**Batch: A1 Roll No.: 1911004**

**Experiment No. 7**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Title: Implementation of All Pair Shortest Path using Dynamic Programming** |

**Objective** To learn the All Pair Shortest Path using Knapsack Problemalgorithm

**CO to be achieved:**

|  |  |
| --- | --- |
| Sr. No | Objective |
| CO 1 | Analyze the asymptotic running time and space complexity of algorithms. |
| CO 2 | Describe various algorithm design strategies to solve different problems & analyze  Complexity. |
| CO 3 | Develop string matching techniques |
| CO 4 | Describe the classes P, NP, and NP-Complete |

**Books/ Journals/ Websites referred:**

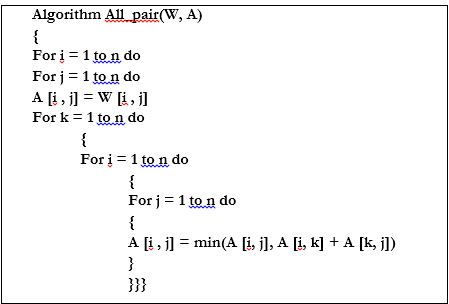
1. **Ellis horowitz, Sarataj Sahni, S.Rajsekaran,” Fundamentals of computer algorithm”, University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein,” Introduction to algortihtms”,2nd Edition ,MIT press/McGraw Hill,2001**
3. **http://users.cecs.anu.edu.au/~Alistair.Rendell/Teaching/apac\_comp3600/module4/all\_pairs\_shortest\_paths.xhtml**
4. **https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/**
5. **http://www.cs.bilkent.edu.tr/~atat/502/AllPairsSP.ppt**

