Aula 16: Relações de Recorrência

Luís Felipe

UFF

24 de Novembro de 2020

Luís Felipe

24/11/20

Dividir para conquistar

Em computação existe uma técnica muito utilizada para construirmos algoritmos.

Esta técnica é denominada divisão e conquista (divide and conquer) e consiste em: determinar a solução de problema buscando sua solução através da solução de problemas menores.

Curiosidade: Termo clássico em estratégias de guerras. Gere uma divisão de seus inimigos para aí sim, conquistá-los. Estratégia utilizado por Júlio César e termo conhecido desde a Antiguidade.

Em computação: As vezes conseguimos soluções mais eficientes quando pensamos dessa forma.

Luís Felipe
24/11/20

Exemplo: Problema de Busca numa lista ordenada:

- Entrada: 2 5 8 10 15 20 25
- Saída: Posição do 16.

Possível algoritmo: Percorra do início até o fim da lista, indo para a posição seguinte se elemento for menor ao Buscado. Pare se encontrar ou se chegar ao final sem encontrar.

Resposta: 6 passos para concluir que 16 não se encontra.

Algoritmo de Busca Binária: Veja quem está no meio da lista. Se o elemento que Buscamos for menor, então descarte a segunda metade e faça a mesma análise para a primeira metade. Analogamente, se elemento que Buscamos for maior que o meio, descarte a primeira metade e faça a mesma análise para a segunda metade.

Resposta: 3 passos para concluir que 16 não se encontra.

Exemplo: Problema de ordenação de uma lista:

- Entrada: 25 5 10 2 20 8 15
- Saída: transformação em 2 5 8 10 15 20 25

Algoritmo MergeSort: divida ao meio; ordene cada metade (recursivamente); compõe soluções das metades.

Luís Felipe

24/11/20

Definição recursiva

Uma definição recursiva define objetos (funções, sequências, elementos etc.) em termos de objetos previamente definidos, da mesma classe que está sendo definida.

Exemplo: Cálculo do fatorial de um número inteiro não negativo n.

Cálculo direto, não recursivo:

$0! = 1$

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

Na forma recursiva:

$0! = 1$

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

Luís Felipe

24/11/20

Versão Algorítmica: Iterativa

Algoritmo iterativo para cálculo de $n!$

Entrada: $n \in \mathbb{Z}^+$

Saída: $n!$

```
Início
  fatorial = 1;
  Se n ≠ 0 então
   Para i = 1, ..., n faça
    fatorial = fatorial * i;
```

Luís Felipe

24/11/20

Versão Algorítmica: Recursiva

Algoritmo recursivo para cálculo de $n!$

Entrada: $n \in \mathbb{Z}^+$

Saída: $n!$

```
Func. fatorial(i)
 Se i = 0 então
   fatorial (i) = 1
 Senão
  fatorial(i) = i * fatorial(i-1)
```

Luís Felipe

24/11/20

Sequência de Fibonacci

Objetiva-se determinar o número de pares de coelhos ao final de 12 meses sob as seguintes condições:

- Inicialmente tem-se um único par de coelhos (um macho e uma fêmea) recém nascidos.
- Todo mês, após atingir 2 meses, cada par de coelhos produz um novo par de coelhos de sexos opostos.
- Nenhum coelho morre durante esse processo.

