Estruturas Avançadas de Dados I

Análise de Algoritmos

1

Introdução

- O que é um problema computacional?
  - É o conjunto de entradas válidas e qual é a relação entre a entrada e a saída desejada. Alguns problemas computacionais não possuem solução ótima.
  - Todo problema computacional é uma coleção de "casos particulares" que chamaremos de INSTÂNCIAS.

3

Introdução

- O que é uma instância?
  - É um exemplo concreto do problema, com dados específicos. Cada conjunto de dados de um problema define uma instância do problema;
  - Considere, por exemplo, o problema de determinar a média de dois números, digamos *a* e *b*. Uma instância desse problema consiste em determinar a média de 123 e 9876. Outra instância consiste em determinar a média de 22222 e 34343434.

UNISINOS
DESAFIE
O AMANHÃ.

4

Introdução

- Algoritmos para os problemas
  - Dizemos que um algoritmo resolve um problema se, ao receber a descrição de qualquer instância do problema, devolve uma solução da instância (ou informa que a instância não tem solução).

UNISINOS
DESAFIE
O AMANHÃ.

5

Introdução

- Mas o que é um Algoritmo?
  - Segundo Cormen (2009), um algoritmo é qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores como saída.
  - Algoritmos são utilizados para resolver diversos tipos de problemas!

UNISINOS
DESAFIE O AMANHÃ.

6

Introdução

- E a Estrutura de Dados?
  - Durante o processo de computação de sua saída, um algoritmo manipula dados obtidos de sua entrada. Quando os dados são dispostos e manipulados de forma homogênea, constituem um tipo abstrato de dados.
  - Um algoritmo é projetado em função de tipos abstratos de dados. Para implementá-los em uma linguagem de programação é necessário representá-los de alguma maneira;
  - Na representação do tipo abstrato de dados, emprega-se uma estrutura de dados.

UNISINOS
DESAFIE O AMANHÃ.

7

Introdução

- E a Estrutura de Dados?
  - Segundo Cormen (2009), uma estrutura de dados é um meio para armazenar e organizar dados com o objetivo de facilitar o acesso e as modificações;
  - A escolha de um algoritmo para resolver uma determinada instância de um problema depende também do tipo de estrutura utilizada para armazenamento dos dados, ou seja, a estrutura de dados.

UNISINOS
DESAFIE
O AMANHÃ.

8

Análise de Algoritmos

- Segundo Cormen (2009), analisar um algoritmo significa prever os recursos de que ele necessitará. Em geral, memória, largura de banda de comunicação ou hardware de computação são a preocupação primordial;
- Porém, é o tempo computacional que se deseja medir.

UNISINOS
DESAFIE
O AMANHÃ.

9

Análise de Algoritmos

- Dois aspectos importantes:
  - Um problema pode, geralmente, ser resolvido por diferentes algoritmos;
  - A existência de um algoritmo não implica, necessariamente, que este problema possa ser resolvido na prática.
- A análise de algoritmos pode ser definida como o estudo da estimativa de tempo de execução de algoritmos;
- O tempo computacional é determinado pelos seguintes aspectos:
  - Tempo para executar uma instrução ou passo;
  - A natureza do algoritmo;
  - O tamanho do conjunto de dados que constitui o problema.

10

Análise de Algoritmos

- É necessário ter uma forma de criar medidas de comparação entre algoritmos que resolvem um mesmo problema. Dessa forma, é possível determinar:
  - A viabilidade de um algoritmo;
  - Qual é o melhor algoritmo para a solução de um problema.
- O interessante é ter uma comparação relativa entre algoritmos.
  - Assumir que a execução de qualquer passo de um algoritmo leva a um tempo fixo e igual;
  - O tempo de execução de um computador particular não é interessante.