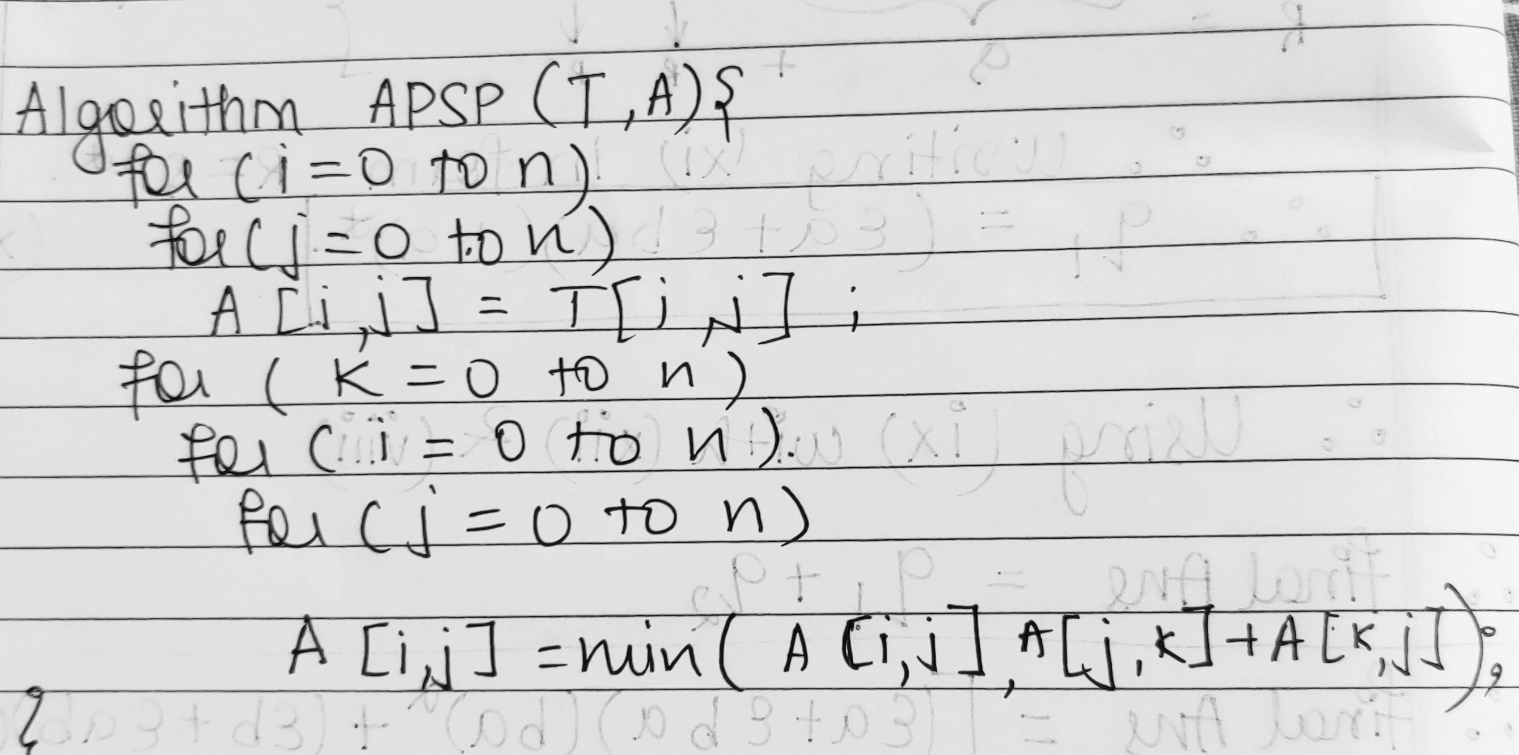
**Theory:**

It aims to figure out the shortest path from each vertex v to every other u.

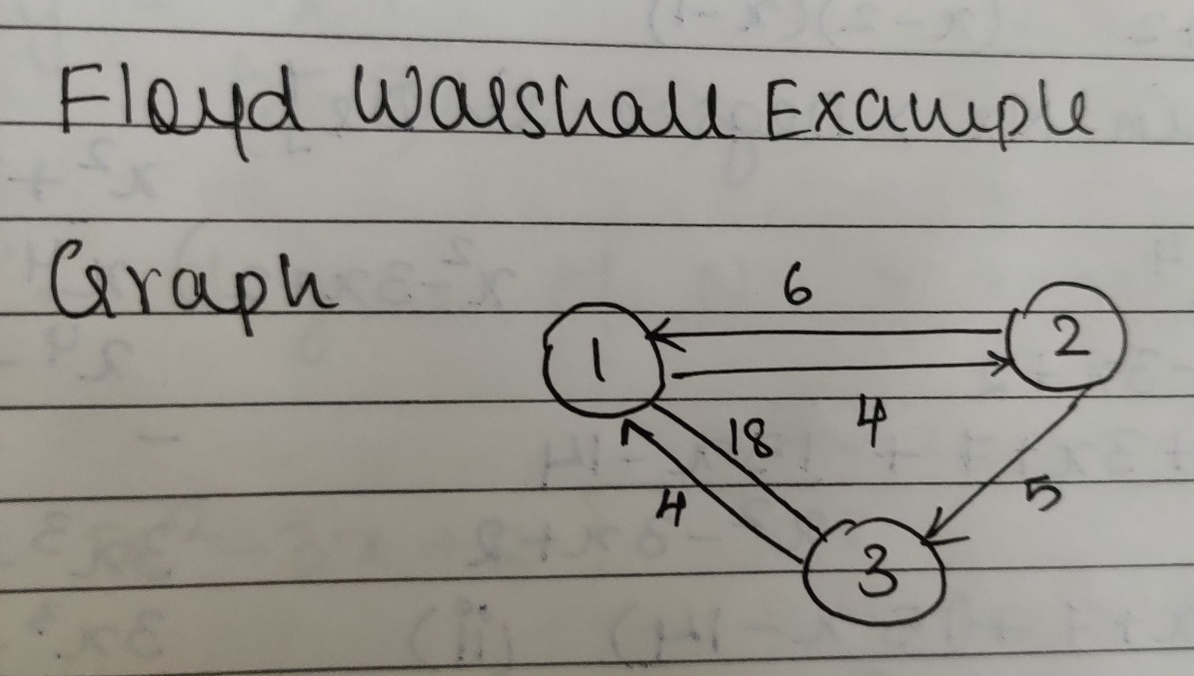
1. In all pair shortest path, when a weighted graph is represented by its weight matrix W then objective is to find the distance between every pair of nodes.
2. Apply dynamic programming to solve the all pairs shortest path.
3. In all pair shortest path algorithm, we first decomposed the given problem into sub problems.
4. In this principle of optimally is used for solving the problem.
5. It means any sub path of shortest path is a shortest path between the end nodes.

