

Aula 06

Recursividade

Introdução ao Conceito

Programação II, 2018-2019

v1.9, 25-03-2018

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- ➊ Introdução
- ➋ Definição
- ➌ Complexidade
- ➍ Relação de Recorrência
- ➎ Exemplo 1: A Função Factorial
- ➏ Relação de Recorrência: Síntese
- ➐ Exemplo 2: Cálculo das Combinações
- ➑ Relação de Recorrência: Classificação
- ➒ Exemplo 3: Torres de Hanói
- ➓ Definição Recursiva: Condições de Sanidade
 - Casos Atípicos
 - Casos com Interesse

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- ➊ Introdução
- ➋ Definição
- ➌ Complexidade
- ➍ Relação de Recorrência
- ➎ Exemplo 1: A Função Factorial
- ➏ Relação de Recorrência: Síntese
- ➐ Exemplo 2: Cálculo das Combinações
- ➑ Relação de Recorrência: Classificação
- ➒ Exemplo 3: Torres de Hanói
- ➓ Definição Recursiva: Condições de Sanidade
 - Casos Atípicos
 - Casos com Interesse

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse



- Se tivesse de descrever a alguém o que é uma boneca *matryoshka*, como o faria?
- Uma possibilidade seria dizer que é uma boneca oca que contém outra boneca oca, que contém outra e assim sucessivamente.
- Podemos fazer uso de uma definição alternativa que talvez nos facilite a resposta:
Uma boneca *matryoshka* é uma boneca oca que contém uma boneca *matryoshka*.
- Este é um exemplo de uma **definição recursiva**.

Introdução

Definição

Complexidade

Relação de Recorrência

Exemplo 1: A Função Factorial

Relação de Recorrência: Síntese

Exemplo 2: Cálculo das Combinações

Relação de Recorrência: Classificação

Exemplo 3: Torres de Hanói

Definição Recursiva: Condições de Sanidade

Casos Atípicos

Casos com Interesse



- Se tivesse de descrever a alguém o que é uma boneca *matryoshka*, como o faria?
- Uma possibilidade seria dizer que é uma boneca oca que contém outra boneca oca, que contém outra e assim sucessivamente.
- Podemos fazer uso de uma definição alternativa que talvez nos facilite a resposta:
 - Uma boneca *matryoshka* é uma boneca oca que contém outra boneca *matryoshka*.
- Este é um exemplo de uma **definição recursiva**.

Introdução

Definição

Complexidade

Relação de Recorrência

Exemplo 1: A Função Factorial

Relação de Recorrência: Síntese

Exemplo 2: Cálculo das Combinações

Relação de Recorrência: Classificação

Exemplo 3: Torres de Hanói

Definição Recursiva: Condições de Sanidade

Casos Atípicos

Casos com Interesse



- Se tivesse de descrever a alguém o que é uma boneca *matryoshka*, como o faria?
- Uma possibilidade seria dizer que é uma boneca oca que contém outra boneca oca, que contém outra e assim sucessivamente.
- Podemos fazer uso de uma definição alternativa que talvez nos facilite a resposta:
 - Uma boneca *matryoshka* é uma boneca oca que contém outra boneca *matryoshka*.
- Este é um exemplo de uma **definição recursiva**.

Introdução

Definição

Complexidade

Relação de Recorrência

Exemplo 1: A Função Factorial

Relação de Recorrência: Síntese

Exemplo 2: Cálculo das Combinações

Relação de Recorrência: Classificação

Exemplo 3: Torres de Hanói

Definição Recursiva: Condições de Sanidade

Casos Atípicos

Casos com Interesse



- Se tivesse de descrever a alguém o que é uma boneca *matryoshka*, como o faria?
- Uma possibilidade seria dizer que é uma boneca oca que contém outra boneca oca, que contém outra e assim sucessivamente.
- Podemos fazer uso de uma definição alternativa que talvez nos facilite a resposta:
 - Uma boneca *matryoshka* é uma boneca oca que contém outra boneca *matryoshka*.
- Este é um exemplo de uma **definição recursiva**.

Introdução

Definição

Complexidade

Relação de Recorrência

Exemplo 1: A Função Factorial

Relação de Recorrência: Síntese

Exemplo 2: Cálculo das Combinações

Relação de Recorrência: Classificação

Exemplo 3: Torres de Hanói

Definição Recursiva: Condições de Sanidade

Casos Atípicos

Casos com Interesse



- Se tivesse de descrever a alguém o que é uma boneca *matryoshka*, como o faria?
- Uma possibilidade seria dizer que é uma boneca oca que contém outra boneca oca, que contém outra e assim sucessivamente.
- Podemos fazer uso de uma definição alternativa que talvez nos facilite a resposta:
 - Uma boneca *matryoshka* é uma boneca oca que contém outra boneca *matryoshka*.
- Este é um exemplo de uma **definição recursiva**.

Introdução

Definição

Complexidade

Relação de Recorrência

Exemplo 1: A Função Factorial

Relação de Recorrência: Síntese

Exemplo 2: Cálculo das Combinações

Relação de Recorrência: Classificação

Exemplo 3: Torres de Hanói

Definição Recursiva: Condições de Sanidade

Casos Atípicos

Casos com Interesse



- Se tivesse de descrever a alguém o que é uma boneca *matryoshka*, como o faria?
- Uma possibilidade seria dizer que é uma boneca oca que contém outra boneca oca, que contém outra e assim sucessivamente.
- Podemos fazer uso de uma definição alternativa que talvez nos facilite a resposta:
 - Uma boneca *matryoshka* é uma boneca oca que contém outra boneca *matryoshka*.
- Este é um exemplo de uma **definição recursiva**.

Introdução

Definição

Complexidade

Relação de Recorrência

Exemplo 1: A Função Factorial

Relação de Recorrência: Síntese

Exemplo 2: Cálculo das Combinações

Relação de Recorrência: Classificação

Exemplo 3: Torres de Hanói

Definição Recursiva: Condições de Sanidade

Casos Atípicos

Casos com Interesse

Definição

Definição Recursiva

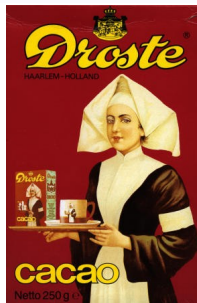
Uma definição de um conceito diz-se recursiva se envolver uma ou mais instâncias do próprio conceito.

Recursividade

Se ainda não entendeu, ver *recursividade*.

Podemos encontrar recursividade um pouco por todo o lado:

- Na descrição das árvores genealógicas.
- Nas imagens de espelhos paralelos.
- Na sintaxe das linguagens de programação.
- ...



(circa 1904)

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Definição

Definição Recursiva

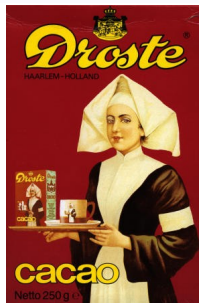
Uma definição de um conceito diz-se recursiva se envolver uma ou mais instâncias do próprio conceito.

Recursividade

Se ainda não entendeu, ver *recursividade*.

Podemos encontrar recursividade um pouco por todo o lado:

- Na descrição das árvores genealógicas.
- Nas imagens de espelhos paralelos.
- Na sintaxe das linguagens de programação.
- ...



(circa 1904)

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Definição

Definição Recursiva

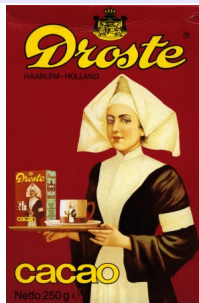
Uma definição de um conceito diz-se recursiva se envolver uma ou mais instâncias do próprio conceito.

Recursividade

Se ainda não entendeu, ver *recursividade*.

Podemos encontrar recursividade um pouco por todo o lado:

- Na descrição das árvores genealógicas.
- Nas imagens de espelhos paralelos.
- Na sintaxe das linguagens de programação.
- ...



(circa 1904)

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Definição

Definição Recursiva

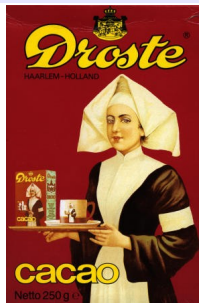
Uma definição de um conceito diz-se recursiva se envolver uma ou mais instâncias do próprio conceito.

Recursividade

Se ainda não entendeu, ver *recursividade*.

Podemos encontrar recursividade um pouco por todo o lado:

- Na descrição das árvores genealógicas.
- Nas imagens de espelhos paralelos.
- Na sintaxe das linguagens de programação.
- ...



(circa 1904)

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Definição

Definição Recursiva

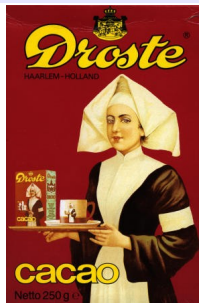
Uma definição de um conceito diz-se recursiva se envolver uma ou mais instâncias do próprio conceito.

Recursividade

Se ainda não entendeu, ver *recursividade*.

Podemos encontrar recursividade um pouco por todo o lado:

- Na descrição das árvores genealógicas.
- Nas imagens de espelhos paralelos.
- Na sintaxe das linguagens de programação.
- ...



(circa 1904)

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Definição Recursiva

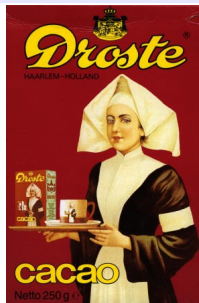
Uma definição de um conceito diz-se recursiva se envolver uma ou mais instâncias do próprio conceito.

Recursividade

Se ainda não entendeu, ver *recursividade*.

Podemos encontrar recursividade um pouco por todo o lado:

- Na descrição das árvores genealógicas.
- Nas imagens de espelhos paralelos.
- Na sintaxe das linguagens de programação.
- ...



(circa 1904)

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos
Casos com Interesse

Definição Recursiva

Uma definição de um conceito diz-se recursiva se envolver uma ou mais instâncias do próprio conceito.

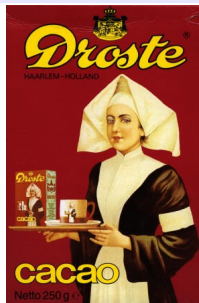
Recursividade

Se ainda não entendeu, ver *recursividade*.

Podemos encontrar recursividade um pouco por todo o lado:

- Na descrição das árvores genealógicas.
- Nas imagens de espelhos paralelos.
- Na sintaxe das linguagens de programação.

• . . .



(circa 1904)

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Definição

Definição Recursiva

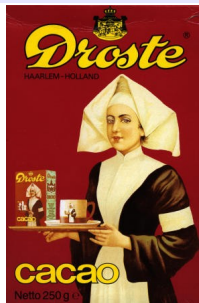
Uma definição de um conceito diz-se recursiva se envolver uma ou mais instâncias do próprio conceito.

Recursividade

Se ainda não entendeu, ver *recursividade*.

Podemos encontrar recursividade um pouco por todo o lado:

- Na descrição das árvores genealógicas.
- Nas imagens de espelhos paralelos.
- Na sintaxe das linguagens de programação.
- ...



(circa 1904)

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos
Casos com Interesse

- Como veremos, as definições recursivas podem também aparecer nos dois aspectos essenciais da programação:
 - nas definições de problemas
 - nos algoritmos.
- Tal como nos exemplos apresentados, a justificação para a sua utilização é a **simplicidade** que ela por vezes nos dá na descrição de problemas complexos.
- Desde Programação 1 temos vindo a apresentar e aplicar tecnologias e métodos para controlar a complexidade inerente à resolução de problemas.
- Uma característica comum à maioria delas é o facto de **reduzirem a redundância** do código necessário para a solução.
- A estratégia tem sido tirar proveito das **semelhanças formais** entre as várias partes do código.

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos
Casos com Interesse

- Como veremos, as definições recursivas podem também aparecer nos dois aspectos essenciais da programação:
 - nas **estruturas de dados**;
 - nos **algoritmos**.
- Tal como nos exemplos apresentados, a justificação para a sua utilização é a **simplicidade** que ela por vezes nos dá na descrição de problemas complexos.
- Desde Programação 1 temos vindo a apresentar e aplicar tecnologias e métodos para controlar a complexidade inerente à resolução de problemas.
- Uma característica comum à maioria delas é o facto de **reduzirem a redundância** do código necessário para a solução.
- A estratégia tem sido tirar proveito das **semelhanças formais** entre as várias partes do código.

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- Como veremos, as definições recursivas podem também aparecer nos dois aspectos essenciais da programação:
 - nas **estruturas de dados**;
 - nos **algoritmos**.
- Tal como nos exemplos apresentados, a justificação para a sua utilização é a **simplicidade** que ela por vezes nos dá na descrição de problemas complexos.
- Desde Programação 1 temos vindo a apresentar e aplicar tecnologias e métodos para controlar a complexidade inerente à resolução de problemas.
- Uma característica comum à maioria delas é o facto de **reduzirem a redundância** do código necessário para a solução.
- A estratégia tem sido tirar proveito das **semelhanças formais** entre as várias partes do código.

