INSTITUTO FEDERAL DE MINAS GERAIS

Bacharelado em Ciência da Computação

Disciplina: Matemática Discreta

Trabalho Prático

Prof. Diego Mello da Silva

Formiga-MG
5 de novembro de 2017

Sumário

1 Informações Gerais

Este documento descreve a especificação do Trabalho Prático da disciplina Matemática Discreta, e deve ser seguido de forma a contemplar os itens considerados na avaliação por parte do professor da disciplina. O trabalho deve ser feito em grupo de até 03 (três) alunos e tem o valor de 30 pontos.

Em relação ao prazo o trabalho prático deverá ser realizado em 3 semanas (21 dias) contados a partir da data da publicação do mesmo em sala de aula pelo professor. O trabalho deverá ser entregue até as 23:59 hs da data limite. Trabalhos entregues após este prazo serão desconsiderados (isto é, valerão zero).

O documento é organizado como segue. Na Seção 2 será apresentado o problema que este trabalho prático pretende resolver; na Seção 3 serão apresentados os requisitos de *software* que, se implementados, permitirão resolver o problema proposto; na Seção 4 serão apresentados os critérios de avaliação do trabalho e respectiva pontuação; na seção 5 serão abordadas algumas considerações importantes sobre o trabalho em equipe e código de conduta.

2 O problema

O presente documento especifica um *software* capaz de processar um arquivo de entrada que especifica um relação R sobre o produto cartesiano $A \times A$, onde A é um conjunto formado por números inteiros na faixa $1, \ldots, N$, com N informado no arquivo. O *software* deverá carregar o arquivo em memória, armazenando seu conteúdo em uma estrutura de dados do tipo matriz para representar os pares ordenados da relação e permitir que operações sejam realizadas sobre a estrutura. Dentre as operações a serem realizadas estão a verificação das propriedades da relação (reflexiva, simétrica e transitiva), e cálculo do fecho correspondente se a relação não tiver uma das propriedades em questão.

Uma relação R em um conjunto A é chamada de ***reflexiva*** se $(x, x) \in R$ para todo elemento $x \in A$, ou seja, $\forall x \in A\big((x, x) \in R\big)$. Em outras palavras, R é reflexiva se contiver todos os pares do tipo (x, x), onde x é elemento do domínio A. Uma relação R em um conjunto A é chamada de ***simétrica*** se $(y, x) \in R$ sempre que $(x, y) \in R$, ou seja, $\forall x \forall y \ \big((x, y) \in R \rightarrow (y, x) \in R\big)$. Novamente, uma relação será simétrica desde que, para cada par ordenado (x, y) existente em R também exista o par ordenado (y, x) correspondente. Por fim, uma relação é ***transitiva*** se,sempre que o par $(x, y) \in R$ e $(y, z) \in R$, então $(x, z) \in R$ para todo $x, y, z \in A$.

Quando ao fecho de um relação, este consiste em um conjunto que contêm os pares da relação original adicionado de novos pares que são acrescentados até que se obtenha a propriedade P ao qual o fecho se refere. Sob o ponto de vista mais forma, seja A um conjunto, R uma relação binária em A e P uma das três propriedades comentadas acima. O ***fecho*** de R é uma relação binária R^* em A que possui a propriedade P e satisfaz três condições, a saber: R^* tem a propriedade P; $R \subseteq R^*$; e S é uma relação qualquer que contém R e satisfaz P, então $R^* \subseteq S$. Em outras palavras, o fecho de uma propriedade P sobre R é o menor conjunto de pares que contêm os pares de R e acrescentam novos para atingir P.

A próxima seção apresentará a especificação técnica do *software* classificador proposto neste trabalho, destacando os requisitos que ele deverá cumprir para com-

pletar o trabalho prático.

