Project 8:

DYNAMIC PROGRAMMING

OBJECTIVE

- Understand how the Floyd algorithm works.
- Improve graphs usage.
- Improve templates usage.

TO BE HANDED OVER

- floyd.cpp, floyd2.cpp, graph.h
- makefile (without parameters should compile all programs)

PREVIOUS EXERCISES

- Declaration of the Graph class. It is not necessary to write the implementation of the methods.

DEVELOPMENT

In this project we will implement the Floyd algorithm. This algorithm calculates the shortest path between all nodes of a graph by means of dynamic programming. Whenever possible, you should choose an efficient way of implementing the code.

1. Implementation of the Graph class

In this section you will implement a graph class that will be used later in the implementation of the Floyd algorithm. The class will be a class template with 2 parameters: the type of the edge weight and the number of nodes in the graph. This means that the graph size cannot be modified once declared.

The graph will be implemented by means of an adjacency matrix. You should not use dynamic memory for declaring the matrix. It is recommended to use an *assert* or an exception in the method for accessing the elements for range checking.

Check that the class works correctly.

2. Floyd algorithm

Implement the basic Floyd algorithm in *floyd.cpp*. The function will have as input parameter an object of class ***Graph*** and will have as output parameter a matrix (or graph) with the minimum distance between all nodes. The Floyd algorithm should be an independent function and **not a method** of the *Graph* class.

For generating the graph for the problem, you should implement a method of ***Graph*** for creating an undirected random graph with weights between *min* and *max*. The method should work with any numerical weight.

Create an undirected random graph of size 6 with weights between 0.5 and 5.0. Apply the Floyd algorithm to this graph and check that it works correctly.

3. Calculation of the shortest path

The basic Floyd algorithm only calculates the length of the path. Modify the algorithm (in *floyd2.cpp*) for obtaining also the shortest path, i.e. the nodes that have to be visited.

Show the shortest path from node **0** to all other nodes. Check that the program works correctly.