CSE- 3004 LAB- Assignment

Academic year: 2020-2021

Semester: WIN

Faculty Name: Dr. D Sumathi Mam

Date: 14 /11/2021

Student name: M.Taran

Reg. no.: 19BCE7346

All pair shortest path problem using Dynamic Programming

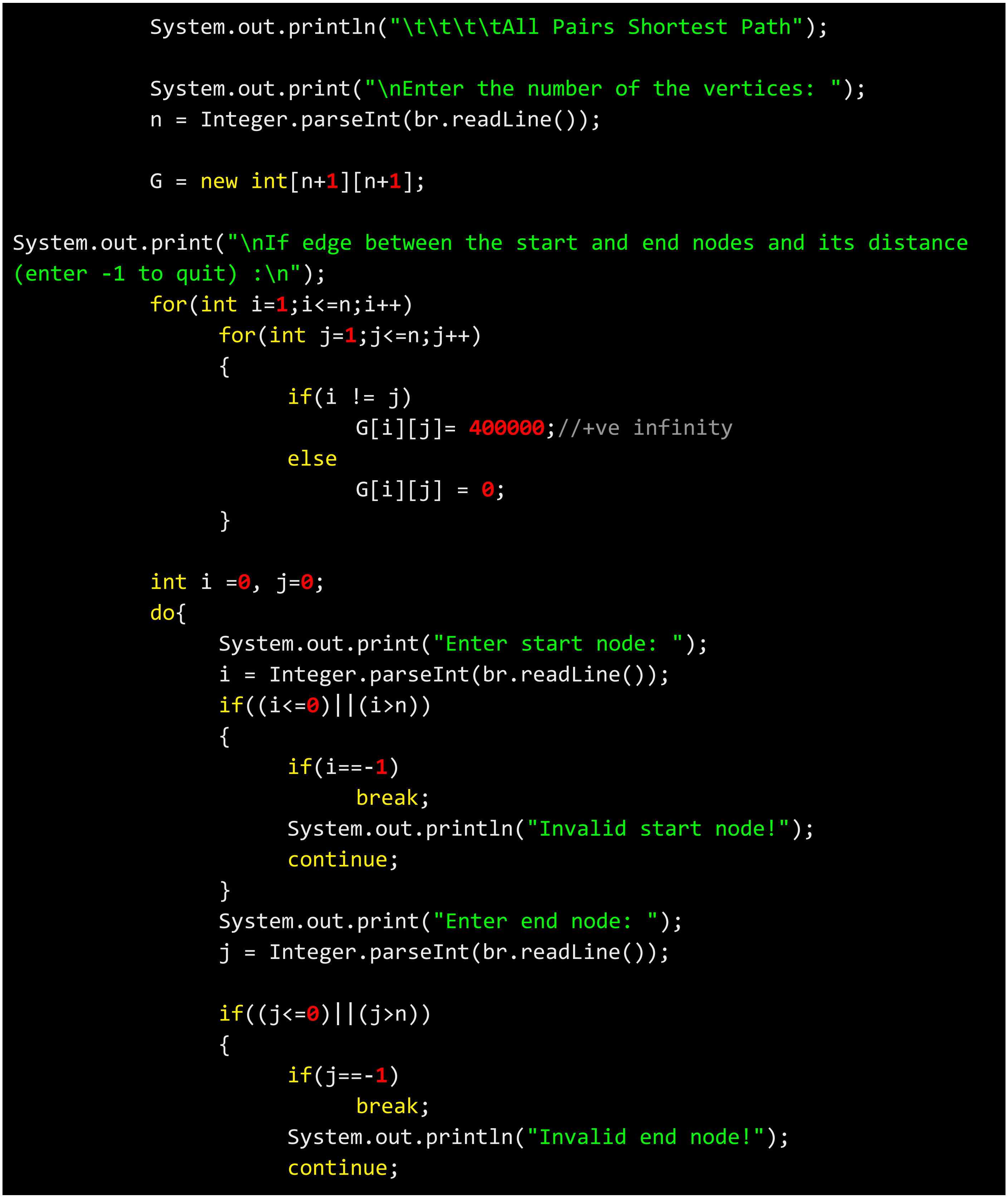
1) implement a dynamic programming algorithm to solve all pairs shortest path problems in java.

