Aula 11: Algoritmos de ordenação em arranjos

O algoritmo *Quick-Sort*

David Déharbe
Programa de Pós-graduação em Sistemas e Computação
Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada

23 de março de 2015

PPgSC

Plano da aula

Introdução

O algoritmo

Análise

Estratégias para evitar o pior caso

Aperfeiçoamentos práticos

Introdução

Ref: Cormen et al. Seção 4.1 (método da substituição), Capítulo 8 (*quicksort*.

- Sem arranjo auxiliar (como ordenação por inserção);
- Baseado no conceito de divisão e conquista.
  - Processamento é realizado *antes* da divisão.
- No pior caso, ordenação em $\Theta(n^2)$;
- Em média, ordenação em $\Theta(n \lg n)$;
- Provavelmente o mais utilizado nas aplicações.

Introdução

Ref: Cormen et al. Seção 4.1 (método da substituição), Capítulo 8 (*quicksort*.

- Sem arranjo auxiliar (como ordenação por inserção);
- Baseado no conceito de divisão e conquista.
  - Processamento é realizado *antes* da divisão.
- No pior caso, ordenação em $\Theta(n^2)$;
- Em média, ordenação em $\Theta(n \lg n)$;
- Provavelmente o mais utilizado nas aplicações.
- C.A.R. Hoare, 1960.

Seja A um arranjo, para ordenar a faixa $A[l\,..\,u]$ ($l < u$):

1. Escolhe um valor na faixa: $A[l]$ (pivô).
2. Remaneje os valores da faixa em duas sub-faixas $A[l\,..\,m-1]$ e $A[m\,..\,u]$ tais que
   - todos os valores na faixa $A[l\,..\,m-1]$ são menores que $A[l]$;
   - todos os valores na faixa $A[m\,..\,u]$ são maiores ou iguais a $A[l]$;
   - o índice m é um dos resultados do remanejamento.
   - caso particular: $A[l]$ é o menor valor da faixa, e neste caso $m = l + 1$.
3. Aplique recursivamente o mesmo processamento às duas sub-faixas $A[l\,..\,m-1]$ e $A[m\,..\,u]$.

O algoritmo *Quick-Sort*

Simulação (chamadas recursivas com faixa de um elemento são omitidas)

A	l	u	m
$\langle 37, 41, 21, 14, 18, 25, 50, 11 \rangle$	1	8	

O algoritmo *Quick-Sort*

Simulação (chamadas recursivas com faixa de um elemento são omitidas)

A	l	u	m
$\langle 37, 41, 21, 14, 18, 25, 50, 11 \rangle$	1	8	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			6

O algoritmo *Quick-Sort*

Simulação (chamadas recursivas com faixa de um elemento são omitidas)

A	l	u	m
$\langle 37, 41, 21, 14, 18, 25, 50, 11 \rangle$	1	8	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			6
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	1	5	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			2

O algoritmo *Quick-Sort*

Simulação (chamadas recursivas com faixa de um elemento são omitidas)

A	l	u	m
$\langle 37, 41, 21, 14, 18, 25, 50, 11 \rangle$	1	8	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			6
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	1	5	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			2
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	2	5	

O algoritmo *Quick-Sort*

Simulação (chamadas recursivas com faixa de um elemento são omitidas)

A	l	u	m
$\langle 37, 41, 21, 14, 18, 25, 50, 11 \rangle$	1	8	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			6
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	1	5	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			2
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	2	5	
$\langle 11, 18, 14, 25, 21, 50, 41, 37 \rangle$			4

O algoritmo *Quick-Sort*

Simulação (chamadas recursivas com faixa de um elemento são omitidas)

A	l	u	m
$\langle 37, 41, 21, 14, 18, 25, 50, 11 \rangle$	1	8	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			6
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	1	5	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			2
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	2	5	
$\langle 11, 18, 14, 25, 21, 50, 41, 37 \rangle$			4
$\langle 11, 18, 14, 25, 21, 50, 41, 37 \rangle$	2	3	

