

---

---

# Estrutura de Dados

— Métodos de Ordenação —  
Shellsort

---

---

# Métodos de ordenação

- Dentro da categoria de métodos de ordenação em memória primária existem alguns métodos de ordenação que são mais eficientes que os métodos simples:
  - Mergesort;
  - Quicksort;
  - Heapsort;
  - **Shellsort;**

# Ordenação por incremento decrescente

- Uma outra forma de ordenar vetores é por meio do método de ordenação por incrementos decrescentes;
  - Também chamado de shellsort, pois a primeira ordenação foi feita utilizando
- Esse método ordena subvetores do vetor original;
  - Podem ser:
    - Subvetores físicos, ou seja fora do vetor;
    - Subvetores “lógicos”, ou seja, dentro do vetor;
      - Manipulação dos subvetores feita pelos índices;
- O tamanho dos subvetores cresce a cada iteração;
  - O tamanho do último subvetor deve ser a quantidade de elementos a serem ordenados;
- Os subvetores são ordenados a cada iteração pelo método da inserção;

# Shellsort

- Cada subvetor tem seus elementos escolhidos de acordo com uma variável de incremento;
  - Dado um incremento  $g$ :
    - O primeiro subvetor será formado pelos elementos:  
 $\text{vetor}[0], \text{vetor}[g], \text{vetor}[2*g], \text{vetor}[3*g], \dots$
    - O segundo subvetor será formado pelos elementos:  
 $\text{vetor}[1], \text{vetor}[g+1], \text{vetor}[2*g+1], \text{vetor}[3*g+1], \dots$
    - O  $n$ -ésimo subvetor será formado pelos elementos:  
 $\text{vetor}[n-1], \text{vetor}[1*g+n-1], \text{vetor}[2*g+n-1], \text{vetor}[3*g+n-1], \dots$

# Shellsort

- O método de ordenação de Shell ordena os valores parcialmente;
- Esse algoritmo tira proveito de que alguns métodos simples são rápidos quando os vetores estão parcialmente ordenados e não tem custo muito grande para vetores pequenos;
- O truque do algoritmo é comparar primeiro os elementos que estão mais distantes uns dos outros, e ir reduzindo esse intervalo até que chegue em 1.

# Shellsort

- Um problema a ser tratado é como escolher o valor ótimo do incremento;
  - Não há prova formal que indique a melhor sequência;
  - Uma boa sequência conhecida é da forma ;
    - $G_1 = 1$ ;
    - $G_{i+1} = 3G_i + 1$ ;
  - Onde  $G_1$  é o incremento da última iteração
    - Por exemplo:
      - Para  $n = 1000$  os incrementos seriam: 634, 121, 13, 4, 1;
- Um detalhe importante é que o último valor do gap (G) **deve** ser 1;

# Shellsort

- Algoritmo:
  - Enquanto o incremento for maior ou igual a 1:
    - Divida o vetor em h subvetores;
    - Para cada subvetor:
      - Ordene o subvetor com o insertion sort;
    - Vá para o próximo incremento

