Universidade de São Paulo – Escola de Artes, Ciências e Humanidades

Bacharelado em Sistemas de Informação

Introdução à Ciência da Computação II

Professores Fátima, Fábio, Luciano e Norton

Data de entrega: 28/11/2010

EXERCÍCIO PROGRAMA 3

Problema da Mochila Binária

Neste trabalho você deverá desenvolver uma classe que utilize diferentes critérios de seleção para implementar soluções gulosas para o problema da Mochila Binária.

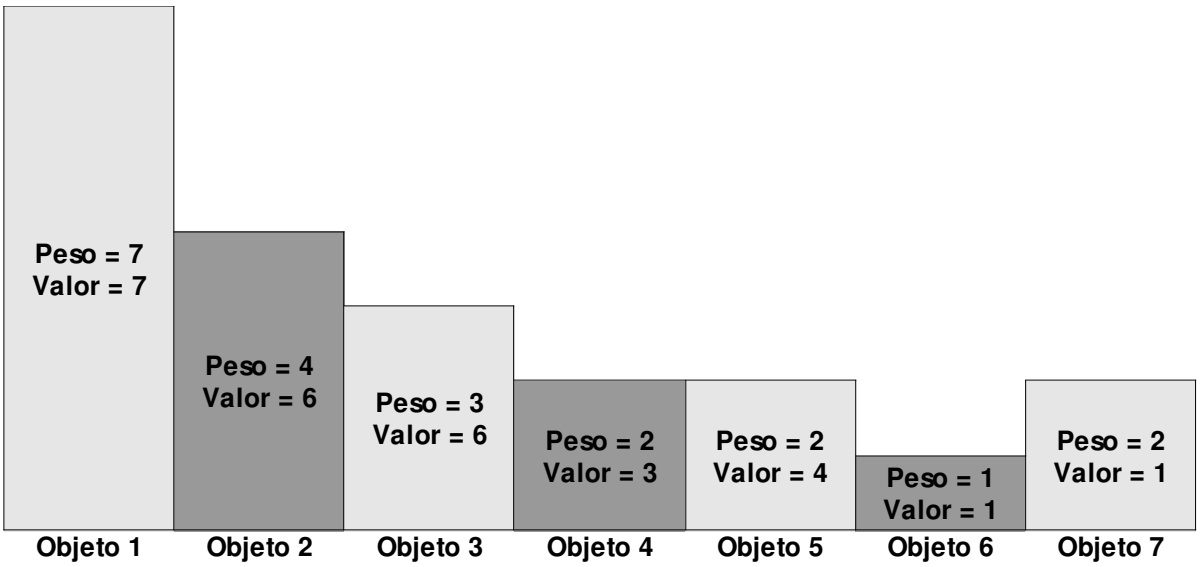
O problema da Mochila Binária consiste em selecionar entre um conjunto de objetos aqueles que deverão ser colocados dentro da mochila, dados: (a) uma mochila que suporta um dado peso e (b) um conjunto de objetos, cada um com um peso e um valor.

A priori, o objetivo é selecionar os objetos de forma a maximizar o valor que está dentro da mochila (sem colocar mais peso na mochila do que ela suporte), porém, este algoritmo possui complexidade exponencial no número de objetos.

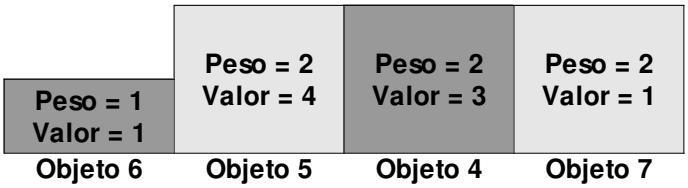
Para evitar uma implementação com complexidade exponencial é possível desenvolver métodos gulosos para a solução deste problema, infelizmente esta solução nem sempre será ótima. Conforme visto em aula, para implementar métodos gulosos é necessário definir a estratégia gulosa. Para este EP vocês deverão implementar três métodos gulosos para a solução deste problema, cada um utilizando uma estratégia gulosa diferente.

As estratégias gulosas serão: (i) seleção dos objetos por menor peso; (ii) seleção dos objetos por maior valor; e (iii) seleção dos objetos por maior relação valor/peso (valor dividido pelo peso).

