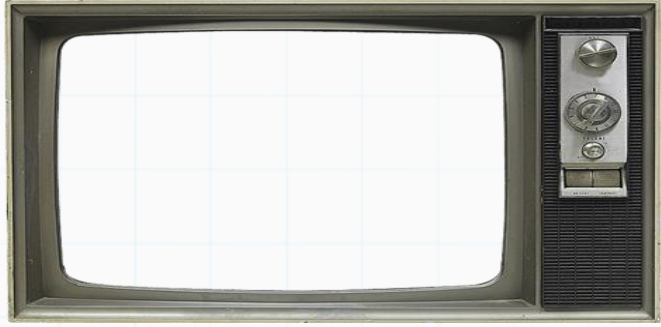
Programação Estruturada

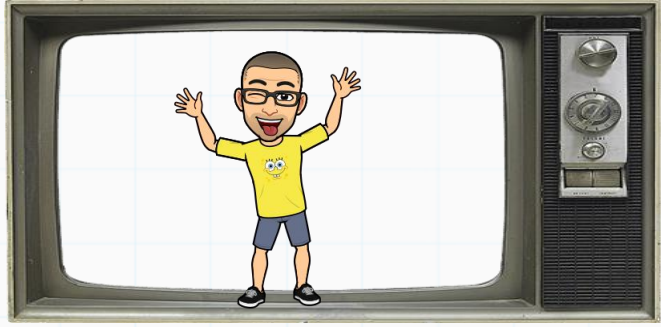
Professor : Yuri Frota

yuri@ic.uff.br



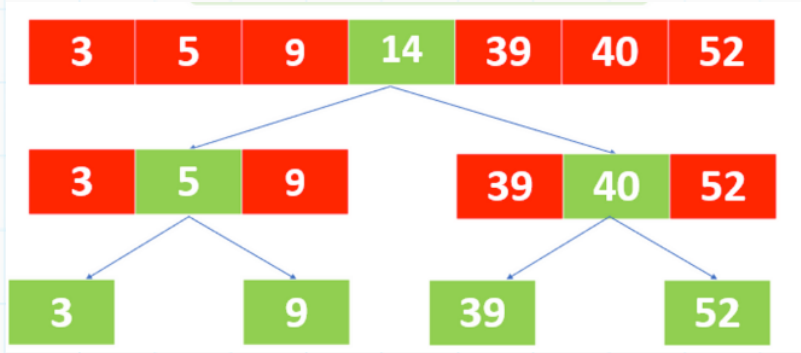
cuidado, vamos usar apenas os comando e estruturas do C, nada de C++

Buscas e Ordenação - LAB



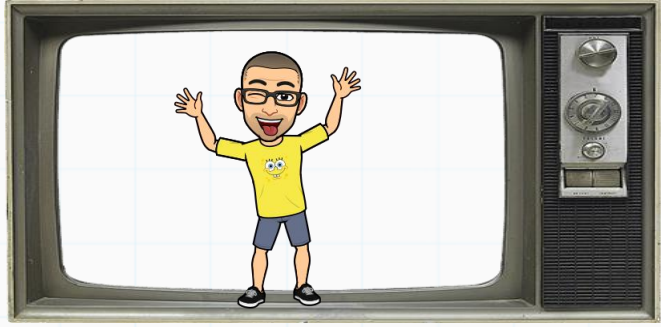
Exercício 1) Refaça a busca binária assumindo que o vetor possui elementos que podem aparecer repetidos. Neste caso, você deve retornar imprimir todas as posições onde o elemento foi encontrada.

```c
void busca_binaria(int* v, int tam, int el)
{
    int ini, fim;

    ini = 0; fim = tam-1;
    while (ini <= fim)
    {
        int meio = ini + (fim - ini) / 2;

        if (v[meio] == el)
        {
            printf("achou %d na posicao %d\n", el, meio);
            return;
        }

        if (v[meio] < el)
          ini = meio + 1;

        else
          fim = meio - 1;
    }

    printf("Nao achou\n");
    return;
}
```



Veja o exemplo de execução:

Buscas e Ordenação - LAB



```
vetor (tam=8) = 1, 1, 5, 5, 5, 8, 8, 8,

buscar numero: 5
achou 5 na posicao 3, 2, 4,
```

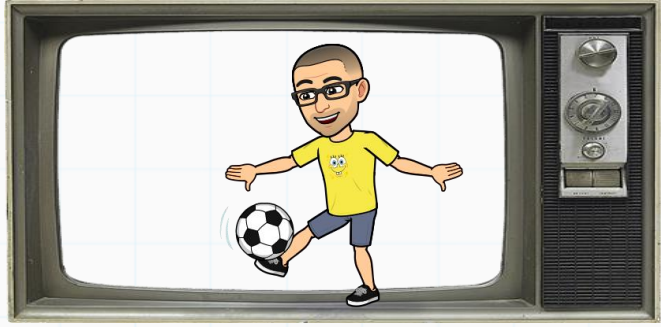
1	1	5	5	5	8	8	8

```
vetor (tam=8) = 1, 1, 5, 5, 5, 8, 8, 8,

buscar numero: 1
achou 5 na posicao 0, 1,
```

```
vetor (tam=8) = 1, 1, 5, 5, 5, 8, 8, 8,

buscar numero: 8
achou 5 na posicao 5, 6, 7,
```

```
vetor (tam=8) = 1, 1, 1, 1, 1, 1, 1, 1,

buscar numero: 1
achou 1 na posicao 0, 1, 2, 3, 4, 5, 6, 7,
```

Buscas e Ordenação - LAB

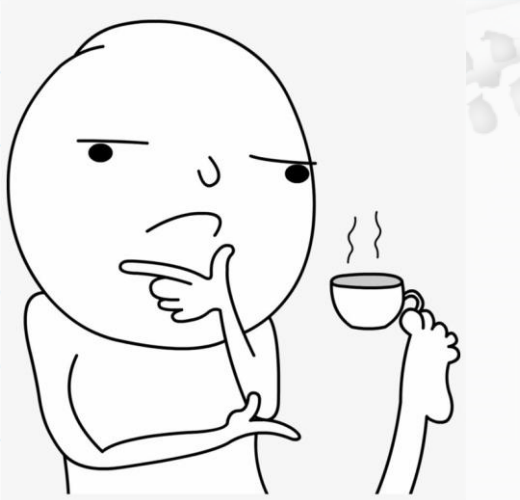


Exercício 2) Faça um programa que receba um vetor de inteiros ordenado (pode acreditar) e imprimir. O programa deve perguntar se o usuário quer alterar alguma posição, se sim, então alterar, reordenar e imprimir o vetor. **A reordenação tem que ser feita percorrendo os elementos do vetor no máximo apenas um vez (ou empurrando o elemento alterado para frente ou para trás)**.

