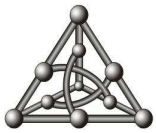


UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

Linhas Telefônicas

Trabalho 2

Algoritmos e Programação II

1 Descrição

Suponha que um grupo de voluntários tenha se prontificado a realizar trabalhos de assistência social aos habitantes de diversas cidades longínquas e miseráveis do nosso estado. Como a comunicação é muito importante, tanto para os habitantes quanto para os voluntários, uma primeira reivindicação de infra-estrutura que o grupo deve levar aos nossos mandantes é a instalação de linhas telefônicas ao longo de algumas estradas existentes que conectam essas vilas umas com as outras. Idealmente, as linhas telefônicas devem ser instaladas de tal forma que todo par de cidades possa se comunicar por telefone. O custo em reais da conexão telefônica entre duas cidades depende da distância em quilômetros entre elas, dos acidentes geográficos, de infra-estrutura básica já existente, etc. O governo estadual, sempre sensível aos problemas sociais e sempre muito honesto, acatou de prontidão as reivindicações dos voluntários, impondo apenas, por respeito aos contribuintes, a otimização dos recursos, isto é, o custo total da instalação da rede telefônica para interconexão das cidades deve ser minimizado.

Dessa forma, a reivindicação desta rede telefônica ligando algumas cidades do estado tranforma-se no problema de determinar ao longo de quais estradas estas linhas telefônicas devem ser construídas, considerando seus custos de instalação, para que enfim se obtenha o sistema telefônico otimizado. A construção de tal rede de telefonia tem uma representação gráfica óbvia, como mostrado na figura **??**. Sua tarefa é ajudar a construir este sistema.

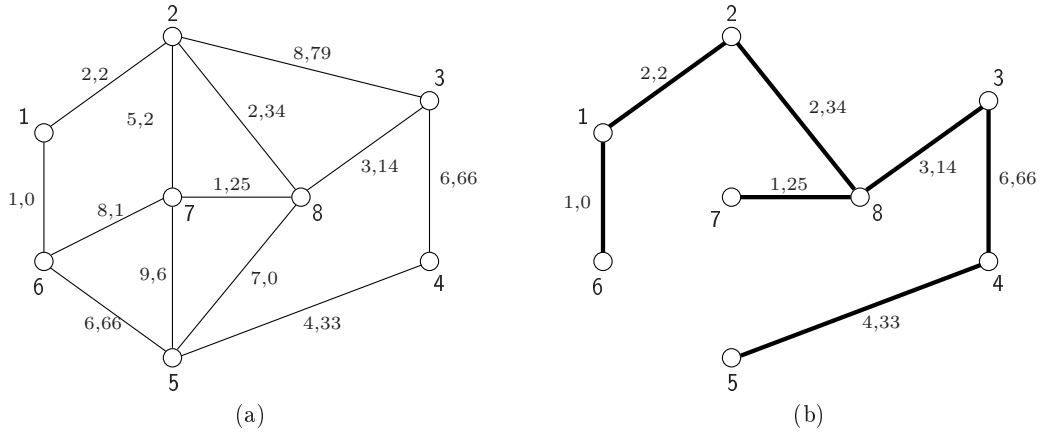
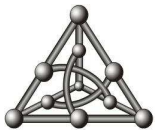


Figura 1: Uma representação gráfica do problema da instalação de uma rede telefônica que conecta cidades do estado. (a) As cidades estão representadas por círculos e rotuladas com um número entre 1 e 8. As possíveis conexões estre as cidades estão representadas por linhas conectando os círculos correspondentes. Sobre cada linha é associado um número de ponto flutuante que representa o custo para conectar as duas cidades, em milhares de reais. (b) Uma solução para o problema com custo 20,92.



2 Entrada

A primeira linha da entrada contém um único número inteiro k com $0 < k < 100$, representando o número de casos de testes. Segue uma linha em branco. Depois desta linha, seguem k descrições de problemas, cada uma começando com um número do tipo inteiro n, com $0 < n \leq 100$, que representa o número de cidades a serem conectadas. Este número determina a identificação de cada cidade, sendo que cada cidade será rotulada com um número do tipo inteiro entre 1 e n. Nas linhas que se seguem, ocorrem triplas de números a b c, uma por linha, onde cada par de números é separado por um espaço, e c é um valor do tipo ponto flutuante, com $0 < c \leq 500{,}0$, que representa o custo de conexão entre as cidades a e b, onde a e b são valores do tipo inteiro. A última dessas triplas é `0 0 0.0`. Para o exemplo da figura **??**, a entrada é

```
1

8
1 2 2.2
2 3 8.79
2 8 2.34
2 7 10.2
5 7 9.6
5 4 4.33
3 4 6.66
5 6 6.66
8 5 7.0
3 8 3.14
1 6 1.0
6 7 8.1
8 7 1.25
0 0 0.0
```

Note que a ordem em que as conexões entre as cidades ocorrem na entrada é indiferente. Isto é, as cidades `1` e `2` estão conectadas no exemplo da figura **??** e a entrada correspondente pode ser

```
1 2 2.2
```

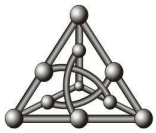
ou indistintamente

```
2 1 2.2
```

Você pode assumir que todas as conexões são feitas entre cidades existentes, isto é, entre cidades rotuladas com números de 1 a n.

