Code for All Pair Shortest Path Algorithm

import java.util.\*;

public class EXPT\_06\_All\_Pair\_Shortest\_Path {

    void APSP(int[][] A, int n) {

        int i, j, k;

        for (k = 0; k < n; k++) {

            for (i = 0; i < n; i++) {

                for (j = 0; j < n; j++) {

                    A[i][j] = Math.min(A[i][j], A[i][k] + A[k][j]);

                }

            }

        }

    }

    public static void main(String args[]) {

        Scanner in = new Scanner(System.in);

        int n, i, j;

        System.out.print("Enter the number of vertices: ");

        n = in.nextInt();

        /\*

         \* int A[][] = new int[n][n];

         \* System.out.print("\nEnter the weighted matrix: ");

         \* for (i = 1; i <= n; i++) {

         \* for (j = 1; j <= n; j++) {

         \* System.out.print("\nEnter value for A[" + i + "][" + j + "]: ");

         \* A[i][j] = in.nextInt();

         \* }

         \* }

         \*/

        // It is tedious to enter each element of a graph many times for

        // different test cases. So, initializing graph at once before running.

        int A[][] = { { 0, 4, 11, 7, 999, 999 },

                { 6, 0, 2, 999, 999, 999 },

                { 3, 999, 0, 999, 999, 12 },

                { 5, 999, 999, 0, 9, 999 },

                { 999, 999, 999, 8, 0, 10 },

                { 999, 999, 14, 999, 13, 0 } };

        System.out.print("\nDisplaying entry:\n ");

        for (i = 0; i < n; i++) {

            for (j = 0; j < n; j++) {

                System.out.print("\t" + A[i][j]);

            }

            System.out.print("\n");

        }

        EXPT\_06\_All\_Pair\_Shortest\_Path obj = new EXPT\_06\_All\_Pair\_Shortest\_Path();

        long start = System.nanoTime();

        obj.APSP(A, n);

        long end = System.nanoTime();

        System.out.println("\nThe shortest path matrix is: ");

        for (i = 0; i < n; i++) {

            for (j = 0; j < n; j++) {

                System.out.print("\t" + A[i][j]);

            }

            System.out.print("\n");

        }

        in.close();

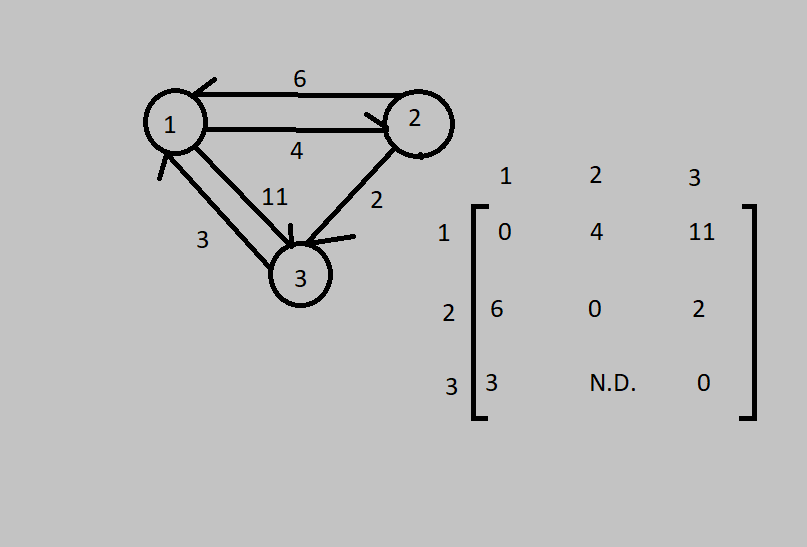
        System.out.print("\nThe time taken by All Pair Shortest Path (Floyd Warshall's Algorithm) is " + (end - start)

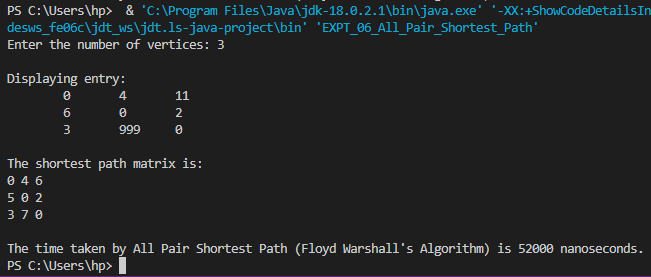
                + " nanoseconds.");