Luís Felipe

24/11/20

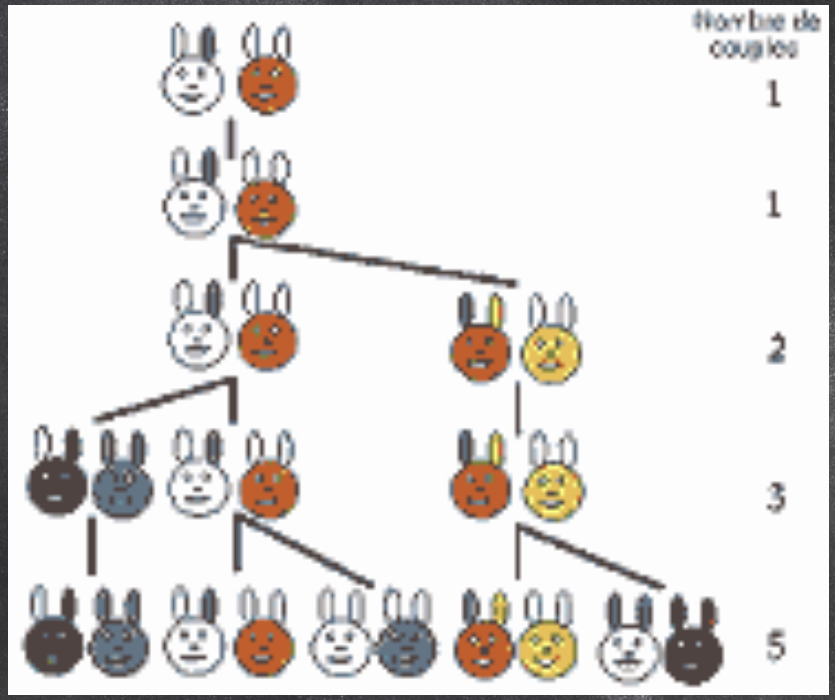


Ilustração da sequência de Fibonacci para $n = 4$

Luís Felipe

24/11/20

Um pouco mais de detalhes

Mês	*Qtd*	*Obs*
1	1	início do mês
2	1	casal do mês 1 não procriou
3	$1+1=2$	1 do mês 2 + 1 novo pra cada do mês 1
4	$2+1=3$	2 do mês 3 + 1 novo pra cada do mês 2
5	$3+2=5$	3 do mês 4 + 1 novo pra cada do mês 3
6	$5+3=8$	5 do mês 5 + 1 novo pra cada do mês 4

Luís Felipe

24/11/20

Definição recursiva

Seja $\{F_n\}$ definida recursivamente da seguinte forma:

$$\begin{cases} F_0 = 1 \\ F_1 = 1 \\ F_n = F_{n-1} + F_{n-2},\ n \geq 2 \end{cases}$$

Luís Felipe

24/11/20

Exemplo:

1. Mostre que $F_n < \left(\frac{7}{4}\right)^n$, $\forall n \in \mathbb{N}$.
   BASE: $F_1 = 1$ e $1 < \frac{7}{4}$.
   H.I: suponha $F_i < (\frac{7}{4})^k$, para todo $1 \leq i \leq k$.
   PASSO: Queremos mostrar que $F_{k+1} < (\frac{7}{4})^{k+1}$. Note que $F_{k+1} = F_k + F_{k-1}$, e HI, a desigualdade vale para ambos F_k e F_{k-1}. Assim:
   $F_{k+1} = F_k + F_{k-1} < (\frac{7}{4})^k + (\frac{7}{4})^{k-1} = (\frac{7}{4})^{k+1}[(\frac{7}{4})^{-1} + (\frac{7}{4})^{-2}] =$
   $(\frac{7}{4})^{k+1}[(\frac{4}{7}) + (\frac{4^2}{7^2})] = (\frac{7}{4})^{k+1}[(\frac{4.7+4^2}{7^2})] = (\frac{7}{4})^{k+1}[(\frac{44}{49})]$.
   Assim, como $49 > 44$, então $\frac{44}{49} < 1$. Com isso:
   $F_{k+1} < (\frac{7}{4})^{k+1}(\frac{44}{49}) < (\frac{7}{4})^{k+1}$.

Luís Felipe

24/11/20

Problema das Torres de Hanoi

Dados 3 pinos A, B e C com base circular e n discos concêntricos $D_1, D_2, \ldots, D_n$ tais que $D_1 > D_2 > \ldots > D_n$ arrumados no pino A (origem) em ordem decrescente de tamanho, objetiva-se rearrumá-los em C (destino), utilizando o pino B (trabalho) da mesma forma como estavam postos em A e sabendo que não é permitido:

- pôr um disco maior sobre um disco menor e
- a cada passo é possível movimentar apenas um disco.

Pergunta: Qual o número mínimo de jogadas para cumprir o objetivo do jogo?

