INSTITUTO FEDERAL DE GOIÁS
CÂMPUS GOIÂNIA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO
TEORIA GERAL DA ADMINISTRAÇÃO

Nome do Aluno: Luiz Antônio Rodrigues dos Santos
Data: 18/06/2018
Prof. Renan Rodrigues

Pesquisa Sequencial e Binária

Observação:
Em todos os casos a seguir, faça as implementações com base no código disponível no Moodle.

1. Escreva uma função que decida se um vetor v[0..n-1] está em ordem crescente.

```
int OrdemCrescente(TipoTabela *T) {
   int verifica = 0;

     for(int i=1; i<=T->n - 1; i++) {
          if (T->Item[i].Chave > T->Item[i+1].Chave) {
               verifica = 1;
          }
          printf("T->Item[i].Chave = %i | T->Item[i+1].Chave = %i\n", T-
>Item[i].Chave, T->Item[i+1].Chave);
     }
     return verifica;
}
```

2. Implemente uma versão do algoritmo de Pesquisa Sequencial que retorne o índice de todos os registros com uma mesma chave em um vetor (retorne em um array).

```
int* RegistroIgual(TipoTabela *T) {
   int verifica = 0;
   int v[T->n];

     for(int i = 1; i <= T->n; i++) {
      for(int j = 1; j <= T->n; j++) {
         if (i != j) {
            if (T->Item[i].Chave == T->Item[j].Chave) {
               v[verifica] = i;
               verifica++;
            }
         }
      }
     }
     return v;
}
```

3. Use o algoritmo HeapSort para ordenar o vetor 17 12 8 5 3 6 2 4 2 1. Mostre o estado do vetor no início de cada iteração.

v[17, 12, 8, 5, 3, 6, 2, 4, 2, 1]

```
                     | 17 |
            | 12 |             | 8 |
     | 5 |         | 3 |  | 6 |     | 2 |
| 4 |     | 2 |  | 1 |
```
v[17, 12, 8, 5, 3, 6, 2, 4, 2, 1]

```
                     | 17 |
            | 12 |             | 8 |
     | 5 |         | 3 |  | 6 |     | 2 |
| 4 |     | 2 |  | 1 |
```
v[17, 12, 8, 5, 3, 6, 2, 4, 2, 1]

```
                     | 17 |
            | 12 |             | 8 |
     | 5 |         | 3 |  | 6 |     | 2 |
| 4 |     | 2 |  | 1 |
```
v[17, 12, 8, 5, 3, 6, 2, 4, 2, 1]

```
                     | 17 |
            | 12 |             | 8 |
     | 5 |         | 3 |  | 6 |     | 2 |
| 4 |     | 2 |  | 1 |
```
v[17, 12, 8, 5, 3, 6, 2, 4, 2, 1]

```
                     | 17 |
            | 12 |             | 8 |
     | 5 |         | 3 |  | 6 |     | 2 |
| 4 |     | 2 |  | 1 |
```
v[17, 12, 8, 5, 3, 6, 2, 4, 2, 1]

v[12, 5, 8, 4, 3, 6, 2, 1, 2, 17]

v[8, 5, 6, 4, 3, 2, 2, 1, 12, 17]

v[6, 5, 2, 4, 3, 1, 2, 8, 12, 17]

v[5, 4, 2, 2, 3, 1, 6, 8, 12, 17]

v[4, 3, 2, 2, 1, 5, 6, 8, 12, 17]

v[3, 2, 2, 1, 4, 5, 6, 8, 12, 17]

v[2, 1, 2, 3, 4, 5, 6, 8, 12, 17]

v[2, 1, 2, 3, 4, 5, 6, 8, 12, 17]

v[1, 2, 2, 3, 4, 5, 6, 8, 12, 17]

4. Apresente o teste de mesa em cada passo do algoritmo da busca binária no vetor abaixo para a busca dos seguintes valores:

I. 5
II. 90
III. 46
14 21 5 45 12 3 86 98 46 53 24 2 1 15 90 47

Teste de mesa: Primeiro usar um algoritmo que ordena o vetor.
1 2 3 5 12 14 15 21 24 45 46 47 53 86 90 98
Fazendo a pesquisa binária:

I. 5
1 2 3 5 12 14 15 21 | 24 45 46 47 53 86 90 98
1 2 3 5 | 12 14 15 21 | 24 45 46 47 53 86 90 98
Encontrado 5

I. 90
1 2 3 5 12 14 15 21 | 24 45 46 47 53 86 90 98
1 2 3 5 12 14 15 21 | 24 45 46 47 | 53 86 90 98
1 2 3 5 12 14 15 21 | 24 45 46 47 | 53 86 | 90 98
1 2 3 5 12 14 15 21 | 24 45 46 47 | 53 86 | 90 98
Encontrado 90

I. 46
1 2 3 5 12 14 15 21 | 24 45 46 47 53 86 90 98
1 2 3 5 12 14 15 21 | 24 45 46 47 | 53 86 90 98
1 2 3 5 12 14 15 21 | 24 45 | 46 47 | 53 86 90 98
1 2 3 5 12 14 15 21 | 24 45 | 46 47 | 53 86 90 98
Encontrado 46

5. Considerando o algoritmo de ordenação QuickSort:

a) Explique detalhadamente o processo de partição, uma vez que esta é a parte mais delicada do algoritmo de ordenação Quicksort.

1. Escolha arbitrariamente um pivô x.
2. Percorra o vetor a partir da esquerda até que A[i] ≥ x.
3. Percorra o vetor a partir da direita até que A[j] ≤ x.
4. Troque A[i] com A[j].
5. Continue este processo até os apontadores i e j se cruzarem.

b) Considerando o vetor{8, 10, 3, 2, 9, 4, 1, 5, 7, 11, 6}, mostre um possível estado deste vetor após a operação de partição (para uma implementação qualquer), considerando o pivô como sendo:

I) o elemento central;

Pivô | 4 |

vetor{1, 2, 3, 4, 9, 10, 8, 5, 7, 11, 6}

II) o último elemento do vetor.

Pivô | 6 |

vetor{4, 5, 3, 2, 1, 6, 9, 10, 7, 11, 8}

6. Crie um programa em C que permita realizar cadastros de clientes. Cada cadastro de cliente deve ser composto pelos seguintes dados: código, nome, cidade e estado.

Além de permitir o cadastro de novos clientes, este aplicativo também deve permitir a exclusão de cadastros, pesquisa binária, ordenação e a produção de uma lista com todos eles.

O programa deve permitir que o usuário escolha entre ordenar os cadastros dos clientes com base no código ou no nome dos mesmos.

O programa também deve permitir a busca de registros.

Para realizar os procedimentos, utilize os algoritmos estudados que for mais eficiente para cada caso.

Código em anexo.