CENTRO DE CIÊNCIA E TECNOLOGIA
LABORATÓRIO DE CIÊNCIAS MATEMÁTICAS
UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE

Algoritmos de Ordenação
Shellsort

Disciplina: Estrutura de Dados II

Prof. Fermín Alfredo Tang Montané

Curso: Ciência da Computação

Algoritmo ShellSort

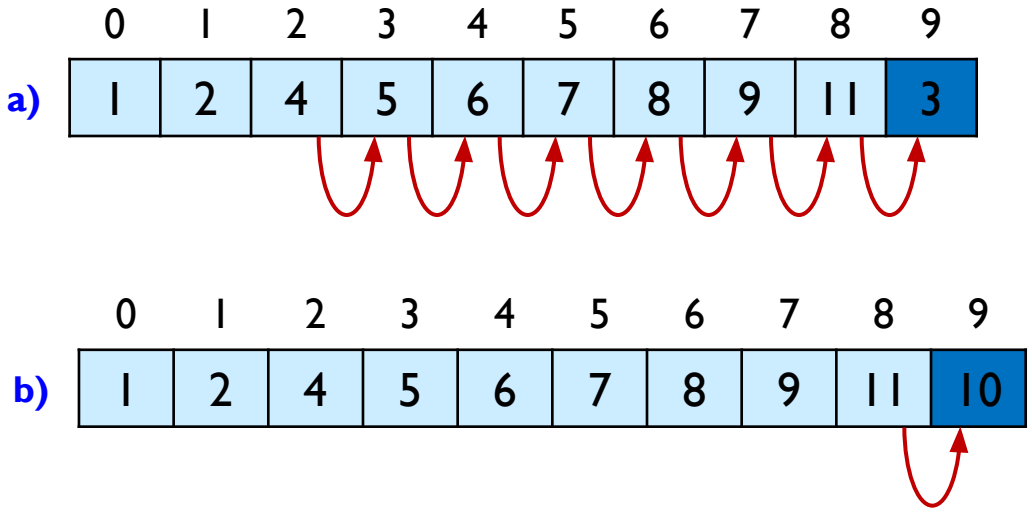
Um Algoritmo de Inserção Otimizado

- A ordenação Shell é nomeada segundo Donald L. Shell, o cientista que a descobriu em 1959.
- É baseada na ordenação por inserção, mas adiciona um novo recurso que melhora muito o desempenho da ordenação por inserção.
- A ordenação é boa para vetores de tamanho médio, talvez com até alguns milhares de elementos, dependendo da implementação.
- O desempenho de pior caso não é muito pior que o desempenho médio.

Algoritmo ShellSort

O ponto fraco do Método de Inserção

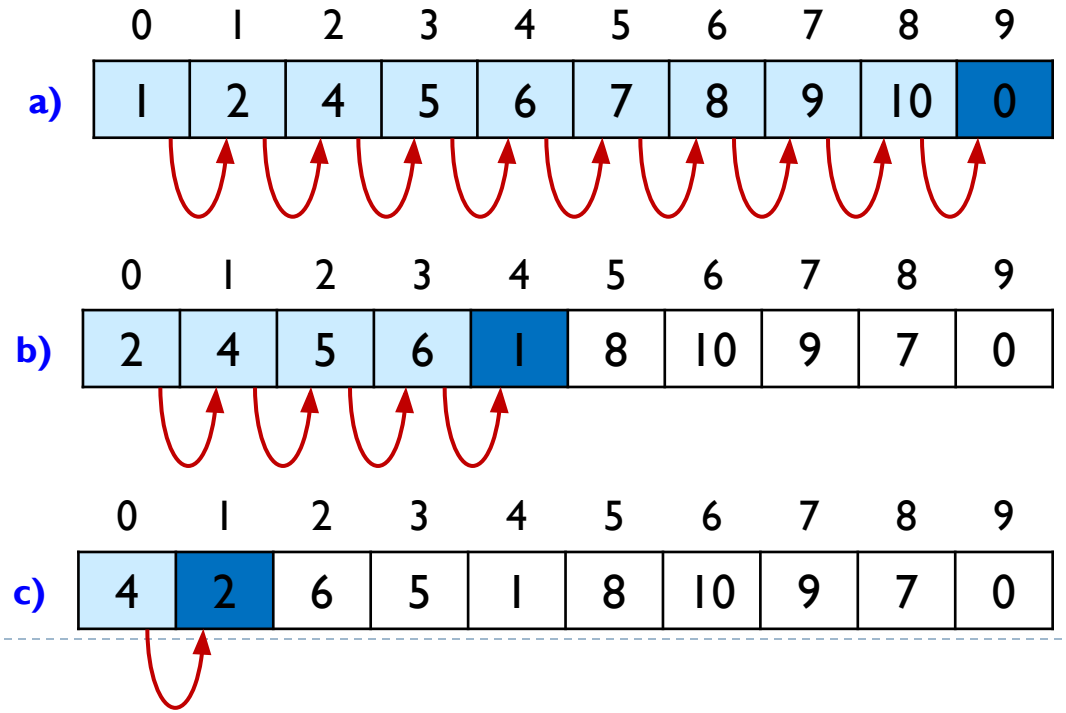
- O método de inserção procura a posição correta para a chave de um determinado elemento e realiza deslocamentos para viabilizar a inserção.
- Dependendo do valor da chave, a posição para inserção pode ficar **muito distante** da posição atual do elemento, resultando em um grande número de deslocamentos.
- Quando a posição de inserção fica **próxima** da posição atual do elemento, o número de deslocamentos é pequeno, reduzindo-se o tempo de execução. O algoritmo Shellsort aproveita este fato.



