DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Estruturas de Dados

Técnicas de Análise

Professores: Luiz Chaimowicz e Raquel Prates

Análise de Algoritmos

- Determinar a **função de complexidade** do tempo de execução de um programa pode ser um problema matemático difícil;

- Determinar a **ordem de complexidade** do tempo de execução, sem preocupação com o valor da constante envolvida, pode ser uma tarefa mais simples.

Análise do Tempo de Execução

- Comando de atribuição, de leitura ou de escrita: O(1).
  - `a = 0;`
  - `v[0] = 12;`
  - `b = a + 1;`
  - `return b;`

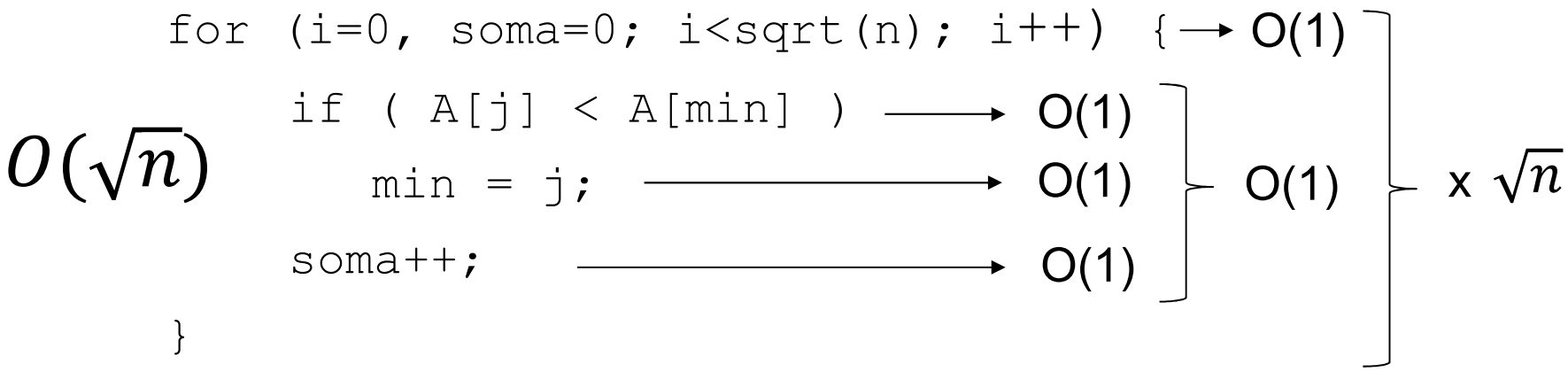
Análise do Tempo de Execução

- Comando de decisão:
  - Tempo dos comandos dentro do condicional, mais tempo para avaliar a condição que, em geral, é O(1).
  - Eventualmente, se a complexidade dos comandos seja diferente caso a condição seja **V** ou **F** pode haver melhor caso e pior caso

```
if ( A[j] < A[min] )
    min = j;
else
    return min;
```

Análise do Tempo de Execução

- Laço:
  - Inicialização **+** número de iterações **x** (tempo de avaliar a condição de parada **+** tempo de execução do corpo do laço **+** tempo para incrementar o contador).



Análise do Tempo de Execução

- Sequência de comandos:
  - Determinado pelo maior tempo de execução de qualquer comando da sequencia.

$O(n)$

```
int Somatorio(int n)
    int soma, i;

    soma = 0;                    → O(1)
    i = 1;                       → O(1)
    while (i<=n) {               → O(1)
       soma += i;                → O(1)   } x n = O(n)
       i++;                      → O(1)
    }
    return soma;                 → O(1)
}
```

Análise do Tempo de Execução

- Cada chamada de uma função deve ter seu tempo computado separadamente, iniciando pelos que não chamam outros procedimentos.

- Avalia-se então os chamam os já avaliados (utilizando os tempos desses).

- O processo é repetido até chegar no programa principal (função main)

Análise do Tempo de Execução

- Qual a **ordem de complexidade** da função mostrada abaixo

```
void FazAlgo(int n){
    int x, i, j;

    scanf("%d",&x);                    → O(1)
    if(x < 10)                         → O(1)
        return;                        → O(1)
    else
      for(i=0; i<n; i++) {             → O(1)
          j = 0;                       → O(1)
          while(j < n){                → O(1)
              x++;                     → O(1)
              j++;                     → O(1)
          }                            } x n = O(n)
      }                                } x n = O(n²)
}
```

Melhor Caso = **O(1)**
Pior Caso = **O(n^2)**

Análise do Tempo de Execução

- Qual a **ordem de complexidade** da função mostrada abaixo

Melhor Caso = **O(n)**
Pior Caso = **O(n^2)**

```
int Exemplo1(int n){
      int i, a;
      a=0;                          → O(1)
      for (i=0; i<n; i++)           → O(1)
            a+=i;                   → O(1)
      FazAlgo(n);                   → O(1) Melhor, O(n²) Pior
      return a;                     → O(1)
}
```

x n = O(n)

Exemplo: Algoritmo de Ordenação

Algoritmo da Seleção

- Seleciona o menor elemento do conjunto.
- Troca este com o primeiro elemento A[0].
- Repita as duas operações acima com os *n* - 1 elementos restantes, depois com os *n* - 2, até que reste apenas um.

```
void Ordena(Vetor A) {
  /*ordena o vetor A em ordem ascendente*/
    int i, j, min, x;
    for (i = 0; i < n-1; i++) {
        min = i;
        for (j = i + 1; j < n; j++)
            if ( A[j] < A[min] )
                min = j;
        /*troca A[min] e A[i]*/
        x = A[min];
        A[min] = A[i];
        A[i] = x;
    }
}
```

Qual a função de complexidade para o número de comparações de elementos do vetor? $\frac{n(n-1)}{2}$

Qual a função de complexidade para o número de trocas? $n-1$

Qual a Ordem de Complexidade da função Ordena? $O(n^2)$