EXERCÍCIOS

3)

	O(1)	O(lg n)	O(n)	O(n.lg(n))	$O(n^2)$	$O(n^3)$	$O(n^5)$	$O(n^{20})$
f(n) = lg(n)	F	V	V	V	V	V	V	V
f(n) = n . lg(n)	F	F	F	V	V	V	V	V
f(n) = 5n + 1	F	F	V	V	V	V	V	V
$f(n) = 7n^5 - 3n^2$	F	F	F	F	F	F	V	V
$f(n) = 99n^3 - 1000n^2$	F	F	F	F	F	V	V	V
$f(n) = n^5 - 99999n^4$	F	F	F	F	F	F	V	V

4)

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

5)

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

6)

a) O(máximo(f(n),g(n)) - O(h(n)))
b) O(máximo(f(n),g(n)) - O(h(n)))
c) O(f(n) x g(n))
d) O(máximo(f(n) x g(n)),O(h(n)))
e) O(f(n) x g(n) x l(n))
f) O(f(n))

7)

a) c = 3
m = 6

b) c = 2
m = 4

c) n não é limite superior para n^2 pois sua potência é menor

8)

a) c = 2
   m = 3

 b) c = 1
    m = 3

 c) Porque n^3 não é o limite inferior de n^2

9)
a) c1 = 2
    c2 = 3
    m = 5

 b) não é limite justo pois n não é igual a n^2

 c) não é limite justo pois n^3 não é igual a n^2

11)
a)Alarme
Pior Caso: O(n)
Melhor Caso: O(2)

b)Outros metodos
Pior Caso: O(n) + n
Melhor Caso: O(n)

12)

```
if(n<3) n . 2;
else{
        for(int i=0; i<n; i++;){
                n+=3;
                n = n . i;
        }
}
```

Pior caso: n>3 | O(n . 2)
Melhor cas:=o: n<3 | O(1)

13)A solução que ordena o array primeiro é mais eficiente a longo prazo, pois o array já vai estar ordenado para todas pesquisas futuras

EXERCÍCIOS RESOLVIDOS

10)
O aluno deve escolher a primeira opção, pois a pesquisa sequencial tem custo Θ(n). A segunda opção tem custo Θ (n * lg n) para ordenar mais Θ(lg n) para a pesquisa binária

11)
a)Falsa
b)Verdadeira
c)Verdadeira
d)Verdadeira
e)Verdadeira
f)Falsa
g)Falsa
h)Verdadeira
i)Falsa

12)
----------Função de Complexidade----------

	MOV	CMP
PIOR:	f(n)=2+(n-2)	f(n) = 1 + 2(n-2)
MELHOR:	f(n) = 2+(n-2) x 0	f(n) = 1 + (n-2)

----------Complexidade----------

	MOV	CMP
PIOR:	O(n), Ω(n) e Θ(n)	O(n), Ω(n) e Θ(n)
MELHOR:	O(1), Ω(1) e Θ(1)	O(n), Ω(n) e Θ(n)

13)

	Função	complexidade
PIOR:	f(n)= n + 2	O(n), Ω(n) e Θ(n)
MELHOR:	f(n) = n + 1	O(n), Ω(n) e Θ(n)

14)

	Função	complexidade
TODOS:	f(n) = (2n + 1)n	O(n2), Ω(n2) e Θ(n2)

15)

	função	complexidade
TODOS:	f(n) = (lg(n)+1)*n=n*lg(n)+n	O(n x lg(n)), Ω(nxlg(n)) e Θ(n x lg(n))

16)

	Constante	Linear	Polinomial	Exponencial
3n		v		
1	v			
(3/2)n		v		
2n3			v	
2n				v
3n2			v	
1000	v			
(3/2)n				v

17)

f6(n) = 1

f2(n) = n

f1(n) = n2

f5(n) = n3

f4(n) = (3/2)n

f3(n) = 2n

18)

f6(n) = 64
f3(n) = log8(n)
f2(n) = lg(n)
f9(n) = 4n
f1(n) = n.log6(n)
f5(n) = n.lg(n)
f4(n) = $8n^2$
f7(n) = $6n^3$
f8(n) = 8^2n

19)

f(n)		g(n)
n + 30		3n-1
n^2 + 2n - 10		n^2+3n
n^3 x 3n	n^4	
lg(n)		lg(2n)

Resumo de Complexidade

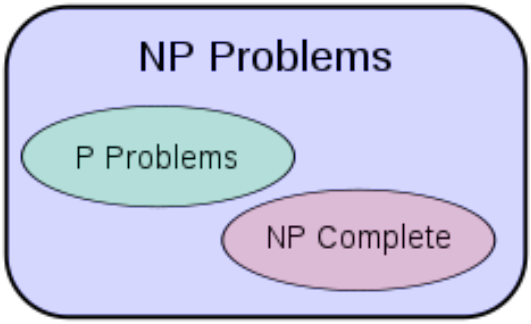
João Pedro de Castro Ribeiro

[1]Ciência da Computação – Pontifícia Universidade Católica de Minas Gerais (PUC-MG)

joaopedrocastro05@gmail.com

A teoria da complexidade computacional é um ramo da teoria da computação que se concentra em classificar problemas computacionais de acordo com sua dificuldade inerente, e relacionar essas classes entre si. Um problema computacional consiste de instâncias do problema e soluções para essas instâncias do problema.

A classe de complexidade P é vista na maioria dos casos como uma abstração matemática de modelagem que admitem um algoritmo eficiente. A classe de complexidade NP contém muitos problemas que as pessoas gostariam de resolver de forma eficiente, mas nenhum algoritmo que seja eficiente é conhecido.



A complexidade de muitos problemas computacionais de interesse prático é desconhecida. Para muitos desses problemas, não temos um algoritmo polinomial nem sabemos se um tal algoritmo é impossível. Isso continua assim mesmo que tratemos apenas de problemas polinomialmente verificáveis. Em particular, para muito problemas de decisão importantes, não sabemos se estão na classe P. Resta-nos procurar entender, ao menos, quais desses problemas são menos provavelmente polinomiais.

Um problema é considerado como difícil se a sua resolução requer muito recursos, não importando qual algoritmo foi usado. A teoria formaliza esta intuição para estudar estes problemas e quantificar os recursos necessários para resolvê-los, tais como tempo e armazenamento. Outras medidas de complexidade também são utilizadas, tais como a quantidade de comunicação, o número de portas em um circuito e o número de processadores. Uma das funções da teoria da complexidade computacional é determinar os limites práticos do que os computadores conseguem e não conseguem fazer.

Campos relacionados com a ciência da computação teórica são a análise de algoritmos e a teoria da computabilidade. A análise de algoritmos é dedicada a analisar a quantidade de recursos necessários para um certo algoritmo resolver um problema, enquanto a teoria da computabilidade faz uma pergunta mais geral sobre todos os algoritmos existentes que poderiam ser usados para resolver o mesmo problema. Ele tenta classificar os problemas que podem ou não serem resolvidos com os recursos devidamente restritos.

Por sua vez, impondo restrições sobre os recursos disponíveis é o que distingue a complexidade computacional da teoria da computabilidade: a segunda pergunta que tipos de problemas podem, em princípio, ser resolvidos através de algoritmos.

Para provar que P = NP, basta encontrar um algoritmo polinomial para um único problema NP-completo. Mas até hoje ninguém conseguiu ainda realizar esse feito.