Algoritmo ShellSort

O ponto fraco do Método de Inserção

- No pior caso:
  - O último elemento de um vetor de tamanho n, precisará $n-1$ deslocamentos para ser inserido na posição certa.
  - Já um elemento na metade do mesmo vetor precisará aproximadamente $n/2$ deslocamentos.
  - O segundo elemento, precisará de apenas 1 deslocamento.
- O número total de deslocamentos é de ordem $O(n^2)$.



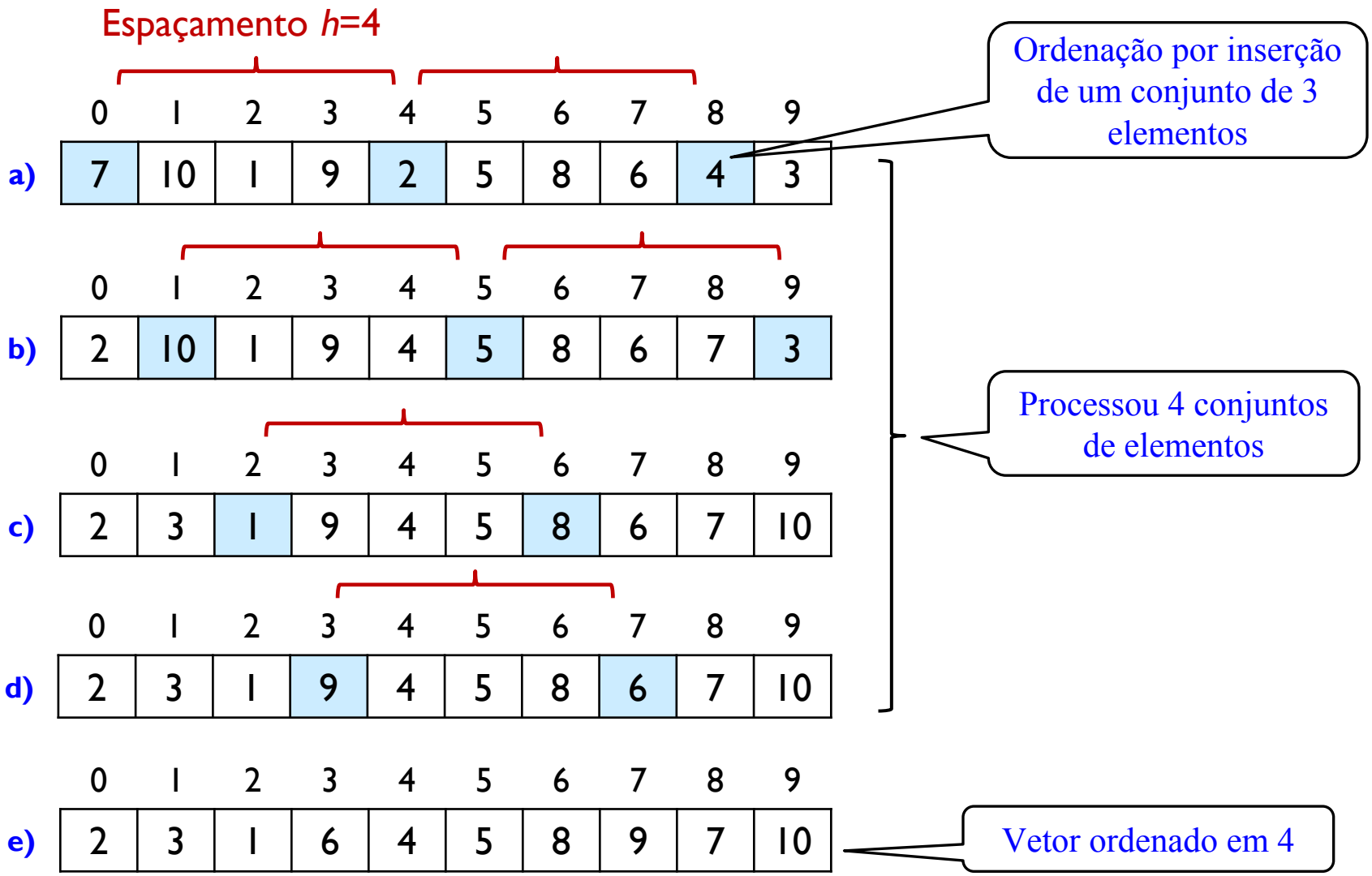
Algoritmo ShellSort

O principio de ordenação em h

- A ordenação Shell consegue evitar grandes deslocamentos, **ordenando por inserção**, primeiro, aqueles elementos com posições **muitos espaçadas** entre si, para isso usa uma **medida de espaçamento**, denotada com h.
- Depois deles serem ordenados, o método reduz essa medida de espaçamento e ordena os elementos com **menor espaçamento (menor valor de h)** e assim sucessivamente.
- Na última etapa o espaçamento será $h = 1$, e o vetor estará quase ordenado, o que significa que a ordenação por inserção, neste caso, terá desempenho próximo de $O(n)$.
