Agenda

Exemplo

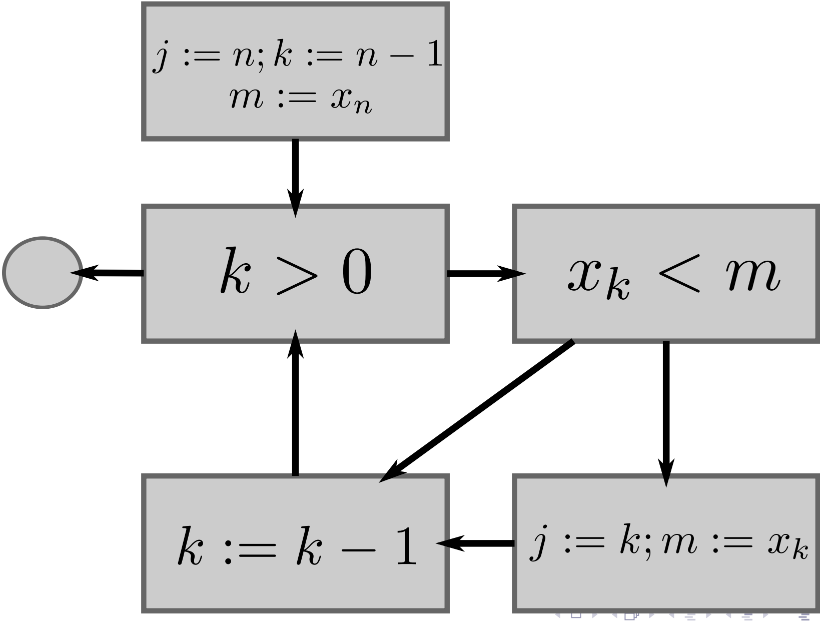
Máximo

Entrada n números $x_1, \ldots, x_n$

Saída O máximo $m = \max_{1 \le i \le n} x_i$.

$j := n; k := n - 1; m := x_n$
$\{$ **Invariante:** $m = x_j = \max_{k < i \le n} x_i$ $\}$
while $k > 0$ **do**
    if $x_k < m$ **then**
        $j := k; m := x_k$
    end if
    $k := k - 1$
end while

Fluxograma do algoritmo máximo



Custo computacional

Linha	Repetições
1	1
2	1
3	n
4	$n-1$
5	A
6	$n-1$
7	$n-1$
8	$n-1$

- Complexidade: $c_1 n + c_2 A + c_3$
- Única informação desconhecida: A
- Caso otimista: $c_1 n + c_3 = \Theta(n)$

$$x_n > \cdots > x_1 \implies A = 0$$

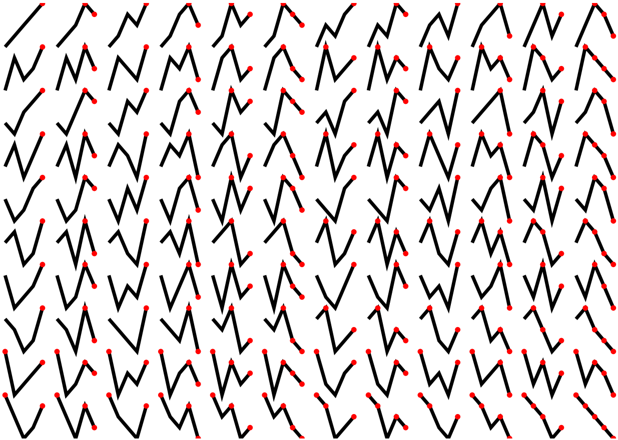
- Caso pessimista $c_1' n + c_3' = \Theta(n)$

$$x_n < \cdots < x_1 \implies A = n-1$$

- Caso médio? $O(n)$

E caso queremos saber mesmo?

- *A* não depende dos valores. A questão é:
  "Qual o número médio de picos de uma permutação randômica?"



E caso queremos saber mesmo?

- Para uma permutação π considere a **tabela de inversões** $b_1, \ldots, b_n$.
- b_i é o número de elementos na esquerda de i que são maiores que i.
- Exemplo: Para 53142

b_1	b_2	b_3	b_4	b_5
2	3	1	1	0

- Os b_i obedecem $0 \leq b_i \leq n - i$.

Tabelas de inversões

- Observação: Cada tabela de inversões corresponde com uma permutação e vice versa.
- Exemplo: A permutação correspondente com

b_1	b_2	b_3	b_4	b_5
3	1	2	1	0

  é 52413.
- Vantagem para a análise: Podemos escolher os b_i independentemente.
- Observação: i é máximo local (da esquerda), caso não tem números maiores na esquerda, i.e. para $b_i = 0$.

Número esperado de atualizações

- Seja X_i a variável aleatória $X_i = [i$ é máximo local$]$.
- Temos $\Pr[X_i = 1] = \Pr[b_i = 0] = 1/(n - i + 1)$.
- O número de máximos locais é $X = \sum_{1 \le i \le n} X_i$.
- Portanto, o número esperado de máximos locais é

$$
\begin{aligned}
E[X] &= E\left[\sum_{1 \le i \le n} X_i\right] = \sum_{1 \le i \le n} E[X_i] \\
&= \sum_{1 \le i \le n} \Pr[X_i] = \sum_{1 \le i \le n} \frac{1}{n - i + 1} = \sum_{1 \le i \le n} \frac{1}{i} = H_n
\end{aligned}
$$

- Contando atualizações: tem uma a menos que os máximos locais $H_n - 1$.

