Melhores momentos

AULA 2

Conceitos discutidos

- um pouco mais de recursão
- um pouco de análise experimental de algoritmos
- um pouco de análise algoritmos

Desempenho de binomialR1

```
 long
 binomialR1(int n, int k)
 {
1   if (n < k) return 0;
2   if (n == k || k == 0) return 1;
3   return binomialR1(n-1, k) +
4          binomialR1(n-1, k-1);
 }
```

Resolve subproblemas muitas vezes?

```
binomialR1(6,4)
  binomialR1(5,4)
    binomialR1(4,4)
    binomialR1(4,3)
      binomialR1(3,3)
      binomialR1(3,2)
        binomialR1(2,2)
        binomialR1(2,1)
          binomialR1(1,1)
          binomialR1(1,0)
  binomialR1(5,3)
    binomialR1(4,3)
      binomialR1(3,3)
      binomialR1(3,2)
        binomialR1(2,2)
```

```
        binomialR1(2,1)
          binomialR1(1,1)
          binomialR1(1,0)
      binomialR1(4,2)
        binomialR1(3,2)
          binomialR1(2,2)
          binomialR1(2,1)
            binomialR1(1,1)
            binomialR1(1,0)
        binomialR1(3,1)
          binomialR1(2,1)
            binomialR1(1,1)
            binomialR1(1,0)
          binomialR1(2,0)
binom(6,4)=15.
```

Número de chamadas recursivas

T	0	1	2	3	4	5	6	7	8	...	k
0	0	0	0	0	0	0	0	0	0	...	
1	0	0	0	0	0	0	0	0	0	...	
2	0	2	0	0	0	0	0	0	0	...	
3	0	4	4	0	0	0	0	0	0	...	
4	0	6	10	6	0	0	0	0	0	...	
5	0	8	18	18	8	0	0	0	0	...	
6	0	10	28	38	28	10	0	0	0	...	
7	0	12	40	68	68	40	12	0	0	...	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	
n											

Conclusões

Devemos **evitar** resolver o mesmo subproblema várias vezes.

O número de chamadas recursivas feitas por `binomialR1(n,k)` é

$$2 \times \binom{n}{k} - 2 .$$

Mais conclusões

O consumo de tempo da chamada `binomialR1(n,k)` é *proporcional a*

$$2 \times \binom{n}{k} - 2\,.$$

Mais conclusões ainda

Quando o valor de `k` é aproximadamente `n`/2 o consumo de tempo da chamada `binomialR1(n,k)` é *exponencial* pois

$$\binom{n}{k} \geq 2^{\frac{n}{2}}$$

Pausa para nossos comerciais

- Encontro do BCC: de 16 a 22 de agosto

  Dojo de Love2D, Iniciação científica, Intercâmbio, Desenvolvimento de Software no mundo real, Como o Google oferece infraestrutura para manter seus serviços, Computação Natural, ...

  http://www.ime.usp.br/~encontrobcc/2013/
- XVII Maratona de Programação: 17 de agosto

  http://www.ime.usp.br/~cef/XVIImaratona/
- Página do BCC:

  http://bcc.ime.usp.br/

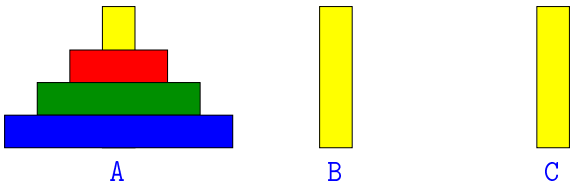
AULA 3

Mais recursão ainda

PF 2.1, 2.2, 2.3 S 5.1

http://www.ime.usp.br/~pf/algoritmos/aulas/recu.html

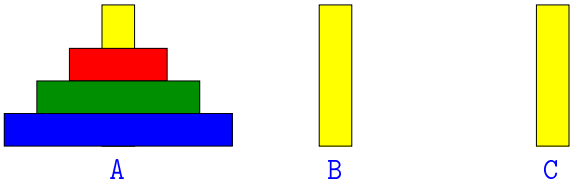
Torres de Hanoi: epílogo



Desejamos transferir `n` discos do pino `A` para o pino `C` usando o pino `B` como auxiliar e repeitando as regras:

