Estrutura de Dados I

Análise de complexidade de algoritmos

Bruno Prado

Departamento de Computação / UFS

Introdução

- O que é eficiência?
  - Tempo ou esforço empregado para realizar algo
  - Otimização do uso dos recursos

$$\uparrow \textit{Eficiência} \longleftrightarrow \textit{Tempo} \downarrow$$

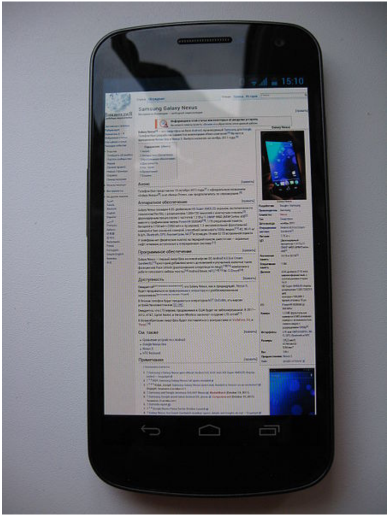
$$\uparrow \textit{Eficiência} \longleftrightarrow \textit{Recursos} \downarrow$$

Introdução

- Qual a história e por que era importante?
  - Os recursos computacionais eram muito limitados
  - Grande consumo de potência e uso compartilhado

Introdução

- Por que hoje é importante?
  - Restrições de custo
  - Baixo consumo de potência



Introdução

- Quais os tipos de eficiência ou de complexidade computacional?
  - Tempo
    - Número de passos realizados
  - Espaço
    - Memória alocada
    - Armazenamento em disco

Complexidade de Tempo

- Indica quão rápido um algoritmo executa
- Ordenação simples da sequência de n números $a_1, a_2, \dots, a_{n-1}, a_n$ para gerar uma sequência ordenada $a'_1, a'_2, \dots, a'_{n-1}, a'_n$

$$\begin{array}{|c|c|c|c|c|}\hline a_1 & a_2 & \quad\dots\quad & a_{n-1} & a_n \\ \hline \end{array}$$

$$\swarrow \downarrow \searrow$$

$$\begin{array}{|c|c|c|c|c|}\hline a'_1 & a'_2 & \quad\dots\quad & a'_{n-1} & a'_n \\ \hline \end{array}$$

- Número de passos realizados
  - $(n-1) + (n-2) + \cdots + 2 + 1 = \frac{(n-1)[1+(n-1)]}{2} \approx n^2$

Complexidade de Tempo

- Como medir o tempo de um procedimento?
  - Conceito de constante
  - Análise depende do tamanho da entrada *n*

```c
#include <stdio.h>
...
void procedimento(int n) {
	c1();
	for(int i = 0; i < n; i++) {
		c2();
		for(int j = 0; j < n; j++) {
			c3();
		}
	}
}
```

Complexidade de Tempo

- Como medir o tempo de um procedimento?
  - Expressão em função do valor de n
  - Procedimentos $c1$, $c2$ e $c3$ não dependem de n

$$
\begin{aligned}
procedimento(n) &= c1 + n \times [c2 + (n \times c3)] \\
&= c1 + n \times c2 + n^2 \times c3
\end{aligned}
$$

Complexidade de Tempo

- Cálculo do tempo com os valores médios das constantes
  - Entrada de tamanho 1.000
  - Valores de $c1 = 200\ ns,\ c2 = 150\ ns$ e $c3 = 250\ ns$

$$
\begin{aligned}
procedimento(1000) &= 200 + 10^3 \times 150 + 10^6 \times 250\ ns \\
&= 200 + 1,5 \times 10^5 + 2,5 \times 10^8\ ns \\
&= 0,00000200 \times 10^8 + 0,0015 \times 10^8 + \\
&\quad 2,5 \times 10^8\ ns \\
&= 2,50150200 \times 10^8\ ns \approx 0,25\ s
\end{aligned}
$$

Complexidade de Tempo

- Cálculo do tempo com os valores médios das constantes
  - Quanto maior o valor do tamanho da entrada n, maior é o domínio do fator de maior grau da função
  - Para um valor de n suficientemente grande $n > n_0$

$$procedimento(n) \leq g(n)$$

$$g(n) = c \times n^2$$

Complexidade de Tempo

- Aplicando o conceito de limite (análise assintótica)
  - Valores das constantes dependem da máquina
  - Com $n \to \infty$ se analisa a ordem das funções

$$\lim_{n\to\infty} \frac{procedimento(n)}{g(n)} = \begin{cases} 0 & procedimento(n) < g(n) \\ k & procedimento(n) = g(n) \\ \infty & procedimento(n) > g(n) \end{cases}$$

