Lista de Exercícios de Ordenação

Consultar arquivos:

cap4.pdf

EstrDadosAlgoritmos-ClaudioEsperanca.pdf

1) Comparação de complexidade entre os métodos:

Métodos de Ordenação	Complexidade
Inserção	**$O(n^2)$**
Seleção	**$O(n^2)$**
Bolha (Bubble sort)	**$O(n^2)$**
Shellsort	**O(n * log n)**
Quicksort	**O(n * log n)**
Heapsort	**O(n * log n)**
MergeSort	**O(n * log n)**

2) O que significa “Ordenar” e qual seu objetivo:

Consultar: cap4.pdf

3) Quando um algoritmo é estável?

Consultar: cap4.pdf

Resposta:

- Um método de ordenação é **estável** se a ordem relativa dos itens com chaves iguais não se altera durante a ordenação.

Um algoritmo de ordenação diz-se **estável** se preserva a ordem de registros de chaves iguais. Isto é, se tais registros aparecem na sequência ordenada na mesma ordem em que estão na sequência inicial.

Exemplo:

Por exemplo, um algoritmo estável ordenando a sequência de números (chaves) com letras associadas (registros):

```
3[a], 2[b], 2[c], 1[d]
```

obrigatoriamente retornará:

```
1[d], 2[b], 2[c], 3[a]
```

enquanto algoritmos instáveis sujeitam os elementos associados aos objetos a serem ordenados a mudanças:

```
1[d], 2[c], 2[b], 3[a]
```

Observe aqui acima que 2[c] trocou de lugar com 2[b]. No algoritmo estável isto não acontece.

4) Diferencie método de ordenação interna de externa.

Consultar: cap4.pdf

5) Na classificação dos métodos de ordenação interna temos: métodos simples e métodos eficientes. Quais suas características?

Consultar: cap4.pdf

6) Como funciona a ordenação por seleção

Consultar: cap4.pdf

7) Execute o algoritmo de ordenação por seleção nas letras abaixo:

Consultar: cap4.pdf

a) O R D E N A
b) J A C I N T O

8) Quais as vantagens e desvantagens da ordenação por seleção?

Consultar: cap4.pdf

9) Como funciona a ordenação por inserção?

Consultar: cap4.pdf

10) Execute o algoritmo de ordenação por inserção nas letras abaixo:

Consultar: cap4.pdf

a) O R D E N A
b) J A C I N T O

11) Sobre a ordenação por inserção responda:

Consultar: cap4.pdf

a) quando ocorre o máximo e o mínimo de comparações
b) quando se deve usar este método
c) é estável ou instável
d) por que é bom usá-lo para se adicionar poucos elementos a um arquivo quase ordenado

12) Como funciona a ordenação Shellsort?

Consultar: cap4.pdf

13) Execute o algoritmo de ordenação Shellsort nas letras abaixo:

Consultar: cap4.pdf

a) O R D E N A

Considerar na primeira passada, h = 4.

Considerar na segunda passada, h = 2.

Considerar na terceira passada, h = 1.

14) Quais as vantagens e desvantagens da ordenação Shellsort?

Consultar: cap4.pdf

15) Como funciona o algoritmo de ordenação Quicksort?

Consultar: cap4.pdf

16) Como funciona o particionamento no Quicksort?

Consultar: cap4.pdf

17) Execute o algoritmo Quicksort nas letras abaixo e descreva as chamadas as funções envolvidas:

Consultar: cap4.pdf

a) O R D E N A

18) Quando ocorre o pior caso no Quicksort?

Consultar: cap4.pdf

19) Cite as vantagens e desvantagens do Quicksort

Consultar: cap4.pdf

20) Como funciona o algoritmo heapsort

Consultar: cap4.pdf

21) Aplique o heapsort nas letras abaixo:

a) O R D E N A

Consultar: cap4.pdf

22) Vantagens e desvantagens do Heapsort

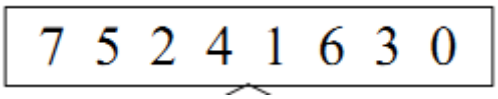
Consultar: cap4.pdf

24) Como funciona o algoritmo MergeSort

Consultar: EstrDadosAlgoritmos-ClaudioEsperanca.pdf

25) Aplique o algoritmo MergeSort no vetor abaixo e descreva a chamada as funções envolvidas.

Consultar: EstrDadosAlgoritmos-ClaudioEsperanca.pdf

a) 

26) Comparação entre os Métodos:

Consultar: cap4.pdf

Comparação entre os Métodos

Observações sobre os métodos:

1. Shellsort, Quicksort e Heapsort têm a mesma ordem de grandeza.
2. O Quicksort é o mais rápido para todos os tamanhos aleatórios experimentados.
3. A relação Heapsort/Quicksort é constante para todos os tamanhos.
4. A relação Shellsort/Quicksort aumenta se o número de elementos aumenta.
5. Para arquivos pequenos (500 elementos), o Shellsort é mais rápido que o Heapsort.
6. Se a entrada aumenta, o Heapsort é mais rápido que o Shellsort.
7. O Inserção é o mais rápido se os elementos estão ordenados.
8. O Inserção é o mais lento para qualquer tamanho se os elementos estão em ordem descendente.
9. Entre os algoritmos de custo $O(n^2)$, o Inserção é melhor para todos os tamanhos aleatórios experimentados.

