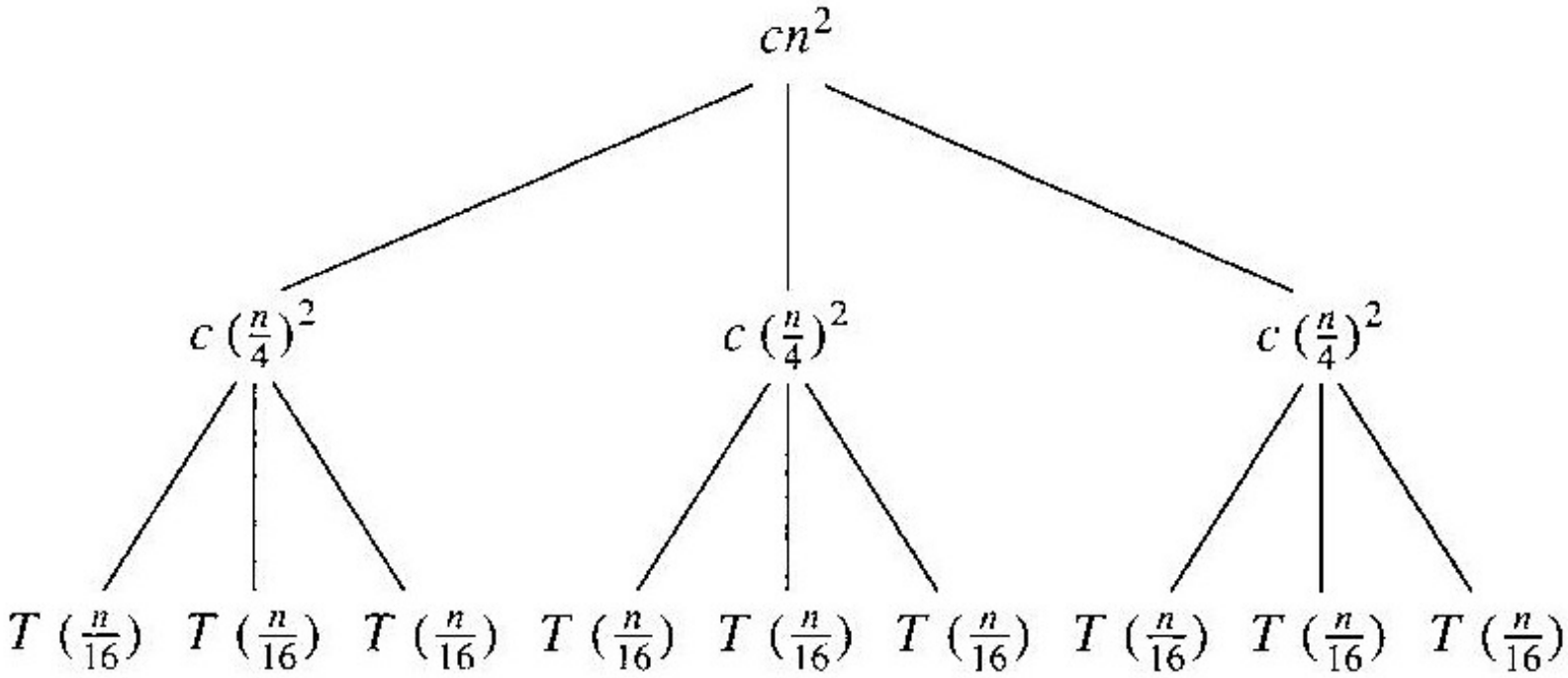
$$\begin{aligned} T(n) &= \Theta(1) && \text{para } n = 1, \\ T(n) &= 3T(n/4) + cn^2 && \text{para } n = 4, 16, \ldots, 4^i, \ldots. \end{aligned}$$

Isto permite descobrir mais facilmente a solução. Depois usamos o método da substituição para formalizar.

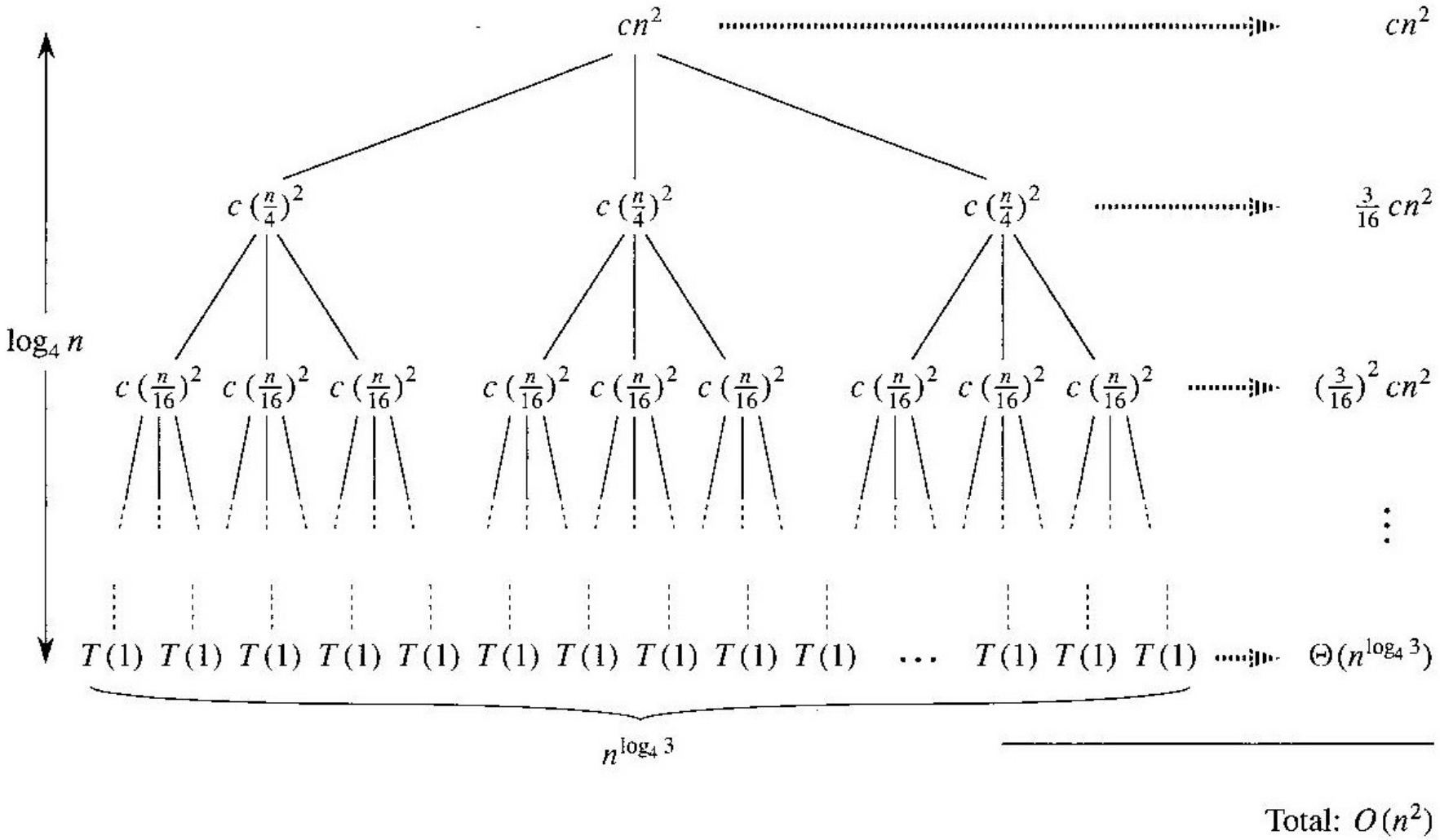
Árvore de recorrência



Árvore de recorrência



Árvore de recorrência



Árvore de recorrência

Árvore de recorrência

- O número de níveis é $\log_4 n + 1$ $(0, 1, 2, \ldots, \log_4 n)$.

Árvore de recorrência

- O número de níveis é $\log_4 n + 1$ $(0, 1, 2, \ldots, \log_4 n)$.
- No nível i o tempo gasto (sem contar as chamadas recursivas) é $(3/16)^i cn^2$.

Árvore de recorrência

- O número de níveis é $\log_4 n + 1$ $(0, 1, 2, \ldots, \log_4 n)$.
- No nível i o tempo gasto (sem contar as chamadas recursivas) é $(3/16)^i cn^2$.
- Uma árvore ternária de altura h tem (no máximo) 3^h folhas.

Árvore de recorrência

- O número de níveis é $\log_4 n + 1$ $(0, 1, 2, \ldots, \log_4 n)$.
- No nível i o tempo gasto (sem contar as chamadas recursivas) é $(3/16)^i cn^2$.
- Uma árvore ternária de altura h tem (no máximo) 3^h folhas.

  Logo, no último nível há $3^{\log_4 n} = n^{\log_4 3}$ folhas.