O algoritmo *Quick-Sort*

Simulação (chamadas recursivas com faixa de um elemento são omitidas)

A	l	u	m
$\langle 37, 41, 21, 14, 18, 25, 50, 11 \rangle$	1	8	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			6
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	1	5	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			2
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	2	5	
$\langle 11, 18, 14, 25, 21, 50, 41, 37 \rangle$			4
$\langle 11, 18, 14, 25, 21, 50, 41, 37 \rangle$	2	3	
$\langle 11, 14, 18, 25, 21, 50, 41, 37 \rangle$	4	5	

O algoritmo *Quick-Sort*

Simulação (chamadas recursivas com faixa de um elemento são omitidas)

A	l	u	m
$\langle 37, 41, 21, 14, 18, 25, 50, 11 \rangle$	1	8	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			6
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	1	5	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			2
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	2	5	
$\langle 11, 18, 14, 25, 21, 50, 41, 37 \rangle$			4
$\langle 11, 18, 14, 25, 21, 50, 41, 37 \rangle$	2	3	
$\langle 11, 14, 18, 25, 21, 50, 41, 37 \rangle$	4	5	
$\langle 11, 14, 18, 21, 25, 50, 41, 37 \rangle$	6	8	

O algoritmo *Quick-Sort*

Simulação (chamadas recursivas com faixa de um elemento são omitidas)

A	l	u	m
$\langle 37, 41, 21, 14, 18, 25, 50, 11 \rangle$	1	8	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			6
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	1	5	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			2
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	2	5	
$\langle 11, 18, 14, 25, 21, 50, 41, 37 \rangle$			4
$\langle 11, 18, 14, 25, 21, 50, 41, 37 \rangle$	2	3	
$\langle 11, 14, 18, 25, 21, 50, 41, 37 \rangle$	4	5	
$\langle 11, 14, 18, 21, 25, 50, 41, 37 \rangle$	6	8	
$\langle 11, 14, 18, 21, 25, 37, 41, 50 \rangle$			8

O algoritmo *Quick-Sort*

Simulação (chamadas recursivas com faixa de um elemento são omitidas)

A	l	u	m
$\langle 37, 41, 21, 14, 18, 25, 50, 11 \rangle$	1	8	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			6
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	1	5	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			2
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	2	5	
$\langle 11, 18, 14, 25, 21, 50, 41, 37 \rangle$			4
$\langle 11, 18, 14, 25, 21, 50, 41, 37 \rangle$	2	3	
$\langle 11, 14, 18, 25, 21, 50, 41, 37 \rangle$	4	5	
$\langle 11, 14, 18, 21, 25, 50, 41, 37 \rangle$	6	8	
$\langle 11, 14, 18, 21, 25, 37, 41, 50 \rangle$			8

O algoritmo *Quick-Sort*

Simulação (chamadas recursivas com faixa de um elemento são omitidas)

A	l	u	m
$\langle 37, 41, 21, 14, 18, 25, 50, 11 \rangle$	1	8	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			6
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	1	5	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			2
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	2	5	
$\langle 11, 18, 14, 25, 21, 50, 41, 37 \rangle$			4
$\langle 11, 18, 14, 25, 21, 50, 41, 37 \rangle$	2	3	
$\langle 11, 14, 18, 25, 21, 50, 41, 37 \rangle$	4	5	
$\langle 11, 14, 18, 21, 25, 50, 41, 37 \rangle$	6	8	
$\langle 11, 14, 18, 21, 25, 37, 41, 50 \rangle$			8
$\langle 11, 14, 18, 21, 25, 37, 41, 50 \rangle$	6	7	

O algoritmo *Quick-Sort*

Simulação (chamadas recursivas com faixa de um elemento são omitidas)