- Este método ordena parcialmente os elementos do vetor, formando conjuntos de elementos ordenados e entrelaçados, de maneira que nenhum elemento se encontre muito distante da sua posição final no vetor ordenado.
- Inicialmente, o espaçamento h deve ser grande para reduzir o esforço do método de inserção, fazendo que no final seja próximo de $O(n)$.

Algoritmo ShellSort

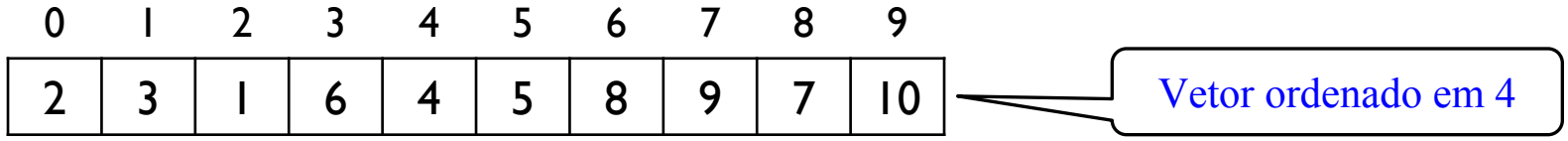
Exemplo1 – Ordenação em 4



Algoritmo ShellSort

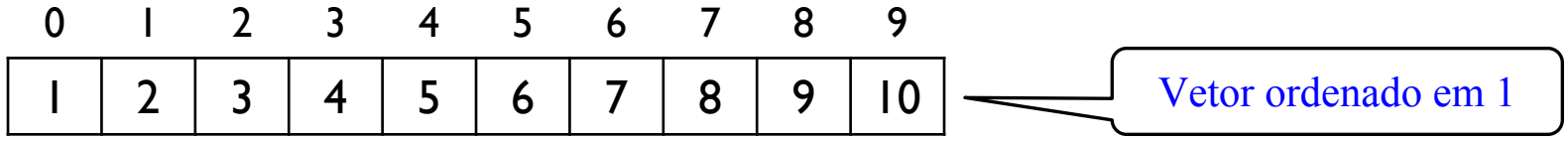
Exemplo 1 – Ordenação em 4

e)



- Depois da ordenação em 4, o vetor deverá ser ordenado em 1 usando a ordenação por inserção comum.

f)

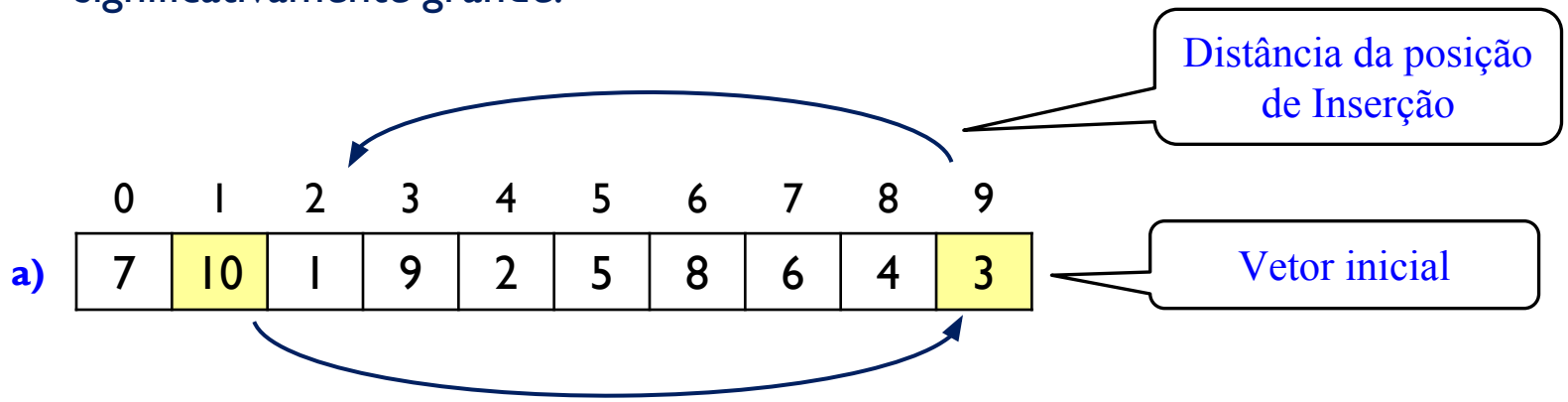


- A combinação da ordenação em 4 e a ordenação em 1 é muito mais rápida que a ordenação por inserção comum, porque a ordenação em 4 reduz a distancia de cada elemento da sua posição final no vetor ordenado, minimizando o número de deslocamentos.

