CSE225L – Data Structures and Algorithms Lab
Lab 15
Graph

In today's lab we will design and implement the Graph ADT.

graphtype.h

```
#ifndef GRAPHTYPE_H_INCLUDED
#define GRAPHTYPE_H_INCLUDED
#include "stacktype.h"
#include "quetype.h"
template<class VertexType>
class GraphType
{
    public:
        GraphType();
        GraphType(int maxV);
        ~GraphType();
        void MakeEmpty();
        bool IsEmpty();
        bool IsFull();
        void AddVertex(VertexType);
        void AddEdge(VertexType,
VertexType, int);
        int WeightIs(VertexType,
VertexType);
        void GetToVertices(VertexType,
QueType<VertexType>&);
        void ClearMarks();
        void MarkVertex(VertexType);
        bool IsMarked(VertexType);
        void DepthFirstSearch(VertexType,
VertexType);
        void BreadthFirstSearch(VertexType,
VertexType);
    private:
        int numVertices;
        int maxVertices;
        VertexType* vertices;
        int **edges;
        bool* marks;
};
#endif // GRAPHTYPE_H_INCLUDED
```

heaptype.cpp

```
#include "graphtype.h"
#include "stacktype.cpp"
#include "quetype.cpp"
#include <iostream>
using namespace std;
const int NULL_EDGE = 0;

template<class VertexType>
GraphType<VertexType>::GraphType()
{
    numVertices = 0;
    maxVertices = 50;
    vertices = new VertexType[50];
    edges = new int*[50];
    for(int i=0;i<50;i++)
        edges[i] = new int [50];
    marks = new bool[50];
}
template<class VertexType>
GraphType<VertexType>::GraphType(int maxV)
{
    numVertices = 0;
    maxVertices = maxV;
    vertices = new VertexType[maxV];
    edges = new int*[maxV];
    for(int i=0;i<maxV;i++)
        edges[i] = new int [maxV];
    marks = new bool[maxV];
}
template<class VertexType>
GraphType<VertexType>::~GraphType()
{
    delete [] vertices;
    delete [] marks;
    for(int i=0;i<maxVertices;i++)
        delete [] edges[i];
    delete [] edges;
}
template<class VertexType>
void GraphType<VertexType>::MakeEmpty()
{
    numVertices = 0;
}
template<class VertexType>
bool GraphType<VertexType>::IsEmpty()
{
    return (numVertices == 0);
}
template<class VertexType>
bool GraphType<VertexType>::IsFull()
{
    return (numVertices == maxVertices);
}
template<class VertexType>
void GraphType<VertexType>::AddVertex(VertexType
vertex)
{
    vertices[numVertices] = vertex;
    for (int index=0; index<numVertices; index++)
    {
        edges[numVertices][index] = NULL_EDGE;
        edges[index][numVertices] = NULL_EDGE;
    }
    numVertices++;
}
template<class VertexType>
int IndexIs(VertexType* vertices, VertexType
vertex)
{
    int index = 0;
    while (!(vertex == vertices[index]))
        index++;
    return index;
}
template<class VertexType>
void GraphType<VertexType>::ClearMarks()
{
    for(int i=0; i<maxVertices; i++)
        marks[i] = false;
}
template<class VertexType>
void GraphType<VertexType>::MarkVertex(VertexType
vertex)
{
    int index = IndexIs(vertices, vertex);
    marks[index] = true;
}
template<class VertexType>
bool GraphType<VertexType>::IsMarked(VertexType
vertex)
{
    int index = IndexIs(vertices, vertex);
    return marks[index];
}
```