3 Especificação

Esta seção apresenta uma especificação técnica de elementos que devem ser implementados pelo grupo para atender à aplicação de cálculo de fecho especificado neste trabalho. A aplicação deverá ser escrita em linguagem `C` usando o compilador `gcc` disponível em distribuições Linux. A aplicação será do tipo console (linha de comando), sendo que toda a entrada e saída de dados ocorrerá mediante uso de teclado e monitor.

O código fonte da aplicação será compilado no ambiente operacional Linux, distribuição Ubuntu. Certifique-se que o seu código fonte é compatível com tal ambiente. Códigos que não compilarem no ambiente mencionado serão desconsiderados. Toda a implementação deverá ser entregue em um único arquivo de nome `closure.c`. **O código fonte deverá possuir um cabeçalho em comentários contendo os nomes e matrículas dos integrantes do grupo.** O trabalho deverá ser enviado ao professor até a data limite em um único arquivo compactado denominado `discreta-pratico-2017.zip` contendo o código fonte da aplicação e outros arquivos que venham a ser requeridos na especificação do trabalho. As próximas sub-seções detalham cada requisito da aplicação.

3.1 Req. 01 - Entrada de Dados

A entrada de dados para a aplicação que calcula fechos deve ser feita por meio de arquivo informado via linha de comando. Um argumento de linha de comando é qualquer string informada quando se invoca uma aplicação via terminal. No caso específico deste trabalho, os argumentos de linha de comando que devem ser informados são os seguintes:

```
user@machine$ ./closure <arquivo-entrada> <preâmbulo-saída>
```

sendo que:

- `arquivo-entrada`: nome do arquivo de entrada, no formato ASCII texto plano, contendo a descrição da relação e seus pares;
- `preâmbulo-saída`: nome do arquivo de saída, em formato ASCII texto plano, contendo comandos da linguagem DOT capazes de descrever o dígrafo do fecho correspondente destacando os arcos originais e acrescentados no cálculo do fecho, para cada tipo de relação. Maiores detalhes na seção correspondente ao requisito de saída.

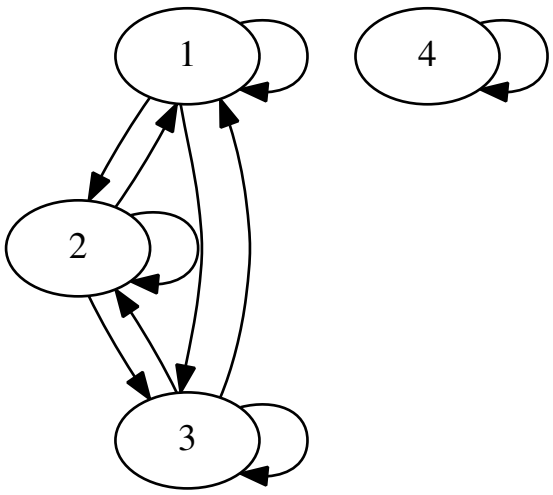
O conteúdo do arquivo de saída será melhor detalhado adiante, no requisito correspondente. Já o conteúdo do arquivo de entrada é descrito a seguir. Ele contêm informações linha por linha sobre a relação, sempre iniciando com um caracter de

controle seguido dos dados correspondentes. Seu conteúdo é descrito pelos seguintes caracteres de controle:

- `n`: inteiro. Indica o total de elementos do conjunto A sobre o qual a relação R é construída. Em outras palavras identifica quantos nós terá o dígrafo da relação R;
- `r`: inteiro inteiro. Indica o par $(x, y) \in R$, ou em outras palavras, que um arco parte do nó x em direção ao nó y no dígrafo correspondente à R.
- `f`: fim de arquivo. Indica que não haverá mais informações sobre R que seguem do arquivo.

