



ALGORITMOS E COMPLEXIDADE (ARA0174)

AULA 11

DATA: 12/05/2022



Aula 11- ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMOS DE ORDENAÇÃO AVANÇADOS *ORDENAÇÃO QUICKSORT*



Estácio

Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

O Quicksort é um dos algoritmos de ordenação mais conhecidos. Por ser um dos mais eficientes é largamente utilizado/customizado em projetos de linguagens de programação. Ele pertence a categoria de Ordenação por Troca ou *Exchange Sort*.



O QuickSort foi desenvolvido em 1960 pelo cientista da computação britânico Charles Antony Richard Hoare, também conhecido como Tony Hoare ou C. A. R. Hoare.

Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

- ❑ A ideia básica é dividir o problema de ordenar um conjunto com n itens em dois problemas menores;
- ❑ Os problemas menores são ordenados independentemente;
- ❑ Os resultados são combinados para produzir a solução final.

Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

- ☐ Ordenação por troca de partições.
- ☐ Dividir e Conquistar.
- ☐ Um elemento é escolhido como pivô.
 - ☐ Valores menores que o pivô são colocados antes dele e os maiores, depois.
- ☐ Melhor caso: $O(n \log n)$.
- ☐ Pior caso (raro): $O(n^2)$.
- ☐ Não é estável.

Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

Um algoritmo de ordenação é considerado estável quando consegue preservar a ordem de registro de chaves iguais, em outras palavras se os registros aparecem na sequencia ordenada na mesma ordem em que estão na sequencia inicial. Um exemplo de algoritmo estável, ordenando a sequencia de números (chaves) com letras (registros):

EXEMPLO: 3[a], 2[b], 2[c], 1[d]

Obrigatoriamente o resultado será:

ESTÁVEL: 1[d], 2[b], 2[c], 3[a]

Os algoritmos **NÃO ESTÁVEIS** sujeitam os elementos associados aos objetos a serem ordenados:

NÃO ESTÁVEL (QUICKSORT): 1[d], 2[c], 2[b], 3[a]

Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

- Escolha um elemento (pivô) e coloque-o no início da lista.
- Varra o array a partir da esquerda até encontrar um elemento que seja maior que o pivô e pela direita até encontrar um elemento menor que o pivô.
- Os dois devem ser trocados de lugar.
- Quando os ponteiros se cruzam, troque o pivô com o elemento mais a esquerda do array.

Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

Para demonstrar o processo de Ordenação Quicksort será trabalhado o vetor abaixo onde o mesmo será ordenado de forma crescente.

Vetor Original: [9 25 10 18 5 7 15 3]

Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

Vetor Original: [9 25 10 18 5 7 15 3]

pivô (p) = 9; a = v[1] = 25; b = v[7] = 3

Como acontece o processo?

1. *Nesse exemplo seleciono o pivô: 9
(O primeiro da posição do vetor desse exemplo).*
2. *O processo: if((esquerda > pivo) && (direita <= pivo)) então*

*FAÇO A TROCA DE POSIÇÃO ENTRE ELES
(ESQUERDA E DIREITA)*

else

CONTINUO A ANÁLISE...

Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort (INICIANDO...)

Vetor Original: [9 25 10 18 5 7 15 3]

pivô (p) = 9; a = v[1] = 25; b = v[7] = 3

esquerda *direita*

0	1	2	3	4	5	6	7	
9	25	10	18	5	7	15	3	(25 > 9 ok!; 3 ≤ 9 ok!, troca)

Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

Vetor Original: [9 25 10 18 5 7 15 3]

pivô (p) = 9; a = v[1] = 25; b = v[7] = 3

0	1	2	3	4	5	6	7
9	25	10	18	5	7	15	3

9	3	10	18	5	7	15	25	(10 > 9 ok!; 15 <= 9 não!)
		↑ _a				↑ _b		

esquerda *direita*

Aula 11 - ALGORITMOS E COMPLEXIDADE


TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

Vetor Original: [9 25 10 18 5 7 15 3]

pivô (p) = 9; a = v[1] = 25; b = v[7] = 3

0	1	2	3	4	5	6	7
9	25	10	18	5	7	15	3

9	3	10	18	5	7	15	25	(10 > 9 ok!;		7 <= 9 ok!, troca)
		\uparrow_a			\uparrow_b			<i>esquerda</i>		<i>direita</i>

Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

Vetor Original: [9 25 10 18 5 7 15 3]

pivô (p) = 9; a = v[1] = 25; b = v[7] = 3

0	1	2	3	4	5	6	7
9	25	10	18	5	7	15	3

9	3	10	18	5	7	15	25
---	---	----	----	---	---	----	----

9	3	7	18	5	10	15	25
---	---	---	----	---	----	----	----

(18 > 9 ok!; 5 <= 9 ok!, troca)

