Projeto em Grafos

DESCRIÇÃO DO PROJETO

Entrada/Saída

Formato de entrada para a realização dos testes.
Entrada textual no seguinte formato: qtde de vértices e arestas, informações de cada vértice, determinação se grafo é orientado (S) ou não (N), enumeração das arestas, separadas por linhas. Segue *template* abaixo:
|V|, |E| <newline>
string1 <newline>
string2 <newline>
...
stringn <newline>
S ou N <newline>
int1, int2 <newline>
int3, int4 <newline>
...
Exemplo: suponha o K3.
3,3
a
b
c
N
1,2
2,3
3,1

Saída: o programador tem liberdade para a escolha do formato de saída.

Classe GRAFO

Métodos Públicos
Construtor();
Destrutor();
Display();
Display_Ordem();
Busca_Profundidade();
Busca_Largura();
Componentes_Conexas();
Encontre_Ciclo();
Coloração_Mínima();
Arvore_Geradora_Min();
Caminho_Minimo (vértice);
Fim Classe

Métodos Públicos
Construtor ()
(leitura do qtde de vértices e arestas do grafo)
(fornecimento das informações de cada vértice)
(*inicializa todos os campos dos vértices);*;
% *Pedir para usuário determinar se grafo é orientado (S) ou não (N)*
% Leitura das arestas e *geração da adjacência entre os vértices*
Display ()
(Exibição do grafo na tela do computador na forma : (v,w) (y,z), ...
Display-Ordem ()
(Exibição da ordem de visita de cada vértice, obtida a partir da realização de um algoritmo (busca, ordenação topológica, etc.)
Display-Caminho (u0)
(Exibição dos caminhos realizados no grafo do vértice u0 até todos os demais vértices do grafo).
Busca_Profundidade () *% Para Grafos e Dígrafos*
Busca_Largura () % grafos quaisquer
Componentes_Conexas () % Para grafos não orientados
(Utilizar o método de ***Busca_Profundidade ()*** *para achar o número de componentes conexas do grafo)*
int Distância (v)
(Determina distância de v a todos os demais vértices do grafo)
% Uso de Busca em Largura.
int Diâmetro (w)
(Determina diâmetro do grafo)
% Uso de Distância(v) para todo vértice do Grafo.
Boolean Gafro_Acíclico () % Deterrmina se grafo é acíclico ou não
% Iniciar uma busca em profundidade a partir de vértice x arbitrário.
Coloração_Mínima ()
(*Algoritmo aproximado de coloração*
Heurística: os vértices de maior grau têm prioridade na coloração)
1. Ordena vértices de acordo com o seu grau e coloca vértices numa estrutura.
2. Para k = 1 até n faça
f(k) = 0; (*f (k) = v indica que vértice v é adjacente a algum outro de cor k.)*
3. Para v = 1 até n faça
v.cor = 0;
4. Para v = 1 até n faça (*retira vértice da estrutura por ordem não-crescente de graus)*
Para todo vértice w adjacente a v faça
Se (w.cor é diferente de zero) então
f(w.cor) = v;
FimPara
c = 1;
Enquanto v.cor = 0 faça
Se f(c) é diferente de v) então
v.cor = c;
Senão

c = c + 1;
FimEnquanto

Arvore_Geradora_Min ()
(Solução baseada no algoritmo de Kruskall)
1. Ordenar arestas do grafo G de acordo com o seu peso.
2. Enquanto qtde de arestas de T (árvore geradora) é < |V| - 1 faça
2.1 Escolha aresta ek = (vi, vj) de G, tal que:
i) ek.peso() é tão pequeno quanto possível
ii) T [e1, e2, ..., ek-1, ek] é acíclico. T é a árvore geradora sendo formada, que contém as arestas e1, ..., ek do grafo G.
Obs.: Para verificar se T + ek é acíclico, realizar Busca_Profundidade
entre seus extremos (vi, vj).
2.2 G = G – ek; T = T + ek

Caminho_Mínimo (u0)
(Solução baseada no algoritmo de Dïjkstra)
(A função também pode ser modificada para exibir caminho mínimo de u0 até v
específico: Cam_Minimo (u0, v))
(Inicialização)
label (u0) = 0 ; // u0 é o vértice origem
Set = {u0}; // *Set* é o conjunto de vértices em que já foram obtidos caminhos mínimos.
Cset = V - Set; // *CSet* = complemento de Set.
x = u0; // x é o último vértice acrescentado a Set
/* Procedimento já realizado no construtor
Para todo vértice v diferente de u0 faça
v.label = infinito; // infinito corresponde a valor muito grande
fimpara
*/
(Caminho Mínimo)
Para i = 0 até n - 1 faça // para todos os vértices
Para (todo vértice v em Cset **e** que seja ajacente a x) faça
v.label = min { v.label; x.label + peso (x,v) }
fimpara
// determinar mínimo label dentre todos os vértices de Cset
x = min {v.label, para todo vértice v em Cset};
Set = Set + x; // acrescentar o vértice x a Set;
CSet = CSet - x; // retirar o vértice x de CSet
Fimpara
Fim Classe

Exemplo de Programa Principal
Início
% Um exemplo de programa
GRAFO G; *(instanciação do objeto)*
G.Display(); *(exibição total do objeto)*
G.Busca_Profundidade();

G.Display_Ordem();
G.Ordem_Topológica();
G.Display_Ordem();
G.Cam_Mínimo(1); *(efetua cam_mínimo a partir de vértice 1)*
G.Display_Caminho (); *(exibição dos caminhos de 1 até demais vértices do grafo)*
Fim