Análise de Algoritmos

Slides de Paulo Feofiloff

[com erros do coelho e agora também da cris]

$f1\ (x, y)$
    se $x = 1$ **ou** $y = 1$
        devolva 0
    senão
        devolva $f1(x-1, y) + f1(x, y-1) + xy$

$f2\ (x, y)$
    para $i \leftarrow 1$ **até** x **faça**
        $t[i, 1] \leftarrow 0$
    para $j \leftarrow 2$ **até** y **faça**
        $t[1, j] \leftarrow 0$
    para $i \leftarrow 2$ **até** x **faça**
        para $j \leftarrow 2$ **até** y **faça**
            $t[i, j] \leftarrow t[i-1, j] + t[i, j-1] + ij$
    devolva $t[x, y]$

Exercício da aula passada

Qual é mais rápida?

- 4 responderam que $f1$ é mais rápida.
- 1 não respondeu qual é mais rápida.
- 4 responderam que $f2$ é mais rápida mas não justificaram.
- 10 responderam que $f2$ é mais rápida e deram uma boa justificativa.
- 2 responderam que $f2$ é mais rápida mas deram uma justificativa errada.

Exercício da aula passada

Qual consome mais memória?

- 11 responderam que $f1$ consome mais memória.
- 3 deram respostas inconclusivas ou não responderam.
- 7 responderam que $f2$ consome mais memória mas não justificaram ou não deram uma justificativa conclusiva.

Frases que apareceram...

- $f2$ é a melhor pela eliminação da recursividade.
- o consumo de memória é maior em $f1$ devido à recursão.
- $f1$ consome mais memória por causa da pilha de recursão.

Frases que apareceram...

- $f2$ é a melhor pela eliminação da recursividade.
- o consumo de memória é maior em $f1$ devido à recursão.
- $f1$ consome mais memória por causa da pilha de recursão.

Uma frase a se pensar...

- $f1$ funciona para x e y maiores que 100.

Matriz t inicializada com -1 em todas as posições.

$f3\ (x,\ y)$
  se $t[x, y] \neq -1$
    devolva $t[x, y]$
  se $x = 1$ **ou** $y = 1$
    $r \leftarrow 0$
  senão
    $r \leftarrow f3(x-1, y) + f3(x, y-1) + xy$
  $t[x, y] \leftarrow r$
  devolva r

MEMOIZAÇÃO

$f4\ (x,\ y)$
  para $j \leftarrow 1$ **até** y **faça**
    $t[j] \leftarrow 0$
  para $i \leftarrow 2$ **até** x **faça**
    para $j \leftarrow 2$ **até** y **faça**
      $t[j] \leftarrow t[j] + t[j-1] + ij$
  devolva $t[y]$

Mais econômica em relação à memória.

Ordenação

$A[1 \mathinner{.\,.} n]$ é crescente se $A[1] \leq \cdots \leq A[n]$.

Problema: Rearranjar um vetor $A[1 \mathinner{.\,.} n]$ de modo que ele fique crescente.

Entra:

1										n
33	55	33	44	33	22	11	99	22	55	77

Ordenação

$A[1 \mathinner{.\,.} n]$ é crescente se $A[1] \leq \cdots \leq A[n]$.

Problema: Rearranjar um vetor $A[1 \mathinner{.\,.} n]$ de modo que ele fique crescente.

Entra:

1										n
33	55	33	44	33	22	11	99	22	55	77

Sai:

1										n
11	22	22	33	33	33	44	55	55	77	99

Ordenação por inserção

Algoritmo rearranja $A[1 \mathinner{.\,.} n]$ em ordem crescente.

ORDENA-POR-INSERÇÃO (A, n)
para $j \leftarrow 2$ **até** n **faça**
  $chave \leftarrow A[j]$

  $i \leftarrow j - 1$
  enquanto $i \geq 1$ **e** $A[i] > chave$ **faça**
    $A[i+1] \leftarrow A[i]$ $\triangleright$ desloca
    $i \leftarrow i - 1$

  $A[i+1] \leftarrow chave$ $\triangleright$ insere

Análise

Se a execução da linha i consome t_i unidades de tempo, para $i = 1, \ldots, 7$, qual o consumo total?