↑a ↑b esquerda direita

Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

Vetor Original: [9 25 10 18 5 7 15 3]

pivô (p) = 9; a = v[1] = 25; b = v[7] = 3

0	1	2	3	4	5	6	7
9	25	10	18	5	7	15	3

9	3	10	18	5	7	15	25
---	---	----	----	---	---	----	----

9	3	7	18	5	10	15	25
---	---	---	----	---	----	----	----

9	3	7	5	18	10	15	25
			\uparrow_a	\uparrow_b			
			\uparrow_b	X	\uparrow_a		

→ cruzaram

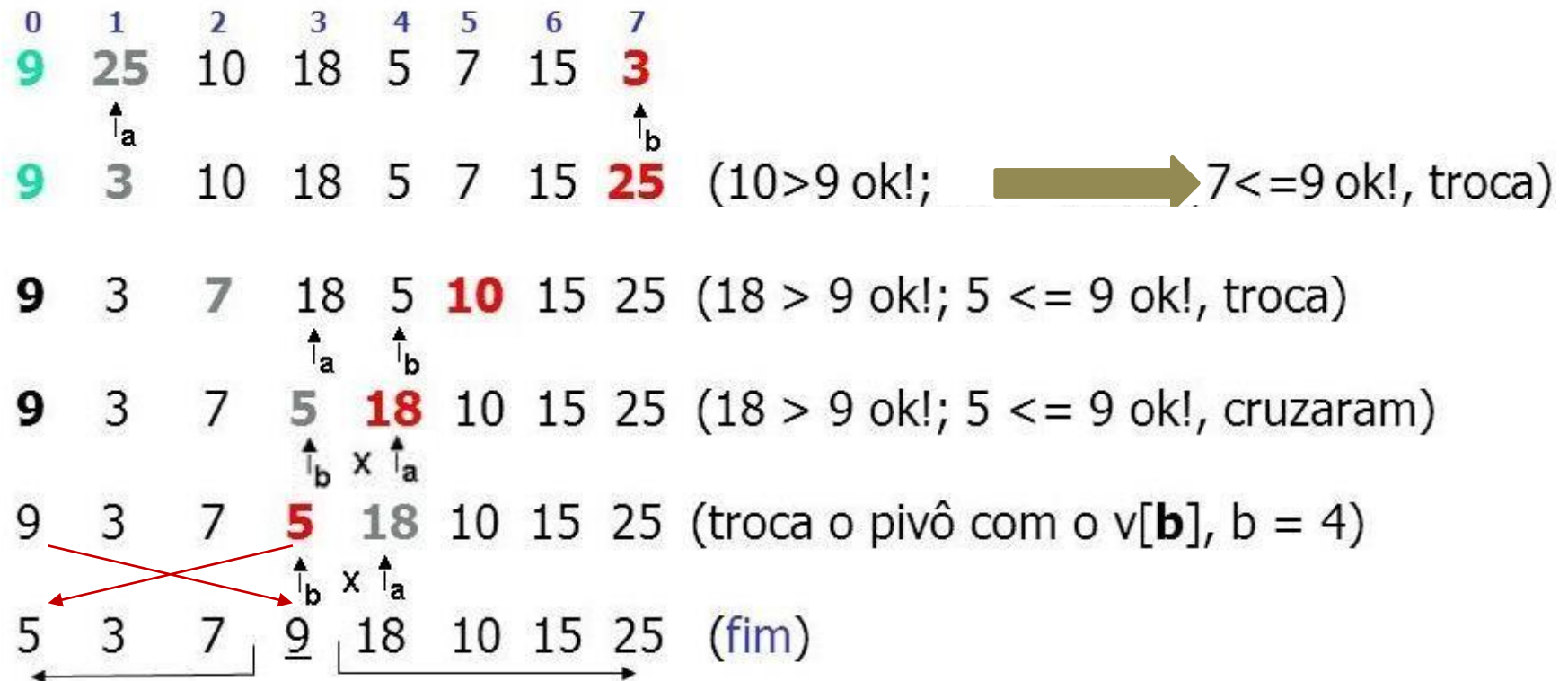
Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

