Segundo Trabalho de Técnicas Alternativas de Programação

(Prof. Alexandre Direne - 2010/2)

Atenção:

Este trabalho é obrigatório e deverá ser entregue, impreterivelmente, até o dia 29 de novembro de 2010 (segunda-feira). A solução é individual e deverá ser arquivada no diretório "~alexd/TAP/" onde o nome do arquivo terá como prefixo o seu nome-de-usuário no sistema do laboratório e, como extensão, ".p" para indicar que seu conteúdo possui um programa em Flavours/Pop11. Assim, por exemplo, se o seu nome de usuário no sistema fosse "grs00" então o nome o arquivo seria "grs00.p" (dentro do diretório "~alexd/TAP/"). Não se esqueça de proteger completamente o arquivo criado, de maneira a permitir a leitura do mesmo apenas por você! Isso pode ser feito aplicando `chmod og-rwx grs00.p` antes de efetuar a cópia com a preservação das permissões (`cp -p grs00.p ~alexd/TAP/`). Não se preocupe com as permissões do professor que irá corrigir o trabalho. A correção dos trabalhos será parcialmente automatizada, sendo assim, é importante que todos os arquivos com as soluções individuais estejam no diretório citado acima, dentro do prazo estipulado. Não será permitida a entrega do arquivo por e-mail.

Enunciado:

A sub-rede de computadores sem redundância de conectividade e de menor custo pode ser calculada por meio do conceito de "Árvore Geradora Mínima" (AGM), da Teria dos Grafos. A AGM garante a conectividade, sem redundância, entre todos os vértices de qualquer componente conexa de um grafo. Para isso, a rede de computadores deve ser representada sob a forma de um grafo *não* direcionado, no qual cada aresta possui um peso, expressando o custo de construção da ligação.

Fazer um programa orientado a objetos (em liguagem Flavours do ambiente Poplog) que permite a criação de instâncias de classes capazes de reponder de maneiras simbólica e gráfica a mensagens como as do exemplo abaixo:

```
vars g1 a b c d e;
make_instance([grafo estrutura grafo_vazio]) -> g1;
make_instance([vertice  nome a  X 87  Y 22]) -> a;
a <- ligue_se_ao_grafo_com_adjacencia(g1, [b e]);
make_instance([vertice  nome b  X 110  Y 150]) -> b;
b <- ligue_se_ao_grafo_com_adjacencia(g1, [a c e]);
make_instance([vertice  nome c  X 150  Y 238]) -> c;
c <- ligue_se_ao_grafo_com_adjacencia(g1, [b d]);
make_instance([vertice  nome d  X 70  Y 238]) -> d;
d <- ligue_se_ao_grafo_com_adjacencia(g1, [c e]);
make_instance([vertice  nome e  X 15  Y 150]) -> e;
e <- ligue_se_ao_grafo_com_adjacencia(g1, [a b d]);
```

Sendo assim, o programa deve refazer o cômputo estrutural e gráfico da AGM sempre que: (1) um novo nodo for inserido na rede por meio da mansagem *ligue_se_ao_grafo_com_adjacencia*; (2) uma nova aresta for inserida por meio da mesma mansagem *ligue_se_ao_grafo_com_adjacencia*. A representação gráfica da AGM deve ser feita em uma janena pré-fabricada com objetos MOTIF (vide uso da biblioteca gráfica abaixo). O exemplo acima resultará no cômputo estrutural da seguinte AGM:

```
[ [a b] [e b] [c b] [d c] ]
```

Para o mesmo exemplo, o programa deve apresentar a forma gráfica do grafo com as arestas da AGM realçadas, conforme mostra a Figura 1.

Para simplificar o trabalho, considere que qualquer grafo só terá uma componente conexa. Adicionalmente, considere que o peso de cada aresta do grafo corresponde à Distância Euclidiana entre ambos os vértices que a compoẽm, calculada com os dados das coordenadas no espaço Cartesiano (ver variáveis de instância X e Y da classe vértice).

As 4 (quatro) únicas primitivas gráficas que devem ser utilizadas estão dispoíveis no arquivo de biblioteca adicional (ver detalhes abaixo). Note que a origem dos eixos X e Y do sistema de coordenadas fica localizada no canto superior esquerdo da janela gráfica. As primitivas são:

```
1 - desenha_nodo(NOME_VERTICE, X, Y);
```

Este é um Procedimento que recebe, como argumento, 3 elementos: uma "word" ($NOME\_VERTICE$) e dois valores inteiros representando as coordenadas (X, Y) do vértice superior esquerdo do retângulo gráfico correspondente ao vértice conceitual que tem o rótulo $NOME\_VERTICE$ no grafo. Ele produz apenas o efeito gráfico de desenhar um retângulo de fundo preto e letras brancas A altura do retângulo é aproximadamente a mesma da letra (a qual não precisa ser conhecida).

