Instruções sobre o Trabalho 1

480339 — Teoria dos Grafos

Cândida Nunes da Silva

1º Semestre de 2015

1 Problema

Dado um grafo direcionado estrito G representado por uma matriz de adjacências, retornar as listas de adjacências do grafo G^T obtido a partir de G pela inversão das orientações de todos os seus arcos.

Implemente um programa em C que resolva este problema. A representação interna dos grafos manipulados pelo seu programa deve ser feita por lista de adjacências. Na saída, as listas de adjacências dos vértices de G^T devem ser retornadas em ordem crescente dos rótulos dos vértices.

2 Entrada

A entrada deve ser lida da entrada padrão e será composta por diversos casos de teste. A primeira linha de cada caso de teste contém um inteiro N ($1 \leq N \leq 10^8$) que indica o número de vértices do grafo. As N linhas subsequentes contém N inteiros separados por espaços, sendo que o j-ésimo inteiro da i-ésima linha A_{ij} será 1 se existir arco ligando i a j no grafo e 0 caso contrário.

O último caso de teste é seguido por uma linha contendo um zero.

A entrada deve ser lida da entrada padrão.

3 Saída

A saída deve ser escrita na saída padrão. Para cada caso de teste a saída deve ter uma linha contendo dois inteiros N e M separados por espaços, onde N representa o número de vértices e M o número de arcos do grafo G^T. Cada linha i subsequente deve conter inteiros separados por espaços representando a lista de adjacências do vértice i em G^T, sendo que essa lista deve estar em ordem crescente de rótulos.

A saída deve ser escrita da saída padrão.

4 Exemplo

Entrada	Saída
3	3 3
0 1 0	2
0 0 1	0
1 0 0	1
5	5 11
0 0 0 1 0	1 3
1 0 0 1 1	2 4
0 1 0 1 0	3 4
1 0 1 0 1	0 1 2
0 1 1 0 0	1 3
10	10 44
0 0 0 1 0 0 0 0 1 0	1 2 3 4 8 9
1 0 0 1 1 0 1 0 1 0	6 7
1 0 1 0 1 0 1 1 0 0	2 4 7 9
1 0 0 1 1 0 1 0 1 0	0 1 3 5 6 8
1 0 1 0 1 0 1 1 0 0	1 2 3 4 8 9
0 0 0 1 0 1 0 0 1 1	5 6
0 1 0 1 0 1 0 1 0 1	1 2 3 4 8 9
0 1 1 0 0 0 0 0 1 0	2 4 6 9
1 0 0 1 1 0 1 0 1 0	0 1 3 5 7 8
1 0 1 0 1 0 1 1 0 0	5 6
0	

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 04 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.

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