# Execução do Algoritmo

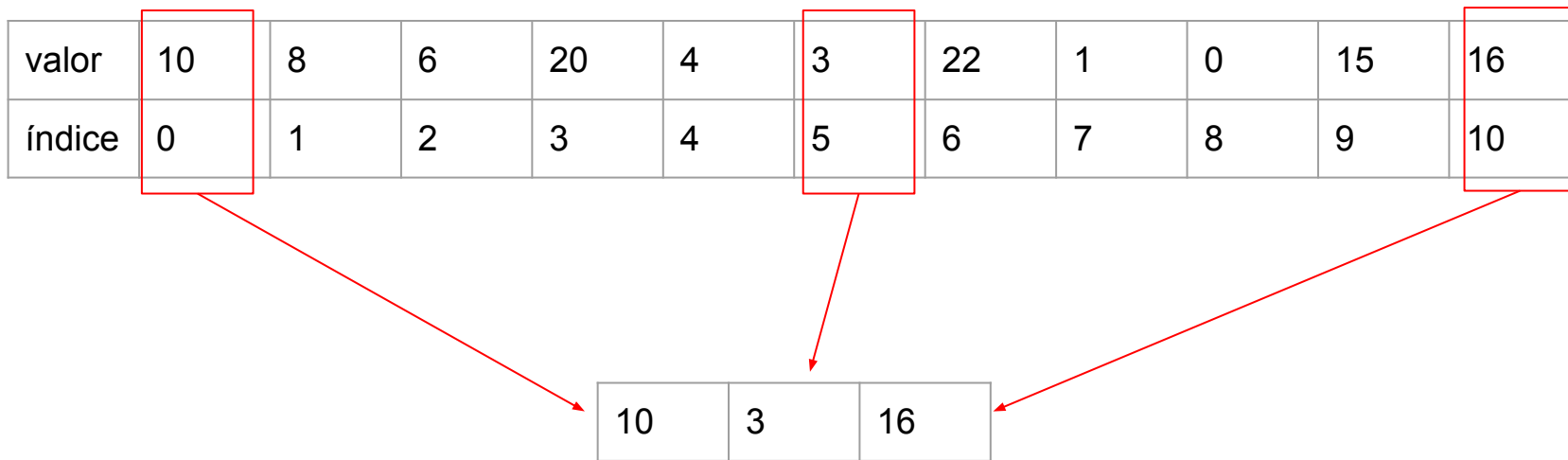
Suponha a execução para os valores de incremento 5, 3 e 1

valor	10	8	6	20	4	3	22	1	0	15	16
índice	0	1	2	3	4	5	6	7	8	9	10



# Execução do Algoritmo

Incremento 5



# Execução do Algoritmo

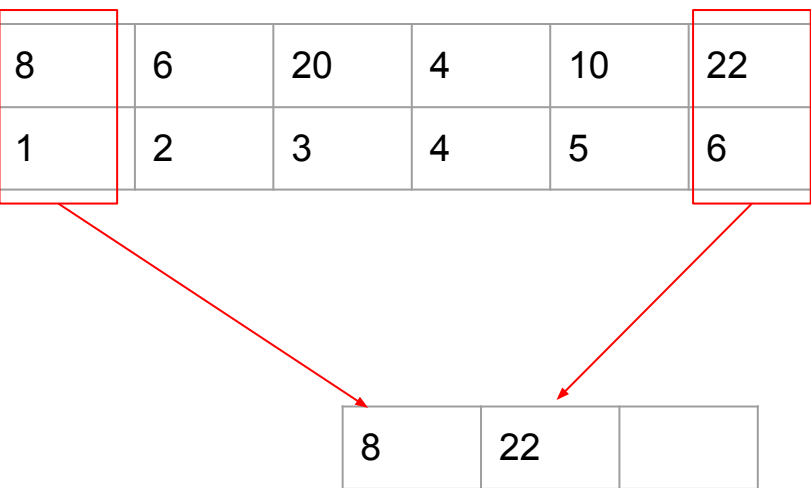
valor	3	8	6	20	4	10	22	1	0	15	16
índice	0	1	2	3	4	5	6	7	8	9	10

The diagram illustrates the execution of an algorithm. It shows a table with two rows: 'valor' (value) and 'índice' (index). The values are 3, 8, 6, 20, 4, 10, 22, 1, 0, 15, and 16, corresponding to indices 0 through 10. Three specific values (3, 10, and 16) are highlighted with red boxes. Red arrows point from these boxes to a separate box below containing the values 3, 10, and 16.

3	10	16
---	----	----

# Execução do Algoritmo

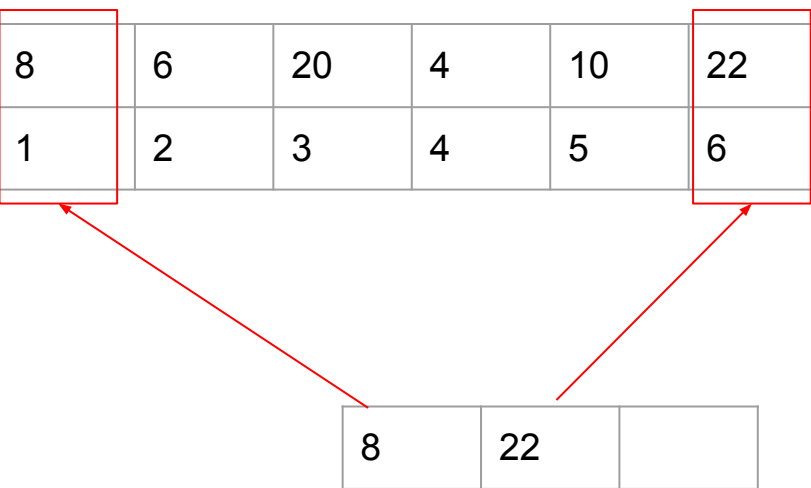
valor	3	8	6	20	4	10	22	1	0	15	16
índice	0	1	2	3	4	5	6	7	8	9	10



