AULA 5: ALGORTIMOS RECURSIVOS (RADIX SORT)

Fonte: https://pixabay.com/pt/photos/c%c3%b3digo-html-digital-codifica%c3%a7%c3%a3o-1076533/

Caro(a) aluno(a), bem-vindo(a) ao nosso capítulo sobre algoritmos recursivos. Neste daremos continuidade ao entendimento e funcionamento destes algoritmos, com ênfase no algoritmo ***RadixSort***.

13.1 O que é um algoritmo RadixSort?

Um algoritmo ***radixsort***, ao contrário do ***merge-sort*** e ***quick-sort***, é um algoritmo de ordenação não comparativo. A ideia por trás disso é classificar uma lista de inteiros com base em seus dígitos individuais. Em outras palavras, a classificação é realizada do dígito menos significativo para o dígito mais significativo.

O algoritmo é denominado classificação raiz, pois especifica a raiz ***rr*** a ser usada, o que muda a forma como a classificação é realizada. A raiz, ou base, do sistema numérico é o número de dígitos que representam uma única posição no

número; uma raiz de 2 é binária (0-1), 10 é decimal (0-9), 16 é hexadecimal (0-F) e assim por diante. Uma vez que a raiz determina o número de intervalos, além do tamanho da palavra ***ww*** usado no algoritmo, modificá-la pode alterar drasticamente a forma como a classificação ocorre (Figura 5.1).

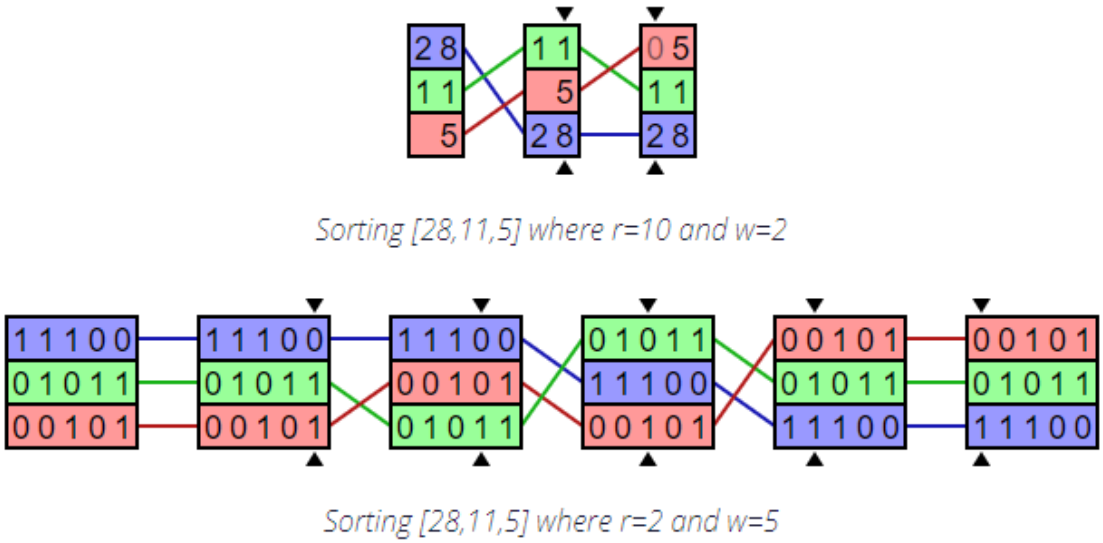


Figura 5.1: Algoritmo RadixSort

Fonte: https://www.growingwiththeweb.com/sorting/radix-sort-lsd/

Assim, de modo geral, este é um algoritmo de classificação não baseado em comparação. A palavra raiz (***radix***), por definição, refere-se à base ou ao número de dígitos únicos usados para representar números. Na classificação ***radix***, classificamos os elementos processando-os em várias passagens, dígito por dígito.

Cada passagem classifica os elementos de acordo com os dígitos em um valor de lugar particular, usando um algoritmo de classificação estável (geralmente contando a classificação como uma sub-rotina).

O número de passagens necessárias para classificar totalmente os elementos é igual ao número de valores de casas (dígitos) presentes no maior elemento entre todos os elementos de entrada a serem classificados. Essas passagens continuam a ser executadas para cada valor de posição até que a classificação seja concluída.

#ANOTE ISSO#

Outros algoritmos realizam a ordenação de dados (vetores). Não esqueçam de estudar Quick-Sort, Bubble-Sort, Merge-Sort e Shell-Sort.

