Instruções sobre o Trabalho 2

1000608/CCS025 — Algoritmos em Grafos

Cândida Nunes da Silva

1º Semestre de 2018

1 Problema – Erdös Number

Paul Erdös (1913-1996) foi um matemático húngaro que gostava de resolver problemas e chegou a ganhar alguns importantes prêmios por resolver problemas matemáticos famosos. Erdös gostava particularmente das soluções mais simples e elegantes, e era brilhante na tarefa de encontrá-las. Diversas histórias curiosas permeiam sua biografia, como o fato de ter sido uma vez preso nos Estados Unidos por suspeita de espionagem ao ter adentrado sem permissão uma área militar restrita, enquanto caminhava distraidamente sem destino certo pensando na solução de um problema. Em parte por conta desse incidente, mas também por outras circunstâncias de sua vida pessoal, Erdös viveu de forma nômade por grande parte da sua vida. Passou temporadas como professor visitante em diversas universidades do mundo, trabalhando, em cada uma delas, com novos parceiros que traziam a ele novos problemas interessantes para resolver. Cada uma dessas parcerias normalmente resultava na publicação de novos teoremas matemáticos, totalizando mais de 1500 publicações em sua vida, com um conjunto de mais de 500 co-autores diferentes. Esse fato nada comum, chamou a atenção para essa "rede social" de coautores de Erdös, e motivou a definição do folclórico "Número de Erdös" (Erdös Number). Erdös é a única pessoa com Número de Erdös zero. Os co-autores de Erdös têm número de Erdös um. Co-autores de co-autores de Erdös têm Número de Erdös dois. Co-autores de co-autores de co-autores de Erdös têm Número de Erdös três, e assim por diante. Quando não é possível estabelecer uma sequência de co-autorias partindo de Erdös até uma pessoa, define-se que o Número de Erdös dessa pessoa é infinito. Por exemplo, dadas as seguintes relações de co-autoria em uma tal rede social

Autor	**Co-autores**
P. Erdös	J. A. Bondy, L. Lovasz, E. G. Strauss, W. T. Tutte
J. A. Bondy	P. Erdös, L. Lovasz, U. S. R. Murty
L. Lovasz	J. A. Bondy, P. Erdös
E. G. Strauss	A. Einstein, P. Erdös
W. T. Tutte	P. Erdös, D. H. Younger
U. S. R. Murty	J. A. Bondy, M. H. Carvalho, C. L. Lucchesi
A. Einstein	E. G. Strauss
D. H. Younger	C. L. Lucchesi, W. T. Tutte
C. L. Lucchesi	M. H. Carvalho, A. A. A. Miranda, U. S. R. Murty, C. N. Silva, D. H. Younger
M. H. Carvalho	C. L. Lucchesi, U. S. R. Murty
A. A. A. Miranda	C. L. Lucchesi
C. N. Silva	C. L. Lucchesi

observamos que o número de Erdös de cada autor é

Autor	**Número de Erdös**
P. Erdös	0
J. A. Bondy	1
L. Lovasz	1
E. G. Strauss	1
W. T. Tutte	1
U. S. R. Murty	2
A. Einstein	2
D. H. Younger	2
C. L. Lucchesi	3
M. H. Carvalho	3
A. A. A. Miranda	4
C. N. Silva	4

Um professor da UFSCar que gosta muito de programar coletou dados de diversas redes sociais centradas em Erdös e quer calcular, para cada uma delas, o maior número de Erdös daquela rede. Porém, esse professor, muito atarefado, nunca encontra tempo para fazer um programa para resolver esse problema e precisa da sua ajuda para fazê-lo.

2 Entrada

A entrada contém vários casos de teste. A primeira linha de cada caso de teste contém dois inteiros N e M, separados por um espaço em branco, que representam, respectivamente, quantas pessoas fazem parte ($1 \leq N \leq 1.000$) e quantas relações de co-autoria existem na rede social ($1 \leq M \leq 1.000.000$). Cada pessoa da rede é representada por um inteiro entre 0 e $N - 1$, sendo a pessoa representada por 0 o próprio Erdos. Cada uma das M linhas subsequentes de cada caso de teste contém dois inteiros A e B ($1 \leq A, B \leq N$, $A \neq B$), separados por um espaço em branco, indicando que a pessoa A é co-autora de B e vice-versa. O final da entrada é indicado por $N = M = 0$.

A entrada deve ser lida da entrada padrão.

3 Saída

Para cada caso de teste da entrada seu programa deve imprimir uma única linha, contendo o maior Número de Erdos da rede, que pode ser infinito. Nesse caso, a resposta deve ser textual "infinito", caso contrário deve ser um inteiro.

A saída deve ser escrita da saída padrão.

4 Exemplo

Entrada	Saída
11 13 0 1 0 2 0 3 0 4 1 2 1 5 3 6 4 7 5 8 5 9 7 8 8 9 8 10 5 4 0 1 0 2 1 2 3 4 10 15 0 1 0 4 0 5 1 2 1 6 2 3 2 7 3 4 3 8 4 9 5 7 5 8 6 8 6 9 7 9 0 0	4 infinito 2

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 "t2-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 (t2.in) e as respectivas saídas esperadas (t2.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 "t2.in" seja o arquivo com os casos de teste, a linha de comando:

```
shell$ ./t2-nomesn < t2.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, "t2.my.sol". A respectiva linha de comando seria:

```
shell$ ./t2-nomesn < t2.in > t2.my.sol
```

Após a execução, a comparação pode ser feita usando o comando `diff` do Unix. Por exemplo, se o arquivo "t2.sol" contém as saídas esperadas, a linha de comando:

```
shell$ diff t2.sol t2.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 23 de outubro. 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.

Em caso de cola, todos os envolvidos ficarão com nota ZERO.