**Algorithm:**

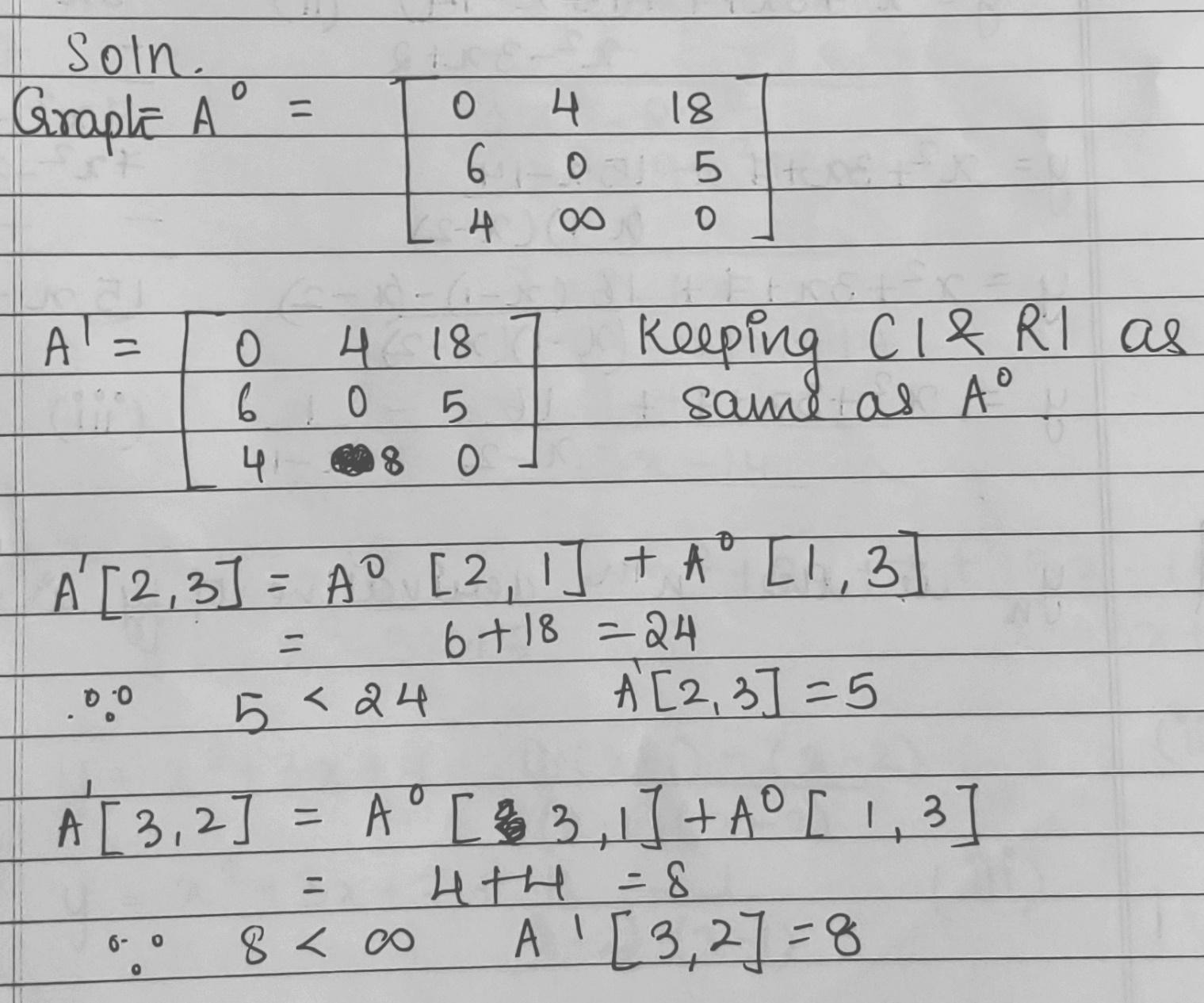


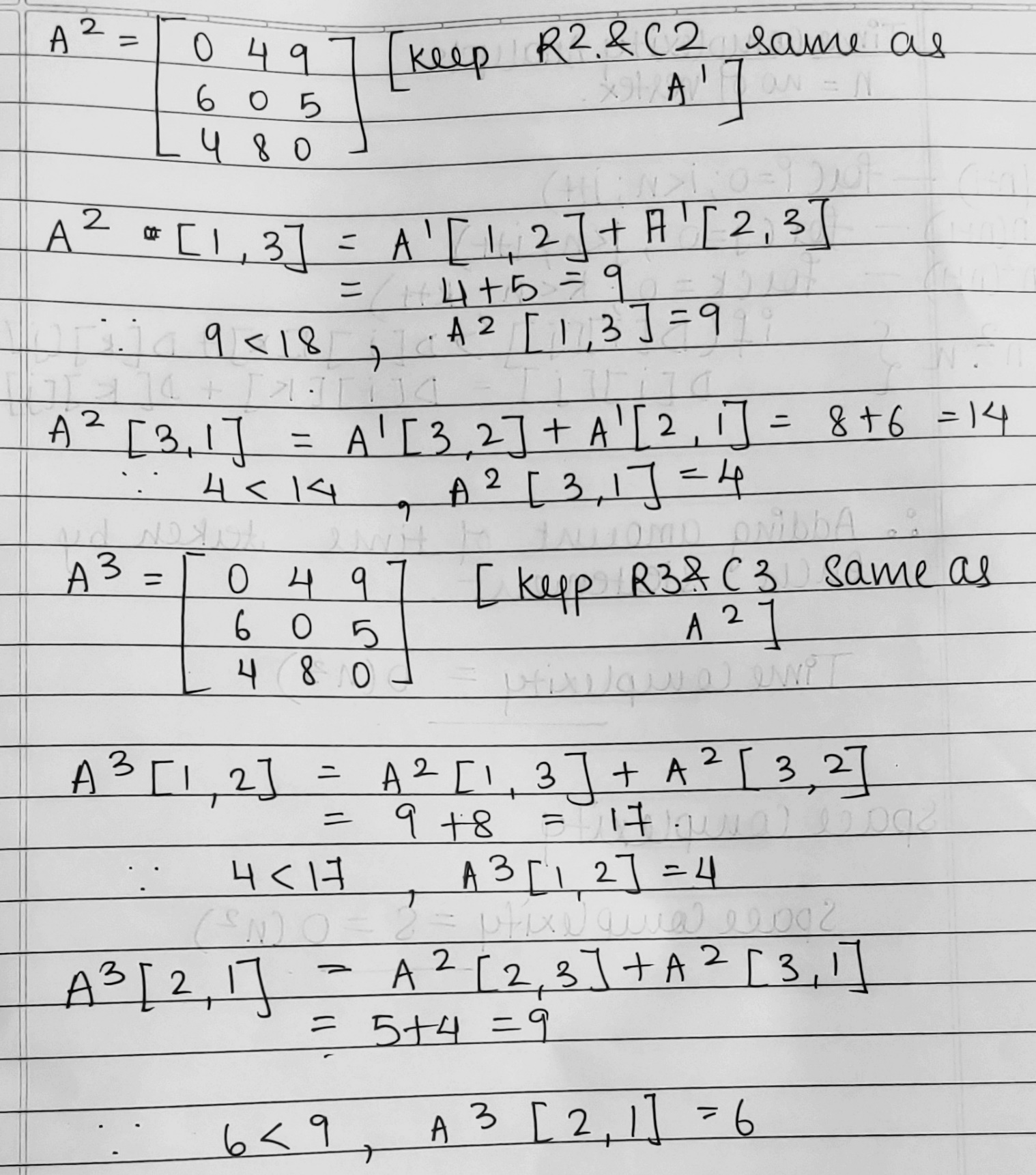


**Example :**



**Solution for the example**





CODE:

import java.util.\*;

class Main {

  public static void print\_path(int path[][],int begin,int end){

    Stack<Integer> s = new Stack<Integer>();

    s.push(end);

    while(path[begin][end]!=begin){

      s.push(path[begin][end]);

      end = path[begin][end];