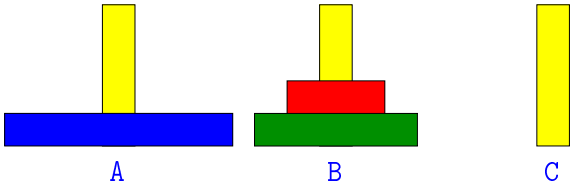
- podemos mover apenas um disco por vez;
- nunca um disco de diâmetro maior poderá ser colocado sobre um disco de diâmetro menor.

Algoritmo recursivo



Para resolver Hanoi(n,A,B,C) basta:

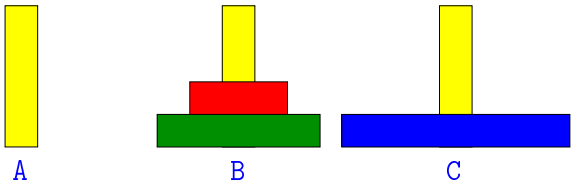
Algoritmo recursivo



Para resolver Hanoi(n,A,B,C) basta:

1. resolver Hanoi(n-1,A,C,B)

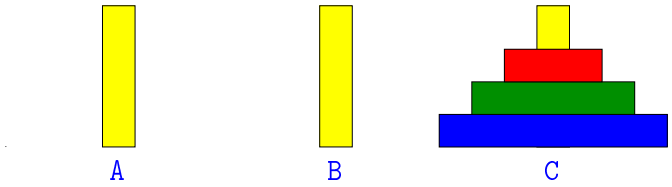
Algoritmo recursivo



Para resolver Hanoi(n,A,B,C) basta:

1. resolver Hanoi(n-1,A,C,B)
2. mover o disco n de A para C

Algoritmo recursivo



Para resolver Hanoi(n,A,B,C) basta:

1. resolver Hanoi(n-1,A,C,B)
2. mover o disco n de A para C
3. resolver Hanoi(n-1,B,A,C)

Base: sabemos revolver Hanoi(**0**,...,...,...)

Número de movimentos

Seja T(n) o número de movimentos feitos pelo algoritmo para resolver o problema das torres de Hanoi com n disco.

Temos que

$$T(0) = 0$$
$$T(n) = 2\,T(n-1) + 1 \quad \text{para } n = 1, 2, 3 \ldots$$

Quanto vale T(n)?

Recorrência

Temos que

$$\begin{aligned} T(n) &= 2\,T(n-1) + 1 \\ &= 2\,(2\,T(n-2) + 1) + 1 \\ &= 2\,(2\,(2\,T(n-3) + 1) + 1) + 1 \\ &= 2\,(2\,(2\,T(n-4) + 1) + 1) + 1) + 1 \\ &= \cdots \\ &= 2\,(2\,(2\,(\cdots(2\,T(0) + 1)) + 1) + 1) + 1 \end{aligned}$$

Recorrência

Logo,

$$\begin{aligned} T(n) &= 2^{n-1} + \cdots + 2^3 + 2^2 + 2 + 1 \\ &= 2^n - 1\,. \end{aligned}$$

n	0	1	2	3	4	5	6	7	8	9
T(n)	0	1	3	7	15	31	63	127	255	511

Conclusões

O número de movimentos feitos pela chamada `hanoi(n, ..., ..., ...)` é

$$2^n - 1\,.$$

Notemos que a função `hanoi` faz o **número mínimo** de movimentos: **não é possível** resolver o quebra-cabeça com menos movimentos.

Enquanto isto . . . os monges . . .

