Instruções sobre o Exercício 1

1000608/1001323 — Algoritmos em Grafos

Cândida Nunes da Silva

2º Semestre de 2020 – ENPE

1 Problema

Um grafo é *euleriano* se é possível encontrar um percurso (uma *trilha euleriana fechada*) que passa por cada aresta exatamente uma vez e volta à origem da trilha. Para que um grafo seja euleriano é necessário que ele seja conexo, senão não há como o percurso sair de uma componente conexa e chegar em uma componente conexa diferente. Quando o grafo é conexo, sabemos que ele é euleriano se, e somente se, todos os seus vértices têm grau par. Sua tarefa é, dado um grafo de entrada, determinar se ele é ou não euleriano.

2 Entrada

A entrada deve ser lida da entrada padrão (teclado). A entrada contém uma linha com dois inteiros N e M ($1 < N \leq 300, 1 \leq M \leq 500$), separados por espaços, que representam o número de vértices e o número de arestas do grafo de entrada. As M linhas subsequentes contêm dois inteiros u e v separados por espaços, que informam que existe uma aresta ligando u a v no grafo.

3 Saída

A saída deve ser escrita na saída padrão (terminal). A saída deve ser uma linha que diz "O grafo eh Euleriano." em caso afirmativo ou "O grafo nao eh Euleriano." caso contrário.

4 Exemplos

Entrada	Saída
10 16 0 1 0 2 0 3 0 4 0 5 1 2 1 3 1 4 2 3 2 5 3 4 4 5 6 7 6 9 7 8	O grafo nao eh Euleriano.

Entrada	Saída
15 30 0 1 0 5 0 12 0 14 1 2 1 4 1 14 2 3 2 7 2 13 3 4 3 6 3 7 4 5 4 7 5 6 5 8 6 10 6 12 7 8 7 9 7 14 8 9 8 10 9 10 9 13 10 11 11 12 12 13 13 14	O grafo eh Euleriano.

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 "ex01-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).

O juiz online verificará seu programa comparando para cada um dos casos de teste se a saída gerada pelo seu programa é igual à saída esperada. É **imprescindível** que o **algoritmo** implementado esteja correto, isto é, retorne a solução esperada para **qualquer** entrada. 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.

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 "ex01.in" seja o arquivo com os casos de teste, a linha de comando:

```
shell$ ./ex01-nomesn < ex01.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, "ex01.my.out". A respectiva linha de comando seria:

```
shell$ ./ex01-nomesn < ex01.in > ex01.my.out
```

Após a execução, a comparação pode ser feita usando o comando `diff` do Unix. Por exemplo, se o arquivo "ex01.out" contém as saídas esperadas, a linha de comando:

```
shell$ diff ex01.out ex01.my.out
```

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

7 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.