

ALGORITMOS E ESTRUTURAS DE DADOS

2018/2019

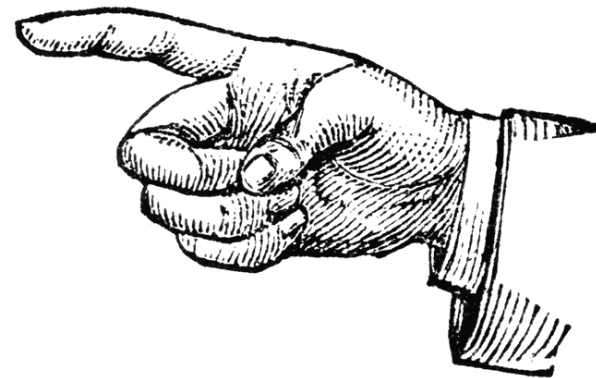
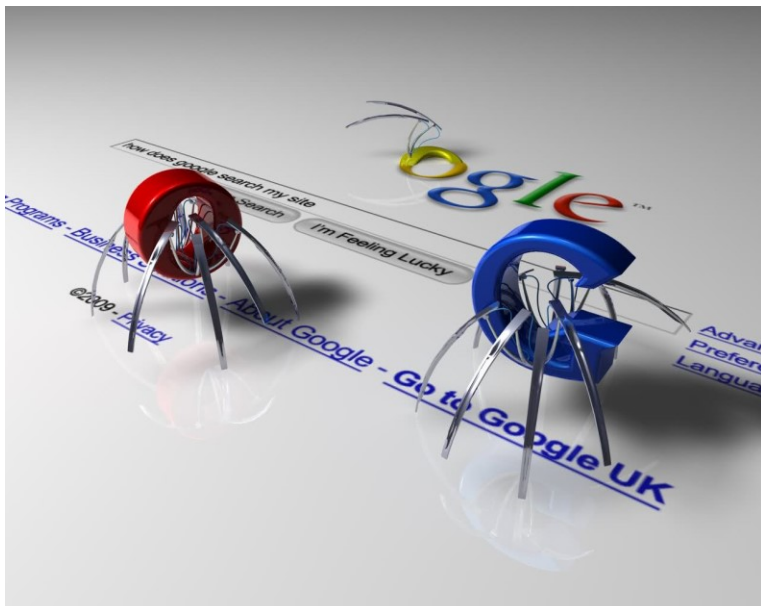
ALGORITMOS DE ORDENAÇÃO POR INDEXAÇÃO

Armanda Rodrigues

29 de Novembro de 2018

Ordenação (por indexação)

- Existem algoritmos de ordenação que possam correr assintoticamente mais rápido que $O(n \log n)$?
- A resposta é sim!
- No entanto, para que isto possa acontecer, a sequência a ordenar tem de seguir um conjunto de requisitos.



Bucket Sort

- Considera-se uma sequência S de n entradas cujas chaves são inteiros entre 0 e $C - 1$ sendo $C \geq 2$.
- S deverá ser ordenada de acordo com as chaves das entradas.
- Neste caso, é possível ordenar S em $O(n+C)$.
- Se C for $O(n)$ então a complexidade do algoritmo será $O(n)$.

Bucket Sort (1)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Não existem chaves repetidas.

Bucket Sort (1)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Não existem chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>

Bucket Sort (1)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Não existem chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>

Bucket Sort (1)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Não existem chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>

7 *4*

Bucket Sort (1)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Não existem chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>

7 *4*

Bucket Sort (1)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Não existem chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>

7 *4* *1*

Bucket Sort (1)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Não existem chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>

7 *4* *1*

Bucket Sort (1)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Não existem chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>

7 *4* *1* *9*

Bucket Sort (1)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Não existem chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>

7 4 1 9

Bucket Sort (1)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Não existem chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>

7 4 1 9 6

Bucket Sort (1)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Não existem chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>

7 4 1 9 6

Bucket Sort (1)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Não existem chaves repetidas.

Complexidade Temporal: $O(n+C)$

n – número de registos a ordenar

C – número de chaves distintas

0	1	2	3	4	5	6	7	8	9	10
<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>

7 4 1 9 6

Sequência Ordenada: *1 4 6 7 9*

Bucket Sort (2)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Podem existir chaves repetidas.