8	22	
---	----	--

# Execução do Algoritmo

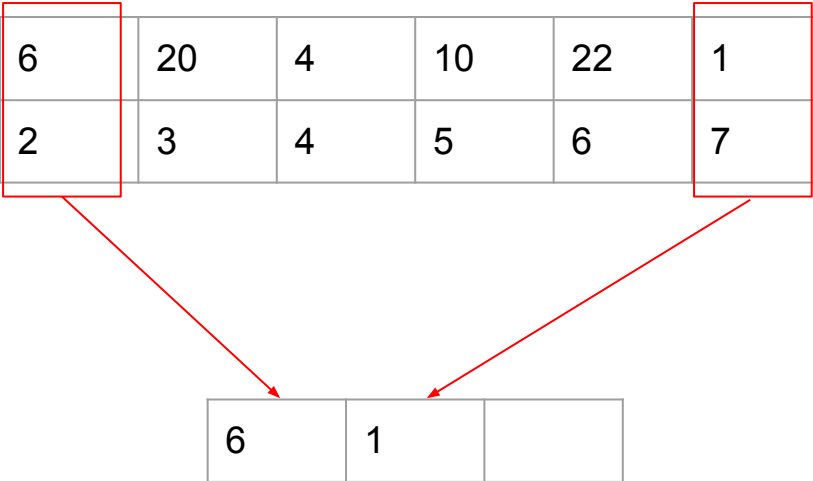
valor	3	8	6	20	4	10	22	1	0	15	16
índice	0	1	2	3	4	5	6	7	8	9	10



8	22	
---	----	--

# Execução do Algoritmo

valor	3	8	6	20	4	10	22	1	0	15	16
índice	0	1	2	3	4	5	6	7	8	9	10



6	1	
---	---	--

# Execução do Algoritmo

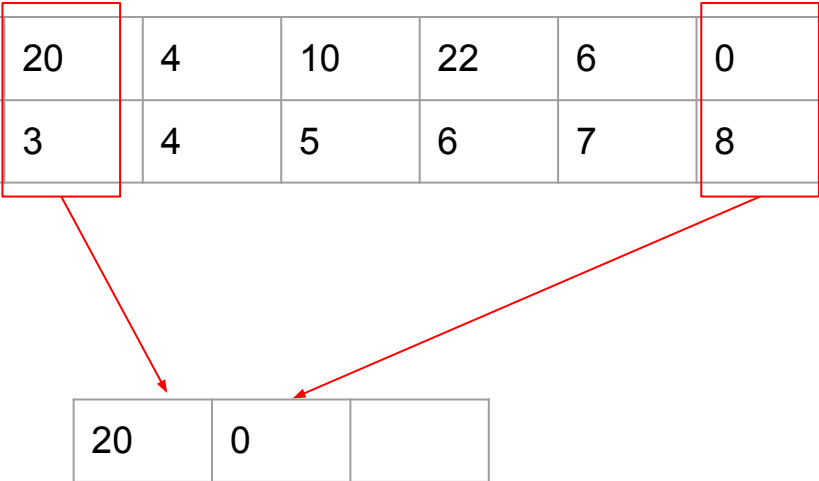
valor	3	8	1	20	4	10	22	6	0	15	16
índice	0	1	2	3	4	5	6	7	8	9	10

1	6	
---	---	--

# Execução do Algoritmo

valor	3	8	1	20	4	10	22	6	0	15	16
índice	0	1	2	3	4	5	6	7	8	9	10



20	0	
----	---	--

# Execução do Algoritmo

valor	3	8	1	0	4	10	22	6	20	15	16
índice	0	1	2	3	4	5	6	7	8	9	10

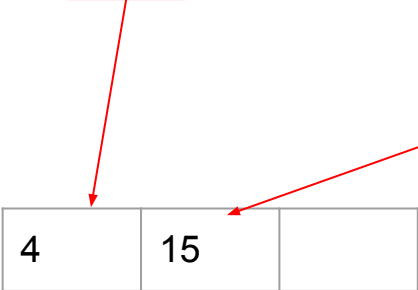
Diagram illustrating the execution of an algorithm. The main array contains values at indices 0 to 10. The values 0 (at index 3) and 20 (at index 8) are highlighted. Red arrows point from these highlighted values to a smaller array below, which contains the values 0 and 20 in its first two cells.