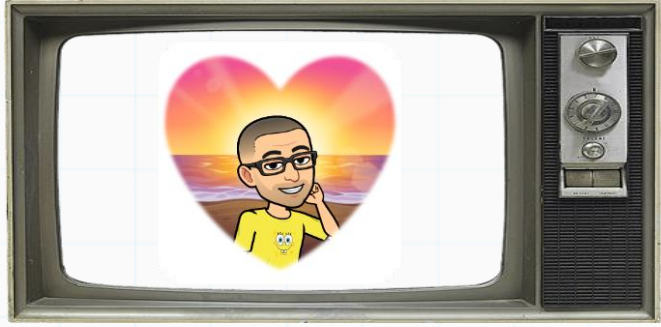
8	23	28	47	58	69	72	83	90	94

Veja exemplo de execução:

```
vetor (tam=10) = 8, 23, 28, 47, 58, 69, 72, 83, 90, 94,

Quer alterar ? (0-sim 1-nao): 0
posicao: 2
valor: 280
vetor (tam=10) = 8, 23, 47, 58, 69, 72, 83, 90, 94, 280,

Quer alterar ? (0-sim 1-nao): 0
posicao: 9
valor: -280
vetor (tam=10) = -280, 8, 23, 47, 58, 69, 72, 83, 90, 94,

Quer alterar ? (0-sim 1-nao): 0
posicao: 0
valor: 28000
vetor (tam=10) = 8, 23, 47, 58, 69, 72, 83, 90, 94, 28000,

Quer alterar ? (0-sim 1-nao): 1
```



Buscas e Ordenação - LAB

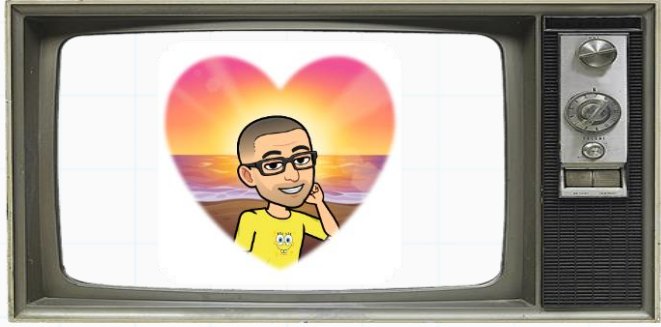


Exercício 3) CountSort: Como ordenar o vetor abaixo ?

6	3	9	10	15	6	8	12	3	6

vetor com elementos positivos

Buscas e Ordenação - LAB



Exercício 3) CountSort: Como ordenar o vetor abaixo ?

6	3	9	10	15	6	8	12	3	6

vetor com elementos positivos

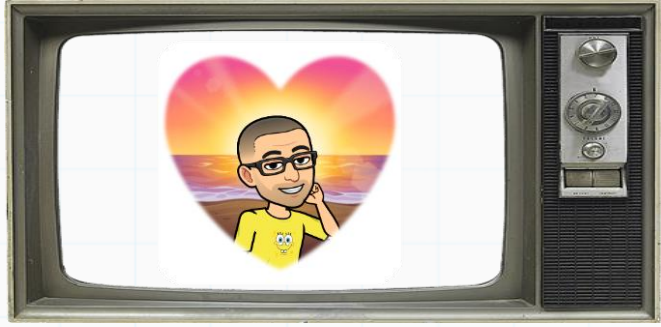
V=

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Primeiro descobrimos o maior elemento (MAX) e alocamos um vetor de inteiros de tamanho (MAX+1)

No exemplo MAX=15

Buscas e Ordenação - LAB



Exercício 3) CountSort: Como ordenar o vetor abaixo ?

6	3	9	10	15	6	8	12	3	6

vetor com elementos positivos

V=

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Primeiro descobrimos o maior elemento (MAX) e alocamos um vetor de inteiros de tamanho (MAX+1)

No exemplo MAX=15

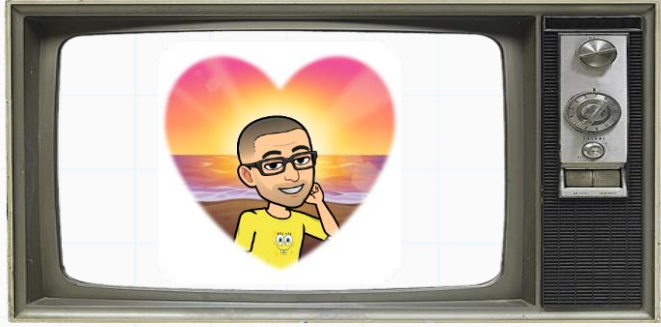
V=

0	0	0	2	0	0	3	0	1	1	1	0	1	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Devemos percorrer o vetor a ser ordenado e sempre que aparecer um valor, ele deve ser contado na posição respectiva

No exemplo, o 3 aparece 2 vezes, então v[3]=2

Buscas e Ordenação - LAB



Exercício 3) CountSort: Como ordenar o vetor abaixo ?

6	3	9	10	15	6	8	12	3	6

vetor com elementos positivos

V=

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Primeiro descobrimos o maior elemento (MAX) e alocamos um vetor de inteiros de tamanho (MAX+1)

No exemplo MAX=15

V=

0	0	0	2	0	0	3	0	1	1	1	0	1	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

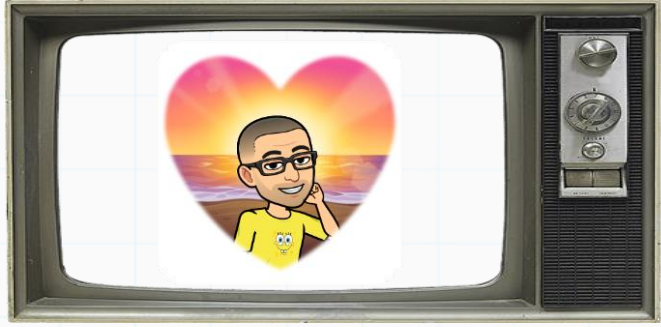
Devemos percorrer o vetor a ser ordenado e sempre que aparecer um valor, ele deve ser contado na posição respectiva

No exemplo, o 3 aparece 2 vezes, então v[3]=2

3	3								

Devemos então percorrer o vetor contador V, vendo as posições diferentes de 0 e preenchendo o vetor original. Veja que o elemento 3 é o primeiro diferente de zero, e aparece 2 vezes.

Buscas e Ordenação - LAB



Exercício 3) CountSort: Como ordenar o vetor abaixo ?

