Pontifícia Universidade Católica de Minas Gerais
Curso de Ciência da Computação
Disciplina: Algoritmos e Estruturas de Dados II

Lista de Exercícios

Fundamentos de Análise de Complexidade

---

1. Determine a forma fechada para cada somatório a seguir:

   (a) $\sum_{i=0}^{n} 3i + 4$

   (b) $\sum_{i=0}^{n-1} (2i + 3)^2$

   (c) $\sum_{i=1}^{n} i + 2 \cdot 5^i$

2. Prove por indução matemática as formas fechadas encontradas para os somatórios do exercício 1.

3. O somatório de Gauss é dado por:

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

   Em aula, a forma fechada do somatório foi obtida a partir da soma dos termos de uma PA. De forma alterativa, aplique a P2 ao somátorio de Gauss para obter a sua forma fechada. Dica: lembre-se que foi preciso aplicar a P2 sobre de $\sum_{i=0}^{n} i^3$ para determinar $\sum_{i=0}^{n} i^2$.

4. Aplique a P2 para determinar a forma fechada do somatório a seguir:

$$P(n) = \sum_{i=0}^{n} i^3$$

5. Considere as expressões abaixo como sendo a complexidade de tempo de um algoritmo para resolver um problema de tamanho n. Para cada algoritmo, indique o termo dominante, a ordem de complexidade em notação Θ e se é Ω e O para as seguintes classes: $\lg n$, n, $n \lg n$, n^2 e n^3.

   (a) $5 + 0,001n^3 + 0,025n$

   (b) $500n + 100n^{1,5} + 50n \log_{10} n$

   (c) $0,3n + 5n^{1,5} + 2,5n^{1,75}$

   (d) $n^2 \lg n + n \lg^2 n$

   (e) $n \log_3 n + n \lg n$

   (f) $3 \log_8 n + \lg \lg \lg n$

(g) $100n + 0,01n^2$

(h) $0,01n + 100n^2$

(i) $2n + n^{0,5} + 0,5n^{1,25}$

(j) $0,01n \lg n + n \lg^2 n$

(k) $100n \log_3 n + n^3 + 100n$

(l) $0,003 \log_4 n + \lg \lg n$

6. Um método de ordenação com complexidade $\Theta(n \log n)$ gasta exatamente 1 milissegundo para ordenar 1.000 itens de dados. Supondo que o tempo $T(n)$ para ordenar n itens seja diretamente proporcional a $n \log n$, ou seja, $T(n) = c \cdot n \log n$, estime quanto tempo esse método levará para ordenar 1.000.000 de itens.

7. Um algoritmo quadrático em tempo de processamento gasta 1 ms para processar 100 itens de dados. Quanto tempo será gasto para processar 5000 itens de dados?

8. Dado o código abaixo, apresente a função de complexidade para a operação "process()" e seu custo computacional usando a notação Θ. Em seguida, responda (e justifique) se esse custo é $O(n^3)$ e $\Omega(n^2 \lg n)$.

```
for (int i = 0 ; i < n; i++) {
    for (int j = 0 ; j < n; j++) {
        process( );
    }
}

if (n > 10) {
    process ( ) ;
    for (int k = n; k > 1 ; k /= 2) {
        process( );
    }
}
else {
    for (int m= 0 ; m< n; m++) {
        process( );
        process( );
    }
}

for (int p = 0 ; p < n * n; p++) {
    process( );
}
```

9. Dado o código abaixo, apresente a função de complexidade para a operação "func()" e seu custo computacional usando a notação Θ. Em seguida, responda (e justifique) se esse custo é $O(n^2 \lg n)$ e $\Omega(n^2 \lg n)$.

```java
if( (n < a + func( )) == true || func( ) < 2 * func( ) + c ){
    func( ); func( );
}
else {
    func( ); func( ); func( ); func( );
}

for(int i = n-1; i > 3; i--){
    func( );
}

for(int i = n; i > 0; i = i >> 1){
    func( );
}

for(int i = 0 ; i < n-1; i++){
    func( );
    for(int j = 0 ; j < n; j++){
        func( );
    }
}

for(int i = n-4 ; i > 1; i /= 2){
    func ( );
}

Random gerador = new Random( );
gerador.setSeed(4);
for(int i = 2 ; i < n; i++){
    if(Math.abs(gerador.nextInt( )) % 5 == 1 ||
            Math.abs(gerador.nextInt( )) % 5 == 2) {
        func( );
        func( );
    }
    else if(Math.abs(gerador.nextInt( )) % 5 == 3) {
        func( );
    }
}
```

10. Analise o código a seguir e determine a função de complexidade, considerando a subtração como operação relevante. Faça a prova por indução matemática. Apresente a ordem de complexidade usando notação Θ. Verifique, experimentalmente, a complexidade obtida.

```java
for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++) {
        b = b -1;
        c --;
    }
    d = c - d;
}
```