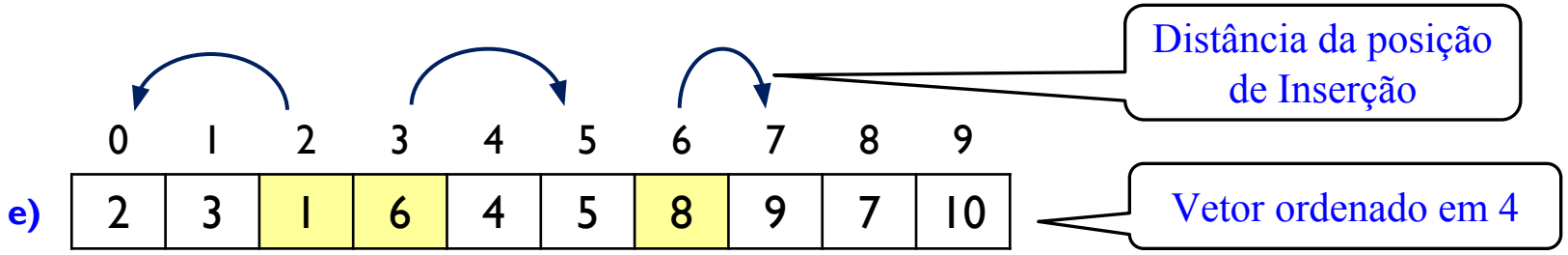
Algoritmo ShellSort

Exemplo1 – Reduzindo o esforço de deslocamento

- Inicialmente, a distancia de cada elemento para a sua posição ordenada é significativamente grande.



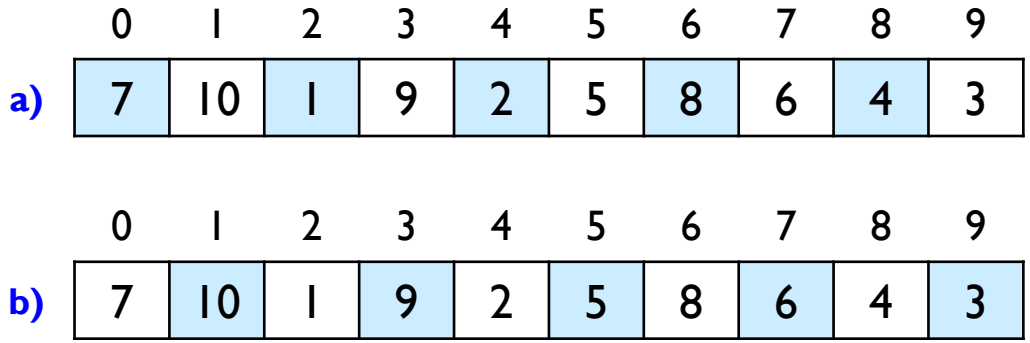
- Depois da ordenação em 4, observa-se que a distancia de cada elemento para a sua posição ordenada ficou bastante reduzida.



Algoritmo ShellSort

Subconjuntos de Elementos

- Observe que ao definir um espaçamento de h posições são formados h subconjuntos de elementos.

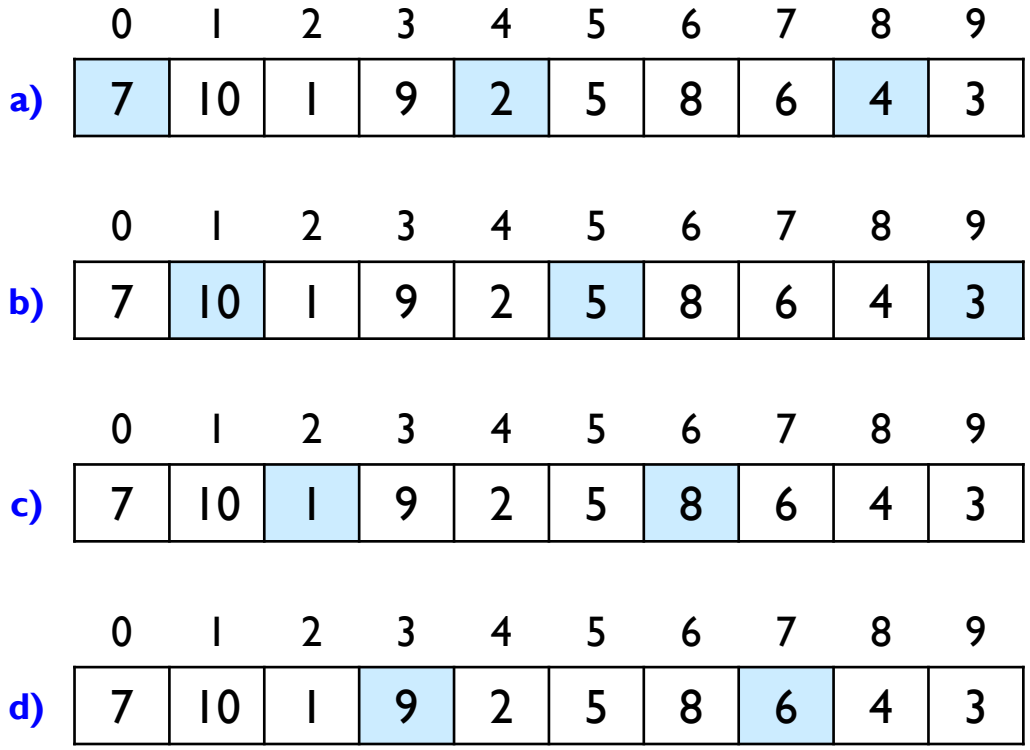


O espaçamento $h = 2$ define 2 Subconjuntos

Algoritmo ShellSort

Subconjuntos de Elementos

- Observe que ao definir um espaçamento de h posições são formados h subconjuntos de elementos.