  Como $T(1) = \Theta(1)$ o tempo gasto é $\Theta(n^{\log_4 3})$.

Árvore de recorrência

Logo,

$$
\begin{aligned}
T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \left(\frac{3}{16}\right)^3 cn^2 + \cdots + \\
&\quad + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \Theta(n^{\log_4 3}) \\
&= \left[\sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i\right] cn^2 + \Theta(n^{\log_4 3}) \\
&\leq \left[\sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i\right] cn^2 + \Theta(n^{\log_4 3}) = \frac{16}{13}cn^2 + \Theta(n^{\log_4 3}),
\end{aligned}
$$

e $T(n) \in O(n^2)$. **Observação:** $\sum_{i=0}^{\infty} q^i = \frac{1}{1-q}$ para $0 < q < 1$.

Árvore de recorrência

Mas $T(n) \in O(n^2)$ é realmente a solução da recorrência original?

Árvore de recorrência

Mas $T(n) \in O(n^2)$ é realmente a solução da recorrência original?

Com base na árvore de recorrência, chutamos que $T(n) \leq dn^2$ para alguma constante $d > 0$.

Árvore de recorrência

Mas $T(n) \in O(n^2)$ é realmente a solução da recorrência original?

Com base na árvore de recorrência, chutamos que $T(n) \leq dn^2$ para alguma constante $d > 0$.

$$\begin{aligned} T(n) &= 3T(\lfloor n/4 \rfloor) + cn^2 \\ &\leq 3d\lfloor n/4 \rfloor^2 + cn^2 \\ &\leq 3d(n/4)^2 + cn^2 \\ &= \frac{3}{16}dn^2 + cn^2 \\ &\leq dn^2 \end{aligned}$$

onde a última desigualdade vale se $d \geq (16/13)c$.

Árvore de recorrência

Resumo

- O número de nós em cada nível da árvore é o número de chamadas recursivas.

Árvore de recorrência

Resumo

- O número de nós em cada nível da árvore é o número de chamadas recursivas.
- Em cada nó indicamos o "tempo" ou "trabalho" gasto naquele nó que **não** corresponde a chamadas recursivas.

Árvore de recorrência

Resumo

- O número de nós em cada nível da árvore é o número de chamadas recursivas.
- Em cada nó indicamos o "tempo" ou "trabalho" gasto naquele nó que **não** corresponde a chamadas recursivas.
- Na coluna mais à direita indicamos o tempo total naquele nível que **não** corresponde a chamadas recursivas.

Árvore de recorrência

Resumo

- O número de nós em cada nível da árvore é o número de chamadas recursivas.
- Em cada nó indicamos o "tempo" ou "trabalho" gasto naquele nó que **não** corresponde a chamadas recursivas.
- Na coluna mais à direita indicamos o tempo total naquele nível que **não** corresponde a chamadas recursivas.
- Somando ao longo da coluna determina-se a solução da recorrência.

Vamos tentar juntos?

Eis um exemplo um pouco mais complicado.

Vamos tentar juntos?

Eis um exemplo um pouco mais complicado.

Vamos resolver a recorrência

$$\begin{aligned} T(n) &= \Theta(1) && \text{para } n = 1, 2, \\ T(n) &= T(\lceil n/3 \rceil) + T(\lfloor 2n/3 \rfloor) + cn && \text{para } n \geq 3. \end{aligned}$$

usando a árvore de recorrência.

Vamos tentar juntos?

Eis um exemplo um pouco mais complicado.

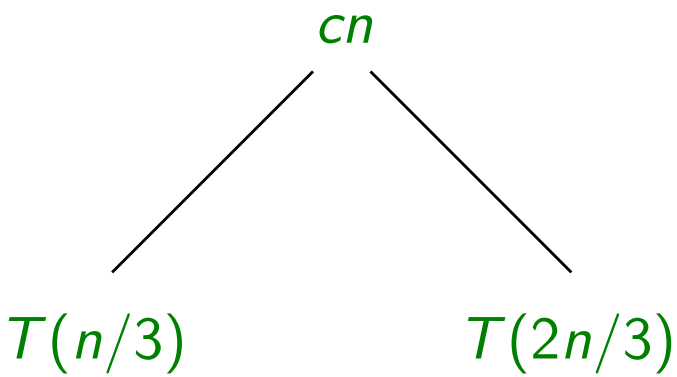
Vamos resolver a recorrência

$$\begin{aligned} T(n) &= \Theta(1) && \text{para } n = 1, 2, \\ T(n) &= T(\lceil n/3 \rceil) + T(\lfloor 2n/3 \rfloor) + cn && \text{para } n \geq 3. \end{aligned}$$

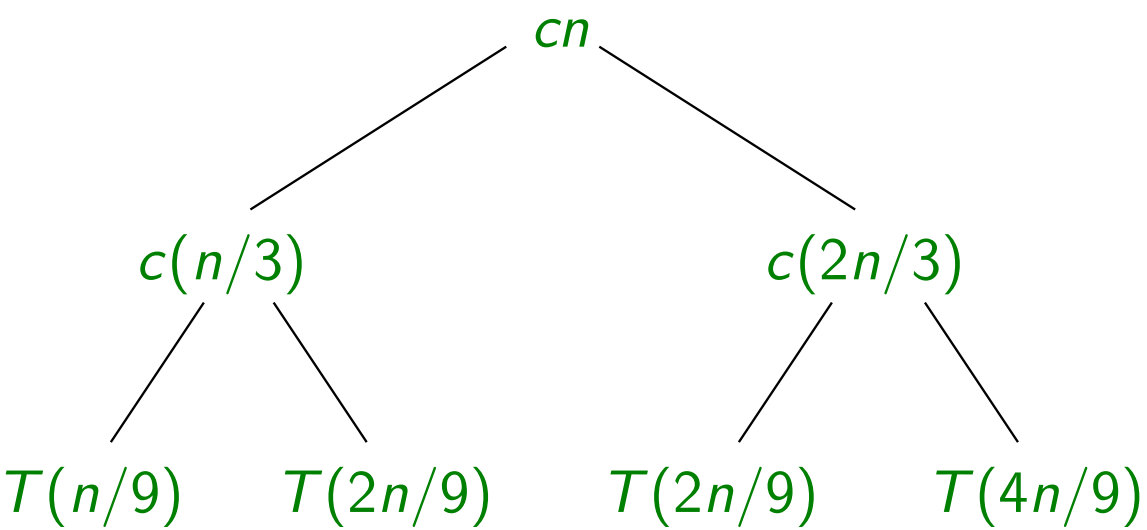
usando a árvore de recorrência.

Para simplificar as contas, vamos aproximar $\lceil n/3 \rceil$ e $\lfloor 2n/3 \rfloor$ por $n/3$ e $2n/3$, respectivamente.

