<u>EXERCÍCIO RESOLVIDO</u>

1 a) 2^10 = 1024

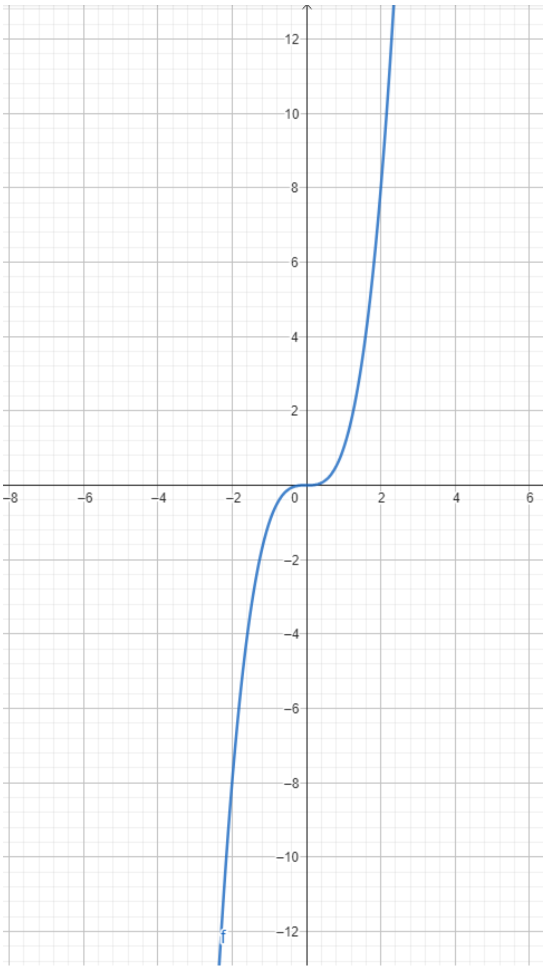
b) lg(1024) = 10

c) lg(17) = 4,087

