① 7ª Aula

Limite inferior para ordenação

Um algoritmo de ordenação baseia-se em comparações se o fluxo do algoritmo para uma entrada de tamanho n depende apenas de comparações do tipo $a_i \leq a_j$.

Todos os algoritmos de ordenação que estudamos até o momento baseiam-se em comparações.

② **Teorema**

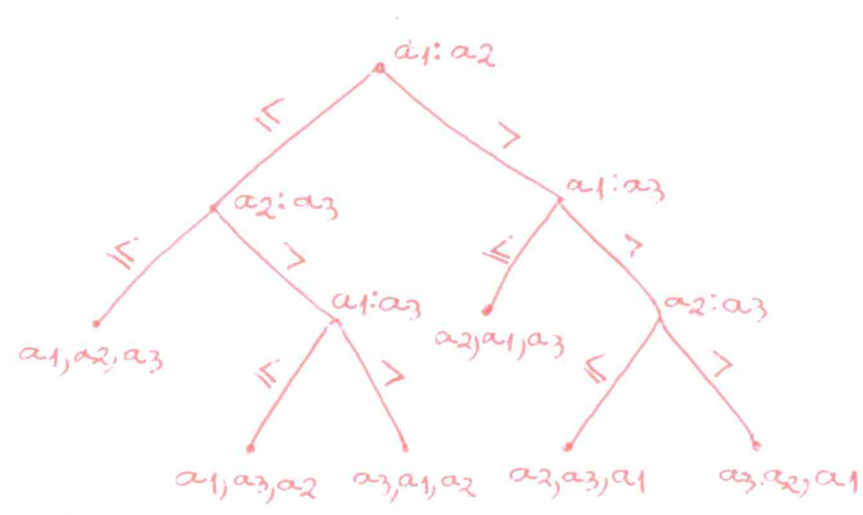
Qualquer algoritmo de ordenação por comparação exige $\Omega(n \lg n)$ comparações no pior caso.

Prova:

Árvore de decisão

a_1	a_2	a_3

Insertion-Sort



③ Dado um vetor com n elementos, o vetor ordenado pode ser qualquer uma das $n!$ permutações.

Um algoritmo de ordenação efetua um número de comparações equivalente a altura da árvore com $n!$ folhas.

Uma árvore binária com altura h tem no máximo 2^h folhas.

④ Logo, temos:

$$n! \leq 2^h \Rightarrow \lg(n!) \leq h \quad (a \leq b \Rightarrow \lg a \leq \lg b)$$

$$h \geq \lg(n!) \Rightarrow h = \Omega(n \lg n)$$

$$\lg(n!) \geq \frac{n}{4} \lg n, \quad n \geq 16.$$

Portanto, $h = \Omega(n \lg n)$

⑤ Counting-Sort (Ordenação por contagem)

- Todos os elementos do vetor são inteiros i no intervalo $[0..K]$
- Ordena-se o vetor contando quantos elementos são menores do que i.
- Utiliza dois vetores auxiliares para ordenar

⑥

```
Counting-Sort (A, B, n, K)
for i = 0 to K do                                   Θ(K)
    C[i] = 0                                        Θ(K)
for i = 1 to n do                                   Θ(n)
    C[A[i]] = C[A[i]] + 1                           Θ(n)
// C[i] contém o número de elementos iguais a i
for i = 1 to K do                                   Θ(K)
    C[i] = C[i] + C[i-1]                            Θ(K)
// C[i] contém número de elementos ≤ i
for j = n downto 1 do                               Θ(n)
    B[C[A[j]]] = A[j]                               Θ(n)
    C[A[j] = C[A[j]]-1                              Θ(n)
```

7 Complexidade do Counting-Sort

Tempo de execução: $T(n) = \Theta(5n) + \Theta(4K)$

$T(n) = \Theta(n + K)$

Se $K = O(n)$, então $T(n) = \Theta(n)$

Note que o algoritmo Counting-Sort não realiza comparações do tipo $a_i \leq a_j$

8 ~~[illegible]~~

Classificação de algoritmos de ordenação

- Estabilidade: Um algoritmo é estável se elementos idênticos ocorrem no vetor ordenado na mesma ordem que foram recebidos como entrada
- Localidade: Um algoritmo é local se a quantidade de memória adicional requerida é constante

Note que Counting-Sort é estável mas não local.

9 Radix-Sort (Ordenação digital)

- Números a serem ordenados tem todos d dígitos
- Inicia-se pelo dígito menos significativo

Radix-Sort(A, n, d)

1. for i = 1 to d do
2. ordena A[1..n] pelo i-ésimo dígito com um algoritmo estável.

