Segundo Exercício-Programa

Norton Trevisan Roman

25 de outubro de 2018

1 Problema da Mochila Binária

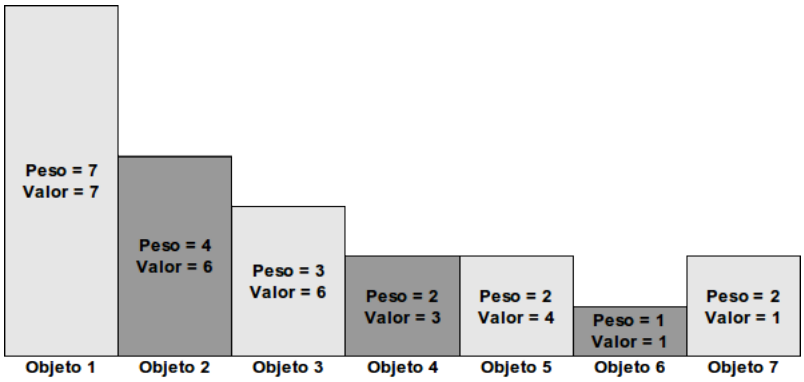
Neste trabalho você deverá desenvolver uma classe que utilize diferentes critérios de seleção para implementar soluções gulosas para o problema da Mochila Binária. O problema da Mochila Binária consiste em dados uma mochila que suporta um dado peso, e um conjunto de objetos, cada um com um peso e um valor, selecionar, dentre esses objetos, aqueles que deverão ser colocados dentro da mochila.

O objetivo é selecionar os objetos de forma a maximizar o valor que está dentro da mochila (sem colocar mais peso do que ela suporta). O algoritmo para tal, contudo, possui complexidade exponencial no número de objetos, o que o torna muitas vezes inviável. Dessa forma, e para evitar uma implementação com complexidade exponencial, é possível desenvolver métodos gulosos para a solução deste problema.

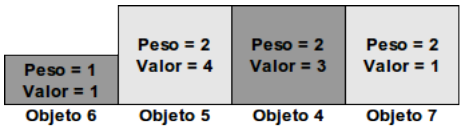
O problema é que, infelizmente, esta solução nem sempre será ótima, então há que se definir e testar diferentes estratégias de abordagem. Para este EP, vocês deverão implementar três métodos gulosos para a solução deste problema, cada um utilizando uma estratégia gulosa diferente. As estratégias gulosas serão:

1. Seleção dos objetos por menor peso;
2. Seleção dos objetos por maior valor; e
3. Seleção dos objetos por maior relação valor/peso (valor dividido pelo peso).

Como exemplo, considere o seguinte conjunto de 7 objetos, além de uma mochila com capacidade de 9 kg:

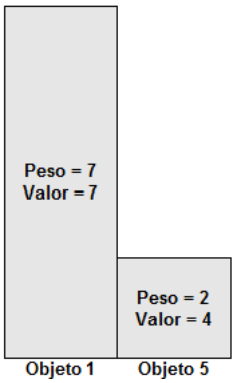


Considerando a estratégia de seleção dos objetos por menor peso, e utilizando como critério de desempate o objeto de maior valor primeiro, teremos a seguinte solução:

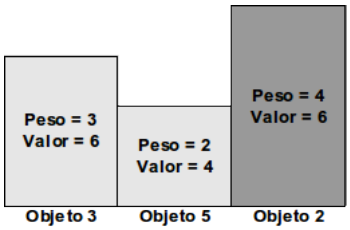


de acordo com a qual o valor total dentro da mochila será $1 + 4 + 3 + 1 = 9$.

Considerando a estratégia de 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 $7 + 4 = 11$:



Por fim, considerando a estratégia de 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 será $6 + 4 + 6 = 16$:



2 Tarefa

Para este EP, cada aluno deverá implementar as três estratégias gulosas acima (com seus respectivos critérios de desempate) para o problema da Mochila Binária.

O sistema será composto pelas seguintes classes (a maioria já implementadas, disponibilizadas juntamente com o código do EP):

- `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.
- `Mochila`: Classe que representa uma mochila (contendo os atributos: `pesoMaximo`, `pesoUsado`, `valorDentroDaMochila` e `numObjetosNaMochila`). Para este EP, colocar

um objeto na mochila significa alterar os atributos: `pesoUsado`, `valorDentroDaMochila` e `numObjetosNaMochila`.