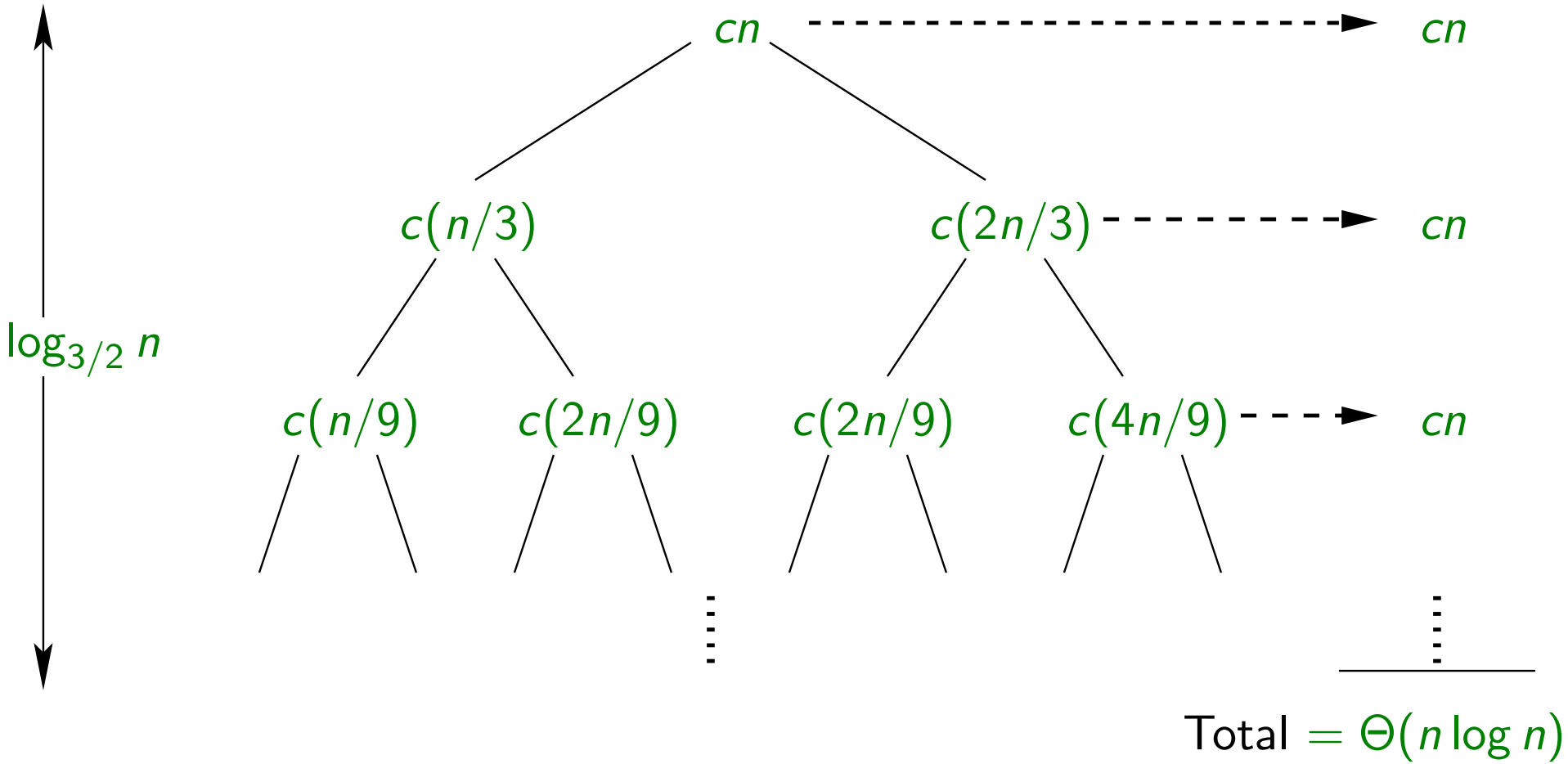
Árvore de recorrência



Árvore de recorrência



Árvore de recorrência



Árvore de recorrência

Árvore de recorrência

- No nível i o tempo gasto (sem contar as chamadas recursivas) é cn.

Árvore de recorrência

- No nível i o tempo gasto (sem contar as chamadas recursivas) é cn.
- Nem todas as folhas têm a mesma profundidade. A altura da árvore é obtida descendo pelo ramo direito (lado $2n/3$).

Árvore de recorrência

- No nível i o tempo gasto (sem contar as chamadas recursivas) é cn.
- Nem todas as folhas têm a mesma profundidade. A altura da árvore é obtida descendo pelo ramo direito (lado $2n/3$).

  Os níveis correspondem a: $n, (2/3)n, (2/3)^2 n, \ldots, (2/3)^i n, \ldots$.

  Assim, a altura é $\Theta(\log_{3/2} n) = \Theta(\log n)$.

Árvore de recorrência

- No nível i o tempo gasto (sem contar as chamadas recursivas) é cn.
- Nem todas as folhas têm a mesma profundidade. A altura da árvore é obtida descendo pelo ramo direito (lado $2n/3$).

  Os níveis correspondem a: $n, (2/3)n, (2/3)^2 n, \ldots, (2/3)^i n, \ldots$.

  Assim, a altura é $\Theta(\log_{3/2} n) = \Theta(\log n)$.
- Assim o tempo total é $cn\Theta(\log n) = \Theta(n \log n)$.

Árvore de recorrência

- No nível i o tempo gasto (sem contar as chamadas recursivas) é cn.
- Nem todas as folhas têm a mesma profundidade. A altura da árvore é obtida descendo pelo ramo direito (lado $2n/3$).

  Os níveis correspondem a: $n, (2/3)n, (2/3)^2 n, \ldots, (2/3)^i n, \ldots$.

  Assim, a altura é $\Theta(\log_{3/2} n) = \Theta(\log n)$.
- Assim o tempo total é $cn\Theta(\log n) = \Theta(n \log n)$.
- Para formalizar o resultado prove que $T(n) = \Theta(n \log n)$ usando o método de substituição. (Exercício!)

Recorrências com Θ, O, Ω à direita (CLRS)

Uma "recorrência"

$$T(n) = \Theta(1) \qquad \text{para } n = 1, 2,$$
$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2) \qquad \text{para } n \geq 3$$

Recorrências com Θ, O, Ω à direita (CLRS)

Uma "recorrência"

$$T(n) = \Theta(1) \qquad \text{para } n = 1, 2,$$
$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2) \qquad \text{para } n \geq 3$$

representa todas as recorrências da forma

$$T(n) = a \qquad \text{para } n = 1, 2,$$
$$T(n) = 3T(\lfloor n/4 \rfloor) + bn^2 \qquad \text{para } n \geq 3$$

onde a e b são constantes.

Recorrências com Θ, O, Ω à direita (CLRS)

Uma "recorrência"

$$T(n) = \Theta(1) \qquad \text{para } n = 1, 2,$$
$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2) \qquad \text{para } n \geq 3$$

representa todas as recorrências da forma

$$T(n) = a \qquad \text{para } n = 1, 2,$$
$$T(n) = 3T(\lfloor n/4 \rfloor) + bn^2 \qquad \text{para } n \geq 3$$

onde a e b são constantes.

As soluções exatas dependem dos valores de a e b, mas estão todas na mesma classe Θ.

Recorrências com Θ, O, Ω à direita (CLRS)

Uma "recorrência"

$$T(n) = \Theta(1) \qquad \text{para } n = 1, 2,$$
$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2) \qquad \text{para } n \geq 3$$

representa todas as recorrências da forma

$$T(n) = a \qquad \text{para } n = 1, 2,$$
$$T(n) = 3T(\lfloor n/4 \rfloor) + bn^2 \qquad \text{para } n \geq 3$$

onde a e b são constantes.

As soluções exatas dependem dos valores de a e b, mas estão todas na mesma classe Θ.

A "solução" é $T(n) = \Theta(n^2)$, ou seja, $T(n) \in \Theta(n^2)$.

Recorrências com Θ, O, Ω à direita (CLRS)

Uma "recorrência"

$$T(n) = \Theta(1) \qquad \text{para } n = 1, 2,$$
$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2) \qquad \text{para } n \geq 3$$

representa todas as recorrências da forma

$$T(n) = a \qquad \text{para } n = 1, 2,$$
$$T(n) = 3T(\lfloor n/4 \rfloor) + bn^2 \qquad \text{para } n \geq 3$$

onde a e b são constantes.

As soluções exatas dependem dos valores de a e b, mas estão todas na mesma classe Θ.

A "solução" é $T(n) = \Theta(n^2)$, ou seja, $T(n) \in \Theta(n^2)$.

As mesmas observações valem para as classes O, Ω, o, ω.

Recorrência do Mergesort

Podemos escrever a recorrência de tempo do MergeSort da seguinte forma

Recorrência do Mergesort

Podemos escrever a recorrência de tempo do MergeSort da seguinte forma

$$\begin{aligned} T(1) &= \Theta(1) \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) \qquad \text{para } n \geq 2. \end{aligned}$$

Recorrência do Mergesort

Podemos escrever a recorrência de tempo do MergeSort da seguinte forma

$$\begin{aligned} T(1) &= \Theta(1) \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) \qquad \text{para } n \geq 2. \end{aligned}$$

A solução da recorrência é $T(n) = \Theta(n \lg n)$.

Recorrência do Mergesort

Podemos escrever a recorrência de tempo do MergeSort da seguinte forma

$$\begin{aligned} T(1) &= \Theta(1) \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) \qquad \text{para } n \geq 2. \end{aligned}$$

A solução da recorrência é $T(n) = \Theta(n \lg n)$.

A prova é essencialmente a mesma do primeiro exemplo. (Exercício!)

Cuidados com a notação assintótica

A notação assintótica é muito versátil e expressiva. Entretanto, deve-se tomar alguns cuidados.

Cuidados com a notação assintótica

A notação assintótica é muito versátil e expressiva. Entretanto, deve-se tomar alguns cuidados.

Considere a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(\lfloor n/2 \rfloor) + n \qquad \text{para } n \geq 2. \end{aligned}$$

Cuidados com a notação assintótica

A notação assintótica é muito versátil e expressiva. Entretanto, deve-se tomar alguns cuidados.

Considere a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(\lfloor n/2 \rfloor) + n \qquad \text{para } n \geq 2. \end{aligned}$$

É similar à recorrência do MergeSort!

Cuidados com a notação assintótica

A notação assintótica é muito versátil e expressiva. Entretanto, deve-se tomar alguns cuidados.

Considere a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(\lfloor n/2 \rfloor) + n \qquad \text{para } n \geq 2. \end{aligned}$$

É similar à recorrência do MergeSort!

Mas eu vou "provar" que $T(n) = O(n)$!

Cuidados com a notação assintótica

Vou mostrar que $T(n) \leq cn$ para alguma constante $c > 0$.

