­Floyd’s Algorithm for Shortest Paths

Complexity Summary

The complexity of Floyd’s algorithm of shortest paths scales with the input size n, the number of vertices that connect into paths. As part of the algorithm, the lengths of the paths are compared in a three level loop, which serves as a basic operation to measure the algorithm. Each layer of the loop is set to run n times. On each of n iterations of the outermost loop, it must run the second layer’s loop n times. Each iteration of the second layer’s loop also runs the third layer’s loop n times. The total complexity takes each comparison into account, combining into n \* n \* n total comparisons. Based on the comparison performed in the triple-nested loop, the complexity of Floyd’s algorithm is Θ(n3).

Floyd Algorithm and Floyd() Method

Floyd’s algorithm for shortest paths works using dynamic programming. Each run of the algorithm is split into multiple instances for efficiency. In dynamic programming, the results of each instance are saved to be re-used by other instances to avoid repeating operations unnecessarily. Different candidate paths between the same two vertices are saved and compared to determine which is the shortest.

The Floyd() method finds the shortest path by building matrices of information on the total weights of paths and the connecting matrices of possible paths. The matrices are optimized to find the shortest paths by comparing the weights. The two matrices Floyd() uses to determine the shortest paths are D, a matrix to track the paths by weight, and P, the path matrix, which tracks the vertices used on the shortest paths. Both matrices are built using the triple-nested for loop mentioned in the complexity section. The comparison operation finds any paths that become shorter by adding an intermediate vertex to a 2-vertex path. Using matrix D, the total weight of a candidate path is compared against the existing path’s weight. When a path with a lower weight is found, the new intermediate vertex is added to P and the new combined path weight is added to D.

createGraph() Method

Coding the algorithm in Floyd() required building the adjacency matrix (W) input first from the original graph using the createGraph() method. This method reads from an input file created with observations on the graph’s edges. The direction of each edge, the connected vertices, and the weight of each edge are described by the file. First, W is populated with default values based on the properties of adjacency matrices. The weight of a path from a vertex to itself is always 0, and any missing adjacent paths are initialized as infinity. A Streamreader object reads the file, adding the information to W line by line.