Complexidade de Espaço

- Indica quanto de espaço de memória e de armazenamento são necessários
- Ordenação simples da sequência de n números $a_1, a_2, \ldots, a_{n-1}, a_n$ para gerar uma sequência ordenada $a'_1, a'_2, \ldots, a'_{n-1}, a'_n$

a_1	a_2	$\ldots$	a_{n-1}	a_n

$$\swarrow \downarrow \searrow$$

a'_1	a'_2	$\ldots$	a'_{n-1}	a'_n

- Número de posições de memória é de $2n$

Complexidade de Espaço

- Como medir o espaço alocado?
  - Procedimento para ordenação

```
#include <stdio.h>
...
void procedimento(int n) {
	int entrada() = (int*)(malloc(n * sizeof(int));
	int saida() = (int*)(malloc(n * sizeof(int));
	FILE* arquivo = fopen("arquivo.txt", "r");
	for(int i = 0; i < n; i++)
		fscanf(arquivo, "%i", entrada(i));
	fclose(arquivo);
	ordenar(saida, entrada);
}
```

Complexidade de Espaço

- Como medir o espaço alocado?
  - Expressão em função do valor de n
  - Constantes dependem do tamanho do dado

$$
\begin{aligned}
procedimento(n) &= c_1 \times n + c_1 \times n \\
&= 2c_1 \times n \\
&= c_2 \times n
\end{aligned}
$$

Complexidade de Espaço

- Cálculo do espaço alocado
  - Quanto maior o valor do tamanho da entrada n, maior é o domínio do fator de maior grau da função
  - Para um valor de n suficientemente grande $n > n_0$

$$procedimento(n) \leq g(n)$$

$$g(n) = c \times n$$

Complexidade de Espaço

- Aplicando o conceito de limite (análise assintótica)
  - Valores das constantes dependem da máquina
  - Com $n \to \infty$ se analisa a ordem das funções

$$\lim_{n \to \infty} \frac{procedimento(n)}{g(n)} = \begin{cases} 0 & procedimento(n) < g(n) \\ k & procedimento(n) = g(n) \\ \infty & procedimento(n) > g(n) \end{cases}$$

Ordem de Crescimento

- Tamanho de entrada n
- Complexidade versus número de passos

n	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n	$n!$
10^1	$3,3$	10^1	$3,3 \times 10^1$	10^2	10^3	10^3	$3,6 \times 10^6$
10^2	$6,6$	10^2	$6,6 \times 10^2$	10^4	10^6	$1,3 \times 10^{30}$	$9,3 \times 10^{157}$
10^3	10	10^3	$1,0 \times 10^4$	10^6	10^9	-	-
10^4	13	10^4	$1,3 \times 10^5$	10^8	10^{12}	-	-
10^5	17	10^5	$1,7 \times 10^6$	10^{10}	10^{15}	-	-
10^6	20	10^6	$2,0 \times 10^7$	10^{12}	10^{18}	-	-

Exemplo

- Calcular a complexidade de tempo e de espaço do algoritmo fatorial
  - Descrever sua implementação iterativa
  - Tudo deve ser claramente justificado

$$Fatorial(n) = \begin{cases} 1 & n = 0 \\ n \times Fatorial(n-1) & n > 0 \end{cases}$$

Notação O

- Necessidade de formalização da complexidade dos algoritmos
  - Notação matemática
  - Análise assintótica
- Notações para melhor caso (Ω), pior caso (O) e caso médio (Θ)
  - Definições e aplicações

Notação O

- Função de busca sequencial
  - Descrita pela equação $busca(n) = c_A + c_B \times n$

```
#include <stdio.h>
...
int busca(int elem, int v[], int n) {
	int r = -1;
	for(int i = 0; r == -1 && i < n; i++)
		if(v[i] == elem)
			r = i;
	return r;
}
```

Melhor Caso

- O que é a análise de melhor caso de um algoritmo?
  - Descreve a situação o menor número de passos são realizados
  - Estabelece um limitante inferior ou melhor caso

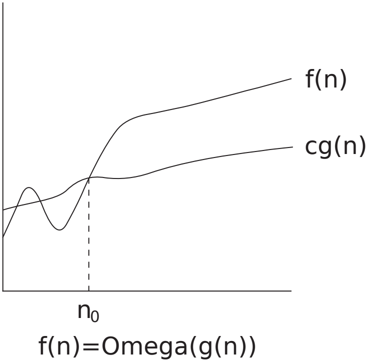
Melhor Caso

- O que é a análise de melhor caso de um algoritmo?
  - Descreve a situação o menor número de passos são realizados
  - Estabelece um limitante inferior ou melhor caso
- Busca sequencial pelo elemento 33
  - Primeira ocorrência
  - Vetor possui 1.000 elementos sem repetições

