Ciências da Computação – PUC Minas Algoritmo e Estrutura de Dados II Trabalho Teórico 6

```
1) for (int i = 0;i < n; i++ ) {
      if(maior < aray[i]){
          maior = array[i];
      }
      if (menor > array[i]) {
          menor = array[i];
      }
}
```

2)

	O(1)	O(lg n)	O(n)	O(n.lg(n))	O(n²)	O(n³)	O(n ⁵)	O(n ²⁰)
f(n) = Ig(n)		٧	٧	V	٧	٧	٧	V
$f(n) = n \cdot lg(n)$				V	٧	V	٧	٧
f(n) = 5n + 1			٧	٧	٧	٧	٧	٧
$f(n) = 7n^5 - 3n^2$							٧	٧
$f(n) = 99n^3 - 1000n^2$						٧	٧	٧
f(n) = n ⁵ - 99999n ⁴							٧	٧

3)

	Ω(1)	Ω(lg n)	Ω(n)	Ω(n.lg(n))	Ω(n²)	Ω(n³)	Ω(n ⁵)	Ω(n ²⁰)
f(n) = Ig(n)	٧	٧						
$f(n) = n \cdot lg(n)$	٧	٧	٧	V				
f(n) = 5n + 1	٧	٧	٧					
$f(n) = 7n^5 - 3n^2$	٧	٧	٧	V	>	>	>	
$f(n) = 99n^3 - 1000n^2$	٧	V	^	V	V	٧		
f(n) = n ⁵ - 99999n ⁴	٧	V	٧	V	٧	V	V	

	Ω(1)	Ω(lg n)	Ω(n)	$\Omega(n.lg(n))$	Ω(n²)	$\Omega(n^3)$	Ω(n ⁵)	Ω(n ²⁰)
f(n) = Ig(n)	٧	٧						
$f(n) = n \cdot lg(n)$	٧	٧	>	V				
f(n) = 5n + 1	٧	٧	٧					
$f(n) = 7n^5 - 3n^2$	٧	\	٧	V	٧	٧	٧	
$f(n) = 99n^3 - 1000n^2$	٧	V	٧	V	٧	٧		
f(n) = n ⁵ - 99999n ⁴	V	٧	٧	V	V	٧	V	

5)

	⊖ (1)	⊖ (lg n)	Θ (n)	⊖ (n.lg(n))	⊖ (n²)	Θ(n³)	Θ (n⁵)	Θ(n ²⁰)
f(n) = Ig(n)		V						
$f(n) = n \cdot lg(n)$				V				
f(n) = 5n + 1			٧					
$f(n) = 7n^5 - 3n^2$							V	
$f(n) = 99n^3 - 1000n^2$						٧		
f(n) = n ⁵ - 99999n ⁴							V	

6)

$$F-C-E-A-B-D\\$$

7)

$$C = 4$$

$$M = 6$$

$$F(n) = 3n^2 + 5n + 1 = 0 (n^2)$$

$$C = 1$$

$$M = 5$$

$$F(n) = 3n^2 + 5n + 1 = 0 \ (n^2)$$

$$C = 10$$

$$M = 2$$

$$F(n) = 3n^2 + 5n + 1 \neq 0 \ (n^2)$$

8)

$$C = 1$$

$$M >= 0$$

$$F(n) = 3n^2 + 5n + 1 = \Omega(n^2)$$

$$C = 1$$

$$M = 5$$

$$F(n) = 3n^2 + 5n + 1 = \Omega(n)$$

$$C = 1$$

$$M = 5$$

$$F(n) = 3n^2 + 5n + 1 \neq \Omega(n)$$

9)

$$C_1 = 1$$

$$C_2 = 4$$

$$M = 6$$

$$F(n) = 3n^2 + 5n + 1 = \Theta(n^2)$$

$$C_1=1\\$$

$$C_2 = 10$$

$$M = 2$$

$$F(n) = 3n^2 + 5n + 1 \neq \Theta(n)$$

$$C_1 = 1$$

$$C_2 = 2$$

$$M >= 0$$

$$F(n) = 3n^2 + 5n + 1 \neq \Theta(n^3)$$

12)

#include <stdio.h>

void printMaxMin(int array[], int n){
 int max, min;

```
if(array[0] > array[1]){
    max = array[0];
    min = array[1];
}
else{
    max = array[1];
    min = array[0];
}

for (int i = 2;i < n; i++){
    if(array[i] > max){
        max = array[i];
    }
    else if(array[i] < min){
        min = array[i];
    }
}
printf("maior: %d\n", max);
printf("menor: %d\n", min);
}</pre>
```

a solução mais eficiente é a pesquisa binária pois a ordenação só será necessária uma única vez e após a ordenação o custo de n pesquisas binárias será n * $\Theta(\lg(n))$ e o custo de n pesquisas sequenciais será n * $\Theta(n)$.