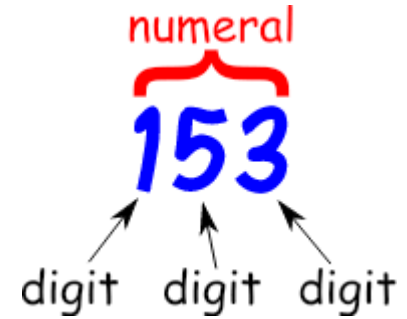


# Pesquisa e Ordenação de Dados

Unidade 2.8:

## Radix Sort

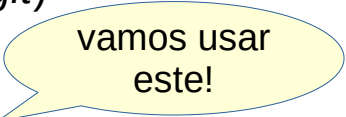


# Radix Sort

- Criado originalmente para ordenação de cartões perfurados
- O algoritmo computacional foi proposto por Harold H. Seward em 1954 no MIT
- Baseado na ordenação de chaves sem comparações
  - A chave é decomposta e tratada por partes (dígitos)
  - Utiliza a lógica do **Counting Sort** para contagem das ocorrências dos dígitos

# Radix Sort

- Supondo que as chaves que desejamos ordenar possuem **d** **dígitos**
- Ideia geral:
  - Ordenar pelos dígitos, um de cada vez:
    - do dígito mais significativo para o menos (MSD – *most significant digit*)
      - adequado para chaves do tipo string
    - ou do menos significativo para o mais (LSD – *least significant digit*)
      - adequado para chaves numéricas
- Deste modo, apenas **d** passadas pela lista são necessárias para realizar a ordenação



vamos usar este!

# Radix Sort

Seleciona o maior e verifica o número de dígitos

30	21	43	3	9	<b>82</b>	15
0	1	2	3	4	5	6

Faremos 2 passadas sobre os dados:  
para o primeiro dígito (mais à direita),  
depois para o segundo dígito

# Radix Sort

## Conta as ocorrências – 1º dígito

30	21	43	3	9	82	15
0	1	2	3	4	5	6

Neste exemplo os valores são decimais,  
portanto, o vetor de contagem terá 10  
posições (0 a 9)

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

# Radix Sort

## Conta as ocorrências – 1º dígito

atual

30	21	43	3	9	82	15
0	1	2	3	4	5	6

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

# Radix Sort

## Conta as ocorrências – 1º dígito

atual						
30	21	43	3	9	82	15
0	1	2	3	4	5	6

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

# Radix Sort

## Conta as ocorrências – 1º dígito

atual						
30	21	43	3	9	82	15
0	1	2	3	4	5	6

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



# Radix Sort

## Conta as ocorrências – 1º dígito

atual						
30	21	43	3	9	82	15
0	1	2	3	4	5	6

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

# Radix Sort

## Conta as ocorrências – 1º dígito

atual						
30	21	43	3	9	82	15
0	1	2	3	4	5	6

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

# Radix Sort

## Conta as ocorrências – 1º dígito

atual						
30	21	43	3	9	82	15
0	1	2	3	4	5	6

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

# Radix Sort

## Conta as ocorrências – 1º dígito

							atual
30	21	43	3	9	82	15	
0	1	2	3	4	5	6	

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

# Radix Sort

## Soma acumulada – 1º dígito

30	21	43	3	9	82	15
0	1	2	3	4	5	6

dig	cont	soma
0	1	1
1	1	2
2	1	3
3	2	5
4	0	5
5	1	6
6	0	6
7	0	6
8	0	6
9	1	7

# Radix Sort

## Reposiciona – 1º dígito

30	21	43	3	9	82	15
0	1	2	3	4	5	6

atual

					15	
0	1	2	3	4	5	6

dig	cont	soma
0	1	1
1	1	2
2	1	3
3	2	5
4	0	5
5	1	<del>6</del> 5
6	0	6
7	0	6
8	0	6
9	1	7

# Radix Sort

## Reposiciona – 1º dígito

atual

30	21	43	3	9	82	15
0	1	2	3	4	5	6

		82			15	
0	1	2	3	4	5	6

dig	cont	soma
0	1	1
1	1	2
2	1	3 2
3	2	5
4	0	5
5	1	5
6	0	6
7	0	6
8	0	6
9	1	7

# Radix Sort

## Reposiciona – 1º dígito

atual						
30	21	43	3	9	82	15
0	1	2	3	4	5	6

		82			15	9
0	1	2	3	4	5	6

dig	cont	soma
0	1	1
1	1	2
2	1	2
3	2	5
4	0	5
5	1	5
6	0	6
7	0	6
8	0	6
9	1	<del>7</del> 6