A	l	u	m
$\langle 37, 41, 21, 14, 18, 25, 50, 11 \rangle$	1	8	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			6
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	1	5	
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$			2
$\langle 11, 21, 14, 18, 25, 50, 41, 37 \rangle$	2	5	
$\langle 11, 18, 14, 25, 21, 50, 41, 37 \rangle$			4
$\langle 11, 18, 14, 25, 21, 50, 41, 37 \rangle$	2	3	
$\langle 11, 14, 18, 25, 21, 50, 41, 37 \rangle$	4	5	
$\langle 11, 14, 18, 21, 25, 50, 41, 37 \rangle$	6	8	
$\langle 11, 14, 18, 21, 25, 37, 41, 50 \rangle$			8
$\langle 11, 14, 18, 21, 25, 37, 41, 50 \rangle$	6	7	
$\langle 11, 14, 18, 21, 25, 37, 41, 50 \rangle$			7

O algoritmo *Quick-Sort*

```
QUICK-SORT(A, l, u)
if l < u
    m = DIVISION(A, l, u)
    QUICK-SORT(A, l, m − 1)
    QUICK-SORT(A, m, u)
```

- Dê uma especificação ao algoritmo de divisão (remanejamento dos elementos da faixa).

- Pré-condição $A = \langle a_1, \ldots, a_n \rangle \wedge 1 \leq l < u \leq n$

- Pré-condição $A = \langle a_1, \ldots, a_n \rangle \wedge 1 \leq l < u \leq n$
- Pós-condição

$$
\begin{array}{ll}
 & A[1\,..\,l-1] = \langle a_1, \ldots a_{l-1} \rangle \\
\wedge & A[u+1\,..\,n] = \langle a_{u+1}, \ldots a_n \rangle \\
\wedge & A[l\,..\,u] \in permutation\langle a_l, \ldots, a_u \rangle \\
\wedge & l < m \leq u \\
\wedge & \forall i, j | l \leq i < m \leq j \leq u \cdot A[i] \leq A[j]
\end{array}
$$

- Dê uma definição do algoritmo de divisão.

O algoritmo *Quick-Sort*

O algoritmo de divisão

```
DIVISION(A, l, u)
    pivot = A[l]
    i = l − 1
    j = u + 1
    while TRUE
        repeat
            j = j − 1
        until A[j] ≤ pivot
        repeat
            i = i + 1
        until A[i] ≥ pivot
        if i < j
            SWAP(A, i, j)
        else return i
```

O algoritmo *Quick-Sort*

O algoritmo de divisão

```
DIVISION(A, l, u)
pivot = A[l]
i = l − 1
j = u + 1
while TRUE
    repeat
        j = j − 1
    until A[j] ≤ pivot
    repeat
        i = i + 1
    until A[i] ≥ pivot
    if i < j
        SWAP(A, i, j)
    else return i
```

Simulação

$$A[l..u] = \langle 5, 3, 2, 6, 4, 1, 3, 7 \rangle$$

O algoritmo *Quick-Sort*

O algoritmo de divisão

```
Division(A, l, u)
pivot = A[l]
i = l − 1
j = u + 1
while True
    repeat
        j = j − 1
    until A[j] ≤ pivot
    repeat
        i = i + 1
    until A[i] ≥ pivot
    if i < j
        Swap(A, i, j)
    else return i
```

Condições de correção ???

O algoritmo *Quick-Sort*

O algoritmo de divisão

```
Division(A, l, u)
pivot = A[l]
i = l − 1
j = u + 1
while True
    repeat
        j = j − 1
    until A[j] ≤ pivot
    repeat
        i = i + 1
    until A[i] ≥ pivot
    if i < j
        Swap(A, i, j)
    else return i
```

Condições de correção :

1. A sempre é acessado em uma posição válida

O algoritmo *Quick-Sort*

O algoritmo de divisão

```
DIVISION(A, l, u)
pivot = A[l]
i = l − 1
j = u + 1
while TRUE
    repeat
        j = j − 1
    until A[j] ≤ pivot
    repeat
        i = i + 1
    until A[i] ≥ pivot
    if i < j
        SWAP(A, i, j)
    else return i
```

Condições de correção :

