UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
http://www.icmc.usp.br
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÂO

SCC0216 – Modelagem Computacional em Grafos
Prof.ª Rosane Minghim

ROTEIRO DE LABORATORIO

Informações Gerais

- A seguir são apresentados dois problemas que devem ser trabalhados **individualmente** ao longo desse período de laboratório: 1) Classificação de arestas; e 2) Sistema de distribuição de águas. Em ambos os casos o problema deve ser modelado utilizando grafos e deve ser buscada a solução mais eficiente. Mantenha o código organizado fazendo uso de boas práticas de programação.

- Ao chegar a uma solução satisfatória, submeta-a no Sistema de Submissão de Programas (SSP), disponível no endereço **`ssp.icmc.usp.br`**, para que possa ser validada em alguns casos de teste. Os padrões de entrada e saída devem ser seguidos **rigorosamente**.

- O SSP aceita submissões de um único arquivo fonte (.c ou .cpp) ou de um arquivo .zip com fontes e bibliotecas e um arquivo **`makefile`** com as rotinas de compilação do programa. Mais detalhes estão disponíveis no documento em **`ssp.icmc.usp.br/SSP.pdf`**.

- Entradas e saídas devem ser lidas e escritas a partir dos dispositivos de entrada e saída padrões, logo, não é necessária a manipulação de arquivos, e são suficientes a utilização das funções `scanf()`e `printf()` da biblioteca stdio do C ou os objetos e operadores `cin >>` e `cout <<` da biblioteca iostream do C++.

- Para testar o programa fora do SSP, pode-se usar redirecionamento de arquivos. Para isso utilize os operadores **<** e **>**, como no seguinte exemplo:

```
$ ./exercicio < entrada.txt > saida.txt
```

- O SSP estará aberto para a submissão desses exercícios apenas durante o período deste laboratório, das 13:20 às 16:00 hs. Para cada exercício que você obtiver sucesso **em todos os casos de teste**, será somado 1 ponto adicional à nota do trabalho da unidade.

- No final da tarde da segunda-feira, 31/03/2014, serão disponibilizados, na página da disciplina na Wiki e no SSP, a descrição do trabalho da unidade e os casos de teste, respectivamente.

UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
http://www.icmc.usp.br
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÂO

SCC0216 – Modelagem Computacional em Grafos
Prof.ª Rosane Minghim

ROTEIRO DE LABORATORIO

Classificação de arestas

Uma função importante da travessia em profundidade é que ela pode ser utilizada para classificar as arestas de um grafo e, da classificação das arestas, podem-se derivar algoritmos para resolver alguns problemas, tais como a detecção de ciclos e caracterização de componentes fortemente conectados. Considere um grafo direcionado.

- Uma aresta (u, v) é dita **aresta da árvore** quando o vértice v inicia seu processamento ao ser visitado através desta, isto é, v, que era branco, torna-se cinza.

- Uma aresta (u, v) é dita **aresta de retorno** quando, na árvore de travessia em profundidade, o vértice v é antecessor do vértice u. Quando visitado através de (u, v), v já se encontra cinza. *Self-loops* são considerados arestas de retorno.