11. Determine a função de complexidade para o número de adições do algoritmo que se segue e prove usando indução matemática. Apresente a ordem de complexidade usando notação Θ. Verifique, experimentalmente, a complexidade obtida.

```
for (int i = 1; i <= n; i++) {
    for (int j = i; j <= n; j++) {
        a+= 1;
    }
}
b += ++a;
```

12. Analise o código a seguir e determine a função de complexidade, considerando a mutiplicação como operação relevante. Faça a prova por indução matemática. Apresente a ordem de complexidade usando notação Θ. Verifique, experimentalmente, a complexidade obtida.

```
for (int i = 0; i < n; i++) {
    for (int j = i; j < n; j++) {
        for (int k = 0; k < j; k++) {
            b = b * 1;
            c = c * b;
            c--;
        }
    }
}
```

13. Determine a função de complexidade para o número de subtrações do algoritmo que se segue e prove usando indução matemática. Apresente a ordem de complexidade usando notação Θ. Verifique, experimentalmente, a complexidade obtida. Dica: note que o loop interno não será executado para valores menores de n.

```
for (int i = 5; i < n; i++) {
    --d;
    for (int j = i+1; j < n -2; j++) {
        b = b -1;
        c --;
    }
    e = (d + c - b) * 2;
}
```

Respostas

1. (a) $\frac{3n^2+11n+8}{2}$

   (b) $\frac{4n^3+12n^2+11n}{3}$

   (c) $\frac{n^2+n-5+5^{n+1}}{2}$

4. $P(n) = \sum_{i=0}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2$

5. (a) $\Theta(n^3)$

   (b) $\Theta(n^{1,5})$

   (c) $\Theta(n^{1,75})$

   (d) $\Theta(n^2 \lg n)$

   (e) $\Theta(n \lg n)$

   (f) $\Theta(\log_8 n)$

   (g) $\Theta(n^2)$

   (h) $\Theta(n^2)$

   (i) $\Theta(n^{1,25})$

   (j) $\Theta(n \lg^2 n)$

   (k) $\Theta(n^3)$

   (l) $\Theta(\log_4 n)$

6.
$$T(10^3) = c \cdot 10^3 \lg 10^3 = 10^{-3}$$

$$c = \frac{1}{10^6 \lg 10^3}$$

$$T(n) = \frac{n \lg n}{10^6 \lg 10^3}$$

$$T(10^6) = \frac{10^6 \lg 10^6}{10^6 \lg 10^3} = \frac{\lg 10^6}{\lg 10^3} = \frac{\log 10^6}{\log 10^3} = \frac{6}{3} = 2 \text{ segundos}$$

8. Melhor caso: $[n^2] + [\lfloor \lg(n) \rfloor + 1] + [n^2]$

   Pior caso: $[n^2] + [2n] + [n^2]$

   Melhor e Pior Caso: $\Theta(n^2)$, o que implica que é $O(n^3)$ e não é $\Omega(n^2 \lg(n))$.

9. MELHOR: $3 + [n-4] + [\lfloor \lg(n) \rfloor + 1] + [(n-1) * (n+1)] + \lfloor \lg(n-4) \rfloor + 0$

   PIOR: $7 + [n-4] + [\lfloor \lg(n) \rfloor + 1] + [(n-1) * (n+1)] + \lfloor lg(n-4) \rfloor + [(n-2) \times 2]$

   MELHOR E PIOR: $\Theta(n^2) = O(n^2) = O(n^2 \lg(n)) \neq \Omega(n^2 \lg(n))$

10.
$$P(n) = \sum_{i=0}^{n-1} 2i + 1 = n^2$$

```c
#include <stdio.h>

void foo(int n)
{
  int experimental = 0;
  int b = 100, c = 100, d = 100;
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < i; j++)
    {
      b = b - 1;
      c--;
      experimental += 2;
    }
    d = c - d;
    experimental += 1;
  }

  double analitico = n * n;
  printf("n = %3d | experimental = %4d | analitico = %7.2f | confere = %d\n", n,
    experimental, analitico, (double)experimental == analitico);
}

int main()
{
  for (int n = 1; n < 11; n++)
  {
    foo(n);
  }

  return 0;
}
```

11.

$$P(n) = \left[\sum_{i=1}^{n} 2(n-i+1)+1\right] + 2 = n^2 + 2n + 2$$

12.

$$P(n) = \sum_{i=0}^{n-1}\sum_{j=i}^{n-1} 2j = \frac{4n^3 - 4n}{6}$$

13.

$$P(n) = \begin{cases} 3(n-5), & 5 \le n \le 8 \\ 3(n-5) + \frac{3(n^2-15n+56)}{2}, & n \ge 9 \end{cases}$$