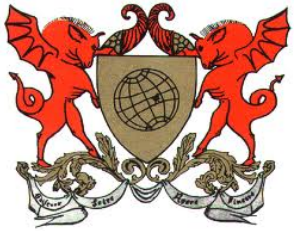
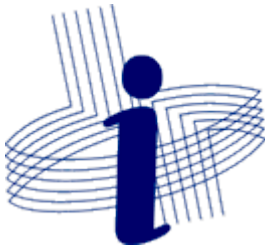


Universidade Federal de Viçosa
Departamento de Informática
Centro de Ciências Exatas e Tecnológicas



INF 112 – Programação 2
Aula 3

Introdução à Análise de Algoritmos

Aula baseada nos slides disponibilizados por Anany Levitin (autor do livro: *The Design and Analysis of Algorithms*) e nas aulas do Prof. Fabio Ribeiro

Introdução à Análise de Algoritmos

- Dado um algoritmo, o que podemos analisar nele?
  - Corretude
  - Eficiência de tempo
  - Eficiência de espaço
  - Otimalidade
  - ...
- Possíveis abordagens:
  - Análise teórica
  - Análise empírica

Introdução à Análise de Algoritmos

- Nosso foco será a análise teórica de tempo de processamento.
- Em geral, quanto maior o tamanho da “entrada” do algoritmo, maior o tempo de processamento.
- Assim, vamos analisar o algoritmo em função do “tamanho” *n* da entrada.

Introdução à Análise de Algoritmos

- Como avaliamos o tempo de processamento? Poderíamos estimar o tempo (segundos, milisegundos, etc) de processamento de uma implementação do algoritmo.
- Problemas:
  - Dependência do computador onde o programa seria executado.
  - Dependência da qualidade implementação do programa.
  - Dependência da linguagem de programação/compilador.
  - Como medir o tempo de execução? cpu time? wall time?
  - etc.

Introdução à Análise de Algoritmos

- Outra solução: contar quantas vezes cada operação do algoritmo é executada.
- Poderia ser uma forma bem precisa de se analisar o algoritmo, mas é trabalhosa!
- Normalmente se deseja apenas uma estimativa do comportamento do algoritmo em função do tamanho da entrada. Para isso, normalmente não é necessário avaliar todas as operações.

Introdução à Análise de Algoritmos

- Assim, o que se faz é calcular o número de vezes que a operação básica do algoritmo é executada.
  - Operação básica: a mais importante e que mais contribui para o tempo de processamento.
- Exercício: no código abaixo, qual é a operação básica?

```
double x = 3.14*5.2*sqrt(10.5)*pow(23.2,44.2);
int soma = 0;
for(int i=0;i<10000;i++) {
        double y = 3.14*5.2*sqrt(10.5)*pow(23.2,44.2);
        for(int j=0;j<10000;j++)
                soma++;
}
```

Introdução à Análise de Algoritmos

- A operação básica é a operação mais executada no algoritmo (em algoritmos iterativos, é a que está no laço mais interno).
- Em geral, é ela que pesa mais (MUITO MAIS!!!) no tempo de execução do algoritmo. Veja o exemplo no próximo slide!

```
int soma = 0;
t0 = clock();
for(int i=0;i<2000;i++)
        for(int j=0;j<2000;j++)
                soma += 3.14*5.2*sqrt(soma)*pow(soma/100,44.2);

cout << "Tempo 1 (ms): " << ((clock()-t0)*1.0/CLOCKS_PER_SEC)*1000 << endl;

soma = 0;
t0 = clock();
for(int i=0;i<2000;i++)
        for(int j=0;j<2000;j++)
                for(int k=0;k<2000;k++)
                        soma++;

cout << "Tempo 2 (ms): " << ((clock()-t0)*1.0/CLOCKS_PER_SEC)*1000 << endl;
```

```
int soma = 0;
t0 = clock();
for(int i=0;i<2000;i++)
        for(int j=0;j<2000;j++)
                soma += 3.14*5.2*sqrt(soma)*pow(soma/100,44.2);

cout << "Tempo 1 (ms): " << ((clock()-t0)*1.0/CLOCKS_PER_SEC)*1000 << endl;

soma = 0;
t0 = clock();
for(int i=0;i<2000;i++)
        for(int j=0;j<2000;j++)
                for(int k=0;k<2000;k++)
                        soma++;

cout << "Tempo 2 (ms): " << ((clock()-t0)*1.0/CLOCKS_PER_SEC)*1000 << endl;
```