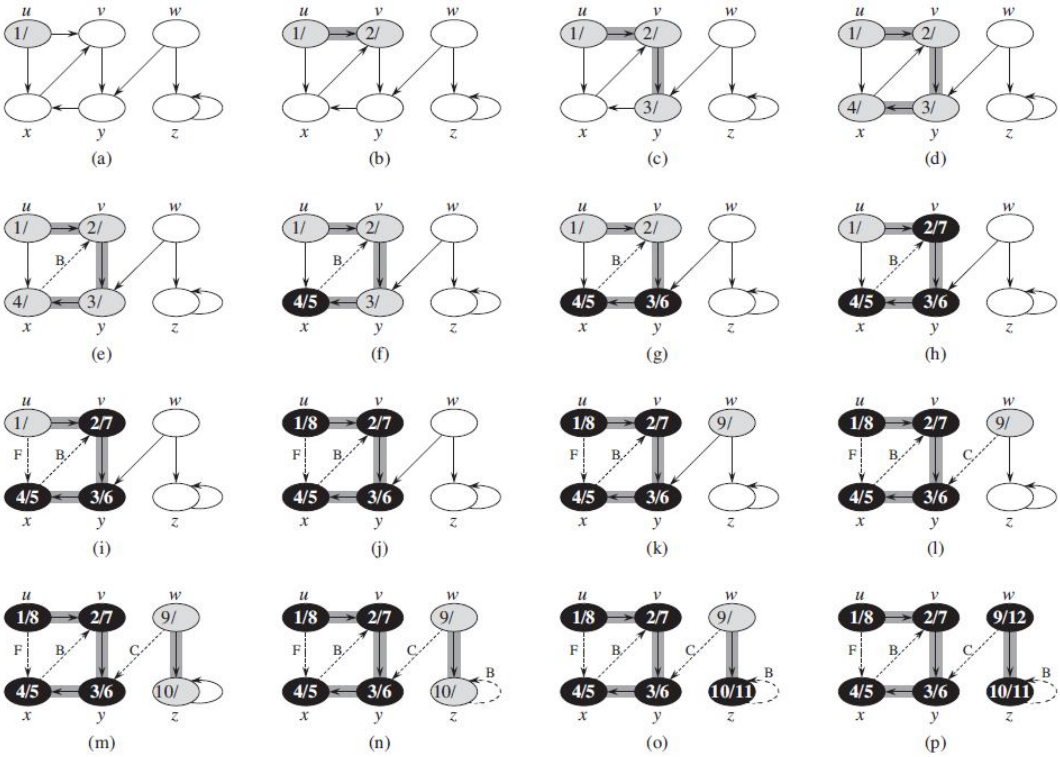
- Uma aresta (u, v) é dita **aresta de avanço** quando esta não pertence à arvore de travessia mas o vértice v é descendente do vértice u. Quando visitado através de (u, v), v encontra-se preto e o *timestamp* em que v tornou-se cinza é maior de que o *timestamp* em que u tornou-se cinza.

- Por fim, uma aresta (u, v) é dita **aresta de cruzamento** quando não se encaixa em nenhum dos casos anteriores. Por exemplo, em uma floresta de arvores de travessia em profundidade, uma aresta de conecta uma árvore a outra. Quando visitado através de (u, v), v encontra-se preto e o *timestamp* em que v tornou-se cinza é maior que o *timestamp* em que u tornou-se cinza.

Arestas de cruzamento não ocorrem em grafos não direcionados. Também em grafos não direcionados, uma aresta de retorno também é uma aresta de avanço, e vice-versa.

Exemplo de classificação de arestas

Considere o seguinte exemplo de travessia em profundidade extraído do livro Cormen et al. (2009). Veja o passo a passo no quadro.



Seu trabalho aqui é o seguinte. Dado um grafo direcionado, apresentado na forma de um conjunto de arestas, você deve realizar uma travessia em profundidade e classificar todas as suas arestas.

Entrada

Seja N, $0 \leq N \leq 26$, o número máximo de vértices do dígrafo, a primeira entrada do programa informa o número total de arestas do mesmo, M. Cada uma das M linhas subsequentes apresenta duas entradas dadas por letras do nosso alfabeto correspondentes aos rótulos dos vértices de origem e destino de cada dos arcos.

Saída

A saída deve ser apresentada na forma de uma matriz, $N \times N$, cujas entradas referem-se à classificação das arestas. Caso a aresta seja uma aresta de árvore, deve ser exibido o caractere ‘T’, caso seja uma aresta de retorno, deve ser exibido o caractere ‘B’, uma aresta de avanço, o caractere ‘F’, e uma aresta de cruzamento, ‘C’. Nas entradas correspondentes às arestas inexistentes, o caractere ‘-’ deve ser exibido.

Exemplo

Entrada

```
8
u v
u x
v y
w y
w z
x v
y x
z z
```

Saída

```
- T F - - -
- - - T - -
- B - - - -
- - T - - -
- - - C - T
- - - - - B
```

Sugestão de roteiro de atividades:

1. Crie um TAD dígrafo. Funções como inicialização do dígrafo, definição do número de vértices, e inserção de arestas são essenciais. Note que o dígrafo possuirá no máximo 26 vértices.

2. Implemente as funções e estruturas necessárias à leitura e armazenamento das entradas do programa. As funções **`scanf()`** e **`printf()`** do C são suficientes para a leitura e escrita nos dispositivos de entrada e saída padrões.

3. Implemente o algoritmo DFS. Embora menos intuitivo, pode-se usar o TAD Pilha, disponível na página da disciplina na Wiki.