11

Análise de Algoritmos

- Qual a quantidade de recursos utilizados para resolver um problema?
  - Tempo
  - Espaço
- Expressar como uma função o tamanho do problema.
  - Como os requisitos crescem com o aumento do problema?
- Tamanho do problema:
  - Número de elementos a ser tratado
  - Tamanho dos elementos

UNISINOS
DESAFIE O AMANHÃ.

12

Análise de Algoritmos

- Considerar eficiência de tempo:
  - Número de operações expresso em termos do tamanho da entrada;
  - Se dobramos o tamanho da entrada, qual o tempo de resposta?

- Por que a eficiência é importante?
  - Velocidade de computação aumentou (hardware);
  - Crescimento de aplicações com o aumento do poder computacional;
  - Maior demanda por aumento na velocidade de computação.

UNISINOS
DESAFIE O AMANHÃ.

13

13

Análise de Algoritmos

- Quando a velocidade de computação aumenta, podemos tratar mais dados?
  - Um algoritmo toma n^2 comparações para ordenar "n" números;
  - Necessitamos de 1 segundo para ordenar 5 números (25 comparações);
  - Velocidade de computação aumenta de um fator de 100
  - Usando 1 segundo, podemos executar 100x25 comparações e ordenar 50 números com 100 vezes de ganho em velocidade, ordenamos apenas 10 vezes mais números.

UNISINOS
DESAFIE O AMANHÃ.

14

Análise de Algoritmos

Como medir a eficiência do algoritmo?

- Estudo experimental e/ou Benchmarking
  - Desenvolver um programa que implemente o algoritmo;
  - Executar o programa com diferentes instâncias;
  - Usar um método para obter medidas acuradas do tempo de execução real.

- Limitações:
  - Necessidade de implementar e testar o algoritmo;
  - Experimentos em cenários limitados;
  - Dependem de compilador, hardware, quantidade de memória, entre outros.

UNISINOS
DESAFIE O AMANHÃ.

15

Análise de Algoritmos

Como medir a eficiência do algoritmo?

- Análise assintótica
  - Usa uma descrição de alto nível dos algoritmos em vez de testar uma de suas implementações;
  - Leva em consideração todas as possíveis entradas;
  - Permite a avaliação da eficiência de algoritmos de uma forma que é independente do hardware e ambiente de software utilizado;
  - Modelo matemático.

UNISINOS
DESAFIE O AMANHÃ.

16

Análise Assintótica

- O tempo de execução de um algoritmo será representado por uma função de custo T, onde T(n) é a medida do tempo necessário para executar um algoritmo para um problema de tamanho n;
- Logo, T é chamada "função complexidade de tempo" do algoritmo;
- Se T(n) é a medida de memória necessária para a execução de um algoritmo, então T é chamada de "função de complexidade de espaço";
- Conforme Ziviani (2004), é importante enfatizar que T(n) não representa diretamente o tempo de execução, mas o número de vezes que certa operação relevante é executada.

UNISINOS
DESAFIE O AMANHÃ.

17

Análise Assintótica

- Ex.:

```
public int getMenor(int[] a) {
        int menor = a[0];
        for(int i = 1; i < a.length; i++)
                if (a[i] < menor)
                        menor = a[i];
        return menor;
}
```

- A função de complexidade de tempo T(n) é o número de comparações entre os elementos do array a, visto que, para encontrar o menor elemento do array, é preciso mostrar que cada um dos n – 1 elementos é menor do que algum outro, e para isso, gasta-se pelo menos n – 1 comparações, logo:

$$T(n) = n - 1$$

18

Análise Assintótica

- Porém, algoritmos de busca sequencial, não se comportam dessa maneira, ou seja, uniforme!
- Neste caso, identificamos três casos:
  - Pior caso: corresponde ao maior tempo de execução sobre todas as entradas de tamanho n;
  - Melhor caso: corresponde ao menor tempo de execução sobre todas as entradas de tamanho n;
  - Caso médio: corresponde à média dos tempos de execução do algoritmo sobre todas as entradas de tamanho n.