6	3	9	10	15	6	8	12	3	6

vetor com elementos positivos

V=

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Primeiro descobrimos o maior elemento (MAX) e alocamos um vetor de inteiros de tamanho (MAX+1)

No exemplo MAX=15

V=

0	0	0	2	0	0	3	0	1	1	1	0	1	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

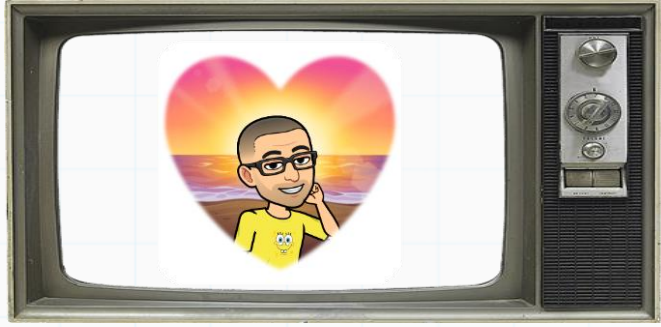
Devemos percorrer o vetor a ser ordenado e sempre que aparecer um valor, ele deve ser contado na posição respectiva

No exemplo, o 3 aparece 2 vezes, então v[3]=2

3	3	6	6	6					

Devemos então percorrer o vetor contador V, vendo as posições diferentes de 0 e preenchendo o vetor original. Veja que o elemento 3 é o primeiro diferente de zero, e aparece 2 vezes.
- Depois o 6 que aparece 3 vezes

Buscas e Ordenação - LAB



Exercício 3) CountSort: Como ordenar o vetor abaixo ?

6	3	9	10	15	6	8	12	3	6

vetor com elementos positivos

V=

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Primeiro descobrimos o maior elemento (MAX) e alocamos um vetor de inteiros de tamanho (MAX+1)

No exemplo MAX=15

V=

0	0	0	2	0	0	3	0	1	1	1	0	1	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Devemos percorrer o vetor a ser ordenado e sempre que aparecer um valor, ele deve ser contado na posição respectiva

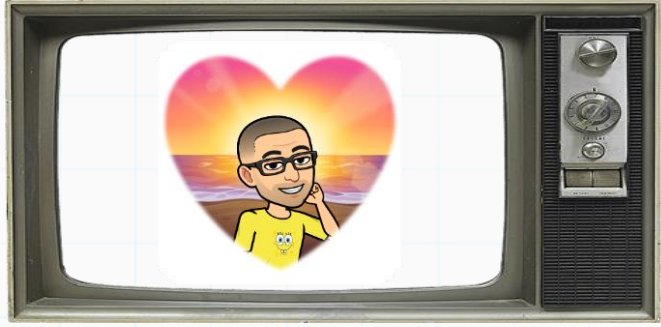
No exemplo, o 3 aparece 2 vezes, então v[3]=2

3	3	6	6	6	8				

Devemos então percorrer o vetor contador V, vendo as posições diferentes de 0 e preenchendo o vetor original. Veja que o elemento 3 é o primeiro diferente de zero, e aparece 2 vezes.

- Depois o 6 que aparece 3 vezes
- O 8 aparece 1 vez

Buscas e Ordenação - LAB



Exercício 3) CountSort: Como ordenar o vetor abaixo ?

6	3	9	10	15	6	8	12	3	6

vetor com elementos positivos

Primeiro descobrimos o maior elemento (MAX) e alocamos um vetor de inteiros de tamanho (MAX+1)

No exemplo MAX=15

V=

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Devemos percorrer o vetor a ser ordenado e sempre que aparecer um valor, ele deve ser contado na posição respectiva

V=

0	0	0	2	0	0	3	0	1	1	1	0	1	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

No exemplo, o 3 aparece 2 vezes, então v[3]=2

3	3	6	6	6	8	9	10	12	15

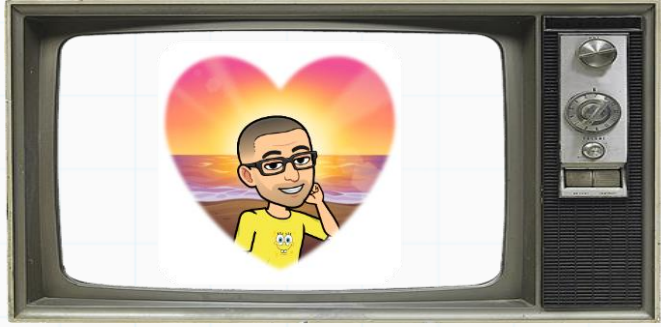
Devemos então percorrer o vetor contador V, vendo as posições diferentes de 0 e preenchendo o vetor original. Veja que o elemento 3 é o primeiro diferente de zero, e aparece 2 vezes.

- Depois o 6 que aparece 3 vezes
- O 8 aparece 1 vez

...

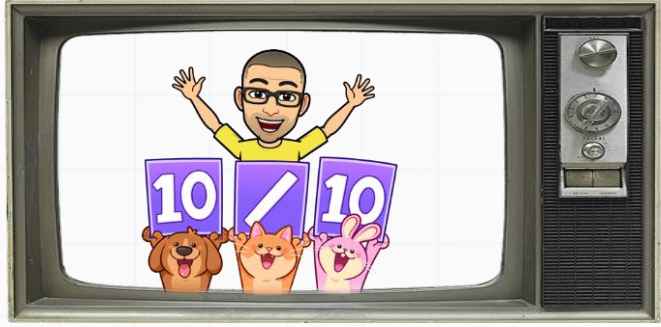
Veja o exemplo de execução:

Buscas e Ordenação - LAB



```
vetor (tam=10) = 6, 3, 9, 10, 15, 6, 8, 12, 3, 6,

   maior elemento = 15
   vetor de contagem = 0) 0, 1) 0, 2) 0, 3) 2, 4) 0, 5) 0, 6) 3, 7) 0, 8) 1, 9) 1, 10) 1, 11) 0, 12) 1, 13) 0, 14) 0, 15) 1,

vetor ordenado (tam=10) = 3, 3, 6, 6, 6, 8, 9, 10, 12, 15,
```

Buscas e Ordenação - LAB



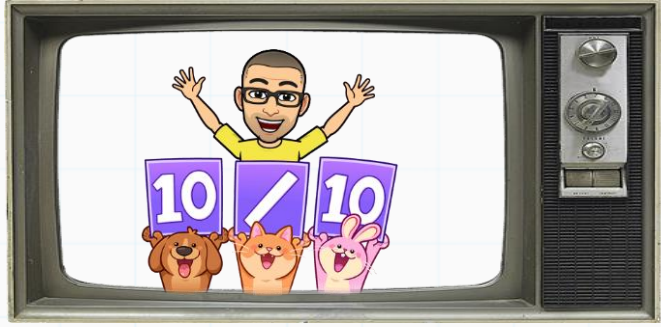
Exercício 4) RadixSort: Como ordenar o vetor abaixo ?

V=

237	146	259	348988	152	163	235	48	36	62

Veja que se o valor máximo do vetor for muito alto (348988), método do CountSort, tem que usar um vetor auxiliar de tamanho absurdo. Esta é a motivação do RadixSort, uma adaptação do método do CountSort. Como funciona ?

Buscas e Ordenação - LAB



Exercício 4) RadixSort: Como ordenar o vetor abaixo ?