# Radix Sort

## Reposiciona – 1º dígito

atual

30	21	43	3	9	82	15
0	1	2	3	4	5	6

		82		3	15	9
0	1	2	3	4	5	6

dig	cont	soma
0	1	1
1	1	2
2	1	2
3	2	<del>5</del> 4
4	0	5
5	1	5
6	0	6
7	0	6
8	0	6
9	1	6

# Radix Sort

## Reposiciona – 1º dígito

atual

30	21	43	3	9	82	15
0	1	2	3	4	5	6

		82	43	3	15	9
0	1	2	3	4	5	6

dig	cont	soma
0	1	1
1	1	2
2	1	2
3	2	4 3
4	0	5
5	1	5
6	0	6
7	0	6
8	0	6
9	1	6

# Radix Sort

## Reposiciona – 1º dígito

atual

30	21	43	3	9	82	15
0	1	2	3	4	5	6

	21	82	43	3	15	9
0	1	2	3	4	5	6

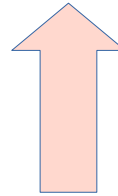
dig	cont	soma
0	1	1
1	1	<del>2</del> 1
2	1	2
3	2	3
4	0	5
5	1	5
6	0	6
7	0	6
8	0	6
9	1	6

# Radix Sort

## Reposiciona – 1º dígito

atual

30	21	43	3	9	82	15
0	1	2	3	4	5	6



dig	cont	soma
0	1	<del>1</del> 0
1	1	1
2	1	2
3	2	3
4	0	5
5	1	5
6	0	6
7	0	6
8	0	6
9	1	6

Fim da ordenação  
pelo 1º dígito

Copia do auxiliar para  
o vetor original

30	21	82	43	3	15	9
0	1	2	3	4	5	6

# Radix Sort

## Conta as ocorrências – 2º dígito

30	21	82	43	3	15	9
0	1	2	3	4	5	6

Recomeça o processo, agora  
para o segundo dígito

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

# Radix Sort

## Conta as ocorrências – 2º dígito

atual

30	21	82	43	3	15	9
0	1	2	3	4	5	6

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

# Radix Sort

## Conta as ocorrências – 2º dígito

atual						
30	21	82	43	3	15	9
0	1	2	3	4	5	6

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

# Radix Sort

## Conta as ocorrências – 2º dígito

atual						
30	21	82	43	3	15	9
0	1	2	3	4	5	6

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



# Radix Sort

## Conta as ocorrências – 2º dígito

atual

30	21	82	43	3	15	9
0	1	2	3	4	5	6

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

# Radix Sort

## Conta as ocorrências – 2º dígito

atual						
30	21	82	43	3	15	9
0	1	2	3	4	5	6

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

# Radix Sort

## Conta as ocorrências – 2º dígito

atual						
30	21	82	43	3	15	9
0	1	2	3	4	5	6

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

# Radix Sort

## Conta as ocorrências – 2º dígito

atual						
30	21	82	43	3	15	9
0	1	2	3	4	5	6

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

# Radix Sort

## Soma acumulada – 2º dígito

30	21	43	3	9	82	15
0	1	2	3	4	5	6

dig	cont	soma
0	2	2
1	1	3
2	1	4
3	1	5
4	1	6
5	0	6
6	0	6
7	0	6
8	1	7
9	0	7

# Radix Sort

## Reposiciona – 2º dígito

atual						
30	21	43	3	9	82	15
0	1	2	3	4	5	6

		15				
0	1	2	3	4	5	6

dig	cont	soma
0	2	2
1	1	<del>3</del> 2
2	1	4
3	1	5
4	1	6
5	0	6
6	0	6
7	0	6
8	1	7
9	0	7

# Radix Sort

## Reposiciona – 2º dígito

atual

30	21	43	3	9	82	15
0	1	2	3	4	5	6

		15				82
0	1	2	3	4	5	6

dig	cont	soma
0	2	2
1	1	2
2	1	4
3	1	5
4	1	6
5	0	6
6	0	6
7	0	6
8	1	<del>7</del> 6
9	0	7

