PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO – PPGComp - MESTRADO - CAMPUS CASCAVEL

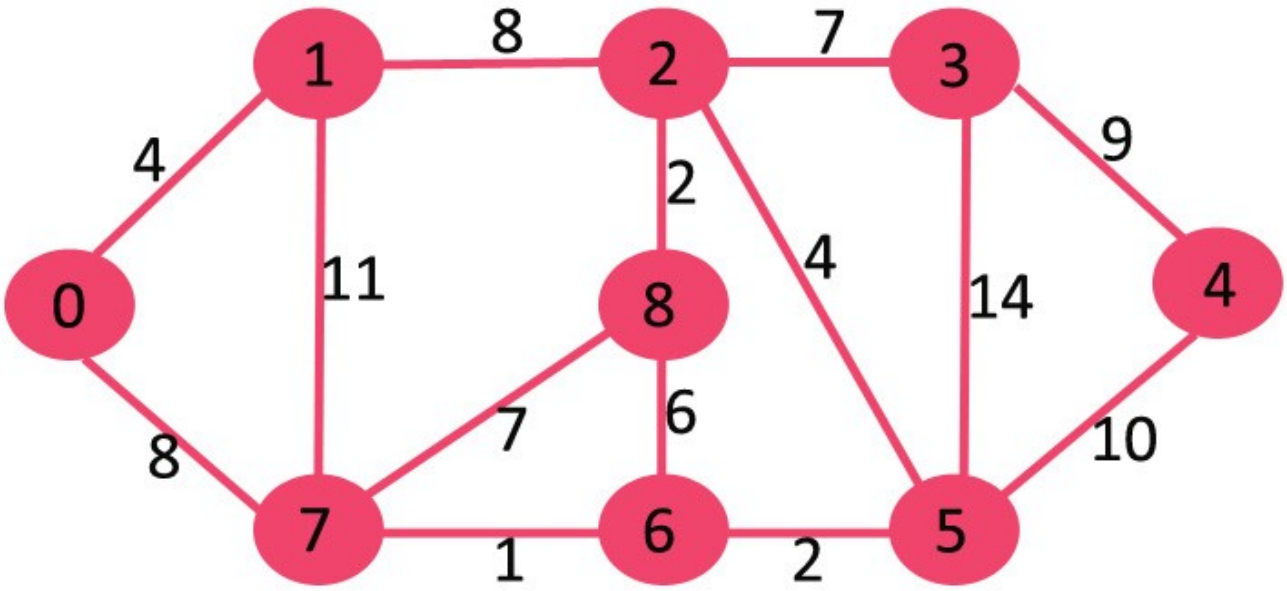
Nome: *Ederson Schmeing*

Disciplina: *Estrutura de dados e Análise de algoritmo* Data: 15/10/2019

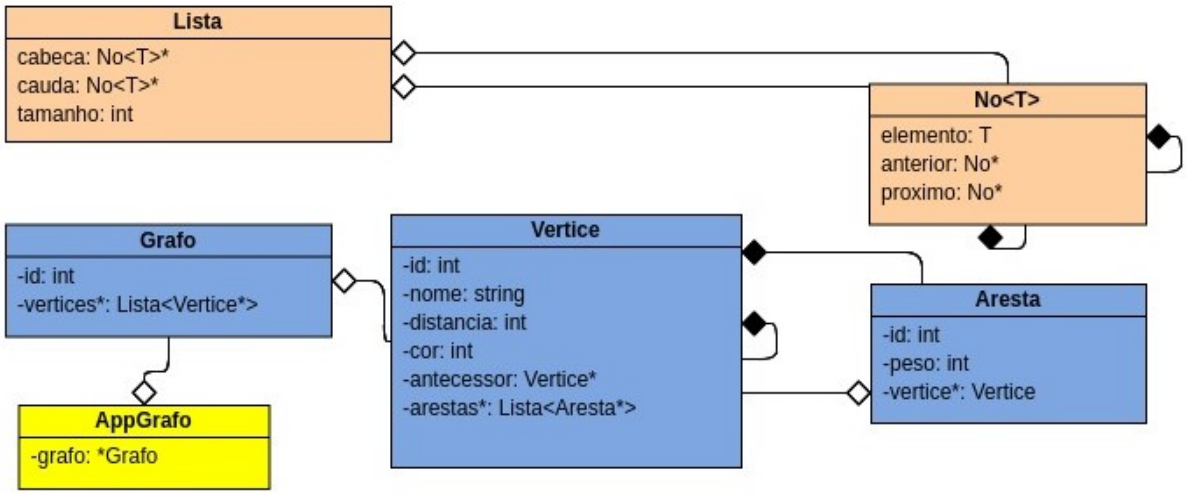
Professor: *Marcio Oyamada*

Teoria dos Grafos

1. Aplicação do algoritmo de caminho mínimo de fonte única no grafo da imagem abaixo. Vértice de origem é **0** e de destino é **4.**



2. Estrutura do grafo foi desenvolvido utilizando classes conforme o diagrama de classes abaixo.



3. A entrada de dados é um arquivo texto com os dados do grafo, onde o primeiro campo da linha é o vértice origem, segundo é o peso da aresta e o terceiro é o vértice destino.

```
V0 4 V1
V0 8 V7
V1 4 V0
V1 11 V7
V1 8 V2
V7 8 V0
V7 11 V1
V7 1 V6
V7 7 V8
V2 8 V1
V2 2 V8
V2 7 V3
V2 4 V5
V8 2 V2
V8 7 V7
V8 6 V6
V6 1 V7
V6 6 V8
V6 2 V5
V3 7 V2
V3 14 V5
V3 9 V4
V5 2 V6
V5 4 V2
V5 14 V3
V5 10 V4
V4 9 V3
V4 10 V5
```

4. Algoritmo de caminho mínimo de fonte única - Dijkstra

```
void Grafo::dijkstra(Vertice *verticeOrigem, Vertice *verticeDestino ) {

    for (int i = 0; i < this->vertices->getTamanho(); i++)     {
        Vertice *vertice = this->vertices->getElemento(i);
        vertice->setDistancia(-1);
        vertice->setAntecessor(NULL);
    }
    Lista<Vertice *> *fila = new Lista<Vertice*>();
    verticeOrigem->setDistancia(0);
    fila->adicionarNaCauda(verticeOrigem);
    while (!fila->estaVazia()) {
        Vertice *visitado = fila->getPrimeiroElemento();
        fila->excluirDaCabeca();
        Lista<Aresta *> *arestas = visitado->getArestas();
        for (int i = 0; i < arestas->getTamanho(); i++) {