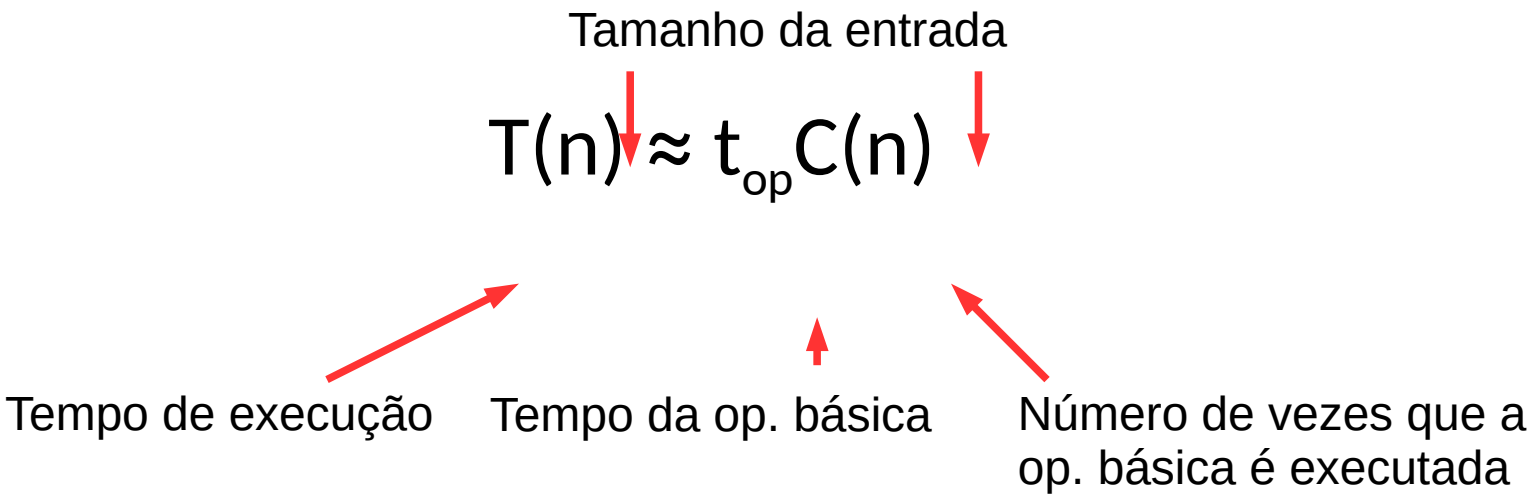
```
Tempo 1 (ms): 140
Tempo 2 (ms): 19850
```

Introdução à Análise de Algoritmos

Problema	Medida típica da entrada	Operação básica
Buscar por uma chave em uma lista	Número de elementos na lista: n	Comparação das chaves
Multiplicação de duas matrizes	Dimensões das matrizes ou número de células	Multiplicação dos dois números
Verificar a primalidade de um número n	Tamanho de n: número de dígitos (na representação binária)	Divisão
Problema de grafos	Número de vértices e/ou arestas	Visitar um vértice e/ou atravessar uma aresta

Introdução à Análise de Algoritmos

- Assim, o tempo de execução do algoritmo pode ser estimado da seguinte forma:



Introdução à Análise de Algoritmos

- Suponha o seguinte código:

```
for(int i=0;i<n;i++)
        for(int j=0;j<i;j++)
                soma++;
```

- Qual é a operação básica dele?
- Quanto vale C(n) ? (ou seja, quantas vezes a operação básica é executada)
- Quanto vale t_{op} ?

Introdução à Análise de Algoritmos

- Suponha o seguinte código:

```
for(int i=0;i<n;i++)
        for(int j=0;j<i;j++)
                soma++;
```

- Qual é a operação básica dele? R: incremento da soma
- Quanto vale C(n) ? (ou seja, quantas vezes a operação básica é executada) R: n(n-1)/2
- Quanto vale t_{op} ? R: ???

Introdução à Análise de Algoritmos

- Suponha o seguinte código:

```
for(int i=0;i<n;i++)
        for(int j=0;j<i;j++)
                soma++;
```

- Qual é a operação básica dele? R: incremento da soma
- Quanto vale C(n) ? (ou seja, quantas vezes a operação básica é executada) R: n(n-1)/2
- Quanto vale t_{op} ? R: ???
- Se dobrarmos o tamanho da entrada, quantas vezes mais tempo o algoritmo gasta?