Cuidados com a notação assintótica

Vou mostrar que $T(n) \leq cn$ para alguma constante $c > 0$.

$$\begin{aligned} T(n) &= 2T(\lfloor n/2 \rfloor) + n \\ &\leq 2c\lfloor n/2 \rfloor + n \\ &\leq cn + n \\ &= O(n) \end{aligned}$$

Cuidados com a notação assintótica

Vou mostrar que $T(n) \leq cn$ para alguma constante $c > 0$.

$$\begin{aligned} T(n) &= 2T(\lfloor n/2 \rfloor) + n \\ &\leq 2c\lfloor n/2 \rfloor + n \\ &\leq cn + n \\ &= O(n) \qquad \Longleftarrow \textbf{ERRADO!!!} \end{aligned}$$

Por quê?

Cuidados com a notação assintótica

Vou mostrar que $T(n) \leq cn$ para alguma constante $c > 0$.

$$\begin{aligned} T(n) &= 2T(\lfloor n/2 \rfloor) + n \\ &\leq 2c\lfloor n/2 \rfloor + n \\ &\leq cn + n \\ &= O(n) \qquad \Longleftarrow \textbf{ERRADO!!!} \end{aligned}$$

Por quê?

Não foi feito o passo indutivo, ou seja, não foi mostrado que $T(n) \leq cn$.

Teorema Master

Teorema Master

- Veremos agora um resultado que descreve soluções para recorrências da forma

$$T(n) = aT(n/b) + f(n),$$

onde $a \geq 1$ e $b > 1$ são constantes.

Teorema Master

- Veremos agora um resultado que descreve soluções para recorrências da forma

$$T(n) = aT(n/b) + f(n),$$

onde $a \geq 1$ e $b > 1$ são constantes.

- O caso base é omitido na definição e convenciona-se que é uma constante para valores pequenos.

Teorema Master

- Veremos agora um resultado que descreve soluções para recorrências da forma

$$T(n) = aT(n/b) + f(n),$$

onde $a \geq 1$ e $b > 1$ são constantes.

- O caso base é omitido na definição e convenciona-se que é uma constante para valores pequenos.
- A expressão n/b pode indicar tanto $\lfloor n/b \rfloor$ quanto $\lceil n/b \rceil$.

Teorema Master

- Veremos agora um resultado que descreve soluções para recorrências da forma

$$T(n) = aT(n/b) + f(n),$$

onde $a \geq 1$ e $b > 1$ são constantes.

- O caso base é omitido na definição e convenciona-se que é uma constante para valores pequenos.
- A expressão n/b pode indicar tanto $\lfloor n/b \rfloor$ quanto $\lceil n/b \rceil$.
- O Teorema Master **não** fornece a resposta para **todas** as recorrências da forma acima.

Teorema Master

Teorema Master (CLRS). Sejam $a \geq 1$ e $b > 1$ constantes, seja $f(n)$ uma função e seja $T(n)$ definida para os inteiros não-negativos pela relação de recorrência

$$T(n) = aT(n/b) + f(n).$$

Então $T(n)$ pode ser limitada assintoticamente da seguinte maneira:

Teorema Master

Teorema Master (CLRS). Sejam $a \geq 1$ e $b > 1$ constantes, seja $f(n)$ uma função e seja $T(n)$ definida para os inteiros não-negativos pela relação de recorrência

$$T(n) = aT(n/b) + f(n).$$

Então $T(n)$ pode ser limitada assintoticamente da seguinte maneira:

1. Se $f(n) \in O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$

Teorema Master

Teorema Master (CLRS). Sejam $a \geq 1$ e $b > 1$ constantes, seja $f(n)$ uma função e seja $T(n)$ definida para os inteiros não-negativos pela relação de recorrência

$$T(n) = aT(n/b) + f(n).$$

Então $T(n)$ pode ser limitada assintoticamente da seguinte maneira:

1. Se $f(n) \in O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$

Teorema Master

Teorema Master (CLRS). Sejam $a \geq 1$ e $b > 1$ constantes, seja $f(n)$ uma função e seja $T(n)$ definida para os inteiros não-negativos pela relação de recorrência

$$T(n) = aT(n/b) + f(n).$$

Então $T(n)$ pode ser limitada assintoticamente da seguinte maneira:

1. Se $f(n) \in O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a + \epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Exemplos de Recorrências

Exemplos onde o Teorema Master se aplica:

Exemplos de Recorrências

Exemplos onde o Teorema Master se aplica:

- Caso 1:
  $T(n) = 9T(n/3) + n$
  $T(n) = 4T(n/2) + n \log n$

Exemplos de Recorrências

Exemplos onde o Teorema Master se aplica:

- Caso 1:
  $T(n) = 9T(n/3) + n$
  $T(n) = 4T(n/2) + n \log n$
- Caso 2:
  $T(n) = T(2n/3) + 1$
  $T(n) = 2T(n/2) + (n + \log n)$

Exemplos de Recorrências

Exemplos onde o Teorema Master se aplica:

- Caso 1:
  $T(n) = 9T(n/3) + n$
  $T(n) = 4T(n/2) + n \log n$
- Caso 2:
  $T(n) = T(2n/3) + 1$
  $T(n) = 2T(n/2) + (n + \log n)$
- Caso 3:
  $T(n) = 3T(n/4) + n \log n$

Exemplos de Recorrências

Exemplos onde o Teorema Master **não se aplica**:

Exemplos de Recorrências

Exemplos onde o Teorema Master **não se aplica**:

- $T(n) = T(n-1) + n$

Exemplos de Recorrências

Exemplos onde o Teorema Master **não se aplica**:

- $T(n) = T(n-1) + n$
- $T(n) = T(n-a) + T(a) + n$, ($a \geq 1$ inteiro)

Exemplos de Recorrências

Exemplos onde o Teorema Master **não se aplica**:

- $T(n) = T(n-1) + n$
- $T(n) = T(n-a) + T(a) + n$, ($a \geq 1$ inteiro)
- $T(n) = T(\alpha n) + T((1-\alpha)n) + n$, ($0 < \alpha < 1$)

Exemplos de Recorrências

Exemplos onde o Teorema Master **não se aplica**:

- $T(n) = T(n-1) + n$
- $T(n) = T(n-a) + T(a) + n$, ($a \geq 1$ inteiro)
- $T(n) = T(\alpha n) + T((1-\alpha)n) + n$, ($0 < \alpha < 1$)
- $T(n) = T(n-1) + \log n$

Exemplos de Recorrências

Exemplos onde o Teorema Master **não se aplica**:

- $T(n) = T(n-1) + n$
- $T(n) = T(n-a) + T(a) + n$, ($a \geq 1$ inteiro)
- $T(n) = T(\alpha n) + T((1-\alpha)n) + n$, ($0 < \alpha < 1$)
- $T(n) = T(n-1) + \log n$
- $T(n) = 2T(\frac{n}{2}) + n \log n$

Teorema Master Especializado

Teorema Master (Manber). Sejam $a \geq 1$, $b > 1$ e $k \geq 0$ constantes. Seja $T(n)$ definida para os inteiros não-negativos pela relação de recorrência

$$T(n) = aT(n/b) + \Theta(n^k).$$

Então

$$T(n) = \begin{cases} \Theta(n^{\log_b a}) & \text{se } a > b^k, \\ \Theta(n^k \log n) & \text{se } a = b^k, \\ \Theta(n^k) & \text{se } a < b^k. \end{cases}$$

Exemplos de análise de complexidade

- Mostraremos a seguir exemplos de como fazer análise de complexidade de algoritmos descritos em pseudo-código.

Exemplos de análise de complexidade

- Mostraremos a seguir exemplos de como fazer análise de complexidade de algoritmos descritos em pseudo-código.
- Para algoritmos iterativos simplesmente "contamos" o tempo gasto em cada linha e depois somamos o total.

Exemplos de análise de complexidade

- Mostraremos a seguir exemplos de como fazer análise de complexidade de algoritmos descritos em pseudo-código.
- Para algoritmos iterativos simplesmente "contamos" o tempo gasto em cada linha e depois somamos o total.
- Para algoritmos recursivos é necessário obter alguma fórmula de recorrência e depois resolvê-la.

Análise de complexidade do INSERTION-SORT

```
INSERTION-SORT(A, n)
1   para j ← 2 até n faça
2       chave ← A[j]
3       ▷ Insere A[j] no subvetor ordenado A[1 . . . j − 1]
4       i ← j − 1
5       enquanto i ≥ 1  e A[i] > chave faça
6           A[i + 1] ← A[i]
7           i ← i − 1
8       A[i + 1] ← chave
```

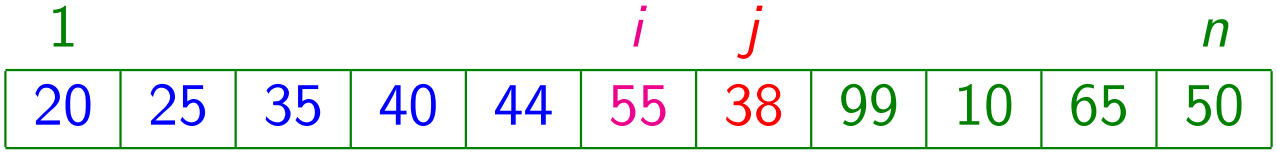
Vamos analisar a complexidade de tempo de pior caso de INSERTION-SORT.