Para exemplificar, seja o arquivo descrito abaixo, que descreve uma relação sobre o conjunto $A = \{1, 2, 3, 4\}$. O arquivo descreve então que o conjunto possui 4 elementos, e 10 pares ordenados, descritos um por linha. à direita o dígrafo da relação correspondente.

```
n 4
r 1 1
r 3 2
r 2 2
r 1 3
r 4 4
r 3 3
r 2 3
r 2 1
r 3 1
r 1 2
f
```



O grupo deverá pensar em situações problemas que devem ser tratadas na entrada de dados. Caso alguma delas ocorra a aplicação deverá ser abortada, exibindo uma mensagem de erro para o usuário no terminal. A entrada de dados deverá ser implementada em uma função auxiliar que percorrerá o arquivo lendo seu conteúdo.

3.2 Req. 02 - Estrutura de Dados Matriz

Neste requisito pede-se que o aplicativo guarde em memória os pares contidos na relação R descrita no arquivo de entrada em memória.

Para tal sugere-se usar uma estrutura de dados do tipo matricial, contendo valores booleanos que representam se dois elementos quaisquer da matriz estão ou não relacionados segundo R. A interpretação e gravação dos dados deverá seguir a convenção de que as linhas quando percorridas referem-se aos elementos x do par ordenado $(x, y) \in R$, e que as colunas quando percorridas referem-se aos elementos

y do mesmo par ordenado. Os elementos das linhas e colunas devem ser rotulados de forma sequencial, partindo de 1 até n. Para exemplificar, a relação

$$R = \Big\{(1,1),(3,2),(2,2),(1,3),(4,4),(3,3),(2,3),(2,1),(3,1),(1,2)\Big\}$$

seria representada matricialmente por

$$R = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.3 Req. 03 - Check Propriedade Reflexiva

Neste requisito o grupo deve implementar um algoritmo capaz de verificar se a relação R representada em matriz possui a propriedade reflexiva. Caso a relação não seja identificada com esta propriedade, então deve-se invocar o algoritmo que irá computar o fecho reflexivo e persistir o resultado em arquivo, no formato DOT. Mais detalhes na seção 3.6.

3.4 Req. 04 - Check Propriedade Simétrica

Neste requisito o grupo deve implementar um algoritmo capaz de verificar se a relação R representada em matriz possui a propriedade simétrica, isto é, se para cada arco $(x, y) \in R$ existe também o arco (y, x) na mesma relação. Caso a relação não seja identificada com esta propriedade, então deve-se invocar o algoritmo que irá computar o fecho simétrico e persistir o resultado em arquivo, no formato DOT. Mais detalhes na seção 3.7.

3.5 Req. 05 - Check Propriedade Transitiva

Neste requisito o grupo deve implementar um algoritmo capaz de verificar se a relação R representada em matriz possui a propriedade transitiva, isto é, para cada arco $(x, y) \in R$ onde existe um outro arco (y, z) que parte de y, então deve haver um 'atalho' direto entre x e z para que a relação seja transitiva. Caso a relação não seja identificada com esta propriedade, então deve-se invocar o algoritmo que irá computar o fecho transitivo e persistir o resultado em arquivo, no formato DOT. Mais detalhes na seção 3.8.

3.6 Req. 06 - Fecho Propriedade Reflexiva

Neste requisito o grupo deverá encontrar o fecho reflexivo de uma relação R informada que não possua essa propriedade. O grupo deverá projetar e implementar um algoritmo capaz de identificar os novos arcos a adicionar na relação tal que R torne-se reflexiva. Estes novos arcos deverão ser destacados em vermelho no arquivo .DOT a ser construído como saída, no formato dado pelo requisito da seção 3.9. Neste caso, o nome do arquivo de saída deverá ser montado pela concatenação do preâmbulo informado por linha de comando seguido da substring `'-ref.dot'`. Exemplo: se o preâmbulo é `fecho-relacao01`, logo arquivo de saída deverá ser `fecho-relacao01-ref.dot`.