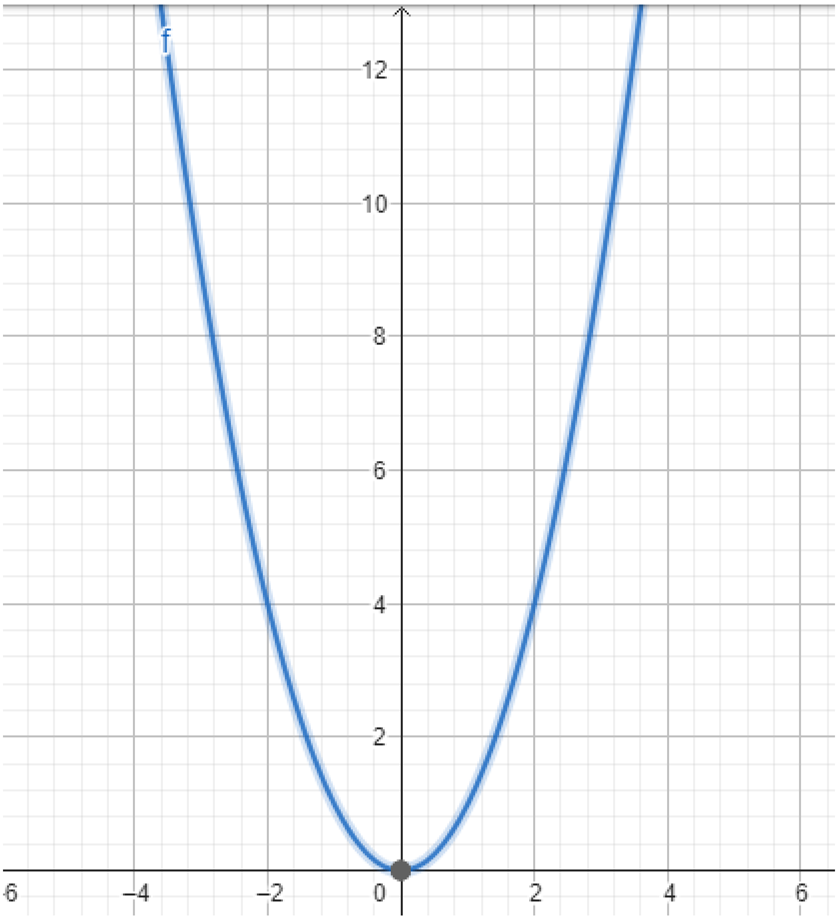
d) teto de lg(17) = 5

e) piso de lg(17) = 4

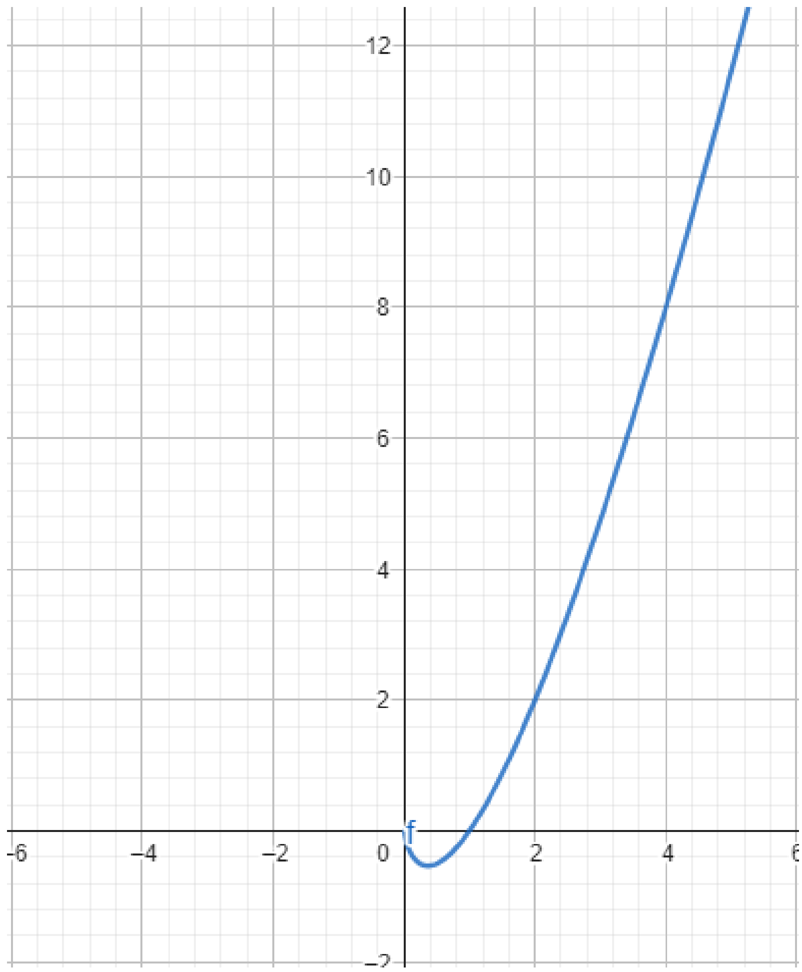
2 a) f(n) = n^3