The All Pairs Shortest Path (APSP) problem is to compute the shortest path between every pair of points in a directed weighted graph.

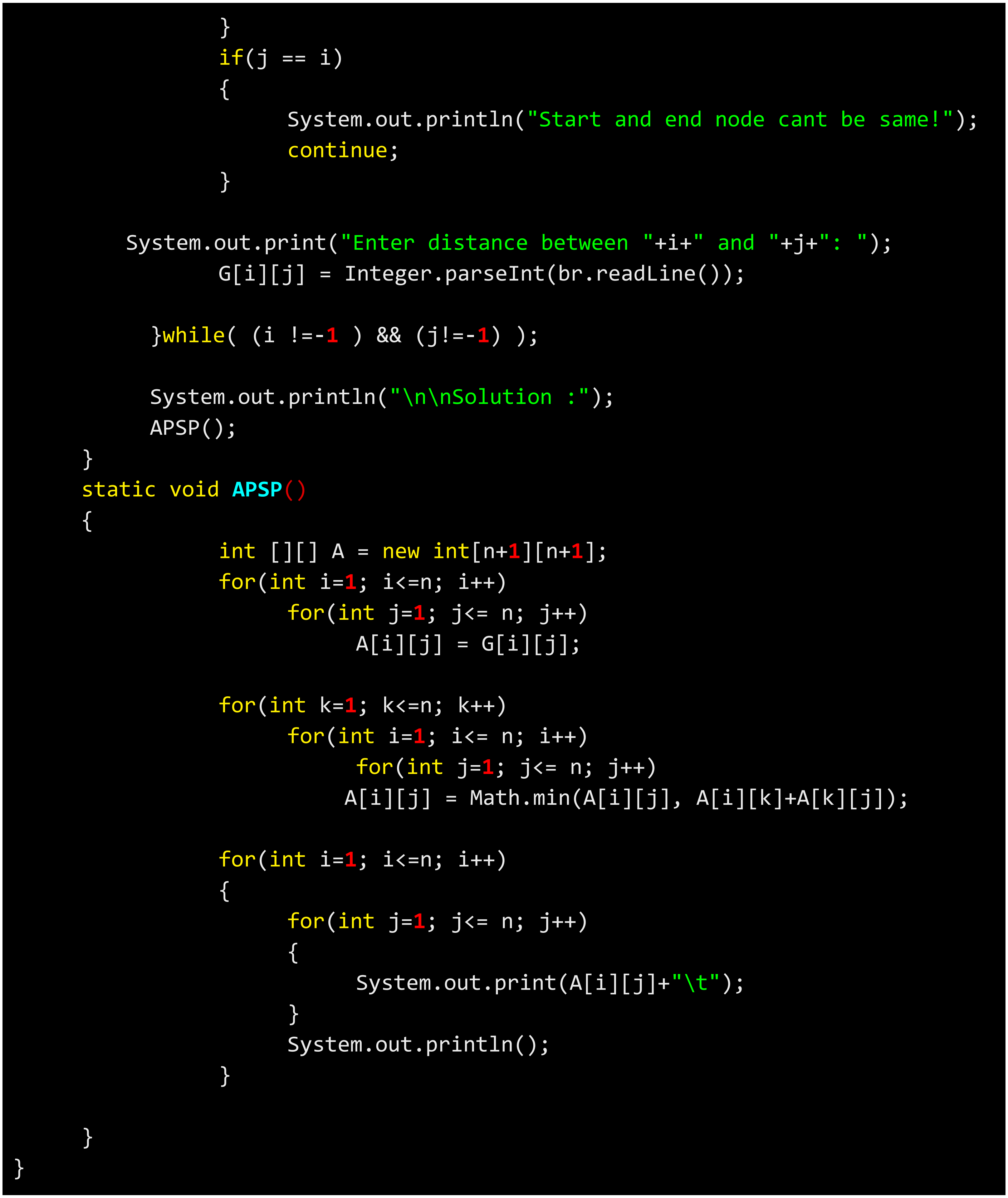
It iteratively revises path lengths between all pairs of vertices (i, j), including where i = j. Initially, the size of the path (i, i) is zero. A path [i, k...i] can only improve upon this if it has a length less than zero, i.e., denotes a negative cycle. Thus, after the algorithm, (i, i) will be negative if there exists a negative-length path from i back to i

