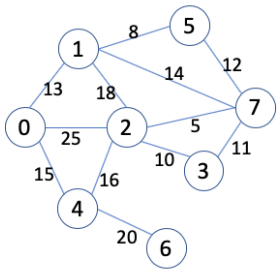
PUC-Rio – Estruturas de Dados – INF1010
Prova 3 – Turma 3wa – 26/6/2019

Nas questões em que for necessário considerar a representação interna de grafos, suponha a representação que usamos nos laboratórios, dada a seguir.

```
typedef struct _grafo Grafo;
typedef struct _viz Viz;
struct _viz {
  int noj;
  float peso;
  Viz*  prox;
};
struct _grafo {
  int nv;           /* numero de nos ou vertices */
  int na;           /* numero de arestas */
  Viz** viz;         /* viz[i] aponta para a lista de arestas vizinhas do no i */
};
```

1. Considere o grafo a seguir:



   (a) (1 ponto) Diga qual seria uma possível ordem de visitação a partir do nó 0 com cada uma das técnicas:

      i. percurso em profundidade

      ii. percurso em largura

   (b) (1 ponto) Crie uma árvore geradora de custo mínimo sobre esse grafo, usando o algoritmo de Kruskal, mostrando passo a passo como aplicou o algoritmo. (Não é necessário desenhar o grafo n vezes desde que a ordem dos passos fique clara.)

2. (2 pontos) Considere o retorno da função de caminhos mais curtos vista em laboratório. Essa função retorna um array com o último nó visitado no caminho mais curto encontrado a partir de um nó origem até cada nó do grafo. Escreva uma função MostraCaminhos, com a interface a seguir, que, dado esse array retornado e o grafo original, mostre a sequência de nós no caminho mais curto encontrado da origem até determinado nó e o custo total deste caminho. (*sugestão:* Faça primeiro a parte da sequência de nós e acrescente depois o cálculo de custo. Caso deseje, acrescente uma função auxiliar.)

```
void mostraCaminhos (Grafo *g, int* cmc, int no);
```

3. (2 pontos) Escreva uma função que use a técnica de visita em profundidade (recursiva ou não recursiva) para determinar se um grafo tem ciclos ou não. Sua função *não* deve usar a estrutura de união e busca. Vc pode criar funções auxiliares. Caso prefira usar a forma não recursiva, por favor suponha que já existem funções prontas para empilhar e desempilhar nós.

```
int temCiclos (Grafo *g);
```

4. (2 pontos) Suponha a disponibilidade de um módulo de união e busca com a interface a seguir.

```
typedef struct suniaoBusca UniaoBusca;
UniaoBusca* ub_cria(int tam);
/* cria particao de conjunto com tam elementos */
/* cada elemento está inicialmente em parte separada */
int ub_busca (UniaoBusca* ub, int u);
/* retorna o representante da parte em que está u */
int ub_uniao (UniaoBusca* ub, int u, int v);
/* retorna o representante do resultado */
void ub_libera (UniaoBusca* ub);
/* libera a estrutura */
```

Suponha também que *já está implementada* uma função `ConstroiUBGrafo` que retorna uma estrutura de união e busca onde já foram "conectados" todos os nós ligados por arestas em determinado grafo, isto é, uma estrutura de união e busca semelhante à construída no Laboratório 12 mas examinando todas as arestas do grafo.

```
UniaoBusca* ConstroiUBGrafo (Grafo *g);
```

Usando `ConstroiUBGrafo`, escreva uma função `numCompsConexos` que retorne o número de componentes conexos em um grafo não direcionado:

```
int numCompsConexos (Grafo *g);
```

(*sugestão:* Utilize um array auxiliar com tamanho igual ao número de nós para contabilizar o tamanho dos componentes.)

5. Considere o código a seguir, que implementa a função de busca da estrutura de *união e busca* da questão anterior.

```
 1 int ub_busca (UniaoBusca* ub, int u){
 2   int x = u;
 3   int aux;
 4   if ((u < 0) || (u > ub->n)) return -1;
 5   while (ub->v[u] >= 0) u = ub->v[u];
 6   while (ub->v[x] >= 0) {
 7     aux = x; x = ub->v[x]; ub->v[aux] = u;
 8   }
 9   return u;
10 }
```

   (a) (1 ponto) O que essa função está retornando em `u`? Explique o que este valor representa.

   (b) (1 ponto) O que fazem as linhas 6-8? Qual a importância dessas linhas para o uso dessa estrutura?

Boa prova!