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- Como veremos, as definições recursivas podem também aparecer nos dois aspectos essenciais da programação:
 - nas **estruturas de dados**;
 - nos **algoritmos**.
- Tal como nos exemplos apresentados, a justificação para a sua utilização é a **simplicidade** que ela por vezes nos dá na descrição de problemas complexos.
- Desde Programação 1 temos vindo a apresentar e aplicar tecnologias e métodos para controlar a complexidade inerente à resolução de problemas.
- Uma característica comum à maioria delas é o facto de **reduzirem a redundância** do código necessário para a solução.
- A estratégia tem sido tirar proveito das **semelhanças formais** entre as várias partes do código.

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- Como veremos, as definições recursivas podem também aparecer nos dois aspectos essenciais da programação:
 - nas **estruturas de dados**;
 - nos **algoritmos**.
- Tal como nos exemplos apresentados, a justificação para a sua utilização é a **simplicidade** que ela por vezes nos dá na descrição de problemas complexos.
- Desde Programação 1 temos vindo a apresentar e aplicar tecnologias e métodos para controlar a complexidade inerente à resolução de problemas.
- Uma característica comum à maioria delas é o facto de **reduzirem a redundância** do código necessário para a solução.
- A estratégia tem sido tirar proveito das **semelhanças formais** entre as várias partes do código.

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- Como veremos, as definições recursivas podem também aparecer nos dois aspectos essenciais da programação:
 - nas **estruturas de dados**;
 - nos **algoritmos**.
- Tal como nos exemplos apresentados, a justificação para a sua utilização é a **simplicidade** que ela por vezes nos dá na descrição de problemas complexos.
- Desde Programação 1 temos vindo a apresentar e aplicar tecnologias e métodos para controlar a complexidade inerente à resolução de problemas.
- Uma característica comum à maioria delas é o facto de **reduzirem a redundância** do código necessário para a solução.
- A estratégia tem sido tirar proveito das **semelhanças formais** entre as várias partes do código.

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- Como veremos, as definições recursivas podem também aparecer nos dois aspectos essenciais da programação:
 - nas **estruturas de dados**;
 - nos **algoritmos**.
- Tal como nos exemplos apresentados, a justificação para a sua utilização é a **simplicidade** que ela por vezes nos dá na descrição de problemas complexos.
- Desde Programação 1 temos vindo a apresentar e aplicar tecnologias e métodos para controlar a complexidade inerente à resolução de problemas.
- Uma característica comum à maioria delas é o facto de **reduzirem a redundância** do código necessário para a solução.
- A estratégia tem sido tirar proveito das **semelhanças formais** entre as várias partes do código.

- Como veremos, as definições recursivas podem também aparecer nos dois aspectos essenciais da programação:
 - nas **estruturas de dados**;
 - nos **algoritmos**.
- Tal como nos exemplos apresentados, a justificação para a sua utilização é a **simplicidade** que ela por vezes nos dá na descrição de problemas complexos.
- Desde Programação 1 temos vindo a apresentar e aplicar tecnologias e métodos para controlar a complexidade inerente à resolução de problemas.
- Uma característica comum à maioria delas é o facto de **reduzirem a redundância** do código necessário para a solução.
- A estratégia tem sido tirar proveito das **semelhanças formais** entre as várias partes do código.

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Vejamos alguns casos:

- **Variáveis:** as variáveis permitem que o mesmo código seja parametrizável para diferentes valores.
- **Instrução iterativa:** sempre que existe uma repetição de comandos estruturalmente semelhantes, os mesmos podem ser expressos como a repetição de um único comando (recorrendo muitas vezes ao uso de variáveis auxiliares).
- **Funções:** a semelhança formal algorítmica de certas operações pode ser abstraída e modularizada numa função. Há uma separação clara entre a utilização da função e a respectiva implementação. Quem a utiliza, delega a responsabilidade da resolução na função. Quem a implementa, pode livremente escolher o melhor algoritmo.

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos
Casos com Interesse

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos
Casos com Interesse

Vejamos alguns casos:

- **Variáveis:** as variáveis permitem que o mesmo código seja parametrizável para diferentes valores.
- **Instrução iterativa:** sempre que existe uma repetição de comandos estruturalmente semelhantes, os mesmos podem ser expressos como a repetição de um único comando (recorrendo muitas vezes ao uso de variáveis auxiliares).
- **Funções:** a semelhança formal algorítmica de certas operações pode ser abstraída e modularizada numa função. Há uma separação clara entre a utilização da função e a respectiva implementação. Quem a utiliza, delega a responsabilidade da resolução na função. Quem a implementa, pode livremente escolher o melhor algoritmo.

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos
Casos com Interesse

Vejamos alguns casos:

- **Variáveis:** as variáveis permitem que o mesmo código seja parametrizável para diferentes valores.
- **Instrução iterativa:** sempre que existe uma repetição de comandos estruturalmente semelhantes, os mesmos podem ser expressos como a repetição de um único comando (recorrendo muitas vezes ao uso de variáveis auxiliares).
- **Funções:** a semelhança formal algorítmica de certas operações pode ser abstraída e modularizada numa função. Há uma separação clara entre a utilização da função e a respectiva implementação. Quem a utiliza, delega a responsabilidade da resolução na função. Quem a implementa, pode livremente escolher o melhor algoritmo.

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Vejamos alguns casos:

- **Variáveis:** as variáveis permitem que o mesmo código seja parametrizável para diferentes valores.
- **Instrução iterativa:** sempre que existe uma repetição de comandos estruturalmente semelhantes, os mesmos podem ser expressos como a repetição de um único comando (recorrendo muitas vezes ao uso de variáveis auxiliares).
- **Funções:** a semelhança formal algorítmica de certas operações pode ser abstraída e modularizada numa função. Há uma separação clara entre a utilização da função e a respectiva implementação. Quem a utiliza, delega a responsabilidade da resolução na função. Quem a implementa, pode livremente escolher o melhor algoritmo.

Vejamos alguns casos:

- **Variáveis:** as variáveis permitem que o mesmo código seja parametrizável para diferentes valores.
- **Instrução iterativa:** sempre que existe uma repetição de comandos estruturalmente semelhantes, os mesmos podem ser expressos como a repetição de um único comando (recorrendo muitas vezes ao uso de variáveis auxiliares).
- **Funções:** a semelhança formal algorítmica de certas operações pode ser abstraída e modularizada numa função. Há uma separação clara entre a **utilização** da função e a respectiva **implementação**. Quem a utiliza, delega a responsabilidade da resolução na função. Quem a implementa, pode livremente escolher o melhor algoritmo.

- O caso das funções é particularmente interessante. Se quem as **implementa** é livre para escolher o melhor algoritmo, porque não escolher um que **utiliza** a própria função?
- Se o problema se presta a ser descrito recursivamente, então porque não implementá-lo da mesma forma?
- Para se poder fazer isso mesmo torna-se necessário ter uma descrição recursiva formal do problema: esse é o papel das **Relações de Recorrência**.
- Uma relação de recorrência é uma formulação recursiva formal de um problema.
- As relações de recorrência podem ser sempre implementadas de uma forma **iterativa** ou de uma forma **recursiva**.
- A implementação recursiva é estruturalmente muito próxima da própria relação de recorrência (donde resulta a sua simplicidade).

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- O caso das funções é particularmente interessante. Se quem as **implementa** é livre para escolher o melhor algoritmo, porque não escolher um que **utiliza** a própria função?
- Se o problema se presta a ser descrito recursivamente, então porque não implementá-lo da mesma forma?
- Para se poder fazer isso mesmo torna-se necessário ter uma descrição recursiva formal do problema: esse é o papel das **Relações de Recorrência**.
- Uma relação de recorrência é uma formulação recursiva formal de um problema.
- As relações de recorrência podem ser sempre implementadas de uma forma **iterativa** ou de uma forma **recursiva**.
- A implementação recursiva é estruturalmente muito próxima da própria relação de recorrência (donde resulta a sua simplicidade).

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- O caso das funções é particularmente interessante. Se quem as **implementa** é livre para escolher o melhor algoritmo, porque não escolher um que **utiliza** a própria função?
- Se o problema se presta a ser descrito recursivamente, então porque não implementá-lo da mesma forma?
- Para se poder fazer isso mesmo torna-se necessário ter uma descrição recursiva formal do problema: esse é o papel das **Relações de Recorrência**.
- Uma relação de recorrência é uma formulação recursiva formal de um problema.
- As relações de recorrência podem ser sempre implementadas de uma forma **iterativa** ou de uma forma **recursiva**.
- A implementação recursiva é estruturalmente muito próxima da própria relação de recorrência (donde resulta a sua simplicidade).

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- O caso das funções é particularmente interessante. Se quem as **implementa** é livre para escolher o melhor algoritmo, porque não escolher um que **utiliza** a própria função?
- Se o problema se presta a ser descrito recursivamente, então porque não implementá-lo da mesma forma?
- Para se poder fazer isso mesmo torna-se necessário ter uma descrição recursiva formal do problema: esse é o papel das **Relações de Recorrência**.
- Uma relação de recorrência é uma formulação recursiva formal de um problema.
- As relações de recorrência podem ser sempre implementadas de uma forma **iterativa** ou de uma forma **recursiva**.
- A implementação recursiva é estruturalmente muito próxima da própria relação de recorrência (donde resulta a sua simplicidade).

[Introdução](#)

[Definição](#)

[Complexidade](#)

[Relação de Recorrência](#)

[Exemplo 1: A Função Factorial](#)

[Relação de Recorrência: Síntese](#)

[Exemplo 2: Cálculo das Combinações](#)

[Relação de Recorrência: Classificação](#)

[Exemplo 3: Torres de Hanói](#)

[Definição Recursiva: Condições de Sanidade](#)

[Casos Atípicos](#)

[Casos com Interesse](#)

- O caso das funções é particularmente interessante. Se quem as **implementa** é livre para escolher o melhor algoritmo, porque não escolher um que **utiliza** a própria função?
- Se o problema se presta a ser descrito recursivamente, então porque não implementá-lo da mesma forma?
- Para se poder fazer isso mesmo torna-se necessário ter uma descrição recursiva formal do problema: esse é o papel das **Relações de Recorrência**.
- Uma relação de recorrência é uma formulação recursiva formal de um problema.
- As relações de recorrência podem ser sempre implementadas de uma forma **iterativa** ou de uma forma **recursiva**.
- A implementação recursiva é estruturalmente muito próxima da própria relação de recorrência (donde resulta a sua simplicidade).

[Introdução](#)

[Definição](#)

[Complexidade](#)

[Relação de Recorrência](#)

[Exemplo 1: A Função Factorial](#)

[Relação de Recorrência: Síntese](#)

[Exemplo 2: Cálculo das Combinações](#)

[Relação de Recorrência: Classificação](#)

[Exemplo 3: Torres de Hanói](#)

[Definição Recursiva: Condições de Sanidade](#)

[Casos Atípicos](#)

[Casos com Interesse](#)

- O caso das funções é particularmente interessante. Se quem as **implementa** é livre para escolher o melhor algoritmo, porque não escolher um que **utiliza** a própria função?
- Se o problema se presta a ser descrito recursivamente, então porque não implementá-lo da mesma forma?
- Para se poder fazer isso mesmo torna-se necessário ter uma descrição recursiva formal do problema: esse é o papel das **Relações de Recorrência**.
- Uma relação de recorrência é uma formulação recursiva formal de um problema.
- As relações de recorrência podem ser sempre implementadas de uma forma **iterativa** ou de uma forma **recursiva**.
- A implementação recursiva é estruturalmente muito próxima da própria relação de recorrência (donde resulta a sua simplicidade).

[Introdução](#)

[Definição](#)

[Complexidade](#)

[Relação de Recorrência](#)

[Exemplo 1: A Função Factorial](#)

[Relação de Recorrência: Síntese](#)

[Exemplo 2: Cálculo das Combinações](#)

[Relação de Recorrência: Classificação](#)

[Exemplo 3: Torres de Hanói](#)

[Definição Recursiva: Condições de Sanidade](#)

[Casos Atípicos](#)

[Casos com Interesse](#)

- O caso das funções é particularmente interessante. Se quem as **implementa** é livre para escolher o melhor algoritmo, porque não escolher um que **utiliza** a própria função?
- Se o problema se presta a ser descrito recursivamente, então porque não implementá-lo da mesma forma?
- Para se poder fazer isso mesmo torna-se necessário ter uma descrição recursiva formal do problema: esse é o papel das **Relações de Recorrência**.
- Uma relação de recorrência é uma formulação recursiva formal de um problema.
- As relações de recorrência podem ser sempre implementadas de uma forma **iterativa** ou de uma forma **recursiva**.
- A implementação recursiva é estruturalmente muito próxima da própria relação de recorrência (donde resulta a sua simplicidade).

