Experiment No. 9

Title: Write a program to find All Pairs Shortest Path

Theory:

Let G = (V, E) be a directed graph with n vertices. Let cost be a cost adjacency matrixfor G such that cost(i,i) = 0,1< i < n. Then cost(i,j) is the length (or cost) of edge<i,j> if <i,j> belongs E(G) and cost(i,j)= INFINITY if i not equal j and (i,j) does not belong to E(G).The all-pairs shortest-path problem is to determine a matrix A such that A(i, j) is the length of a shortest path from i to j. Let us examine a shortest i to j path in G, i not equal j. This path originates at vertex i and goes through some intermediate vertices (possibly none) and terminates at vertex j. We can assume that this path contains no cycles for if there is a cycle, then this can be deleted without increasing the path length (no cycle has negative length). If k is an intermediate vertex on this shortestpath, then the subpaths from i to k and from k to j must be shortest paths from i to k and k to j, respectively. Otherwise, the i to j path is not of minimum length. So, the principle of optimality holds. This alerts us to the prospect of using dynamic programming. If. k is the intermediate vertex with highest index, then the i to k path is a shortest i to k path in G going through no vertex with index greater than k-1. Similarly the k to j path is a shortest k to j path in G going through no vertex of index greater than k-1.

The following equation is obtained to find all pairs shortest path

$$A^k(i,j) = \min \ \{A^{k-1}(i,j),\ A^{k-1}(i,k) + A^{k-1}(k,j)\}, \quad k \geq 1$$

Algorithm

```
Algorithm AllPairsShortestPaths(cost, A, n)
// cost[l :n, 1:n] is the cost adjacency matrix of a graph with n vertices;
//A[i, j] is the cost of a shortest path from vertex i to vertex j. cost[i,i]=0 for 1<i <n.
{
for (i :=1 to n)
{
   for (j :=1 to n )
   {
     A[i, j]:=cost[i, j]
   }
}
 for (k :=1 to n)
  {
   for (i :=1 to n)
   {
      for (j :=1 to n)
      {
      A[i,j]:=min(A[i,j], A[i,k]+A[k,j]);
      }
   }
 }
}
```

Complexity:

The time needed by algorithm is $O(n^3)$ as the equation is iterated in three for loops each executing n times.