    }

}

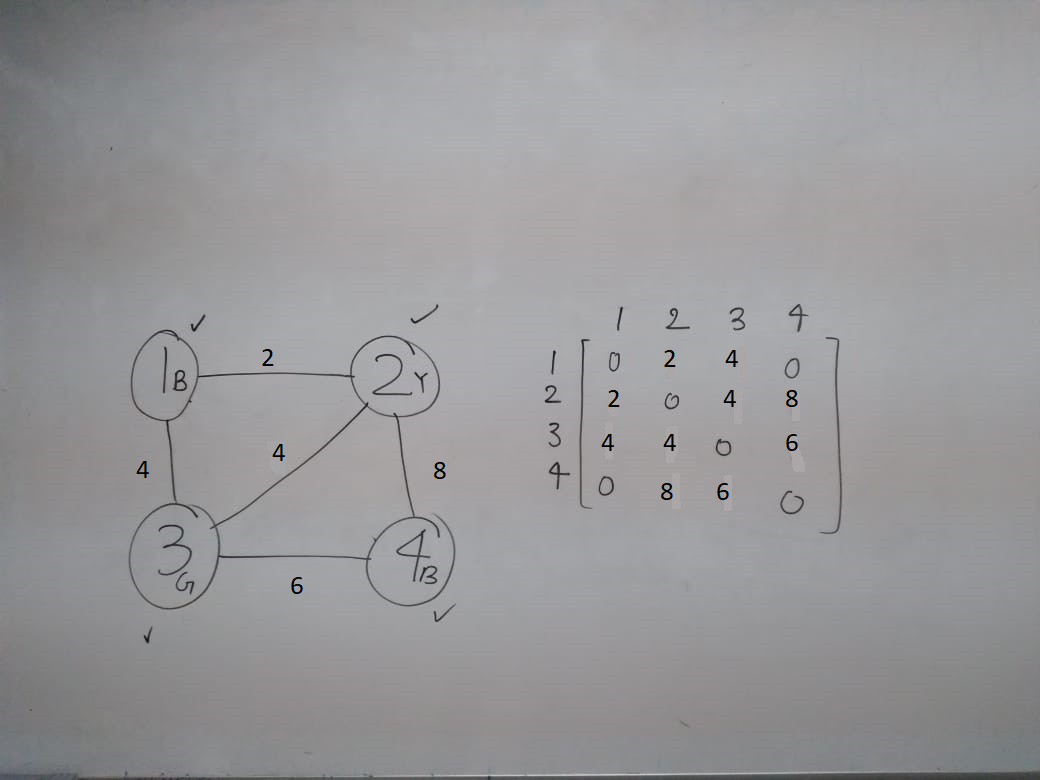
Case 1: When the number of vertices is 3

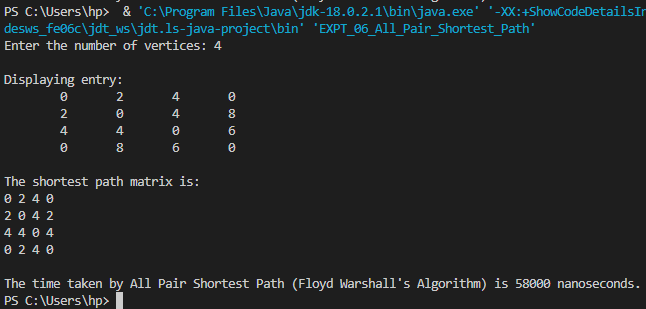




Time taken is 52,000 nanoseconds.

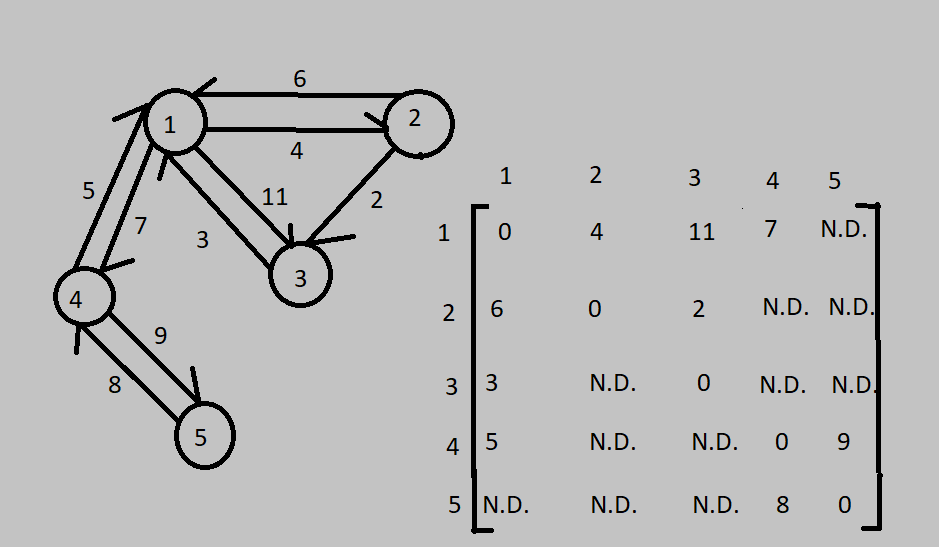
Case 2: When the number of vertices is 4.