3 Saída

Para cada conjunto de cidades e conexões com custos, você deve mostrar como resultado as conexões escolhidas, uma em cada linha, ordenadas pelos seus rótulos, e, na última linha, o custo total de interconexão das cidades, com três casas decimais depois da vírgula. Ao final, você deve imprimir uma linha em branco. Por exemplo, para o exemplo da figura **??**, a saída deve ser



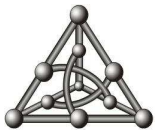

```
1 2
1 6
2 8
3 4
3 8
4 5
7 8
20.920
```

4 Exemplo de entrada

```
2

8
1 2 2.2
2 3 8.79
2 8 2.34
2 7 10.2
5 7 9.6
5 4 4.33
3 4 6.66
5 6 6.66
8 5 7.0
3 8 3.14
1 6 1.0
6 7 8.1
8 7 1.25
0 0 0.0

3
1 2 61.343
2 3 54.293
3 1 42.176
0 0 0.0
```

5 Exemplo de saída

```
1 2
1 6
2 8
3 4
3 8
4 5
7 8
20.920

1 3
2 3
96.469
```



6 Entrega

Leia atentamente a seguir as instruções para entrega do seu trabalho. Trabalhos que não seguirem essas instruções terão desconto de nota.

1. **Cabeçalho**

   Seu trabalho deve ter um cabeçalho com o seguinte formato:

```
/*************************************************************
 *
 *  Nome do(a) estudante:
 *  Trabalho 4
 *
 */
```

2. **Compilador**

   Os(as) professores(as) usam o compilador da linguagem C da coleção de compiladores GNU `gcc`, com as opções `-O2 -Wall -ansi -pedantic` para corrigir os programas. Se você usar algum outro compilador para desenvolver seu programa, antes de entregar verifique se o seu programa tem extensão `.c`, compila sem mensagens de alerta e executa corretamente.

3. **Forma de entrega**

   A entrega será realizada diretamente no Sistema de Suporte a Disciplinas da Facom (SSD/Facom), na disciplina de Algoritmos e Programação II. Para entrega do trabalho, você deve estar cadastrado na página `http://moodle.facom.ufms.br` na disciplina. Após abrir uma sessão digitando seu *login* e sua senha, vá até o tópico – Trabalhos e escolha "Entrega do trabalho 2". Você pode entregar o trabalho quantas vezes quiser até às **23 horas e 55 minutos** do dia **09 de dezembro de 2013**. A última versão entregue é aquela que será corrigida. Encerrado o prazo, não serão mais aceitos trabalhos.

4. **Atrasos**

   Trabalhos atrasados não serão aceitos. Não deixe para entregar seu trabalho na última hora. Para prevenir imprevistos como queda de energia, problemas com o sistema, falha de conexão com a internet, etc, sugerimos que a entrega do trabalho seja feita pelo menos um dia antes do prazo determinado.

5. **Erros**

   Trabalhos com erros de compilação receberão nota ZERO. Faça todos os testes necessários para garantir que seu programa está livre de erros de compilação.

6. **O que entregar?**

   Você deve entregar um único arquivo contendo APENAS o seu programa fonte, como por exemplo, `michael_jackson.c`. NÃO entregue qualquer outro arquivo, tal como o programa executável, já compilado.

7. **Verificação dos dados de entrada**

   Não se preocupe com a verificação dos dados de entrada do seu programa. Seu programa não precisa fazer consistência dos dados de entrada. Isto significa que se, por exemplo, o seu programa pede um número entre 1 e 10 e o usuário digita um número negativo, uma letra, um cifrão, etc, o seu programa pode fazer qualquer coisa, como travar o computador ou encerrar a sua execução abruptamente com respostas erradas.

8. **Arquivo com o programa fonte**

   Seu arquivo contendo o programa fonte na linguagem C deve estar bem organizado. Um programa na linguagem C tem de ser muito bem compreendido por uma pessoa. Verifique se seu programa tem a indentação adequada, se não tem linhas muito longas, se tem variáveis com nomes significativos, entre outros. Não esqueça que um programa bem descrito, bem organizado e com comentários é a chave de seu sucesso.

   Dê o nome do seu usuário no servidor da Facom para seu programa e adicione a extensão `.c` a este arquivo. Por exemplo, `michael_jackson.c`.

9. **Makefile**

   Para facilitar o teste de programas em C, podemos usar um arquivo chamado `Makefile` semelhante ao seguinte:

```
#Makefile para o trabalho 2

PROG  = xxx
ENTRA = yyy
SAI   = zzz

CFLAGS = -O2 -Wall -ansi -pedantic

default :
     @echo ""
     gcc $(CFLAGS) $(PROG).c -o $(PROG)
     @echo ""
     @echo "-----------------------------------"
     @echo "Teste:"
     ./$(PROG) <$(ENTRA) >$(SAI)
     cat $(SAI)
```

   As linhas precedidas por # são comentários. Com este arquivo armazenado no diretório onde você desenvolveu seu programa, digite `make` em uma janela de terminal, para que seu programa `xxx.c` seja compilado, transformado em um programa executável com nome `xxx`, executado com a entrada `yyy` e, por fim, armazene os resultados do processamento no arquivo `zzz` e mostre esses resultados na janela do terminal. Troque os nomes `xxx`, `yyy` e `zzz` por nomes significativos como, por exemplo, `michael_jackson`, `entrada` e `saida`. O arquivo `entrada` é preparado por você, contendo entradas de teste para seu programa, evitando que você as digite a cada execução. Se você quiser saber mais sobre o utilitário `make`, consulte este manual.

10. **Conduta ética**

    O trabalho deve ser feito INDIVIDUALMENTE. Cada estudante tem responsabilidade sobre cópias de seu trabalho, mesmo que parciais. Não faça o trabalho em grupo e não compartilhe seu programa ou trechos de seu programa. Você pode consultar seus colegas para esclarecer dúvidas e discutir idéias sobre o trabalho, ao vivo ou no fórum de discussão da disciplina, mas NÃO copie o programa! Trabalhos considerados plagiados terão nota ZERO.