Agenda

Componentes básicos

- Componentes conjuntivas

$$\longrightarrow \boxed{C_1} \longrightarrow \boxed{C_2} \longrightarrow$$

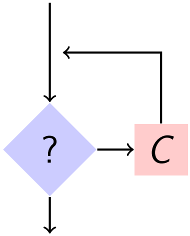
$$\boxed{O(f)} + \boxed{O(g)} = O(f+g)$$

- Componentes disjuntivas

$$\begin{tikzcd}
 & \boxed{C_1} \arrow[dr] & \\
{} \arrow[ur] \arrow[dr] & & {} \\
 & \boxed{C_2} \arrow[ur] &
\end{tikzcd}$$

$$\begin{aligned} &\boxed{O(f)} + \boxed{O(g)} \\ &= O(\max(f,g)) \\ &= O(f+g) \end{aligned}$$

Laços



$$O(k)\,O(f) = O(kf)$$

- Laço definido: Número $k = k(n)$ de iterações conhecido
- Laço indefinido: Número de iterações não conhecido
  - Estimar um limite
  - Problema é indecidível no caso geral

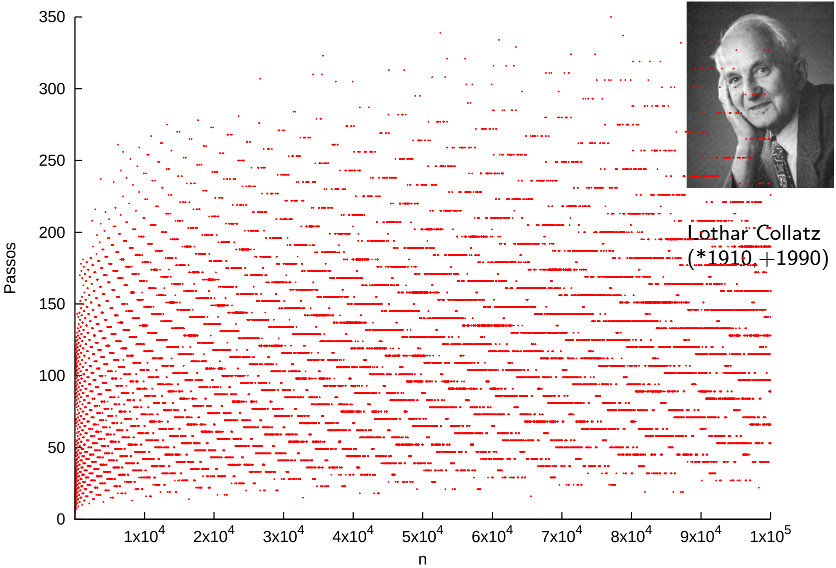
Número de iterações?

C

Entrada Número $x \in \mathbb{N}$.

```
while x ≠ 1 do
    if x mod 2 = 0 then
        x := x/2
    else
        x := 3x + 1
    end if
end while
```

Conjetura de Collatz



Bubblesort

Entrada Uma seqüência $a_1, \ldots, a_n$ de números inteiros.

Saída Uma seqüência $a_{\pi(1)}, \ldots, a_{\pi(n)}$ de números inteiros com π uma permutação de $[1, n]$ tal que para $i < j$ temos $a_{\pi(i)} \leq a_{\pi(j)}$.

```
for i:=1 to n
  for j:=1 to n−i
    if a_j > a_{j+1} then
      swap a_j, a_{j+1}
    end if
  end for
end for
```

$$\sum_{1 \leq i \leq n} \sum_{1 \leq j \leq n-i} O(1) = O(n/2(n-1)) = O(n^2)$$

Busca Binária

Entrada Número $x \in \mathbb{Z}$ e seqüência ordenada $S = a_1, \ldots, a_n$

Saída Posição i com $a_i = x$, ou -1 caso $x \notin S$.

```
i := 1; f := n; m := ⌊(f−i)/2⌋ + i
while i ≤ f do
    if a_m = x then return m
    if a_m < x then f := m − 1
    else i := m + 1
    m := ⌊(f−i)/2⌋ + i
end while
return −1
```

$$\sum_{1 \le i \le \log_2 n} O(1) = O(\log_2 n) = O(\log n)$$

Dependência de valores

P

Entrada Um número $n \in \mathbb{N}$.

```
r:=1
for i = 1, ..., n do
    r := 2r
end for
```

Complexidade?

Dependência de valores

P

Entrada: Um número $n \in \mathbb{N}$.

```
r := 1
for i = 1, ..., n do
    r := 2r
end for
```

Complexidade?

- O tamanho da entrada é $m = \Theta(\log n)$.
- Logo: a complexidade é $\Theta(2^m)$ **exponencial**.
- Algoritmo **pseudo-polinomial**: polinomial no valor, não no tamanho da entrada.