Lab 12

**Implementation of Graphs**

### C++ Representation of Graphs

Suppose that the number of vertices in the graph is constant: that is, edges may be added or deleted but vertices may not.

* + A graph with 50 vertices could then be declared as follows: #define MAXVERTEXS 50

**struct** vertex

{

/\* information associated with each vertex \*/

};

**struct** edge

{

**int** adj;

/\* information associated with each edge \*/

};

**class** Graph

{

private:

**struct** vertex vertices[MAXVERTEXS];

**struct** edge edges[MAXVERTEXS][MAXVERTEXS];

……

…….

};

Graph g;

* Each vertex of the graph is represented by an integer between 0 and *MAXVERTEXS*-1 and the array field *vertices* represents the appropriate information assigned to each vertex.
* The array field edges is a two-dimensional array representing every possible ordered pair of vertices.
* The value of g.edges[i][j].adj is either TRUE or FALSE depending on whether or not vertex j is adjacent to vertex i.
* The two-dimensional array *g*.*edges*[][].*adj* is called an ***adjacency matrix***. In the case of a weighted graph, each edge can also be assigned information.
* Frequently the vertices of a graph are numbered from 0 to *MAXVERTEXS*-1 and no information is assigned to them. Also, we may be interested in the existence of edges but not in any weights or other information about them. In such cases the graph could be declared simply by

**int** adj[MAXVERTEXS][MAXVERTEXS];

* In effect, the graph is totally described by its adjacency matrix. We present the code for the primitive operations just described in the case where a graph is described by its adjacency matrix.

**join** (int adj[][MAXVERTEXS], int vertex1, int vertex2)

{

/\* add an edge from vertex1 to vertex2 \*/ adj[vertex1][vertex2] = TRUE;

}

// Sets the matrix entry adj[vertex1][vertex2] to TRUE, indicating an edge exists from vertex1 to vertex2

**remv** (int adj[][MAXVERTEXS], int vertex1, int vertex2)

{

/\* delete edge from vertex1 to vertex2 if one exists \*/ adj[vertex1][vertex2] = FALSE;

}

//Sets the matrix entry adj[vertex1][vertex2] to FALSE, indicating no edge exists.

**adjacent** (int adj[][MAXVERTEXS], int vertex1, int vertex2)

{

**return** ((adj[vertex1][vertex2]==TRUE) ? TRUE : FALSE);

}

// Returns TRUE if an edge exists between vertex1 and vertex2; otherwise, returns FALSE.

### C++ Representation of Weighted Graphs

A **weighted graph** with a fixed number of vertices may be declared by

**struct** edge

{

**int** adj;

**int** weight; };

**struct** edge g[MAXVERTEXS][MAXVERTEXS];

* + The routine *joinwt*, which adds an edge from *vertex*1 to *vertex*2 with a given weight wt, may be coded as follows:

void **joinwt** (struct edge g[] [MAXVERTEXS], int vertex1, int vertex2, int wt)

{

g[vertex1][vertex2].adj = TRUE; g[vertex1][vertex2].weight = wt;

}

**Task1:**

Implement a class in C++ for a **Weighted Graph** using **adjacency matrix** representation.