Prova P3 de Estruturas de Dados (INF1010)

Profs Augusto Baffa e Luiz Fernando Seibel

13 de dezembro de 2021

- Responda as questões em uma folha de papel (utilizando sua caligrafia). 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.
- Utilize seu celular para bater fotos da folha de respostas e verifique se a foto está legível.
- Anexe a foto da resposta na tarefa relativa.

---

1. Implemente em linguagem C as funções das operações de conjuntos abaixo: (1 ponto - 0.25 cada)

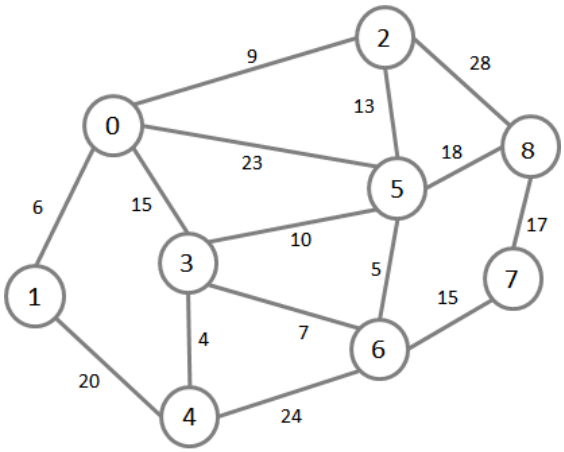
```
/* insere o elemento i no conjunto */
void setInsert(Set *set, int i);

/* testa se o elemento i pertence ao conjunto */
int setIsMember(Set *set, int i);

/* calcula a intersecao de dois conjuntos */
Set *setIntersection(Set *set1,  Set *set2)

/* calcula a diferenca de dois conjuntos set1 – set2*/
Set *setDifference(Set *set1, Set *set2);
```

2. Considere o código a seguir, que implementa a função de busca da estrutura de união e busca (2 pontos)

```
typedef struct uniaoBusca UniaoBusca;

/* cria particao de conjunto com tam elementos */
UniaoBusca* ub_cria(int tam);

/* retorna o representante da parte em que está u */
int ub_busca (UniaoBusca* ub, int u);

/* retorna o representante do resultado */
int ub_uniao (UniaoBusca* ub, int u, int v);

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) (0.5 ponto) O que essa função está retornando em u? Explique o que este valor representa.
(b) (0.5 ponto) O que fazem as linhas 6-8? Qual a importância dessas linhas para o uso dessa estrutura?
(c) (1 ponto) Escreva a função *ub_uniao* em linguagem C e explique seu funcionamento.

3. Dado o grafo (5 pontos):



(a) (2 pontos) Calcule o caminho mais curto do vértice 0 aos demais vértices, utilizando o algoritmo de Dijkstra exibindo passo a passo os vértices já visitados.

(b) (1 ponto) Após encontrar o caminho mais curto (cmc), o algoritmo de Dijkstra retorna um array que representa o último nó visitado para cada vértice encontrado dentro do caminho mais curto entre a origem e o destino. Escreva uma função MostraCaminhos, com a interface a seguir, que recebe o grafo original e a array resultante do algoritmo de cmc e mostre a sequência de nós no caminho mais curto encontrado da origem (vértice 0) até determinado nó e o custo total deste caminho.

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

... onde *int*cmc* é o vetor resultado da execução do algoritmo de dijkstra e *int no* é o nó destino

(c) (2 pontos) Crie uma árvore geradora de custo mínimo sobre o 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.)

4. Escreva uma função em linguagem C, para determinar se um grafo possui ciclos ou não. A função deverá utilizar uma "busca em profundidade" (recursiva ou iterativa). **NÃO utilize a estrutura de União e Busca** - se necessário, proponha funções auxiliares. Também considere que estruturas básicas como filas ou pilhas estão disponíveis para uso e não necessitam detalhamento. A função para detectar ciclos segue a interface abaixo (2 pontos):

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