1. A sempre é acessado em uma posição válida
2. $j \neq u$ quando termina o algoritmo

O algoritmo *Quick-Sort*

O algoritmo de divisão

```
DIVISION(A, l, u)
pivot = A[l]
i = l − 1
j = u + 1
while TRUE
    repeat
        j = j − 1
    until A[j] ≤ pivot
    repeat
        i = i + 1
    until A[i] ≥ pivot
    if i < j
        SWAP(A, i, j)
    else return i
```

Condições de correção :

1. A sempre é acessado em uma posição válida
2. $j \neq u$ quando termina o algoritmo
3. cada elemento de $A[l \,..\, j]$ é menor que cada elemento de $A[j+1 \,..\, u]$ quando termina o algoritmo

O algoritmo *Quick-Sort*

O algoritmo de divisão

```
DIVISION(A, l, u)
pivot = A[l]
i = l − 1
j = u + 1
while TRUE
    repeat
        j = j − 1
    until A[j] ≤ pivot
    repeat
        i = i + 1
    until A[i] ≥ pivot
    if i < j
        SWAP(A, i, j)
    else return i
```

- Qual é a complexidade do algoritmo de divisão?

O algoritmo *Quick-Sort*

O algoritmo de divisão

```
DIVISION(A, l, u)
pivot = A[l]
i = l − 1
j = u + 1
while TRUE
    repeat
        j = j − 1
    until A[j] ≤ pivot
    repeat
        i = i + 1
    until A[i] ≥ pivot
    if i < j
        SWAP(A, i, j)
    else return i
```

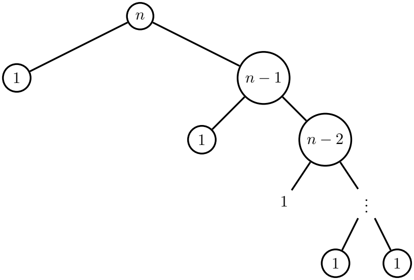
- Qual é a complexidade do algoritmo de divisão?
  - Observe que i nunca ultrapassa u e j nunca fica menor que l.
  - O número de incrementos e decrementos pertence a $\Theta(u - l)$.

- Adaptar o algoritmo para tratar elementos repetidos

- Adaptar o algoritmo para tratar elementos repetidos
- Dividir a faixa em três sub-faixas:
  1. elementos menores que o pivô,
  2. elementos iguais ao pivô,
  3. elementos maiores que o pivô.

- Qual arranjo inicial corresponde ao pior caso deste algoritmo?

- Qual arranjo inicial corresponde ao pior caso deste algoritmo?



$$\Theta(n) + \Theta(n-1) + \Theta(n-2) + \cdots + \Theta(2) = \Theta(n^2)$$

$\Longrightarrow$ Quando o arranjo é inicialmente ordenado.

- Qual arranjo inicial corresponde ao pior caso deste algoritmo?
- Como modificar o algoritmo para evitar isto?

- Qual arranjo inicial corresponde ao pior caso deste algoritmo?
- Como modificar o algoritmo para evitar isto?
  $\Longrightarrow$ Escolher outro elemento que o primeiro.

Análise de complexidade

Melhor caso: divisão perfeita

- No melhor caso, cada divisão de uma faixa de n elementos produz duas sub-faixas de tamanho $n/2$.
- O custo é $T(n) = 2.T(n/2) + \Theta(n)$.
- Teorema mestre (caso 2): $T(n) = \Theta(n \lg n)$.

- Objetivo: argumentar que o caso médio se aproxima do melhor caso.
- Exemplo: imagine que a divisão produz uma repartição de 9 para 1.
- Qual a complexidade neste caso?
- $T(n) = T(9n/10) + T(n/10) + \Theta(n)$
- O custo de cada nível de recursão é $O(n)$.
- Há $\log_{10/9} n \in \Theta(\lg n)$ níveis.
- Logo, $T(n) \in O(n \lg n)$.

- Objetivo: mostrar que o caso médio se aproxima do melhor caso.
- Exemplo: imagine que a divisão ora produz a pior repartição possível, ora produz a melhor repartição possível.
- Qual a complexidade neste caso?
- $T(n) = T(1) + T(n-1/2) + T(n-1/2) + \Theta(n) = 2T(n-1/2) + \Theta(n)$
- Logo, $T(n) \in O(n \lg n)$.

