Escola de Artes, Ciências e Humanidades – EACH-USP

**ACH2002 – Introdução à Ciência da Computação II**

Professor: Delano Medeiros Beder revisada pelo professor: Luciano Digiampietri
EACH – Segundo Semestre de 2011

**Lista de Exercícios**

# Classes, Herança e Interfaces

1. Defina uma classe `IntervaloDeTempo` cujos objetos representam um intervalo de tempo em número de horas, minutos e segundos. O construtor de objetos dessa classe deve receber como argumento um número inteiro positivo, representando o número de segundos decorridos desde o instante inicial 00:00:00 horas e retornar um objeto da classe `IntervaloDeTempo` correspondente. Por exemplo, a expressão `new IntervaloDeTempo(3500)` deve retornar um objeto que represente 0 horas, 58 minutos e 20 segundos. Crie também uma classe para testar a classe definida. (Adaptado de [1])

2. Defina uma classe `Pessoa` cujos objetos representam uma pessoa contendo as seguintes informações: nome, sexo, número do documento de identidade e data de nascimento da pessoa (dia, mês e ano). Defina um construtor de objetos dessa classe e também um método `idade()` para retornar a idade da pessoa (um número inteiro de anos) dado como argumento a data atual. Caso a data passada como argumento seja anterior à data de nascimento da pessoa o método deve retornar -1. (Adaptado de [1])

3. Defina uma classe `Aluno`, subclasse da classe `Pessoa`, que contenha as seguintes informações adicionais: número de matrícula, curso no qual está matriculado, data de ingresso na universidade. Redefina o construtor de objetos da classe de maneira apropriada. Crie também uma classe para testar as classes definidas. (Adaptado de [1])

4. Defina uma classe `Conta`, cujos objetos representam contas bancárias, contendo cada qual as informações: número da conta, nome e CPF do correntista, data de abertura da conta, senha e saldo corrente. Defina um construtor de objetos dessa classe que tenha como parâmetros essas informações (exceto o saldo da conta) e retorne um objeto da classe `Conta` com o saldo corrente igual a zero. Defina também dois métodos para implementação das operações de saque e de depósito na conta. Esses métodos devem retornar o saldo corrente da conta depois de efetuada a operação correspondente (suponha que o valor passado como argumento para o método é sempre um valor positivo). O método de saque não deverá alterar o saldo corrente da conta corrente da conta caso o montante do saque seja superior a esse saldo. Crie também uma classe para testar a classe definida. (Adaptado de [1])

5. Defina uma classe `ChequeEspecial`, subclasse da classe `Conta`, que contenha como informação adicional o valor limite (negativo) para o saldo da conta. Redefina o construtor de objetos da classe de maneira apropriada e os demais métodos levando em consideração o limite do cheque especial. Crie também uma classe para testar a classe definida. (Adaptado de [1])

6. Escreva duas classes que implementam a interface abaixo que contém métodos para ordenação e busca em vetores (arrays) de Strings e também para determinar a interseção entre dois vetores de Strings:

```
interface OperacoesComVetoresDeStrings {
  int busca (String[] v, String s);
  void ordena (String[] v);
  String[] intersecao(String[] v1, String[] v2)
}
```

O método `busca` busca em qual posição do vetor `v`, a String `s` aparece. Se `s` não aparece no vetor, o método devolve -1, caso contrário ele devolve o índice no qual `s` se encontra. O método `ordena` recebe um vetor desordenado e o ordena. O método `intersecao` recebe dois vetores de Strings e retorna um novo vetor que contém a intersecção dois dois vetores passados como parâmetros. As classes que implementam a interface, porém, possuem algumas restrições:

- Não podem utilizar algoritmos iguais para implementar os métodos da interface. Em relação ao método de busca: utilize os algoritmos de busca binária e ternária.

