K. J. Somaiya College of Engineering

(A Constituent College of Somaiya Vidyavihar University)

Batch: A1 Roll No.: 16010120015

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 Floyd-Warshall algorithm

CO to be achieved:

CO 2 Describe various algorithm design strategies to solve different problems and analyze Complexity.

Books/ Journals/ Websites referred:

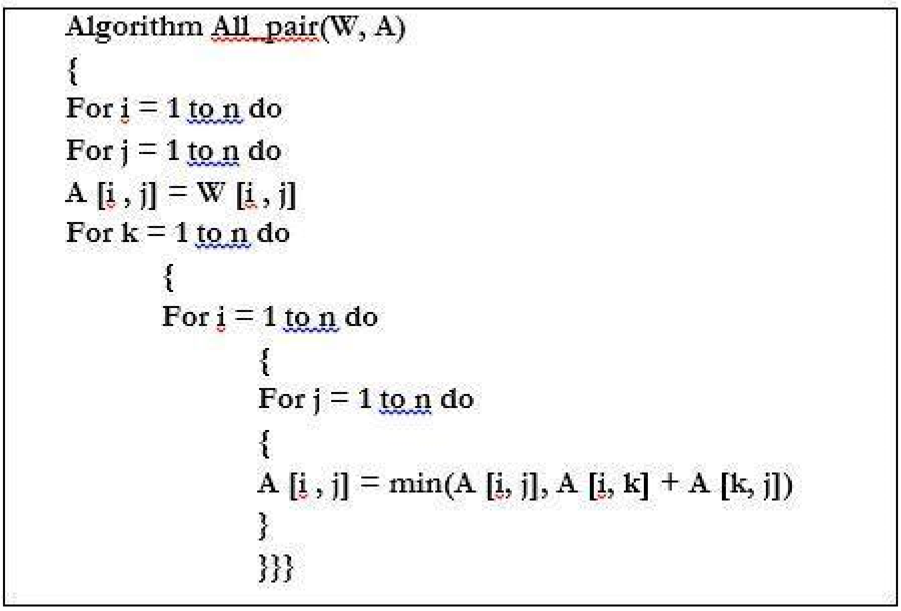
1. **Ellis horowitz, Saratai Sahni, S.Rajsekaran,” Fundamentals of computer algorithm”, University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein,” Introduction to algortihtms”,2nd Edition ,MIT press/McGraw Hill,2001**
3. **http://users.cecs.anu.edu.au/~Alistair.Rendell/Teaching/apac_comp3600/module4/all_pairs_shortest_paths.xhtml**
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.

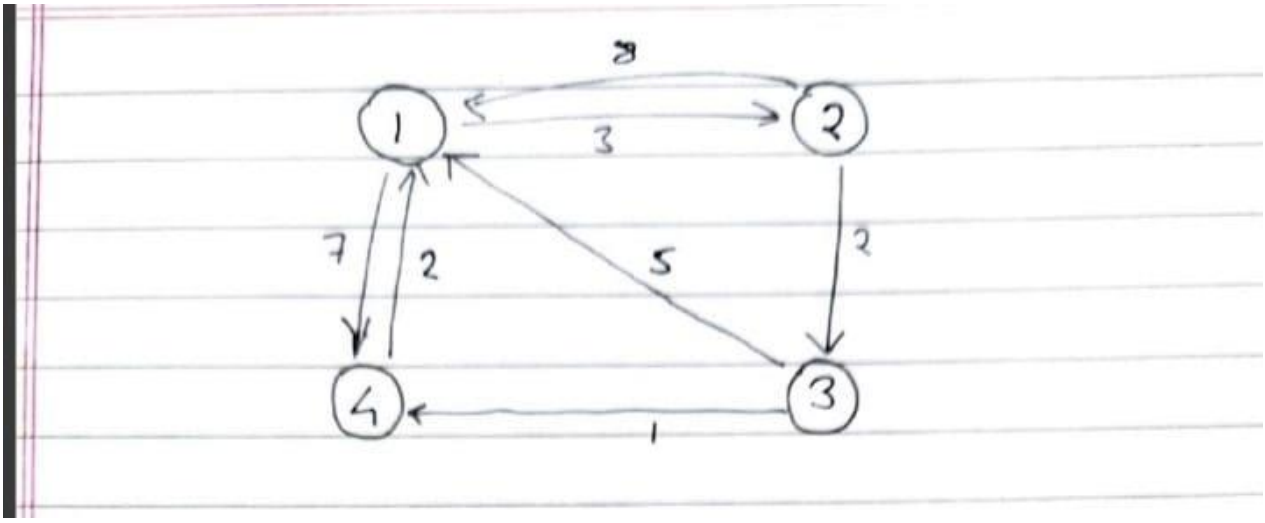
1. In all pair shortest path, when a weighted graph is represented by its weight matrix W then objective is to find the distance between every pair of nodes.
2. Apply dynamic programming to solve the all pairs shortest path.
3. In all pair shortest path algorithm, we first decomposed the given problem 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:

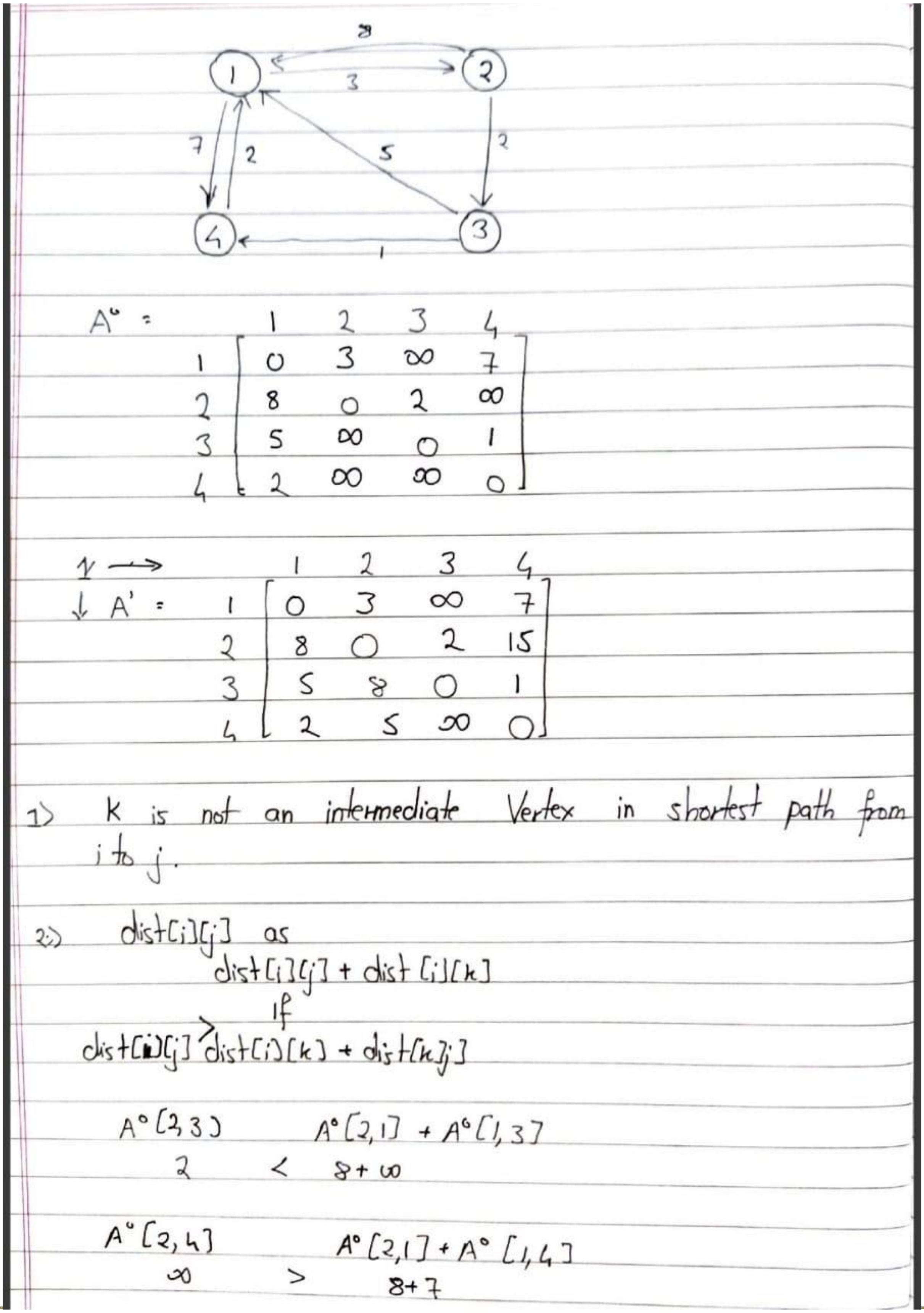


```
Algorithm All_pair(W, A)
{
For i = 1 to n do
For j = 1 to n do
A [i , j] = W [i , j]
For k = 1 to n do
          {
          For i = 1 to n do
                    {
                    For j = 1 to n do
                    {
                    A [i , j] = min(A [i, j], A [i, k] + A [k, j])
                    }
                    }}}
```