3.7 Req. 07 - Fecho Propriedade Simétrica

Neste requisito o grupo deverá encontrar o fecho simétrico de uma relação R informada que não possua essa propriedade. O grupo deverá projetar e implementar um algoritmo capaz de identificar os novos arcos a adicionar na relação tal que R torne-se simétrica. Estes novos arcos deverão ser destacados em vermelho no arquivo .DOT a ser construído como saída, no formato dado pelo requisito da seção 3.9. Neste caso, o nome do arquivo de saída deverá ser montado pela concatenação do preâmbulo informado por linha de comando seguido da substring `'-sim.dot'`. Exemplo: se o preâmbulo é `fecho-relacao01`, logo arquivo de saída deverá ser `fecho-relacao01-sim.dot`.

3.8 Req. 08 - Fecho Propriedade Transitiva

http://www.graphviz.org/Documentation/dotguide.pdf Neste requisito o grupo deverá encontrar o fecho transitivo de uma relação R informada que não possua essa propriedade. O grupo deverá projetar e implementar um algoritmo capaz de identificar os novos arcos a adicionar na relação tal que R torne-se transitiva. Estes novos arcos deverão ser destacados em vermelho no arquivo .DOT a ser construído como saída, no formato dado pelo requisito da seção 3.9. Neste caso, o nome do arquivo de saída deverá ser montado pela concatenação do preâmbulo informado por linha de comando seguido da substring `'-tra.dot'`. Exemplo: se o preâmbulo é `fecho-relacao01`, logo arquivo de saída deverá ser `fecho-relacao01-tra.dot`.

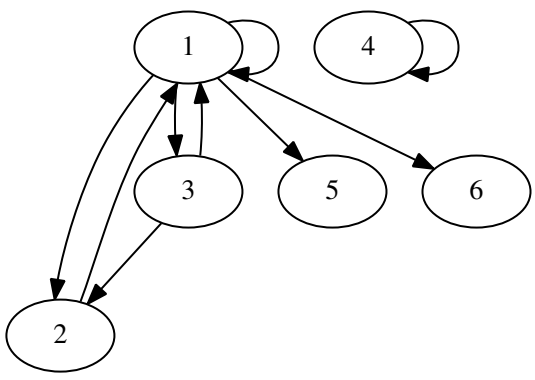
3.9 Req. 09 - Saída de Dados

Neste requisito o grupo deverá implementar a saída de dados sob a forma de arquivo de saída. Para cada propriedade que a relação R informada não possuir deve-se criar uma arquivo contendo seu fecho, conforme os requisitos contidos nas seções 3.6, 3.7 e 3.8.

Os arquivos devem ser construídos em formato texto plano, ASCII, contendo comandos da linguagem .DOT que descreve grafos. Para maiores detalhes sobre a sintaxe desta linguagem consultar o artigo '*Drawing graphs with dot*', disponível em http://www.graphviz.org/Documentation/dotguide.pdf, além de outros materiais de interesse.

Para exemplificar, seja a relação R descrita pelo seguinte arquivo de entrada:

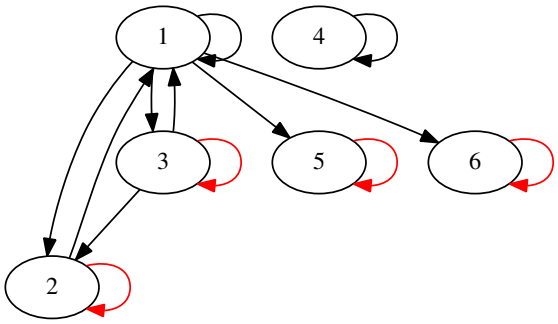
```
n 6
r 1 1
r 3 2
r 1 3
r 4 4
r 2 1
r 3 1
r 1 2
r 1 6
r 1 5
f
```



Como pode ser visto, o grafo não possui nenhuma das três propriedades. Logo a aplicação deverá criar três arquivos de saída, conforme os requisitos expostos acima, cada qual contendo a descrição do grafo da relação R com novos arcos (isto é, o fecho de cada propriedade). Desta forma espera-se como saída arquivos no formato .DOT que destacam os novos arcos. Vamos exemplificar a saída usando o mesmo grafo de entrada, com três arquivos de saída hipotéticos que devem servir de modelo para a construção da saída. Observem o arco em destaque.

