CURSO: Sistemas de Informação
DISCIPLINA: Estrutura, Pesquisa e Ordenação de Dados
PROFESSOR (**A**): Frederico Brito Fernandes
SEMESTRE/ANO: 2010.2
TURMA: Noite
CARGA HORÁRIA: 80 h/a

Exercícios de Fixação 3: Busca, Ordenação e Hash

EF 1.1) Descreva o funcionamento das buscas Linear e Binária, e das Tabelas Hash. Cite uma vantagem e desvantagem para cada uma delas.

EF 1.2) Pegue a implementação da Busca Binária vista em sala, e disponível em slide, e modifique-a, considerando agora que o vetor está ordenado de forma decrescente.

EF 1.3) Identifique quais são os algoritmos a seguir:

a) SelectionSort

```
for i ← 1 to N-1 do {
   current  ←  A[i]
   j  ←  i
   while (A[j-1] > current) And (j>0)
            A[j] ←  A[j-1]
            j  ←  j-1
   A[j] ←  current}
```

b) InsertionSort

```
if (inicio >= fim) then return;
indicePivo ← Partition(inicio, fim, v)
sort(inicio, indicePivo-1, v)
sort(indicePivo+1, fim, v)
```

c) QuickSort

```
for i ← 0 to N-2 do
   min ← i
   for j ← i+1 to N-1 do
      if A[j] < A[min] then min ← j
  temp ← A[min]
  A[min] ← A[i]
  A[i] ← temp
```

d) BubbleSort

```
for i ← 0 to n − 1 do
      for j ← i+1  to n-1  do

                        if (X[i]>X[j]) then
                            s ← X[i];
               X[i] ← X[j];
               X[j] ← s;
```

EF 1.4) Faça o acompanhamento dos valores do vetor a seguir, passo a passo, para os algoritmos abaixo. Após fazer o acompanhamento manual, pegue os algoritmos implementados em sala de aula, e modifique-os, fazendo-os imprimir o vetor a cada iteração. Por fim, compare com seu resultado manual.

int vetor[] = {23, 5, 2, 7, 3, 9, 11, 11}

23	5	2	7	3	9	11	1

a) BubbleSort
b) InsertionSort
c) SelectionSort
d) QuickSort
e) MergeSort

EF 1.5) No algoritmo BubbleSort visto em sala de aula, sempre que existia uma troca, o algoritmo retornar ao início (i=0). Realize uma modificação nesse código, com que as trocas sempre sejam realizadas até o fim, e evidentemente, o algoritmo deve parar quando não realizar mais trocas.

EF 1.6) Nos algoritmos InsertionSort e SelectionSot vistos em sala de aula, apenas um vetor é utilizado como entrada e saída, ou seja, a função recebe apenas um vetor desordenado, e retorna esse vetor de forma ordenada. Modifique essas funções, para elas recebam um vetorDesordenado com os valores a serem ordenados, mas que não modifique esse vetor, e que utilize o mesmo princípio de cada algoritmo para colocar os valores de forma ordenada em outro vetor de nome vetorOrdenado.

EF 1.7) Implemente os algoritmos InsertionSort e SelectionSort de forma recursiva.

EF 1.8) Pesquise sobre o método ShellSort e explique seu funcionamento, fazendo um acompanhamento dos valores das variáveis mais importantes, a cada iteração.

EF 1.9) A ordenação por transposição de par-ímpar ocorre da seguinte maneira: percorra o vetor várias vezes. Na primeira passagem compare x[i] com x[i+1] para todo i ímpar. Na segunda passagem compare x[i] com x[i+1] para todo i par. Todav vez que x[i] > x[i+1] troque os dois. Continue alternando dessa maneira até que o vetor esteja ordenado. Implemente essa função.

EF 1.10) Implemente uma função hash perfeita (sem colisões), para uma tabela de tamanho 11, considerando as chaves a seguir. Use o algoritmo de Cichelli.

São Paulo
Santos
Cruzeiro
Internacional
Flamengo
Grêmio
Vasco
Palmeiras
Atlético Mineiro
Corinthians
Botafogo

chaves

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

tabela