V=	237	146	259	348988	152	163	235	48	36	62

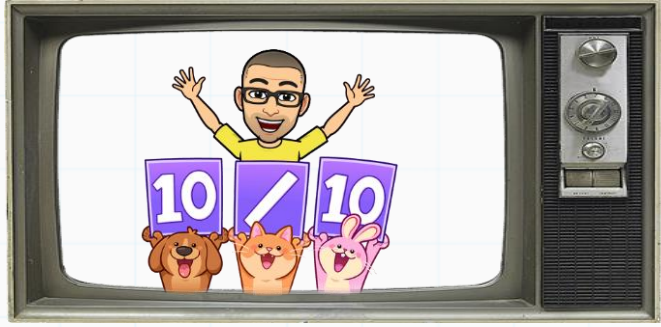
Veja que se o valor máximo do vetor for muito alto (348988), método do CountSort, tem que usar um vetor auxiliar de tamanho absurdo. Esta é a motivação do RadixSort, uma adaptação do método do CountSort. Como funciona ?

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

1) Vamos criar dois vetores auxiliares, um de tamanho 10, que iremos chamar de contador e outro vetor para armazenar o vetor ordenado (tamanho do vetor original)

Buscas e Ordenação - LAB

DESAFIO



Exercício 4) RadixSort: Como ordenar o vetor abaixo ?

V=	237	146	259	348988	152	163	235	48	36	62

Veja que se o valor máximo do vetor for muito alto (348988), método do CountSort, tem que usar um vetor auxiliar de tamanho absurdo. Esta é a motivação do RadixSort, uma adaptação do método do CountSort. Como funciona ?

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

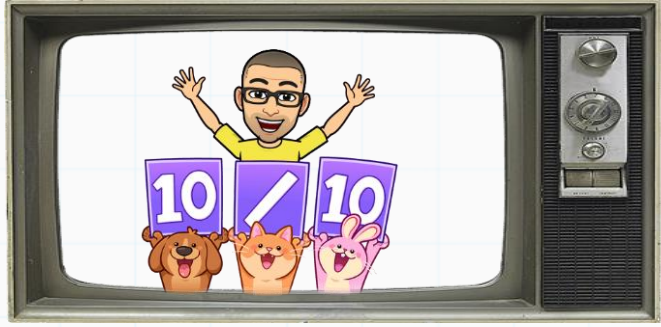
1) Vamos criar dois vetores auxiliares, um de tamanho 10, que iremos chamar de contador e outro vetor para armazenar o vetor ordenado (tamanho do vetor original)

2) Vamos aplicar o CountSort no vetor, mas para descobrir a posição no vetor de contagem, usaremos apenas o último dígito de cada número (isto é num%10), então teremos.

DESAFIO

Buscas e Ordenação - LAB

DESAFIO



Exercício 4) RadixSort: Como ordenar o vetor abaixo ?

V=	237	146	259	348988	152	163	235	48	36	62

Veja que se o valor máximo do vetor for muito alto (348988), método do CountSort, tem que usar um vetor auxiliar de tamanho absurdo. Esta é a motivação do RadixSort, uma adaptação do método do CountSort. Como funciona ?

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

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

1) Vamos criar dois vetores auxiliares, um de tamanho 10, que iremos chamar de contador e outro vetor para armazenar o vetor ordenado (tamanho do vetor original)

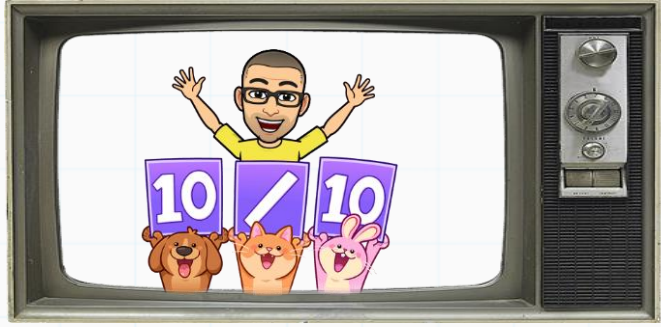
2) Vamos aplicar o CountSort no vetor, mas para descobrir a posição no vetor de contagem, usaremos apenas o último dígito de cada número (isto é num%10), então teremos.

3) Vamos fazer a soma acumulada dos elementos do vetor contador, isto é, o valor da posição i é igual a soma dos elementos da posição 0 até i.

DESAFIO

Buscas e Ordenação - LAB

DESAFIO



Exercício 4) RadixSort: Como ordenar o vetor abaixo ?

V=	237	146	259	348988	152	163	235	48	36	62

Veja que se o valor máximo do vetor for muito alto (348988), método do CountSort, tem que usar um vetor auxiliar de tamanho absurdo. Esta é a motivação do RadixSort, uma adaptação do método do CountSort. Como funciona ?

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

			62						

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

1) Vamos criar dois vetores auxiliares, um de tamanho 10, que iremos chamar de contador e outro vetor para armazenar o vetor ordenado (tamanho do vetor original)

2) Vamos aplicar o CountSort no vetor, mas para descobrir a posição no vetor de contagem, usaremos apenas o último dígito de cada número (isto é num%10), então teremos.

3) Vamos fazer a soma acumulada dos elementos do vetor contador, isto é, o valor da posição i é igual a soma dos elementos da posição 0 até i.

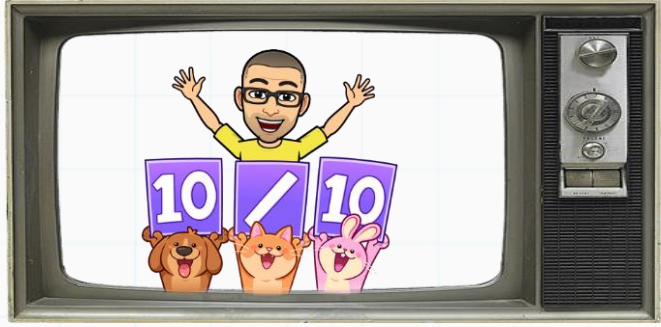
4) Vamos percorrer o vetor original (do fim para o começo), para descobrir onde o elemento da posição i (isto é v[i]) vai ficar, basta ver seu dígito (isto é v[i]%10) qual valor está armazenado no vetor ordenado (isto é contador[v[i]%10]-1).

- Veja o ultimo elemento 62 (com ultimo digito 2), tem posição no vetor ordenado de 2, vai para posição 1

DESAFIO

Buscas e Ordenação - LAB

DESAFIO



Exercício 4) RadixSort: Como ordenar o vetor abaixo ?

V=	237	146	259	348988	152	163	235	48	36	62

Veja que se o valor máximo do vetor for muito alto (348988), método do CountSort, tem que usar um vetor auxiliar de tamanho absurdo. Esta é a motivação do RadixSort, uma adaptação do método do CountSort. Como funciona ?

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

			62						

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

1) Vamos criar dois vetores auxiliares, um de tamanho 10, que iremos chamar de contador e outro vetor para armazenar o vetor ordenado (tamanho do vetor original)

2) Vamos aplicar o CountSort no vetor, mas para descobrir a posição no vetor de contagem, usaremos apenas o último dígito de cada número (isto é num%10), então teremos.

3) Vamos fazer a soma acumulada dos elementos do vetor contador, isto é, o valor da posição i é igual a soma dos elementos da posição 0 até i.