Insertion sort – iteração genérica

$chave = 38$

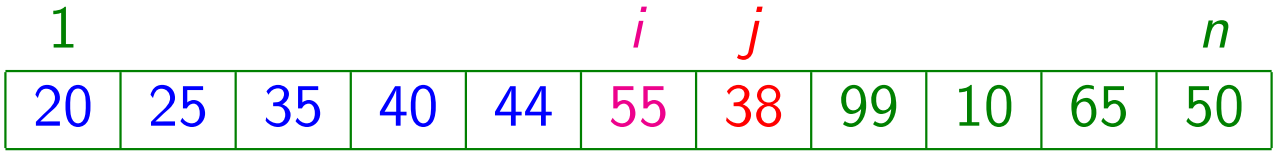
Insertion sort – iteração genérica

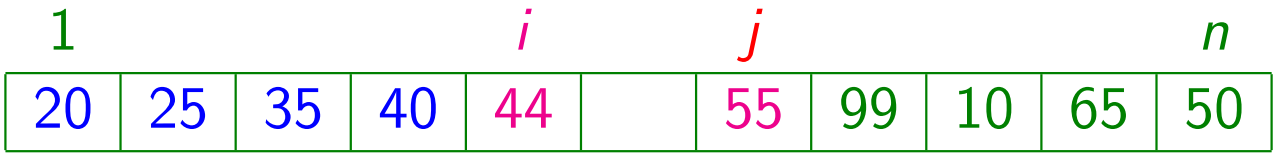
$chave = 38$



Insertion sort – iteração genérica

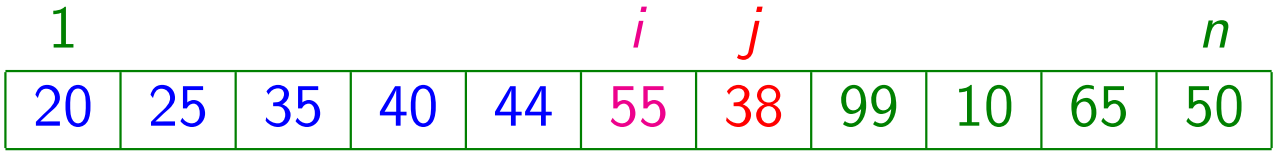
$chave = 38$

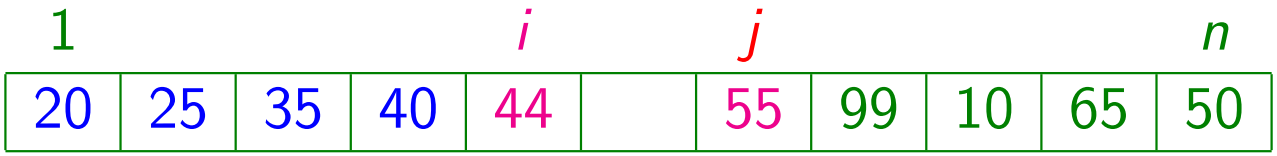


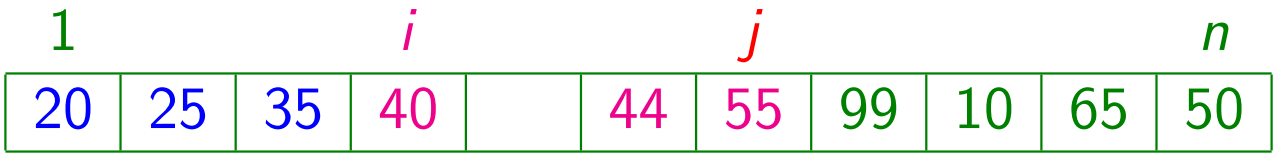


Insertion sort – iteração genérica

$chave = 38$

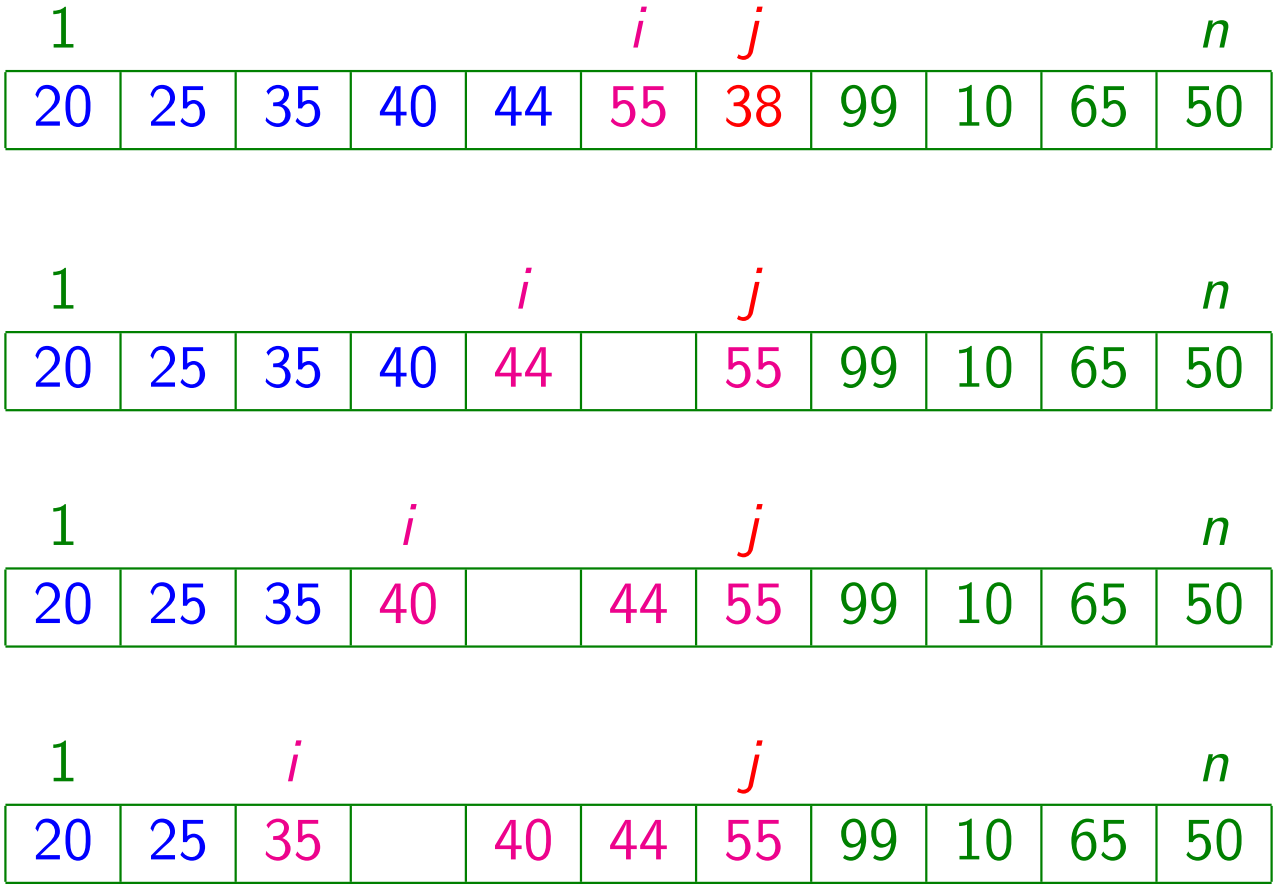






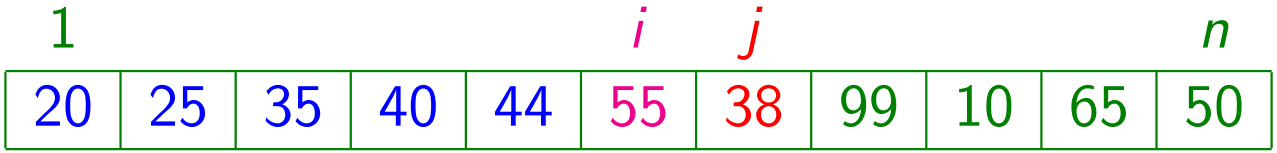
Insertion sort – iteração genérica

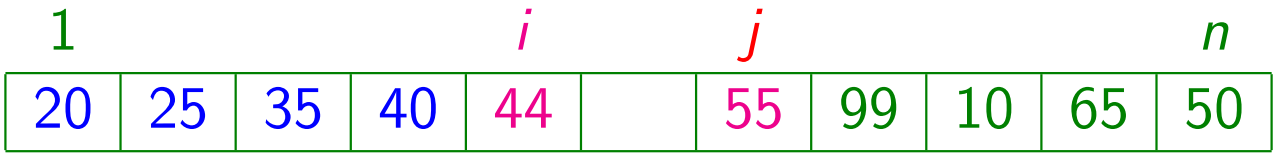
$chave = 38$

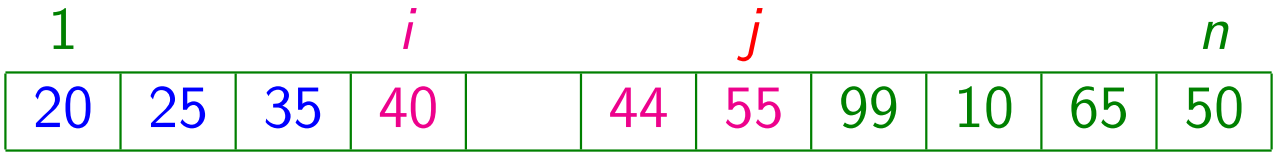


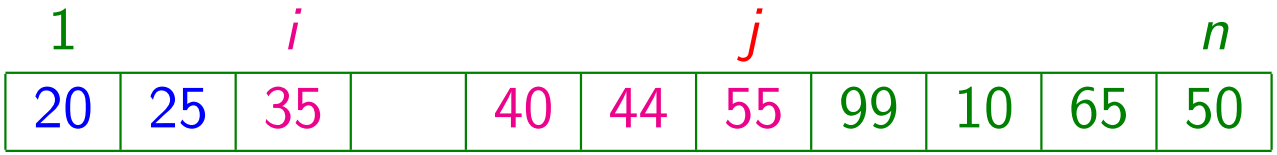
Insertion sort – iteração genérica

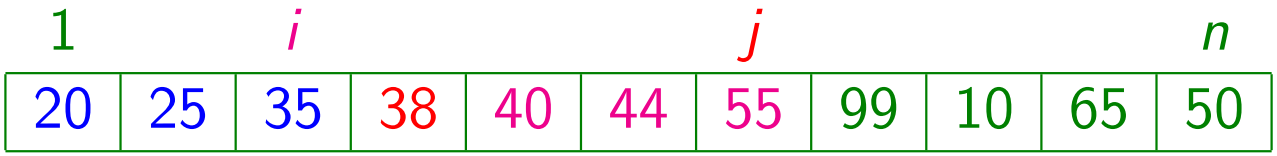
$chave = 38$





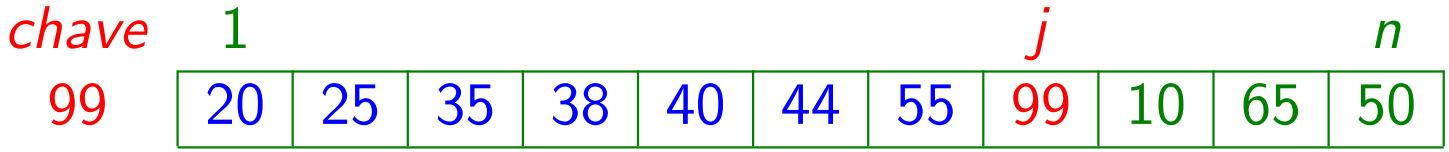




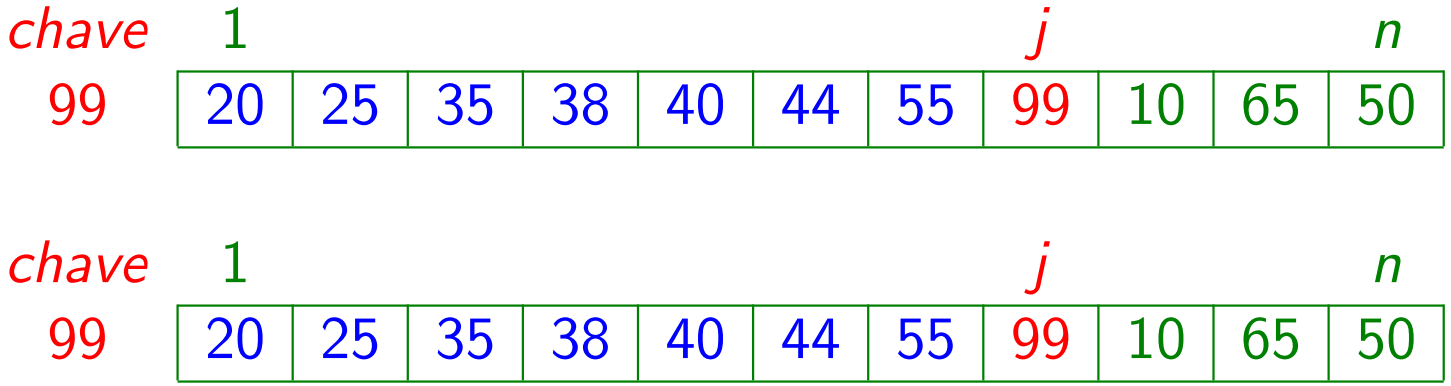


Insertion sort

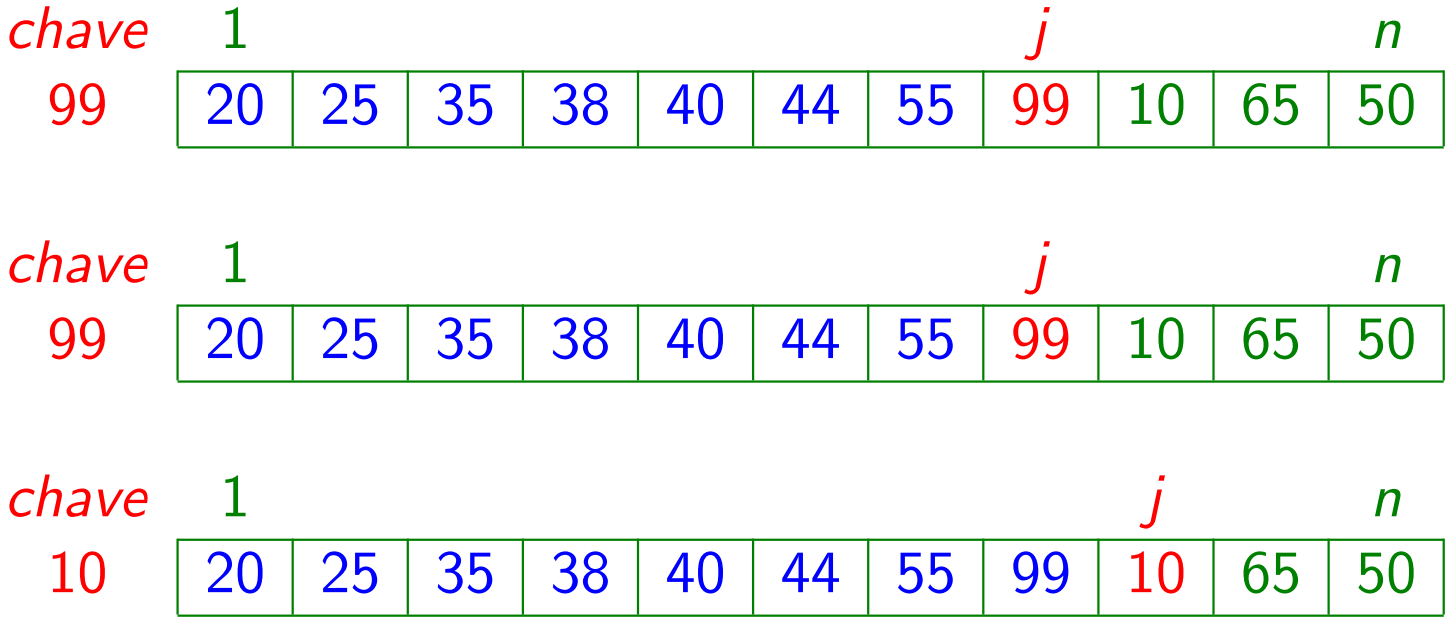
Insertion sort



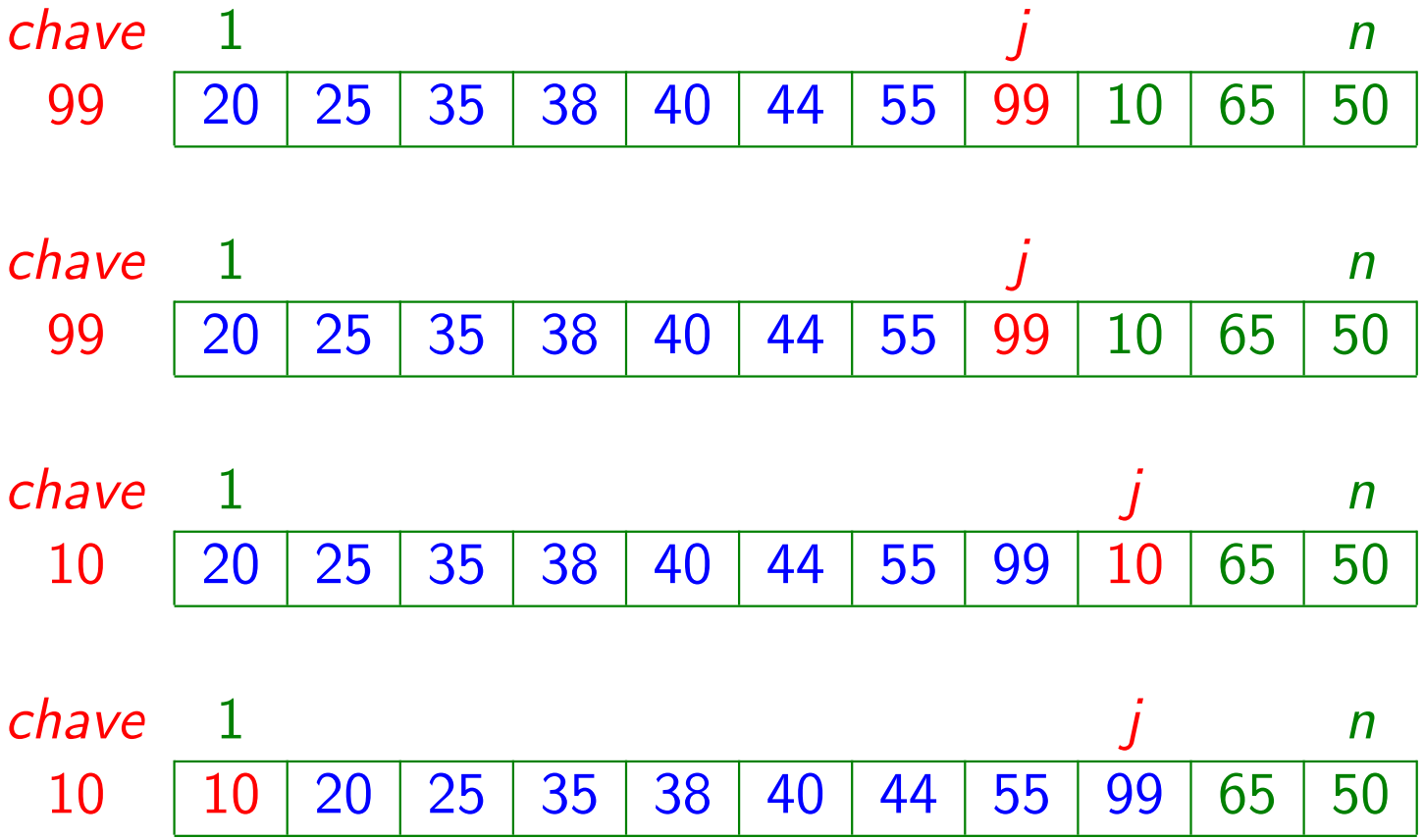
Insertion sort



Insertion sort

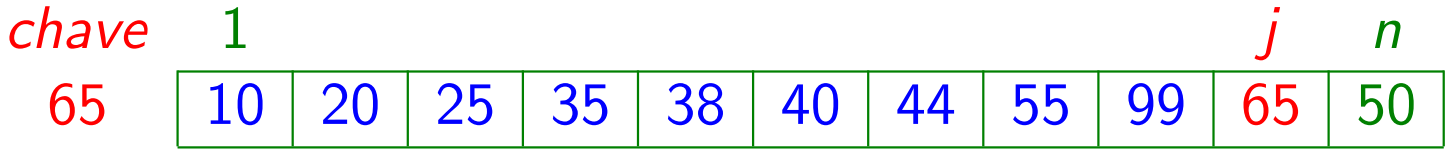


Insertion sort

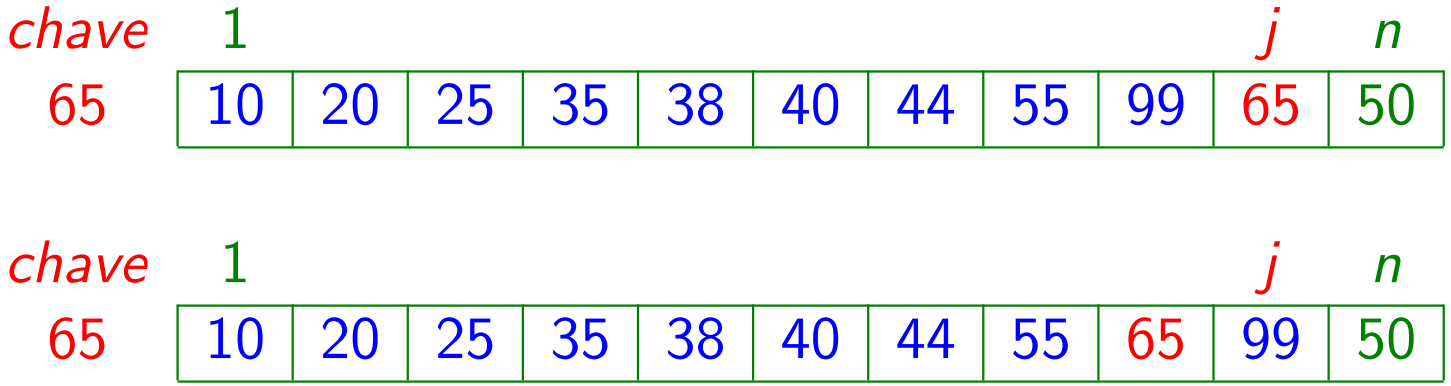


Insertion sort

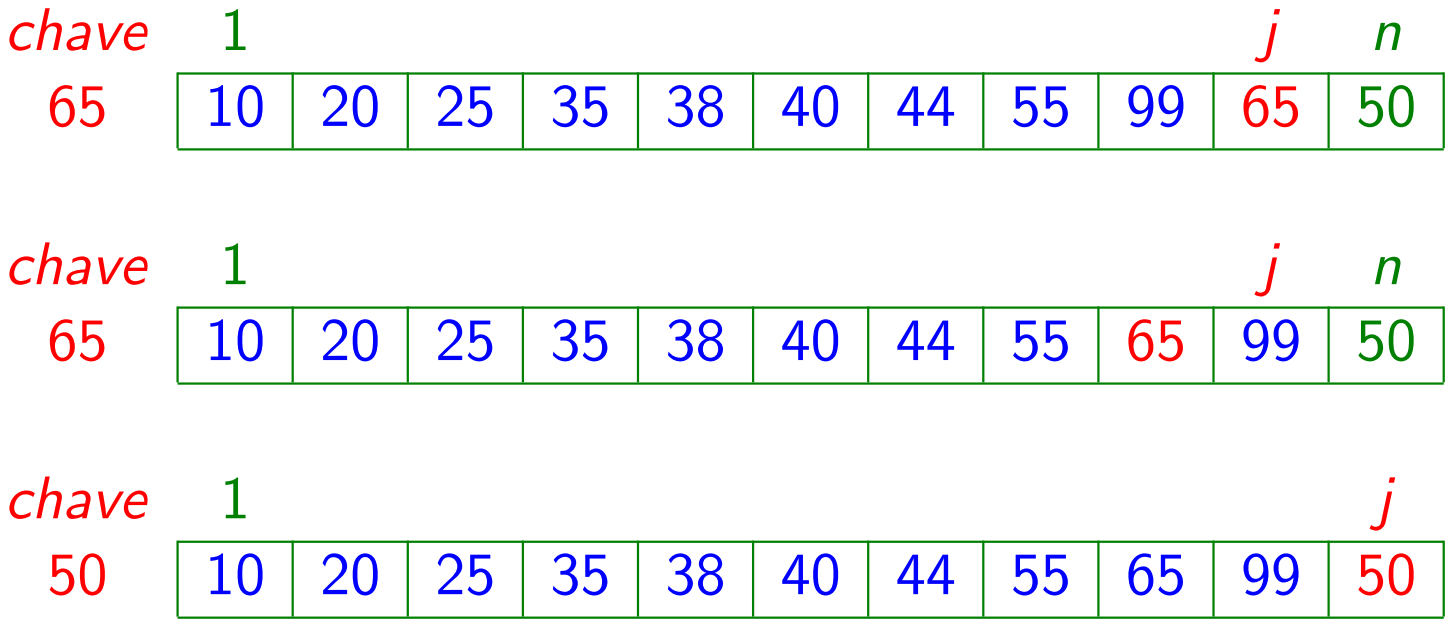
Insertion sort



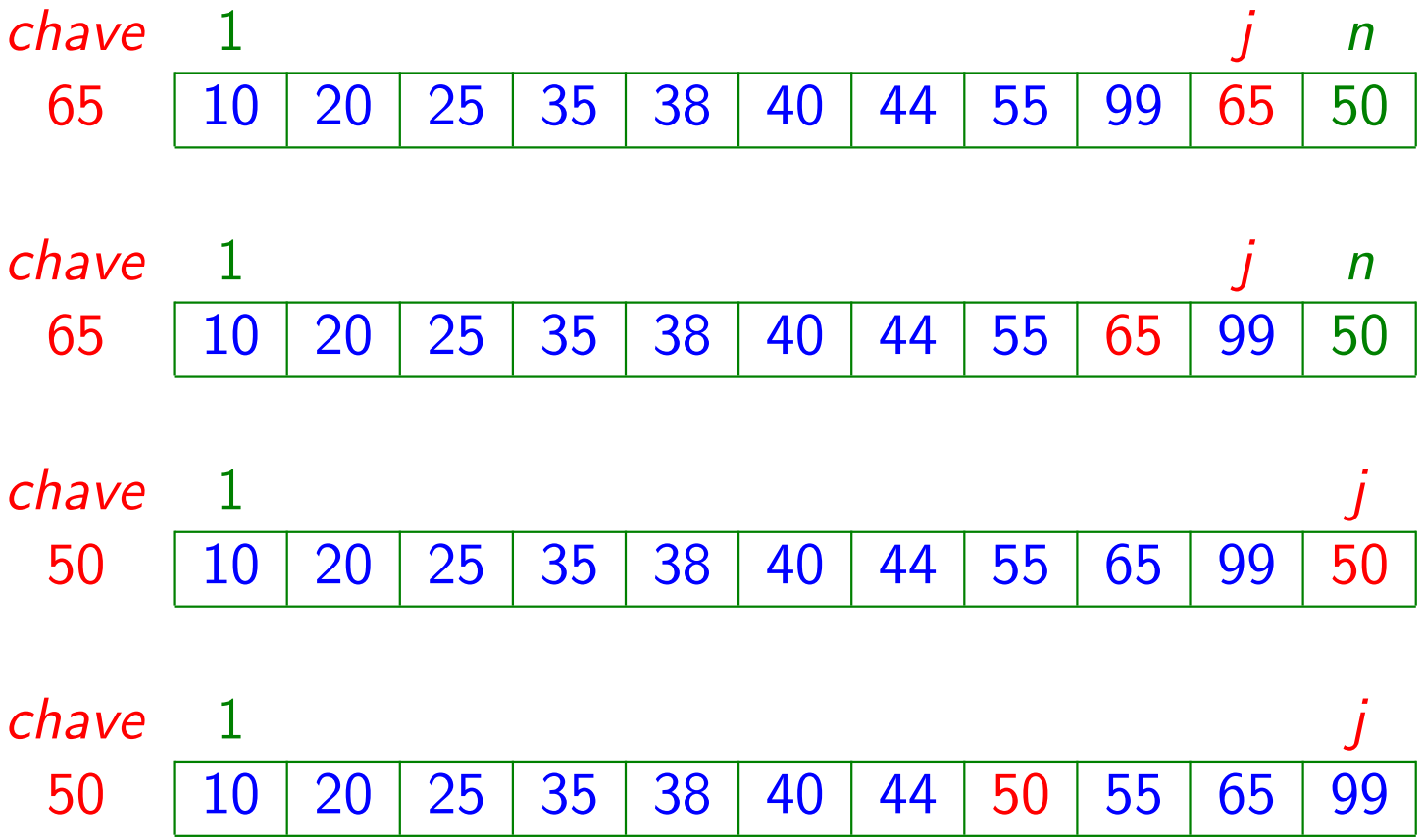
Insertion sort



Insertion sort