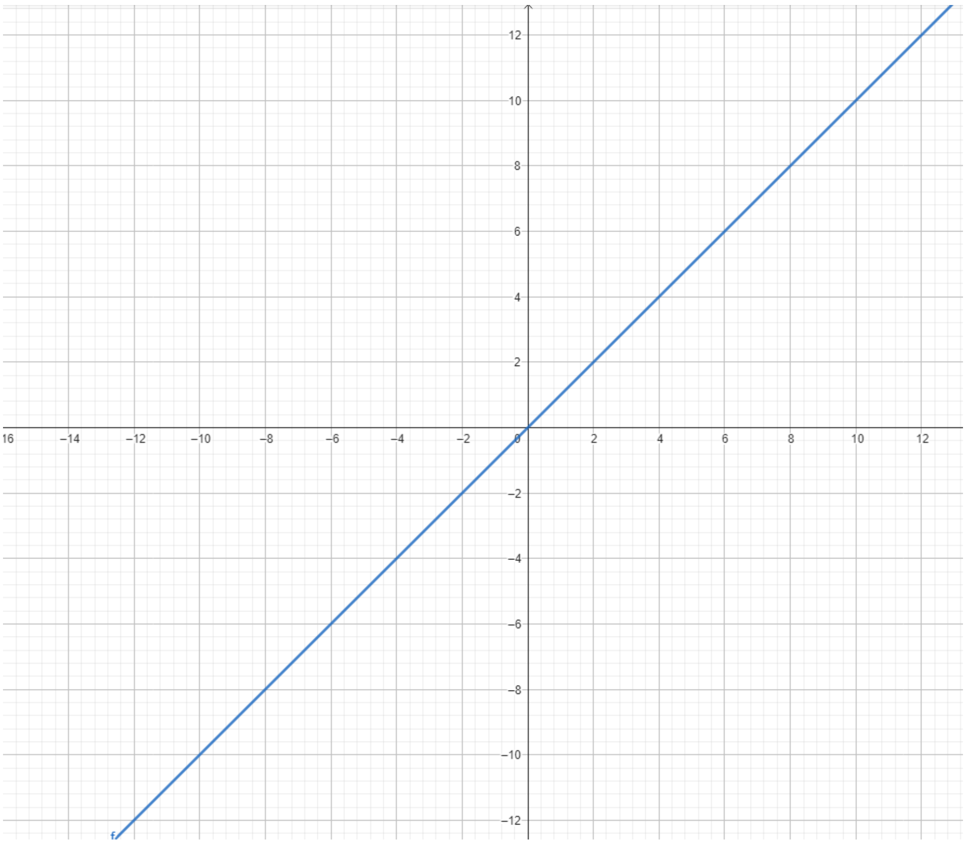
B) f(n) = n^2



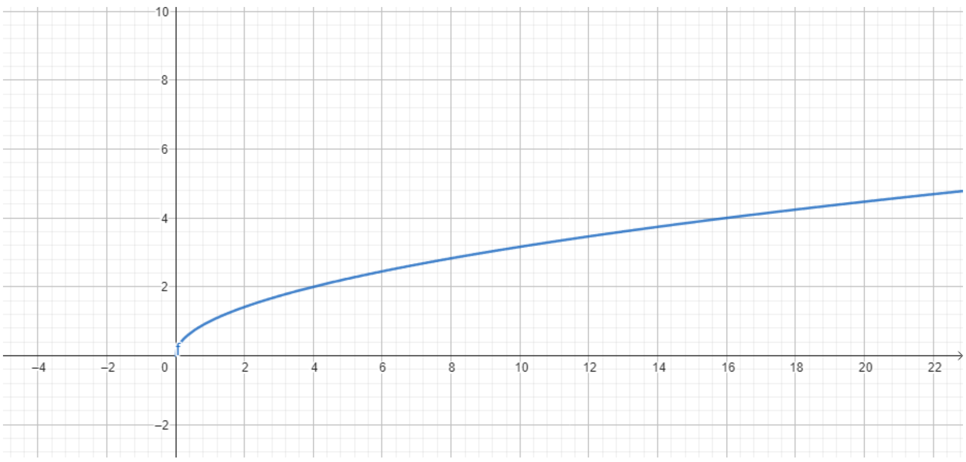
C) f(n) = n x lg(n)