ORDENA-POR-INSERÇÃO (A, n)
1 **para** $j \leftarrow 2$ **até** n **faça**
2   $chave \leftarrow A[j]$

3   $i \leftarrow j - 1$
4   **enquanto** $i \geq 1$ **e** $A[i] > chave$ **faça**
5     $A[i+1] \leftarrow A[i]$ $\triangleright$ desloca
6     $i \leftarrow i - 1$

7   $A[i+1] \leftarrow chave$ $\triangleright$ insere

linha	todas as execuções da linha
1	$= \; n$
2	$= \; n-1$
3	$= \; n-1$
4	$\le \; 2+3+\cdots+n \; = \; (n-1)(n+2)/2$
5	$\le \; 1+2+\cdots+(n-1) \; = \; n(n-1)/2$
6	$\le \; 1+2+\cdots+(n-1) \; = \; n(n-1)/2$
7	$= \; n-1$
total	$\le \; (3/2)n^2 + (7/2)n - 4$

linha	todas as execuções da linha	
1	$= \; n$	$\times t_1$
2	$= \; n-1$	$\times t_2$
3	$= \; n-1$	$\times t_3$
4	$\le \; 2+3+\cdots+n \; = \; (n-1)(n+2)/2$	$\times t_4$
5	$\le \; 1+2+\cdots+(n-1) \; = \; n(n-1)/2$	$\times t_5$
6	$\le \; 1+2+\cdots+(n-1) \; = \; n(n-1)/2$	$\times t_6$
7	$= \; n-1$	$\times t_7$
total	$\le$?	

Para t_i arbitrário

linha	todas as execuções da linha	
1	$= \; n$	$\times t_1$
2	$= \; n-1$	$\times t_2$
3	$= \; n-1$	$\times t_3$
4	$\le \; 2+3+\cdots+n \; = \; (n-1)(n+2)/2$	$\times t_4$
5	$\le \; 1+2+\cdots+(n-1) \; = \; n(n-1)/2$	$\times t_5$
6	$\le \; 1+2+\cdots+(n-1) \; = \; n(n-1)/2$	$\times t_6$
7	$= \; n-1$	$\times t_7$

$$\begin{aligned}\text{total} \;\le\;& ((t_4+t_5+t_6)/2) \times n^2 \\ +\;& (t_1+t_2+t_3+t_4/2-t_5/2-t_6/2+t_7) \times n \\ -\;& (t_2+t_3+t_4+t_7)\end{aligned}$$

Para t_i arbitrário

linha	todas as execuções da linha	
1	$= \; n$	$\times t_1$
2	$= \; n-1$	$\times t_2$
3	$= \; n-1$	$\times t_3$
4	$\le \; 2+3+\cdots+n \; = \; (n-1)(n+2)/2$	$\times t_4$
5	$\le \; 1+2+\cdots+(n-1) \; = \; n(n-1)/2$	$\times t_5$
6	$\le \; 1+2+\cdots+(n-1) \; = \; n(n-1)/2$	$\times t_6$
7	$= \; n-1$	$\times t_7$
total	$\le \; c_2 \times n^2 + c_1 \times n + c_0$	

c_2, c_1, c_0 são constantes que dependem da máquina.

n^2 é para sempre! Está nas entranhas do algoritmo!

n^2+3n-3 versus n^2

n	n^2+3n-3	n^2
1	1	1
2	7	4

n^2+3n-3 versus n^2

n	n^2+3n-3	n^2
1	1	1
2	7	4
3	15	9
10	127	100

n	$n^2 + 3n - 3$	n^2
1	1	1
2	7	4
3	15	9
10	127	100
100	10297	10000
1000	1002997	1000000

n	$n^2 + 3n - 3$	n^2
1	1	1
2	7	4
3	15	9
10	127	100
100	10297	10000
1000	1002997	1000000
10000	100029997	100000000
100000	10000299997	10000000000

n^2 domina os outros termos