Bucket Sort (2)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Podem existir chaves repetidas.

[illegible]

Bucket Sort (2)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Podem existir chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0

Bucket Sort (2)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Podem existir chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	1	0	0	0

7

Bucket Sort (2)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Podem existir chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	1	0	0	0

7 4

Bucket Sort (2)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Podem existir chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	1	0	0	1	0	0	0

7 4

Bucket Sort (2)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Podem existir chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	1	0	0	1	0	0	0

7 4 1

Bucket Sort (2)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Podem existir chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
0	1	0	0	1	0	0	1	0	0	0

7 4 1

Bucket Sort (2)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Podem existir chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
0	1	0	0	1	0	0	1	0	0	0

7 4 1 9

Bucket Sort (2)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Podem existir chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
0	1	0	0	1	0	0	1	0	0	0

7 4 1 9

Bucket Sort (2)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Podem existir chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
0	1	0	0	1	0	0	1	0	1	0

7 4 1 9

Bucket Sort (2)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Podem existir chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
0	1	0	0	1	0	0	1	0	1	0

7 4 1 9 7

Bucket Sort (2)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Podem existir chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
0	1	0	0	1	0	0	2	0	1	0

7 4 1 9 7

Bucket Sort (2)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Podem existir chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
0	1	0	0	1	0	0	2	0	1	0

7 4 1 9 7 1

Bucket Sort (2)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Podem existir chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
0	2	0	0	1	0	0	2	0	1	0

7 4 1 9 7 1

Bucket Sort (2)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Podem existir chaves repetidas.

0	1	2	3	4	5	6	7	8	9	10
0	2	0	0	1	0	0	2	0	1	0

7 4 1 9 7 1 6

Bucket Sort (2)

- As chaves variam entre 0 e 10.
- Os registos são as chaves.
- Podem existir chaves repetidas.

Complexidade Temporal: $O(n+C)$

n – número de registos a ordenar

C – número de chaves distintas

0	1	2	3	4	5	6	7	8	9	10
0	2	0	0	1	0	1	2	0	1	0

7 4 1 9 7 1 6

Sequência Ordenada: 1 1 4 6 7 7 9

Bucket Sort (3)

Complexidade Temporal: $O(n+C)$

n – número de registos a ordenar

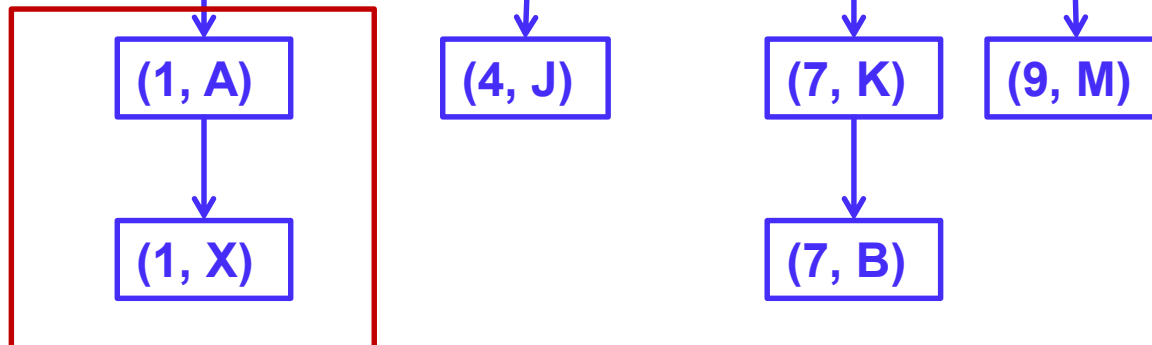
C – número de chaves distintas

- As chaves variam entre 0 e 10.
- Os registos não possuem apenas a chave.
- Podem existir chaves repetidas.