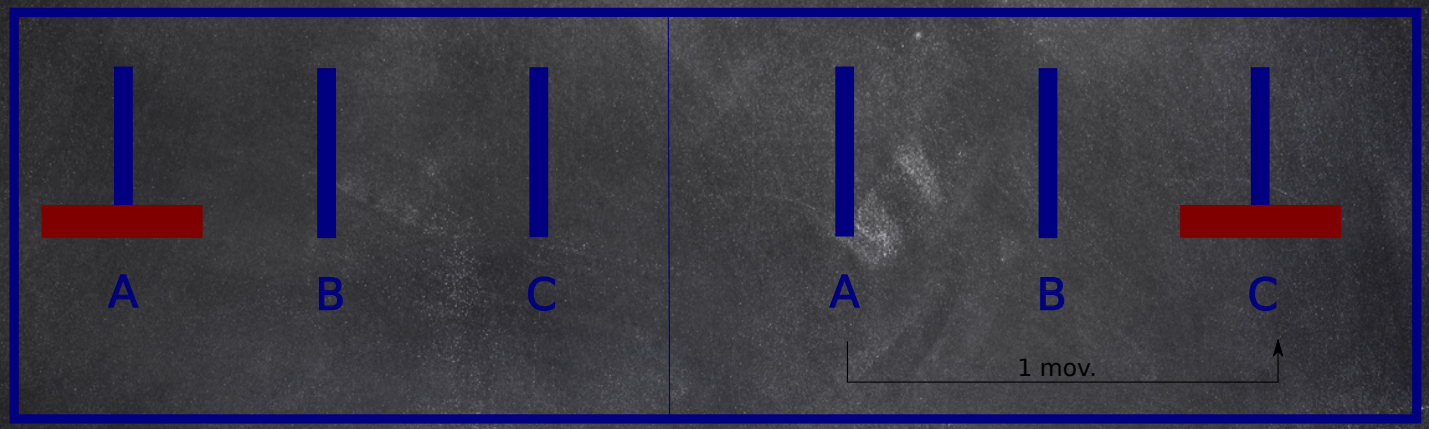
Luís Felipe

24/11/20

Torres de Hanoi

Seja T_n o número mínimo de jogadas para mover n discos do pino origem para o pino destino respeitando as regras do jogo.