Example :



Solution:


8
1
3
2
7
2
5
2
4
3
1

$$A^0 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & \infty & 7 \\ 2 & 8 & 0 & 2 & \infty \\ 3 & 5 & \infty & 0 & 1 \\ 4 & 2 & \infty & \infty & 0 \end{array}$$

$1 \longrightarrow$

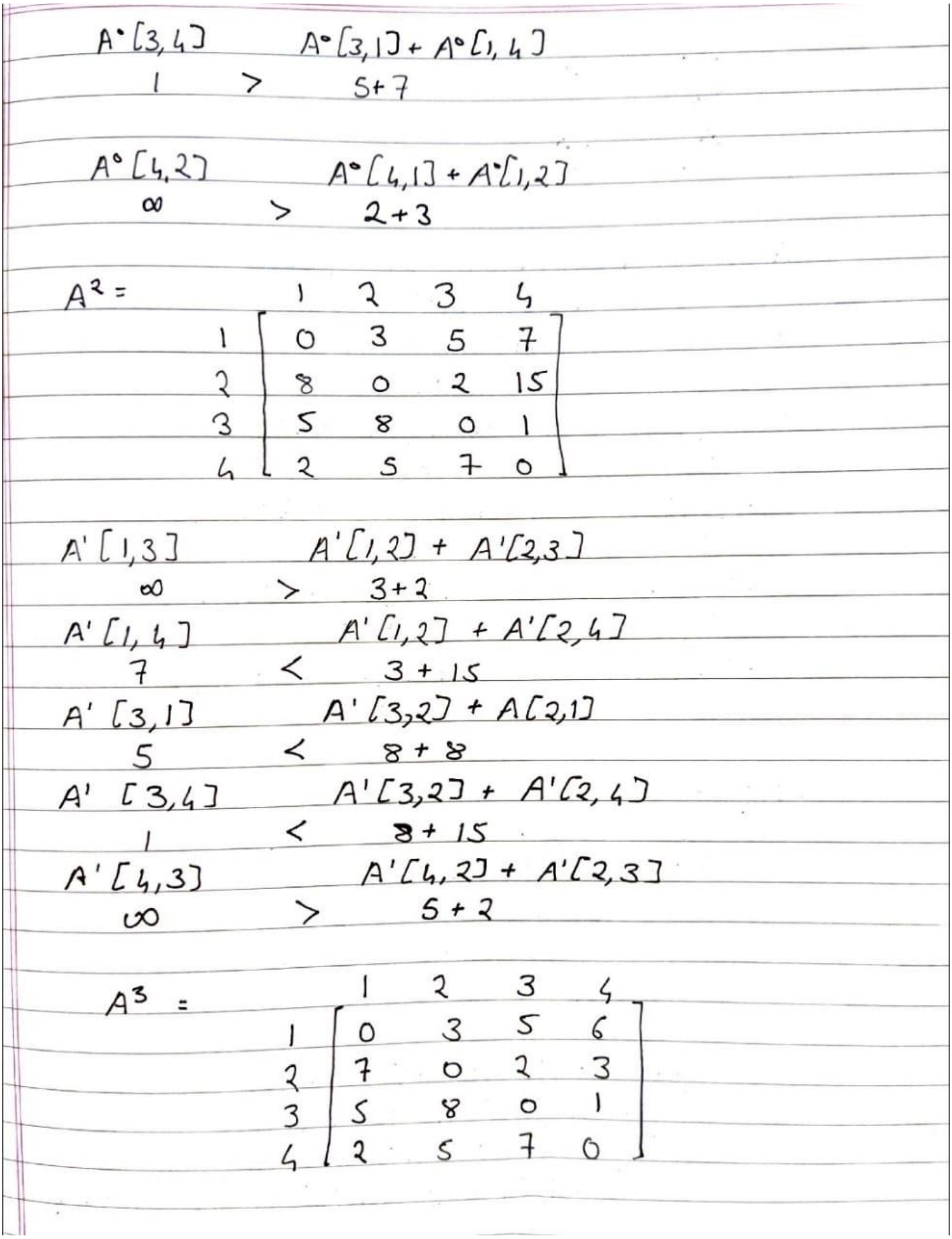
$$\downarrow A' = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & \infty & 7 \\ 2 & 8 & 0 & 2 & 15 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & \infty & 0 \end{array}$$

