```cpp
// digraph.hpp -- adjacency matrix based directed graph
// c. 2017 T. O'Neil, C. Reilly; Node class c. 2008 P. Rathore

#ifndef DIGRAPH_HPP
#define DIGRAPH_HPP

#include <string>
#include <vector>
#include <iostream>

using std::string;
using std::cout;
using std::endl;

enum Status { NOT_VISITED, VISITED };

class Node {
    private:
    string name;
    enum Status status;

    public:
    Node(string id) { name = id; status = NOT_VISITED; }
    enum Status getStatus() { return status; }
    void setStatus(enum Status st) { status = st; }
    string getName() { return name; }
};

class Digraph {

protected:
   unsigned int numberOfVertices = 0;
   unsigned int numberOfEdges = 0;
   std::vector<Node*> vertex;
   std::vector< std::vector< int > > distMatrix;

public:

   void addVertex(string s) {
       Node* n = new Node(s);
       vertex.push_back(n);
       numberOfVertices++;
       distMatrix.resize(numberOfVertices);
       for (int i = 0; i < numberOfVertices; i++) distMatrix[i].resize(numberOfVertices);
   }

   unsigned int noVertices();
   unsigned int noEdges();
   void resetEdges();
   void addEdge(int source, int dest, int wt);
   void delEdge(int source, int dest);
   int isEdge(int source, int dest);
   int dijkstra(int source, int dest);
};

#endif
```