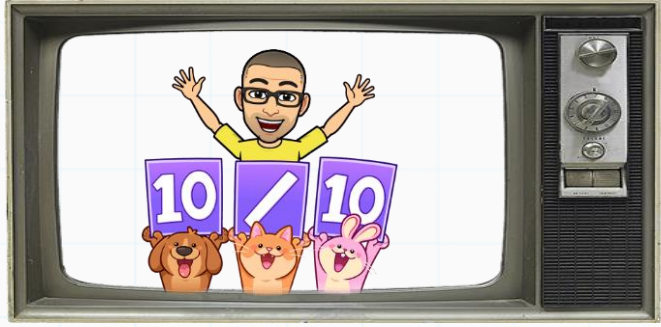
4) Vamos percorrer o vetor original (do fim para o começo), para descobrir onde o elemento da posição i (isto é v[i]) vai ficar, basta ver seu dígito (isto é v[i]%10) qual valor está armazenado no vetor ordenado (isto é contador[v[i]%10]-1).

5) Diminuímos de uma unidade o valor dessa posição no vetor ordenado.

- Veja o ultimo elemento 62 (com ultimo digito 2), tem posição no vetor ordenado de 2, vai para posição 1 , diminui de um os elementos de ultimo dígito 1.

Buscas e Ordenação - LAB

DESAFIO



Exercício 4) RadixSort: Como ordenar o vetor abaixo ?

V=	237	146	259	348988	152	163	235	48	36	62

Veja que se o valor máximo do vetor for muito alto (348988), método do CountSort, tem que usar um vetor auxiliar de tamanho absurdo. Esta é a motivação do RadixSort, uma adaptação do método do CountSort. Como funciona ?

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

	62				36				

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

1) Vamos criar dois vetores auxiliares, um de tamanho 10, que iremos chamar de contador e outro vetor para armazenar o vetor ordenado (tamanho do vetor original)

2) Vamos aplicar o CountSort no vetor, mas para descobrir a posição no vetor de contagem, usaremos apenas o último dígito de cada número (isto é num%10), então teremos.

3) Vamos fazer a soma acumulada dos elementos do vetor contador, isto é, o valor da posição i é igual a soma dos elementos da posição 0 até i.

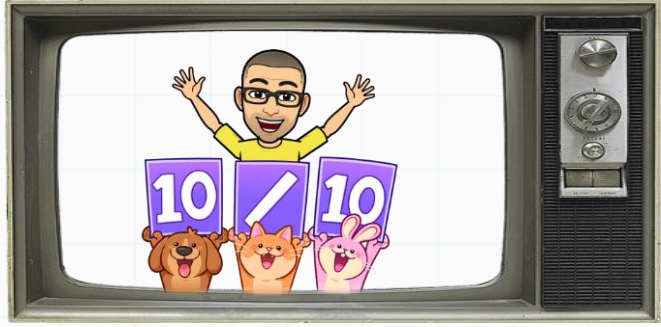
4) Vamos percorrer o vetor original (do fim para o começo), para descobrir onde o elemento da posição i (isto é v[i]) vai ficar, basta ver seu dígito (isto é v[i]%10) qual valor está armazenado no vetor ordenado (isto é contador[v[i]%10]-1).

5) Diminuímos de uma unidade o valor dessa posição no vetor ordenado.

- Veja o elemento 36 (com ultimo digito 6), tem posição no vetor ordenado de 6, vai para posição 5

Buscas e Ordenação - LAB

DESAFIO



Exercício 4) RadixSort: Como ordenar o vetor abaixo ?

V=	237	146	259	348988	152	163	235	48	36	62

Veja que se o valor máximo do vetor for muito alto (348988), método do CountSort, tem que usar um vetor auxiliar de tamanho absurdo. Esta é a motivação do RadixSort, uma adaptação do método do CountSort. Como funciona ?

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

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

	62				36				

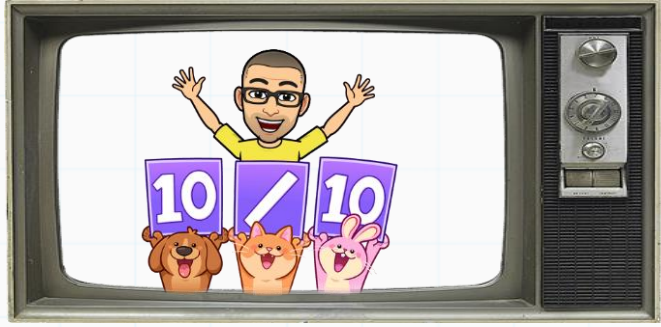
1) Vamos criar dois vetores auxiliares, um de tamanho 10, que iremos chamar de contador e outro vetor para armazenar o vetor ordenado (tamanho do vetor original)

2) Vamos aplicar o CountSort no vetor, mas para descobrir a posição no vetor de contagem, usaremos apenas o último dígito de cada número (isto é num%10), então teremos.
3) Vamos fazer a soma acumulada dos elementos do vetor contador, isto é, o valor da posição i é igual a soma dos elementos da posição 0 até i.
4) Vamos percorrer o vetor original (do fim para o começo), para descobrir onde o elemento da posição i (isto é v[i]) vai ficar, basta ver seu dígito (isto é v[i]%10) qual valor está armazenado no vetor ordenado (isto é contador[v[i]%10]-1).
5) Diminuímos de uma unidade o valor dessa posição no vetor ordenado.

- Veja o elemento 36 (com ultimo digito 6), tem posição no vetor ordenado de 6, vai para posição 5 , diminui de um os elementos de ultimo dígito 6.

Buscas e Ordenação - LAB

DESAFIO



Exercício 4) RadixSort: Como ordenar o vetor abaixo ?

V=

237	146	259	348988	152	163	235	48	36	62

Veja que se o valor máximo do vetor for muito alto (348988), método do CountSort, tem que usar um vetor auxiliar de tamanho absurdo. Esta é a motivação do RadixSort, uma adaptação do método do CountSort. Como funciona ?

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

	62				36			48	

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

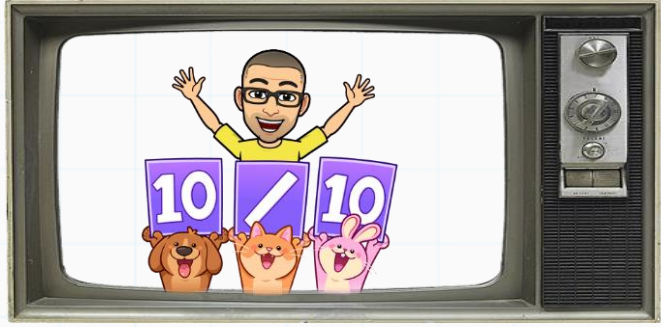
1) Vamos criar dois vetores auxiliares, um de tamanho 10, que iremos chamar de contador e outro vetor para armazenar o vetor ordenado (tamanho do vetor original)

2) Vamos aplicar o CountSort no vetor, mas para descobrir a posição no vetor de contagem, usaremos apenas o último dígito de cada número (isto é num%10), então teremos.
3) Vamos fazer a soma acumulada dos elementos do vetor contador, isto é, o valor da posição i é igual a soma dos elementos da posição 0 até i.
4) Vamos percorrer o vetor original (do fim para o começo), para descobrir onde o elemento da posição i (isto é v[i]) vai ficar, basta ver seu dígito (isto é v[i]%10) qual valor está armazenado no vetor ordenado (isto é contador[v[i]%10]-1).
5) Diminuímos de uma unidade o valor dessa posição no vetor ordenado.

