Instruções sobre o Exercício 3

1001323 — Algoritmos em Grafos

Cândida Nunes da Silva

1º Semestre de 2022

1 Problema – Companhia Aérea

Uma nova companhia aérea pretende se instalar no país e oferecer vôos para alguns destinos em um primeiro momento e, se tiver sucesso, expandir a cobertura. Para minimizar as despesas de instalação inicial, a companhia optou por oferecer o menor número possível de trechos de viagem, embora deva ser possível viajar de qualquer cidade de origem a qualquer cidade de destino atendida pela companhia utilizando apenas trechos operados pela própria companhia. Desta forma, deve sempre haver uma forma de viajar de uma cidade A para um destino B, a qual pode incluir várias escalas, mas essa sequência de escalas entre quaisquer duas cidades A e B deve ser única para que as despesas de instalação inicial sejam minimizadas.

Por outro lado, a companhia quer maximizar os lucros para ampliar o mais cedo possível sua cobertura de operação. Para isso, é necessário selecionar cuidadosamente qual conjunto de trechos deve entrar em operação logo de início. Para cada possível trecho a ser escolhido foi feito um levantamento de uma estimativa mensal de número de clientes interessados em voar naquele trecho. Sua tarefa é determinar qual seleção de trechos deve entrar em operação no momento inicial de forma a maximizar o número total mensal de clientes interessados em seus vôos.

Sua implementação da solução deve ter complexidade limitada a $O(n^2 \log n)$.

2 Entrada

A entrada deve ser lida da entrada padrão (teclado). A primeira linha de cada caso de teste contém dois inteiros N e M, separados por um espaço em branco, que representam, respectivamente, quantas são as cidades cobertas pela companhia ($1 \leq N \leq 2.000$) e quantos são os trechos considerados como opções para serem oferecidos no momento inicial de operações ($1 \leq M \leq 5.000$). Cada cidade é representada por um inteiro entre 0 e $N-1$. Cada uma das M linhas subsequentes de cada caso de teste contém três inteiros A, B e C ($0 \leq A, B \leq N-1$ e $1 \leq C \leq 10.000$), separados por um espaço em branco, indicando que o trecho entre as cidades A e B possui C clientes interessados em voar a cada mês.

3 Saída

A saída deve ser escrita na saída padrão (terminal). Para cada caso de teste seu programa deve imprimir uma linha para cada trecho de A a B que deve ser escolhido para entrar em operação. Os trechos apresentados na saída devem estar em ordem lexicográfica. Isto é, se os trechos A e B e A' e B' fazem parte da solução e $A < A'$, então o trecho A e B deve aparecer antes de A' e B'; já

quando $A = A'$, apenas quando $B < B'$, o trecho A e B deve aparecer antes de A' e B'. Note que o problema pode ter mais de uma solução. Neste caso, você deve escrever na saída aquela solução que é lexicograficamente menor dentre todas as possíveis soluções.

4 Exemplo

Entrada	Saída
8 10 0 3 4 1 3 3 2 4 7 3 4 12 3 5 3 3 6 2 3 7 1 4 5 6 4 7 3 5 6 4	0 3 1 3 2 4 3 4 4 5 4 7 5 6

Entrada	Saída
5 10 0 1 1 0 2 1 0 3 2 0 4 2 1 2 1 1 3 1 1 4 2 2 3 2 2 4 1 3 4 2	0 3 0 4 1 4 2 3

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 "ex03-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 "ex03.in" seja o arquivo com os casos de teste, a linha de comando:

```
shell$ ./ex03-nomesn < ex03.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, "ex03.my.out". A respectiva linha de comando seria:

```
shell$ ./ex03-nomesn < ex03.in > ex03.my.out
```

Após a execução, a comparação pode ser feita usando o comando `diff` do Unix. Por exemplo, se o arquivo "ex03.out" contém as saídas esperadas, a linha de comando:

```
shell$ diff ex03.out ex03.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.