4. Identifique em quais etapas do algoritmo as arestas podem ser classificadas devidamente e quais informações precisam ser registradas para a exibição da saída. Note que as cores dos vértices e o *timestamp* da primeira visita a cada um dos vértices são informações muito importantes para a classificação de arestas.

5. Modele a saída do programa de acordo com o padrão estabelecido. Note que embora não seja necessária a exibição dos rótulos dos vértices na saída do programa, será necessário, para cada rótulo apresentado nas adjacências da entrada, que seja verificado se este já possui um índice associado, e, caso não possua, que lhe seja atribuído um.

ATENÇÂO:

Para a indexação dos vértices, seleção de vértices de início da travessia e inserções nas listas de adjacências, utilize como convenção a ordem que os rótulos dos vértices e arestas são apresentados na entrada. No exemplo acima, devido à ordem em que são apresentados, os vértices u, v, w, x, y, e z receberão os índices 1, 2, 5, 3, 4, e 6, respectivamente. Portanto, a travessia deverá ser iniciada no vértice u e, no caso, continuada no vértice w, como mostram as figuras acima.

A implementação da travessia em profundidade adotando as convenções supracitadas e o TAD Pilha ou utilizando pilhas nas listas de adjacências resultará em uma árvore de busca diferente daquela utilizando recursão, pois os vértices são processados na ordem em que são desempilhados. Se você optou por uma dessas soluções, ao submeter sua solução no SSP, considere o exercício "Classificação de arestas – Pilha" e o seguinte exemplo.

Exemplo - Classificação de arestas – Pilha

Entrada

```
8
u v
u x
v y
w y
w z
x v
y x
z z
```

Saída

```
- T T - - -
- - - T - -
- B - - - -
- - C - - -
- - - C – T
- - - - - B
```

UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÂO
http://www.icmc.usp.br

SCC0216 – Modelagem Computacional em Grafos
Prof.ª Rosane Minghim

ROTEIRO DE LABORATORIO

Sistema de distribuição de águas

Devido às poucas chuvas dos últimos meses, as represas da região metropolitana de São Paulo estão com os níveis de águas bem abaixo do normal. Hoje, o Sistema Cantareira e Alto Tietê atingiram, respectivamente, os níveis de 14% e 37,8% de suas capacidades totais. Temendo o agravamento da situação, a SABESP criou um plano de emergência para evitar que alguns dos reservatórios de bairro sequem em uma situação mais extrema. A empresa decidiu que deveriam ser criadas redundâncias no sistema de distribuição de águas, interligando fisicamente os reservatórios de cada subprefeitura. Como o orçamento é restritivo, dada a localização geográfica de cada um dos reservatórios, você deve identificar a maneira mais econômica para interliga-los, minimizando os custos com infraestrutura. É certo que nenhuma subprefeitura possui mais que 40 reservatórios e que cada nova tubulação pode ser construída de forma que o fluxo de águas possa ocorrer em ambos os sentidos.

Entrada

A primeira entrada informa o número de reservatórios da subprefeitura, N, $0 \leq N \leq 40$. Cada uma das N linhas subsequentes apresenta duas entradas com as coordenadas GPS de cada um dos N reservatórios da subprefeitura.

Saída

A saída deve ser apresentada na forma de uma matriz binária, $N \times N$, na qual as entradas referentes às tubulações entre reservatórios a serem implantadas possuem valor 1 e, no contrário, 0.

Exemplo

Entrada

```
4
9.2 10.1
2.4 5.8
3.1 8.5
5 6.4
```

Saída

```
0 0 0 1
0 0 1 1
0 1 0 0
1 1 0 0
```

Sugestão de roteiro de atividades:

1. Crie um TAD grafo. Funções como inserção de aresta e definição do número de vértices do grafo são essenciais. Note que a entrada possuirá no máximo 40 reservatórios.

2. Implemente as funções e estruturas necessárias à leitura e armazenamento das entradas do programa. As funções **`scanf()`** e **`printf()`** da linguagem C são suficientes para a leitura e escrita nos dispositivos de entrada e saída padrões. A partir das coordenadas GPS dos reservatórios será necessário calcular a distância de cada reservatório a todos os demais.

3. Implemente um dos algoritmos para a identificação de árvores geradoras mínimas. Pode-se usar o TAD Heap, disponível na página da disciplina na Wiki.

4. Modele a saída do programa de acordo com o padrão estabelecido.