



RADIX SORT

Apresentado por

Wallan Melo,
Clebson Pereira



Introdução

- Algoritmo de tempo linear para ordenar números inteiros e strings
- O Primeiro algoritmo computacional para o radix sort foi inventado em 1954 no MIT por Harold H. Seward.
- A ideia original de Harold era de ordenar pelo dígito mais significativo, mas atualmente é muito usado a ordenação pelo dígito menos significativo e utilizando um algoritmo de ordenação estável(ou seja, a ordem dos elementos em que há empate é preservada) auxiliar.



Lógica do Radix Sort

123	142	87	263	233	14	132

Lógica do Radix Sort

123	142	87	263	233	14	132

Comece pelo **DÍGITO MENOS SIGNIFICATIVO**, que é o dígito mais a direita em cada número, ou seja, o dígito das **UNIDADES**.

DÍGITO	CONTADOR	POSIÇÃO
0	0	0
1	0	0
2	2	0
3	3	0
4	1	0
5	0	0
6	0	0
7	1	0
8	0	0
9	0	0

Lógica do Radix Sort

123	142	87	263	233	14	132
142	132	123	263	233	14	87

DÍGITO	CONTADOR	POSIÇÃO
0	0	0
1	0	0
2	2	0
3	3	2
4	1	5
5	0	0
6	0	0
7	1	6
8	0	0
9	0	0

Lógica do Radix Sort

123	142	87	263	233	14	132
142	132	123	263	233	14	87

Agora passe para o próximo dígito, lembrando que sempre será da **direita para a esquerda**. Então, agora será o dígito que representa as **DEZENAS**.

DÍGITO	CONTADOR	POSIÇÃO
0	0	0
1	1	0
2	1	0
3	2	0
4	1	0
5	0	0
6	1	0
7	0	0
8	1	0
9	0	0

Lógica do Radix Sort

123	142	87	263	233	14	132
142	132	123	263	233	14	87
14	123	132	233	142	263	87

DÍGITO	CONTADOR	POSIÇÃO
0	0	0
1	1	0
2	1	1
3	2	2
4	1	4
5	0	0
6	1	5
7	0	0
8	1	6
9	0	0

Lógica do Radix Sort

123	142	87	263	233	14	132
142	132	123	263	233	14	87
014	123	132	233	142	263	087

Agora vá para o último dígito, sempre da direita para a esquerda, que agora será a casa das **CENTENAS**.

Nesse exemplo, há números que não têm centena, logo será atribuído o valor 0 para esses números.


DÍGITO	CONTADOR	POSIÇÃO
0	2	0
1	3	0
2	2	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0

Lógica do Radix Sort

123	142	087	263	233	014	132
142	132	123	263	233	014	087
14	123	132	233	142	263	87
14	87	123	132	142	233	263

DÍGITO	CONTADOR	POSIÇÃO
0	2	0
1	3	2
2	2	5
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0

Lógica do Radix Sort



123	142	087	263	233	014	132
142	132	123	263	233	014	087
014	123	132	233	142	263	087
14	87	123	132	142	233	263

PRONTO! Agora nosso vetor de números inteiros está devidamente ordenado em ordem crescente, utilizando a lógica por trás do algoritmo **Radix Sort**.



Complexidade Assintótica

A complexidade do algoritmo Radix Sort pode alterar em relação ao algoritmo estável que será utilizado (no caso em questão, o Counting Sort). Utilizando-se o Counting Sort como ordenador auxiliar, teremos um algoritmo de ordenação que não utiliza comparações e sim contagem.

Neste caso, o consumo de tempo é $\Theta(d(n + k))$. Se d é limitado por uma constante (ou seja, se $d == O(1)$) e $k == O(n)$, então o consumo de tempo é $O(n)$.

- d = Número de dígitos ou caracteres do maior número/string
 - k = Valor máximo de um dígito (ou bit ou caractere), ou seja, 10 valores possíveis (no caso de string, 2 ou 256)
 - n = Número de itens a serem ordenados, ou seja, tamanho do vetor.
-
- Complexidade de pior caso: $\Theta(d(n + k))$
 - Complexidade de melhor caso: $\Theta(d(n + k))$
 - Complexidade de caso médio: $\Theta(d(n + k))$



Complexidade Assintótica

Para o vetor dado [123, 142, 87, 263, 233, 14, 132], temos:

$n == 7$

$d == 3$

$k == 10$

Portanto, nesse vetor de exemplo, temos a notação:

$\Theta(3(7 + 10))$

$\Theta(51)$

Isso tanto para o **pior, médio e melhor** caso.

•



Agora, um código em C