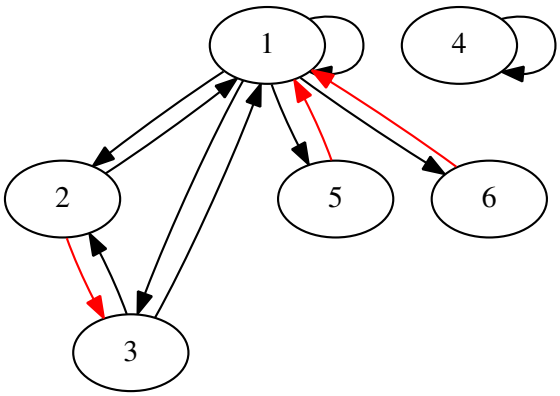
Fecho reflexivo:

```
digraph fecho
{
    1;
    2;
    3;
    4;
    5;
    6;
    1 -> 1;
    3 -> 2;
    1 -> 3;
    4 -> 4;
    2 -> 1;
    3 -> 1;
    1 -> 2;
    1 -> 6;
    1 -> 5;
    2 -> 2 [color=red];
    3 -> 3 [color=red];
    5 -> 5 [color=red];
    6 -> 6 [color=red];
}
```



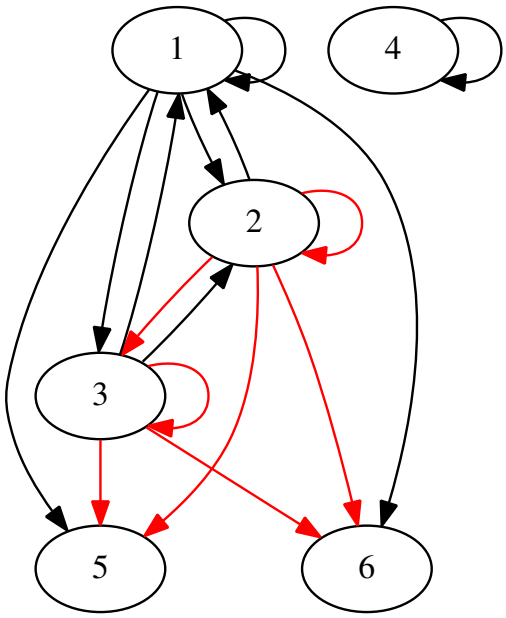
Fecho simétrico:

```
digraph fecho
{
    1;
    2;
    3;
    4;
    5;
    6;
    1 -> 1;
    3 -> 2;
    1 -> 3;
    4 -> 4;
    2 -> 1;
    3 -> 1;
    1 -> 2;
    1 -> 6;
    1 -> 5;
    2 -> 3 [color=red];
    5 -> 1 [color=red];
    6 -> 1 [color=red];
}
```



Fecho transitivo:

```
digraph fecho
{
    1;
    2;
    3;
    4;
    5;
    6;
    1 -> 1;
    3 -> 2;
    1 -> 3;
    4 -> 4;
    2 -> 1;
    3 -> 1;
    1 -> 2;
    1 -> 6;
    1 -> 5;
    2 -> 3 [color=red];
    2 -> 5 [color=red];
    2 -> 6 [color=red];
    3 -> 5 [color=red];
    3 -> 6 [color=red];
    2 -> 2 [color=red];
    3 -> 3 [color=red];
}
```



Uma vez construídos os arquivos de saída, pode-se utilizar o aplicativo GraphViz

para gerar imagens a partir da descrição dos grafos. No Linux, uma vez instalado, o aplicativo pode ser invocado por linha de comando usando o seguinte comando

```
user@machine$ neato -Tpdf arquivo.dot -o saida.pdf
```

onde `arquivo.dot` consiste em qualquer arquivo com descrição de grafos no formato .DOT, e `saida.pdf` é o nome do arquivo de saída especificado que conterá a imagem do grafo. Para maiores detalhes sobre o pacote GraphViz e seus comandos, consultar o site http://www.graphviz.org/. Na distribuição Ubuntu, o programa pode ser instalado usando-se `sudo apt-get install graphviz`.

