Análise de Complexidade: Big-O

1. Determine a ordem de complexidade dos algoritmos traduzidos nos seguintes blocos de código em linguagem C:

a)

```
...
for (k = 0; k < n; k++) {
   j = 0;
   while (j <= k)
      j = j + 1;
}
...
```

b)

```
...
for (k = 0; k < n; k++) {
   j = k;
   while (j < n)
      j = j + 1;
}
...
```

c)

```
...
for (k = 0; k < n; k++) {
   for (j = 0; j < n; j = j + 2)
      printf("%d\n", j);
}
...
```

d)

```
...
for (k = 0; k < n; k++) {
   for (j = 0; j < n; j = j + n/2)
      printf("%d\n", j);
}
...
```

e)

```
...
soma = 0;
for (k = 0; k < n; k++) {
   for (j = 0; j != k; j++)
      soma = soma + X[k][j];
}
...
```

f)

```
...
fim =  n % 10;
for (k = 0; k < fim; k++) {
   for (j = k; j < n; j++)
      printf("%d\n", j);
}
...
```

g)

```
...
cont = 0;
for (k = 0; k < n; k++) {
   p = k % 10;
   for (j = 0; j <= p; j++)
      cont++;
}
...
```

h)

```
...
soma = 0;
for (k = 0; k < n; k++)
   for (j = 0; j == k; j++)
      soma = soma + X[k][j];
...
```

i)

```
...
for (lin = 0; lin < n; lin++) {
   for (col = 0; col < n; col++) {
      Z[lin][col] = 0;
      for (k = 0; k < n; k++)
         Z[lin][col] = Z[lin][col] + X[lin][k] * Y[k][col];
   }
}
...
```

2. Determine a ordem de complexidade dos algoritmos que a seguir se apresentam (para mais detalhes, consultar os apontamentos sobre "Algoritmos de ordenação em arrays 1D", que se encontram disponíveis na página web da disciplina, na secção das folhas práticas).

a) Ordenação por seleção:

```
void ordenarSeleccao (int *V, int N)
{
   int   k, kk, pos_menor, aux;
   for (k = 0; k < N-1; k++) {
      pos_menor = k;
      for (kk = k+1; kk < N; kk++) {
         if (V[kk] < V[pos_menor])
            pos_menor = kk;
      }
      if (pos_menor != k) {
         aux = V[pos_menor];
         V[pos_menor] = V[k];
         V[k] = aux;
      }
   }
}
```

b) Ordenação por borbulhagem (BubbleSort):

```
void ordenarBorbulagem (int *V,  int N)
{
   int   k, kk, fim, aux;
   fim = N - 1;
   do{
      kk = 0;
      for (k = 0; k < fim; k++) {
         if (V[k] > V[k+1]) {
            aux = V[k];
            V[k] = V[k+1];
            V[k+1] = aux;
            kk = k;
         }
      }
      fim = kk;
   }while (kk != 0);
}
```

3. Determine a ordem de complexidade dos algoritmos que a seguir se apresentam (para mais detalhes, consultar os apontamentos sobre "Algoritmos de pesquisa em arrays 1D", que se encontram disponíveis na página web da disciplina, na secção das folhas práticas).

a) Pesquisa exaustiva:

```
int pesquisaExaustiva (int elem, int *V, int N)
{
   int  k;
   k = 0;
   while (k < N  &&  V[k] != elem) {
      k = k + 1;
   }
   if (k == N)
      return  -1;
   else
      return k;
}
```

b) Pesquisa sequencial:

```
int pesquisaSequencial (int elem, int *V, int N)
{
   int   k;
   k = 0;
   while (k < N  &&  V[k] < elem) {
      k = k + 1;
   }
   if (k < N  &&  elem == V[k])
      return  k;
   else
      return  -1;
}
```

c) Pesquisa binária (versão iterativa):

```
int pesquisaBinaria (int elem, int *V, int N)
{
   int  inicio, fim, meio, pos;
   inicio = 0;
   fim = N – 1;
   pos = -1;
   while (inicio <= fim  &&  pos == -1) {
      meio = (inicio + fim) / 2;
      if (elem == V[meio])
         pos = meio;
      else
         if (elem < V[meio])
            fim = meio - 1;
         else
            inicio = meio + 1;
   }
   return  pos;
}
```

Análise Amortizada de Algoritmos

1. Determine a complexidade amortizada (sobre as estruturas de dados) do algoritmo de ordenação por seleção. Consultar os apontamentos sobre "Algoritmos de ordenação em arrays 1D" (disponível na página web da disciplina, na secção das folhas práticas).

2. Determine a complexidade amortizada (sobre as estruturas de dados) do algoritmo de ordenação por borbulhagem (BubbleSort). Consultar os apontamentos sobre "Algoritmos de ordenação em arrays 1D" (disponível na página web da disciplina, na secção das folhas práticas).

3. Determine a complexidade amortizada (sobre as estruturas de dados) do algoritmo de ordenação rápida (QuickSort). Consultar os apontamentos sobre "Algoritmos de ordenação em arrays 1D" (disponível na página web da disciplina, na secção das folhas práticas).

4. Determine a complexidade amortizada (sobre as estruturas de dados) do algoritmo de ordenação por fusão (MergeSort). Consultar os apontamentos sobre "Algoritmos de ordenação em arrays 1D" (disponível na página web da disciplina, na secção das folhas práticas)..