Insertion sort



Análise de complexidade do INSERTION-SORT

INSERTION-SORT(A, n)	Tempo
1 **para** $j \leftarrow 2$ **até** n **faça**	?
2 $\quad$ chave $\leftarrow A[j]$	?
3 $\quad \triangleright$ Insere $A[j]$ em $A[1\,..\,j-1]$	?
4 $\quad i \leftarrow j-1$	?
5 $\quad$ **enquanto** $i \geq 1$ **e** $A[i] >$ chave **faça**	?
6 $\quad\quad A[i+1] \leftarrow A[i]$	?
7 $\quad\quad i \leftarrow i-1$	?
8 $\quad A[i+1] \leftarrow$ chave	?
Total	?

Método típico de análise de um algoritmo iterativo.
Preenchemos em cada linha na coluna à direita o tempo gasto durante a execução do algoritmo. Depois calculamos a soma (ou o máximo) das entradas para obter a complexidade total do algoritmo.

Complexidade de InsertionSort

INSERTION-SORT(A, n)	Tempo
1 **para** $j \leftarrow 2$ **até** n **faça**	$\Theta(n)$
2 $\quad$ chave $\leftarrow A[j]$	$\Theta(n)$
3 $\quad \triangleright$ Insere $A[j]$ em $A[1..j-1]$	0
4 $\quad i \leftarrow j-1$	$\Theta(n)$
5 $\quad$ **enquanto** $i \geq 1$ **e** $A[i] >$ chave **faça**	$\Theta(n).O(n)$
6 $\quad\quad A[i+1] \leftarrow A[i]$	$\Theta(n).O(n)$
7 $\quad\quad i \leftarrow i-1$	$\Theta(n).O(n)$
8 $\quad A[i+1] \leftarrow$ chave	$\Theta(n)$
Total	$O(n^2)$

Note que em alguns pontos usamos $\Theta(\ )$ e em outros $O(\ )$.