Introdução à Análise de Algoritmos

- Se dobrarmos o tamanho da entrada, quantas vezes mais tempo o algoritmo gasta?
- $n(n-1)/2 = ½ (n^2 – n)$
- Para n grande, o n^2 “pesa” muito mais no tempo de execução! podemos considerar apenas o $½n^2$ nos cálculos!
- Assim, podemos ver que o tempo fica 4 vezes maior! (faça os cálculos!)
- Note que isso independe do valor de t_{op} !

Introdução à Análise de Algoritmos

- Na análise de algoritmos estamos preocupados apenas com a taxa de crescimento do mesmo (ou "ordem de complexidade")... Isso é analisado supondo que a entrada é muito grande.
- Ao analisar a taxa de crescimento, desconsideramos constantes e monômios de grau menor.
- Assim, se $C(n) = 5n^3 + 3n^2 + n + 100000$, aproximamos C(n) assim: $C(n) \approx n^3$
- Se n for pequeno, essa aproximação não fica muito boa... mas normalmente estamos interessados em valores grandes de n. Por que?

Introdução à Análise de Algoritmos

- Exemplo de como essa aproximação é válida.
- C(n) = n(n-1)/2 + 2n + 1

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

n	Tempo (s)	Tempo estimado de forma “grosseira”*
10	0	-
100	0	-
1000	0	-
10000	0.130	0.130
20000	0.490	0.520
40000	1.950	2.080
80000	7.800	8.320
160000	31.720	33.280

Introdução à Análise de Algoritmos

- Exemplo de como essa aproximação é válida.
- C(n) = n(n-1)/2 + 2n + 1

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

n	Tempo (s)	Tempo estimado de forma “grosseira”*
10	0	-
100	0	-
1000	0	-
10000	0.130	0.130
20000	0.490	0.520
40000	1.950	2.080
80000	7.800	8.320
160000	31.720	33.280

*Com base no tempo de 100000 e supondo que quando n dobra, o tempo quadruplica

Introdução à Análise de Algoritmos

- A seguinte tabela (do livro do Levitin) mostra o crescimento de várias funções

TABLE 2.1 Values (some approximate) of several functions important for analysis of algorithms

n	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n	$n!$
10	3.3	10^1	$3.3 \cdot 10^1$	10^2	10^3	10^3	$3.6 \cdot 10^6$
10^2	6.6	10^2	$6.6 \cdot 10^2$	10^4	10^6	$1.3 \cdot 10^{30}$	$9.3 \cdot 10^{157}$
10^3	10	10^3	$1.0 \cdot 10^4$	10^6	10^9		
10^4	13	10^4	$1.3 \cdot 10^5$	10^8	10^{12}		
10^5	17	10^5	$1.7 \cdot 10^6$	10^{10}	10^{15}		
10^6	20	10^6	$2.0 \cdot 10^7$	10^{12}	10^{18}		

Introdução à Análise de Algoritmos

- Exemplo: suponha que um computador execute 1 bilhão de operações por segundo (um valor razoável) e que a operação básica seja executada 2^n vezes. Temos que:

$$T(n) \approx 10^{-9} \times 2^n$$

Assim:

n	Tempo
10	~0s
20	0.001s
30	1s
40	1099s
50	13 dias!
100	4×10^{13} anos!

Introdução à Análise de Algoritmos

- Note que o algoritmo do exemplo é tão ineficiente que se conseguíssemos juntar o poder de processamento de 1000 computadores (supondo que cada um deles seja 1000 vezes mais rápido do que o computador do exemplo), ainda assim o algoritmo levaria 40000000 anos para terminar!

n	Tempo
10	~0s
20	0.001s
30	1s
40	1099s
50	13 dias!
100	$4x10^{13}$ anos!

Introdução à Análise de Algoritmos

- Dependendo do algoritmo, o tempo de execução pode variar em função da entrada (não apenas do tamanho dela).
- Assim, avaliamos o tempo de execução em cenários de “melhor caso”, “caso médio” e “pior caso”.
- O melhor caso representa quantas vezes a operação básica será executada na entrada “mais fácil de ser processada”.
- O pior caso é para a entrada mais difícil.
- O caso médio é o tempo esperado supondo uma entrada qualquer (supondo que todas as entradas possuem uma probabilidade igual de ocorrer).

Introdução à Análise de Algoritmos

- Exemplo:
- Busca sequencial de um elemento em um vetor (suponha que o elemento buscado sempre esteja no vetor).
- Qual seria o número de operações no melhor caso?
- Qual seria o número de operações no pior caso?
- Qual seria o número de operações no caso médio?