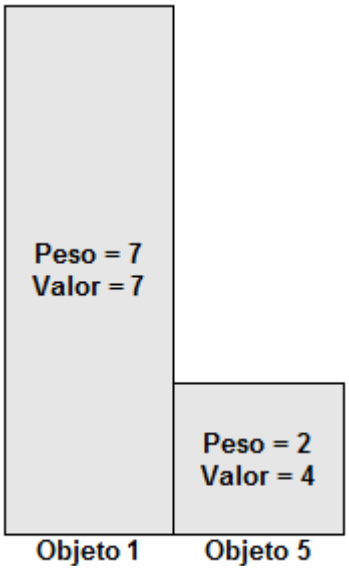
A seguir, serão mostrados exemplos de cada uma dessas estratégias. Considere o seguinte conjunto de 7 objetos e uma mochila com capacidade de 9 quilogramas.



Considerando a estratégia: "**seleção dos objetos por menor peso**" e utilizando como critério de desempate o objeto de maior valor primeiro, teremos a seguinte solução, cujo valor total dentro da mochila será 1 + 4 + 3 + 1 = **9**:

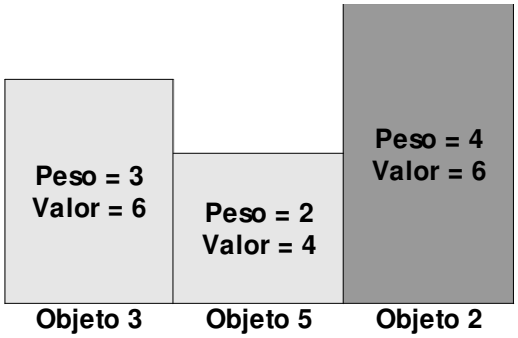


Considerando a estratégia: "**seleção dos objetos por maior valor**" e utilizando como critério de desempate o objeto de menor peso primeiro, teremos a seguinte solução, cujo valor total dentro da mochila será 7 + 4 = **11**:



Considerando a estratégia: "**seleção dos objetos por maior relação valor/peso**" e utilizando como critério de desempate o objeto de maior peso primeiro, teremos a seguinte solução, cujo valor

total dentro da mochila será 6 + 4 + 6 = **16**:



Para este EP, cada aluno deverá implementar essas três estratégias gulosas para o problema da Mochila Binária.

O sistema será composto pelas seguintes classes (muitas delas já implementadas, disponibilizadas juntamente com o código do EP). A classe hachurada é aquela que vocês precisarão implementar/completar.

Nome	Tipo	Resumo
Objeto	classe	Classe que representa um objeto (contendo peso e valor).
Mochila	classe	Classe que corresponde a uma mochila (contendo os atributos: pesoMaximo, pesoUsado, valorDentroDaMochila e numObjetosNaMochila)
MetodosGulosos	classe abstrata	Classe que contém os métodos para a solução gulosa do problema da Mochila Binária.
ExecutaGuloso	classe	Classe "executável" (que possui método *main*) que pode ser usada para testar as demais classes.

Uso de *get*s and *set*s:

Todos os atributos das classes Objeto e Mochila são privados (*private*), para acessá-los existem métodos iniciados por get (para receber o valor de um atributo, por exemplo getPeso()) e set (para atualizar o valor de um atributo, por exemplo setPesoUsado()).

A seguir cada uma das classes será detalhada.

Classe Objeto

Classe que representa um objeto (que poderá ou não ser colocado dentro da mochila). Cada objeto possui um peso > 0 e um valor > 0.

Classe Mochila

Classe que representa uma mochila. Uma mochila possui os seguintes atributos:

```
private double pesoMaximo;              // peso maximo comportado pela mochila
private double pesoUsado = 0;           // peso dentro da mochila
private double valorDentroDaMochila = 0; // valor dentro da mochila
private int numObjetosNaMochila = 0;     // numero de objetos dentro da mochila
```

Além dos métodos gets e sets, a classe mochila possui o método *imprimirDados()* que imprime os dados da mochila atual (todos estes métodos já estão implementado).

Para este EP, colocar um objeto na mochila significa alterar os atributos: pesoUsado, valorDentroDaMochila e numObjetosNaMochila.

