PUC-Rio – Estruturas de Dados – INF1010
Prova 3 – Turma 3wb – 5/12/2016

Responda as questões da prova nas folhas em separado distribuídas junto com a prova. As respostas podem ser escritas a lápis ou caneta. Indique claramente a questão que está respondendo e, quando cabível, os passos que seguiu para chegar a sua resposta.

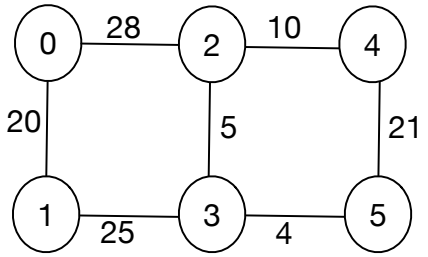
Por favor guarde seu celular/tablet/o_que_for antes de começar a prova e não o utilize até sair da sala.

1. Considere a função abaixo, que calcula os menores caminhos em um grafo com a representação vista nos laboratórios. O retorno é um pouco diferente do visto no Laboratório 12: em vez de retornar o tamanho dos caminhos, o código abaixo retorna o último nó visitado antes de chegar a cada nó pelo menor caminho de `no_inicial` até ele.

```
int* menoresCaminhos (Grafo *grafo, int no_inicial){

  if (no_inicial >= grafo->nv) return NULL;

  int* caminhos = (int *) malloc (sizeof(int)*grafo->nv);
  int* ultimono = (int *) malloc (sizeof(int)*grafo->nv);
  char* visitado = (char *) malloc (sizeof(int)*grafo->nv);
  int no;
  Viz* v;
  ListaP* visitar = listap_cria(grafo->nv);

  for (no=0; no<grafo->nv;no++) {
    caminhos[no] = INT_MAX;
    listap_insere(visitar, INT_MAX, no);
    visitado[no] = 0;
  }
  listap_corrige (visitar, 0, no_inicial);
  caminhos[no_inicial] = 0;
  ultimono[no_inicial] = no_inicial;

  no = listap_remove (visitar);
  while (no!=INT_MAX) {  /* <--- MOSTRAR caminhos, ultimono e visitado */
    visitado[no] = 1;
    v = grafo->viz[no];
    while (v!=NULL) {
      if (!visitado[v->noj]) {
        if (v->peso + caminhos[no] < caminhos[v->noj]) {
          caminhos[v->noj] = v->peso + caminhos[no];
          ultimono[v->noj] = no;
          listap_corrige (visitar, v->peso + caminhos[no], v->noj);
        }
      }
      v = v->prox;
    }
    no = listap_remove (visitar);
  }
  free(visitado);
  free(caminhos);
  return ultimono;
}
```

   (a) (1 ponto) Suponha que, para o grafo mostrado a seguir, seja feita a seguinte chamada: `menoresCaminhos(g, 0);`

Faça o chinês da execução dessa chamada, desenhando os valores dos arrays `visitado`, `caminho` e `ultimono` a cada passagem pelo `while` indicado pela seta.



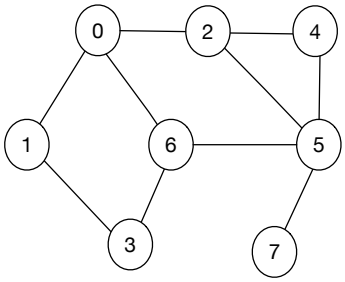
(b) (1,5 pontos) Escreva uma função com a assinatura abaixo, que recebe um grafo e um array como o retornado pela função `menoresCaminhos`, com o último nó usado para chegar a determinado nó, e escreve a sequencia de nós usada para chegar a cada nó do grafo, a partir do nó inicial dos caminhos, segundo indicado por esse array (mostre cada caminho em uma linha separada). Caso o grafo não seja conexo, sua função deve imprimir a mensagem "nó não alcançável" para nós não alcançaveis a partir do nó inicial.

```
void mostracaminhos (Grafo* g, int* caminhos);
```

2. (1 ponto) Mostre o passo a passo seguido pelo algoritmo de Kruskal para a construção da árvore geradora mínima para o grafo utilizado na questão 1.

3. (2 pontos) Escreva uma função que determine se um grafo é conexo ou não utilizando uma estrutura de união e busca.

4. Considere o grafo a seguir.



(a) (0,9 pontos) Indique uma ordem de visitação dos nós possível resultante do percurso em profundidade a partir do nó 6.

(b) (0,9 pontos) Indique uma ordem de visitação dos nós possível resultante do percurso em largura a partir do nó 2.

(c) (0,7 pontos) Desenhe a árvore geradora resultante do percurso realizado em cada um dos itens anteriores.

5. (2 pontos) Considere as declarações vistas em laboratório para implementação de grafos, com um alteração nas funções de enfileiramento e retirada da fila:

```
typedef struct _viz Viz;
struct _viz {
```

```
  int noj;
  int 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 */
};
...
static SQ* enqueue(SQ* queue, int no, int nivel);
/* enfileira um no e seu nivel - retorna fila resultante*/
static SQ* pop(SQ* stack,int* popped_info, int* nivel);
/* retira no e nivel mais antigos - retorna fila resultante e esses dois dados */
```

Escreva uma função, com a assinatura mostrada a seguir, que recebe um grafo e um nó inicial e faz um percurso em largura a partir desse nó inicial, mostrando (printf) cada nó visitado e sua distância em arestas, nesse percurso, a partir do nó inicial. (*Sugestão*: Faça o percurso em largura normal e pense em como modificá-lo para gerar essa info a mais.)

```
void grafoPercorreLarguraMostrandoNivel (Grafo *grafo, int no_inicial);
/* mostra percurso em largura e distância de cada nó a no_inicial neste percurso */
```

Boa prova!