↓				
33	15	...	1	14
0	1		998	999

Melhor Caso

- Análise do melhor caso da busca sequencial
  - Existem constantes positivas c e n_0 tal que $0 \leq cg(n) \leq busca(n)$, para todo $n \geq n_0$
  - Aplicando a notação: $\Omega(busca(n)) = \Omega(g(n)) = \Omega(c_A + c_B) = c_{MC}$



Pior Caso

- O que é a análise de pior caso de um algoritmo?
  - Descreve a situação com maior número de passos
  - Estabelece um limitante superior

Pior Caso

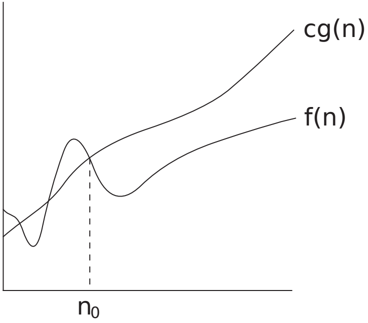
- O que é a análise de pior caso de um algoritmo?
  - Descreve a situação com maior número de passos
  - Estabelece um limitante superior

- Busca sequencial pelo elemento 14
  - Última ocorrência
  - Vetor possui 1.000 elementos sem repetições

33	15	...	1	14
0	1		998	999

Pior Caso

- Análise do pior caso da busca sequencial
  - Existem constantes positivas c e n_0 tal que $0 \leq busca(n) \leq cg(n)$, para todo $n \geq n_0$
  - Aplicando a notação: $O(busca(n)) = O(cg(n)) = O(c_A + c_B \times n) = c_{PC} \times n$



f(n)=O(g(n))

Notação O

- Propriedades da notação O
  - Termos constantes: $O(c) = O(1)$
  - Multiplicação por constantes: $O(c \times f(n)) = O(f(n))$
  - Adição: $O(f_1(n)) + O(f_2(n)) = O(|f_1(n)| + |f_2(n)|)$
  - Produto: $O(f_1(n)) \times O(f_2(n)) = O(f_1(n) \times f_2(n))$

Caso Médio

- Não confundir com caso prático ou real
  - Observa o comportamento real do algoritmo
  - Utiliza dados estatísticos

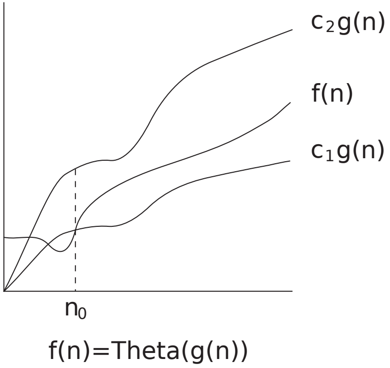
Caso Médio

- Não confundir com caso prático ou real
  - Observa o comportamento real do algoritmo
  - Utiliza dados estatísticos
- Busca sequencial por um elemento qualquer
  - São executadas na busca entre 1 e n iterações
  - Vetor possui 1.000 elementos sem repetições

$$
\begin{array}{ccccc}
 & & \downarrow & & \\
\hline
\multicolumn{1}{|c|}{33} & \multicolumn{1}{c|}{15} & \multicolumn{1}{c|}{\dots} & \multicolumn{1}{c|}{1} & \multicolumn{1}{c|}{14} \\
\hline
0 & 1 & & 998 & 999
\end{array}
$$

Caso Médio

- Análise do caso médio da busca sequencial
  - Existem constantes positivas c e n_0 tal que $0 \leq c_1 g(n) \leq busca(n) \leq c_2 g(n)$, para todo $n \geq n_0$
  - Aplicando a notação: $\Omega(c_{MC}) \leq busca(n) \leq O(n)$



Caso Médio

- Ordem exata de execução de um algoritmo $f(n)$

$$f(n) = \Omega(c_1 g(n)) \text{ e } f(n) = O(c_2 g(n))$$
$$\downarrow$$
$$f(n) = \Theta(g(n))$$

Exemplo

- Calcule a complexidade de tempo e espaço do código abaixo, utilizando as 3 notações vistas:

```
void procedimento(int n) {
    int a() = (int*)(malloc((n*n+10) * sizeof(int)));
    for(int i = 0; i <10; i++) a(i) = 1;
    for(int i = 0; i < n; i++) {
        int b = 3;
        for(int j = 0; j < n; j++) {
            a(i)(j) = b * a(i)(j);
            for(int k = 0; k <10; k++)
                a(i)(j)=a(i)(j) * a(k);
        }
    }
    for(int i = n; i < n * n; i++)
        a(i) = a(i) + 2;
}
```

Exemplo

- Descreva com suas palavras o que você entende por pior caso, melhor caso e caso médio.
- Por que é utilizada a análise assintótica?