            Aresta *aresta = arestas->getElemento(i);
            Vertice *vizinho = aresta->getVertice();
            int peso = aresta->getPeso();
            int minimaDistancia = visitado->getDistancia() + peso;
            if (minimaDistancia < vizinho->getDistancia() || vizinho-
>getDistancia() == -1) {
                vizinho->setAntecessor(visitado);
                vizinho->setDistancia(minimaDistancia);
                fila->adicionarNaCauda(vizinho);
            }
        }
    }
    delete fila;
   cout << "Vértices e distancias" << endl;
   for (int i = 0; i < this->vertices->getTamanho(); i++)     {
        Vertice *vertice = this->vertices->getElemento(i);
        cout << vertice->getNome() << " -> " << vertice->getDistancia() << endl
;
    }
    cout << "Caminho Mínimo" << endl;
    Lista<Vertice *> *pilha = new Lista<Vertice*>();
    Vertice *verticeAntecessor = verticeDestino;
    while (verticeAntecessor != NULL) {
        pilha->adicionarNaCabeca(verticeAntecessor);
        verticeAntecessor = verticeAntecessor->getAntecessor();
    }
    for (int i = 0; i < pilha->getTamanho(); i++)     {
        Vertice *vertice = pilha->getElemento(i);
        cout << vertice->getNome()<< " - " <<vertice->getDistancia() << " -> ";
    }
    delete pilha;
}
```

5. O aplicativo desenvolvido em C++, carrega os dados do grafo do arquivo texto e cria os objetos do grafo conforme a estrutura das classes. Com o grafo criado é executado o algoritmo de Dijkstra.

6. Segue abaixo o resultado da execução mostrando o caminho mínimo do vértice origem **V0** até o vértice destino **V4**.

```
V0 - 0 -> V7 - 8 -> V6 - 9 -> V5 - 11 -> V4 - 21
```