[Introdução](#)

[Definição](#)

[Complexidade](#)

[Relação de Recorrência](#)

[Exemplo 1: A Função Factorial](#)

[Relação de Recorrência: Síntese](#)

[Exemplo 2: Cálculo das Combinações](#)

[Relação de Recorrência: Classificação](#)

[Exemplo 3: Torres de Hanói](#)

[Definição Recursiva: Condições de Sanidade](#)

[Casos Atípicos](#)

[Casos com Interesse](#)

Exemplo: a função factorial

- Fórmula iterativa:

$$n! = \begin{cases} \prod_{k=1}^n k & , n \in \mathbb{N} \\ 1 & , n = 0 \end{cases}$$

- Fórmula recursiva (relação de recorrência):

$$n! = \begin{cases} n \times (n-1)! & , n \in \mathbb{N} \\ 1 & , n = 0 \end{cases}$$

Exemplo: a função factorial

- Fórmula iterativa:

$$n! = \begin{cases} \prod_{k=1}^n k & , n \in \mathbb{N} \\ 1 & , n = 0 \end{cases}$$

- Fórmula recursiva (relação de recorrência):

$$n! = \begin{cases} n \times (n-1)! & , n \in \mathbb{N} \\ 1 & , n = 0 \end{cases}$$

Exemplo: a função factorial

- Fórmula iterativa:

$$n! = \begin{cases} \prod_{k=1}^n k & , n \in \mathbb{N} \\ 1 & , n = 0 \end{cases}$$

- Fórmula recursiva (relação de recorrência):

$$n! = \begin{cases} n \times (n-1)! & , n \in \mathbb{N} \\ 1 & , n = 0 \end{cases}$$

Exemplo: a função factorial

Implementação Iterativa

```
static int factorial(int n)
{
    assert n >= 0;

    int result = 1;

    for (int i=2; i <= n; i++)
        result = result * i;

    return result;
}
```

$$n! = 1 \times 2 \times \dots \times (n-1) \times n$$

O índice pode variar do caso limite 0 até ao valor n , ou *vice-versa*.

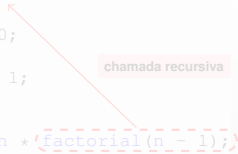
Implementação Recursiva

```
static int factorial(int n)
{
    assert n >= 0;

    int result = 1;

    if (n > 1)
        result = n * factorial(n - 1);

    return result;
}
```



$$n! = n \times ((n-1) \times \dots \times (2 \times (1)) \dots)$$

O argumento varia na direcção do caso limite (de n até 0).

Exemplo: a função factorial

Implementação Iterativa

```
static int factorial(int n)
{
    assert n >= 0;

    int result = 1;

    for (int i=2; i <= n; i++)
        result = result * i;

    return result;
}
```

$$n! = 1 \times 2 \times \dots \times (n-1) \times n$$

O índice pode variar do caso limite 0 até ao valor n , ou *vice-versa*.


Implementação Recursiva

```
static int factorial(int n)
{
    assert n >= 0;

    int result = 1;

    if (n > 1)
        result = n * factorial(n-1);

    return result;
}
```



$$n! = n \times ((n-1) \times \dots \times (2 \times (1)) \dots)$$

O argumento varia na direcção do caso limite (de n até 0).

Exemplo: a função factorial

Implementação Iterativa

```
static int factorial(int n)
{
    assert n >= 0;

    int result = 1;

    for (int i=2; i <= n; i++)
        result = result * i;

    return result;
}
```

$$n! = 1 \times 2 \times \dots \times (n-1) \times n$$

O índice pode variar do caso limite 0 até ao valor n , ou *vice-versa*.

Implementação Recursiva

```
static int factorial(int n)
{
    assert n >= 0;

    int result = 1;

    if (n > 1)
        result = n * factorial(n - 1);

    return result;
}
```

chamada recursiva

$$n! = n \times ((n-1) \times \dots \times (2 \times (1)) \dots)$$

O argumento varia na direcção do caso limite (de n até 0).

Exemplo: a função factorial

Implementação Iterativa

```
static int factorial(int n)
{
    assert n >= 0;

    int result = 1;

    for (int i=2; i <= n; i++)
        result = result * i;

    return result;
}
```

$$n! = 1 \times 2 \times \dots \times (n-1) \times n$$

O índice pode variar do caso limite 0 até ao valor n , ou *vice-versa*.

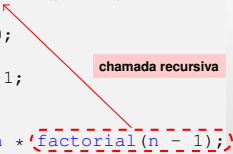
Implementação Recursiva

```
static int factorial(int n)
{
    assert n >= 0;

    int result = 1;

    if (n > 1)
        result = n * factorial(n - 1);

    return result;
}
```



$$n! = n \times ((n-1) \times \dots \times (2 \times (1)) \dots)$$

O argumento varia na direcção do caso limite (de n até 0).

Relação de Recorrência: Síntese

- **Método Iterativo (Repetitivo)**

- O algoritmo avança num ciclo em que o código resolve uma dada situação correspondente às situações anteriores ao valor pretendido.

- **Método Recursivo**

- Uma solução recursiva para um problema é expressa em função de si própria.
- Para isso, os algoritmos recursivos, são desenvolvidos recorrendo a uma dada mais pequena situação, do que a atual.
- Método baseado em recursão, onde a função chama a si mesma, criando uma nova instância da função, sempre que a situação atual não permite a resolução do problema.
- O código recursivo, sempre utiliza uma condição de terminação, para não gerar uma situação de loop infinito.

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- **Método Iterativo** (Repetitivo)
 - O algoritmo assenta num ciclo em que o índice pode variar desde o valor correspondente às situações limite até ao valor pretendido.
- **Método Recursivo**
 - Uma solução recursiva para um problema é expressa em função de si própria.
 - Para que se atinja uma solução, cada invocação recursiva deve estar mais próxima de uma situação limite.
 - Método poderoso e compacto de resolução de problemas mas potencialmente menos eficiente em termos de recursos pois tem de guardar o estado das várias invocações da função.

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- **Método Iterativo** (Repetitivo)
 - O algoritmo assenta num ciclo em que o índice pode variar desde o valor correspondente às situações limite até ao valor pretendido.
- **Método Recursivo**
 - Uma solução recursiva para um problema é expressa em função de si própria.
 - Para que se atinja uma solução, cada invocação recursiva deve estar mais próxima de uma situação limite.
 - Método poderoso e compacto de resolução de problemas mas potencialmente menos eficiente em termos de recursos pois tem de guardar o estado das várias invocações da função.

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- **Método Iterativo** (Repetitivo)
 - O algoritmo assenta num ciclo em que o índice pode variar desde o valor correspondente às situações limite até ao valor pretendido.
- **Método Recursivo**
 - Uma solução recursiva para um problema é expressa em função de si própria.
 - Para que se atinja uma solução, cada invocação recursiva deve estar mais próxima de uma situação limite.
 - Método poderoso e compacto de resolução de problemas mas potencialmente menos eficiente em termos de recursos pois tem de guardar o estado das várias invocações da função.

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- **Método Iterativo** (Repetitivo)
 - O algoritmo assenta num ciclo em que o índice pode variar desde o valor correspondente às situações limite até ao valor pretendido.
- **Método Recursivo**
 - Uma solução recursiva para um problema é expressa em função de si própria.
 - Para que se atinja uma solução, cada invocação recursiva deve estar mais próxima de uma situação limite.
 - Método poderoso e compacto de resolução de problemas mas potencialmente menos eficiente em termos de recursos pois tem de guardar o estado das várias invocações da função.

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- **Método Iterativo** (Repetitivo)
 - O algoritmo assenta num ciclo em que o índice pode variar desde o valor correspondente às situações limite até ao valor pretendido.
- **Método Recursivo**
 - Uma solução recursiva para um problema é expressa em função de si própria.
 - Para que se atinja uma solução, cada invocação recursiva deve estar mais próxima de uma situação limite.
 - Método poderoso e compacto de resolução de problemas mas potencialmente menos eficiente em termos de recursos pois tem de guardar o estado das várias invocações da função.

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- **Método Iterativo** (Repetitivo)
 - O algoritmo assenta num ciclo em que o índice pode variar desde o valor correspondente às situações limite até ao valor pretendido.
- **Método Recursivo**
 - Uma solução recursiva para um problema é expressa em função de si própria.
 - Para que se atinja uma solução, cada invocação recursiva deve estar mais próxima de uma situação limite.
 - Método poderoso e compacto de resolução de problemas mas potencialmente menos eficiente em termos de recursos pois tem de guardar o estado das várias invocações da função.

Exemplo 2: Combinações

- Fórmula:

$$\begin{aligned}C_k^n &= \frac{A_k^n}{A_k^k} = \frac{n \times (n-1) \times \cdots \times (n-k+1)}{k!} \\&= \frac{n!}{(n-k)! \times k!}, \text{ com } n, k \in \mathbb{N}_0 \wedge n \geq k\end{aligned}$$

- A aplicação destas fórmulas pode levantar problemas de cálculo numérico devido ao facto de os registos internos de armazenamento de um valor terem uma capacidade limitada.
- Exemplo:

$$C_{23}^{25} = \frac{15511210043330985984000000}{51704033477769953280000} = 300$$

- Para representar estes números necessitaríamos de pelo menos 84 bits (mesmo o tipo `long` tem apenas 64).
- Solução?

Exemplo 2: Combinações

- Fórmula:

$$\begin{aligned}C_k^n &= \frac{A_k^n}{A_k^k} = \frac{n \times (n-1) \times \dots \times (n-k+1)}{k!} \\&= \frac{n!}{(n-k)! \times k!}, \text{ com } n, k \in \mathbb{N}_0 \wedge n \geq k\end{aligned}$$

- A aplicação destas fórmulas pode levantar problemas de cálculo numérico devido ao facto de os registos internos de armazenamento de um valor terem uma capacidade limitada.
- Exemplo:

$$C_{23}^{25} = \frac{15511210043330985984000000}{51704033477769953280000} = 300$$

- Para representar estes números necessitaríamos de pelo menos 84 bits (mesmo o tipo `long` tem apenas 64).
- Solução?

Exemplo 2: Combinações

- Fórmula:

$$\begin{aligned}C_k^n &= \frac{A_k^n}{A_k^k} = \frac{n \times (n-1) \times \dots \times (n-k+1)}{k!} \\&= \frac{n!}{(n-k)! \times k!}, \text{ com } n, k \in \mathbb{N}_0 \wedge n \geq k\end{aligned}$$

- A aplicação destas fórmulas pode levantar problemas de cálculo numérico devido ao facto de os registos internos de armazenamento de um valor terem uma capacidade limitada.
- Exemplo:

$$C_{23}^{25} = \frac{15511210043330985984000000}{51704033477769953280000} = 300$$

- Para representar estes números necessitaríamos de pelo menos 84 bits (mesmo o tipo `long` tem apenas 64).
- Solução?

Exemplo 2: Combinações

- Fórmula:

$$\begin{aligned}C_k^n &= \frac{A_k^n}{A_k^k} = \frac{n \times (n-1) \times \dots \times (n-k+1)}{k!} \\&= \frac{n!}{(n-k)! \times k!}, \text{ com } n, k \in \mathbb{N}_0 \wedge n \geq k\end{aligned}$$

- A aplicação destas fórmulas pode levantar problemas de cálculo numérico devido ao facto de os registos internos de armazenamento de um valor terem uma capacidade limitada.
- Exemplo:

$$C_{23}^{25} = \frac{15511210043330985984000000}{51704033477769953280000} = 300$$

- Para representar estes números necessitaríamos de pelo menos 84 bits (mesmo o tipo `long` tem apenas 64).
- Solução?

Exemplo 2: Combinações

- Fórmula:

$$\begin{aligned}C_k^n &= \frac{A_k^n}{A_k^k} = \frac{n \times (n-1) \times \dots \times (n-k+1)}{k!} \\&= \frac{n!}{(n-k)! \times k!}, \text{ com } n, k \in \mathbb{N}_0 \wedge n \geq k\end{aligned}$$

- A aplicação destas fórmulas pode levantar problemas de cálculo numérico devido ao facto de os registos internos de armazenamento de um valor terem uma capacidade limitada.
- Exemplo:

$$C_{23}^{25} = \frac{15511210043330985984000000}{51704033477769953280000} = 300$$

- Para representar estes números necessitaríamos de pelo menos 84 bits (mesmo o tipo `long` tem apenas 64).
- Solução?

Exemplo 2: Combinações

- Fórmula:

$$\begin{aligned}C_k^n &= \frac{A_k^n}{A_k^k} = \frac{n \times (n-1) \times \dots \times (n-k+1)}{k!} \\&= \frac{n!}{(n-k)! \times k!}, \text{ com } n, k \in \mathbb{N}_0 \wedge n \geq k\end{aligned}$$

- A aplicação destas fórmulas pode levantar problemas de cálculo numérico devido ao facto de os registos internos de armazenamento de um valor terem uma capacidade limitada.
- Exemplo:

$$C_{23}^{25} = \frac{15511210043330985984000000}{51704033477769953280000} = 300$$

- Para representar estes números necessitaríamos de pelo menos 84 bits (mesmo o tipo `long` tem apenas 64).
- Solução?

Exemplo 2: Combinações

- Fórmula:

$$\begin{aligned}C_k^n &= \frac{A_k^n}{A_k^k} = \frac{n \times (n-1) \times \dots \times (n-k+1)}{k!} \\&= \frac{n!}{(n-k)! \times k!}, \text{ com } n, k \in \mathbb{N}_0 \wedge n \geq k\end{aligned}$$

- A aplicação destas fórmulas pode levantar problemas de cálculo numérico devido ao facto de os registos internos de armazenamento de um valor terem uma capacidade limitada.
- Exemplo:

$$C_{23}^{25} = \frac{15511210043330985984000000}{51704033477769953280000} = 300$$

- Para representar estes números necessitaríamos de pelo menos 84 bits (mesmo o tipo `long` tem apenas 64).
- Solução?

