Instruções sobre o Trabalho 1

480339 — Teoria dos Grafos

Cândida Nunes da Silva

1º Semestre de 2013

1 Problema

Resolver o problema computacional de verificar se dois grafos dados na entrada são isomorfos. Você pode supor que os grafos da entrada são conexos e simples.

2 Entrada

A entrada deve ser lida da entrada padrão (teclado) e será composta por diversos casos de teste. Cada caso de teste contém uma linha com dois inteiros N_1 e M_1 ($0 < N_1 \leq 200, 0 \leq M_1 \leq 40000$), separados por espaços, que representam o número de vértices e de arestas do primeiro grafo, respectivamente. As M_1 linhas subsequentes contém dois inteiros u e v separados por espaços, que representam os rótulos dos vértices extremos de cada aresta do primeiro grafo. A linha seguinte contém dois inteiros N_2 e M_2 ($0 < N_2 \leq 200, 0 \leq M_1 l 240000$), separados por espaços, que representam o número de vértices e de arestas do segundo grafo, respectivamente. As M_2 linhas subsequentes contém dois inteiros u e v separados por espaços, que representam os rótulos dos vértices extremos de cada aresta do segundo grafo.

O último caso de teste é seguido por uma linha contendo dois zeros.

3 Saída

A saída deve ser escrita na saída padrão (terminal). Para cada caso de teste a saída deve ter uma linha contendo ISOMORFOS caso os dois grafos do caso de teste sejam isomorfos e NAO ISOMORFOS caso contrário.

4 Exemplo

Entrada	Saída
6 6 1 2 2 3 2 5 4 5 4 6 5 6 6 6 1 2 1 3 1 4 2 5 2 6 4 5 6 6 1 2 1 6 2 3 3 4 4 5 5 6 6 6 1 2 1 5 2 4 3 5 3 6 4 6	NAO ISOMORFOS ISOMORFOS

Entrada	Saída
10 15 1 2 1 5 1 6 2 3 2 9 3 4 3 8 4 5 4 7 5 10 6 7 6 8 7 9 8 10 9 10 10 15 1 2 1 6 1 10 2 8 2 9 3 4 3 6 3 8 4 9 4 10 5 6 5 7 5 9 7 8 7 10	ISOMORFOS

5 Desenvolvimento e Apresentação

Cada aluno deve implementar a sua solução individual. A implementação da solução do problema deve ser em C em arquivo único. O nome do arquivo deve estar na forma "t1-nomesn.c", onde "nomesn" representa o primeiro nome do aluno seguido das iniciais de seu sobrenome. Note que todas as letras são minúsculas e o separador é "-" (hífen) e não "_" (underscore).

Serão disponibilizados arquivos com diversas entradas (t1.in) e as respectivas saídas esperadas (t1.sol). É **imprescindível** que o **algoritmo** implementado esteja correto, isto é, retorne a solução esperada para **qualquer** entrada do arquivo de testes. É **desejável** que a implementação seja eficiente.

6 Ambiente de Execução e Testes

O programa deve ser compilável em ambiente Unix com `gcc`. Sugere-se que os testes também sejam feitos em ambiente Unix. Deve-se esperar que a entrada seja dada na entrada padrão (teclado) e não por leitura do arquivo de testes. Da mesma forma, a saída deve ser impressa na saída padrão (terminal), e não em arquivo. Será disponibilizado no moodle um trabalho modelo (trabalho 0) que faz a entrada e a saída da forma requerida.

A motivação dessa exigência é apenas simplificar a implementação de entrada e saída, permitindo o uso das funções `scanf` e `printf` da biblioteca padrão para leitura e escrita dos dados, sem precisar manipular arquivos.

Por outro lado, é evidente que efetivamente entrar dados no teclado é muito trabalhoso. Em ambiente Unix, é possível usar redirecionamento de entrada na linha de comando de execução para contornar esse problema. Supondo que o nome do arquivo executável seja análogo ao arquivo fonte, e "t1.in" seja o arquivo com os casos de teste, a linha de comando:

```
shell$ ./t1-nomesn < t1.in
```

executa o programa para todos os casos de teste de uma só vez, retornando todas as saídas em sequência para o teminal. Novamente, pode-se usar o redirecionamento de saída na linha de comando para escrever a saída em um arquivo de saída de nome, por exemplo, "t1.my.sol". A respectiva linha de comando seria:

```
shell$ ./t1-nomesn < t1.in > t1.my.sol
```

Após a execução, a comparação pode ser feita usando o comando `diff` do Unix. Por exemplo, se o arquivo "t1.sol" contém as saídas esperadas, a linha de comando:

```
shell$ diff t1.sol t1.my.sol
```

serve para comparar automaticamente os dois arquivos, retornando nada caso sejam idênticos e as linhas onde há discrepâncias caso contrário.

7 Entrega e Prazos

A data para a entrega do projeto é dia 28 de maio. Cada aluno deve entregar via moodle seu único arquivo fonte com nome no padrão requerido até essa data. Lembre-se que é **imprescindível** que o código compile em ambiente Unix e que a entrada e a saída sejam feitas pela entrada e saída padrão.

8 Notas

As notas serão baseadas na correção da solução implementada, clareza do código fonte e eficiência da solução.

Trabalhos que não atendam aos requisitos mínimos expressos neste documento de forma a invibializar o teste do programa receberão nota ZERO. Em particular, receberá nota ZERO todo programa que:

- não compila em ambiente Unix;
- dá erro de execução;
- não usa entrada e saída padrão para leitura e escrita.