0	20	
---	----	--



# Execução do Algoritmo

valor	3	8	1	0	4	10	22	6	20	15	16
índice	0	1	2	3	4	5	6	7	8	9	10



4	15	
---	----	--

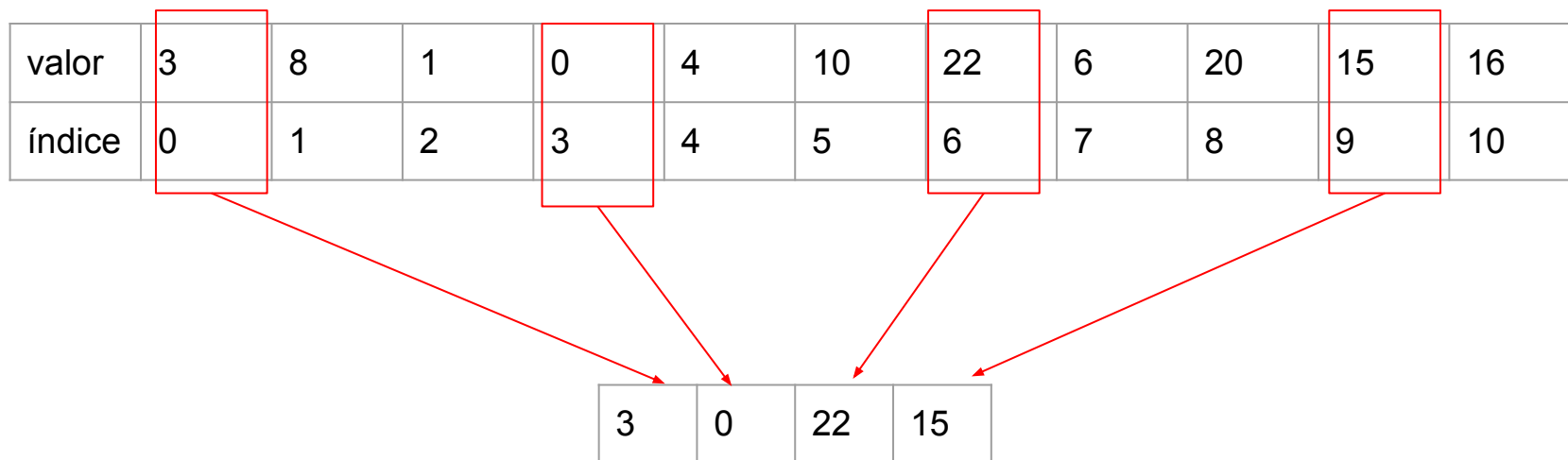
# Execução do Algoritmo

valor	3	8	1	0	4	10	22	6	20	15	16
índice	0	1	2	3	4	5	6	7	8	9	10

4	15	
---	----	--

# Execução do Algoritmo

Incremento 3



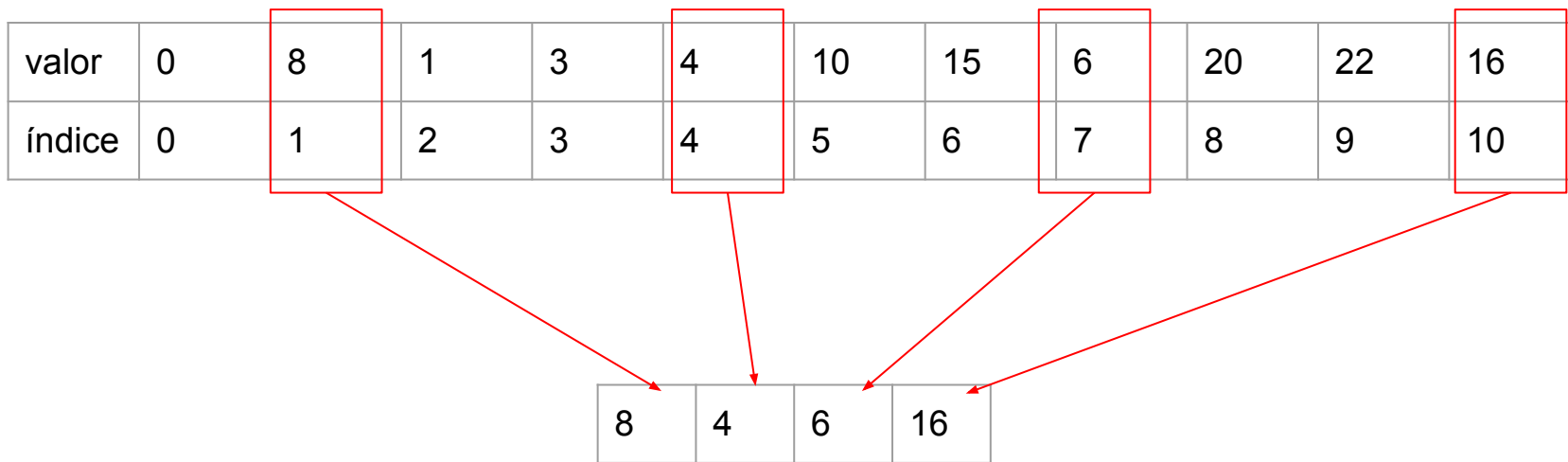