- $n = 1 \Rightarrow T_1 = ??$



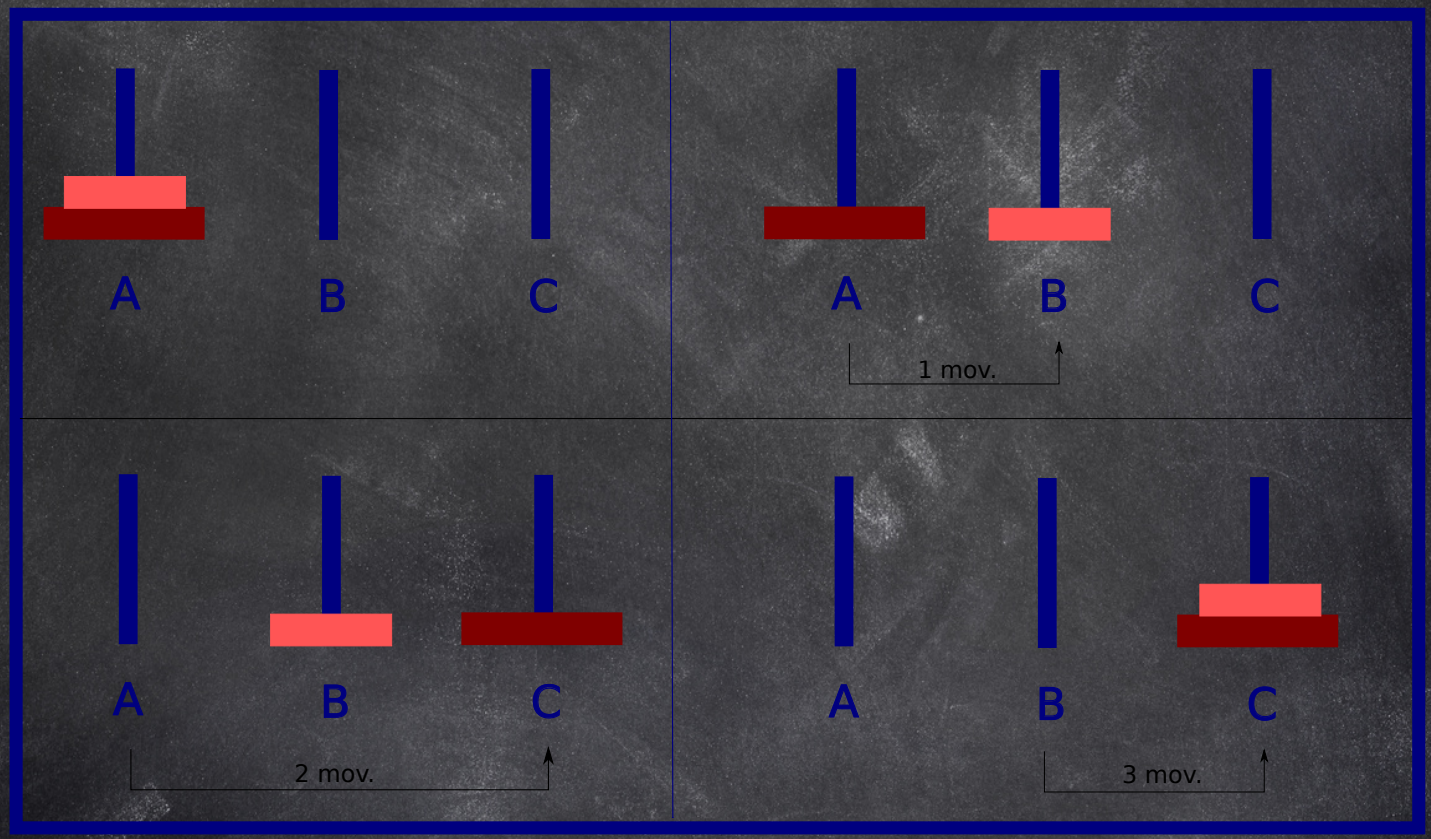
$T_1 = 1$

Luís Felipe

24/11/20

Torres de Hanoi

- $n = 2 \Rightarrow T_2 = ??$



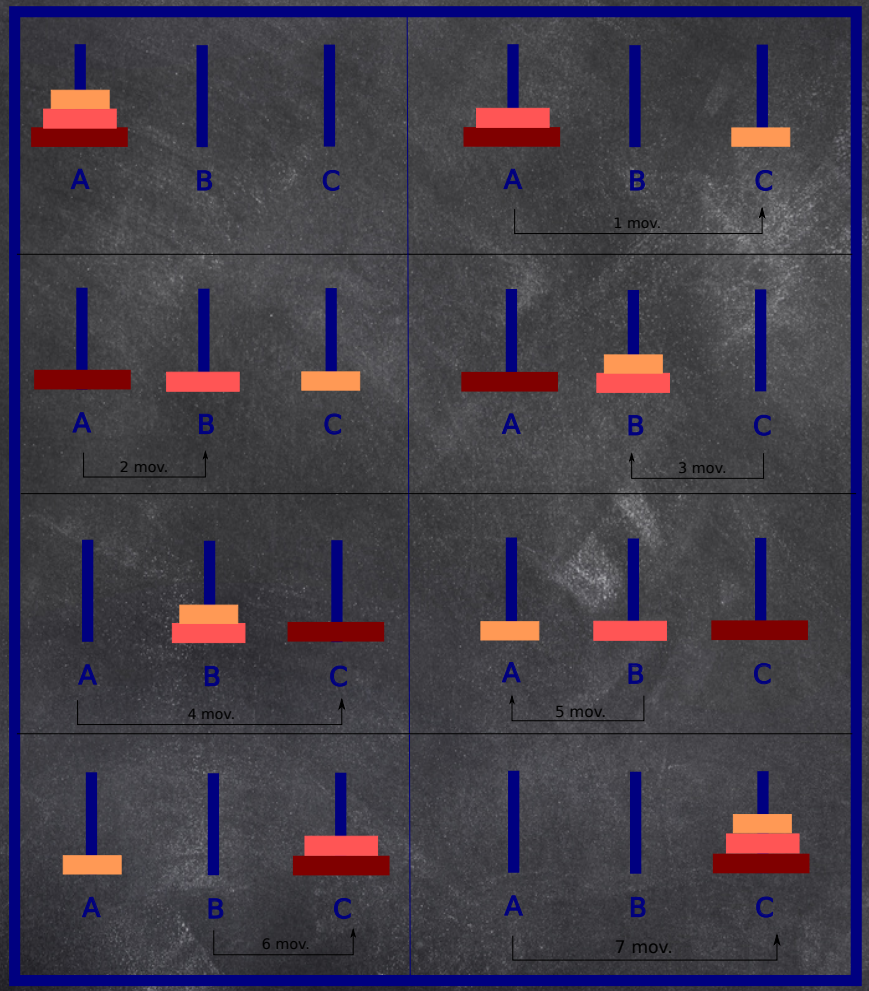
$T_2 = 3$

Luís Felipe

24/11/20

Torres de Hanoi

- $n = 3 \Rightarrow T_3 = ??$



$T_3 = 7$

Luís Felipe

24/11/20

Algoritmo

Algoritmo para mover n discos

1. Mova os $n-1$ discos do topo de A para B;
   Isto pode ser feito em T_{n-1} jogadas.
2. Mova o disco D_1 (maior) de A para C.
   Isto pode ser feito em uma jogada.
3. Mova os $n-1$ discos de B para C.
   Isto pode ser feito em T_{n-1} jogadas.

Luís Felipe

24/11/20

Relação de Recorrência

Torres de Hanoi

$$\begin{cases} T_1 &= 1 \\ T_n &= 2T_{n-1} + 1, \ n \geq 2 \end{cases}$$

Luís Felipe

24/11/20

Aplicação do PIM

Mostre que $T_n = 2^n - 1, \forall n \in \mathbb{N}$.

BASE: Pela relação de recorrência: $T_1 = 1$. Além disso, $2^1 - 1 = 1$.

HI: Suponha $T_i = 2^i - 1, \forall i \leq k$.

PASSO: Queremos mostrar que $T_{k+1} = 2^{k+1} - 1$.

Pela Relação de Recorrência, temos que: $T_{k+1} = 2T_k + 1$.
Assim, pela HI, temos que:
$T_{k+1} = 2T_k + 1 = 2(2^k - 1) + 1 = 2^{1+k} - 2 + 1 = 2^{k+1} - 1$.

Luís Felipe

24/11/20

Fórmula fechada

Note que para a relação a relação de recorrência $T_n = 2T_{n-1} + 1$, $T_1 = 1$, temos a fórmula fechada: $T_n = 2^n - 1$.