O espaçamento $h = 4$ define 4 Subconjuntos

Algoritmo ShellSort

Critério para definir a sequência de espaçamentos

- A sequência de espaçamentos mais utilizada no algoritmo de Shell é a chamada **Sequência de Knuth.**
- Para aplicar o algoritmo Shell, aplica-se primeiro a formula de Knuth para descobrir o **maior valor** menor que o tamanho do vetor.
- Logo, calculam-se os valores de espaçamento mediante a função inversa de Knuth.

Incremento	Função de Knuth	Função Inversa de Knuth
h	3**h*+1	(*h*-1)/3
1	4	
4	13	1
13	40	4
40	121	13
121	364	40
364	1093	121
1093	3280	364

Vetor 1000 Elementos

Algoritmo ShellSort

Modificação do Algoritmo de inserção

- Considere o algoritmo de inserção original (Versão2).
- Neste algoritmo, podemos considerar que o espaçamento entre elementos é igual a um $(h = 1)$. Tanto as comparações quanto os deslocamentos acontecem entre elementos adjacentes, com espaçamento igual a um.

```
INSERTION_SORT (V)                                  /* Ordena o arranjo V[n] */
1.      para j ← 1 até (n − 1) fazer
2.          key ← V[ j ];                           /* Variável temporária Key */
3.          i ← j;
4.          enquanto (i >0) e (key<=V[i-1]) fazer
5.              V[i] ← V[ i -1];                    /* Deslocamento de elementos */
6.              i ← i −1;
7.          fim-enquanto
8.          V[i] ← key;                             /* Inserção da Key */
9.      fim-para
```

Espaçamento h=1!

Algoritmo ShellSort

Descrição do Algoritmo

- O algoritmo Shellsort realiza a ordenação de subconjuntos de elementos com espaçamento h, definido inicialmente pela sequência de Knuth. Para isso utiliza um algoritmo de inserção modificado para incorporar esse espaçamento.
- O valor do espaçamento inicial h é reduzido, nas iterações, até atingir o valor $(h = 1)$.

```
SHELLSORT (V)                                   /* Ordena o arranjo V[N] */
1.  h=1;
2.  enquanto (h<= (n − 1)/3) fazer
3.      h=3*h+1;                                /* Valor inicial de h */
4.  fim-enquanto
5.  enquanto (h>0) fazer
6.      para j ← h até (n − 1) fazer
7.          key ← V[ j ];                       /* Variável temporária Key */
8.          i ← j;
9.          enquanto (i-h>=0) e (key<=V[i-h]) fazer
10.             V[i] ← V[ i-h];                 /* Deslocamento de elementos */
11.             i ← i –h;
12.         fim-enquanto
13.         V[i] ← key;                         /* Inserção da Key */
14.     fim-para
15.     h=(h-1)/3;                              /* Diminui h */
16. fim-enquanto
```