- Veja o elemento 48 (com ultimo digito 8), tem posição no vetor ordenado de 9, vai para posição 8

Buscas e Ordenação - LAB

DESAFIO



Exercício 4) RadixSort: Como ordenar o vetor abaixo ?

V=	237	146	259	348988	152	163	235	48	36	62

Veja que se o valor máximo do vetor for muito alto (348988), método do CountSort, tem que usar um vetor auxiliar de tamanho absurdo. Esta é a motivação do RadixSort, uma adaptação do método do CountSort. Como funciona ?

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

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

		62				36			48		

1) Vamos criar dois vetores auxiliares, um de tamanho 10, que iremos chamar de contador e outro vetor para armazenar o vetor ordenado (tamanho do vetor original)

2) Vamos aplicar o CountSort no vetor, mas para descobrir a posição no vetor de contagem, usaremos apenas o último dígito de cada número (isto é num%10), então teremos.

3) Vamos fazer a soma acumulada dos elementos do vetor contador, isto é, o valor da posição i é igual a soma dos elementos da posição 0 até i.

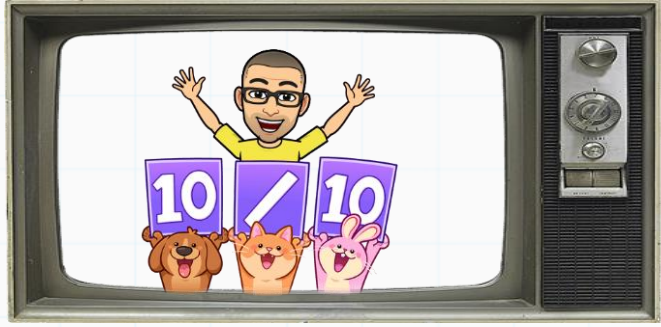
4) Vamos percorrer o vetor original (do fim para o começo), para descobrir onde o elemento da posição i (isto é v[i]) vai ficar, basta ver seu dígito (isto é v[i]%10) qual valor está armazenado no vetor ordenado (isto é contador[v[i]%10]-1).

5) Diminuímos de uma unidade o valor dessa posição no vetor ordenado.

- Veja o elemento 48 (com ultimo digito 8), tem posição no vetor ordenado de 9, vai para posição 8 , diminui de um os elementos de ultimo dígito 8.

Buscas e Ordenação - LAB

DESAFIO



Exercício 4) RadixSort: Como ordenar o vetor abaixo ?

V=

237	146	259	348988	152	163	235	48	36	62

Veja que se o valor máximo do vetor for muito alto (348988), método do CountSort, tem que usar um vetor auxiliar de tamanho absurdo. Esta é a motivação do RadixSort, uma adaptação do método do CountSort. Como funciona ?

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

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

152	62	163	235	146	36	237	348988	48	259

1) Vamos criar dois vetores auxiliares, um de tamanho 10, que iremos chamar de contador e outro vetor para armazenar o vetor ordenado (tamanho do vetor original)

2) Vamos aplicar o CountSort no vetor, mas para descobrir a posição no vetor de contagem, usaremos apenas o último dígito de cada número (isto é num%10), então teremos.

3) Vamos fazer a soma acumulada dos elementos do vetor contador, isto é, o valor da posição i é igual a soma dos elementos da posição 0 até i.

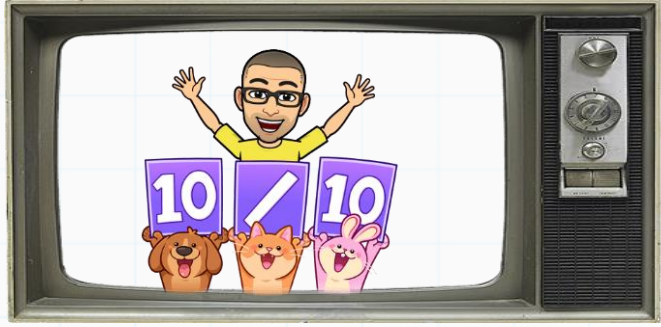
4) Vamos percorrer o vetor original (do fim para o começo), para descobrir onde o elemento da posição i (isto é v[i]) vai ficar, basta ver seu dígito (isto é v[i]%10) qual valor está armazenado no vetor ordenado (isto é contador[v[i]%10]-1).

5) Diminuímos de uma unidade o valor dessa posição no vetor ordenado.

6) O vetor ordenado pode ser copiado de volta para o vetor original para começarmos uma nova iteração. O vetor atual está ordenado em relação ao ultimo digito.

Buscas e Ordenação - LAB

DESAFIO



Exercício 4) RadixSort: Como ordenar o vetor abaixo ?

V=	152	62	163	235	146	36	237	348988	48	259

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

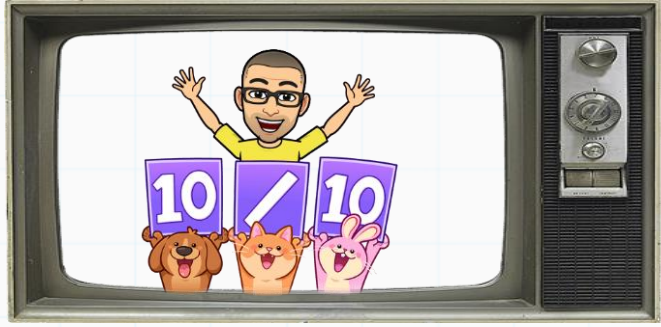
2) Vamos aplicar o CountSort no vetor, mas para descobrir a posição no vetor de contagem, usaremos apenas o penúltimo dígito de cada número (isto é (num/10)%10), então teremos.

Veja que podemos pegar o dígito de cada iteração com a formula:

1 iteração -> ultimo digito -> (num / 1) % 10
2 iteração -> penúltimo digito -> (num / 10) % 10
3 iteração -> antepenúltimo digito -> (num / 100) % 10
...
N Iteração -> -> (num / 10^{N-1}) % 10

Buscas e Ordenação - LAB

DESAFIO



Exercício 4) RadixSort: Como ordenar o vetor abaixo ?