Exemplo 2: Combinações – Relação de Recorrência

- Demonstração:

$$\begin{aligned}C_k^n &= \frac{n!}{(n-k)! \times k!} = \frac{(n-1)! \times (k+n-k)}{(n-k)! \times k!} \\&= \frac{(n-1)! \times k}{(n-k)! \times k!} + \frac{(n-1)! \times (n-k)}{(n-k)! \times k!} \\&= \frac{(n-1)!}{(n-k)! \times (k-1)!} + \frac{(n-1)!}{(n-k-1)! \times k!} \\&= C_{k-1}^{n-1} + C_k^{n-1}\end{aligned}$$

- Relação de recorrência:

$$C_k^n = C_{k-1}^{n-1} + C_k^{n-1}, \text{ com } n, k \in \mathbb{N} \wedge n > k$$

$$C_0^n = 1, \text{ com } n \in \mathbb{N}_0 \quad (\text{caso limite})$$

$$C_n^n = 1, \text{ com } n \in \mathbb{N}_0 \quad (\text{caso limite})$$

Exemplo 2: Combinações – Relação de Recorrência

- Demonstração:

$$\begin{aligned}C_k^n &= \frac{n!}{(n-k)! \times k!} = \frac{(n-1)! \times (k+n-k)}{(n-k)! \times k!} \\&= \frac{(n-1)! \times k}{(n-k)! \times k!} + \frac{(n-1)! \times (n-k)}{(n-k)! \times k!} \\&= \frac{(n-1)!}{(n-k)! \times (k-1)!} + \frac{(n-1)!}{(n-k-1)! \times k!} \\&= C_{k-1}^{n-1} + C_k^{n-1}\end{aligned}$$

- Relação de recorrência:

$$C_k^n = C_{k-1}^{n-1} + C_k^{n-1}, \text{ com } n, k \in \mathbb{N} \wedge n > k$$

$$C_0^n = 1, \text{ com } n \in \mathbb{N}_0 \quad (\text{caso limite})$$

$$C_n^n = 1, \text{ com } n \in \mathbb{N}_0 \quad (\text{caso limite})$$

Exemplo 2: Combinações – Relação de Recorrência

- Demonstração:

$$\begin{aligned}C_k^n &= \frac{n!}{(n-k)! \times k!} = \frac{(n-1)! \times (k+n-k)}{(n-k)! \times k!} \\&= \frac{(n-1)! \times k}{(n-k)! \times k!} + \frac{(n-1)! \times (n-k)}{(n-k)! \times k!} \\&= \frac{(n-1)!}{(n-k)! \times (k-1)!} + \frac{(n-1)!}{(n-k-1)! \times k!} \\&= C_{k-1}^{n-1} + C_k^{n-1}\end{aligned}$$

- Relação de recorrência:

$$C_k^n = C_{k-1}^{n-1} + C_k^{n-1}, \text{ com } n, k \in \mathbb{N} \wedge n > k$$

$$C_0^n = 1, \text{ com } n \in \mathbb{N}_0 \quad (\text{caso limite})$$

$$C_n^n = 1, \text{ com } n \in \mathbb{N}_0 \quad (\text{caso limite})$$

Exemplo 2: Combinações – Relação de Recorrência

- Demonstração:

$$\begin{aligned}C_k^n &= \frac{n!}{(n-k)! \times k!} = \frac{(n-1)! \times (k+n-k)}{(n-k)! \times k!} \\&= \frac{(n-1)! \times k}{(n-k)! \times k!} + \frac{(n-1)! \times (n-k)}{(n-k)! \times k!} \\&= \frac{(n-1)!}{(n-k)! \times (k-1)!} + \frac{(n-1)!}{(n-k-1)! \times k!} \\&= C_{k-1}^{n-1} + C_k^{n-1}\end{aligned}$$

- Relação de recorrência:

$$C_k^n = C_{k-1}^{n-1} + C_k^{n-1} \quad , \text{ com } n, k \in \mathbb{N} \wedge n > k$$

$$C_0^n = 1 \quad , \text{ com } n \in \mathbb{N}_0 \quad (\text{caso limite})$$

$$C_n^n = 1 \quad , \text{ com } n \in \mathbb{N}_0 \quad (\text{caso limite})$$


Exemplo Combinações: Implementação Recursiva

```
static int combNKK(int n, int k)
{
    assert 0 <= k && k <= n;

    int result = 1;

    if (k > 0 && k < n)
        result = (combNKK(n-1, k-1)) + (combNKK(n-1, k));

    return result;
}
```



- Método Recursivo:

- Simples
- Compacto
- Legível
- Fácil de implementar

- E se tentarmos implementar uma solução com o método iterativo?

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos
Casos com Interesse


Exemplo Combinações: Implementação Recursiva

```
static int combNKK(int n, int k)
{
    assert 0 <= k && k <= n;

    int result = 1;

    if (k > 0 && k < n)
        result = combNKK(n-1, k-1) + combNKK(n-1, k);

    return result;
}
```



- Método Recursivo:

- Síntese
- Definição
- Complexidade
- Relação de Recorrência
- Fatoriais

- E se tentarmos implementar uma solução com o método iterativo?

Recursividade

Introdução

Definição

Complexidade

Relação de Recorrência

Exemplo 1: A Função Factorial

Relação de Recorrência: Síntese

Exemplo 2: Cálculo das Combinações

Relação de Recorrência: Classificação

Exemplo 3: Torres de Hanói

Definição Recursiva: Condições de Sanidade

Casos Atípicos
Casos com Interesse

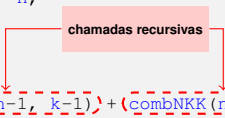
Exemplo Combinações: Implementação Recursiva

```
static int combNKK(int n, int k)
{
    assert 0 <= k && k <= n;

    int result = 1;

    if (k > 0 && k < n)
        result = combNKK(n-1, k-1) + combNKK(n-1, k);

    return result;
}
```



Método Recursivo:

- Síntese
- Caso Base
- Passo
- Função

E se tentarmos implementar uma solução com o método iterativo?

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos
Casos com Interesse

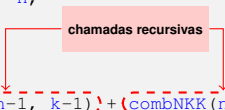
Exemplo Combinações: Implementação Recursiva

```
static int combNKK(int n, int k)
{
    assert 0 <= k && k <= n;

    int result = 1;

    if (k > 0 && k < n)
        result = combNKK(n-1, k-1) + combNKK(n-1, k);

    return result;
}
```



- Método Recursivo:
 - Simples;
 - Compacto;
 - Legível;
 - Fácil detectar erros.
- E se tentarmos implementar uma solução com o método iterativo?

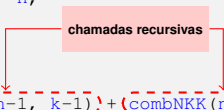
Exemplo Combinações: Implementação Recursiva

```
static int combNKK(int n, int k)
{
    assert 0 <= k && k <= n;

    int result = 1;

    if (k > 0 && k < n)
        result = combNKK(n-1, k-1) + combNKK(n-1, k);

    return result;
}
```



- Método Recursivo:
 - Simples;
 - Compacto;
 - Legível;
 - Fácil detectar erros.
- E se tentarmos implementar uma solução com o método iterativo?

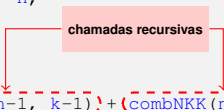
Exemplo Combinações: Implementação Recursiva

```
static int combNKK(int n, int k)
{
    assert 0 <= k && k <= n;

    int result = 1;

    if (k > 0 && k < n)
        result = combNKK(n-1, k-1) + combNKK(n-1, k);

    return result;
}
```



- Método Recursivo:
 - Simples;
 - Compacto;
 - Legível;
 - Fácil detectar erros.
- E se tentarmos implementar uma solução com o método iterativo?

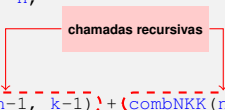
Exemplo Combinações: Implementação Recursiva

```
static int combNKK(int n, int k)
{
    assert 0 <= k && k <= n;

    int result = 1;

    if (k > 0 && k < n)
        result = combNKK(n-1, k-1) + combNKK(n-1, k);

    return result;
}
```



- Método Recursivo:
 - Simples;
 - Compacto;
 - Legível;
 - Fácil detectar erros.
- E se tentarmos implementar uma solução com o método iterativo?

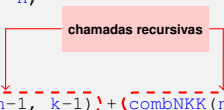
Exemplo Combinações: Implementação Recursiva

```
static int combNKK(int n, int k)
{
    assert 0 <= k && k <= n;

    int result = 1;

    if (k > 0 && k < n)
        result = combNKK(n-1, k-1) + combNKK(n-1, k);

    return result;
}
```



- Método Recursivo:
 - Simples;
 - Compacto;
 - Legível;
 - Fácil detectar erros.
- E se tentarmos implementar uma solução com o método iterativo?

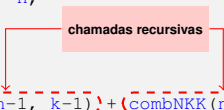
Exemplo Combinações: Implementação Recursiva

```
static int combNKK(int n, int k)
{
    assert 0 <= k && k <= n;

    int result = 1;

    if (k > 0 && k < n)
        result = combNKK(n-1, k-1) + combNKK(n-1, k);

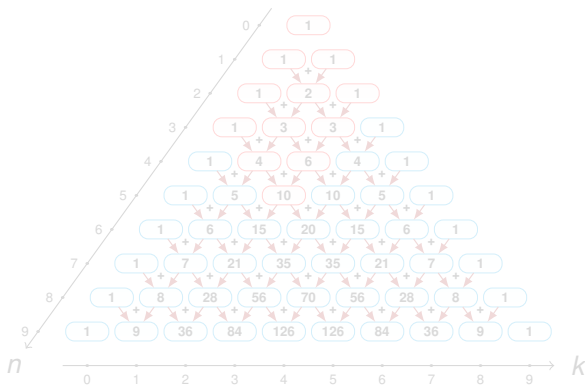
    return result;
}
```



- Método Recursivo:
 - Simples;
 - Compacto;
 - Legível;
 - Fácil detectar erros.
- E se tentarmos implementar uma solução com o método iterativo?

Exemplo Combinações: Implementação Iterativa

• Triângulo de Pascal:



$$C_2^5 = C_1^4 + C_2^4 \left\{ \begin{array}{l} C_1^4 = C_0^3 + C_1^3 \{ \dots \\ C_2^4 = C_1^3 + C_2^3 \{ \dots \end{array} \right.$$

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

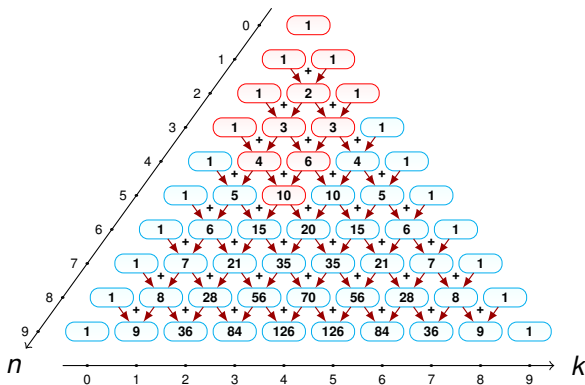
Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Exemplo Combinações: Implementação Iterativa

- Triângulo de Pascal:



$$C_2^5 = C_1^4 + C_2^4 \left\{ \begin{array}{l} C_1^4 = C_0^3 + C_1^3 \{ \dots \\ C_2^4 = C_1^3 + C_2^3 \{ \dots \end{array} \right.$$

Exemplo Combinações: Implementação Iterativa

- Necessitamos de um *array* de $k + 1$ elementos para guardar os valores de uma linha (inicializado a zeros).
- O processo iterativo pode seguir as regras seguintes:
 - Criar um $n + 1$ iterações (uma por linha);
 - Na primeira linha ($n = 0$) tem apenas o valor 1 na posição $k = 0$ do array; esse valor muda-se de 1 para todos as linhas;
 - Para as restantes n linhas, os valores do array desde o $k = 0$ até ao $n - k$ são calculados como sendo a soma dos dois valores referidos pela relação de recorrência (se o índice do array for k , então soma-se os dois valores de $k - 1$ e k).
- O resultado é o elemento de índice k da linha n .
- Este algoritmo pode ser optimizado considerando as seguintes factos:
 - Não é necessário calcular um triângulo completo (para n e k bastam os valores armazenados a verticais na figura);
 - O triângulo de Pascal é simétrico, por isso basta calcular metade;
- O programa mostrado a seguir faz todas essas optimizações.

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Exemplo Combinações: Implementação Iterativa

- Necessitamos de um *array* de $k + 1$ elementos para guardar os valores de uma linha (inicializado a zeros).
- O processo iterativo pode seguir as regras seguintes:
 - 1 existem $n + 1$ iterações (uma por linha);
 - 2 a primeira linha ($n = 0$) tem apenas o valor 1 (na posição $k = 0$ do *array*), esse valor manter-se-á fixo para todas as linhas;
 - 3 para as restantes n linhas, os valores do *array* desde o índice 1 até ao índice k são calculados como sendo a soma dos dois valores referidos pela relação de recorrência (se o índice do array for i , então será a soma dos valores com índice $i - 1$ e i).
- O resultado é o elemento de índice k da linha n .
- Este algoritmo pode ser optimizado considerando as seguintes factos:
 - Não é necessário calcular um triângulo completo (para C_2^5 bastam os valores assinalados a vermelho na figura).
 - O triângulo de Pascal é simétrico, por isso basta calcular metade.
- O programa mostrado a seguir faz todas essas optimizações.

