Instruções sobre o Trabalho 1

1000608/CCS025 — Algoritmos em Grafos

Cândida Nunes da Silva

1º Semestre de 2018

1 Problema

No mundo bruxo, os alunos de Hogwarts - a escola de magia e bruxaria mais importante de toda a história - aprendem técnicas de defesa contra as Artes das Trevas. Recentemente, Dumbledore, o excepcional diretor da escola, foi afastado pelo Ministro da magia. Em seu lugar assumiu a antipática e arrogante Dolores Umbridge, que além de ser cruel com os alunos não os ensina a se defender contra as artes das trevas. Juntamente com seus amigos, Harry Potter montou um grupo denominado Armada de Dumbledore, com o intuito de aprenderem a se defender sozinhos. Como Umbridge não permite reuniões em grupo eles têm que realizar os encontros em uma sala escondida que aparece apenas quando se é necessário, chamada Sala Precisa. Durante um dos encontros, eles decidiram fazer uma atividade de treinamento onde eles devem se dividir em dois times e cada time deve se posicionar na sala de forma que apenas pessoas do time oposto encontrem-se no seu ângulo de possível ataque, ou seja, nenhuma pessoa que seja do seu time deve ficar sob sua mira.

Seu trabalho é verificar se, dada uma formação dos alunos, há a possibilidade de dividi-los em dois times de forma que apenas pessoas do time oposto fiquem no ângulo de ataque de um aluno.

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 e M $(0 < N \leq 2000, 0 \leq M \leq 1000000)$, separados por espaços, que representam o número de alunos que estão participando do treinamento e a quantidade de pares de alunos no ângulo de ataque um do outro. As M linhas subsequentes contém dois inteiros u e v separados por espaços, que informam que esses dois alunos estão no ângulo de ataque um do outro. Os alunos são numerados sequencialmente de 1 a N. 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 SIM caso seja possível dividir os alunos em dois times que não atacam a si próprios e NAO caso contrário.

4 Exemplo

Entrada	Saída
5 8 1 2 1 3 1 4 1 5 2 3 2 4 3 4 4 5 7 11 1 2 1 5 1 6 2 3 2 7 3 4 3 5 3 6 4 7 5 7 6 7 3 3 1 2 2 3 1 3 4 2 1 2 3 4	NAO SIM NAO SIM

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.

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 09 de outubro. Cada aluno deve entregar de forma a ser definida 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.