10 Exemplo

329	72$\boxed{0}$	7$\boxed{2}$0	$\boxed{3}$29
457	35$\boxed{5}$	3$\boxed{2}$9	$\boxed{3}$55
657	43$\boxed{6}$	4$\boxed{3}$6	$\boxed{4}$36
839	45$\boxed{7}$	8$\boxed{3}$9	$\boxed{4}$57
436	65$\boxed{7}$	3$\boxed{5}$5	$\boxed{6}$57
720	32$\boxed{9}$	4$\boxed{5}$7	$\boxed{7}$20
355	83$\boxed{9}$	6$\boxed{5}$7	$\boxed{8}$39
	dígito 1	dígito 2	dígito 3

Elementos ordenados

(11) Complexidade do Radix-Sort

Usando Counting-Sort, a complexidade de pior caso é $\Theta(d(n+k))$.

Se $k = O(n)$ e $d = O(1)$, então $T(n) = \Theta(n)$

Note que se for utilizado, por exemplo, o Insertion-Sort como algoritmo estável a complexidade de tempo não será linear.

(12) Bucket-Sort (Ordenação por "balde")

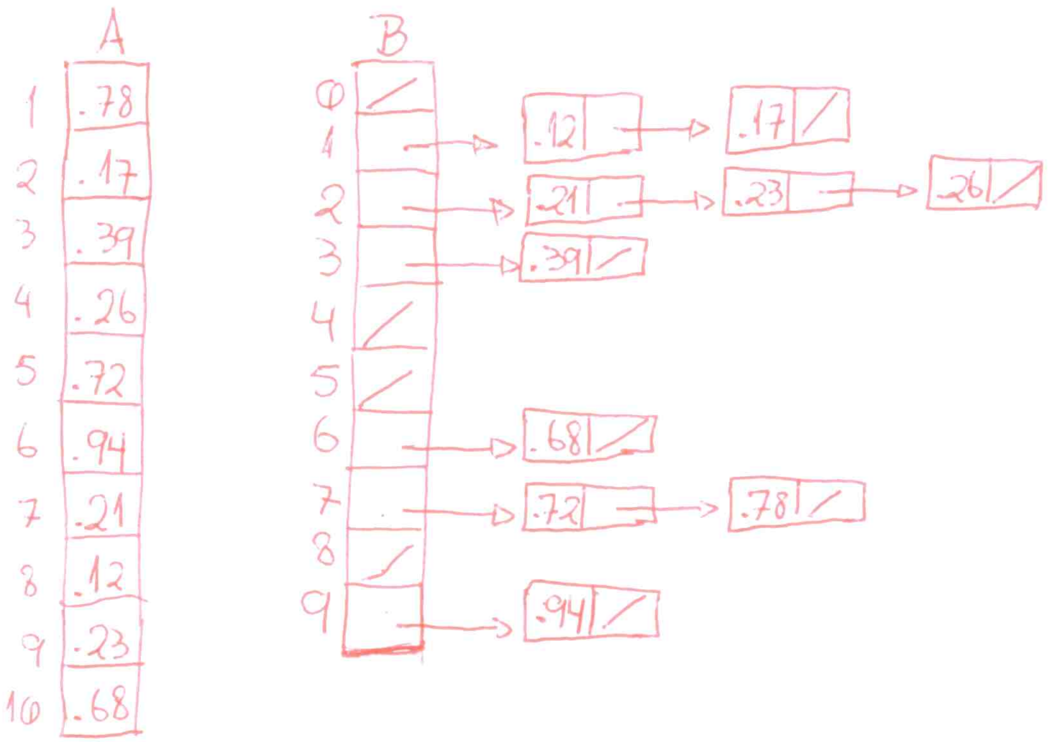
- Os n elementos do vetor são reais distribuídos uniformemente em $[0, 1)$
- Dividir o intervalo $[0, 1)$ em n buckets de mesmo tamanho e distribuir os n elementos nos respectivos buckets
- Cada bucket é ordenado por um método qualquer
- Por fim, os buckets ordenados são concatenados em ordem crescente

(13)

Bucket-Sort (A, n)

1. for $i = 0$ to $n-1$ do
2. $B[i]$ = NULL
3. for $i = 1$ to n do
4.   insira $A[i]$ na lista $B[\lfloor n A[i] \rfloor]$
5. for $i = 0$ to $n-1$ do
6.   Insertion-Sort ($B[i]$)
7. Concatene as listas $B[0], B[1], \ldots, B[n-1]$

(14) Exemplo



15 Complexidade do Bucket-Sort

- Pior caso: Supondo Insertion-Sort para ordenar as listas.

  $\Theta(n^2)$

- Caso médio: Número de elementos em cada lista é $\Theta(1)$, e o tempo esperado para ordenar uma lista $B[i]$ também é $\Theta(1)$.

  $\Theta(n)$

- Melhor caso:

  $\Theta(n)$

- Complexidade de espaço:

  $\Theta(n)$