    }

    System.out.print(begin+" ");

    while(!s.isEmpty()){

      if(s.peek()==s.lastElement()){

          System.out.print(s.pop()+" ");

      }

      else{

        System.out.print(s.pop()+" ");

      }

    }

  }

  static void FloydWarshall(int cost[][],int path[][],int n)

  {

    for(int i=0;i<n;i++){

      for(int j=0;j<n;j++){

        for(int k=0;k<n;k++){

          if(cost[i][k]!=9999 && cost[k][j]!=9999){

            if(cost[i][j]>cost[i][k]+cost[k][j]){

              cost[i][j] = cost[i][k] + cost[k][j];

              path[i][j] = path[k][j];

            }

          }

        }

      }

      System.out.println("A^"+(i+1));

      print(cost);

    }

  }

    static void print(int a[][])

    {

    int i,j;

    for(i=0;i<a.length;i++)

    {

      for(j=0;j<a.length;j++)

      System.out.print(String.format("%-6d",a[i][j]));

      System.out.println("");

    }

    System.out.println("");

    }

  public static void main(String[] args) {

    Scanner sc = new Scanner(System.in);

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

    int n = sc.nextInt();

    System.out.println("Enter the number of edges");

    int e = sc.nextInt();

    int cost[][] = new int [n][n];

    int path[][] = new int[n][n];

    for(int i=0;i<n;i++){

      for(int j=0;j<n;j++){

        if(i==j){

          cost[i][j] = 0;

        }

        else{

          cost[i][j] = 9999;

        }

      }

    }

    for(int i=0;i<n;i++){

      for(int j=0;j<n;j++){

        path[i][j] = 9999;

      }

    }

    System.out.println("\nEnter the start,end vertices and the cost");

    for(int i=0;i<e;i++){

      int s = sc.nextInt();

      int v = sc.nextInt();

      int cost\_temp = sc.nextInt();

      cost[s][v] = cost\_temp;

      path[s][v] = s;

    }

    System.out.println("\nA^0:");

    print(cost);

    FloydWarshall(cost,path,n);

    System.out.println(String.format("%-10s %-15s %-20s ","Source","Destination","MinCost")+"Path");

    for(int i=0;i<n;i++){

      for(int j=0;j<n;j++){

        System.out.print(String.format("%-10s %-15s %-20s ",i,j,cost[i][j]));

        if(i!=j){

          print\_path(path,i,j);

        }

        else{

          System.out.print(i);

        }

        System.out.println("");

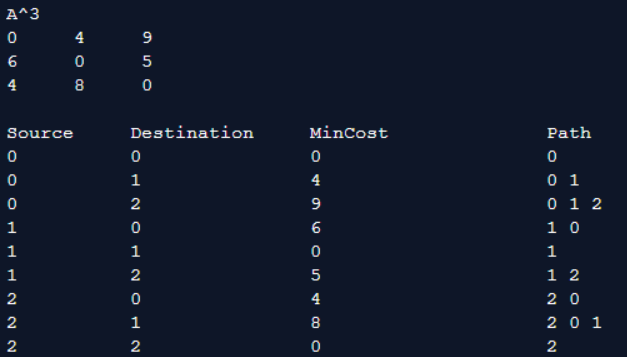
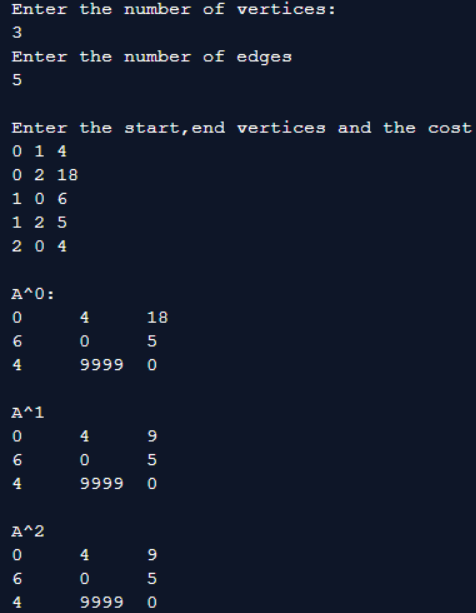
      }

