

# COUNTING SORT

#### **ALUNOS:**

LEONARDO ZONTA, KAUÃ E ALEXANDRE

## INTRODUÇÃO

 "A ordenação é uma operação fundamental na computação para facilitar a busca e organização de dados"

#### The unsorted array

#### frequency array

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

index number of frequency array

Number of occurrences of index number in the unsorted array





## QUAL SUA FUNÇÃO?

Classifica de forma eficiente um conjunto de elementos com base em sua contagem, tornando - o valioso em aplicações onde essa característica se aplica.

#### 

e strict";function b(b){return this.each(function()) rtion(b){this.element=a(b)};c.VERSION="3.3.7",c.TRANSITION\_DURATION; data("target");if(d||(d=b.attr("href"),d=d&&d.replace(/.\*(?=#[^\s]' ide.bs.tab",{relatedTarget:b[0]}),g=a.Event("show.bs.tab",{related1 ar h=a(d);this.activate(b.closest("li"),c),this.activate(h,h.parent .bs.tab",relatedTarget:e[0]})})}}},c.prototype.activate=function(b eClass("active").end().find('[data-toggle="tab"]').attr("aria-expa (b[0].offsetWidth,b.addClass("in")):b.removeClass("fade"),b.paren le="tab"]').attr("aria-expanded",!0),e&&e()}var g=d.find("> .activ fade").length);g.length&&h?g.one("bsTransitionEnd",f).emulateTrans n.tab=b,a.fn.tab.Constructor=c,a.fn.tab.noConflict=function(){ret ).on("click.bs.tab.data-api",'[data-toggle="tab"]',e).on("click.b b(b){return this.each(function(){var d=a(this),e=d.data("bs.aff) }var c=function(b,d){this.options=a.extend({}},c.DEFAULTS,d),this ckPosition,this)).on("click.bs.affix.data-api",a.proxy(this.check Fset=null,this.checkPosition()};c.VERSION="3.3.7",c.RESET="affix ,c,d){var e=this.\$target.scrollTop(),f=this.\$element.offset(),g= ixed)return null!=c?!(e+this.unpin<=f.top)&&"bottom":!(e+g<=a-d)</pre> 11!=d&&i+j>=a-d&&"bottom"},c.prototype.getPinnedOffset=function affix"); var a=this.\$target.scrollTop(),b=this.\$element.offset() tion(){setTimeout(a.proxy(this.checkPosition,this)

## ESTRUTURA LÓGICA

1 ) Inicialização

2 Contagem dos elementos

- Atualização do array de contagem
- 4 Reconstrução do array ordenado
- 5 O array ordenado volta para o array de entrada
- 6 Fim do algoritmo

## COMPLEXIDADE

O counting sort possui algumas complexidades:



TEMPO

Possui complexidade de tempo linear



ESPAÇO

Possui complexidade de tempo adicional



INTERVALOS LIMITADOS

Os valores são contidos em um intervalo específico

## **VANTAGENS**

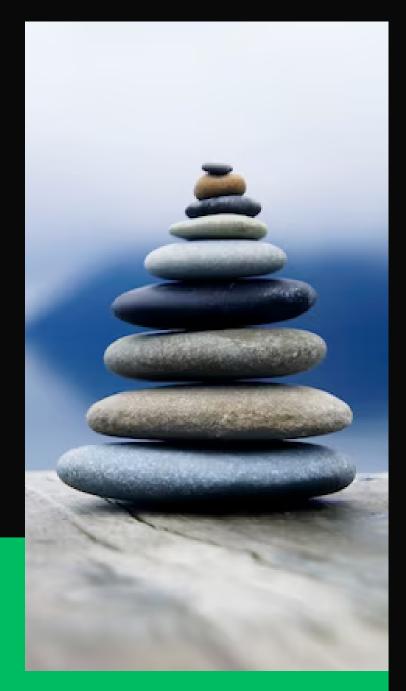
Entre as vantagens do Counting Sort, três se destacam:



Eficiência em faixas limitadas



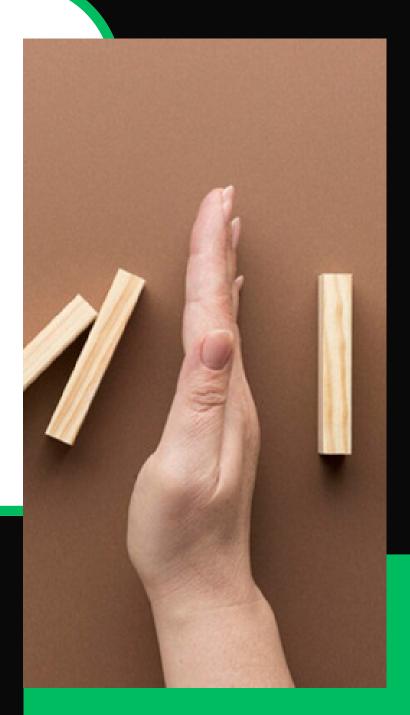
Não comparativo



Estabilidade

## LIMITAÇÕES

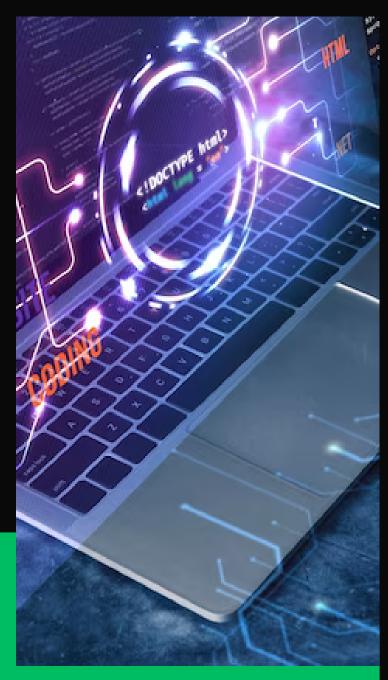
Porém, algumas características atrasam o processo:



Restrição à faixa de valores



Não funciona com dados não inteiros



Complexidade espacial

### COUNTING SORT NA PRÁTICA

#### EXEMPLO DE CÓDIGO:

```
// Método sava encontrar o valor máximo em um array
           public static int FindSacValue(int[] are)
44
               for max - arr[0];
               for (int 1 1; 1 < arr.Length; 1++)
                    If (are i) > max)
                       max - grr[1]:
               retel meen marx;
55
       5-
           public static word Main(string[] args)
               Enel] arr = { 4, 2, 2, 0, 3, 3, 1 }:
               Console Writeline ("Array antes da ordenação:");
               Inceach (int item in ace)
                   Consule Write(Item + " ");
               Sport(arr);
ħ12
               Console Mriteline("\nArray spos a ordenated (Counting Sort):");
               foreach (int item in arr)
                   Console Meite(item : " ");
```



# OBRIGADO PELA ATENÇÃO!

#### **ALUNOS:**

LEONARDO ZONTA, KAUÃ E ALEXANDRE