$T(64) = 18.446.744.073.709.551.615 \approx 1{,}84 \times 10^{19}$

Suponha que os monges façam o movimento de 1 disco por segundo(!).

$$\begin{aligned} 18 \times 10^{19}\,\text{seg} &\approx 3{,}07 \times 10^{17}\,\text{min} \\ &\approx 5{,}11 \times 10^{15}\,\text{horas} \\ &\approx 2{,}13 \times 10^{14}\,\text{dias} \\ &\approx 5{,}83 \times 10^{11}\,\text{anos.} \\ &= \mathbf{583}\,\text{bilhões de anos.} \end{aligned}$$

A idade da Terra é **4,54 bilhões de anos**.

The Tower of Hanoi Story

Taken From W.W. Rouse Ball & H.S.M. Coxeter, Mathematical Recreations and Essays, 12th edition. Univ. of Toronto Press, 1974. The De Parville account of the origen from La Nature, Paris, 1884, part I, pp. 285-286.

> *In the great temple at Benares beneath the dome that marks the centre of the world, rests a brass plate in which are fixed three diamond needles, each a cubit high and as thick as the body of a bee. On one of these needles, at the creation, God placed sixty-four discs of pure gold, the largest disk resting on the brass plate, and the others getting smaller and smaller up to the top one. This is the tower of Bramah. Day and night unceasingly the priest transfer the discs from one diamond needle to another according to the fixed and immutable laws of Bramah, which require that the priest on duty must not move more than one disc at a time and that he must place this disc on a needle so that there is no smaller disc below it. When the sixty-four discs shall have been thus transferred from the needle which at creation God placed them, to one of the other needles, tower, temple, and Brahmins alike will crumble into dust and with a thunderclap the world will vanish. The number of separate transfers of single discs which the Brahmins must make to effect the transfer of the tower is two raised to the sixty-fourth power minus 1 or 18,446,744,073,709,551,615 moves. Even if the priests move one disk every second, it would take more than 500 billion years to relocate the initial tower of 64 disks.*

http://www.rci.rutgers.edu/~cfs/472_html/AI_SEARCH/Story_TOH.html

Números de Fibonacci

PF 2.3 S 5.2

http://www.ime.usp.br/~pf/algoritmos/aulas/recu.html

Números de Fibonacci

$F_0 = 0 \qquad F_1 = 1 \qquad F_n = F_{n-1} + F_{n-2}$

n	0	1	2	3	4	5	6	7	8	9
F_n	0	1	1	2	3	5	8	13	21	34

Algoritmo recursivo para F_n:

```
long fibonacciR(int n)
{
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fibonacciR(n-1) +
           fibonacciR(n-2);
}
```

fibonacciR(4)

```
fibonacciR(4)
  fibonacciR(3)
    fibonacciR(2)
      fibonacciR(1)
      fibonacciR(0)
    fibonacciR(1)
  fibonacciR(2)
    fibonacciR(1)
    fibonacciR(0)
fibonacci(4) = 3.
```

Fibonacci iterativo

```
long fibonacciI(int n) {
  long anterior = 0, atual = 1, proximo;
  int i;
  if (n == 0) return 0;
  if (n == 1) return 1;
  for (i = 1; i < n; i++)
      {
          proximo = atual + anterior;
          anterior = atual;
          atual = proximo;
      }
  return atual;
}
```

Qual é mais eficiente?

```
meu_prompt> time ./fibonacciI 10
fibonacci(10)=55
real                    0m0.003s
user                    0m0.000s
sys                     0m0.000s

meu_prompt> time ./fibonacciR 10
fibonacci(10)=55
real                    0m0.003s
user                    0m0.000s
sys                     0m0.000s
```

Qual é mais eficiente?

```
meu_prompt> time ./fibonacciI 20
fibonacci(20) = 6765
real                    0m0.003s
user                    0m0.000s
sys                     0m0.000s

meu_prompt> time ./fibonacciR 20
fibonacci(20) = 6765
real                    0m0.003s
user                    0m0.000s
sys                     0m0.000s
```