V=	152	62	163	235	146	36	237	348988	48	259

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

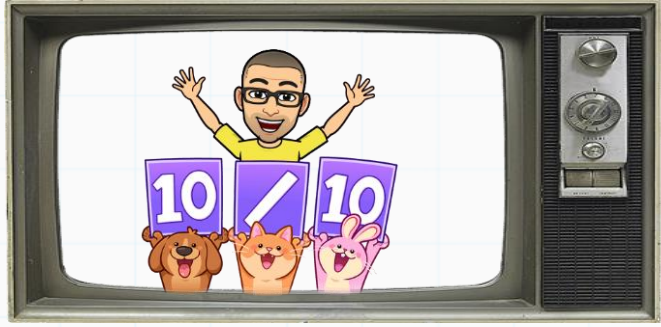
0	0	0	3	5	7	9	9	10	10
0	1	2	3	4	5	6	7	8	9

2) Vamos aplicar o CountSort no vetor, mas para descobrir a posição no vetor de contagem, usaremos apenas o penúltimo dígito de cada número (isto é (num/10)%10), então teremos.

3) Vamos fazer a soma acumulada dos elementos do vetor contador, isto é, o valor da posição i é igual a soma dos elementos da posição 0 até i.

Buscas e Ordenação - LAB

DESAFIO



Exercício 4) RadixSort: Como ordenar o vetor abaixo ?

V=	152	62	163	235	146	36	237	348988	48	259

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

→

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

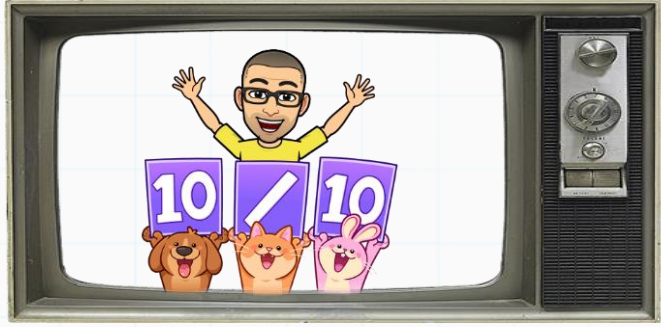
						259			

2) Vamos aplicar o CountSort no vetor, mas para descobrir a posição no vetor de contagem, usaremos apenas o penúltimo dígito de cada número (isto é (num/10)%10), então teremos.
3) Vamos fazer a soma acumulada dos elementos do vetor contador, isto é, o valor da posição i é igual a soma dos elementos da posição 0 até i.
4) Vamos percorrer o vetor (do fim para o começo), para descobrir onde o elemento da posição i (isto é v[i]) vai ficar, basta ver seu dígito (isto é (v[i]/10)%10) qual valor está armazenado no vetor ordenado (isto é contador[(v[i]/10)%10]-1).
5) Diminuímos de uma unidade o valor dessa posição no vetor ordenado

- Veja o ultimo elemento 259 (com penúltimo digito 5), tem posição no vetor ordenado de 7, vai para posição 6, diminui de um os elementos de ultimo dígito 6.

Buscas e Ordenação - LAB

DESAFIO



Exercício 4) RadixSort: Como ordenar o vetor abaixo ?

V=	152	62	163	235	146	36	237	348988	48	259

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

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

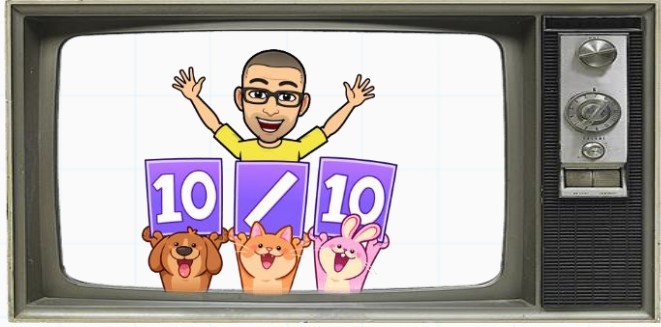
235	36	237	146	48	152	259	62	163	348988

2) Vamos aplicar o CountSort no vetor, mas para descobrir a posição no vetor de contagem, usaremos apenas o penúltimo dígito de cada número (isto é (num/10)%10), então teremos.
3) Vamos fazer a soma acumulada dos elementos do vetor contador, isto é, o valor da posição i é igual a soma dos elementos da posição 0 até i.
4) Vamos percorrer o vetor (do fim para o começo), para descobrir onde o elemento da posição i (isto é v[i]) vai ficar, basta ver seu dígito (isto é (v[i]/10)%10) qual valor está armazenado no vetor ordenado (isto é contador[(v[i]/10)%10]-1).
5) Diminuímos de uma unidade o valor dessa posição no vetor ordenado
6) O vetor ordenado pode ser copiado de volta para o vetor original para começarmos uma nova iteração. O vetor atual está ordenado em relação ao penúltimo digito.

Porém dentre os elementos de mesmo segundo digito, veja que eles estão ordenados em relação ao primeiro digito já.
Ex: 235, 36 e 237 são os primeiros da ordem porque tem 3 como segundo digito, porem estão em ordem em relação ao primeiro digito.

Buscas e Ordenação - LAB

DESAFIO



Exercício 4) RadixSort: Como ordenar o vetor abaixo ?

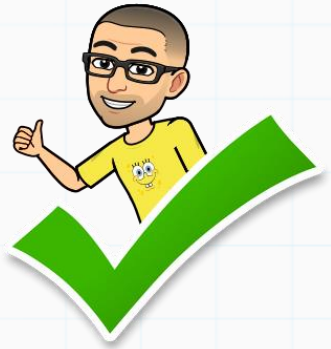
V=	235	036	237	146	048	152	259	062	163	348988

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

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

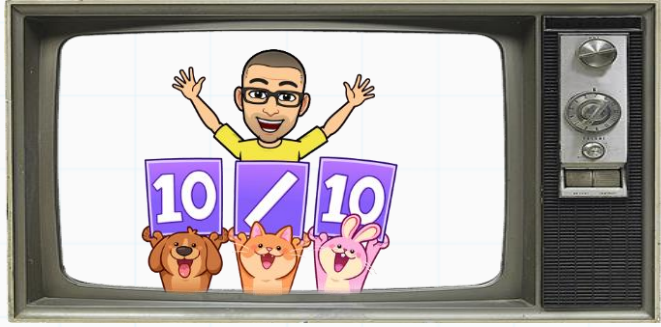
36	48	62	146	152	163	235	237	259	348988

- Ao repetirmos o mesmo processo do CountSort, agora vendo o antepenúltimo digito (isto é (num/100)%10), teremos:
- veja que o vetor já está ordenado



Buscas e Ordenação - LAB

DESAFIO



Exercício 4) RadixSort: Como ordenar o vetor abaixo ?

