MC202GH - ESTRUTURAS DE DADOS

2º Semestre de 2017

Professor: Rafael C. S. Schouery

Monitores: Yulle Glebbyo Felipe Borges (PED)

Caio Vinícius Piologo Véras Fernandes (PAD)

Lab. 10 - Sistema de Alocação de Rins

Peso 4

1. O Problema

Em 2012, Lloyd Shapley e Alvin E. Roth receberam o Prêmio Nobel em Economia por suas contribuições à Teoria de Matchings. Tal contribuição, deu espaço à diversas aplicações na medicina. Entre elas, destaca-se a criação de um sistema alocação de transplantes de rins. A ideia principal era ajudar pacientes que precisavam de um rim, mas que tinha incompatibilidade com familiares e amigos. Assim, a ideia era que um paciente recebesse o rim de um familiar de um outro paciente e que, em compensação, o seu familiar também doasse um rim para aquele ou outro paciente. Para cada paciente era construída uma lista de prioridades dos doadores de acordo com a chance de rejeição, distância do doador, etc. O trabalho de Shapley e Roth possibilitou a criação de algoritmos que encontram um emparelhamento entre esses pacientes e doadores, de forma que os doadores poderiam doar seus rins para outros pacientes, na condição de que seu familiar receba um rim compatível.

O sistema de alocação de rins foi desenvolvido através da generalização de um problema conhecido da Teoria dos Jogos, o Problema da Alocação de Casas. Neste problema, temos **N** moradores, cada um com sua casa e uma ordem de preferência sobre as casas dos outros moradores e da sua, inclusive. O objetivo é realizar as trocas de casas de todos os moradores de forma que todos eles fiquem satisfeitos com a troca, isto é, não exista um grupo de moradores que prefiram trocar as casas entre eles do que seguir o que foi proposto pelo algoritmo.

Para ilustrar considere o seguinte exemplo:

Temos 4 moradores: m_1, m_2, m_3 e m_4.

- m_1 tem a seguinte ordem de preferência: m_4 m_3 m_2 m_1
- m_2 tem a seguinte ordem de preferência: m_4 m_3 m_1 m_2
- m_3 tem a seguinte ordem de preferência: m_2 m_1 m_4 m_3
- m_4 tem a seguinte ordem de preferência: m_2 m_1 m_3 m_4

Neste caso, temos que a melhor resposta seria m_2 e m_4 trocarem de casas entre si, e m_1 e m_3 trocarem entre si. Assim a resposta seria:

- m_1: m_3
- m_2: m_4
- m_3: m_1
- m_4: m_2

1.1 *Top Trading Cycle*

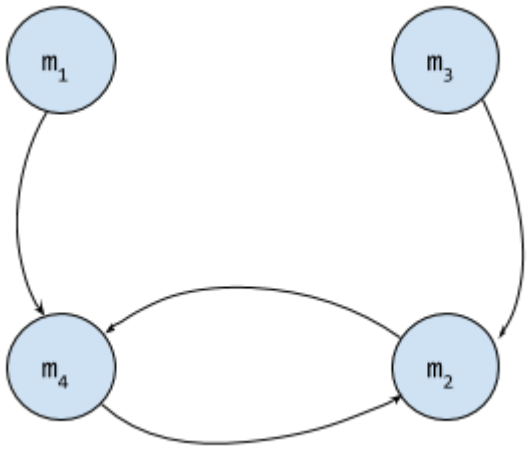
Para resolver o Problema de Alocação de Casas, David Gale propôs o algoritmo Top Trading Cycle. Neste algoritmo, modelamos cada morador como um vértice de um grafo direcionado. **A casa favorita** de cada morador é indicada através de um arco saindo de seu vértice e chegando no vértice que indica sua escolha, ou seja, um arco de `i` para `j` indica que a casa do morador `j` é a favorita do morador `i`. Neste grafo, temos que o grau de saída de cada vértice é exatamente 1, e sempre existe pelo menos um ciclo neste grafo, já que todo morador tem uma preferência pela casa de algum outro morador.

Assim, basta identificar os vértices que fazem parte de um ciclo e trocar as casas de acordo com seus arcos. Uma vez que estes vértices tiveram suas casas trocadas, eles podem realizar a troca entre eles, de acordo com o ciclo e sair do sistema. Logo, construímos um novo grafo com o restante dos jogadores, utilizando suas listas de preferência para selecionar as novas casas favoritas considerando apenas os moradores que ainda fazem parte do sistema. O digrafo resultante tem pelo menos um ciclo, e todos os moradores incluídos em um ciclo terão suas casas trocadas e sairão do sistema. Este processo deve ser repetido até que não reste nenhum vértice. Note que um morador pode preferir sua casa a outras, formando um ciclo com um único vértice.