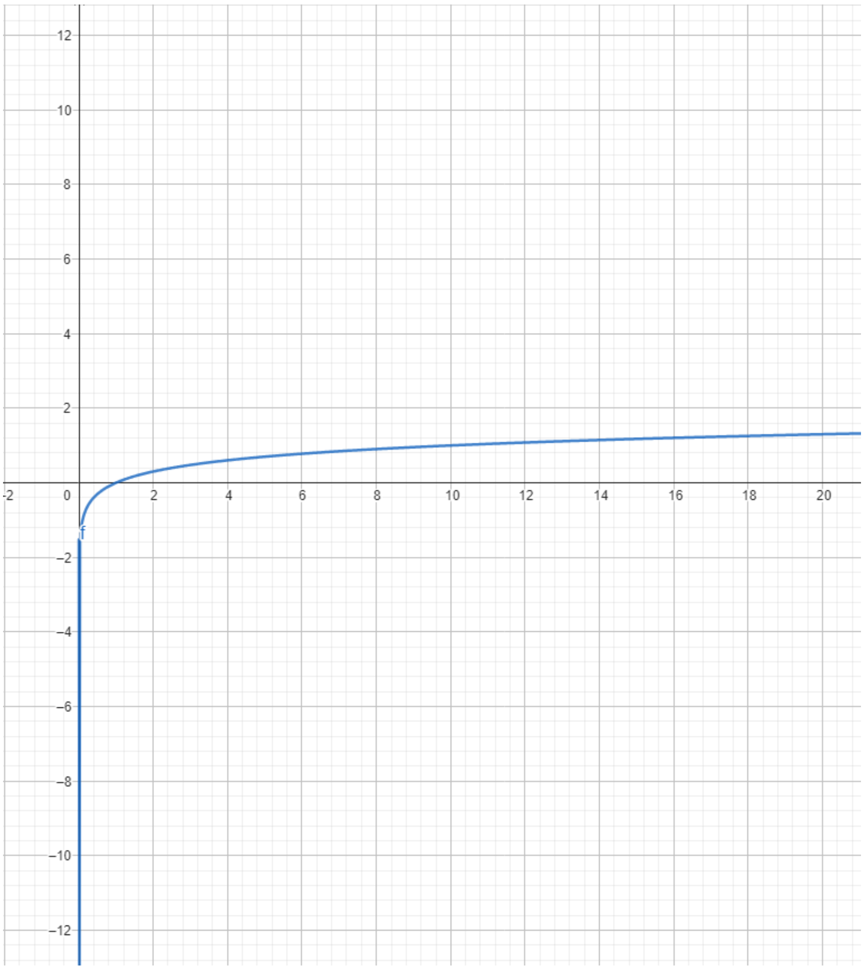
D) f(n) = n



E) f(n) = sqrt(n)



F) f(n) = lg(n)



NOÇÕES DE COMPLEXIDADE

3)

Melhor caso: 3 subtrações. Faz a verificação do if (2 subtrações), dá falso, e entra no else (1 subtração)

Pior caso: 5 subtrações. Faz a verificação do if(2 subtrações), dá true, entra no if e faz mais 3 subtrações

4) O código realiza 2n subtrações. De 0 até n, e as duas subtrações que ocorrem dentro do for.

5) O código realiza 3n^2 subtrações. De 0 até n no for externo e interno, e as três subtrações que ocorrem dentro do for interno.

6) O código realiza lg(n) + 1 multiplicações

ASPECTOS DA ANÁLISE DE ALGORITMOS

7)

Melhor caso: elemento desejado na
primeira posição
t(n) = 1

Pior caso: elemento desejado não está no
array ou está na última posição
t(n) = n

8) A opção mais viável seria ordenar o array e aplicar uma busca binária, porque é mais eficiente. Entretanto, o aluno deve escolher a primeira opção, pois a pesquisa sequencial tem custo $\Theta(n)$. A segunda opção tem custo $\Theta(n \times \lg n)$ para ordenar mais $\Theta(\lg n)$ para a pesquisa binária.

Notações Θ, O e Ω

9)

a) 3n 2 + 5n + 1 é O(n): falsa
b) 3n 2 + 5n + 1 é O(n 2): verdadeira
c) 3n 2 + 5n + 1 é O(n 3): verdadeira

d) $3n^2 + 5n + 1$ é $\Omega(n)$: verdadeira
e) $3n^2 + 5n + 1$ é $\Omega(n^2)$: verdadeira
f) $3n^2 + 5n + 1$ é $\Omega(n^3)$: falsa

g) $3n^2 + 5n + 1$ é $\Theta(n)$: falsa
h) $3n^2 + 5n + 1$ é $\Theta(n^2)$: verdadeira
i) $3n^2 + 5n + 1$ é $\Theta(n^3)$: falsa

10) $\Theta(n^3)$, pois dentro do for é executado a função que realiza um algoritmo de seleção, ou seja, $\Theta(n^2) + \Theta(n) = \Theta(n^3)$

11)
a) h(n) + g(n) - f(n)

99n^8 + n x lg(n) - 3n^2 – 5n – 9 = Θ (n^8)

b) $\Theta(h(n)) + \Theta(g(n)) - \Theta(f(n))$

99n^8 + n x lg(n) - 3n^2 – 5n – 9 = Θ (n^8)

c) f(n) x g(n)

$(3n^2 - 5n - 9) \times (n.lg(n)) = \Theta(n^2) \times \Theta(n.lg(n)) \Rightarrow \Theta(n^3.lg(n))$

d) g(n) x l(n) + h(n)

Θ(n.lg(n)) x Θ(n.lg 2 (n)) + Θ(n 8) ⇒ Θ(n 8)