    }

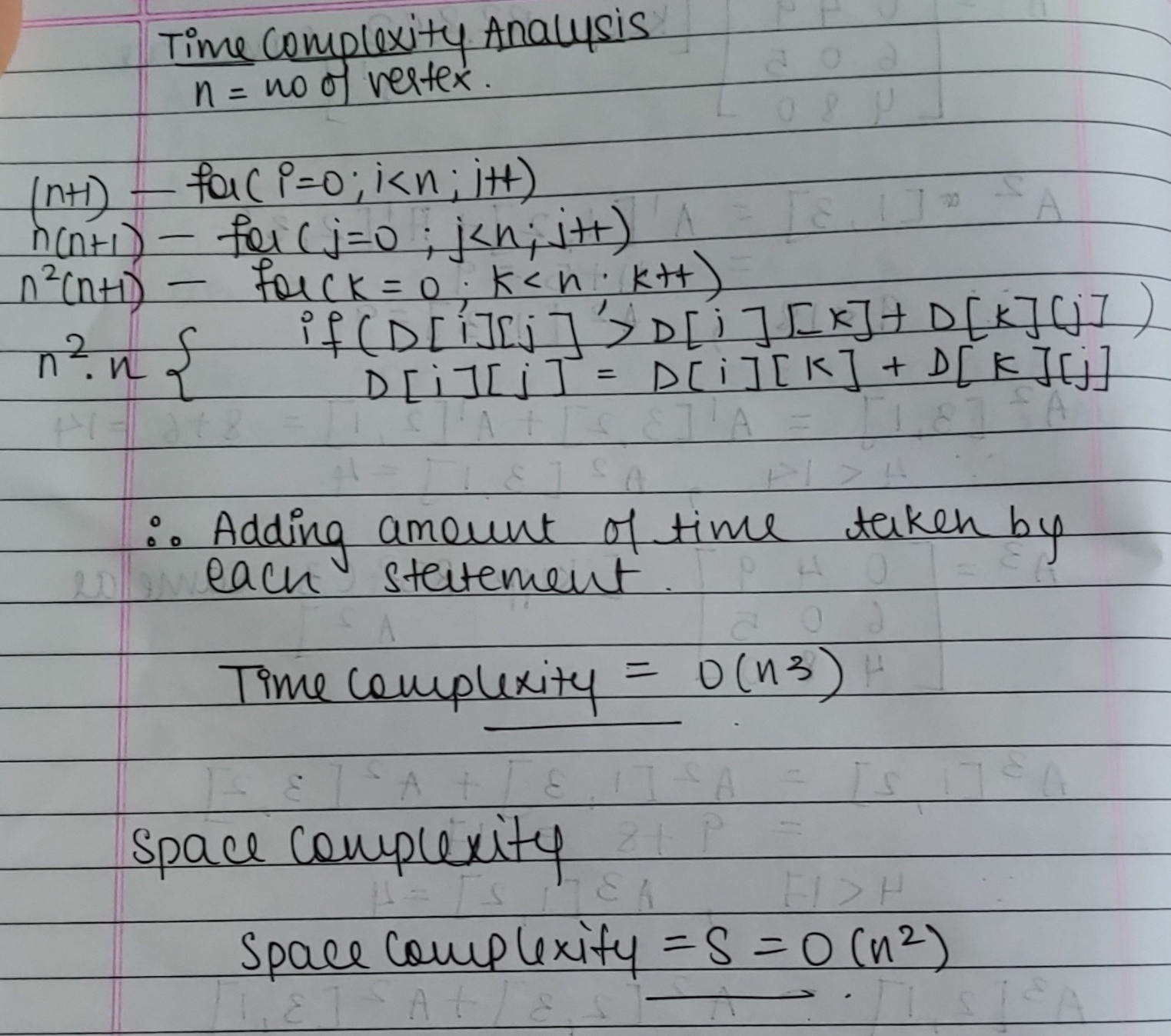
  }

}

Output:



**Analysis of algorithm:**



**Conclusion:**

We successfully understood & implemented All Pair Shortest Path Algorithm using Dynamic Programming ; thus obtained all paths by implementing it in java.