1) K is not an intermediate Vertex in shortest path from i to j.

2:) dist[i][j] as
dist[i][j] + dist [i][k]
if
dist[i][j] > dist[i][k] + dist[k][j]

$A^\circ[2,3]$ $A^\circ[2,1] + A^\circ[1,3]$
$2 < 8+\infty$

$A^\circ[2,4]$ $A^\circ[2,1] + A^\circ[1,4]$
$\infty > 8+7$



$A^{\circ}[3,4]$ $\quad$ $A^{\circ}[3,1] + A^{\circ}[1,4]$

$1 > 5+7$

$A^{\circ}[4,2]$ $\quad$ $A^{\circ}[4,1] + A^{\circ}[1,2]$

$\infty > 2+3$

$$A^2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 7 \\ 2 & 8 & 0 & 2 & 15 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

$A'[1,3]$ $\quad$ $A'[1,2] + A'[2,3]$

$\infty > 3+2$

$A'[1,4]$ $\quad$ $A'[1,2] + A'[2,4]$

$7 < 3 + 15$

$A'[3,1]$ $\quad$ $A'[3,2] + A[2,1]$

$5 < 8 + 8$

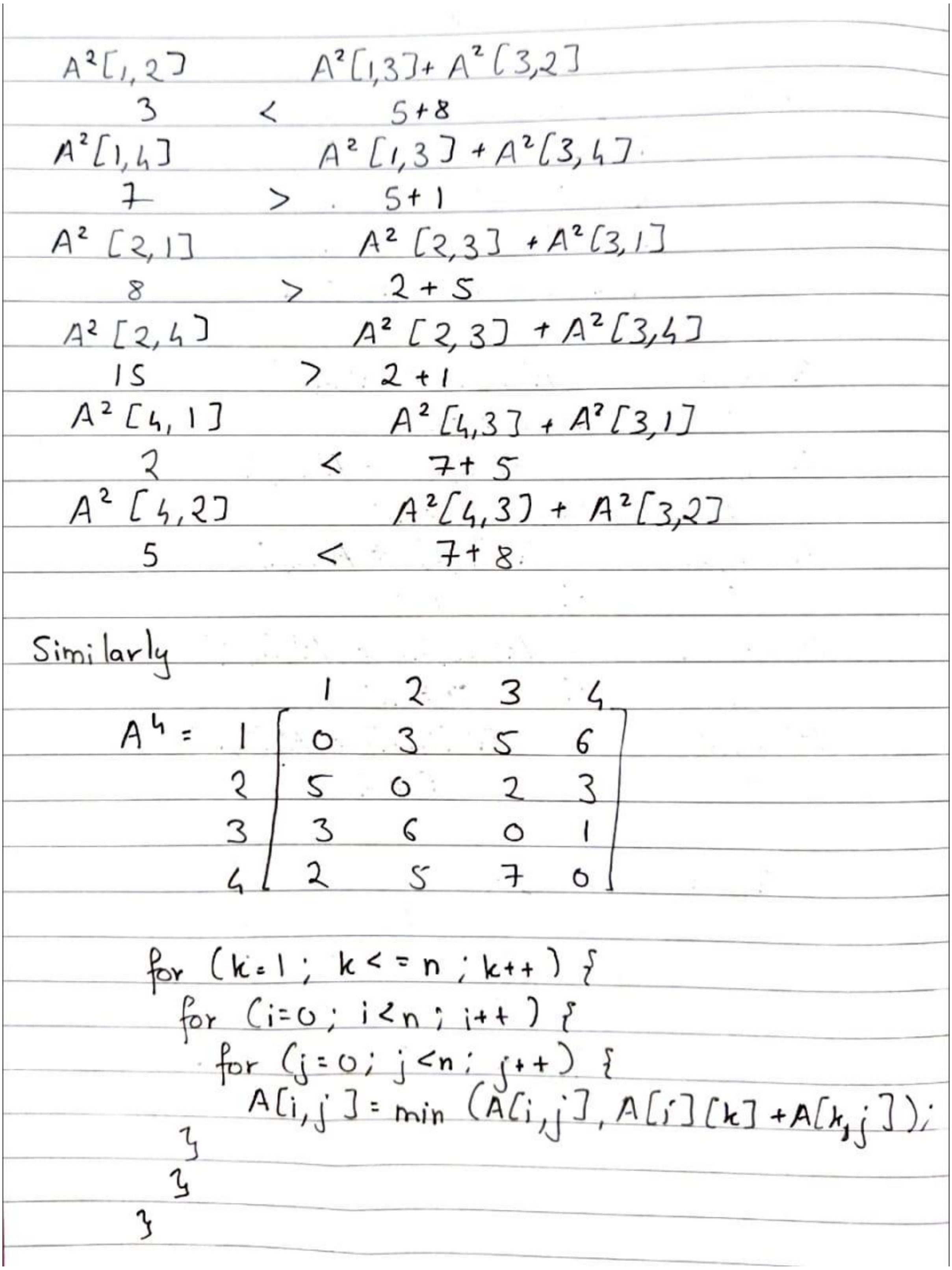
$A'[3,4]$ $\quad$ $A'[3,2] + A'[2,4]$

$1 < 8 + 15$

$A'[4,3]$ $\quad$ $A'[4,2] + A'[2,3]$

$\infty > 5 + 2$

$$A^3 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 6 \\ 2 & 7 & 0 & 2 & 3 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$



$A^2[1,2] \qquad A^2[1,3] + A^2[3,2]$

$3 < 5+8$

$A^2[1,4] \qquad A^2[1,3] + A^2[3,4]$

$7 > 5+1$

$A^2[2,1] \qquad A^2[2,3] + A^2[3,1]$

$8 > 2+5$

$A^2[2,4] \qquad A^2[2,3] + A^2[3,4]$

$15 > 2+1$

$A^2[4,1] \qquad A^2[4,3] + A^2[3,1]$

$2 < 7+5$

$A^2[4,2] \qquad A^2[4,3] + A^2[3,2]$

$5 < 7+8$

Similarly

$$A^4 = \begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ 1 & 0 & 3 & 5 & 6 \\ 2 & 5 & 0 & 2 & 3 \\ 3 & 3 & 6 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

```
for (k=1; k<=n; k++) {
    for (i=0; i<n; i++) {
        for (j=0; j<n; j++) {
            A[i,j] = min (A[i,j], A[i][k] +A[k,j]);
        }
    }
}
```