Code :

```
import java.io.*;

public class AllPairsShortestPath
{
	public static BufferedReader br =new BufferedReader(new
InputStreamReader(System.in));

	static int [][] G;
	static int n;

	public static void main (String[] args) throws IOException
	{
```

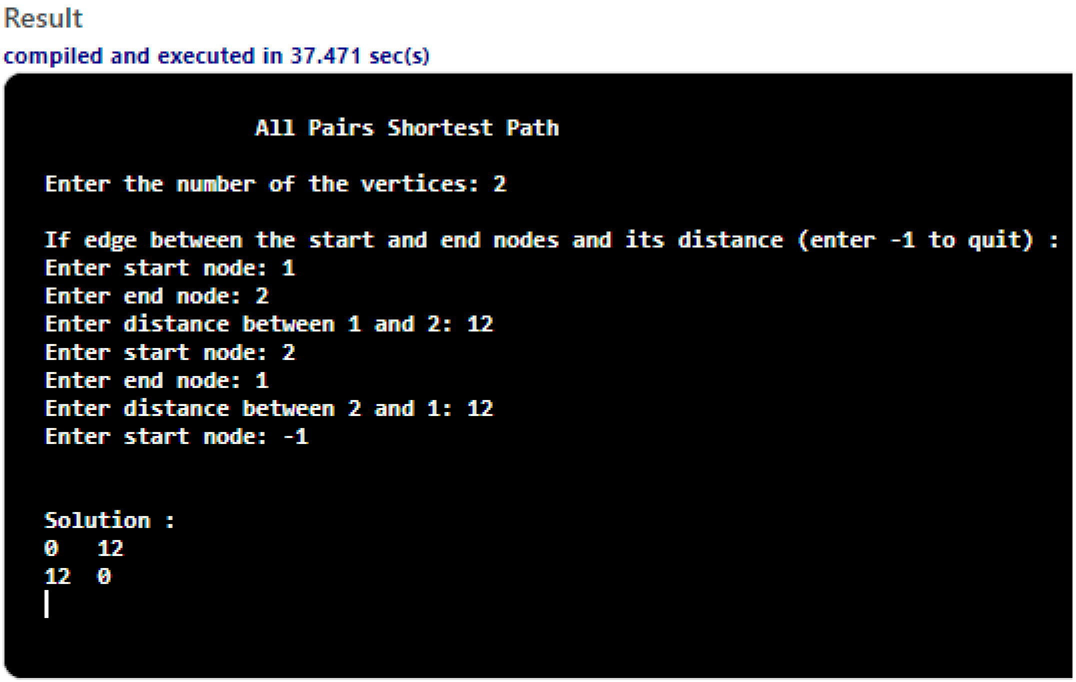


```
        System.out.println("\t\t\t\tAll Pairs Shortest Path");

        System.out.print("\nEnter the number of the vertices: ");
        n = Integer.parseInt(br.readLine());

        G = new int[n+1][n+1];

System.out.print("\nIf edge between the start and end nodes and its distance (enter -1 to quit) :\n");
        for(int i=1;i<=n;i++)
            for(int j=1;j<=n;j++)
            {
                if(i != j)
                    G[i][j]= 400000;//+ve infinity
                else
                    G[i][j] = 0;
            }

        int i =0, j=0;
        do{
            System.out.print("Enter start node: ");
            i = Integer.parseInt(br.readLine());
            if((i<=0)||(i>n))
            {
                if(i==-1)
                    break;
                System.out.println("Invalid start node!");
                continue;
            }
            System.out.print("Enter end node: ");
            j = Integer.parseInt(br.readLine());

            if((j<=0)||(j>n))
            {
                if(j==-1)
                    break;
                System.out.println("Invalid end node!");
                continue;
```



```
                 }
                 if(j == i)
                 {
                      System.out.println("Start and end node cant be same!");
                      continue;
                 }

         System.out.print("Enter distance between "+i+" and "+j+": ");
                 G[i][j] = Integer.parseInt(br.readLine());

           }while( (i !=-1 ) && (j!=-1) );

           System.out.println("\n\nSolution :");
           APSP();
     }
     static void APSP()
     {
                 int [][] A = new int[n+1][n+1];
                 for(int i=1; i<=n; i++)
                      for(int j=1; j<= n; j++)
                           A[i][j] = G[i][j];

                 for(int k=1; k<=n; k++)
                      for(int i=1; i<= n; i++)
                           for(int j=1; j<= n; j++)
                          A[i][j] = Math.min(A[i][j], A[i][k]+A[k][j]);

                 for(int i=1; i<=n; i++)
                 {
                      for(int j=1; j<= n; j++)
                      {
                           System.out.print(A[i][j]+"\t");
                      }
                      System.out.println();
                 }

     }
}
```