## Complexidade de Algoritmos e Técnicas de Projeto de Algoritmos

1. Suponha um algoritmo A e um algoritmo B, com funções de complexidade de tempo $a(n) = n^2 - n + 549$ e $b(n) = 49n + 49$, respectivamente. Determine que valores de $n$ pertencentes ao conjuntos dos números naturais para os quais A leva menos tempo para executar do que B. (Retirado de [4])

2. O que significa dizer que $f(n)$ é $O(f(n))$? (Retirado de [4])

3. Indique se as afirmativas a seguir são verdadeiras ou falsas e justifique suas respostas.

   (a) $2^{n+1} = O(2^n)$
   (b) $2^{2n} = O(2^n)$
   (c) $f(n) = O(u(n))$ e $g(n) = O(v(n)) \Rightarrow f(n) + g(n) = O(u(n) + v(n))$
   (d) $(n + 1)^5$ é $O(n^5)$
   (e) $n^3 \log n$ é $\Omega(n^3)$

   (Retirado de [4])

4. Explique a diferença entre $O(1)$ e $O(2)$? (Retirado de [4])

5. Qual algoritmo possui menor complexidade assintótica: um algoritmo que requer $n^5$ passos ou um algoritmo que requer $2^n$ passos? (Retirado de [4])

6. Resolva as equações de recorrência:

   (a) $T(n) = T(n-1) + c$, sendo $c$ constante e $n > 1$; T(1)=0.

   (b) $T(n) = T(n-1) + 2^n$, para $n > 1$; T(0)=1.

   (c) $T(n) = cT(n-1)$, sendo $c, k$ constantes e $n > 0$; T(0)=k.

   (Retirado de [4])

7. Escreva um método em Java para obter o maior e o segundo maior elemento de um arranjo de valores **int** sem usar nenhum laço. Apresente também a análise desse algoritmo (função de complexidade). (Adaptado de [3])

8. Considere o trecho de programa abaixo:

```
...
 for(i = 0; i < n; i++)
  for( j = 0; j < m; j++)
    {
    // trecho de programa cujo custo é O(3)
    }
...
```

   Qual a complexidade assintótica temporal do trecho de programa acima?

9. Considere o trecho de programa abaixo:

```
...
  for(i = 0; i < n; i++)
   {
    // trecho de programa cujo custo é O(1)
   }

  for( j = 0; j < m; j++)
    {
    // trecho de programa cujo custo é O(3)
    }
...
```

   Qual a complexidade assintótica temporal do trecho de programa acima?

10. Considere o trecho de programa abaixo:

```
...
  for(i=n-1; i > 0 ; i--)
     for(j=1; j <= i; j++)
          {
          // trecho de programa cujo custo é O(1)
          }
...
```

    Qual a complexidade assintótica temporal do trecho de programa acima?

11. Bill tem um algoritmo `buscaEmMatriz` para encontrar um elemento `x` em matriz `A` $n \times n$. O algoritmo faz iterações sobre as linhas de `A` e usa o algoritmo de busca seqüencial em cada linha até que `x` seja encontrado ou que todas as linhas tenham sido examinadas. Qual é o tempo de execução de pior caso do algoritmo `buscaEmMatriz` em função de $n$? O algoritmo é linear $O(n)$? Justifique.

12. Considere o seguinte problema: Dado um conjunto $S$ de $n \geq 2$ números reais, determinar o maior e o menor elemento.

    (a) Implemente um algoritmo (utilizando o paradigma incremental) para esse problema que faz $2n - 3$ comparações.
    Qual seria o caso base ? Qual seria o passo de indução ? Apresente a equação de recorrência desse algoritmo.

    (b) Implemente um algoritmo (utilizando o paradigma divisão e conquista) para esse problema. Apresente a equação de recorrência desse algoritmo. Prove que o seu algoritmo, quando $n$ é potência de 2, faz $\frac{3}{2}n - 2$ comparações.

13. Considere o seguinte problema: Dado um conjunto $S$ de $n \geq 1$ números reais, determinar a somatória dos elementos desse conjunto.

    (a) Implemente um algoritmo (utilizando o paradigma incremental) para esse problema.
    Qual seria o caso base ? Qual seria o passo de indução ? Apresente a equação de recorrência desse algoritmo. Prove através da resolução de recorrências que esse algoritmo é $\Theta(n)$

    (b) Implemente um algoritmo (utilizando o paradigma divisão e conquista) para esse problema. Apresente a equação de recorrência desse algoritmo. Prove, utilizando o teorema mestre, que o seu algoritmo é $\Theta(n)$ (considere na prova que $n$ é potência de 2).

14. Considere o seguinte problema: Dado um vetor (array) $v$ de tamanho $n$, retornar o índice de um elemento $x$ nesse array e -1 caso não esteja presente.

    (a) Implemente um algoritmo (utilizando o paradigma incremental) para esse problema conhecido como *busca seqüencial*.

    Qual seria o caso base ? Qual seria o passo de indução ? Apresente a equação de recorrência desse algoritmo. Prove através da resolução de recorrências que esse algoritmo é $O(n)$

    (b) Implemente dois algoritmos (utilizando o paradigma divisão e conquista) para esse problema: (i) *busca binária* e (ii) *busca ternária*.

