**Universidade Federal de Ouro Preto**
**Departamento de Computação – DECOM**
**BCC241 – Projeto e Análise de Algoritmo**
**Prof. Anderson Almeida Ferreira**
**Exercícios – Divisão e Conquista**

**Turma**: 11
**Matrícula**: 20.1.4003

1.

```c
int max (int arr[], int inicio, int fim) {
    if((fim - inicio) == 0) {
        return arr[fim];
    }
    int meio = (inicio + fim) / 2;
    int a = max(arr, inicio, meio);
    int b = max(arr, meio + 1, fim);
    if (a > b)
        return a;
    else
        return b;
}
```

$T(n) = 2T(n/2) + O(1)$
$a = 2, b = 2, d = 0$
$\log 2\ 2 = 1 > d$
$O(n^{lg2}) = O(n)$

2.

```c
int min (int arr[], int inicio, int fim) {
    if((fim - inicio) == 0) {
        return arr[fim];
    }
    int meio = (inicio + fim) / 2;
    int a = min(arr, inicio, meio);
    int b = min(arr, meio + 1, fim);
    if (a > b)
        return b;
    else
        return a;
}

void minMax (int arr[], int inicio, int fim) {
    int v[2] = {max(arr, inicio, fim), min(arr, inicio, fim)};
    printf("Min: %d ; Max: %d\n", v[1], v[0]);
}
```

min = O(n)
minMax = O(2n) = O(n)

3.

```
int expo(int base, int exp) {
    if (exp == 0)
        return 1;
    int a = expo(base, exp/2);
    a *= a;
    if (exp % 2 == 1)
        a *= base;
    return a;
}
```

T(n) = T(n/2) + O(1)
a = 1, b = 2, d = 0
lg 1 = 0 = d
O(log n)