#ANOTE ISSO#

Vejamos algumas aplicações deste algoritmo:

- São aplicados a dados que podem ser classificados lexicograficamente, como palavras e inteiros. Ele também é usado para classificar ***strings*** de maneira estável.
- É uma boa opção quando o algoritmo roda em máquinas paralelas, tornando a ordenação mais rápida. Para usar a paralelização, dividimos a entrada em vários depósitos, o que nos permite classificar os depósitos em paralelo, pois são independentes uns dos outros.
- É usado para construir uma matriz de sufixo. (Uma matriz que contém todos os sufixos possíveis de uma string em ordem classificada é chamada de matriz de sufixo.

Há duas formas de resolver esse tipo de algoritmo. O primeiro pelo ***Radix Sort LSD (Least Significant Digit)*** e a segunda forma pelo ***Radix Sort MSD (Most Significant Digit).***

No primeiro, ***Radix Sort LSD (Least Significant Digit)*** (dígito menos significativo primeiro) classifica de forma estável a matriz com base em seu dígito menos significativo, e, em seguida, em seu segundo dígito menos significativo e assim por diante até seu dígito mais significativo.

O segundo ***Radix Sort MSD (Most Significant Digit)(***digito mais significativo primeiro) classifica primeiro de forma estável a matriz com base em seu digito mais significativo, a seguir em seu segundo digito mais significativo e assim por diante até seu digito menos significativo.

Vamos identificar como acontece o estruturamento de ambos e a lógica envolvida neste tipo de algortimo.

13.2 Estruturação de um algoritmo ***RadixSort.***

Vamos ver como este algoritmo se comporta na prática. A ideia deste algoritmo é usar uma abordagem onde há um processamento das chaves por partes.

Suponha que necessitamos ordenar o determinado vetor (lista) que possui 8 registros (n). Estes registros possuem chaves de 3 dígitos (d) e estão na base decimal k=10.

[506, 818, 088, 938, 457, 412, 002, 289]

Vamos iniciar pela primeira variante desse algoritmo: ***RadixSort LSD***. Relembrando, nesta variante há uma classificação inicial pelo digito menos significante.

1º Passo: Devemos inserir estes números em tabela que facilita a visualização e o processo de ordenação.

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

Tabela 5.1: Algoritmo RadixSort

Fonte: Elaborado pelo autor, 2021.

2º Passo: Devemos iniciar identificando o digito menos significativo dos números que estão no nosso vetor (em vermelho)

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

Tabela 5.2: Algoritmo RadixSort

Fonte: Elaborado pelo autor, 2021.

3º Passo: Realizamos a ordenação de todos os elementos que estão em vermelho. Verifica-se que a ordenação desses números serão: 2, 2, 6, 7, 8, 8, 8 e 9, cujo seus números respectivos são: 412 (para o número 2), 002 (para o número 2), 506 (para o número 6), 457 (para o número 7), 818 (para o número 8), 088 (para o número 8), 938 (para o número 8) e 289 (para o número 9). Assim, temos:

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

Tabela 5.3: Algoritmo RadixSort

Fonte: Elaborado pelo autor, 2021.

4° Passo: Agora devemos identificar o próximo digito menos significativo. Assim, temos:

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

Tabela 5.4: Algoritmo RadixSort

Fonte: Elaborado pelo autor, 2021.

5º Passo: Realizamos a ordenação de todos os elementos que estão em azul. Verifica-se que a ordenação desses números serão: 0, 0, 1, 1, 3, 5, 8 e 8, cujo seus números respectivos são: 002 (para o número 0), 506 (para o número 0), 412 (para o número 1), 818 (para o número 1), 938 (para o número 3), 457 (para o número 5), 088 (para o número 8) e 289 (para o número 8). Assim, temos:

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

Tabela 5.5: Algoritmo RadixSort

Fonte: Elaborado pelo autor, 2021.

6° Passo: Agora devemos identificar o próximo digito menos significativo, que neste caso será o digito mais significativo. Assim, temos:

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

Tabela 5.6: Algoritmo RadixSort

Fonte: Elaborado pelo autor, 2021.

7º Passo: Realizamos a ordenação de todos os elementos que estão em verde. Verifica-se que a ordenação desses números serão: 0, 0, 2, 4, 4, 5, 8 e 9, cujo seus números respectivos são: 002 (para o número 0), 088 (para o número 0), 289 (para o número 2), 412 (para o número 4), 457 (para o número 4), 506 (para o número 5), 818 (para o número 8) e 938 (para o número 8). Assim, temos:

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

Tabela 5.7: Algoritmo RadixSort

Fonte: Elaborado pelo autor, 2021.

Por fim temos o ordenamento final da nossa lista:

[002, 088, 289, 412, 457, 506, 818, 938]

#ANOTE ISSO#

O RADIX Sort LSD necessita, obrigatoriamente, que as chaves sempre sejam do mesmo tamanho para que ele possa realizar o ordenamento. No nosso exemplo todas apresentam 3 chaves (dígitos)

#ANOTE ISSO#

Agora vamos entender a segunda variante desse algoritmo: ***RadixSort MSD***. Relembrando, nesta variante há uma classificação inicial pelo digito mais significante. A ideia nesta variante é a separação dos números (elementos) com o mesmo digito. Vamos ver como acontece na prática.