Comparação entre os Métodos

Influência da ordem inicial dos registros:

	Shellsort			Quicksort			Heapsort		
	5.000	10.000	30.000	5.000	10.000	30.000	5.000	10.000	30.000
Asc	1	1	1	1	1	1	1,1	1,1	1,1
Des	1,5	1,6	1,5	1,1	1,1	1,1	1	1	1
Ale	2,9	3,1	3,7	1,9	2,0	2,0	1,1	1	1

1. Shellsort é bastante sensível à ordenação ascendente ou descendente da entrada.
2. Em arquivos do mesmo tamanho, o Shellsort executa mais rápido para arquivos ordenados.
3. Quicksort é sensível à ordenação ascendente ou descendente da entrada.
4. Em arquivos do mesmo tamanho, o Quicksort executa mais rápido para arquivos ordenados.
5. O Quicksort é o mais rápido para qualquer tamanho para arquivos na ordem ascendente.
6. O Heapsort praticamente não é sensível à ordenação da entrada.

Comparação entre os Métodos

Método da Inserção:

- É o mais interessante para arquivos com menos do que 20 elementos.
- O método é estável.
- Possui comportamento melhor do que o método da **bolha (Bubblesort)** que também é estável.
- Sua implementação é tão simples quanto as implementações do Bubblesort e Seleção.
- Para arquivos já ordenados, o método é $O(n)$.
- O custo é linear para adicionar alguns elementos a um arquivo já ordenado.

Comparação entre os Métodos

Método da Seleção:

- É vantajoso quanto ao número de movimentos de registros, que é $O(n)$.
- Deve ser usado para arquivos com registros muito grandes, desde que o tamanho do arquivo não exceda 1.000 elementos.

Comparação entre os Métodos

Shellsort:

- É o método a ser escolhido para a maioria das aplicações por ser muito eficiente para arquivos de tamanho moderado.
- Mesmo para arquivos grandes, o método é cerca de apenas duas vezes mais lento do que o Quicksort.
- Sua implementação é simples e geralmente resulta em um programa pequeno.
- Não possui um pior caso ruim e quando encontra um arquivo parcialmente ordenado trabalha menos.

Comparação entre os Métodos

Quicksort:

- É o algoritmo mais eficiente que existe para uma grande variedade de situações.
- É um método bastante frágil no sentido de que qualquer erro de implementação pode ser difícil de ser detectado.
- O algoritmo é recursivo, o que demanda uma pequena quantidade de memória adicional.
- Seu desempenho é da ordem de $O(n^2)$ operações no pior caso.
- O principal cuidado a ser tomado é com relação à escolha do pivô.
- A escolha do elemento do meio do arranjo melhora muito o desempenho quando o arquivo está total ou parcialmente ordenado.
- O pior caso tem uma probabilidade muito remota de ocorrer quando os elementos forem aleatórios.

Comparação entre os Métodos

Quicksort:

- Geralmente se usa a mediana de uma amostra de três elementos para evitar o pior caso.
- Esta solução melhora o caso médio ligeiramente.
- Outra importante melhoria para o desempenho do Quicksort é evitar chamadas recursivas para pequenos subarquivos.
- Para isto, basta chamar um método de ordenação simples nos arquivos pequenos.
- A melhoria no desempenho é significativa, podendo chegar a 20% para a maioria das aplicações (Sedgewick, 1988).

Comparação entre os Métodos

Heapsort:

- É um método de ordenação elegante e eficiente.
- Apesar de ser cerca de duas vezes mais lento do que o Quicksort, não necessita de nenhuma memória adicional.
- Executa sempre em tempo proporcional a $n \log n$,
- Aplicações que não podem tolerar eventuais variações no tempo esperado de execução devem usar o Heapsort.

Comparação entre os Métodos

Considerações finais:

- Para registros muito grandes é desejável que o método de ordenação realize apenas n movimentos dos registros.
- Com o uso de uma **ordenação indireta** é possível se conseguir isso.
- Suponha que o arquivo A contenha os seguintes registros: $A[1], A[2], \ldots, A[n]$.
- Seja P um arranjo $P[1], P[2], \ldots, P[n]$ de apontadores.
- Os registros somente são acessados para fins de comparações e toda movimentação é realizada sobre os apontadores.
- Ao final, $P[1]$ contém o índice do menor elemento de A, $P[2]$ o índice do segundo menor e assim sucessivamente.
- Essa estratégia pode ser utilizada para qualquer dos métodos de ordenação interna.