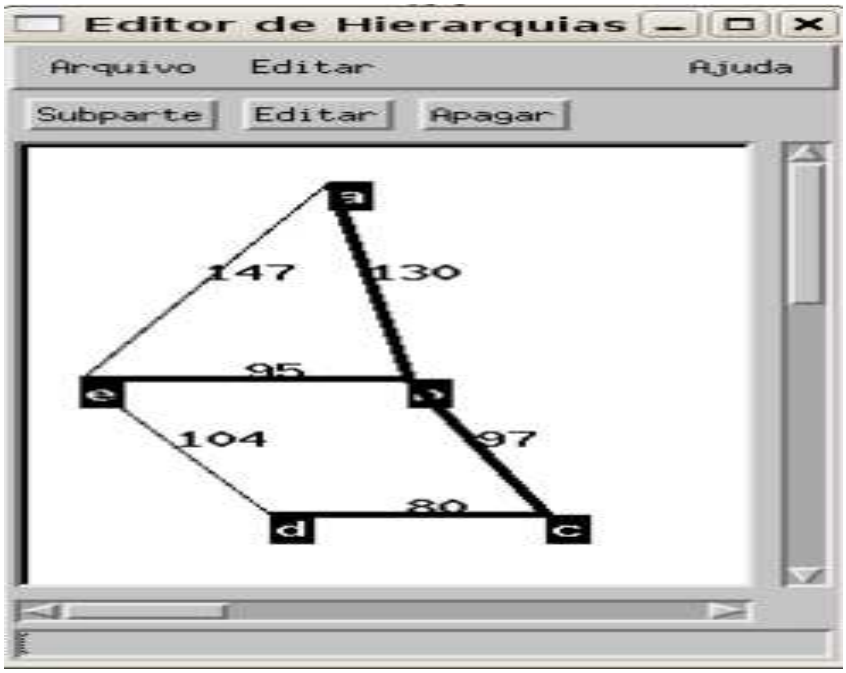

Figura 1: Exemplo de grafo com sua AGM realçada

```
2 - desenha_linha_com_espessura(X1, Y1, X2, Y2, ESPESSURA);
```

Este é um Procedimento que recebe, como argumento, 5 elementos: quatro valores inteiros representando 2 coordenadas do espaço Cartesiano ($(X1, Y1)$ e $(X2, Y2)$) das extremidades de uma linha reta e um valor inteiro representando a espessura da linha. Ele produz apenas o efeito gráfico de desenhar uma linha de cor preta na espessura dada.

```
3 - escreve_peso_aresta(PESO, X, Y);
```

Este é um Procedimento que recebe, como argumento, 3 elementos: um número inteiro ($PESO$) e dois valores inteiros representando as coordenadas (X, Y) do primeiro algarismo do $PESO$.

```
4 - limpa_tela();
```

Este é um Procedimento que não recebe nenhum argumento e produz apenas o efeito gráfico de limpar completamente a superíe gráfica do espaço Cartesiano.

Desenvolva e entregue apenas o código orientado a objetos das classes. Utilize como apoio o arquivo de biblioteca "editor.p", o qual deve ser carregado antes de sua solução para permitir o uso das primitivas gráficas. O referido arquivo está disponível em www.inf.ufpr.br/~alexd/tap/editor.p para ser copiado. Para se ter acesso à execução do referido arquivo, basta abrir uma "shell' na máquina "macalan" (ou qualquer outra) e digitar a seguinte sequência de comandos:

```
macalan> source ~alexd/spop
Sussex Poplog Version 15.53
macalan> cd ..
macalan> cd ..
macalan> pop11 %x im
   ...
Sussex Poplog (Version 15.62 Tue Sep  7 01:07:29 BRT 2010)
Copyright (c) 1982-1999 University of Sussex. All rights reserved.
   ...
Setpop
: uses flavours;
   ...
: load editor.p;
  <<< J A N E L A     G R A F I C A     S U R G E >>>
```

Tente visualizar apenas para teste o que seria um vértice do grafo.

```
: desenha_nodo("Teste", 40, 15);
```

IMPORTANTE: Pense em um modelo Orientado a Objetos para as classes antes de escrever qualquer linha de código. Ele não é complexo mas os princípios e técnicas que irão compor cada fragmento do código a partir de tal modelo devem ser observados da maneira mais explícita que você puder antecipar. Isso deverá servir bem como orientação para seus estudos.