- `MetodosGulosos` (abstrata): Classe que contém os métodos para a solução gulosa do problema da Mochila Binária. **Essa classe precisa ser completada por você**. Cada aluno deverá implementar três métodos, que deverão retornar, como resposta, um objeto do tipo Mochila cujos atributos correspondam à solução da implementação da estratégia gulosa adotada pelo método.

  Cada um destes três métodos recebe 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 (por exemplo, para ordenar os objetos segundo algum critério) dentro de `MetodosGulosos`. Qualquer método implementado dentro desta classe deverá ser estático (`static`), uma vez que os métodos nessa classe assim o são. O aluno não deverá implementar nenhum outro método fora da classe `MetodosGulosos`.

- `ExecutaGuloso`: Classe que pode ser usada para testar as demais classes. Possui um método `main` que pode ser usado para testar partes do EP. Note que a 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.

2.1 Entrada

A entrada ao sistema se dará como exemplificado na classe `ExecutaGuloso`.

2.2 Saída

A saída do sistema, e que será usada para avaliação, é a mochila carregada, conforme demonstrado na classe `ExecutaGuloso`. Então é imprescindível que os atributos desta estejam corretos. Para exemplos, consulte a classe `ExecutaGuloso`.

2.3 Material a Ser Entregue

Este trabalho é individual. Cada aluno deverá implementar e submeter via edisciplinas 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`.

Observação: no arquivo zip não adicionar (sub-)diretórios ou outros arquivos, apenas esse arquivo. Note que o arquivo é .java (e não .class).

Qualquer tentativa de fraude ou cola implicará em nota zero para todos os envolvidos. Guarde uma cópia do trabalho entregue.

Não modifique nada do que foi entregue (assinaturas, pacotes etc)! Você pode apenas adicionar código. **Não crie novas classes**, pois apenas `MetodosGulosos.java` deve ser entregue.

Caso o EP não compile, a nota do trabalho será zero. É importante que você teste seu trabalho executando a classe `ExecutaGuloso.java`. Todas as classes (exceto ExecutaGuloso) pertencem ao pacote `ep2`. **Não mude isso**!

3 Avaliação

Para avaliação, será observada a corretude do programa (ou seja, se faz o que é pedido e **como** é pedido), bem como sua adequação às regras para entrega (nome do zip, arquivos entregues, assinaturas preservadas etc).

É de sua responsabilidade verificar:

- Se o material entregue está de acordo com as especificações
- Se tudo compila e o sistema roda a partir da `ExecutaGuloso`
- Se a entrega realmente ocorreu (ou seja, se o upload foi feito corretamente). Então faça o upload, baixe e teste o que baixou.

Falhas nos itens acima não serão toleradas.
Atrasos não serão tolerados. Então não deixe para a última hora.
Algumas outras observações pertinentes ao trabalho, que influem em sua nota, são:

- Este exercício-programa deve ser elaborado individualmente.
- Não será tolerado plágio, em hipótese alguma.
- Exercícios com erro de sintaxe (ou seja, erros de compilação), receberão nota ZERO

Atenção! A avaliação se dará como nos moldes da classe `ExecutaGuloso` entregue.

3.1 Dicas para Verificação de Funcionamento

Além de formato, corretude etc, sua implementação deve se ajustar perfeitamente ao pacote enviado a vocês. Para aumentar a chance disso acontecer, é aconselhável fazer o seguinte:

1. Crie um diretório qualquer. Digamos que seja `meuDir`
2. Baixe novamente o arquivo `.zip` do EP e o descompacte dentro de `meuDir`. Você verá a `ExecutaGuloso.java`, o diretório `ep2` e, dentro dele, as classes `Mochila` e `Objeto` já compiladas, além da `MetodosGulosos.java` para você completar.
3. Substitua então a classe `MetodosGulosos.java` pela sua implementação

4. Via linha de comando, vá ao diretório `meuDir`

5. Lá compile tudo com `javac ExecutaGuloso.java` (isso irá compilar inclusive as tuas implementações, usando a `ExecutaGuloso.java` fornecida no EP)

6. Rode com `java ExecutaGuloso`, e compare os resultados com os testes fornecidos

Seguindo esses passos, o sistema certamente funcionará nos testes. Quanto a estar totalmente correto, aí são outros quinhentos...