Qual é mais eficiente?

```
meu_prompt> time ./fibonacciI 30
fibonacci(30) = 832040
real                    0m0.003s
user                    0m0.000s
sys                     0m0.000s

meu_prompt> time ./fibonacciR 30
fibonacci(30) = 832040
real                    0m0.049s
user                    0m0.044s
sys                     0m0.000s
```

Qual é mais eficiente?

```
meu_prompt> time ./fibonacciI 40
fibonacci(40) = 102334155
real                    0m0.003s
user                    0m0.000s
sys                     0m0.000s

meu_prompt> time ./fibonacciR 40
fibonacci(40) = 102334155
real                    0m4.761s
user                    0m4.756s
sys                     0m0.000s
```

Qual é mais eficiente?

```
meu_prompt> time ./fibonacciI 45
fibonacci(45) = 1134903170
real                    0m0.003s
user                    0m0.000s
sys                     0m0.000s

meu_prompt> time ./fibonacciR 45
fibonacci(45) = 1134903170
real                    0m52.809s
user                    0m52.727s
sys                     0m0.000s
```

fibonacciR(5)

fibonacciR resolve subproblemas muitas vezes.

```
fibonacciR(5)                 fibonacciR(1)
  fibonacciR(4)               fibonacciR(0)
    fibonacciR(3)           fibonacciR(3)
      fibonacciR(2)           fibonacciR(2)
        fibonacciR(1)           fibonacciR(1)
        fibonacciR(0)           fibonacciR(0)
      fibonacciR(1)           fibonacciR(1)
    fibonacciR(2)         fibonacci(5) = 5.
```

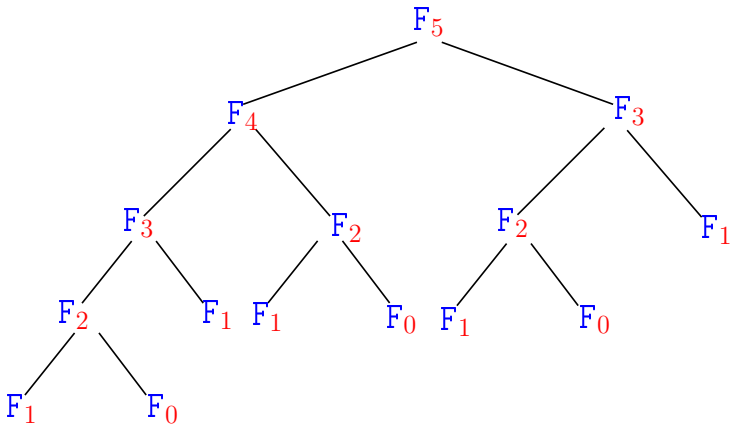
fibonacciR(8)