Valor inicial de h

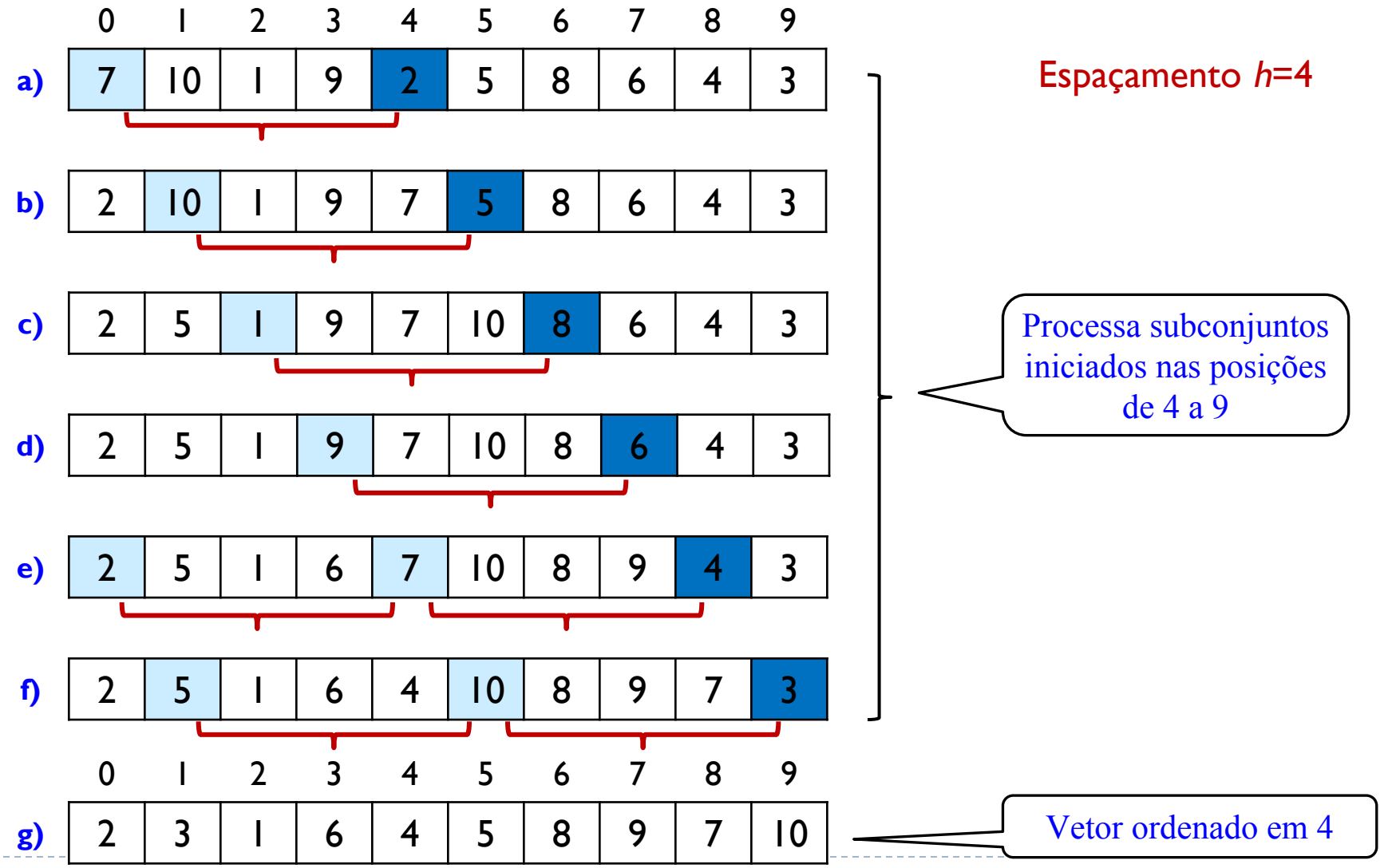
Algoritmo de Inserção Modificado

Espaçamento h!

