Projeto e Análise de Algoritmos

Engenharia da Computação - Aula Remota – Prof.º Philippe Leal

Exercícios Individuais – Encontro 06 – Chave de Resposta

1) Calcule a complexidade **local** e **assintótica** de cada função (na Linguagem C) solicitada:

a) Considere dois números inteiros ***a*** e ***n*** (*n* ≥ 1). Faça uma função para calcular ***S*** de acordo com a expressão abaixo:

$$S=\sum_{i=1}^{n} \frac{i}{a^i}$$

```
int funcaoA(int a, int n){
   float S = 0.0, i;          1
   for(i = 1; i <= n; i++){   n+1
      S += i/pow(a,i);        n
   }
   return S;                  1
}
```

T(n) = 1 + (n+1) + n + 1

= 2n + 3

- **Complexidade Local**: 2n + 3
- **Complexidade Assintótica**: O(n)

b) Considere dois número inteiros e positivos *n* (*n* ≥ 1) e *m* (*m* ≥ 1). Faça uma função para calcular ***S*** de acordo com a expressão abaixo:

$$S = \sum_{i=1}^{n} \sum_{j=1}^{m} i \times j$$

```
int funcaoB(int n, int m){
   double S=0.0;                  1
   for(int i = 1; i <= n; i++){   n+1
      for(int j = 1; j <= m; j++){   n(m+1)
         S += i*j;                nm
      }
   }
   return S;                      1
}
```

T(n,m) = 1 + (n+1) + n(m+1) + nm + 1

= (n+1) + nm + n + nm + 2

= 2nm + 2n + 3

- **Complexidade Local**: 2nm + 2n + 3
- **Complexidade Assintótica**: O(nm)

c) Considere um número inteiro e positivo *n* (*n* > 0). Faça uma função para retornar 1, se *n* for **primo**, ou retornar 0, caso contrário.

```
int funcaoC(int n){
   int cont = 0;                  1
   for(int i = 1; i <= n; i++){   n+1
      if((n % i) == 0)            n
         cont++;                  n
   }
   if(cont == 2)                  1
      return 1;                   1
   else
      return 0;
}
```

$T(n) = 1 + (n+1) + n + n + 1 + 1$

$= 3n + 4$

- **Complexidade Local**: $3n + 4$

- **Complexidade Assintótica**: $O(n)$

d) Considere a matriz $A_{n \times m}$ com números inteiros. Faça uma função para retornar o que se pede na expressão abaixo:

$$\underset{1 \leq j \leq m}{Min} \sum_{i=1}^{n} a_{ij}$$

```
int funcaoD(int A[][m]){
   int soma, min;                 1
   for(int j = 0; j < m; j++){    m+1
      soma = 0;                   m
      for(int i = 0; i < n; i++){ m(n+1)
         soma += A[i][j];         nm
      }
      if(j == 0)                  m
         min = soma;              1
      if(soma < min)              m
         min = soma;              m-1
   }
   return min;                    1
}
```

T(n,m) = 1 + (m+1) + m + m(n+1) + nm + m + 1 + m + (m-1) + 1

= 2mn + 6m + 3

- **Complexidade Local**: 2mn + 6m + 3
- **Complexidade Assintótica**: O(mn)

e) Considere a matriz $B_{n \times m}$ de com números inteiros. Faça uma função para retornar o que se pede na expressão abaixo:

$$\underset{1 \le i \le n}{Max} \prod_{j=1}^{m} b_{ij}$$

```
int funcaoE(int B[][m]){
    double produto, max;          1
    for(int i = 0; i < n; i++){   n+1
        produto = 1.0;            n
        for(int j = 0; j < m; j++){   n(m+1)
            produto *= B[i][j];   nm
        }
        if(i == 0)                n
            max = produto;        1
        if(produto > max)         n
            max = produto;        n-1
    }
    return max;                   1
}
```

T(n,m) = 1 + (n+1) + n + n(m+1) + nm + n + 1 + n + (n-1) + 1

= 2nm + 6n + 3

- **Complexidade Local**: 2nm + 6n + 3
- **Complexidade Assintótica**: O(nm)

f) Considere a matriz $C_{n \times n}$ de com números inteiros. Faça uma função para retornar o que se pede na expressão abaixo:

$$S = \sum_{\substack{i,j=1 \\ i<j}}^{n} C_{ij}$$

```
int funcaoF(int C[][n]){
  int S = 0;                    1
  for(int i = 0; i < n-1; i++){     n
    for(int j = i+1; j < n; j++){   n(n+1)/2 - 1
      S += C[i][j];               n(n-1)/2
    }
  }
  return S;                     1
}
```

Loop j

i	**j**	**#vezes**
0	1 ... n-1	(n-1) + 1 (*sair do loop*) = n
1	2 ... n-1	(n-2) + 1 (*sair do loop*) = n-1
2	3 ... n-1	(n-3) + 1 (*sair do loop*) = n-2
⋮	⋮	⋮
n-2	n-1 ... n-1	1 + 1 (*sair do loop*) = 2
TOTAL		$\sum_{i=2}^{n} i = \frac{n(n+1)}{2} - 1$

Somatório

i	**j**	**#vezes**
0	1 ... n-1	n-1
1	2 ... n-1	(n-2)
2	3 ... n-1	(n-3)
⋮	⋮	⋮
n-2	n-1 ... n-1	1
TOTAL		$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$

T(n) = 1 + n + n(n+1)/2 – 1 + n(n-1)/2 + 1

= $(n^2 + n + n^2 - n)/2 + n + 1$

= $n^2 + n + 1$

- **Complexidade Local:** $n^2 + n + 1$
- **Complexidade Assintótica:** $O(n^2)$