Por quê?

Complexidade de Potência

Entrada: real a e inteiro $d \geq 0$.
Saída: a^d.

```
Potência(a, d)
1   y ← a, n ← d, x ← 1
2   enquanto n > 0 faça
3       se n é ímpar então x ← xy
4       n ← ⌊n/2⌋
5       y ← y²
6   devolva x
```

Complexidade de Potência

Potência(a, d)		Tempo
1	$y \leftarrow a$, $n \leftarrow d$, $x \leftarrow 1$	?
2	**enquanto** $n > 0$ **faça**	?
3	**se** n é ímpar **então** $x \leftarrow xy$	?
4	$n \leftarrow \lfloor n/2 \rfloor$	?
5	$y \leftarrow y^2$	?
6	**devolva** x	?
	Total	?

Quantas vezes a linha 2 é executada?

Complexidade de Potência

Potência(a, d)		Tempo
1	$y \leftarrow a$, $n \leftarrow d$, $x \leftarrow 1$	?
2	**enquanto** $n > 0$ **faça**	?
3	**se** n é ímpar **então** $x \leftarrow xy$	?
4	$n \leftarrow \lfloor n/2 \rfloor$	?
5	$y \leftarrow y^2$	?
6	**devolva** x	?
	Total	?

Quantas vezes a linha 2 é executada? $\Theta(\lg d)$

Complexidade de Potência

	Potência(a, d)		Tempo
1	$y \leftarrow a$, $n \leftarrow d$, $x \leftarrow 1$		$\Theta(1)$
2	**enquanto** $n > 0$ **faça**		$\Theta(\lg d)$
3	**se** n é ímpar **então**	$x \leftarrow xy$	$\Theta(\lg d)$
4	$n \leftarrow \lfloor n/2 \rfloor$		$\Theta(\lg d)$
5	$y \leftarrow y^2$		$\Theta(\lg d)$
6	**devolva** x		$\Theta(1)$