(7, K) (1, A) (9, M) (7,B) (1,X) (4, J)



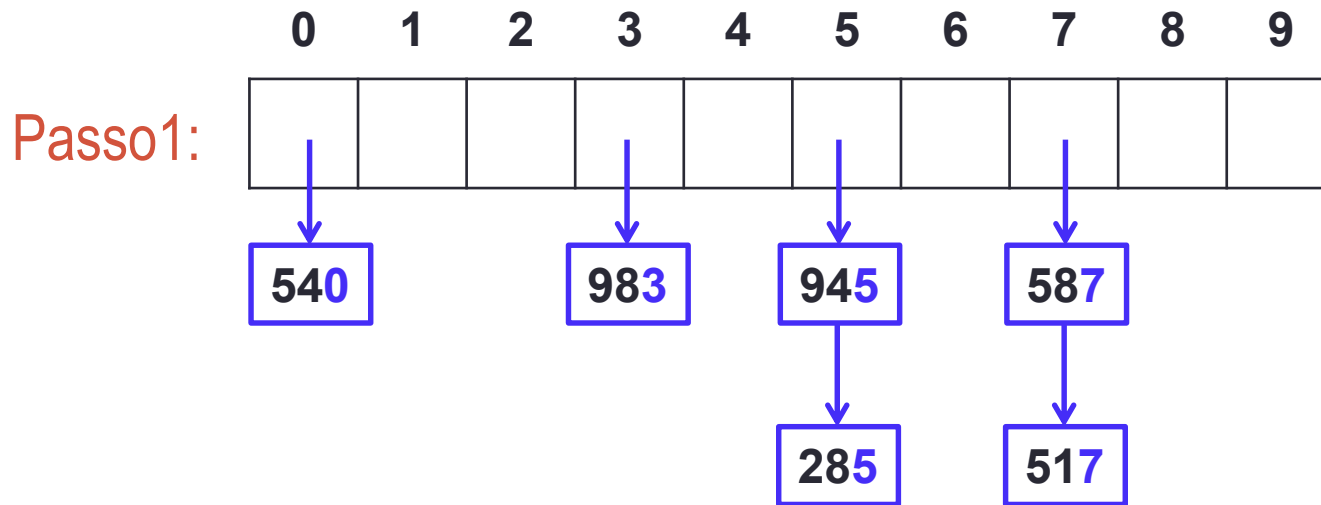
Cada elemento do vetor é uma Fila (em Lista)



Sequência Ordenada: (1, A) (1, X) (4, J) (7,K) (7,B) (9, M)

Radix Sort (1)

Sequência Inicial: **587** **945** **540** **285** **983** **517**

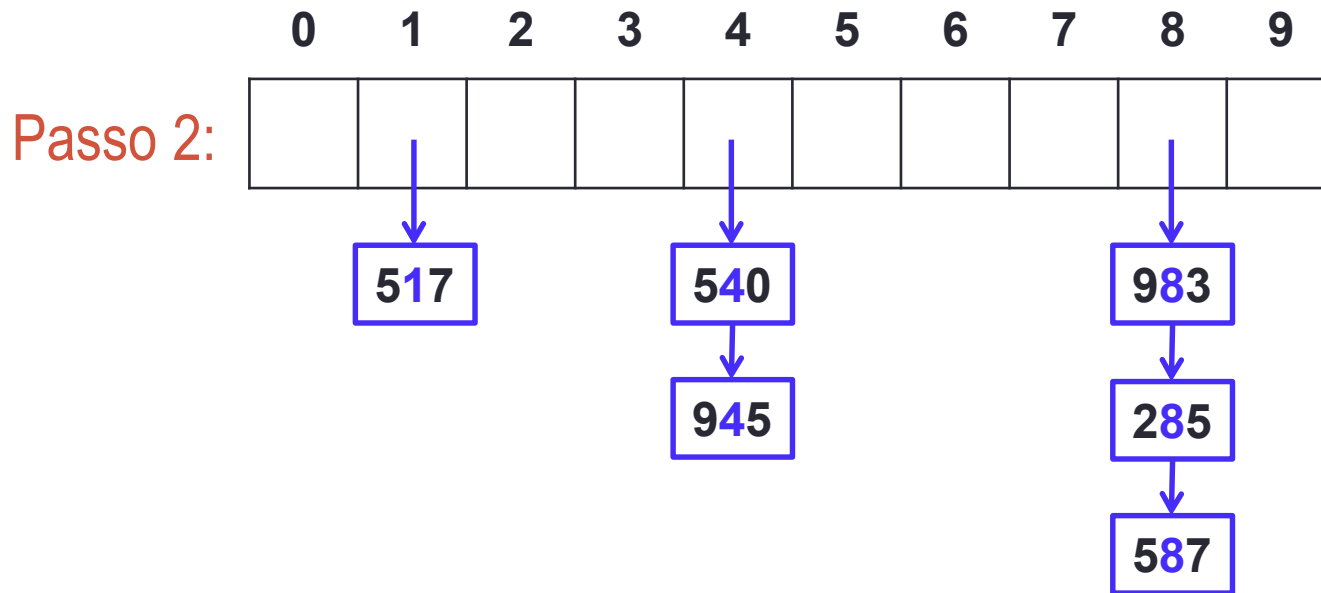


Resultado do passo 1: **540** **983** **945** **285** **587** **517**

(sequência ordenada em relação ao último dígito)