Exemplo Combinações: Implementação Iterativa

- Necessitamos de um *array* de $k + 1$ elementos para guardar os valores de uma linha (inicializado a zeros).
- O processo iterativo pode seguir as regras seguintes:
 - 1 existem $n + 1$ iterações (uma por linha);
 - 2 a primeira linha ($n = 0$) tem apenas o valor 1 (na posição $k = 0$ do *array*), esse valor manter-se-á fixo para todas as linhas;
 - 3 para as restantes n linhas, os valores do *array* desde o índice 1 até ao índice k são calculados como sendo a soma dos dois valores referidos pela relação de recorrência (se o índice do array for i , então será a soma dos valores com índice $i - 1$ e i).
- O resultado é o elemento de índice k da linha n .
- Este algoritmo pode ser optimizado considerando as seguintes factos:
 - Não é necessário calcular um triângulo completo (para C_2^5 bastam os valores assinalados a vermelho na figura).
 - O triângulo de Pascal é simétrico, por isso basta calcular metade.
- O programa mostrado a seguir faz todas essas optimizações.

Exemplo Combinações: Implementação Iterativa

- Necessitamos de um *array* de $k + 1$ elementos para guardar os valores de uma linha (inicializado a zeros).
- O processo iterativo pode seguir as regras seguintes:
 - 1 existem $n + 1$ iterações (uma por linha);
 - 2 a primeira linha ($n = 0$) tem apenas o valor 1 (na posição $k = 0$ do *array*), esse valor manter-se-á fixo para todas as linhas;
 - 3 para as restantes n linhas, os valores do *array* desde o índice 1 até ao índice k são calculados como sendo a soma dos dois valores referidos pela relação de recorrência (se o índice do array for i , então será a soma dos valores com índice $i - 1$ e i).
- O resultado é o elemento de índice k da linha n .
- Este algoritmo pode ser optimizado considerando as seguintes factos:
 - Não é necessário calcular um triângulo completo (para C_2^5 bastam os valores assinalados a vermelho na figura).
 - O triângulo de Pascal é simétrico, por isso basta calcular metade.
- O programa mostrado a seguir faz todas essas optimizações.

Exemplo Combinações: Implementação Iterativa

- Necessitamos de um *array* de $k + 1$ elementos para guardar os valores de uma linha (inicializado a zeros).
- O processo iterativo pode seguir as regras seguintes:
 - 1 existem $n + 1$ iterações (uma por linha);
 - 2 a primeira linha ($n = 0$) tem apenas o valor 1 (na posição $k = 0$ do *array*), esse valor manter-se-á fixo para todas as linhas;
 - 3 para as restantes n linhas, os valores do *array* desde o índice 1 até ao índice k são calculados como sendo a soma dos dois valores referidos pela relação de recorrência (se o índice do array for i , então será a soma dos valores com índice $i - 1$ e i).
- O resultado é o elemento de índice k da linha n .
- Este algoritmo pode ser otimizado considerando as seguintes factos:
 - Não é necessário calcular um triângulo completo (para C_2^5 bastam os valores assinalados a vermelho na figura).
 - O triângulo de Pascal é simétrico, por isso basta calcular metade.
- O programa mostrado a seguir faz todas essas otimizações.

Exemplo Combinações: Implementação Iterativa

- Necessitamos de um *array* de $k + 1$ elementos para guardar os valores de uma linha (inicializado a zeros).
- O processo iterativo pode seguir as regras seguintes:
 - 1 existem $n + 1$ iterações (uma por linha);
 - 2 a primeira linha ($n = 0$) tem apenas o valor 1 (na posição $k = 0$ do *array*), esse valor manter-se-á fixo para todas as linhas;
 - 3 para as restantes n linhas, os valores do *array* desde o índice 1 até ao índice k são calculados como sendo a soma dos dois valores referidos pela relação de recorrência (se o índice do *array* for i , então será a soma dos valores com índice $i - 1$ e i).
- O resultado é o elemento de índice k da linha n .
- Este algoritmo pode ser optimizado considerando as seguintes factos:
 - Não é necessário calcular um triângulo completo (para C_2^5 bastam os valores assinalados a vermelho na figura).
 - O triângulo de Pascal é simétrico, por isso basta calcular metade.
- O programa mostrado a seguir faz todas essas optimizações.

Exemplo Combinações: Implementação Iterativa

- Necessitamos de um *array* de $k + 1$ elementos para guardar os valores de uma linha (inicializado a zeros).
- O processo iterativo pode seguir as regras seguintes:
 - 1 existem $n + 1$ iterações (uma por linha);
 - 2 a primeira linha ($n = 0$) tem apenas o valor 1 (na posição $k = 0$ do *array*), esse valor manter-se-á fixo para todas as linhas;
 - 3 para as restantes n linhas, os valores do *array* desde o índice 1 até ao índice k são calculados como sendo a soma dos dois valores referidos pela relação de recorrência (se o índice do *array* for i , então será a soma dos valores com índice $i - 1$ e i).
- O resultado é o elemento de índice k da linha n .
- Este algoritmo pode ser otimizado considerando as seguintes factos:
 - Não é necessário calcular um triângulo completo (para C_2^5 bastam os valores assinalados a vermelho na figura).
 - O triângulo de Pascal é simétrico, por isso basta calcular metade.
- O programa mostrado a seguir faz todas essas otimizações.

Exemplo Combinações: Implementação Iterativa

- Necessitamos de um *array* de $k + 1$ elementos para guardar os valores de uma linha (inicializado a zeros).
- O processo iterativo pode seguir as regras seguintes:
 - 1 existem $n + 1$ iterações (uma por linha);
 - 2 a primeira linha ($n = 0$) tem apenas o valor 1 (na posição $k = 0$ do *array*), esse valor manter-se-á fixo para todas as linhas;
 - 3 para as restantes n linhas, os valores do *array* desde o índice 1 até ao índice k são calculados como sendo a soma dos dois valores referidos pela relação de recorrência (se o índice do *array* for i , então será a soma dos valores com índice $i - 1$ e i).
- O resultado é o elemento de índice k da linha n .
- Este algoritmo pode ser optimizado considerando as seguintes factos:
 - Não é necessário calcular um triângulo completo (para C_2^5 bastam os valores assinalados a vermelho na figura).
 - O triângulo de Pascal é simétrico, por isso basta calcular metade.
- O programa mostrado a seguir faz todas essas optimizações.

Exemplo Combinações: Implementação Iterativa

- Necessitamos de um *array* de $k + 1$ elementos para guardar os valores de uma linha (inicializado a zeros).
- O processo iterativo pode seguir as regras seguintes:
 - 1 existem $n + 1$ iterações (uma por linha);
 - 2 a primeira linha ($n = 0$) tem apenas o valor 1 (na posição $k = 0$ do *array*), esse valor manter-se-á fixo para todas as linhas;
 - 3 para as restantes n linhas, os valores do *array* desde o índice 1 até ao índice k são calculados como sendo a soma dos dois valores referidos pela relação de recorrência (se o índice do *array* for i , então será a soma dos valores com índice $i - 1$ e i).
- O resultado é o elemento de índice k da linha n .
- Este algoritmo pode ser otimizado considerando as seguintes factos:
 - Não é necessário calcular um triângulo completo (para C_2^5 bastam os valores assinalados a vermelho na figura).
 - O triângulo de Pascal é simétrico, por isso basta calcular metade.
- O programa mostrado a seguir faz todas essas otimizações.

Exemplo Combinações: Implementação Iterativa

- Necessitamos de um *array* de $k + 1$ elementos para guardar os valores de uma linha (inicializado a zeros).
- O processo iterativo pode seguir as regras seguintes:
 - 1 existem $n + 1$ iterações (uma por linha);
 - 2 a primeira linha ($n = 0$) tem apenas o valor 1 (na posição $k = 0$ do *array*), esse valor manter-se-á fixo para todas as linhas;
 - 3 para as restantes n linhas, os valores do *array* desde o índice 1 até ao índice k são calculados como sendo a soma dos dois valores referidos pela relação de recorrência (se o índice do *array* for i , então será a soma dos valores com índice $i - 1$ e i).
- O resultado é o elemento de índice k da linha n .
- Este algoritmo pode ser optimizado considerando as seguintes factos:
 - Não é necessário calcular um triângulo completo (para C_2^5 bastam os valores assinalados a vermelho na figura).
 - O triângulo de Pascal é simétrico, por isso basta calcular metade.
- O programa mostrado a seguir faz todas essas optimizações.

Exemplo Combinações: Implementação Iterativa

- Necessitamos de um *array* de $k + 1$ elementos para guardar os valores de uma linha (inicializado a zeros).
- O processo iterativo pode seguir as regras seguintes:
 - 1 existem $n + 1$ iterações (uma por linha);
 - 2 a primeira linha ($n = 0$) tem apenas o valor 1 (na posição $k = 0$ do *array*), esse valor manter-se-á fixo para todas as linhas;
 - 3 para as restantes n linhas, os valores do *array* desde o índice 1 até ao índice k são calculados como sendo a soma dos dois valores referidos pela relação de recorrência (se o índice do array for i , então será a soma dos valores com índice $i - 1$ e i).
- O resultado é o elemento de índice k da linha n .
- Este algoritmo pode ser optimizado considerando as seguintes factos:
 - Não é necessário calcular um triângulo completo (para C_2^5 bastam os valores assinalados a vermelho na figura).
 - O triângulo de Pascal é simétrico, por isso basta calcular metade.
- O programa mostrado a seguir faz todas essas optimizações.

Exemplo Combinações: Implementação Iterativa

```
static int combIterTP(int n,int k)
{
    assert 0 <= k && k <= n;

    int result = 1;
    if (k > 0 && k < n) {
        int kMin = k < n-k ? k : n-k; // minimo(k, n-k)
        int[] linha = new int[k + 1];
        int c = 0;
        int cIni = 1;
        linha[0] = 1;
        for(int l = 1; l <= n; l++) {
            if (l > n-kMin+1)
                cIni++;
            for(c = kMin; c >= cIni; c--)
                linha[c] = linha[c]+linha[c-1];
        }
        result = linha[kMin];
    }

    return result;
}
```

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos
Casos com Interesse

Exemplo Combinações: Implementação Iterativa

```
static int combIterTP(int n,int k)
{
    assert 0 <= k && k <= n;

    int result = 1;
    if (k > 0 && k < n) {
        int kMin = k < n-k ? k : n-k; // minimo(k, n-k)
        int[] linha = new int[k + 1];
        int c = 0;
        int cIni = 1;
        linha[0] = 1;
        for(int l = 1; l <= n; l++) {
            if (l > n-kMin+1)
                cIni++;
            for(c = kMin; c >= cIni; c--)
                linha[c] = linha[c]+linha[c-1];
        }
        result = linha[kMin];
    }

    return result;
}
```

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos
Casos com Interesse

Relação de Recorrência: Classificação

Em termos de complexidade do mecanismo de descrição:

- **Simples:** quando há apenas uma chamada recursiva.
Exemplo: factorial.
- **Composta:** quando há múltiplas chamadas recursivas.
Exemplo: combinações, Torres de Hanói.

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Relação de Recorrência: Classificação

Em termos de complexidade do mecanismo de descrição:

- **Simples:** quando há apenas uma chamada recursiva.
Exemplo: Fatorial
- **Composta:** quando há múltiplas chamadas recursivas.
Exemplo: combinações, Torres de Hanói

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Relação de Recorrência: Classificação

Em termos de complexidade do mecanismo de descrição:

- **Simples**: quando há apenas uma chamada recursiva.
 - Exemplo: factorial.
- **Composta**: quando há múltiplas chamadas recursivas.
 - Exemplo: combinações, torres de Hanói.

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Relação de Recorrência: Classificação

Em termos de complexidade do mecanismo de descrição:

- **Simples**: quando há apenas uma chamada recursiva.
 - Exemplo: factorial.
- **Composta**: quando há múltiplas chamadas recursivas.
 - Exemplo: combinações, torres de Hanói.

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Em termos de complexidade do mecanismo de descrição:

- **Simple**s: quando há apenas uma chamada recursiva.
 - Exemplo: factorial.
- **Composta**: quando há múltiplas chamadas recursivas.
 - Exemplo: combinações, torres de Hanói.

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

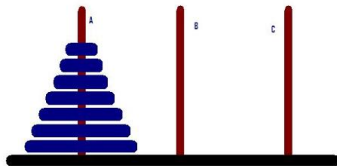
Casos Atípicos

Casos com Interesse

Em termos de complexidade do mecanismo de descrição:

- **Simples**: quando há apenas uma chamada recursiva.
 - Exemplo: factorial.
- **Composta**: quando há múltiplas chamadas recursivas.
 - Exemplo: combinações, torres de Hanói.