Classe MetodosGulosos

Esta classe não possui nenhum atributo. Cada aluno deverá implementar três métodos. Estes métodos deverão retornar como resposta um objeto do tipo Mochila cujos atributos correspondam a solução da implementação da estratégia gulosa:

```
	/* Este metodo deve implementar um algoritmo guloso que selecione objetos
	 *   da listaDeObjetosDisponiveis a serem colocados na mochila de acordo
	 *   com o criterio: 'objetos de menor peso primeiro', caso dois objetos
	 *   tenham o mesmo peso, o criterio de desempate sera:
	 *   'objetos de maior valor primeiro' (apenas para os empates em peso).
	 */
public static Mochila utilizaMenorPeso(double pesoMaximoDaMochila,
Objeto[] listaDeObjetosDisponiveis) {


	/* Este metodo deve implementar um algoritmo guloso que selecione objetos
	 *   da listaDeObjetosDisponiveis a serem colocados na mochila de acordo
	 *   com o criterio: 'objetos de maior valor primeiro', caso dois objetos
	 *   tenham o mesmo valor, o criterio de desempate sera:
	 *   'objetos de menor peso primeiro' (apenas para os empates em valor).
	 */
public static Mochila utilizaMaiorValor(double pesoMaximoDaMochila,
	Objeto[] listaDeObjetosDisponiveis) {


	/* Este metodo deve implementar um algoritmo guloso que selecione objetos
	 *   da listaDeObjetosDisponiveis a serem colocados na mochila de acordo
	 *   com o criterio: 'objetos de maior valor/peso primeiro (valor dividido
	 *   por peso primeiro)', caso dois objetos tenham o mesmo valor/peso,
	 *   o criterio de desempate sera: 'objetos de maior peso primeiro' (apenas
	 *   para os empates).
	 */
public static Mochila utilizaMaiorValorDivididoPorPeso(double
pesoMaximoDaMochila, Objeto[] listaDeObjetosDisponiveis) {
```

Cada um destes três métodos receberá um arranjo de elementos da classe Objeto contendo um ou mais objetos. Todos os elementos do arranjo terão **valores diferentes de null**. O valor do atributo **pesoMaximoDaMochila sempre será maior que zero**, bem como o valor e o peso de cada objeto.

Caso necessário, o aluno poderá implementar métodos auxiliares **dentro da classe MetodosGulosos** (por exemplo, para ordenar os objetos segundo algum critério). Qualquer método implementado dentro desta classe deverá ser estático (*static*). O aluno não deverá implementar nenhum outro método fora da classe MétodosGulosos.

Classe ExecutaGuloso

Esta classe possui o método main e pode ser usada para testar partes do EP. Esta classe possui apenas um conjunto simples de testes.

Note que esta classe não testa exaustivamente todos os métodos implementados (um teste mais cuidadoso é de responsabilidade do aluno). Um exemplo de teste não verificado pela classe ExecutaGuloso é utilizar um conjunto de objetos onde nenhum deles cabe na mochila.

Para o conjunto de sete objetos apresentado anteriormente, a seguinte saída é esperada:

```
Testando Criterio de Selecao: 'Menor Peso'
Peso maximo:      9.0
Peso usado: 7.0
Valor carregado:  9.0
Numero de objetos:      4

Testando Criterio de Selecao: 'Maior Valor'
Peso maximo:      9.0
Peso usado: 9.0
Valor carregado:  11.0
Numero de objetos:      2

Testando Criterio de Selecao: 'Maior Valor Dividido pelo Peso'
Peso maximo:      9.0
Peso usado: 9.0
Valor carregado:  16.0
Numero de objetos:      3
```

Regras de Elaboração e Submissão

- Este trabalho é individual, cada aluno deverá implementar e submeter via COL sua solução.
- A submissão será feita via um arquivo zip (o nome do arquivo deverá ser o número USP do aluno, por exemplo 1234567.zip). Este arquivo deverá conter APENAS o arquivo MetodosGulosos.java
- Além deste enunciado, você encontrará na página da disciplina um zip contendo todos os arquivos envolvidos neste trabalho (note que o arquivo MetodosGulosos.java a ser implementado também está disponível no site, você precisará apenas completá-lo).
- Qualquer tentativa de cola implicará em nota zero para todos os envolvidos.
- Guarde uma cópia do trabalho entregue e verifique, no Col, se o arquivo foi submetido corretamente.
- A data de entrega é 28 de novembro (domingo) até às 23:50h, não serão aceitos trabalhos entregues após esta data (não deixe para submeter na última hora).
- Se necessário, implemente na classe MetodosGulosos métodos auxiliares (por exemplo, para ordenar os objetos). Estes métodos deverão ser estáticos (*static*). Não crie nenhuma classe nova. Não utilize conceitos ou classes que não foram estudados em aula. Só complemente a implementação da classe solicitada (e, no máximo, implemente métodos auxiliares na própria classe).
- **Caso o EP não compile a nota do trabalho será zero.** É importante que você teste seu trabalho executando a classe ExecutaGuloso (note que ela não testa todas as funcionalidades de todas as classes).
- Preencha o cabeçalho existente no inicio do arquivo MetodosGulosos.java (há espaço para se colocar turma, nome do professor, nome do aluno e número USP do aluno).
- Todas as classes pertencem ao pacote ep3.