Dada uma fórmula de recorrência qualquer, como obter sua fórmula fechada?

- Substituições regressivas;
- Dedução e verificação;
- Raízes características;
- Funções geradoras.

Substituições regressivas: útil para resolver problemas do tipo $T_n = cT_{n-1} + f(n)$, onde c é constante, $n \geq 1$ e T_1 é dado.

- Torres de Hanoi pode ser resolvido dessa forma. Sequência de Fibonacci não;
- Aplicamos recursivamente a expressão até obter a base. Dessa forma, temos séries aritméticas e geométricas associadas a expressão T_n.

Luís Felipe

24/11/20

Retornando às Torres de Hanoi

Vamos obter a fórmula fechada da relação de recorrência do problema das Torres de Hanoi pelo método das substituições regressivas:

Sabemos que: $T_n = 2T_{n-1} + 1$, $n \geq 2$ e $T_1 = 1$.

$$\begin{aligned} T_n &= 2T_{n-1} + 1 \\ &= 2(2T_{n-2} + 1) + 1 \\ &= 2(2(2T_{n-3} + 1) + 1) + 1 \\ &\vdots \\ &= 2^k T_{n-k} + 2^{k-1} + 2^{k-2} + 2^{k-3} + \cdots + 2^1 + 2^0 \end{aligned}$$

Queremos chegar a base T_1. Queremos saber o valor de k para que $T_{n-k} = T_1$. Ou seja, $n - k = 1$, e assim: $k = n - 1$.

$$T_n = 2^{n-1} T_1 + 2^{(n-1)-1} + 2^{(n-1)-2} + \cdots + 2^1 + 2^0$$

Como $T_1 = 1$, então: $T_n = 2^{n-1} + 2^{n-2} + \cdots + 2^0 = \sum_{i=0}^{n-1} 2^i$

$T_n = \sum_{i=0}^{n-1} 2^i = \frac{1(2^n-1)}{2-1}$ (soma de PG finita). $T_n = 2^n - 1$.