Para ilustrar, vamos considerar o exemplo da seção anterior:

- m_1 tem a seguinte ordem de preferência: m_4 m_3 m_2 m_1
- m_2 tem a seguinte ordem de preferência: m_4 m_3 m_1 m_2
- m_3 tem a seguinte ordem de preferência: m_2 m_1 m_4 m_3
- m_4 tem a seguinte ordem de preferência: m_2 m_1 m_3 m_4

Construímos o seguinte grafo:



Nesta etapa, os moradores m_2 e m_4 estão inclusos em um circuito, portanto eles devem trocar casas entre si. Com isso construímos um novo grafo considerando apenas os vértices m_1 e m_3, e utilizando o suas preferências para construir os novos arcos:

Nesta nova iteração m_1 e m_3 estão em um circuito, então eles deverão trocar casas entre si. Com isso chegamos ao mesmo resultado da seção anterior.

Coincidentemente, neste exemplo todos os circuitos tem tamanho 2, fazendo com um morador `i` recebe a casa do morador `j`, e o morador `j` recebe a casa do morador `i`. Este nem sempre é o caso, já que um ciclo pode ter qualquer tamanho de 1 a n, onde n é o número de moradores

1.2 Objetivo

Neste trabalho, iremos implementar o algoritmo de Top Trading Cycle para encontrar as trocas de casas que devem ser feitas entre os jogadores. Utilizaremos a estrutura de **grafo direcionado representado com listas de adjacências**. **Utilize uma adaptação do algoritmo de busca em profundidade (DFS) para detectar ciclos em um grafo direcionado**.

Todos os desempates devem ser feitos pelo menor número. Todas as buscas devem começar nos vértices de número mais baixo.

2. Entrada

Ao iniciar o programa, teremos a leitura de um inteiro N representando a quantidade de moradores considerados no problema. Em seguida, temos N linhas, cada uma com N inteiros entre 0 e N-1, representando a preferência dos moradores.

Exemplo de entrada:

```
4
3 2 1 0
3 2 0 1
1 0 3 2
1 0 2 3
```

3. Saída

Ao fim da execução, o programa deve imprimir para cada morador, qual a casa que o algoritmo atribuiu a ele. Assim, a saída será dada em `N` linhas, onde em cada linha teremos o número do morador e o número da casa que este morador recebe. O formato de cada linha deve ser o seguinte:

`i: k`

onde `i` é um inteiro que representa o `i`-ésimo morador, e `k` é um inteiro que representa o número da casa que foi atribuída ao `i`-ésimo morador. Por padronização, deve-se imprimir as alocações em ordem crescente, ou seja, primeiro imprime-se qual casa é atribuída ao morador `0`, depois ao morador `1`, até o morador `N`.

Exemplo de saída:

```
0: 2
1: 3
2: 0
3: 1
```

4. Informações

- Este laboratório possui **peso 4**.
- **Não** há um número máximo de submissões.
- No início de cada arquivo a ser submetido, insira um comentário com seu nome, RA e uma breve descrição do conteúdo do arquivo.
- Apenas comentários no formato `/* comentário */` serão aceitos. Comentários com `//` serão acusados como erros pelo SuSy.
- Os cabeçalhos permitidos são:
  - `stdio.h`
  - `stdlib.h`
  - `math.h`
- A submissão da sua solução deverá conter múltiplos arquivos:
  - **grafo.h**: interface da estrutura de grafo
  - **grafo.c**: implementação da estrutura de grafo
  - **lab10.c**: cliente que utiliza a estrutura
- Será disponibilizado um **Makefile** para este trabalho na página do laboratório:

    - Para compilar seu projeto, basta navegar até o diretório do projeto e utilizar o comando ‘make’ no terminal do Linux.
    - Veja mais instruções na página da disciplina.
- Toda a memória alocada deve ser liberada adequadamente ao fim do programa. Isso pode te ajudar a evitar a ocorrência de falhas relacionadas a memória.
- Indente corretamente todo o seu código. Escolha entre espaços ou tabs para indentação e seja coerente em todos os arquivos. Indentação é fundamental para a legibilidade do seu código, e ajuda no entendimento do fluxo de controle do seu programa.

5. Critérios de Avaliação

- Seu código deverá passar por todos os casos de teste do SuSy definidos para este laboratório. Caso positivo, utilizaremos os seguintes critérios adicionais:
    - A **não utilização/implementação da estrutura de grafo com lista de adjacências** resultará em **nota zero**
    - Falha na implementação do comportamento esperado para a estrutura grafo acarretará em uma penalização
    - Utilização de bibliotecas ou funções não permitidas pelo enunciado resultará em uma **penalização severa** na nota