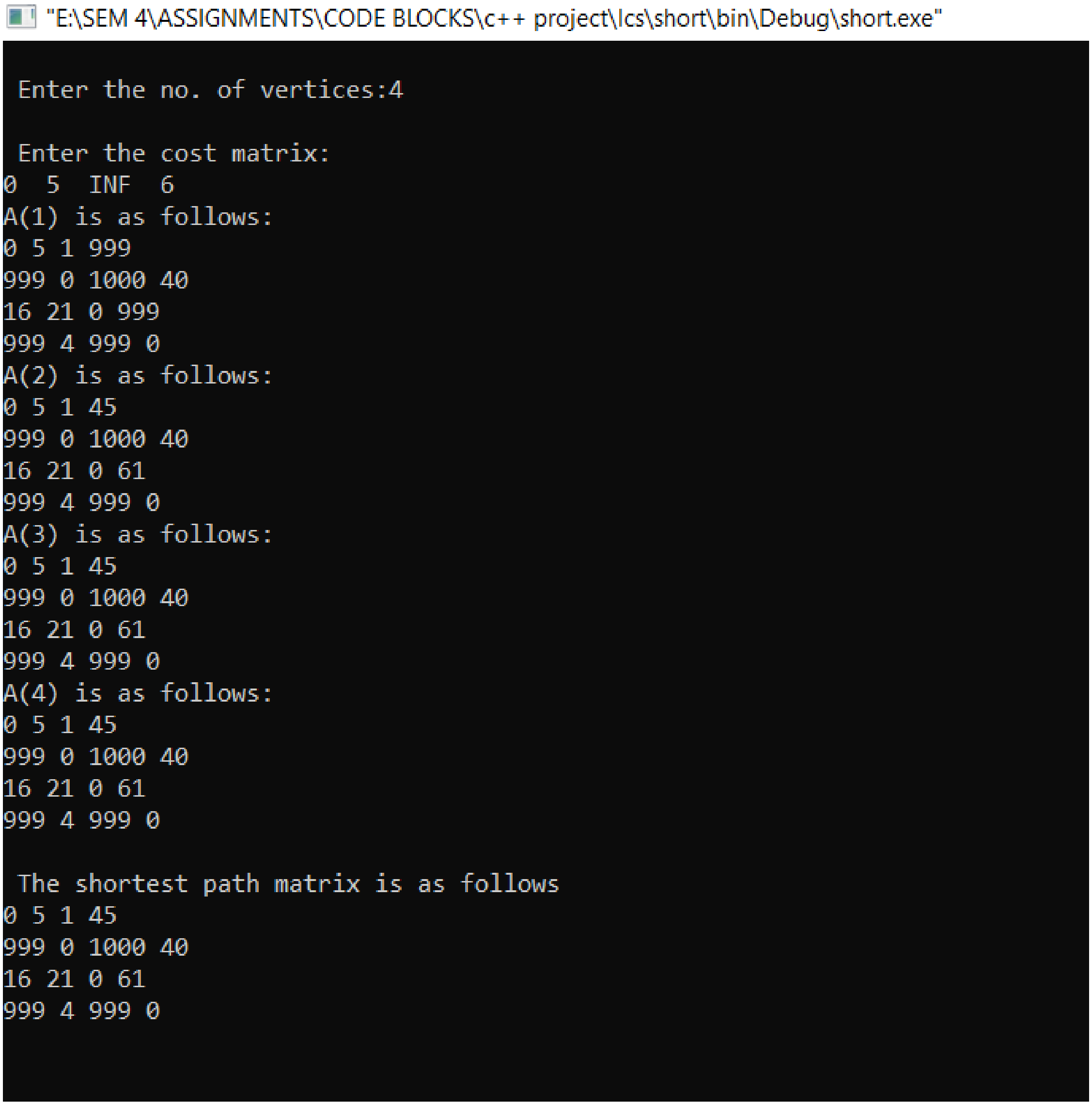
Code :

```
#include <stdio.h>
#include <conio.h>
#define MAX 25
#define INF 999
int min(int a, int b)
{
  if (a < b)
    return a;
  else
    return b;
}
void main()
{
  int n, cost[MAX][MAX], a[MAX][MAX], i, j, k;

  printf("\n Enter the no. of vertices:");
  scanf("%d", &n);
  printf("\n Enter the cost matrix:\n");
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
    {
      scanf("%d", &cost[i][j]);
      if (i == j)
        a[i][j] = 0;
      else if (cost[i][j] == 0)
        a[i][j] = INF;
      else
        a[i][j] = cost[i][j];
    }
  for (k = 0; k < n; k++)
  {
    printf("A(%d) is as follows:\n", k + 1);
    for (i = 0; i < n; i++)
    {
      for (j = 0; j < n; j++)
      {
        a[i][j] = min(a[i][j], a[i][k] + a[k][j]);
        printf("%d ", a[i][j]);
      }
      printf("\n");
    }
  }
  printf("\n The shortest path matrix is as follows\n");
  for (i = 0; i < n; i++)
  {
    for (j = 0; j < n; j++)
      printf("%d ", a[i][j]);
    printf("\n");
  }

  getch();
}
```

