**CS401 Intro to Algorithm**

**Group Name:** ERROR 404

**Group Leader:** Mark Decello

**Group Members:** Mark Decello, Julius Lopez, Gene Olivia, Hanel Duran, Naglaa Saeid

**Topic:** Floyd’s Algorithm (java code)

**Program Drawing and :** ………………………………….page 2

**Program Code:** Floyd.java ……….……………………………page 3

**Program Output** …….…………………………………………….page 5

**Drawing**

A picture containing table, small, items, orange

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** |
| **1** | 0 | 1 | ∞ | 1 | 5 |
| **2** | 9 | 0 | 3 | 2 | ∞ |
| **3** | ∞ | ∞ | 0 | 4 | ∞ |
| **4** | ∞ | ∞ | 2 | 0 | 3 |
| **5** | 3 | ∞ | ∞ | ∞ | 0 |

1 //Group Name: Error\_404  
 2 //Group Leader: Mark Decello  
 3 //Group Members: Mark Decello, Julius Lopez, Gene Olivia, Hanel Duran, Naglaa Saeid  
 4   
 5 class Floyd  
 6 {  
 7 // Recursive Function to print path of given  
 8 // vertex u from source vertex v  
 9 private static void printPath(int[][] path, int v, int u)  
 10 {  
 11 if (path[v][u] == v)  
 12 return;  
 13   
 14 printPath(path, v, path[v][u]);  
 15 System.out.print(path[v][u] + " ");  
 16 }  
 17   
 18 // Function to print the shortest cost with path  
 19 // information between all pairs of vertices  
 20 private static void printSolution(int[][] cost, int[][] path, int N)  
 21 {  
 22 System.out.println("Matrix: Floyd's Algorithm Shortest Path\n");  
 23 System.out.println("{ 0, 1, M, 1, 5 }");  
 24 System.out.println("{ 9, 0, 3, 2, M }");  
 25 System.out.println("{ M, M, 0, 4, M }");  
 26 System.out.println("{ M, M, 2, 0, 3 }");  
 27 System.out.println("{ 3, M, M, M, 0 }\n");   
 28   
 29 for (int v = 0; v < N; v++)  
 30 {  
 31 for (int u = 0; u < N; u++)  
 32 {  
 33 if (u != v && path[v][u] != -1)  
 34 {  
 35 System.out.print("Shortest Path from vertex " + v +  
 36 " to vertex " + u + " is (" + v + " ");  
 37 printPath(path, v, u);  
 38 System.out.println(+u + ")");  
 39 }  
 40 }  
 41 }  
 42 }  
 43   
 44 // Function to run Floyd's algorithm  
 45 public static void Floyd(int[][] adjMatrix, int N)  
 46 {  
 47 // cost[] and parent[] stores shortest-path  
 48 // (shortest-cost/shortest route) information  
 49 int[][] cost = new int[N][N];  
 50 int[][] path = new int[N][N];  
 51   
 52 // initialize cost[] and parent[]  
 53 for (int v = 0; v < N; v++)  
 54 {  
 55 for (int u = 0; u < N; u++)  
 56 {  
 57 // initally cost would be same as weight  
 58 // of the edge  
 59 cost[v][u] = adjMatrix[v][u];  
 60   
 61 if (v == u)  
 62 path[v][u] = 0;  
 63 else if (cost[v][u] != Integer.MAX\_VALUE)  
 64 path[v][u] = v;  
 65 else  
 66 path[v][u] = -1;  
 67 }  
 68 }  
 69   
 70 // run Floyd's  
 71 for (int k = 0; k < N; k++)  
 72 {  
 73 for (int v = 0; v < N; v++)  
 74 {  
 75 for (int u = 0; u < N; u++)  
 76 {  
 77 // If vertex k is on the shortest path from v to u,  
 78 // then update the value of cost[v][u], path[v][u]  
 79   
 80 if (cost[v][k] != Integer.MAX\_VALUE  
 81 && cost[k][u] != Integer.MAX\_VALUE  
 82 && (cost[v][k] + cost[k][u] < cost[v][u]))  
 83 {  
 84 cost[v][u] = cost[v][k] + cost[k][u];  
 85 path[v][u] = path[k][u];  
 86 }  
 87 }  
 88   
 89 // if diagonal elements become negative, the  
 90 // graph contains a negative weight cycle  
 91 if (cost[v][v] < 0)  
 92 {  
 93 System.out.println("Negative Weight Cycle Found!!");  
 94 return;  
 95 }  
 96 }  
 97 }  
 98   
 99 // Print the shortest path between all pairs of vertices  
100 printSolution(cost, path, N);  
101 }  
102   
103 public static void main(String[] args)  
104 {  
105 // Number of vertices in the adjMatrix  
106 final int N = 5;  
107 int M = Integer.MAX\_VALUE;  
108   
109 // given adjacency representation of matrix  
110 int[][] adjMatrix = new int[][]  
111 {  
112 { 0, 1, M, 1, 5 },  
113 { 9, 0, 3, 2, M },  
114 { M, M, 0, 4, M },  
115 { M, M, 2, 0, 3 },  
116 { 3, M, M, M, 0 }  
117 };  
118   
119 // Run Floyd's algorithm  
120 Floyd(adjMatrix, N);  
121 }  
122 }

**Output**

**A screenshot of a cell phone

Description automatically generated**