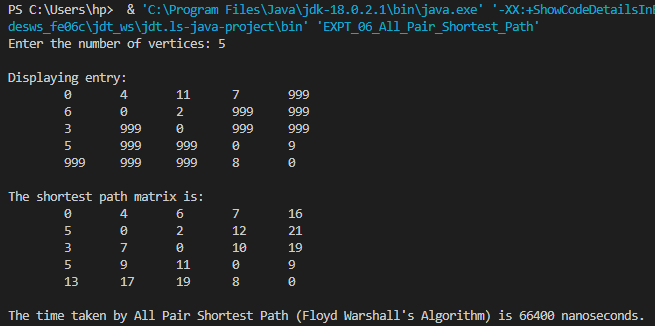




Time taken is 58,000 nanoseconds.

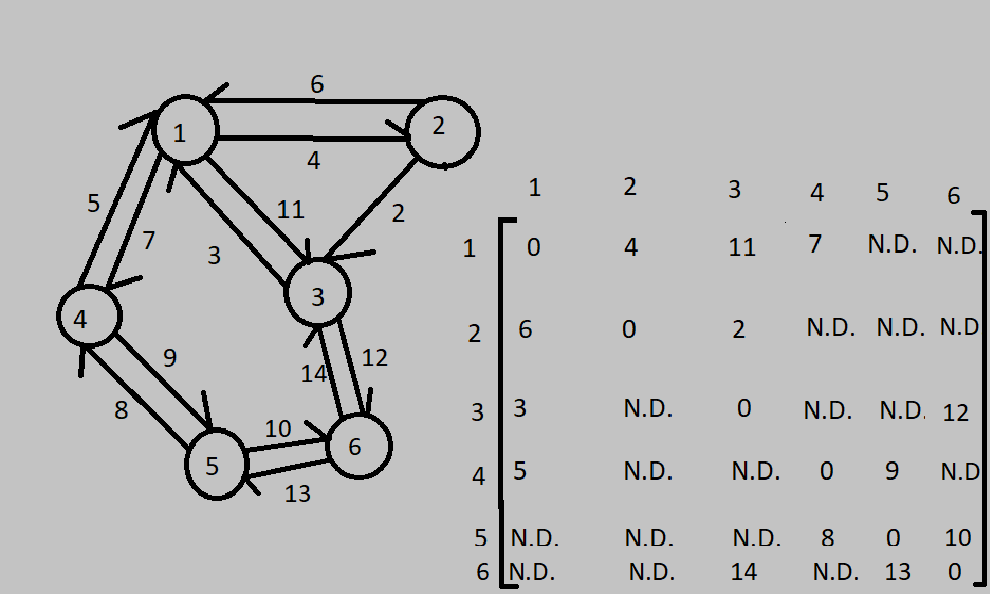
Case 3: When the number of vertices is 5.

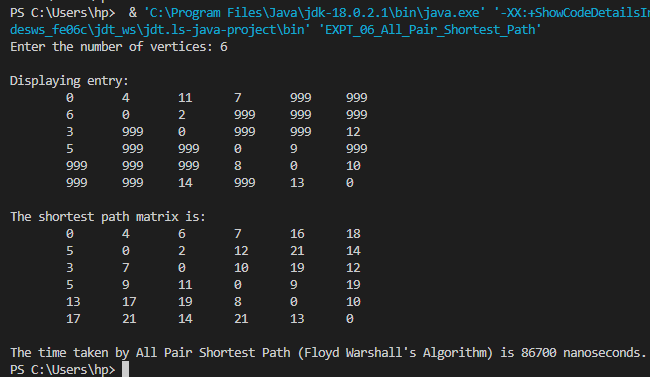




Time taken is 66,400 nanoseconds.

Case 4: When the number of vertices is 6.





Time taken is 86,700 nanoseconds.