#DEFINE INF AS 999



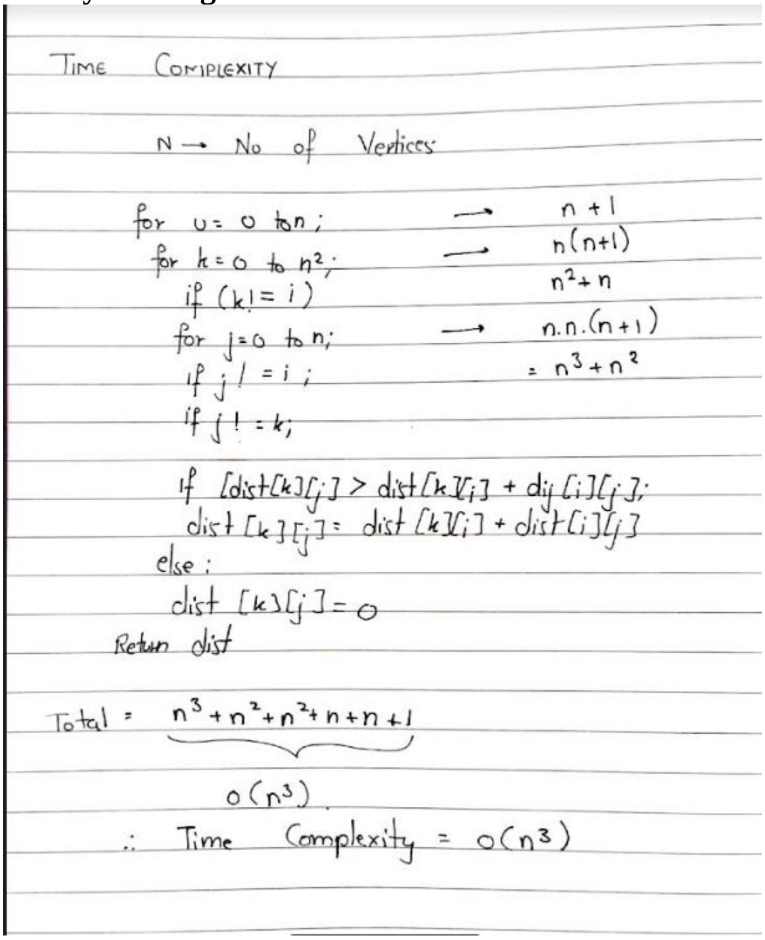
"E:\SEM 4\ASSIGNMENTS\CODE BLOCKS\c++ project\lcs\short\bin\Debug\short.exe"

```
 Enter the no. of vertices:4

 Enter the cost matrix:
0  5  INF  6
A(1) is as follows:
0 5 1 999
999 0 1000 40
16 21 0 999
999 4 999 0
A(2) is as follows:
0 5 1 45
999 0 1000 40
16 21 0 61
999 4 999 0
A(3) is as follows:
0 5 1 45
999 0 1000 40
16 21 0 61
999 4 999 0
A(4) is as follows:
0 5 1 45
999 0 1000 40
16 21 0 61
999 4 999 0

 The shortest path matrix is as follows
0 5 1 45
999 0 1000 40
16 21 0 61
999 4 999 0
```

Analysis of algorithm:



TIME COMPLEXITY

N → No of Vertices

```
for u = 0 to n;                    →   n + 1
  for k = 0 to n²;                 →   n(n+1)
    if (k != i)                        n² + n
    for j = 0 to n;                →   n.n.(n+1)
     if j ! = i ;                      = n³ + n²
     if j ! = k;

     if [dist[k][j] > dist[k][i] + dij [i][j];
      dist [k][j] = dist [k][i] + dist[i][j]
  else :
    dist [k][j] = 0
Return dist
```

$$\text{Total} = \underbrace{n^3 + n^2 + n^2 + n + n + 1}_{}$$

$o(n^3)$.

∴ Time Complexity = $o(n^3)$

Conclusion:

In this Experiment we have determined the shortest graph distances between every pair of vertices in a given graph.

- We have seen a method to compute all pairs shortest paths in a weighted undirected graph with possibly negative weights.
- This method can also be used to detect negative cycles.
- This method is efficient for dense graphs.