# Execução do Algoritmo

valor	0	8	1	3	4	10	15	6	20	22	16
índice	0	1	2	3	4	5	6	7	8	9	10

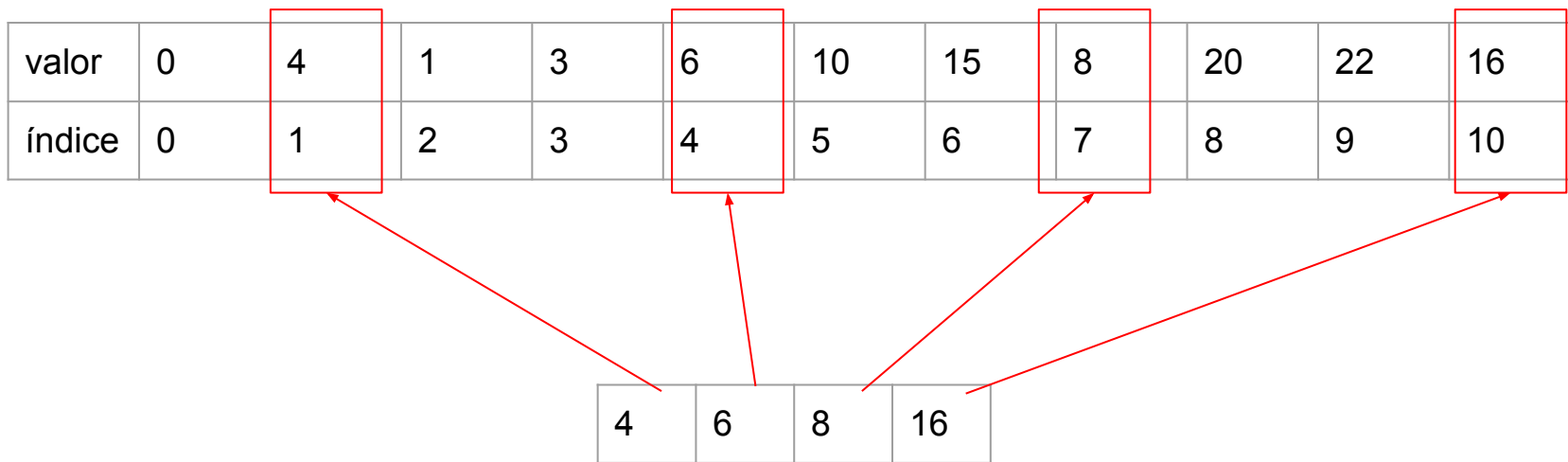
Diagram illustrating the execution of an algorithm. The main table shows values and their corresponding indices. Four specific values (0, 3, 15, 22) are highlighted with red boxes. Red arrows point from these values to a smaller table below, which contains the sequence [0, 3, 15, 22].