    Apresente as equações de recorrência desses dois algoritmos. Prove que assintoticamente os dois algoritmos são equivalentes. Isto é, seja $b_s(n)$, a função de complexidade da *busca binária* e seja $b_t(n)$, a função de complexidade da *busca ternária*, $b_s(n) = \Theta(b_t(n))$.

15. Um polinômio $P_n(x)$ é definido como uma seqüência de números reais $a_n, a_{n-1}, \ldots, a_1, a_0$, e um número real $x$

$$P_n(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$$

    (a) Podemos calcular o valor de $P_n(x)$ aplicando o seguinte raciocínio matemático (indutivo): para calcular $P_n(x)$, basta calcular $x^n$, multiplicar o resultado por $a_n$ e somar o resultado com $P_{n-1}(x)$.

$$P_n(x) = a_n x^n + P_{n-1}(x)$$

    Implemente um método que calcula o valor do polinômio $P_n(x)$ segundo o raciocínio acima. Descreva a equação de recorrência desse algoritmo e mostre que ele realiza $\frac{(n+3)n}{2}$ operações de multiplicação e $n$ operações de adição.

    (b) Agora considere reescrever $P_n(x)$ utilizando a **regra de Horner**

$$P_n(x) = a_0 + x(a_1 + x(a_2 + x(a_3) + \ldots + x(a_{n-1} + xa_n)\ldots)))$$

    Desta forma, podemos calcular o valor de $P_n(x)$ aplicando o seguinte raciocínio indutivo: para calcular $P_n(x)$, basta calcular $P'_{n-1}(x)$, multiplicar o resultado por $x$ e somar o resultado com $a_0$.

$$P'_{n-1}(x) = a_n x^{n-1} + a_{n-1}x^{n-2} + \ldots + a_2 x + a_1$$

$$P_n(x) = xP'_{n-1}(x) + a_0$$

    Implemente um novo método que utiliza essa abordagem. Usando a notação $O$, caracterize o tempo de execução (em relação ao número de operações de multiplicação e adição) desse novo algoritmo. (Adaptado de [3])

16. Seja $S = \{a_1, \ldots, a_n\}$: conjunto de $n$ atividades que podem ser executadas em um mesmo local (Exemplo: palestras em um auditório).

    Para todo $i = 1, \ldots, n$, a atividade $a_i$ começa no instante $s_i$ e termina no instante $f_i$, com $0 \leq s_i < f_i \leq \infty$. Ou seja, supõe-se que a atividade $a_i$ será executada no intervalo de tempo (semi-aberto) $[s_i, f_i)$.

    As atividades $a_i$ e $a_j$ são ditas compatíveis se os intervalos $[s_i,f_i)$ e $[s_j,f_j)$ são disjuntos.

    Descreva um algoritmo guloso, de complexidade $\Theta(n)$, que encontra em $S$ um subconjunto de atividades mutuamente compatíveis que tenha tamanho máximo. Considere que as atividades estão ordenadas em ordem crescente de tempo de término.

    Maiores detalhes sobre esse e outros algoritmos gulosos podem ser encontradas em [2].

17. Considere o seguinte problema: Oito rainhas devem colocadas em um tabuleiro de xadrez, de tal modo que nenhuma delas ataque, ou seja atacada por nenhuma outra. Das regras do xadrez, sabe-se que a rainha pode atacar peças que se encontram na mesma linha, coluna ou diagonal do tabuleiro. Descreva um algoritmo de *backtracking* para esse problema.

    Maiores detalhes sobre esse e outros algoritmos de *backtracking* podem ser encontradas em [5].

## Referências


[1] Carlos Camarão & Lucília Figueiredo. *Programação de Computadores em Java.* LTC Editora, Rio de Janeiro, 2003.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest & Clifford Stein. *Algoritmos - Tradução da 2a. Edição Americana.* Editora Campus, 2002

[3] Michael T. Goodrich & Roberto Tamassia. *Estrutura de Dados e Algoritmos em Java.* Bookman, 2007.

[4] Nívio Ziviani. *Projeto de Algoritmos - com implementação em C e Pascal.* Thomson Editora, 2a edição, 2004.

[5] Niklaus Wirth. *Algoritmos e Estrutura de Dados.* LTC Editora, Rio de Janeiro, 1989.