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

Bacharelado em Sistemas de Informação

Introdução à Ciência da Computação II

Professores Luciano Digiampietri e Fábio Nakano

Data de entrega: 19/11/2011

EXERCÍCIO PROGRAMA 3

Problema da Distribuição de Alunos em Grupos

Neste trabalho você deverá implementar/completar uma classe que encontre a **solução gulosa e a solução ótima** para o problema de distribuição de alunos em grupos.

O problema de distribuição de alunos em grupos consiste em atribuir um grupo a cada aluno, dados: (a) o número total de grupos (b) um conjunto de alunos, cada um com seu nome, número USP e média ponderada.

A priori, o objetivo é montar os grupos de forma a minimizar a diferença entre a soma das médias dos alunos de cada grupo (ou seja, em termos de média total, os grupos devem estar o mais homogêneos possível). A solução ótima para este problema pode ser encontrada utilizando a estratégia de *backtracking* e tem complexidade exponencial no número de alunos. Além disso, é possível encontrar soluções potencialmente boas (mas não necessariamente ótimas) utilizando um algoritmo guloso, cuja complexidade assintótica será $O(n^2)$ Você deve implementar as solução empregando as duas estratégias: *backtracking* e gulosa.

Todos os grupos formados deverão ter o mesmo número de alunos (e em todos os testes, o número de alunos será um múltiplo do número de grupos) e cada aluno deverá pertencer a exatamente um grupo. Desta forma, uma **solução válida** (independente de ser boa ou não) é uma solução na qual todos os alunos pertencem a um grupo e todos os grupos possuem o mesmo número de alunos.

Atribuir um grupo a um aluno significa mudar o atributo grupo de um objeto do tipo Aluno. Para um exemplo que contenha g grupos, o número dos grupos válidos vão de 0 (zero) até g-1. Inicialmente, todos os alunos começam com grupo = -1 (que corresponde a um grupo inválido).

Detalhamento da solução utilizando *backtracking*:

O objetivo deste problema de otimização é minimizar a diferença entre a soma das médias dos grupos, para isto será calculada a diferença entre a **média ótima teórica** e a **média da distribuição de grupos feita em uma dada solução válida**. Estes cálculo (da média ótima teórica e da diferença entre esta média a uma dada solução) já estão implementados na classe Aluno, fornecida juntamente com o enunciado deste EP.

média ótima teórica = (soma das médias de todos os alunos)/(número de grupos)

diferença entre a média teórica e a média de uma dada solução válida:

$$\sqrt{\sum \left(média\,teórica\,ótima - somaDasMédias_i \right)^2}$$

onde $somaDasMédias_i$ corresponde a soma das médias dos alunos que estão no grupo i, e com i variando de 0 até g-1 (sendo g o número de grupos)

Para este problema poderá existir diversas soluções ótimas. No EP3, ao contrário do EP2, não haverá necessidade de implementar nenhum critério de desempate. O método que utiliza *backtracking* poderá retornar qualquer uma das distribuições de alunos em grupos que correspondam uma **solução ótima**.

Detalhamento da solução utilizando um algoritmo guloso:

Conforme visto em sala de aula, um dos segredos para o desenvolvimento de um algoritmo guloso é a definição da estratégia gulosa, ou seja, dado o estado atual esta estratégia determinará qual será a próxima ação a ser executada. Para este problema, a estratégia gulosa fará o seguinte, considerando que há g grupos: vá do grupo 0 até o grupo g-1 e atribua a cada um deles o aluno que tiver a menor média entre os alunos que ainda estão sem grupos (o aluno com pior média no grupo 0, o com segunda pior média no grupo 1 e assim por diante até o grupo g-1); em seguida faça o caminho reverso: vá do grupo g-1 até o grupo 0 atribuindo a cada um deles o aluno que tiver a menor média entre os alunos que ainda estão sem grupos (ou seja, dentre os alunos que ainda estão sem grupos coloque o com menor média no grupo g-1, o com segunda menor no grupo g-2 e assim por diante até o grupo 0. Repita esses passos enquanto existirem alunos sem grupos.

Por exemplo, considerando um conjunto de 6 alunos e no qual queremos formar 2 grupos:

Coloque no grupo zero o aluno com menor média;

Coloque no grupo um o aluno com a segunda menor média;

Coloque no grupo um o aluno com a terceira menor média;

Coloque no grupo zero o aluno com a quarta menor média;

Coloque no grupo zero o aluno com a quinta menor média;

Coloque no grupo um o aluno com a sexta menor média.