19

Análise Assintótica - Consideração

- Pior caso: o pior caso ocorre quando o elemento está na última posição em uma busca sequencial (ou não existe);
- Melhor caso: quando os dados de entrada são tais que o algoritmo tem o menor trabalho possível para encontrar a solução. Exemplo: para determinar a posição de um elemento em um array, o melhor caso ocorre quando o elemento está na primeira posição em uma busca sequencial;
- Caso médio: se refere ao trabalho feito pelo algoritmo em média. Assumimos que os dados de entrada são randômicos;

O melhor caso é muito raro e, portanto, pouco significativo. Na maioria dos algoritmos utilizamos a métrica do pior caso. Em algumas situações podemos utilizar a métrica do caso médio. Existem situações em que o uso da métrica do pior caso é obrigatório (controle de tráfego aéreo, por exemplo).

UNISINOS DESAFIE O AMANHÃ.

22

Análise Assintótica - Exemplo

```
public int getPos(int[] a, int x) {
  boolean existe = false;                (1)
  int i = 0;                             (2)
  while(i < a.length && !existe) {       (3)
    if(a[i] == x)                        (4)
      existe = true;                     (5)
    i++;                                 (6)
  }
  if(!existe)                            (7)
    return -1;                           (8)
  else
    return i - 1;                        (9)
}
```

Número	Melhor caso	Pior caso	Caso médio
1	1	1	1
2	1	1	1
3	2	n+1	(n/2)+1
4	1	n	n/2
5	1	0	1
6	1	n	n/2
7	1	1	1
8	0	1	0
9	1	0	1
T(n)	9	3n + 5	(3n + 12)/2

UNISINOS DESAFIE O AMANHÃ.

23

Análise Assintótica

- Como exemplo, considere o número de operações de cada um dos dois algoritmos abaixo que resolvem o mesmo problema, como função de n.
  - Algoritmo 1: $f_1(n) = 2n^2 + 5n$ operações
  - Algoritmo 2: $f_2(n) = 500n + 4000$ operações

- Dependendo do valor de n, o Algoritmo 1 pode requerer mais ou menos operações que o Algoritmo 2. (Compare as duas funções para n = 10 e n = 256)

Qual algoritmo é mais eficiente?

UNISINOS
DESAFIE O AMANHÃ.

24

Análise Assintótica

- Um caso de particular interesse é quando n tem valor muito grande ($n \to \infty$), denominado comportamento assintótico; por isso, chamamos de Análise Assintótica!

- Os termos inferiores e as constantes multiplicativas contribuem pouco na comparação e podem ser descartados.

UNISINOS
DESAFIE O AMANHÃ.

25

Notação Assintótica

- **Notação O** (ou *Big O*)
- A notação O define um limite superior para a função, por um fator constante;

26

Notação Assintótica

- **Notação Ω** (ou *Big Ômega*)
- A notação **Ω** define um limite inferior para a função, por um fator constante.

28

Notação Assintótica

- Notação o (“O pequeno”) e ω (“ômega pequeno”)

- Possuem o mesmo conceito que o Big O e o Big Ômega, porém, os limites sempre serão inferiores, ou seja, < ou > e não ≤ ou ≥, ao ser comparado com a cg(n).

33

Notação Assintótica - Terminologia

- Terminologia de classes mais comuns de funções:

classe	nome
$O(1)$	constante
$O(\lg n)$	logarítmica
$O(n)$	linear
$O(n \lg n)$	$n \log n$
$O(n^2)$	quadrática
$O(n^3)$	cúbica
$O(n^k)$ com $k >= 1$	polinomial
$O(2^n)$	exponencial
$O(a^n)$ com $a > 1$	exponencial

