

RadixSort

Ciência da Computação
Laboratório de Ordenação e Pesquisa
Prof. M.Sc. Elias Gonçalves

RadixSort

→ **Esse método de ordenação processa as chaves por partes.**

→ **Há duas classificações:**

LSD - Least significant digit (Dígito menos significativo).

MSD - Most significant digit (Dígito mais significativo).

→ **A ideia é quebrar uma chave em vários pedaços. Sendo que toda chave pode ser representada na forma de dígitos de um número em uma base. Ex:**

Base 10:	320	3	2	0		
Base 2:	1010	1	0	1	0	
Base 256:	"Teste"	T	e	s	t	e

→ **Seu tempo de execução assintótico é $O(nk)$, onde n é o número de elementos e k o tamanho médio da chave;**

RadixSort

- Começa do dígito menos significativo;
- Chaves curtas vem antes de chaves longas;
- Chaves de mesmo tamanho são ordenadas lexicograficamente;
- As chaves podem ser caracteres ou números;

→ **Exemplo com string e inteiros de tamanho variável:**

b c d e ba ordenado ficaria → **b ba c d e**

1 7 9 8 10 ordenado ficaria → **1 10 7 8 9**

→ **Como ordenar o vetor abaixo com o RadixSort LSD?**

153	30	92	25	2	98	13	vetor v
0	1	2	3	4	5	6	índice i

RadixSort

- Trata-se de um vetor de inteiros na base 10 (decimal).
- Encontre o maior número do vetor.

153	30	92	25	2	98	13	vetor v
0	1	2	3	4	5	6	índice i

- Maior = 153
- Agora sabemos que o maior número tem 3 dígitos.
- Todos os demais números devem ser tratados como se tivessem 3 dígitos, sendo que para isso pode-se adicionar 0 à esquerda dos números com menos de 3 dígitos. Ex: 30 → 030, 2 → 002, ...

RadixSort

→ Prepare baldes de 0 a 9 (base 10) para colocar temporariamente os números do vetor de acordo com o seu dígito menos significativo.

153	30	92	25	2	98	13	vetor v
0	1	2	3	4	5	6	índice i

Baldes

										[2]
										[1]
										[0]
0	1	2	3	4	5	6	7	8	9	Índices
Números da base 10										

RadixSort

→ Distribua os números nos baldes conforme o **dígito menos significativo**. Coloque-os nos baldes seguindo os índices dos baldes, isto é, começando no índice 0.

153	30	92	25	2	98	13	vetor v
0	1	2	3	4	5	6	índice i

Baldes

										[2]
		002	013							[1]
030		092	153		025			098		[0]
0	1	2	3	4	5	6	7	8	9	
Números da base 10										Índices

RadixSort

- Retire os números dos baldes começando no balde 0.
- Siga a ordem dos índices para cada balde: balde 0, índice 0, balde 0, índice 1... balde 9, índice 0...
- Coloque os valores de volta no vetor.

30	92	2	153	13	25	98	vetor v índice i
0	1	2	3	4	5	6	

Baldes

										índices [2] [1] [0]
		002	013							
030		092	153		025			098		
0	1	2	3	4	5	6	7	8	9	
Números da base 10										

RadixSort

→ Distribua novamente os números nos baldes conforme o **segundo dígito menos significativo**. Coloque-os nos baldes seguindo os índices dos baldes, isto é, começando no índice 0.

30	92	02	153	13	25	98	vetor v
0	1	2	3	4	5	6	índice i

Baldes

										[2]
									098	[1]
002	013	025	030		153				092	[0]
0	1	2	3	4	5	6	7	8	9	
Números da base 10										Índices

RadixSort

- Retire os números dos baldes começando no balde 0.
- Siga a ordem dos índices para cada balde: balde 0, índice 0, balde 0, índice 1... balde 9, índice 0...
- Coloque os valores de volta no vetor.

2	13	25	30	153	92	98	vetor v índice i
0	1	2	3	4	5	6	

Baldes

										[2]
									098	[1]
002	013	025	030		153				092	[0]
0	1	2	3	4	5	6	7	8	9	
Números da base 10										Índices

RadixSort

→ Distribua novamente os números nos baldes conforme o **terceiro dígito menos significativo**. Coloque-os nos baldes seguindo os índices dos baldes, isto é, começando no índice 0

002	013	025	030	153	092	098	vetor v
0	1	2	3	4	5	6	índice i

Baldes	098										[5]	Índices
	092										[4]	
	030										[3]	
	025										[2]	
	013										[1]	
	002	153									[0]	
	0	1	2	3	4	5	6	7	8	9		
Números da base 10												

RadixSort

- Retire os números dos baldes começando no balde 0.
- Siga a ordem dos índices para cada balde: balde 0, índice 0, balde 0, índice 1... balde 9, índice 0...
- Coloque os valores de volta no vetor.

2	13	25	30	92	98	153	vetor v
0	1	2	3	4	5	6	índice i

Baldes	098										[5]	Índices
	092										[4]	
	030										[3]	
	025										[2]	
	013										[1]	
	002	153									[0]	
	0	1	2	3	4	5	6	7	8	9		
Números da base 10												

RadixSort

Vetor no início (desordenado)

153	30	92	25	2	98	13	vetor v
0	1	2	3	4	5	6	índice i

Vetor no final (ordenado)

2	13	25	30	92	98	153	vetor v
0	1	2	3	4	5	6	índice i

RadixSort

```
void radixsort( int vetor[], int tamanho) {
    int *b, i, lsd;
    int maior = vetor[0], exp = 1; // exp: 1-> unid., 10-> dez., 100-> cent., 1000-> mil...
    b = ( int* )calloc( tamanho, sizeof(int) );

    // Encontra o maior número do vetor
    for ( i = 0; i < tamanho; i++ )
        if ( vetor[i] > maior )
            maior = vetor[i];

    while ( maior/exp > 0 ) {
        int balde[10] = { 0 }; // inicializa as posições de 0 a 9 com 0

        // Toma o dígito menos significativo e conta
        // o numero de vezes que ele se repete
        for ( i=0; i<tamanho; i++){
            lsd = (vetor[i]/exp )%10;
            balde[ lsd ]++;
            // printf("%d elementos com lsd %d\n", balde[ ( vetor[i]/exp )%10 ], lsd );
        }
    }
}
```

RadixSort

```
// Calcula a posição de cada elemento no vetor
for ( i=1; i<10; i++ ){
    //printf("balde[%d] = %d + ", i, balde[i]);
    balde[i] += balde[i-1];
    //printf("%d\n", balde[i-1]);
}

// Retira os números do vetor e coloca temporariamente em um ponteiro
for ( i = tamanho-1; i>=0; i-- ){
    b[ --balde[(vetor[i]/exp)%10] ] = vetor[i];
}

// Volta os números par o vetor original
for ( i=0; i<tamanho; i++ )
    vetor[i] = b[i];

// Atualiza o expoente (passa para o próximo lsd)
exp *= 10;
}
free(b);
```