		Total	$\Theta(\lg d)$

Complexidade de Mistério

Entrada: um vetor $A[p\,..\,r]$ de reais.
Saída: é um mistério...

```
MISTÉRIO(A, p, r)
1   n ← r − p + 1
2   se n ≥ 3 faça
3       q ← p + ⌊n/3⌋
4       ARRUMA(A, p, q, r)
5       MISTÉRIO(A, p, q − 1)
6       MISTÉRIO(A, q, r)
```

Suponha que $\textsc{Arruma}(A, p, q, r)$ tenha complexidade de tempo $O(n)$.

Complexidade de Mistério

MISTÉRIO(A, p, r)		Tempo
1	$n \leftarrow r - p + 1$	?
2	**se** $n \geq 3$ **faça**	?
3	$q \leftarrow p + \lfloor n/3 \rfloor$	?
4	ARRUMA(A, p, q, r)	?
5	MISTÉRIO$(A, p, q - 1)$	?
6	MISTÉRIO(A, q, r)	?
	Total	?

Suponha que ARRUMA(A, p, q, r) tenha complexidade de tempo $O(n)$.

Complexidade de Mistério

Mistério(A, p, r)	Tempo
1 $n \leftarrow r - p + 1$	$\Theta(1)$
2 **se** $n \geq 3$ **faça**	$\Theta(1)$
3 $q \leftarrow p + \lfloor n/3 \rfloor$	$O(1)$
4 Arruma(A, p, q, r)	$O(n)$
5 Mistério$(A, p, q-1)$	$T(\lfloor n/3 \rfloor)$
6 Mistério(A, q, r)	$T(\lceil 2n/3 \rceil)$
	$T(n) = T(\lfloor n/3 \rfloor) + T(\lceil 2n/3 \rceil) + O(n)$

A solução de $T(n) = T(\lfloor n/3 \rfloor) + T(\lceil 2n/3 \rceil) + O(n)$ é $T(n) = O(n \lg n)$ (Exercício!).

Note que não dá para usar o Teorema Master...

Observação: $n = \lfloor n/3 \rfloor + \lceil 2n/3 \rceil$ (Exercício!).