Estratégias para evitar o pior caso

Randomização

- Assumimos a existência de uma sub-rotina $\textsc{Random}(l, u)$ que retorna um número k, entre $l \leq k \leq u$ com probabilidade $1/(u - l + 1)$, e de custo $\Theta(1)$.
- Solução 1: permutar aleatoriamente os elementos da faixa antes de fazer a divisão

- Assumimos a existência de uma sub-rotina $\textsc{Random}(l, u)$ que retorna um número k, entre $l \leq k \leq u$ com probabilidade $1/(u - l + 1)$, e de custo $\Theta(1)$.
- Solução 1: permutar aleatoriamente os elementos da faixa antes de fazer a divisão
  Defina um algoritmo de complexidade linear que faz esta permutação

Estratégias para evitar o pior caso
Randomização

- Assumimos a existência de uma sub-rotina $\textsc{Random}(l, u)$ que retorna um número k, entre $l \leq k \leq u$ com probabilidade $1/(u - l + 1)$, e de custo $\Theta(1)$.
- Solução 1: permutar aleatoriamente os elementos da faixa antes de fazer a divisão
  Defina um algoritmo de complexidade linear que faz esta permutação
- Solução 2: permutar o primeiro elemento da faixa com um elemento qualquer da faixa.

$\textsc{Random-Division}(A, l, u)$

$\textsc{Swap}(A, l, \textsc{Random}(l, u))$
$\textsc{Division}(A, l, u)$

Análise no caso médio

- Análise informal: se a divisão separar uma proporção mínima de elementos,
  - a profundidade da recursão é $\Theta(\lg n)$, e
  - o custo em cada nível é Θn, logo
  - o custo total é $\Theta(n \lg n)$.
- Análise rigorosa da versão randomizada:
  - análise da divisão
  - estabelecer recorrência do tempo médio para ordenar um arranjo de n elementos
  - resolução da recorrência

Análise da divisão, caso inicial

- Seja n o tamanho da faixa.
- Seja $Pos(p)$ a posição do pivô na ordem dos elementos da faixa.
- Se $Pos(p) = 1$, então a primeira sub-faixa tem tamanho 1 (probabilidade: $1/n$).
- Se $Pos(p) = i > 1$, então a primeira sub-faixa tem tamanho $1, 2, \ldots n-1$ (probabilidade: $1/n$).
- Probabilidade da primeira sub-faixa ter tamanho 1: $2/n$.
- Probabilidade da primeira sub-faixa ter tamanho $1 < i < n$: $1/n$.

PPgSC

Análise do algoritmo randomizado

- Seja $T(n)$ o custo de ordenar uma faixa de tamanho n.
- Caso de base $T(1) \in \Theta(1)$.
- Em geral:

$$
\begin{aligned}
T(n) &= \frac{1}{n}\left(\sum_{m=1}^{n-1}(T(m) + T(n-m))\right) + \Theta(n) \\
T(n) &= \frac{2}{n}\left(\sum_{m=1}^{n-1} T(m)\right) + \Theta(n)
\end{aligned}
$$

- Resolução pelo método de substituição ($\approx$ indução)

- Dedução de limite superior de funções definidas por recorrência.
- Se aplica se temos uma ideia da fórmula fechada da recorrência: conjectura.
- A conjectura vira teorema se
  - no(s) caso(s) de base, a fórmula é correta;
  - no caso geral, substituindo os termos menores na relação, pode-se provar que o termo maior satisfaz a fórmula fechada.