Se os alunos estiverem ordenados pela média teremos a seguinte atribuição:

Aluno	Aluno1	Aluno2	Aluno3	Aluno4	Aluno5	Aluno6
Média	1.5	6.2	6.25	6.5	6.8	9.25
Grupo	0	1	1	0	0	1

Observações:

1a) o arranjo de alunos de entrada não estará ordenado;

2a) o número de alunos poderá variar de 1 até 32 e sempre será um múltiplo do número de grupos;

3a) o número de grupos poderá variar de 1 até número de alunos (mas sempre será um divisor do número de alunos).

4a) pode haver mais de uma solução seguindo o critério guloso apresentado (caso dois alunos possuam a mesma nota), neste caso qualquer uma das soluções será considerada correta (desde que obedeça ao critério guloso apresentado).

O exemplo ilustrado a seguir corresponde ao Exemplo 3 que já foi incluído no arquivo para teste do EP3 (ExecutaTestesEP3.java). Neste exemplo, há um conjunto de 15 alunos e queremos dividi-los em 1 grupo, 3 grupos e 5 grupos. Os tempos de execução são apenas ilustrativos.

Aluno.nome	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Nota	9.25	5.2	4.25	8.5	3.5	8.8	2.5	7.25	8	7	6	5	4	3	2

Exemplo 3, um grupo:

Tempo de execução do algoritmo guloso: 0 ms.

Aluno.nome	15	7	14	5	13	3	12	2	11	10	8	9	4	6	1
Nota	2	2.5	3	3.5	4	4.25	5	5.2	6	7	7.25	8	8.5	8.8	9.25
Grupo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Média de cada grupo: Grupo 0: 84.25
Diferença algoritmo guloso: 0.0

Tempo de execução do algoritmo usando backtracking: 0 ms.

Aluno.nome	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Nota	**9.25**	5.2	4.25	8.5	3.5	8.8	2.5	7.25	8	7	6	5	4	3	2
Grupo	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Média de cada grupo: Grupo 0: 84.25
Diferença algoritmo usando backtracking: 0.0

Exemplo 3, três grupos:

Tempo de execução do algoritmo guloso: 0 ms.

Aluno.nome	15	7	14	5	13	3	12	2	11	10	8	9	4	6	1
Nota	2	2.5	3	3.5	4	4.25	5	5.2	6	7	7.25	8	8.5	8.8	9.25
Grupo	0	1	2	2	1	0	0	1	2	2	1	0	0	1	2

Média de cada grupo: Grupo 0: 27.75 Grupo 1: 27.75 Grupo 2: 28.75
Diferença algoritmo guloso: 0.816496580927726

Tempo de execução do algoritmo usando backtracking: 227 ms.

Aluno.nome	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Nota	9.25	5.2	4.25	8.5	3.5	8.8	2.5	7.25	8	7	6	5	4	3	2
Grupo	0	0	0	1	0	2	1	2	1	1	0	2	2	2	1

Média de cada grupo: Grupo 0: 28.2 Grupo 1: 28.0 Grupo 2: 28.05
Diferença algoritmo usando backtracking: 0.1471960144387967

Exemplo 3, cinco grupos:

```
Tempo de execucao do algoritmo guloso: 0 ms.
```

Aluno.nome	15	7	14	5	13	3	12	2	11	10	8	9	4	6	1
Nota	2	2.5	3	3.5	4	4.25	5	5.2	6	7	7.25	8	8.5	8.8	9.25
Grupo	0	1	2	3	4	4	3	2	1	0	0	1	2	3	4

```
Media de cada grupo:    Grupo 0: 16.25    Grupo 1: 16.5    Grupo 2: 16.7    Grupo 3: 17.3
      Grupo 4: 17.5
Diferença algoritmo guloso:  1.0630145812734653

Tempo de execução do algoritmo usando backtracking: 67891 ms.
```

Aluno.nome	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Nota	9.25	5.2	4.25	8.5	3.5	8.8	2.5	7.25	8	7	6	5	4	3	2
Grupo	0	0	1	1	2	3	0	2	4	4	2	3	1	3	4

```
Média de cada grupo:    Grupo 0: 16.95    Grupo 1: 16.75    Grupo 2: 16.75    Grupo 3: 16.8
      Grupo 4: 17.0
Diferença algoritmo usando backtracking:  0.23452078799117101
```

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
Aluno	classe	Classe que representa um aluno (contendo nome, numeroUSP, mediaPonderada, grupo). Além disso há diversos métodos implementados que podem ser utilizadados para ajudar a solucionar o EP.
AtribuiGrupos	classe abstrata	Classe que contém os método para a solução (gulosa e usando *backtracking*) do problema de distribuir alunos em grupos..
ExecutaTestesEP3	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 da classe Aluno são privados (*private*), para acessá-los existem métodos iniciados por get (para receber o valor de um atributo.

A seguir cada uma das classes é detalhada.