3.10 Req. 10 - Instâncias de Teste

Neste requisito o grupo deverá construir 04 (quatro) arquivos contendo instância de relações que não sejam nem reflexivas, nem simétricas, nem transitivas. **Não é permitido aproveitar arquivos de outros grupos (sob pena de zerar o valor deste requisito caso ocorra)**. Cada arquivo deverá ter relações sobre o conjunto $\{1, 2, 3, 4, 5, 6, 7, 8, 910\}$, com pelo menos 30 (trinta) arcos cada. Os arquivos devem ser compactados junto com o código fonte e submetidos para o email do professor até a data limite.

3.11 Req. 11 - Documentação de Código

A documentação de código é importante em qualquer implementação computacional e será considerada neste trabalho. Pede-se que o arquivo que contem o código fonte possua ao menos (i) cabeçalho inicial, contendo nome do aplicativo, membros do grupo com nome e matrícula, instruções de compilação, ambiente de desenvolvimento, data e objetivo do arquivo; (ii) cabeçalho das funções auxiliares e procedimentos implementados no trabalho; (iii) comentários nos principais trechos de código de cada algoritmo, explicando resumidamente o que está sendo codificado a seguir.

3.12 Req. 12 - Corretude dos Resultados

O requisito mais importante do trabalho é aquele que lida com resultados corretos. Desta forma, este requisito consiste em garantir que os resultados gerados pela aplicação em termos de identificação das propriedades da relação e em termos da geração do fecho correspondentes funcionem corretamente e gerem resultados corretos. O grupo deverá testar intensivamente cada uma das instâncias geradas. Sugere-se validar a aplicação com muitas instâncias além das solicitadas em requisito. O requisito somente será considerado se a aplicação gerar resultados corretos para todas as instâncias de testes usadas pelo professor.

4 Barema de Correção

Conforme mencionado, o trabalho prático tem valor de 30 pontos. A correção seguirá o barema apresentado a seguir, que lista os requisitos do trabalho prático e suas respectivas pontuações.

Requisito	**Pontos**
Req. 01 - Entrada de Dados	1.0
Req. 02 - Estrutura de Dados Matriz	1.0
Req. 03 - Check Propriedade Reflexiva	1.0
Req. 04 - Check Propriedade Simétrica	2.0
Req. 05 - Check Propriedade Transitiva	3.0
Req. 06 - Fecho Propriedade Reflexiva	1.0
Req. 07 - Fecho Propriedade Simétrica	5.0
Req. 08 - Fecho Propriedade Transitiva	5.0
Req. 09 - Saída de Dados	1.0
Req. 10 - Instâncias de Teste	2.0
Req. 11 - Documentação de Código	1.0
Req. 12 - Corretude dos Resultados	7.0
TOTAL	**30.0 ptos**

5 Considerações Finais

O trabalho prático especificado neste documento procura desenvolver habilidades técnicas e interpessoais entre os membros da equipe de maneira saudável. Para tal, é fundamental que o trabalho seja feito de maneira ética e profissional. Desta forma algumas considerações finais devem ser feitas:

- Este é um trabalho de programação, portanto aqueles que possuem dificuldades devem estudar o assunto, procurando livros, tutoriais e outros materiais que complementem sua deficiência em programação por conta própria;
- Trabalhos plagiados ou feitos por terceiros valerão zero, assim como trabalhos com alto grau de similaridade;
- Todos os membros da equipe deverão participar efetivamente da implementação do trabalho de forma a aumentar suas perícias em matemática e programação;
- Caso o professor julgue necessário poderá fazer uma arguição de um ou mais membros do grupo;
- Trabalhos entregues fora do prazo serão desconsiderados;
- Dúvidas sobre o trabalho devem ser tiradas com no máximo 01 semana do prazo de entrega.