fibonaccir resolve subproblemas muitas vezes.

```
fibonacciR(8)
  fibonacciR(7)
    fibonacciR(6)
      fibonacciR(5)
        fibonacciR(4)
          fibonacciR(3)
            fibonacciR(2)
              fibonacciR(1)
              fibonacciR(0)
            fibonacciR(1)
          fibonacciR(2)
            fibonacciR(1)
            fibonacciR(0)
        fibonacciR(3)
          fibonacciR(2)
            fibonacciR(1)
            fibonacciR(0)
          fibonacciR(1)
      fibonacciR(4)
        fibonacciR(3)
          fibonacciR(2)
            fibonacciR(1)
            fibonacciR(0)
              fibonacciR(1)
              fibonacciR(2)
                fibonacciR(1)
                fibonacciR(0)
              fibonacciR(1)
            fibonacciR(2)
              fibonacciR(1)
              fibonacciR(0)
          fibonacciR(1)
        fibonacciR(5)
          fibonacciR(4)
            fibonacciR(3)
              fibonacciR(2)
                fibonacciR(1)
                fibonacciR(0)
              fibonacciR(1)
            fibonacciR(2)
              fibonacciR(1)
              fibonacciR(0)
          fibonacciR(3)
            fibonacciR(2)
              fibonacciR(1)
              fibonacciR(0)
            fibonacciR(1)
        fibonacciR(6)
          fibonacciR(5)
            fibonacciR(4)
              fibonacciR(3)
                fibonacciR(2)
                  fibonacciR(1)
                  fibonacciR(0)
                fibonacciR(1)
              fibonacciR(2)
                fibonacciR(1)
                fibonacciR(0)
            fibonacciR(3)
              fibonacciR(2)
                fibonacciR(1)
                fibonacciR(0)
              fibonacciR(1)
          fibonacciR(4)
            fibonacciR(3)
              fibonacciR(2)
                fibonacciR(1)
                fibonacciR(0)
              fibonacciR(1)
            fibonacciR(2)
              fibonacciR(1)
              fibonacciR(0)
fibonacci(8) = 21.
```

Árvore da recursão

fibonacciR resolve subproblemas muitas vezes.



Consumo de tempo

T(n) := número de somas feitas por fibonacciR(n)

```
    long fibonacciR(int n)
    {
1       if (n == 0) return 0;
2       if (n == 1) return 1;
3       return fibonacciR(n-1) +
4              fibonacciR(n-2);
    }
```

Consumo de tempo

linha	número de somas
1	$= 0$
2	$= 0$
3	$= T(n-1)$
4	$= T(n-2) + 1$
$T(n)$	$= T(n-1) + T(n-2) + 1$

Recorrência

$T(0) = 0$
$T(1) = 0$
$T(n) = T(n-1) + T(n-2) + 1$ para $n = 2, 3, \ldots$

Uma estimativa para $T(n)$?

Recorrência

$T(0) = 0$
$T(1) = 0$
$T(n) = T(n-1) + T(n-2) + 1$ para $n = 2, 3, \ldots$

Uma estimativa para $T(n)$?

Solução: $T(n) > (3/2)^n$ para $n \geq 6$.

n	0	1	2	3	4	5	6	7	8	9
T_n	0	0	1	2	4	7	12	20	33	54
$(3/2)^n$	1	1.5	2.25	3.38	5.06	7.59	11.39	17.09	25.63	38.44

Recorrência

Prova: $T(6) = 12 > 11.40 > (3/2)^6$ e
$T(7) = 20 > 18 > (3/2)^7$.
Se $n \geq 8$, então

$$\begin{aligned} T(n) &= T(n-1) + T(n-2) + 1 \\ &\overset{\text{hi}}{>} (3/2)^{n-1} + (3/2)^{n-2} + 1 \\ &= (3/2+1)\,(3/2)^{n-2} + 1 \\ &> (5/2)\,(3/2)^{n-2} \\ &> (9/4)\,(3/2)^{n-2} \\ &= (3/2)^2 (3/2)^{n-2} \\ &= (3/2)^n\,. \end{aligned}$$

Logo, $T(n) \geq (3/2)^n$. Consumo de tempo é **exponencial**.

Conclusão

O consumo de tempo é da função `fibonacciI(n)` é proporcional a n.

O consumo de tempo da função `fibonacciR` é **exponencial.**

Exercícios

Prove que

$$T(n) = \frac{\phi^{n+1} - \hat{\phi}^{n+1}}{\sqrt{5}} - 1 \quad \text{para } n = 0, 1, 2, \ldots$$

onde

$$\phi = \frac{1+\sqrt{5}}{2} \approx 1{,}61803 \quad \text{e} \quad \hat{\phi} = \frac{1-\sqrt{5}}{2} \approx -0{,}61803.$$

Prove que $1 + \phi = \phi^2$.

Prove que $1 + \hat{\phi} = \hat{\phi}^2$.