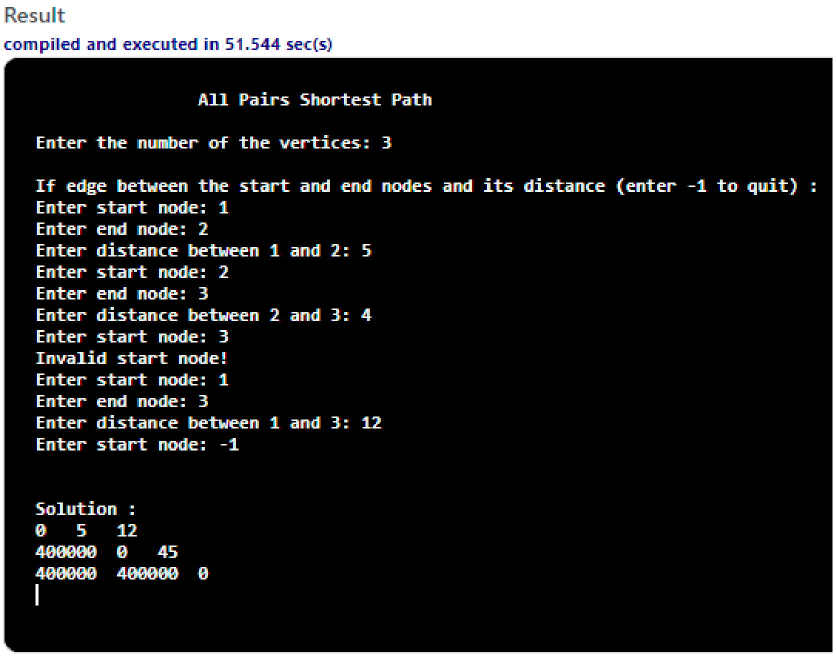
OUTPUT :



Result

compiled and executed in 37.471 sec(s)

```
                All Pairs Shortest Path

Enter the number of the vertices: 2

If edge between the start and end nodes and its distance (enter -1 to quit) :
Enter start node: 1
Enter end node: 2
Enter distance between 1 and 2: 12
Enter start node: 2
Enter end node: 1
Enter distance between 2 and 1: 12
Enter start node: -1


Solution :
0   12
12  0
```



Result

compiled and executed in 51.544 sec(s)

```
                All Pairs Shortest Path

Enter the number of the vertices: 3

If edge between the start and end nodes and its distance (enter -1 to quit) :
Enter start node: 1
Enter end node: 2
Enter distance between 1 and 2: 5
Enter start node: 2
Enter end node: 3
Enter distance between 2 and 3: 4
Enter start node: 3
Invalid start node!
Enter start node: 1
Enter end node: 3
Enter distance between 1 and 3: 12
Enter start node: -1


Solution :
0   5   12
400000  0   45
400000  400000  0
```