34

Notação Assintótica - Ordens

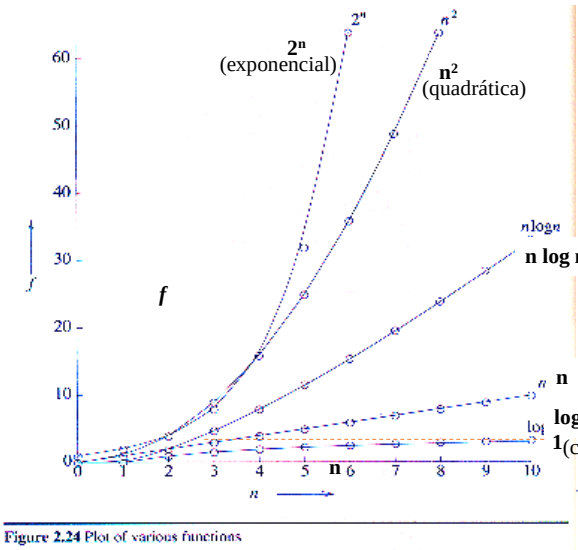

Figure 2.24 Plot of various functions

35

Comparação de Funções de Complexidade

Função de custo	Tamanho n					
	10	20	30	40	50	60
n	0,00001 s	0,00002 s	0,00003 s	0,00004 s	0,00005 s	0,00006 s
n^2	0,0001 s	0,0004 s	0,0009 s	0,0016 s	0,0,35 s	0,0036 s
n^3	0,001 s	0,008 s	0,027 s	0,64 s	0,125 s	0.316 s
n^5	0,1 s	3,2 s	24,3 s	1,7 min	5,2 min	13 min
2^n	0,001 s	1 s	17,9 min	12,7 dias	35,7 anos	366 seg
3^n	0,059 s	58 min	6,5 anos	3855 sec	10^8 sec	10^{13} sec

36

Análise Assintótica – Exercícios: Qual a Ordem?

1.1

	Melhor = Pior = Médio
int conta = 0;	
for(int i = 0; i < n; i++)	
for(int j = 0; j < n; j++)	
conta++;	

1.2

	Melhor = Pior = Médio
int conta = 0;	
int i = 1;	
while(i < n)	
{	
for(int j = 0; j < n ; j++)	
{	
conta++;	
}	
i++	
}	

UNISINOS
DESAFIE
O AMANHÃ.

37

Análise Assintótica – Exercícios

- 1.3. Dois algoritmos A e B possuem complexidade n^5 e 2^n, respectivamente. Você utilizaria o algoritmo B ao invés do A. Em qual caso? Desenvolva um gráfico com a análise dos dois algoritmos.

- 1.4. Baseando-se no algoritmo abaixo determine a ordem de complexidade. Podemos dizer que o algoritmo é $O(n^2)$? Justifique.

```
public void MaxMin(int[] a) {
        int max = a[0];
        int min = a[0];
        for(int i = 1; i < a.length; i++) {
            if(a[i] > max) max = a[i];
            if(a[i] < min) min = a[i];
    }
}
```

UNISINOS
DESAFIE
O AMANHÃ.

38

Referências Bibliográficas

- ASCENCIO, A. F. G; ARAÚJO, G. S. **Estruturas de Dados**. São Paulo: Pearson Prentice Hall, 2010. 432 p.
- CORMEN, Thomas H. et al. **Introduction to algorithms.** 3. ed. Cambridge: MIT, 2009. xix. 1292 p.
- FIGUEIREDO, Jorge. Apresentação **Ánálise e Técnicas de Algoritmos**.
- ZIVIANI, N. **Projeto de Algoritmos: com implementação em Pascal e C**. 2. ed. São Paulo: Pioneira Thomson Learning, 2004.

UNISINOS
DESAFIE
O AMANHÃ.

39

Prof. Márcio Garcia Martins
marciog@unisinos.br

Para anotar: *ao enviar e-mail sempre coloque o seguinte prefixo no assunto*
[EADI-ano-semestre] – Nome do aluno

UNISINOS
DESAFIE
O AMANHÃ.

40