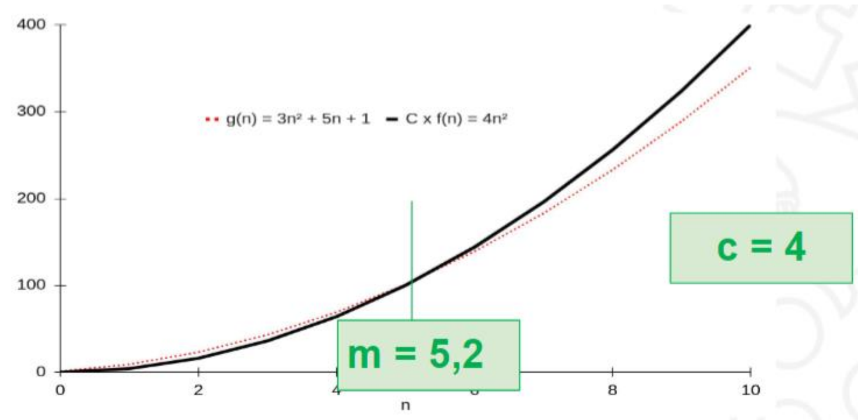
e) f(n) x g(n) x l(n

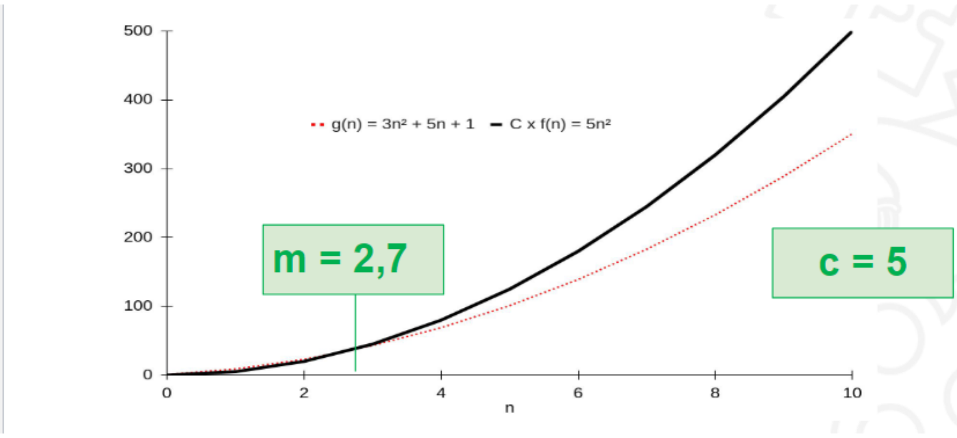
Θ(n 2) x Θ(n.lg(n)) x Θ(n.lg 2 (n)) ⇒ Θ(n 4 .lg 3 (n))

f) Θ(Θ(Θ(Θ(f(n)))))

Θ(n)

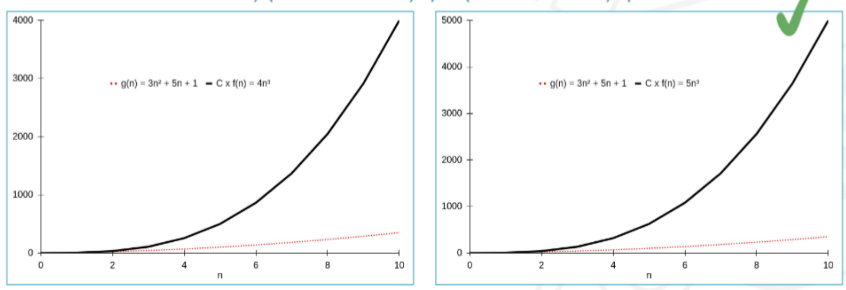
12 a) Para que tal inequação seja verdadeira, c tem que ser maior do que três





b)

RESPOSTA: Novamente, (c = 4 e m = 5,7) e (c = 5 e m = 2,7)



c)

Não existe par (c, m) tal que para $n \geq m$, $|3n^2 + 5n + 1| \leq c \times |n|$ seja verdadeira. Aumentando o valor de c, apenas retardamos o momento em que a curva quadrática supera a linear

Fazendo C = 100		
n	$g(n) = 3n^2 + 5n + 1$	C x f(n) = 100 x n
0	1	0
5	101	500
10	351	1000
15	751	1500
20	1301	2000
25	2001	2500
30	2851	3000
35	3851	3500
40	5001	4000
45	6301	4500
50	7751	5000