Classe Aluno

Classe que representa um aluno. Cada aluno possui um nome, um numeroUSP, uma mediaPonderada e um grupo (que inicialmente possuirá o valor -1 e que deverá ser alterado por vocês pela implementação dos métodos solicitados.

Além disso, a classe aluno possui vários métodos auxiliares que vocês poderão utilizar (caso considerem necessário) na solução do EP. Intencionalmente esses métodos não estão documentados, de maneira que vocês só devem utilizá-los se efetivamente entenderem o que seus códigos fazem. A classe Aluno não deve ser modificada. Segue a assinatura dos métodos auxiliares:

```
public static boolean verificaSeDistribuicaoEhValida(Aluno[] alunos, int grupos);
public static double mediaTotalTeoricaDeCadaGrupo(Aluno[] alunos, int grupos);
public static double diferencaEmRelacaoAMediaTeorica(Aluno[] alunos, int grupos, double
mediaTeorica);
public static Aluno clonarAluno(Aluno alunoOriginal);
public static Aluno[] clonarArranjoDeAlunos(Aluno[] alunos);
public static void imprimirSolucao(Aluno[] alunos);
```

Classe AtribuiGrupos

Esta classe, a priori, não possui nenhum atributo. Vocês deverão implementar (ou melhor, completar) dois métodos. Estes métodos não possuem retorno (seus retornos são vazios [*void*]) mas deverão atribuir um grupo a cada um dos alunos recebidos como entrada de forma a satisfazer a especificação deste EP. Isto é, o método guloso deve seguir o critério guloso e o método que utilize *backtracking* deve retornar a solução ótima considerando o problema de minimizar a diferença entre as somas das médias dos grupos. Vale lembrar que atribuir um grupo a um aluno significa modificar o valor do atributo grupo de um dado objeto da classe Aluno.

Vocês poderão criar atributos (estáticos ou não) e implementar outros métodos auxiliares nesta classe, **desde que não mudem as assinaturas dos dois métodos estáticos que devem ser completados.**

Além de um arranjo de objetos da classe Aluno, os métodos a serem implementados também recebem como parâmetro a quantidade de grupos que deverão ser formados. Os grupos válidos vão de zero até quantidade de grupos menos 1.

Nenhuma outra classe deverá ser implementada ou alterada (exceto para realização de testes pontuais). O EP tem que compilar e funcionar considerando as demais classes originais. Lembre que serão feitos outros testes além dos apresentados.

```
package ep3_2011;

public abstract class AtribuiGrupos {

  /* Este metodo atribui para cada aluno do arranjo alunos o numero de um
   * grupo, os numeros dos grupos variam de 0 ate (numeroDeGrupos - 1).
   * A estrategia gulosa que devera ser adotada esta no enunciado do EP3.
   */
  public static void atribuiGuloso(Aluno[] alunos, int numeroDeGrupos){

    //TODO completar

  }

  /* Este metodo atribui para cada aluno do arranjo alunos o numero de um
   * grupo, os numeros dos grupos variam de 0 ate (numeroDeGrupos - 1).
   * Este metodo devera atribuir aos alunos um numero de grupo de forma a
   * minimizar a diferenca entre as somas das medias dos alunos de cada grupo,
   * conforme o enunciado do EP3.
   */
  public static void atribuiBacktracking(Aluno[] alunos, int numeroDeGrupos){

    //TODO completar

  }

}
```

Classe ExecutaTestesEP3

Esta classe possui o método main e pode ser usada para testar partes do EP. Esta classe possui quatro conjuntos diferentes de alunos testados para formar 1, 2, 3, 4 ou 5 grupos (dependendo do teste).

Note que esta classe não testa exaustivamente o método implementado (um teste mais cuidadoso é de responsabilidade do aluno). Um exemplo simples de teste não verificado pela classe ExecutaTestesEP3 é utilizar um conjunto de um aluno a ser colocado em um grupo.

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 ou rar (o nome do arquivo deverá ser o número USP do aluno, por exemplo 1234567.zip). Este arquivo deverá conter APENAS o arquivo AtribuiGrupos.java
- Além deste enunciado, você encontrará na página da disciplina um zip contendo todos os arquivos envolvidos neste trabalho (note que o arquivo AtribuiGrupos.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 19 de novembro (sábado) até às 23:00h (com um hora de tolerância), não serão aceitos trabalhos entregues após esta data (não deixe para submeter na última hora).
- Se necessário, implemente na classe AtribuiGrupos métodos auxiliares e crie atributos. 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 e/ou crie atributos na própria classe).
- **Caso o EP não compile a nota do trabalho será zero.** É importante que você teste seu trabalho executando a classe ExecutaTestesEP3 (note que ela não testa todas as funcionalidades do método implementado).
- Preencha o cabeçalho existente no início do arquivo AtribuiGrupos.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_2011