Vetor Original: [9 25 10 18 5 7 15 3]

pivô (p) = 9; a = v[1] = 25; b = v[7] = 3

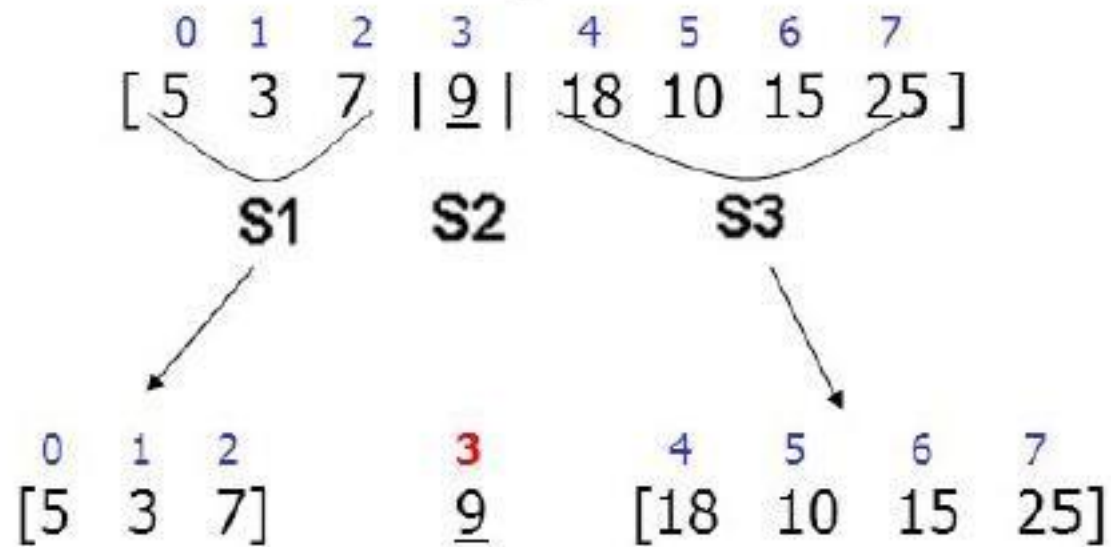


Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

Segmentos resultantes após 1º Particionamento:



Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

S1: ⁰ 5 ¹ 3 ² 7]

pivô (p) = 5 ; a = v[1] = 3 ; b = v[2] = 7

Avaliando pela esquerda
com o pivô

Avaliando pela direita
com o pivô

S1
⁰ 5 ¹ 3 ² 7
↑_a ↑_b
5 3 7
↑_b x ↑_a

(3 > 5 não!, 7 > 5 ok!; 7 ≤ 5 não!, 3 ≤ 5 ok!, cruzaram)

(troca o pivô com o v[b], b = 2)

← 3 | 5 | 7 → (fim, 2º nível de particionamento (S1))

Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

S3: ⁴ 18 ⁵ 10 ⁶ 15 ⁷ 25]

pivô (p) = 18 ; a = v[5] = 10 ; b = v[7] = 25

Avaliando pela esquerda
com o pivô

Avaliando pela direita
com o pivô

S3
⁴ 18 ⁵ 10 ⁶ 15 ⁷ 25
↑_a ↑_b

(10 > 18 não!, 15 > 18 não!, 25 > 18 ok!; 25 ≤ 18 não!, 15 ≤ 18 ok!, cruzaram)

18 10 15 25
 ↑_b x ↑_a

(troca o pivô com o v[b], b = 6)

15 10 | 18 | 25
← S4 →

(fim, 2º nível de particionamento (S3))

Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

S4: ⁴ [15 ⁵ 10]
pivô (p) = 15 ; a = v[5] = 10 ; b = v[5] = 10

S4
⁴ 15 ⁵ 10 (10 > 15 não!; 10 ≤ 15 ok!, cruzaram)
↑_a ↑_b
15 10 (troca o pivô com o v[b], b = 5)
↑_b × ↑_a

← 10 | 15 | → (fim do 2º nível de particionamento (S4))

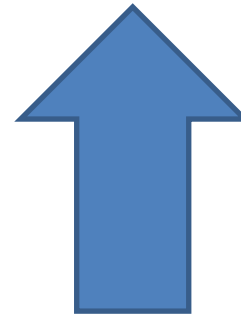

Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

■ **Vetor Original:**

0	1	2	3	4	5	6	7
9	25	10	18	5	7	15	3
5	3	7	9	18	10	15	25



[3 5 7 9 10 15 18 25]

Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

Apesar do seu desempenho no pior caso ser $O(n^2)$, *Quicksort* costuma ser, na prática, a melhor escolha:

Na média, sua performance é **excelente**;

O tempo de execução esperado é $O(n \log_2 n)$;

Executa eficientemente mesmo em ambientes com memória virtual.

Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort (CARACTERÍSTICA)

- A execução do QuickSort pode ser facilmente descrita por uma árvore binária
 - Cada nó representa uma chamada recursiva do QuickSort
 - O nó raiz é a chamada inicial

Aula 11 - ALGORITMOS E COMPLEXIDADE

TEMA 3 - ALGORITMO DE ORDENAÇÃO AVANÇADOS (ORDENAÇÃO RÁPIDA (QUICKSORT))

Quicksort

EXERCÍCIOS