AULA5: Exercício teórico métodos de ordenação lineares

Aluno: Gian Franco Joel Condori Luna

September 10, 2024

Exercices

1 (0,3) Ilustre a ordenação por contagem sobre o vetor A = [6, 0, 2, 0, 1, 3 , 4, 6, 1, 3, 2].

Solução:

$C =$

2	2	2	2	1	0	2

$C =$

2	4	6	8	9	9	11

n = 11

$A =$

6	0	2	0	1	3	4	6	1	3	2

$C =$

2	4	6	8	9	9	11

$B =$

					2					

$C =$

2	4	5	8	9	9	11

n = 10

$A =$

6	0	2	0	1	3	4	6	1	3	2

$C =$

2	4	5	8	9	9	11

$B =$

					2		3			

$C =$

2	4	5	7	9	9	11

n = 9

$A =$

6	0	2	0	1	3	4	6	1	3	2

$C =$

2	4	5	7	9	9	11

$B =$

$C =$

2	3	5	7	9	9	11

n = 8

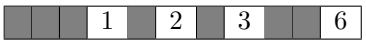
$A =$

6	0	2	0	1	3	4	6	1	3	2

$C =$

2	3	5	7	9	9	11

$B =$



$C =$

2	3	5	7	9	9	10

n = 7

$A =$

6	0	2	0	1	3	4	6	1	3	2

$C =$

2	3	5	7	9	9	10

$B =$

$C =$

2	3	5	7	8	9	10

n = 6

$A =$ | 6 | 0 | 2 | 0 | 1 | 3 | 4 | 6 | 1 | 3 | 2 |

$C =$ | 2 | 3 | 5 | 7 | 8 | 9 | 10 |

$B =$ | | | | 1 | | 2 | 3 | 3 | 4 | | 6 |

$C =$ | 2 | 3 | 5 | 6 | 8 | 9 | 10 |

n = 5

$A =$ | 6 | 0 | 2 | 0 | 1 | 3 | 4 | 6 | 1 | 3 | 2 |

$C =$ | 2 | 3 | 5 | 6 | 8 | 9 | 10 |

$B =$ | | | 1 | 1 | | 2 | 3 | 3 | 4 | | 6 |

$C =$ | 2 | 2 | 5 | 6 | 8 | 9 | 10 |

n = 4

$A =$ | 6 | 0 | 2 | 0 | 1 | 3 | 4 | 6 | 1 | 3 | 2 |

$C =$ | 2 | 2 | 5 | 6 | 8 | 9 | 10 |

$B =$ | | 0 | 1 | 1 | | 2 | 3 | 3 | 4 | | 6 |

$C =$ | 1 | 2 | 5 | 6 | 8 | 9 | 10 |

n = 3

$A =$ | 6 | 0 | 2 | 0 | 1 | 3 | 4 | 6 | 1 | 3 | 2 |

$C =$ | 1 | 2 | 5 | 6 | 8 | 9 | 10 |

$B =$ | | 0 | 1 | 1 | 2 | 2 | 3 | 3 | 4 | | 6 |

$C =$ | 1 | 2 | 4 | 6 | 8 | 9 | 10 |

n = 2

$A =$

6	0	2	0	1	3	4	6	1	3	2

$C =$

1	2	4	6	8	9	10

$B =$

0	0	1	1	2	2	3	3	4		6

$C =$

0	2	4	6	8	9	10

n = 1

$A =$

6	0	2	0	1	3	4	6	1	3	2

$C =$

0	2	4	6	8	9	10

$B =$

0	0	1	1	2	2	3	3	4	6	6

$C =$

0	2	4	6	8	9	9

Resposta

$B =$

0	0	1	1	2	2	3	3	4	6	6

2 (0,3) Ilustre a ordenação radix-sort sobre a seguinte lista de palavras: cow, dog, sea, rug, row, mob, box, tab, bar, ear, tar, dig, big, tea, now, fox.

Solução:

Ordenada pelo 3er digito

cow, dog, sea, rug, row, mob, box, tab,
bar, ear, tar, dig, big, tea, now, fox

$C =$

a	tea, sea
b	tab, mob
c	-
d	-
e	-
f	-
g	-
h	big, dig, rug, dog
i	-
j	-
k	-
l	-
m	-
n	-
o	-
p	-
q	-
r	tar, ear, bar
s	-
t	-
u	-
v	-
w	now, row, wow
x	fox, box
y	-
z	-

sea, tea, mob, tab, dog, rug, dig, big,
bar, ear, tar, cow, row, now, box, fox

Ordenada pelo 2do digito

sea, tea, mob, tab, dog, rug, dig, big,
bar, ear, tar, cow, row, now, box, fox

$C =$

a	bar, ear, tar, tab
b	-
c	-
d	-
e	tea, sea
f	-
g	-
h	-
i	dig, big
j	-
k	-
l	-
m	-
n	-
o	box, fox, cow, row, now, dog, mob
p	-
q	-
r	-
s	-
t	-
u	rug
v	-
w	-
x	-
y	-
z	-

tab, tar, ear, bar, sea, tea, big, dig,
mob, dog, now, row, cow, fox, box, rug

Ordenada pelo 1er digito

tab, tar, ear, bar, sea, tea, big, dig,
mob, dog, now, row, cow, fox, box, rug

$C =$

a	-
b	box, big, bar
c	cow
d	dog, dig
e	ear
f	fox
g	-
h	-
i	-
j	-
k	-
l	-
m	-
n	mob
o	now
p	-
q	-
r	-
s	rug, row
t	sea
u	tea, tar, tab
v	-
w	-
x	-
y	-
z	-
	-

Resposta

bar, big, box, cow, dig, dog, ear, fox, mob, now, row, rug, sea, tab, tar, tea

3 (0,4) Dentre todos os algoritmos de ordenação vistos até o momento, quais seriam os mais eficientes (considerando tempo de execução e uso de memória) para cada caso, explique sua escolha:

3.a. Uma lista ordenada em ordem ascendente

De acordo com os testes que realizei em uma pratica anterior da disciplina para um array ordenado ascendente de 1000 elementos, o algoritmo que me deu melhor tempo en segundos foi o Shellsort, mas se olharmos para o lado da complexidade e a teoria nos disser que o algoritmo Insertion Sort é o mais eficiente neste caso, com complexidade $O(n)$, pois só verificaria se a lista já

está ordenada. Outros algoritmos, como Quicksort ou Mergesort, mantêm sua complexidade $O(nlogn)$.

3.b. Uma lista ordenada em ordem descendente

De acordo com os testes que realizei em uma pratica anterior da disciplina para um array ordenado descendente de 1000 elementos, o algoritmo que me deu melhor tempo en segundos foi o Shellsort, mas se olharmos só a complexidade e a teoria o algoritmo Quicksort é eficiente com $O(nlogn)$ na média, mas pode ser pior em alguns casos ($O(n^2)$). Tambem se tem o Heapsort que a diferença do Quicksort, ele sempre tem $O(nlogn)$ no pior caso.

3.c. Uma lista com os valores desordenados aleatoriamente

Quicksort é um dos mais eficientes na média, com $O(nlogn)$, seguido por Heapsort e Mergesort, ambos com a mesma complexidade.

3.d. Elementos com valores iguais não sejam trocados de ordem (estável).

Mergesort e Counting Sort são estáveis, ou seja, mantêm a ordem relativa dos elementos com valores iguais, com complexidades $O(nlogn)$ e $O(n+k)$, respectivamente.