Instruções sobre o Exercício 2

1001323 — Algoritmos em Grafos

Cândida Nunes da Silva

1º Semestre de 2022

1 Problema – Caminho hamiltoniano direcionado

Dado um grafo direcionado acíclico D, faça um programa que determine se este grafo possui um caminho hamiltoniano direcionado, isto é, um caminho direcionado que passa por todos os vértices. Se houver um tal caminho, ele é único, dado que D é acíclico. Seu programa deve dar o caminho hamiltoniano caso este exista, ou indicar que não existe, caso contrário.

Sua implementação deve ter complexidade limitada a $O(n^2)$, sendo n o número de vértices do grafo de entrada.

2 Entrada

A primeira linha de cada caso de teste contém dois inteiros N e M, separados por um espaço em branco, que representam, respectivamente, quantos vértices ($2 \leq N \leq 500$) e quantos arcos há no grafo ($1 \leq M \leq 1000$). Cada vértice é representado por um inteiro entre 0 e $N-1$. Cada uma das M linhas subsequentes de cada caso de teste contém dois inteiros A e B ($0 \leq A, B < N$), separados por um espaço em branco, indicando que existe um arco de A para B. Você pode supor que para cada par de vértices A e B existe no máximo um arco que os liga, e também que o grafo de entrada é acíclico.

3 Saída

A saída deve ser escrita na saída padrão (terminal). A saída deve ser uma única linha. No caso de não haver um caminho hamiltoniano a resposta deve ser textual “Nao possui.”. Caso contrário, a saída deve listar os rótulos dos vértices no caminho hamiltoniano em sequência, separados por espaços.

4 Exemplos

Entrada	Saída
8 10 0 3 1 3 2 4 3 4 3 5 3 6 3 7 4 5 4 7 5 6	Nao possui.

Entrada	Saída
6 9 0 1 0 3 1 2 1 4 2 5 3 1 3 4 4 2 4 5	0 3 1 4 2 5

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 "ex02-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 "ex02.in" seja o arquivo com os casos de teste, a linha de comando:

```
shell$ ./ex02-nomesn < ex02.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, "ex02.my.out". A respectiva linha de comando seria:

```
shell$ ./ex02-nomesn < ex02.in > ex02.my.out
```

Após a execução, a comparação pode ser feita usando o comando `diff` do Unix. Por exemplo, se o arquivo "ex02.out" contém as saídas esperadas, a linha de comando:

```
shell$ diff ex02.out ex02.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.
- apresentar fortes indícios de cola.