Fazendo C = 1000		
n	$g(n) = 3n^2 + 5n + 1$	C x f(n) = 1000 x n
0	1	0
50	7751	50000
100	30501	100000
150	68251	150000
200	121001	200000
250	188751	250000
300	271501	300000
350	369251	350000
400	482001	400000
450	609751	450000
500	752501	500000

13)

Melhor caso: $f(n) = 1 + (n - 2)$

Pior caso: f(n) = 1 + 2(n – 2)

14)

Melhor caso: f(n) = 2 + (n – 2) x 0

Pior caso: f(n) = 2 + (n – 2)

15)

Melhor caso: f(n) = n +1

Pior caso: f(n) = n + 2

Ordem de complexidade: O(n), Ω(n) e **Θ**(n)

16)

Todos os casos: f(n) = (2n + 1)n

Ordem de complexidade: O(n2), Ω(n2) e **Θ**(n2)

17)

Todos os casos: f(n) = (lg(n) + 1) x n = n x lg(n) + n

Ordem de Complexidades: O(n x lg(n)), Ω(n x lg(n)) e **Θ**(n x lg(n))

18)

Apresente o tipo de crescimento que melhor caracteriza as funções abaixo (Khan Academy, adaptado) ✓

	Constante	Linear	Polinomial	Exponencial
$3n$		✓		
1	✓			
$(3/2)n$		✓		
$2n^3$			✓	
2^n				✓
$3n^2$			✓	
1000	✓			
$(3/2)^n$				✓

19)

f 6 (n) = 1 < f 2 (n) = n < f 1 (n) = n 2 < f5 (n) = n 3 < f4 (n) = (3/2) n < f3 (n) = 2

20)

f6 (n) = 64 < f 3 (n) = log 8 (n) < f 2 (n) = lg(n) < f 9 (n) = 4n < f 1 (n) = n.log 6 (n) <
f 5 (n) = n.lg(n) < f 4 (n) = 8n 2 < f7 (n) = 6n 3 < f8 (n) = 8 2n

21)

Faça a correspondência entre cada função f(n) com sua g(n) equivalente, em termos de O. Essa correspondência acontece quando f(n) = O(g(n)) (Khan Academy, adaptado)

f(n)	g(n)
$n + 30$	n^4
$n^2 + 2n - 10$	$3n - 1$
$n^3 \times 3n$	$\lg(2n)$
$\lg(n)$	$n^2 + 3n$

<u>Exercícios</u>

1)

```
public class MinMaxFinder {
    public static void main(String[] args) {
        int[] array = {5, 2, 9, 1, 5, 6};
        int[] result = findMinMax(array);

        System.out.println("Menor valor: " + result[0]);
        System.out.println("Maior valor: " + result[1]);
    }

    public static int[] findMinMax(int[] arr) {
        if (arr == null || arr.length == 0) {
            return new int[]{};
        }

        int min_value = Integer.MAX_VALUE; // Inicializa com o
maior valor possível
        int max_value = Integer.MIN_VALUE; // Inicializa com o
menor valor possível

        for (int num : arr) {
            if (num < min_value) {
                min_value = num;
            }
```

```
            if (num > max_value) {
                max_value = num;
            }
        }

        return new int[]{min_value, max_value};
    }
}
```

2)

3)

	$\Theta(1)$	$\Theta(\lg n)$	$\Theta(n)$	$\Theta(n.\lg(n))$	$\Theta(n^2)$	$\Theta(n^3)$	$\Theta(n^5)$	$\Theta(n^{20})$
$f(n) = \lg(n)$		✓						
$f(n) = n . \lg(n)$				✓				
$f(n) = 5n + 1$			✓					
$f(n) = 7n^3 - 3n^2$						✓		
$f(n) = 99n^3 - 1000n^2$			✓					
$f(n) = n^5 - 99999n^4$			✓					

4)

5)

6)

7)

8)

9)

10)

Ao considerar múltiplas pesquisas, a segunda abordagem (ordenar + pesquisa binária) é mais eficiente em termos de complexidade de tempo. Embora ordenar o array inicialmente tenha um custo maior, a pesquisa subsequente usando a pesquisa binária é mais rápida em comparação com a pesquisa sequencial, especialmente quando o número de pesquisas é grande.