# Radix Sort

## Reposiciona – 2º dígito

atual

30	21	43	3	9	82	15
0	1	2	3	4	5	6

	9	15				82
0	1	2	3	4	5	6

dig	cont	soma
0	2	<del>2</del> 1
1	1	2
2	1	4
3	1	5
4	1	6
5	0	6
6	0	6
7	0	6
8	1	6
9	0	7



# Radix Sort

## Reposiciona – 2º dígito

atual

30	21	43	3	9	82	15
0	1	2	3	4	5	6

3	9	15				82
0	1	2	3	4	5	6

dig	cont	soma
0	2	<del>1</del> 0
1	1	2
2	1	4
3	1	5
4	1	6
5	0	6
6	0	6
7	0	6
8	1	6
9	0	7

# Radix Sort

## Reposiciona – 2º dígito

atual

30	21	43	3	9	82	15
0	1	2	3	4	5	6

3	9	15			43	82
0	1	2	3	4	5	6

dig	cont	soma
0	2	0
1	1	2
2	1	4
3	1	5
4	1	<del>6</del> 5
5	0	6
6	0	6
7	0	6
8	1	6
9	0	7

# Radix Sort

## Reposiciona – 2º dígito

atual

30	21	43	3	9	82	15
0	1	2	3	4	5	6

3	9	15	21		43	82
0	1	2	3	4	5	6

dig	cont	soma
0	2	0
1	1	2
2	1	4 3
3	1	5
4	1	5
5	0	6
6	0	6
7	0	6
8	1	6
9	0	7

# Radix Sort

## Reposiciona – 2º dígito

atual

30	21	43	3	9	82	15
0	1	2	3	4	5	6

3	9	15	21	30	43	82
0	1	2	3	4	5	6

dig	cont	soma
0	2	0
1	1	2
2	1	3
3	1	5 4
4	1	5
5	0	6
6	0	6
7	0	6
8	1	6
9	0	7

# Radix Sort

## Reposiciona – 2º dígito

30	21	43	3	9	82	15
0	1	2	3	4	5	6

Fim da ordenação  
pelo 2º dígito

Copia do auxiliar para  
o vetor original

03	09	15	21	30	43	82
0	1	2	3	4	5	6

dig	cont	soma
-----	------	------



8	1	6
9	0	7

# Radix Sort

## Implementação

- O algoritmo será implementado em duas partes:
  - **radixSort**
    - Função principal que será chamada pelos demais programas
    - Responsável por identificar o maior número e contar o número de dígitos.
    - Realiza a chamada da função de contagem que ordena baseada nos dígitos
  - **countingSort**
    - Função auxiliar responsável por ordenar os valores de um dígito específico
      - Construída geralmente na base de 10, mas pode ser feita com qualquer base numérica
    - Itera sobre os dígitos fazendo a contagem
    - Após a contagem reescreve o vetor com base na nova ordem obtida pela contagem

# Radix Sort

## Pseudocódigo

função radixSort(A[], n)

Início

max = buscaMax(A, n)

Busca o maior elemento do vetor

para  $pos = 1, \max/pos > 0, pos *= 10$  faça

countingSort(A, n, pos)

Chamada do counting com o dígito

fim

# Radix Sort

## Pseudocódigo

função **countingSort**(A[], n, pos)

Início

declara vetor aux com n posições

count[10]  $\leftarrow$  0

para  $i = 0, i < n$  faça // conta as ocorrências

    digito  $\leftarrow$  (A[i] / pos) % 10 // isola o dig. atual

    count[digito]++

fim para

para  $i = 1, i < 10$  faça

    count[i]  $\leftarrow$  count[i] + count[i-1]; // soma ac

fim para

para  $i = n-1, i \geq 0$  faça

    // constroi o vet de saída

    digito  $\leftarrow$  (A[i] / pos) % 10

    count[digito]--

    aux[count[digito]]  $\leftarrow$  A[i]

fim para

para  $i = 0, i < n$  faça

    A[i]  $\leftarrow$  aux[i] // copia os valores

fim para

fim



# Radix Sort

## Análise

- Complexidade  $O(d * (n + k))$ 
  - Independente da ordenação da entrada
  - se  $d$  é constante e  $k$  é no máximo igual a  $n$ , então o radix sort é  $O(n)$
  - O número de dígitos pode ser visto como  $\log n$ , o que o faz comparável ao quickSort em certas situações
- Estável