Cormen, 4.1

(O método de substituição)

Exemplo

- $T(1) = 1$ e $T(n) = 2T(\lfloor n/2 \rfloor) + n$, se $n > 1$.
- Conjectura: $T(n) \leq cn \lg n$ para alguma constante $c > 0$.
- Prova:
  - Caso geral: $$\begin{aligned} T(n) &= 2T(\lfloor n/2 \rfloor) + n \\ &\leq 2c\lfloor n/2 \rfloor \lg \lfloor n/2 \rfloor + n \\ &\leq cn \lg(n/2) + n \\ &= cn \lg n - cn \lg 2 + n \\ &= cn \lg n - cn + n \\ &\leq cn \lg n \text{ escolhendo } c > 1 \end{aligned}$$

PPgSC

- $T(1) = 1$ e $T(n) = 2T(\lfloor n/2 \rfloor) + n$, se $n > 1$.
- Conjectura: $T(n) \leq cn \lg n$ para alguma constante $c > 0$.
- Prova:
  - Caso geral: $$\begin{aligned} T(n) &= 2T(\lfloor n/2 \rfloor) + n \\ &\leq 2c\lfloor n/2 \rfloor \lg \lfloor n/2 \rfloor + n \\ &\leq cn \lg(n/2) + n \\ &= cn \lg n - cn \lg 2 + n \\ &= cn \lg n - cn + n \\ &\leq cn \lg n \text{ escolhendo } c > 1 \end{aligned}$$
  - Caso de base: Supondo $T(1) = 1$, então como $c \lg 1 = 0$, a prova fracassa.

- $T(1) = 1$ e $T(n) = 2T(\lfloor n/2 \rfloor) + n$, se $n > 1$.
- Conjectura: $\forall n \geq n_0 \cdot T(n) \leq cn \lg n$ para algumas constantes $c > 0$, e $n_0 \geq 1$.

Prova:

- Caso geral: idem

- $T(1) = 1$ e $T(n) = 2T(\lfloor n/2 \rfloor) + n$, se $n > 1$.
- Conjectura: $\forall n \geq n_0 \cdot T(n) \leq cn \lg n$ para algumas constantes $c > 0$, e $n_0 \geq 1$.

Prova:

- Caso geral: idem
- Caso de base: vamos escolher c e n_0 para funcionar!
  - $T(1) = 1$, então $T(2) = 4$, e $T(3) = 5$
  - Para $n_0 = 2$, escolhemos c tal que $T(2) = 4 \leq c2 \lg 2 = 2c$

- $T(1) = 1$ e $T(n) = 2T(\lfloor n/2 \rfloor) + n$, se $n > 1$.
- Conjectura: $\forall n \geq n_0 \cdot T(n) \leq cn \lg n$ para algumas constantes $c > 0$, e $n_0 \geq 1$.

Prova:

- Caso geral: idem
- Caso de base: vamos escolher c e n_0 para funcionar!
  - $T(1) = 1$, então $T(2) = 4$, e $T(3) = 5$
  - Para $n_0 = 2$, escolhemos c tal que $T(2) = 4 \leq c2 \lg 2 = 2c$ $\Longrightarrow c = 2$

Análise do algoritmo randomizado

- Conjectura: $T(n) \leq an \lg n + b$, com $a > 0$ e $b > 0$ a determinar.
- Caso de base $T(1) \in \Theta(1)$.
  - OK

Análise do algoritmo randomizado

- Conjectura: $T(n) \leq an \lg n + b$, com $a > 0$ e $b > 0$ a determinar.
- Em geral:

$$T(n) = \frac{1}{n}\left(\sum_{m=1}^{n-1} T(m) + T(n-m)\right) + \Theta(n)$$

Análise do algoritmo randomizado

- Conjectura: $T(n) \leq an \lg n + b$, com $a > 0$ e $b > 0$ a determinar.
- Em geral:

$$
\begin{aligned}
T(n) &= \frac{1}{n}\left(\sum_{m=1}^{n-1} T(m) + T(n-m)\right) + \Theta(n) \\
&= \frac{2}{n}\left(\sum_{m=1}^{n-1} T(m)\right) + \Theta(n)
\end{aligned}
$$