Decremento de h

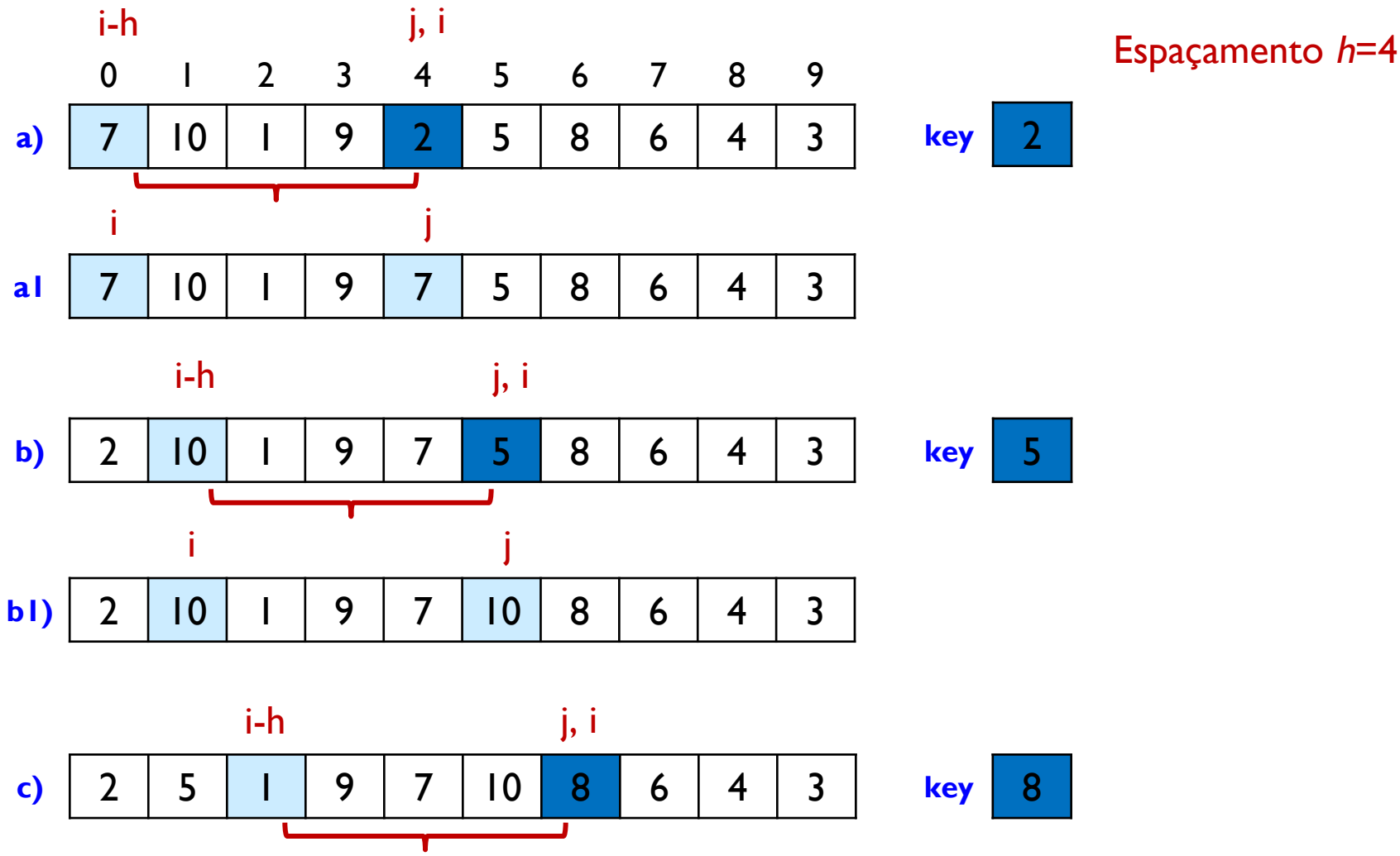
Algoritmo ShellSort

Exemplo2 – Ordenação em 4 - Subconjuntos



Algoritmo ShellSort

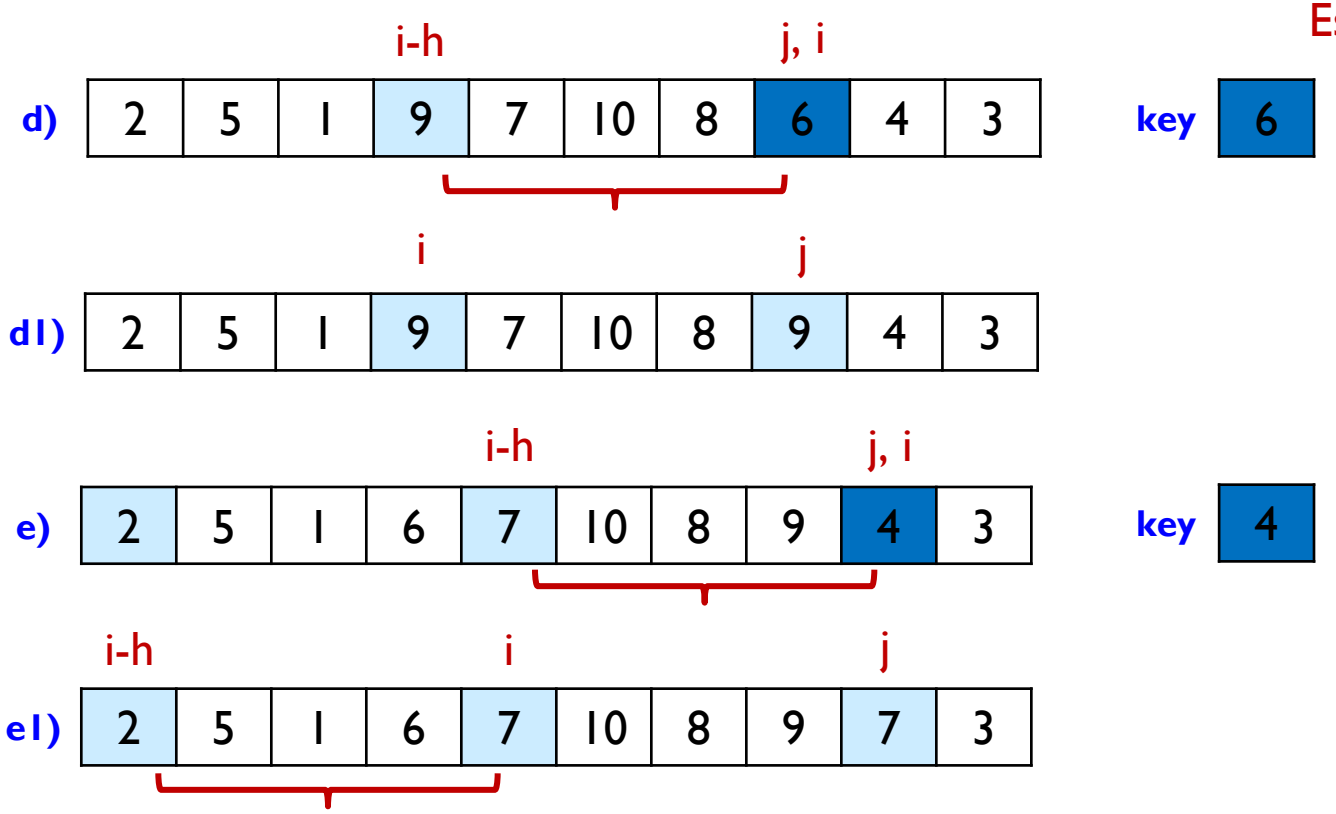
Exemplo2 – Ordenação em 4 - Iterações



Algoritmo ShellSort

Exemplo2 – Ordenação em 4 - Iterações

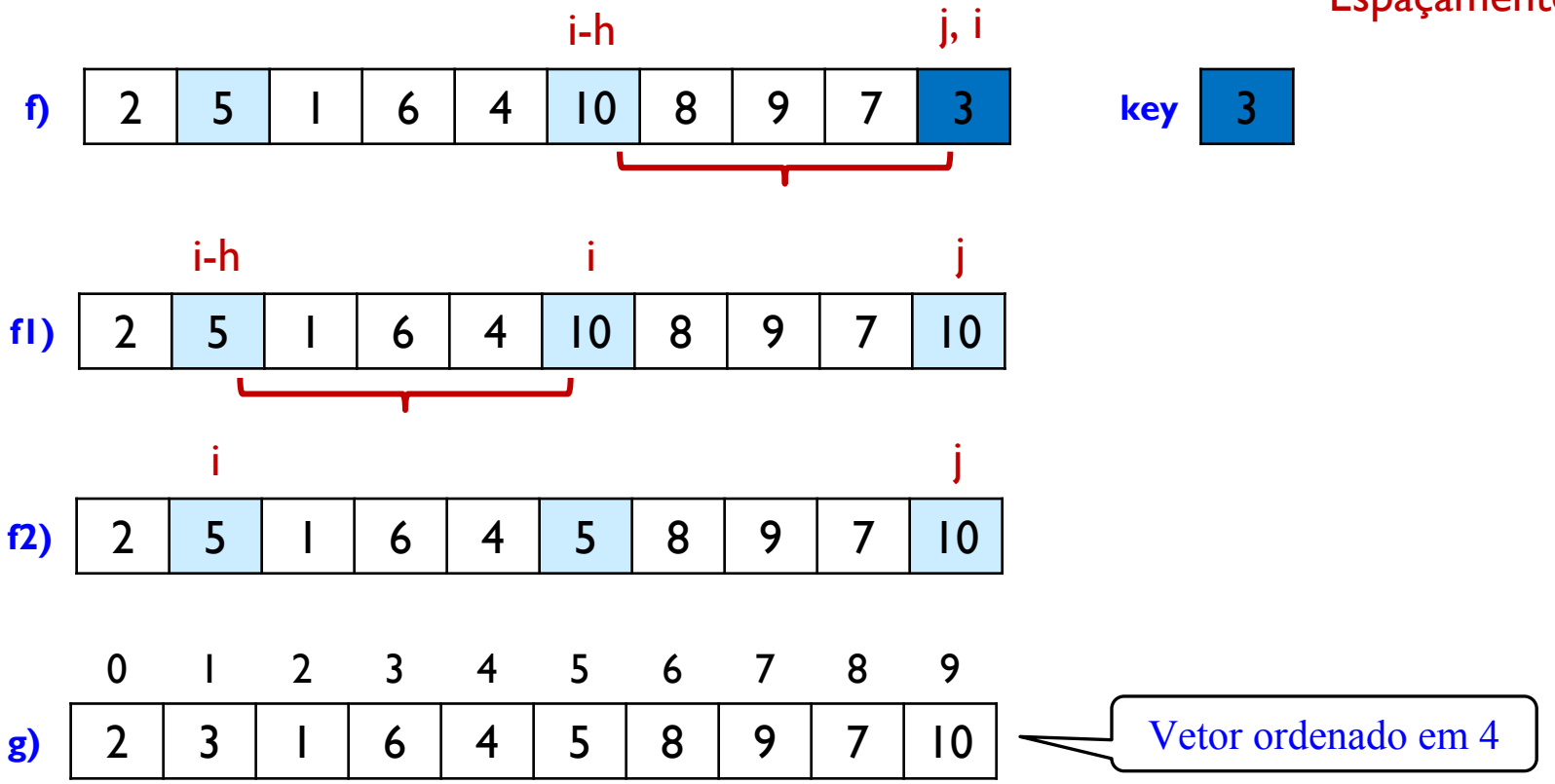
Espaçamento *h*=4



Algoritmo ShellSort

Exemplo2 – Ordenação em 4 - Iterações

Espaçamento *h*=4



Algoritmo ShellSort

Complexidade do Algoritmo

- A análise da complexidade do algoritmo Shellsort não é uma tarefa simples. Esta análise foi realizada para alguns casos especiais.
- No entanto, algumas estimativas sobre a complexidade do tempo foram realizadas com base em experimentos.
- Estas estimativas mostraram que o algoritmo tem um desempenho melhor ao algoritmo de inserção tradicional.

Tamanho	Método de Inserção	Método Shellsort	Método Shellsort
n	$O(n^2)$	Est. Pessimista $O(n^{3/2})$	Est. Otimista $O(n^{7/6})$
10	100	32	15
100	10.000	1.000	216
1.000	1.000.000	31.623	3.162
10.000	100.000.000	1.000.000	46.416

Referências

- Thomas **Cormen**, Charles **Leiserson,** et al.**.** Algoritmos. Teoria e Prática. 2ª Edição. 2002.
- Robert **Lafore**. Estruturas de Dados e Algoritmos em Java. Editora Ciencia Moderna. 2ª Edição. 2004.