0	3	15	22
---	---	----	----

# Execução do Algoritmo



# Execução do Algoritmo



# Execução do Algoritmo

valor	0	4	1	3	6	10	15	8	20	22	16
índice	0	1	2	3	4	5	6	7	8	9	10

Diagram illustrating the execution of an algorithm. The top table shows a sequence of values and their corresponding indices. Three specific values (1, 10, and 20) are highlighted with red boxes, and red arrows point from these boxes to a bottom table, indicating the selection of these values.

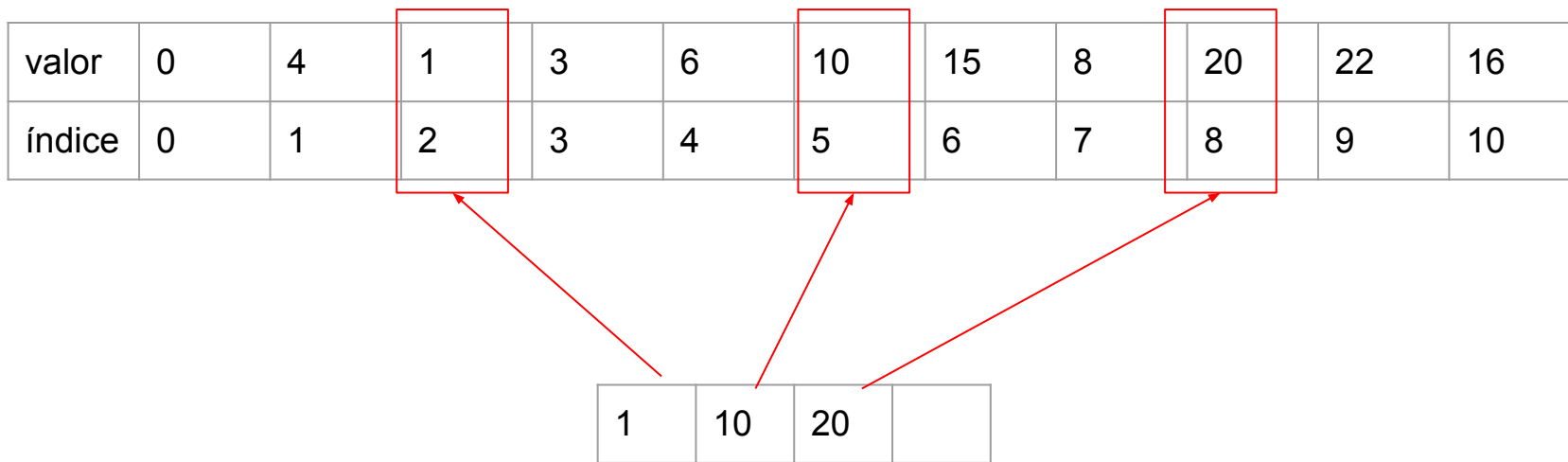
1	10	20	
---	----	----	--

# Execução do Algoritmo

valor	0	4	1	3	6	10	15	8	20	22	16
índice	0	1	2	3	4	5	6	7	8	9	10

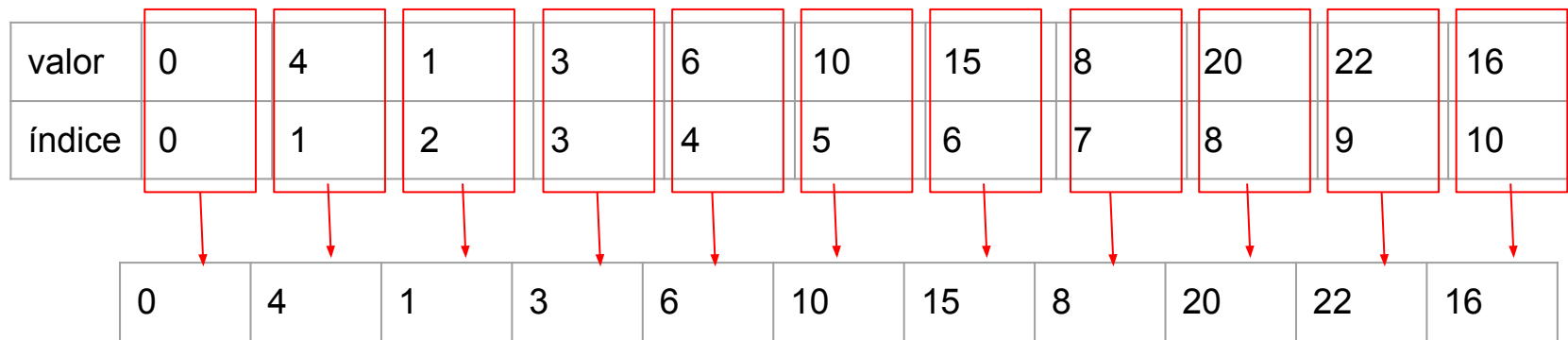
1	10	20	
---	----	----	--





# Execução do Algoritmo

Incremento 1



# Execução do Algoritmo

valor	0	1	3	4	6	8	10	15	16	20	22
índice	0	1	2	3	4	5	6	7	8	9	10

0	1	3	4	6	8	10	15	16	20	22
---	---	---	---	---	---	----	----	----	----	----

# Implementação

- Como implementar o algoritmo?

# Implementação

- Como implementar o algoritmo?
- Duas maneiras podem ser usadas:
  - A maneira mais simples de se implementar o algoritmo é usando um vetor auxiliar
  - Outra maneira é fazer as ordenações dos subvetores dentro do próprio vetor (*in situ*);
    - Essa maneira em geral é implementada alterando a ordem com que os subvetores são ordenados, dessa forma

# Implementação

- Com vetor auxiliar

```
void shell(int v[],int n){  
}
```

# Implementação

- Com vetor auxiliar

```
void shell(int v[],int n){
    int i,j,k,l,gap=1,*aux,val;
    aux=(int *)malloc(n*sizeof(int));
    while(gap<n) gap=3*gap+1;
    do{
        gap=(gap-1)/3;
        for(i=0;i<gap;i++){
            k=0;
            for(j=i;j<n;j+=gap){
                aux[k]=v[j];
                k++;
            }
            insertion(aux,k);
            k=0;
            for(j=i;j<n;j+=gap){