Exemplo 3: Torres de Hanói



- Este jogo, criado pelo matemático francês Édouard Lucas no Século XIX, é um dos exemplos clássicos que mostram as potencialidades dos algoritmos recursivos.
- Existem três postes onde se podem enfiar discos de diâmetros decrescente.
- O objectivo do jogo é mover todos os discos de um poste para outro, de acordo com as seguintes regras:
 - ❑ Só pode enfiar um disco de cada vez.
 - ❑ Não pode colocar um disco em cima de outro de menor diâmetro.

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

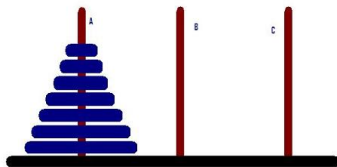
Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

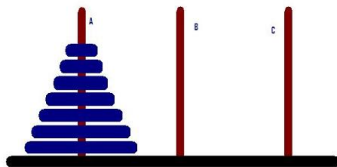
Casos com Interesse

Exemplo 3: Torres de Hanói



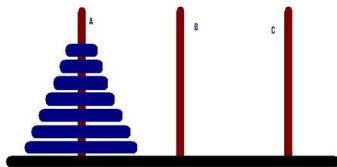
- Este jogo, criado pelo matemático francês Édouard Lucas no Século XIX, é um dos exemplos clássicos que mostram as potencialidades dos algoritmos recursivos.
- Existem três postes onde se podem enfiar discos de diâmetros decrescente.
- O objectivo do jogo é mover todos os discos de um poste para outro, de acordo com as seguintes regras:
 1. Só pode mover um disco de cada vez;
 2. Não pode colocar um disco em cima de outro de menor dimensão.

Exemplo 3: Torres de Hanói



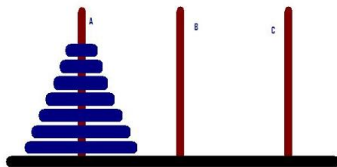
- Este jogo, criado pelo matemático francês Édouard Lucas no Século XIX, é um dos exemplos clássicos que mostram as potencialidades dos algoritmos recursivos.
- Existem três postes onde se podem enfiar discos de diâmetros decrescente.
- O objectivo do jogo é mover todos os discos de um poste para outro, de acordo com as seguintes regras:
 1. Só pode mover um disco de cada vez;
 2. Não pode colocar um disco em cima de outro de menor dimensão.

Exemplo 3: Torres de Hanói



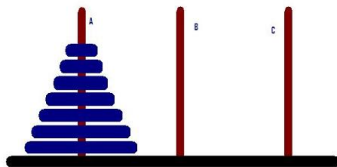
- Este jogo, criado pelo matemático francês Édouard Lucas no Século XIX, é um dos exemplos clássicos que mostram as potencialidades dos algoritmos recursivos.
- Existem três postes onde se podem enfiar discos de diâmetros decrescente.
- O objectivo do jogo é mover todos os discos de um poste para outro, de acordo com as seguintes regras:
 - 1 Só pode mover um disco de cada vez;
 - 2 Não pode colocar um disco em cima de outro de menor dimensão.

Exemplo 3: Torres de Hanói



- Este jogo, criado pelo matemático francês Édouard Lucas no Século XIX, é um dos exemplos clássicos que mostram as potencialidades dos algoritmos recursivos.
- Existem três postes onde se podem enfiar discos de diâmetros decrescente.
- O objectivo do jogo é mover todos os discos de um poste para outro, de acordo com as seguintes regras:
 - 1 Só pode mover um disco de cada vez;
 - 2 Não pode colocar um disco em cima de outro de menor dimensão.

Exemplo 3: Torres de Hanói



- Este jogo, criado pelo matemático francês Édouard Lucas no Século XIX, é um dos exemplos clássicos que mostram as potencialidades dos algoritmos recursivos.
- Existem três postes onde se podem enfiar discos de diâmetros decrescente.
- O objectivo do jogo é mover todos os discos de um poste para outro, de acordo com as seguintes regras:
 - 1 Só pode mover um disco de cada vez;
 - 2 Não pode colocar um disco em cima de outro de menor dimensão.

Relação de recorrência:

```
* moverDiscos(n, tOrigem, tDestino, tAuxiliar)
  if n > 0
    moverDiscos(n-1, tOrigem, tAuxiliar, tDestino)
    moverDisco(tOrigem, tDestino)
    moverDiscos(n-1, tAuxiliar, tDestino, tOrigem)
```

Caso limite:

```
* moverDiscos(1, tOrigem, tDestino, tAuxiliar)
  moverDisco(tOrigem, tDestino)
```

ou, alternativamente:

```
* moverDiscos(0, tOrigem, tDestino, tAuxiliar)
  // não é preciso fazer nada
```

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos
Casos com Interesse

Relação de recorrência:

```
* moverDiscos(n, tOrigem, tDestino, tAuxiliar)
  if (n > 0) {
    moverDiscos(n-1, tOrigem, tAuxiliar, tDestino);
    moverDiscos(tOrigem, tDestino);
    moverDiscos(n-1, tAuxiliar, tDestino, tOrigem);
  }
```

Caso limite:

```
* moverDiscos(1, tOrigem, tDestino, tAuxiliar)
  if (n > 0) moverDiscos(tOrigem, tDestino);
```

ou, alternativamente:

```
* moverDiscos(0, tOrigem, tDestino, tAuxiliar)
  // não é preciso fazer nada
```

[Introdução](#)[Definição](#)[Complexidade](#)[Relação de
Recorrência](#)[Exemplo 1: A Função
Factorial](#)[Relação de
Recorrência: Síntese](#)[Exemplo 2: Cálculo
das Combinações](#)[Relação de
Recorrência:
Classificação](#)[Exemplo 3: Torres de
Hanói](#)[Definição Recursiva:
Condições de
Sanidade](#)[Casos Atípicos](#)[Casos com Interesse](#)

[Introdução](#)[Definição](#)[Complexidade](#)[Relação de
Recorrência](#)[Exemplo 1: A Função
Factorial](#)[Relação de
Recorrência: Síntese](#)[Exemplo 2: Cálculo
das Combinações](#)[Relação de
Recorrência:
Classificação](#)[Exemplo 3: Torres de
Hanói](#)[Definição Recursiva:
Condições de
Sanidade](#)[Casos Atípicos](#)
[Casos com Interesse](#)

Relação de recorrência:

- `moverDiscos(n, tOrigem, tDestino, tAuxiliar)`
 - 1 `moverDiscos(n-1, tOrigem, tAuxiliar, tDestino)`
 - 2 `moverUmDisco(tOrigem, tDestino)`
 - 3 `moverDiscos(n-1, tAuxiliar, tDestino, tOrigem)`

Caso limite:

- `moverDiscos(1, tOrigem, tDestino, tAuxiliar)`
 `moverUmDisco(tOrigem, tDestino)`

ou, alternativamente:

- `moverDiscos(0, tOrigem, tDestino, tAuxiliar)`
 `return; // não é possível mover nada!`

[Introdução](#)[Definição](#)[Complexidade](#)[Relação de
Recorrência](#)[Exemplo 1: A Função
Factorial](#)[Relação de
Recorrência: Síntese](#)[Exemplo 2: Cálculo
das Combinações](#)[Relação de
Recorrência:
Classificação](#)[Exemplo 3: Torres de
Hanói](#)[Definição Recursiva:
Condições de
Sanidade](#)[Casos Atípicos](#)[Casos com Interesse](#)

Relação de recorrência:

- `moverDiscos(n, tOrigem, tDestino, tAuxiliar)`
 - 1 `moverDiscos(n-1, tOrigem, tAuxiliar, tDestino)`
 - 2 `moverUmDisco(tOrigem, tDestino)`
 - 3 `moverDiscos(n-1, tAuxiliar, tDestino, tOrigem)`

Caso limite:

- `moverDiscos(1, tOrigem, tDestino, tAuxiliar)`
 `moverUmDisco(tOrigem, tDestino)`

ou, alternativamente:

- `moverDiscos(0, tOrigem, tDestino, tAuxiliar)`
 `return; // não é possível mover nada!`

[Introdução](#)[Definição](#)[Complexidade](#)[Relação de
Recorrência](#)[Exemplo 1: A Função
Factorial](#)[Relação de
Recorrência: Síntese](#)[Exemplo 2: Cálculo
das Combinações](#)[Relação de
Recorrência:
Classificação](#)[Exemplo 3: Torres de
Hanói](#)[Definição Recursiva:
Condições de
Sanidade](#)[Casos Atípicos](#)[Casos com Interesse](#)

Relação de recorrência:

- `moverDiscos(n, tOrigem, tDestino, tAuxiliar)`
 - 1 `moverDiscos(n-1, tOrigem, tAuxiliar, tDestino)`
 - 2 `moverUmDisco(tOrigem, tDestino)`
 - 3 `moverDiscos(n-1, tAuxiliar, tDestino, tOrigem)`

Caso limite:

```
* moverDiscos(1, tOrigem, tDestino, tAuxiliar)
{
    moverUmDisco(tOrigem, tDestino);
}
```

ou, alternativamente:

```
* moverDiscos(0, tOrigem, tDestino, tAuxiliar)
{
    // não é preciso fazer nada!
}
```

[Introdução](#)[Definição](#)[Complexidade](#)[Relação de
Recorrência](#)[Exemplo 1: A Função
Factorial](#)[Relação de
Recorrência: Síntese](#)[Exemplo 2: Cálculo
das Combinações](#)[Relação de
Recorrência:
Classificação](#)[Exemplo 3: Torres de
Hanói](#)[Definição Recursiva:
Condições de
Sanidade](#)[Casos Atípicos](#)
[Casos com Interesse](#)

Relação de recorrência:

- moverDiscos(n, tOrigem, tDestino, tAuxiliar)
 - 1 moverDiscos(n-1, tOrigem, tAuxiliar, tDestino)
 - 2 moverUmDisco(tOrigem, tDestino)
 - 3 moverDiscos(n-1, tAuxiliar, tDestino, tOrigem)

Caso limite:

```
* moverDiscos(1, tOrigem, tDestino, tAuxiliar)
// moverUmDisco(tOrigem, tDestino)
```

ou, alternativamente:

```
* moverDiscos(0, tOrigem, tDestino, tAuxiliar)
// não é possível mover nada!
```


[Introdução](#)[Definição](#)[Complexidade](#)[Relação de
Recorrência](#)[Exemplo 1: A Função
Factorial](#)[Relação de
Recorrência: Síntese](#)[Exemplo 2: Cálculo
das Combinações](#)[Relação de
Recorrência:
Classificação](#)[Exemplo 3: Torres de
Hanói](#)[Definição Recursiva:
Condições de
Sanidade](#)[Casos Atípicos](#)
[Casos com Interesse](#)

Relação de recorrência:

- moverDiscos(n , tOrigem, tDestino, tAuxiliar)
 - 1 moverDiscos($n-1$, tOrigem, tAuxiliar, tDestino)
 - 2 moverUmDisco(tOrigem, tDestino)
 - 3 moverDiscos($n-1$, tAuxiliar, tDestino, tOrigem)

Caso limite:

```
* moverDiscos(1, tOrigem, tDestino, tAuxiliar)
// moverUmDisco(tOrigem, tDestino)
```

ou, alternativamente:

```
* moverDiscos(0, tOrigem, tDestino, tAuxiliar)
// não é possível mover nada!
```

[Introdução](#)[Definição](#)[Complexidade](#)[Relação de
Recorrência](#)[Exemplo 1: A Função
Factorial](#)[Relação de
Recorrência: Síntese](#)[Exemplo 2: Cálculo
das Combinações](#)[Relação de
Recorrência:
Classificação](#)[Exemplo 3: Torres de
Hanói](#)[Definição Recursiva:
Condições de
Sanidade](#)[Casos Atípicos](#)[Casos com Interesse](#)

Relação de recorrência:

- `moverDiscos(n, tOrigem, tDestino, tAuxiliar)`
 - 1 `moverDiscos(n-1, tOrigem, tAuxiliar, tDestino)`
 - 2 `moverUmDisco(tOrigem, tDestino)`
 - 3 `moverDiscos(n-1, tAuxiliar, tDestino, tOrigem)`

Caso limite:

```
* moverDiscos(1, tOrigem, tDestino, tAuxiliar)
{
    moverUmDisco(tOrigem, tDestino);
}
```

ou, alternativamente:

```
* moverDiscos(0, tOrigem, tDestino, tAuxiliar)
{
    // não é possível mover nada!
}
```

[Introdução](#)[Definição](#)[Complexidade](#)[Relação de
Recorrência](#)[Exemplo 1: A Função
Factorial](#)[Relação de
Recorrência: Síntese](#)[Exemplo 2: Cálculo
das Combinações](#)[Relação de
Recorrência:
Classificação](#)[Exemplo 3: Torres de
Hanói](#)[Definição Recursiva:
Condições de
Sanidade](#)[Casos Atípicos](#)[Casos com Interesse](#)

Relação de recorrência:

- `moverDiscos(n, tOrigem, tDestino, tAuxiliar)`
 - 1 `moverDiscos(n-1, tOrigem, tAuxiliar, tDestino)`
 - 2 `moverUmDisco(tOrigem, tDestino)`
 - 3 `moverDiscos(n-1, tAuxiliar, tDestino, tOrigem)`

Caso limite:

- `moverDiscos(1, tOrigem, tDestino, tAuxiliar)`
 - 1 `moverUmDisco(tOrigem, tDestino)`

ou, alternativamente:

- `moverDiscos(0, tOrigem, tDestino, tAuxiliar)`
`return`

[Introdução](#)[Definição](#)[Complexidade](#)[Relação de
Recorrência](#)[Exemplo 1: A Função
Factorial](#)[Relação de
Recorrência: Síntese](#)[Exemplo 2: Cálculo
das Combinações](#)[Relação de
Recorrência:
Classificação](#)[Exemplo 3: Torres de
Hanói](#)[Definição Recursiva:
Condições de
Sanidade](#)[Casos Atípicos](#)[Casos com Interesse](#)

Relação de recorrência:

- `moverDiscos(n, tOrigem, tDestino, tAuxiliar)`
 - 1 `moverDiscos(n-1, tOrigem, tAuxiliar, tDestino)`
 - 2 `moverUmDisco(tOrigem, tDestino)`
 - 3 `moverDiscos(n-1, tAuxiliar, tDestino, tOrigem)`

Caso limite:

- `moverDiscos(1, tOrigem, tDestino, tAuxiliar)`
 - 1 `moverUmDisco(tOrigem, tDestino)`

ou, alternativamente:

- `moverDiscos(0, tOrigem, tDestino, tAuxiliar)`
`return`

[Introdução](#)[Definição](#)[Complexidade](#)[Relação de
Recorrência](#)[Exemplo 1: A Função
Factorial](#)[Relação de
Recorrência: Síntese](#)[Exemplo 2: Cálculo
das Combinações](#)[Relação de
Recorrência:
Classificação](#)[Exemplo 3: Torres de
Hanói](#)[Definição Recursiva:
Condições de
Sanidade](#)[Casos Atípicos](#)[Casos com Interesse](#)

Relação de recorrência:

- `moverDiscos(n, tOrigem, tDestino, tAuxiliar)`
 - 1 `moverDiscos(n-1, tOrigem, tAuxiliar, tDestino)`
 - 2 `moverUmDisco(tOrigem, tDestino)`
 - 3 `moverDiscos(n-1, tAuxiliar, tDestino, tOrigem)`

Caso limite:

- `moverDiscos(1, tOrigem, tDestino, tAuxiliar)`
 - 1 `moverUmDisco(tOrigem, tDestino)`

ou, alternativamente:

- `moverDiscos(0, tOrigem, tDestino, tAuxiliar)`
`return`

[Introdução](#)[Definição](#)[Complexidade](#)[Relação de
Recorrência](#)[Exemplo 1: A Função
Factorial](#)[Relação de
Recorrência: Síntese](#)[Exemplo 2: Cálculo
das Combinações](#)[Relação de
Recorrência:
Classificação](#)[Exemplo 3: Torres de
Hanói](#)[Definição Recursiva:
Condições de
Sanidade](#)[Casos Atípicos](#)[Casos com Interesse](#)

Relação de recorrência:

- `moverDiscos(n, tOrigem, tDestino, tAuxiliar)`
 - 1 `moverDiscos(n-1, tOrigem, tAuxiliar, tDestino)`
 - 2 `moverUmDisco(tOrigem, tDestino)`
 - 3 `moverDiscos(n-1, tAuxiliar, tDestino, tOrigem)`

Caso limite:

- `moverDiscos(1, tOrigem, tDestino, tAuxiliar)`
 - 1 `moverUmDisco(tOrigem, tDestino)`

ou, alternativamente:

- `moverDiscos(0, tOrigem, tDestino, tAuxiliar)`
 - 1 *(não é preciso fazer nada)*

[Introdução](#)[Definição](#)[Complexidade](#)[Relação de
Recorrência](#)[Exemplo 1: A Função
Factorial](#)[Relação de
Recorrência: Síntese](#)[Exemplo 2: Cálculo
das Combinações](#)[Relação de
Recorrência:
Classificação](#)[Exemplo 3: Torres de
Hanói](#)[Definição Recursiva:
Condições de
Sanidade](#)[Casos Atípicos](#)[Casos com Interesse](#)

Relação de recorrência:

- `moverDiscos(n, tOrigem, tDestino, tAuxiliar)`
 - 1 `moverDiscos(n-1, tOrigem, tAuxiliar, tDestino)`
 - 2 `moverUmDisco(tOrigem, tDestino)`
 - 3 `moverDiscos(n-1, tAuxiliar, tDestino, tOrigem)`

Caso limite:

- `moverDiscos(1, tOrigem, tDestino, tAuxiliar)`
 - 1 `moverUmDisco(tOrigem, tDestino)`

ou, alternativamente:

- `moverDiscos(0, tOrigem, tDestino, tAuxiliar)`
 - 1 *(não é preciso fazer nada)*

```
static void moverDiscos(int n, String origem, String destino, String auxiliar)
{
    assert n >= 0;

    if (n > 0)
    {
        moverDiscos(n-1, origem, auxiliar, destino);
        out.println("Move disco "+n+" da torre "+origem+" para a torre "+destino);
        moverDiscos(n-1, auxiliar, destino, origem);
    }
}
```

- E se tentarmos implementar uma solução com o método iterativo?
- Existe solução para esse problema (como para qualquer outro algoritmo recursivo) mas a implementação é bastante complexa!


```
static void moverDiscos(int n, String origem, String destino, String auxiliar)
{
    assert n >= 0;

    if (n > 0)
    {
        moverDiscos(n-1, origem, auxiliar, destino);
        out.println("Move disco "+n+" da torre "+origem+" para a torre "+destino);
        moverDiscos(n-1, auxiliar, destino, origem);
    }
}
```

- E se tentarmos implementar uma solução com o método iterativo?
- Existe solução para esse problema (como para qualquer outro algoritmo recursivo) mas a implementação é bastante complexa!

```
static void moverDiscos(int n, String origem, String destino, String auxiliar)
{
    assert n >= 0;

    if (n > 0)
    {
        moverDiscos(n-1, origem, auxiliar, destino);
        out.println("Move disco "+n+" da torre "+origem+" para a torre "+destino);
        moverDiscos(n-1, auxiliar, destino, origem);
    }
}
```

- E se tentarmos implementar uma solução com o método iterativo?
- Existe solução para esse problema (como para qualquer outro algoritmo recursivo) mas a implementação é bastante complexa!

```
static void moverDiscos(int n, String origem, String destino, String auxiliar)
{
    assert n >= 0;

    if (n > 0)
    {
        moverDiscos(n-1, origem, auxiliar, destino);
        out.println("Move disco "+n+" da torre "+origem+" para a torre "+destino);
        moverDiscos(n-1, auxiliar, destino, origem);
    }
}
```

- E se tentarmos implementar uma solução com o método iterativo?
- Existe solução para esse problema (como para qualquer outro algoritmo recursivo) mas a implementação é bastante complexa!

Definição Recursiva: Condições de Sanidade

- Uma definição recursiva útil requer que:
 - 1. Exista pelo menos uma alternativa não recursiva (CASO DE BASE);
 - 2. Todas as alternativas recursivas estejam bem fundadas, isto é, diferentes do original (VALIDADE);
 - 3. Em cada alternativa recursiva, o tamanho (2) varie de forma a aproximar-se de um caso base (1) (CONVERGÊNCIA).
- As condições (1) e (2) são **necessárias**. As três juntas são **suficientes** para garantir a terminação da recursão.

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Definição Recursiva: Condições de Sanidade

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- Uma definição recursiva útil requer que:
 - 1 Exista pelo menos uma alternativa não recursiva (**CASO(S) LIMITE**);
 - 2 Todas as alternativas recursivas ocorram num contexto diferente do original (**VARIABILIDADE**);
 - 3 Em cada alternativa recursiva, o contexto (2) varie de forma a aproximar-se de um caso limite (1) (**CONVERGÊNCIA**).
- As condições (1) e (2) são **necessárias**. As três juntas são **suficientes** para garantir a terminação da recursão.

Definição Recursiva: Condições de Sanidade

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- Uma definição recursiva útil requer que:
 - 1 Exista pelo menos uma alternativa não recursiva (**CASO(S) LIMITE**);
 - 2 Todas as alternativas recursivas ocorram num contexto diferente do original (**VARIABILIDADE**);
 - 3 Em cada alternativa recursiva, o contexto (2) varie de forma a aproximar-se de um caso limite (1) (**CONVERGÊNCIA**).
- As condições (1) e (2) são **necessárias**. As três juntas são **suficientes** para garantir a terminação da recursão.

Definição Recursiva: Condições de Sanidade

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- Uma definição recursiva útil requer que:
 - 1 Exista pelo menos uma alternativa não recursiva (**CASO(S) LIMITE**);
 - 2 Todas as alternativas recursivas ocorram num contexto diferente do original (**VARIABILIDADE**);
 - 3 Em cada alternativa recursiva, o contexto (2) varie de forma a aproximar-se de um caso limite (1) (**CONVERGÊNCIA**).
- As condições (1) e (2) são **necessárias**. As três juntas são **suficientes** para garantir a terminação da recursão.

Definição Recursiva: Condições de Sanidade

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- Uma definição recursiva útil requer que:
 - 1 Exista pelo menos uma alternativa não recursiva (**CASO(S) LIMITE**);
 - 2 Todas as alternativas recursivas ocorram num contexto diferente do original (**VARIABILIDADE**);
 - 3 Em cada alternativa recursiva, o contexto (2) varie de forma a aproximar-se de um caso limite (1) (**CONVERGÊNCIA**).
- As condições (1) e (2) são **necessárias**. As três juntas são **suficientes** para garantir a terminação da recursão.

Definição Recursiva: Condições de Sanidade

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- Uma definição recursiva útil requer que:
 - 1 Exista pelo menos uma alternativa não recursiva (**CASO(S) LIMITE**);
 - 2 Todas as alternativas recursivas ocorram num contexto diferente do original (**VARIABILIDADE**);
 - 3 Em cada alternativa recursiva, o contexto (2) varie de forma a aproximar-se de um caso limite (1) (**CONVERGÊNCIA**).
- As condições (1) e (2) são **necessárias**. As três juntas são **suficientes** para garantir a terminação da recursão.

Análise dos Exemplos Apresentados

Todos os exemplos de recursividade apresentados até agora verificam estas três condições:

- **Factorial:**

- $n!$ é um caso final.
- $n!$ expressa em função de $(n-1)!$ e n é $n \cdot (n-1)!$.
- A sequência $n, n-1, \dots$ converge para 0.

- **Combinações:**

- $C(n, 0)$ e $C(n, n)$ são casos finais.
- $C(n, k)$ expressa em função de $C(n-1, k)$ e $C(n-1, k-1)$.
- A sequência para k ou $k-1$ converge para 0.

- **Torres de Hanói:**

- Mover 1 disco (ou 0 discos) é trivial.
- $moveTower(n, \dots)$ expressa em função de $moveTower(n-1, \dots)$.
- n converge para 1 (ou 0).

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Análise dos Exemplos Apresentados

Todos os exemplos de recursividade apresentados até agora verificam estas três condições:

- **Factorial:**

$T(n) = T(n-1) + 1$
 $T(0) = 1$
A função $T(n)$ representa o número de chamadas de T para calcular $n!$.
A função $T(n)$ é calculada recursivamente.

- **Combinações:**

$T(n, k) = T(n-1, k) + T(n-1, k-1) + 1$
 $T(n, 0) = 1$
 $T(0, k) = 1$
A função $T(n, k)$ representa o número de chamadas de T para calcular o número de combinações de n elementos tomados k a k .
A função $T(n, k)$ é calculada recursivamente.

- **Torres de Hanói:**

$T(n) = 2T(n-1) + 1$
 $T(1) = 1$
A função $T(n)$ representa o número de chamadas de T para calcular o número de movimentos necessários para mover n discos de uma torre para outra.
A função $T(n)$ é calculada recursivamente.

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Análise dos Exemplos Apresentados

Todos os exemplos de recursividade apresentados até agora verificam estas três condições:

- **Factorial:**

- 1 $f(0)$ é um caso limite.
- 2 $f(n)$ expresso em função de $f(n-1)$ e $n \neq n-1, \forall n$.
- 3 A sucessão $n, n-1, \dots$ converge para 0.

- **Combinações:**

- 1 $C(n, 0)$ e $C(n, n)$ são casos limite.
- 2 $C(n, k)$ expresso em função de $C(n-1, k)$ e $C(n-1, k-1)$.
- 3 n converge para k ou k converge para 0.

- **Torres de Hanói:**

- 1 Mover 1 disco (ou 0 discos) é trivial.
- 2 $moveTorre(n, \dots)$ expresso em função de $moveTorre(n-1, \dots)$.
- 3 n converge para 1 (ou 0).

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Análise dos Exemplos Apresentados

Todos os exemplos de recursividade apresentados até agora verificam estas três condições:

- **Factorial:**

- 1 $f(0)$ é um caso limite.
- 2 $f(n)$ expresso em função de $f(n-1)$ e $n \neq n-1, \forall n$.
- 3 A sucessão $n, n-1, \dots$ converge para 0.