Análise do algoritmo randomizado

- Conjectura: $T(n) \leq an \lg n + b$, com $a > 0$ e $b > 0$ a determinar.
- Em geral:

$$
\begin{aligned}
T(n) &= \frac{2}{n}\left(\sum_{m=1}^{n-1} T(m)\right) + \Theta(n) \\
&\leq \frac{2}{n}\left(\sum_{m=1}^{n-1} am \lg m + b\right) + \Theta(n)
\end{aligned}
$$

PPgSC

Análise do algoritmo randomizado

- Conjectura: $T(n) \le an\lg n + b$, com $a > 0$ e $b > 0$ a determinar.
- Em geral:

$$
\begin{aligned}
T(n) &\le \frac{2}{n}\left(\sum_{m=1}^{n-1} am\lg m + b\right) + \Theta(n) \\
&\le \frac{2a}{n}\sum_{m=1}^{n-1} m\lg m + \frac{2b}{n}(n-1) + \Theta(n)
\end{aligned}
$$

PPgSC

Análise do algoritmo randomizado

- Conjectura: $T(n) \leq an \lg n + b$, com $a > 0$ e $b > 0$ a determinar.
- Em geral:

$$
\begin{aligned}
T(n) &\leq \frac{2a}{n} \sum_{m=1}^{n-1} m \lg m + \frac{2b}{n}(n-1) + \Theta(n) \\
&\leq \frac{2a}{n} \left(\frac{1}{2} n^2 \lg n - \frac{1}{8} n^2 \right) + \frac{2b}{n}(n-1) + \Theta(n)
\end{aligned}
$$

$\sum_{m=1}^{n-1} m \lg m \leq \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2$.

PPgSC

Análise do algoritmo randomizado

- Conjectura: $T(n) \leq an \lg n + b$, com $a > 0$ e $b > 0$ a determinar.
- Em geral:

$$
\begin{aligned}
T(n) &\leq \frac{2a}{n}\left(\frac{1}{2}n^2 \lg n - \frac{1}{8}n^2\right) + \frac{2b}{n}(n-1) + \Theta(n) \\
&\leq an \lg n - \frac{a}{4}n + 2b + \Theta(n)
\end{aligned}
$$

Análise do algoritmo randomizado

- Conjectura: $T(n) \leq an \lg n + b$, com $a > 0$ e $b > 0$ a determinar.
- Em geral:

$$
\begin{aligned}
T(n) &\leq an \lg n - \frac{a}{4}n + 2b + \Theta(n) \\
&\leq an \lg n + b + \left(\Theta(n) + b - \frac{a}{4}n\right)
\end{aligned}
$$

Análise do algoritmo randomizado

- Conjectura: $T(n) \leq an \lg n + b$, com $a > 0$ e $b > 0$ a determinar.
- Em geral:

$$
\begin{aligned}
T(n) &\leq an \lg n + b + \left(\Theta(n) + b - \frac{a}{4} n\right) \\
&\leq an \lg n + b.
\end{aligned}
$$

Tomando a grande o suficiente para que $\frac{a}{4} n > \Theta(n) + b$.

Análise do algoritmo randomizado

- Conjectura: $T(n) \leq an \lg n + b$, com $a > 0$ e $b > 0$ a determinar.
- Em geral:

$$T(n) \leq an \lg n + b.$$

O que conclui a prova pelo método de substituição.

Análise do algoritmo randomizado

Justificando $\sum_{m=1}^{n-1} m \lg m \leq \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2$

$$\sum_{m=1}^{n-1} m \lg m \quad =$$

Análise do algoritmo randomizado

Justificando $\sum_{m=1}^{n-1} m \lg m \leq \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2$

$$\sum_{m=1}^{n-1} m \lg m = \sum_{m=1}^{\lceil n/2 \rceil - 1} m \lg m + \sum_{m=\lceil n/2 \rceil}^{n-1} m \lg m$$

