1. Exercícios

1.1. Exercício 1

```
1                   public static void maiorMenor(int[] array) {
2                       int menor = array[0];
3                       int maior = array[0];
4                       for (int i = 1; i < array.length; i++) {
5                           if (menor > array[i]) {
6                               menor = array[i];
7                           } else {
8                               if (maior < array[i]) {
9                               maior = array[i];
10                              }
11                          }
12                      }
13                      System.out.println("maior = " + maior);
14                      System.out.println("menor = " + menor);
15                  }
```

Como o código precisa necessariamente percorrer todo o vetor, o índice vai obrigatoriamente de 1 a array.lengh (n).

Desta forma, o código no pior caso seria se o primeiro if desse falso e o else if desse verdadeiro, ou seja, se o array estivesse ordenado de forma crescente. $2 * (n - 1)$ Duas comparações por iteração, $n - 1$ vezes.

Já no melhor caso, seria se o primeiro if desse verdadeiro, ou seja, se o array estivesse ordenado de forma decrescente. $1 * (n - 1)$ Uma comparação entre itens do array por iteração, $n - 1$ vezes.

1.2. Exercício 3 ao 10

Estão em anexo na pasta anexos.

1.3. Exercício 11

Melhor caso

- Telefone: 1, $O(1)\Theta(1)\Omega(1)$
- Alarme: 1, $O(1)\Theta(1)\Omega(1)$
- Luz: 0, $O(0)\Theta(0)\Omega(0)$
- Sensor: $(n-2), O(n)\Theta(n)\Omega(n)$
- Câmera: $(n-2), O(n)\Theta(n)\Omega(n)$

Pior caso

- Telefone: 1, $O(1)\Theta(1)\Omega(1)$
- Alarme: $1 + 2(n-2), O(n)\Theta(n)\Omega(n)$
- Luz: 0, $O(1)\Theta(1)\Omega(1)$
- Sensor: $(n-2), O(n)\Theta(n)\Omega(n)$
- Câmera: $(n-2), O(n)\Theta(n)\Omega(n)$

1.4. Exercício 12

```
1                   public static void maiorMenor(int[] array) {
2                       int menor = array[0];
3                       int maior = array[0];
4                       for (int i = 1; i < array.length; i++) {
5                           if (menor > array[i]) {
6                               menor = array[i];
7                           } else {
8                               if (maior < array[i]) {
9                               maior = array[i];
10                              }
11                          }
12                      }
13                      System.out.println("maior = " + maior);
14                      System.out.println("menor = " + menor);
15                  }
```

Operações: Atribuição e comparação.

- Atribuição: **Melhor caso** $2, O(2), \Theta(2), \Omega(2)$ **Pior caso** $2+(n-1), O(n), \Theta(n), \Omega(n)$
- Comparação: **Melhor caso** $(n-1), O(n), \Theta(n), \Omega(n)$ **Pior caso** $2(n-1), O(n), \Theta(n), \Omega(n)$

1.5. Exercício 13

Neste caso seria mais vantajoso ordenar o array e aplicar pesquisa binária, pelo fato de realizar n pesquisas sequenciais, a complexidade seria de $\Theta(n^2)$. Enquanto ordenar e pesquisar binariamente, teria uma complexidade de $\Theta(n * \log_2(n))$

2. Exercícios Resolvidos

2.1. Exercício 1

$a) 2^{10} = 1024$
$b) \log_2(1024) = 10$
$c) \log_2(17) = 4,08746284125034$
$d) \lceil \log_2(17) \rceil = 5$
$e) \lfloor \log_2(17) \rfloor = 4$

2.2. Exercício 2

2.3. Exercício 3

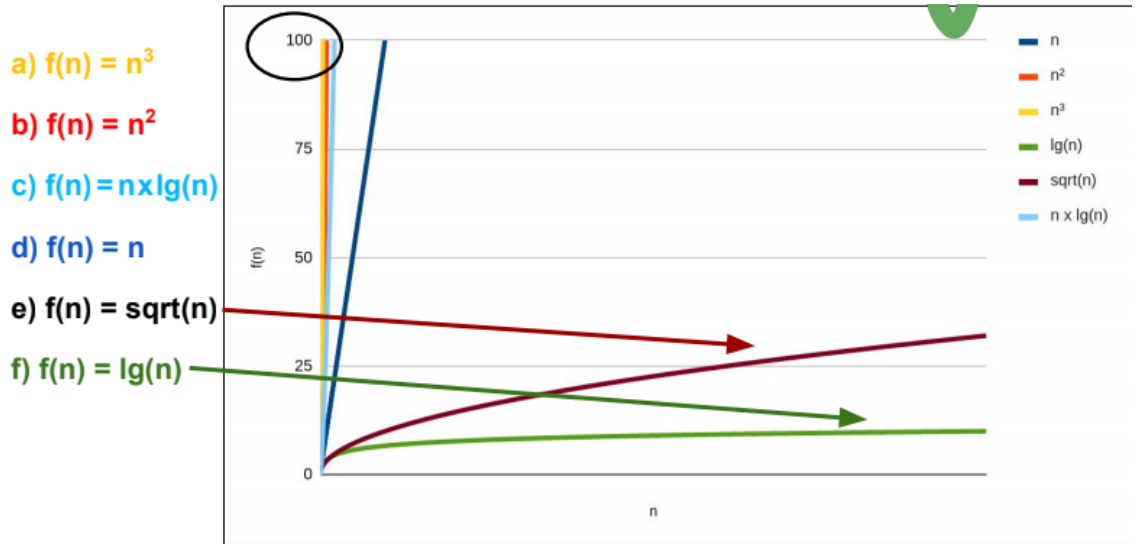
Melhor caso: $f(n) = n$ logo $O(n), \Omega(n), \Theta(n)$
Pior caso: $f(n) = 2n$ logo $O(n), \Omega(n), \Theta(n)$

2.4. Exercício 4

O código realiza $(n-3)$ operações.

2.5. Exercício 5

O código efetua $\log_2(n) + 1$ operações.



2.6. Exercício 8

Sim, porque temos que testar todos os elementos para garantir nossa resposta.

2.7. Exercício 9

Sim, porque temos que testar todos os elementos para garantir nossa resposta.

2.8. Exercício 10

O aluno deve escolher a primeira opção, pois a pesquisa sequencial tem custo $\Theta(n)$. A segunda opção tem custo $\Theta(n * \log_2(n))$ para ordenar, mais $\Theta(n)$ para a pesquisa binária.

2.9. Exercício 12

Função de complexidade

MOV CMP

PIOR $f(n) = 2 + (n–2)|f(n) = 1 + 2(n–2)$
MELHOR $f(n) = 2 + (n–2)x0|f(n) = 1 + (n–2)$

Complexidade

MOV CMP
PIOR $O(n), \Omega(n) e \Theta(n) O(n), \Omega(n) e \Theta(n)$
MELHOR $O(1), \Omega(1) e \Theta(1) O(n), \Omega(n) e \Theta(n)$