- **Combinações:**

- 1 $C(n, 0)$ e $C(n, n)$ são casos limite.
- 2 $C(n, k)$ expresso em função de $C(n-1, k)$ e $C(n-1, k-1)$.
- 3 n converge para k ou k converge para 0.

- **Torres de Hanói:**

- 1 Mover 1 disco (ou 0 discos) é trivial.
- 2 $moveTorre(n, \dots)$ expresso em função de $moveTorre(n-1, \dots)$.
- 3 n converge para 1 (ou 0).

Análise dos Exemplos Apresentados

Todos os exemplos de recursividade apresentados até agora verificam estas três condições:

- **Factorial:**

- 1 $f(0)$ é um caso limite.
- 2 $f(n)$ expresso em função de $f(n-1)$ e $n \neq n-1, \forall n$.
- 3 A sucessão $n, n-1, \dots$ converge para 0.

- **Combinações:**

- 1 $C(n, 0)$ e $C(n, n)$ são casos limite.
- 2 $C(n, k)$ expresso em função de $C(n-1, k)$ e $C(n-1, k-1)$.
- 3 n converge para k ou k converge para 0.

- **Torres de Hanói:**

- 1 Mover 1 disco (ou 0 discos) é trivial.
- 2 $moveTorre(n, \dots)$ expresso em função de $moveTorre(n-1, \dots)$.
- 3 n converge para 1 (ou 0).

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Todos os exemplos de recursividade apresentados até agora verificam estas três condições:

- **Factorial:**

- 1 $f(0)$ é um caso limite.
- 2 $f(n)$ expresso em função de $f(n-1)$ e $n \neq n-1, \forall n$.
- 3 A sucessão $n, n-1, \dots$ converge para 0.

- **Combinações:**

- 1 $C(n, 0)$ e $C(n, n)$ são casos limite.
- 2 $C(n, k)$ expresso em função de $C(n-1, k)$ e $C(n-1, k-1)$.
- 3 n converge para k ou k converge para 0.

- **Torres de Hanói:**

- 1 Mover 1 disco (ou 0 discos) é trivial.
- 2 $moveTorre(n, \dots)$ expresso em função de $moveTorre(n-1, \dots)$.
- 3 n converge para 1 (ou 0).

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Todos os exemplos de recursividade apresentados até agora verificam estas três condições:

- **Factorial:**

- 1 $f(0)$ é um caso limite.
- 2 $f(n)$ expresso em função de $f(n-1)$ e $n \neq n-1, \forall n$.
- 3 A sucessão $n, n-1, \dots$ converge para 0.

- **Combinações:**

- 1 $C(n, 0)$ e $C(n, n)$ são casos limite.
- 2 $C(n, k)$ expresso em função de $C(n-1, k)$ e $C(n-1, k-1)$.
- 3 n converge para k ou k converge para 0.

- **Torres de Hanói:**

- 1 Mover 1 disco (ou 0 discos) é trivial.
- 2 $moveTorre(n, \dots)$ expresso em função de $moveTorre(n-1, \dots)$.
- 3 n converge para 1 (ou 0).

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Todos os exemplos de recursividade apresentados até agora verificam estas três condições:

- **Factorial:**

- 1 $f(0)$ é um caso limite.
- 2 $f(n)$ expresso em função de $f(n-1)$ e $n \neq n-1, \forall n$.
- 3 A sucessão $n, n-1, \dots$ converge para 0.

- **Combinações:**

- 1 $C(n, 0)$ e $C(n, n)$ são casos limite.
- 2 $C(n, k)$ expresso em função de $C(n-1, k)$ e $C(n-1, k-1)$.
- 3 n converge para k ou k converge para 0.

- **Torres de Hanói:**

- 1 Mover 1 disco (ou 0 discos) é trivial.
- 2 $moveTorre(n, \dots)$ expresso em função de $moveTorre(n-1, \dots)$.
- 3 n converge para 1 (ou 0).

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Todos os exemplos de recursividade apresentados até agora verificam estas três condições:

- **Factorial:**

- 1 $f(0)$ é um caso limite.
- 2 $f(n)$ expresso em função de $f(n-1)$ e $n \neq n-1, \forall n$.
- 3 A sucessão $n, n-1, \dots$ converge para 0.

- **Combinações:**

- 1 $C(n, 0)$ e $C(n, n)$ são casos limite.
- 2 $C(n, k)$ expresso em função de $C(n-1, k)$ e $C(n-1, k-1)$.
- 3 n converge para k ou k converge para 0.

- **Torres de Hanói:**

- 1 Mover 1 disco (ou 0 discos) é trivial.
- 2 $moveTorre(n, \dots)$ expresso em função de $moveTorre(n-1, \dots)$.
- 3 n converge para 1 (ou 0).

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Todos os exemplos de recursividade apresentados até agora verificam estas três condições:

- **Factorial:**

- 1 $f(0)$ é um caso limite.
- 2 $f(n)$ expresso em função de $f(n-1)$ e $n \neq n-1, \forall n$.
- 3 A sucessão $n, n-1, \dots$ converge para 0.

- **Combinações:**

- 1 $C(n, 0)$ e $C(n, n)$ são casos limite.
- 2 $C(n, k)$ expresso em função de $C(n-1, k)$ e $C(n-1, k-1)$.
- 3 n converge para k ou k converge para 0.

- **Torres de Hanói:**

- 1 Mover 1 disco (ou 0 discos) é trivial.
- 2 $moveTorre(n, \dots)$ expresso em função de $moveTorre(n-1, \dots)$.
- 3 n converge para 1 (ou 0).

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Todos os exemplos de recursividade apresentados até agora verificam estas três condições:

- **Factorial:**

- 1 $f(0)$ é um caso limite.
- 2 $f(n)$ expresso em função de $f(n-1)$ e $n \neq n-1, \forall n$.
- 3 A sucessão $n, n-1, \dots$ converge para 0.

- **Combinações:**

- 1 $C(n, 0)$ e $C(n, n)$ são casos limite.
- 2 $C(n, k)$ expresso em função de $C(n-1, k)$ e $C(n-1, k-1)$.
- 3 n converge para k ou k converge para 0.

- **Torres de Hanói:**

- 1 Mover 1 disco (ou 0 discos) é trivial.
- 2 $moveTorre(n, \dots)$ expresso em função de $moveTorre(n-1, \dots)$.
- 3 n converge para 1 (ou 0).

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Todos os exemplos de recursividade apresentados até agora verificam estas três condições:

- **Factorial:**

- 1 $f(0)$ é um caso limite.
- 2 $f(n)$ expresso em função de $f(n-1)$ e $n \neq n-1, \forall n$.
- 3 A sucessão $n, n-1, \dots$ converge para 0.

- **Combinações:**

- 1 $C(n, 0)$ e $C(n, n)$ são casos limite.
- 2 $C(n, k)$ expresso em função de $C(n-1, k)$ e $C(n-1, k-1)$.
- 3 n converge para k ou k converge para 0.

- **Torres de Hanói:**

- 1 Mover 1 disco (ou 0 discos) é trivial.
- 2 $moveTorre(n, \dots)$ expresso em função de $moveTorre(n-1, \dots)$.
- 3 n converge para 1 (ou 0).

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Todos os exemplos de recursividade apresentados até agora verificam estas três condições:

- **Factorial:**

- 1 $f(0)$ é um caso limite.
- 2 $f(n)$ expresso em função de $f(n-1)$ e $n \neq n-1, \forall n$.
- 3 A sucessão $n, n-1, \dots$ converge para 0.

- **Combinações:**

- 1 $C(n, 0)$ e $C(n, n)$ são casos limite.
- 2 $C(n, k)$ expresso em função de $C(n-1, k)$ e $C(n-1, k-1)$.
- 3 n converge para k ou k converge para 0.

- **Torres de Hanói:**

- 1 Mover 1 disco (ou 0 discos) é trivial.
- 2 $moveTorre(n, \dots)$ expresso em função de $moveTorre(n-1, \dots)$.
- 3 n converge para 1 (ou 0).

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Todos os exemplos de recursividade apresentados até agora verificam estas três condições:

- **Factorial:**

- 1 $f(0)$ é um caso limite.
- 2 $f(n)$ expresso em função de $f(n-1)$ e $n \neq n-1, \forall n$.
- 3 A sucessão $n, n-1, \dots$ converge para 0.

- **Combinações:**

- 1 $C(n, 0)$ e $C(n, n)$ são casos limite.
- 2 $C(n, k)$ expresso em função de $C(n-1, k)$ e $C(n-1, k-1)$.
- 3 n converge para k ou k converge para 0.

- **Torres de Hanói:**

- 1 Mover 1 disco (ou 0 discos) é trivial.
- 2 $moveTorre(n, \dots)$ expresso em função de $moveTorre(n-1, \dots)$.
- 3 n converge para 1 (ou 0).

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Exemplo de casos atípicos

• Função *McCarthy 91*:

```
static int mc_carthy91(int n) {
    assert n > 0;
    int result;
    if (n > 100)
        result = n - 10;
    else
        result = mc_carthy91(mc_carthy91(n + 11));
    return result;
}
```

• *Curioso que funciona, mas é tão complexo de analisar que nem vale a pena tentar.*

• Conjectura de *Collatz* ($3n + 1$):

```
static long collatz(long n) {
    assert n > 0;
    long result = n;
    if (n == 1)
        result = 1;
    else if (n % 2 == 0)
        result = collatz(n / 2);
    else
        result = collatz(3 * n + 1);
    return result;
}
```

• *Curioso que nem funciona sempre, mas ninguém sabe demonstrar.*

Recursividade

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Exemplo de casos atípicos

- Função *McCarthy 91*:

```
static int mc_carthy91(int n) {  
    assert n > 0;  
    int result;  
    if (n > 100)  
        result = n - 10;  
    else  
        result = mc_carthy91(mc_carthy91(n + 11));  
    return result;  
}
```

- Sabe-se que termina, mas o tipo complexo de recursão dificulta a demonstração.

- Conjectura de *Collatz* ($3n + 1$):

```
static long collatz(long n) {  
    assert n > 0;  
    long result = n;  
    if (n == 1)  
        result = 1;  
    else if (n % 2 == 0)  
        result = collatz(n / 2);  
    else  
        result = collatz(3 * n + 1);  
    return result;  
}
```

- Acredita-se que termina sempre, mas ninguém o demonstrou!

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Exemplo de casos atípicos

- Função *McCarthy 91*:

```
static int mc_carthy91(int n) {  
    assert n > 0;  
    int result;  
    if (n > 100)  
        result = n - 10;  
    else  
        result = mc_carthy91(mc_carthy91(n + 11));  
    return result;  
}
```

- Sabe-se que termina, mas o tipo complexo de recursão dificulta a demonstração.

- Conjectura de *Collatz* ($3n + 1$):

```
static long collatz(long n) {  
    assert n > 0;  
    long result = n;  
    if (n == 1)  
        result = 1;  
    else if (n % 2 == 0)  
        result = collatz(n / 2);  
    else  
        result = collatz(3 * n + 1);  
    return result;  
}
```

- Acredita-se que termina sempre, mas ninguém o demonstrou!

Exemplo de casos atípicos

- Função *McCarthy 91*:

```
static int mc_carthy91(int n) {  
    assert n > 0;  
    int result;  
    if (n > 100)  
        result = n - 10;  
    else  
        result = mc_carthy91(mc_carthy91(n + 11));  
    return result;  
}
```

- Sabe-se que termina, mas o tipo complexo de recursão dificulta a demonstração.

- Conjectura de *Collatz* ($3n + 1$):

```
static long collatz(long n) {  
    assert n > 0;  
    long result = n;  
    if (n == 1)  
        result = 1;  
    else if (n % 2 == 0)  
        result = collatz(n / 2);  
    else  
        result = collatz(3 * n + 1);  
    return result;  
}
```

- *Acredita-se* que termina sempre, mas ninguém o demonstrou!

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Exemplo de casos atípicos

- Função *McCarthy 91*:

```
static int mc_carthy91(int n) {  
    assert n > 0;  
    int result;  
    if (n > 100)  
        result = n - 10;  
    else  
        result = mc_carthy91(mc_carthy91(n + 11));  
    return result;  
}
```

- Sabe-se que termina, mas o tipo complexo de recursão dificulta a demonstração.

- Conjectura de *Collatz* ($3n + 1$):

```
static long collatz(long n) {  
    assert n > 0;  
    long result = n;  
    if (n == 1)  
        result = 1;  
    else if (n % 2 == 0)  
        result = collatz(n / 2);  
    else  
        result = collatz(3 * n + 1);  
    return result;  
}
```

- *Acredita-se* que termina sempre, mas ninguém o demonstrou!

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- Na área da programação, os problemas recursivos considerados são sempre problemas em que as três condições de sanidade estão bem identificadas e podem ser implementadas.

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

Introdução

Definição

Complexidade

Relação de
Recorrência

Exemplo 1: A Função
Factorial

Relação de
Recorrência: Síntese

Exemplo 2: Cálculo
das Combinações

Relação de
Recorrência:
Classificação

Exemplo 3: Torres de
Hanói

Definição Recursiva:
Condições de
Sanidade

Casos Atípicos

Casos com Interesse

- Na área da programação, os problemas recursivos considerados são sempre problemas em que as três condições de sanidade estão bem identificadas e podem ser implementadas.