- separação em duas partes do somatório

Análise do algoritmo randomizado

Justificando $\sum_{m=1}^{n-1} m \lg m \leq \frac{1}{2}n^2 \lg n - \frac{1}{8}n^2$

$$\begin{aligned} \sum_{m=1}^{n-1} m \lg m &= \sum_{m=1}^{\lceil n/2 \rceil - 1} m \lg m + \sum_{m=\lceil n/2 \rceil}^{n-1} m \lg m \\ &\leq (\lg n - 1) \sum_{m=1}^{\lceil n/2 \rceil - 1} m + \lg n \sum_{m=\lceil n/2 \rceil}^{n-1} m \end{aligned}$$

- substituição por um termo maior ou igual: $\lg(n/2)$ no primeiro somatório, e $\lg n$ no segundo;
- fatoração;
- propriedade do logaritmo: $\lg(n/2) = \lg n - \lg 2 = \lg n - 1$.

PPgSC

Análise do algoritmo randomizado

Justificando $\sum_{m=1}^{n-1} m \lg m \leq \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2$

$$
\begin{aligned}
\sum_{m=1}^{n-1} m \lg m &= \sum_{m=1}^{\lceil n/2 \rceil - 1} m \lg m + \sum_{m=\lceil n/2 \rceil}^{n-1} m \lg m \\
&\leq (\lg n - 1) \sum_{m=1}^{\lceil n/2 \rceil - 1} m + \lg n \sum_{m=\lceil n/2 \rceil}^{n-1} m \\
&= \lg n \sum_{m=1}^{n-1} m - \sum_{m=1}^{\lceil n/2 \rceil - 1} m
\end{aligned}
$$

- remanejamento dos termos (comutatividade da adição);
- junção dos somatórios.

Análise do algoritmo randomizado

Justificando $\sum_{m=1}^{n-1} m \lg m \leq \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2$

$$
\begin{aligned}
\sum_{m=1}^{n-1} m \lg m &= \sum_{m=1}^{\lceil n/2 \rceil - 1} m \lg m + \sum_{m=\lceil n/2 \rceil}^{n-1} m \lg m \\
&\leq (\lg n - 1) \sum_{m=1}^{\lceil n/2 \rceil - 1} m + \lg n \sum_{m=\lceil n/2 \rceil}^{n-1} m \\
&= \lg n \sum_{m=1}^{n-1} m - \sum_{m=1}^{\lceil n/2 \rceil - 1} m \\
&\leq \frac{1}{2} n(n-1) \lg n - \frac{1}{2}\left(\frac{n}{2} - 1\right)\frac{n}{2}
\end{aligned}
$$

- simplificação da soma dos termos de progressões aritméticas.

Análise do algoritmo randomizado

Justificando $\sum_{m=1}^{n-1} m \lg m \leq \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2$

$$
\begin{array}{rcl}
\sum_{m=1}^{n-1} m \lg m & = & \sum_{m=1}^{\lceil n/2 \rceil - 1} m \lg m + \sum_{m=\lceil n/2 \rceil}^{n-1} m \lg m \\
& \leq & (\lg n - 1) \sum_{m=1}^{\lceil n/2 \rceil - 1} m + \lg n \sum_{m=\lceil n/2 \rceil}^{n-1} m \\
& = & \lg n \sum_{m=1}^{n-1} m - \sum_{m=1}^{\lceil n/2 \rceil - 1} m \\
& \leq & \frac{1}{2} n(n-1) \lg n - \frac{1}{2}\left(\frac{n}{2} - 1\right)\frac{n}{2} \\
& \leq & \frac{1}{2} n^2 \lg n - \frac{1}{8} N^2
\end{array}
$$

Estratégias para evitar o pior caso

Valor mediano

- Pivô escolhido é o elemento com valor mediano entre $A[l]$, $A[u]$ e $A[l+u/2]$.

```
MED-3(A, l, u)
m = l + (u - l)/2
if A[l] < A[m]
        if A[m] < A[u]      return m
        else if A[l] < A[u]       return u
                else return l
else if A[m] > A[u]      return m
        else if A[l] < A[u]       return l
                else return u
```

- Se a faixa for grande, pode ser escolhido o valor mediano entre mais que três posições).
- Ler uma implementação da função *qsort* da biblioteca padrão da linguagem C.