                v[j]=aux[k];
                k++;
            }
        }
    }while(gap>1);
    free(aux);
}
```

# Implementação

- *In situ*

```
void shell(int v[],int n){  
}
```

# Implementação

- *In situ*

```
void shell(int v[],int n){
    int gap, i, j,aux;
    gap=1;
    while(gap < n) gap=gap*3+1;
    do{
        gap=(gap-1)/3;
        for(i=gap;i<n;i++){
            j=i;
            aux=v[i];
            while(j>=gap && v[j-gap] > aux){
                v[j]=v[j-gap];
                j-=gap;
                comp++;
            }
            v[j]=aux;
        }
    }while(gap>1);
}
```



# Ordenação por incremento decrescente

- O custo da ordenação por incremento decrescente num vetor de  $n$  elementos depende, além da ordem dos valores, dos valores de incremento;
  - Sabe-se que cada incremento não deve ser múltiplo do anterior;
- Existem duas conjecturas para a sequência de incrementos  $G_{i+1} = 3G_i + 1$ ;
  - A conjectura 1 diz que o tempo é  $O(n^{1,25})$ ;
  - A conjectura 2 diz que o tempo é  $O(n (\ln n)^2)$ ;

# Ordenação por incremento decrescente

- Vantagens:
  - Simples de implementar;
    - Requer pouco código;
- Desvantagem:
  - Tempo de execução é sensível ao arquivo;
  - O tempo de execução depende dos valores de gap;

# Exercícios

1. Aplique a ordenação de Shell, destacando todos subvetores e mostrando o vetor todo após a ordenação de cada subvetor, para os seguintes conjuntos de valores:
  - a. 1, 6, 5, 25, 15, 3, 9, 7 com incrementos 4, 2 e 1;
  - b. 1, 0, 9, 13, 22, 7, 12, 14, 3, 8, 4, 30 com incrementos 5,3 e 1

# Referências

Robert Sedgewick. 1990. *Algorithms in C*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Wirth, Niklaus. Algoritmos e Estruturas de Dados, Editora LTC.

Drozdek, Adam. Estrutura de dados e algoritmos em C++, Thomson Pioneira.