Suponha que necessitamos ordenar um vetor (lista) que possui 8 registros (n). Estes registros possuem chaves de 3 dígitos (d) e estão na base decimal k=10. Como lista, temos:

[089, 818, 038, 457, 039, 452, 032, 096]

1º Passo: Devemos inserir estes números em tabela que facilita a visualização e o processo de ordenação.

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

Tabela 5.8: Algoritmo RadixSort

Fonte: Elaborado pelo autor, 2021.

2º Passo: Devemos iniciar identificando o digito mais significativo dos números que estão no nosso vetor (em vermelho):

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

Tabela 5.9: Algoritmo RadixSort

Fonte: Elaborado pelo autor, 2021.

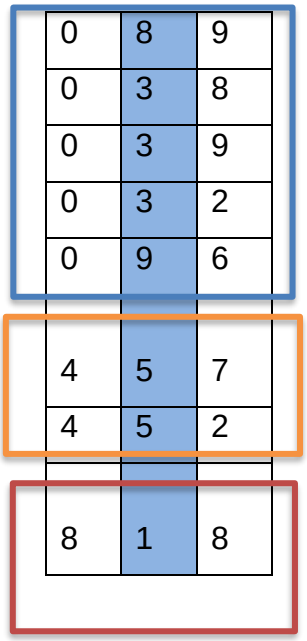
3º Passo: Faremos uma divisão por partes. Iremos ordenar conforme o digito mais significativo. Como ordem temos: 0, 0, 0, 0, 0, 4, 4, 8. Percebam que temos três blocos, o bloco dos números 0, o bloco dos números 4 e o bloco do número 8.

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

Tabela 5.10: Algoritmo RadixSort

Fonte: Elaborado pelo autor, 2021.

4º Passo: Devemos agora verificar o segundo digito mais significativo do nosso vetor.



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

Tabela 5.11: Algoritmo RadixSort

Fonte: Elaborado pelo autor, 2021.

5º Passo: Faremos uma divisão por partes. Iremos ordenar conforme o segundo digito mais significativo, mas agora de cada bloco. No primeiro bloco temos 3, 3, 3, 8 e 9. No segundo bloco temos 5, 5. E no terceiro bloco 1. Logo teremos:

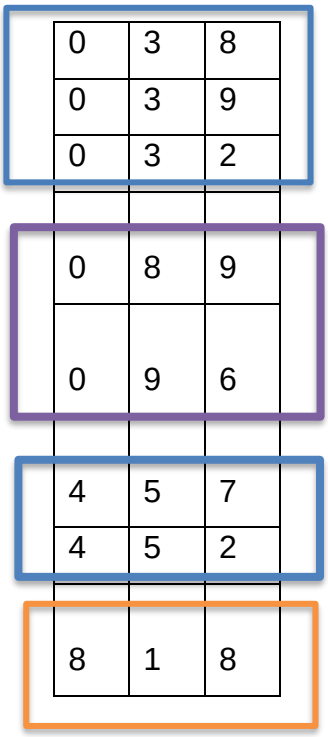
0	3	8
0	3	9
0	3	2
0	8	9
0	9	6
4	5	7
4	5	2

8	1	8

Tabela 5.12: Algoritmo RadixSort

Fonte: Elaborado pelo autor, 2021.

6º Passo: Faremos novamente a divisão por blocos considerando o segundo digito mais significativo. Teremos blocos do número 3, do número 8, do número 9, do número 5 e do número 1:



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

Tabela 5.13: Algoritmo RadixSort

Fonte: Elaborado pelo autor, 2021.

7º Passo: Devemos agora verificar o terceiro digito menos significativo do nosso vetor (ou o mais insignificativo):

0	3	8
0	3	9
0	3	2
0	8	9

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

Tabela 5.14: Algoritmo RadixSort

Fonte: Elaborado pelo autor, 2021.

Onde teremos:

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

Tabela 5.15: Algoritmo RadixSort

Fonte: Elaborado pelo autor, 2021.

Por fim temos o ordenamento final da nossa lista:

[032, 038, 039, 089, 096, 452, 457, 818]

Então aluno (a), neste capítulo entendemos a lógica envolvida em um algoritmo ***RadixSort***. No próximo capítulo estudaremos sobre algoritmos de pesquisa, outra parte bem importante em estruturas de dados e arquivos.

#ANOTE ISSO#

No capítulo 10 e 11 teremos estes e outros algoritmos estruturados em Java e Python.

#ANOTE ISSO#