Tópico: **Complexidade de Algoritmos Iterativos e Recursivos**

1. Encontre fórmula $T(n)$ para expressar a complexidade de tempo dos algoritmos a seguir:

(a)
```
int  exerc1(int n) {
  int soma = 0;
  for(int i = 1; i <= n; i++)
     soma += i*i*i;
  return soma;
 }
```

(b)
```
int exerc2(int n) {
    int a = 0;
    for(int i = 0; i < n; i+=2)
          a +=i;
       return a;
}
```

(c)
```
int exerc3(int n) {
    int a = 0;
    for(int i = 1; i < n; i++)
       for(int j = i+1; j <= n; j++)
          for(int k = 1; k <= i; k++)
             a += i + j + k;
    return a;
}
```

(d)
```
int exerc4(int n) {
    int a = 0;
    for(int i = 0; i < n; i++)
       for(int j = 0; j <= n-i ; j++)
          a += i + j;
    return a;
}
```

(e)
```
int exerc5(int n) {
    int a = 0;
    for(int i = 0; i < n; i++)
       for(int j = 0; j <= n-i ; j++)
          a += i + j;
    return a;
}
```

(f)
```
int exerc6(int n) {
    int a = 0;
    for(int i = 1; i < n; i=i*2)
       a += i;
    return a;
}
```

(g)
```
int exerc7(int n) {
    int a = 0;
    for(int i = 0; i < n*n; i++)
       for(int j = 0; j <= i; j++)
          a +=i;
```

```
      return a;
   }
```

(h)
```
int exerc(int n) {
    if n = 0
      then return 1;
      else return n + exerc(n-1);
}
```

(i)
```
int exerc(int n) {
   if n = 0
     then return 1;
     else return 2*exerc(n-1);
}
```

(j)
```
int exerc(int n) {
   if n = 0
     then return 1;
     else return exerc(n-1) + exerc(n-1);
}
```

(k)
```
int exercL(int n) {
   if n = 0
     then return 1;
     else return n*exerc(n-1);
}
```

(l)
```
int exerc(int n) {
   if n <= 1
     then return 1;
     else return exerc(n div 2) + n;
}
```

(m)
```
int exerc(int n) {
   if n = 0
     then return 1;
     else {
           int a = 0;
           for(i = 1; i <= n; i++)
              a += i;
           return exerc(n-1) + a;
          }
}
```

(n)
```
int exerc(int n) {
   if n = 0
     then return 1;
     else {
           int a = 0;
           for(i = 1; i <= n; i++)
              a += exerc(n-i);
           return a;
          }
}
```

(o)
```
int exerc(int n) {
   if n = 0
     then return 1;
     else return exerc(n-1) + exerc(n-1) +
                 exerc(n-2) + exerc(n-2) + exerc(n-2);
}
```

(p)
```
int exerc(int n) {
    if n = 0
      then return 1;
      else return 2*exerc(n-1) + 3*exerc(n-2);
}
```

(q)
```
int exerc(int n) {
    if n = 0
      then return 1;
      else return exerc(exerc(n-1)) + n;
}
```

2. Considere algoritmos A, B e C de complexidade de tempo $T_A \in O(nlg\,n)$, $T_B \in O(n)$ e $T_C \in O(n^2)$. Calcule a complexidade do melhor caso e do pior caso dos algoritmos abaixo:

(a)
```
Algoritmo P1
  if A
     then B
     else C
  endif
end
```

(b)
```
Algoritmo P2
  A;
  B;
  C;
end
```

(c)
```
Algoritmo P3
  for i = 1 to n do
     A;
  od
  B;
  C;
end
```

(d)
```
Algoritmo P4
  if B
    then A
  endif
  C;
end
```

(e)
```
Algoritmo P5
  for i = 1 to n do
     B;
     C;
  od
  A;
end
```

3. Projete um algoritmo para calcular a moda (valor mais frequente) de um vetor não ordenado. Calcule a complexidade de tempo de seu algoritmo.

4. Projete um algoritmo para calcular a moda de um vetor ordenado. Calcule a complexidade de seu algoritmo.