V=	235	036	237	146	048	152	259	062	163	348988

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

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

36	48	62	146	152	163	235	237	259	348988

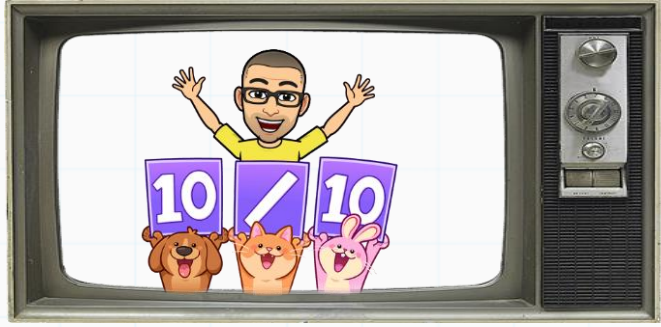
- Ao repetirmos o mesmo processo do CountSort, agora vendo o antepenúltimo digito (isto é (num/100)%10), teremos:
- veja que o vetor já está ordenado.

- Então temos que repetir a aplicação do CountSort analisando quantos dígitos ?
- R: número de dígitos do maior número, começando do ultimo até o primeiro.

- Veja a execução a seguir:

Buscas e Ordenação - LAB

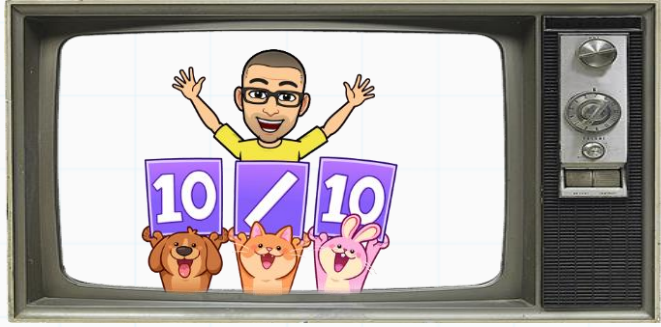
DESAFIO




```
vetor (tam=10) = 237, 146, 259, 348988, 152, 163, 235, 48, 36, 62,

maior elemento = 348988
348988 tem 6 digitos

   digito 1 (de tras para frente)
   contador           = 0) 0, 1) 0, 2) 2, 3) 1, 4) 0, 5) 1, 6) 2, 7) 1, 8) 2, 9) 1,
   contador acumulado = 0) 0, 1) 0, 2) 2, 3) 3, 4) 3, 5) 4, 6) 6, 7) 7, 8) 9, 9) 10,
   ordenado           = 152, 62, 163, 235, 146, 36, 237, 348988, 48, 259,

   digito 2 (de tras para frente)
   contador           = 0) 0, 1) 0, 2) 0, 3) 3, 4) 2, 5) 2, 6) 2, 7) 0, 8) 1, 9) 0,
   contador acumulado = 0) 0, 1) 0, 2) 0, 3) 3, 4) 5, 5) 7, 6) 9, 7) 9, 8) 10, 9) 10,
   ordenado           = 235, 36, 237, 146, 48, 152, 259, 62, 163, 348988,

   digito 3 (de tras para frente)
   contador           = 0) 3, 1) 3, 2) 3, 3) 0, 4) 0, 5) 0, 6) 0, 7) 0, 8) 0, 9) 1,
   contador acumulado = 0) 3, 1) 6, 2) 9, 3) 9, 4) 9, 5) 9, 6) 9, 7) 9, 8) 9, 9) 10,
   ordenado           = 36, 48, 62, 146, 152, 163, 235, 237, 259, 348988,

   digito 4 (de tras para frente)
   contador           = 0) 9, 1) 0, 2) 0, 3) 0, 4) 0, 5) 0, 6) 0, 7) 0, 8) 1, 9) 0,
   contador acumulado = 0) 9, 1) 9, 2) 9, 3) 9, 4) 9, 5) 9, 6) 9, 7) 9, 8) 10, 9) 10,
   ordenado           = 36, 48, 62, 146, 152, 163, 235, 237, 259, 348988,
```

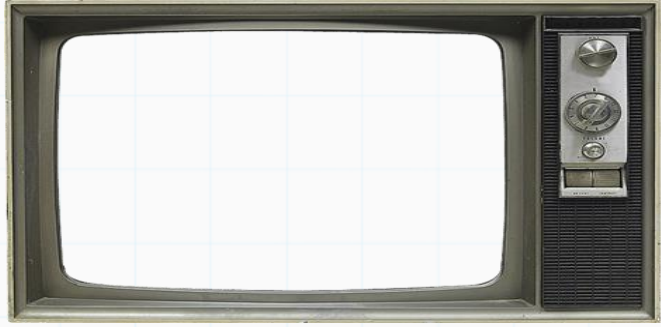
Buscas e Ordenação - LAB

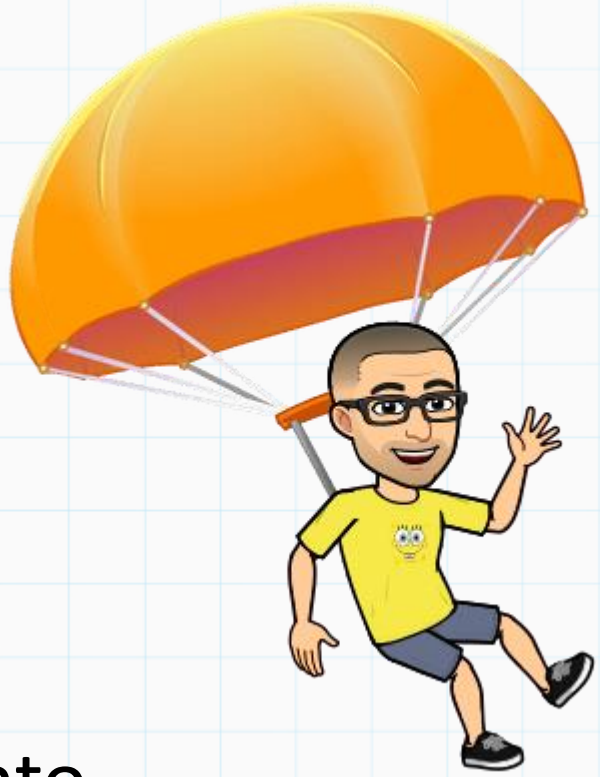
DESAFIO




```
   digito 5 (de tras para frente)
   contador           = 0) 9, 1) 0, 2) 0, 3) 0, 4) 1, 5) 0, 6) 0, 7) 0, 8) 0, 9) 0,
   contador acumulado = 0) 9, 1) 9, 2) 9, 3) 9, 4) 10, 5) 10, 6) 10, 7) 10, 8) 10, 9) 10,
   ordenado           = 36, 48, 62, 146, 152, 163, 235, 237, 259, 348988,

   digito 6 (de tras para frente)
   contador           = 0) 9, 1) 0, 2) 0, 3) 1, 4) 0, 5) 0, 6) 0, 7) 0, 8) 0, 9) 0,
   contador acumulado = 0) 9, 1) 9, 2) 9, 3) 10, 4) 10, 5) 10, 6) 10, 7) 10, 8) 10, 9) 10,
   ordenado           = 36, 48, 62, 146, 152, 163, 235, 237, 259, 348988,

vetor ordenado (tam=10) = 36, 48, 62, 146, 152, 163, 235, 237, 259, 348988,
```

Até a próxima





Slides baseados no curso de Aline Nascimento