Radix Sort (2)

Resultado passo 1: **540** **983** **945** **285** **587** **517**

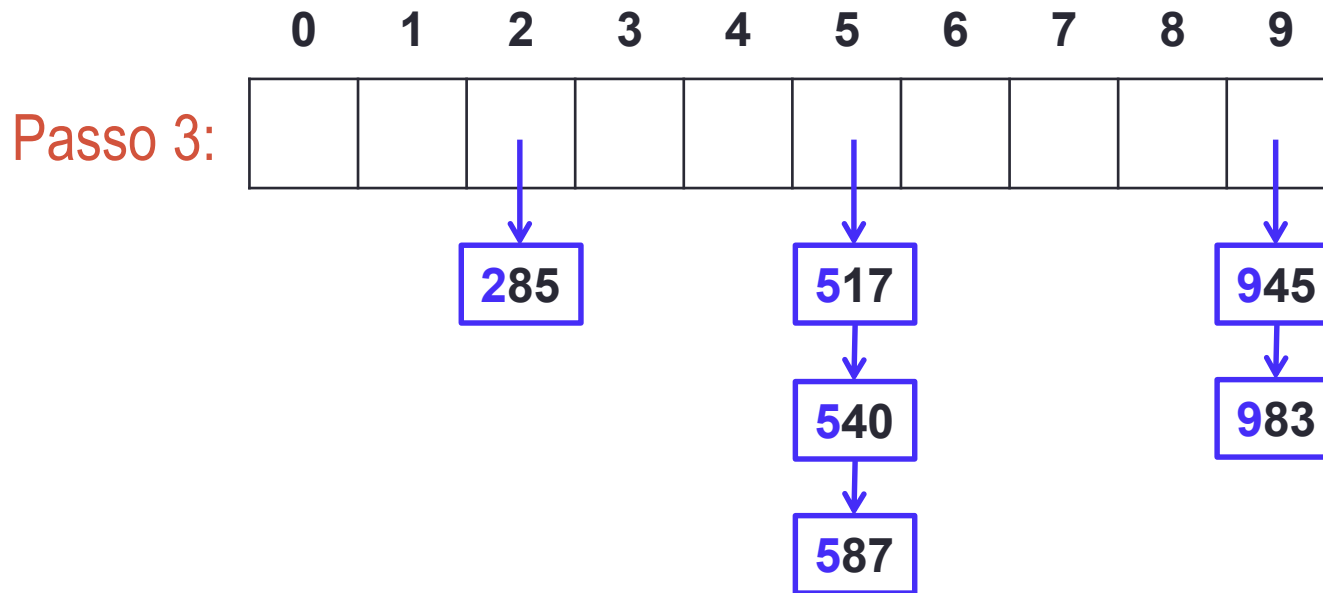


Resultado do passo 2: **517** **540** **945** **983** **285** **587**

(sequência ordenada em relação aos dois últimos dígitos)

Radix Sort (3)

Resultado passo 2: **517** **540** **945** **983** **285** **587**



Resultado do passo 3: **285** **517** **540** **587** **945** **983**

(sequência ordenada em relação aos três últimos dígitos)

Radix Sort

- Existem n registros cujas chaves são compostas por d dígitos.
 - Os dígitos variam num intervalo I pequeno (neste caso entre 0 e 9).
- Efetuam-se d passos, da direita para esquerda (do dígito menos significativo para o mais significativo).
- Num determinado passo k :
 - Distribuem-se os n registros de acordo com o k -ésimo dígito menos significativo, usando esse dígito para indexar a fila concatenável (disciplina FIFO) onde se insere o registro;
 - Concatenam-se as filas, percorrendo o vetor da posição referente ao dígito de menor valor para a direita (dígito de maior valor).
- Complexidade: $O (d (n + I))$
- Algoritmo Estável