All_pair_Shortest_path

Write Java program to Implement All-Pairs Shortest Paths problem using Floyd's algorithm

Input Format

5 0 5 999 2 999 999 0 2 999 999 3 999 0 999 7 999 999 4 0 1 1 3 999 999 0

Constraints

no constraints

Output Format

all pair shortest paths matrix. 0 5 6 2 3 5 0 2 7 8 3 8 0 5 6 2 4 4 0 1 1 3 5 3 0

Sample Input 0

```
5
0 5 999 2 999
999 0 2 999 999
3 999 0 999 7
999 999 4 0 1
1 3 999 999 0
```

Sample Output 0

```
all pair shortest paths matrix.
0 5 6 2 3
5 0 2 7 8
3 8 0 5 6
2 4 4 0 1
1 3 5 3 0
```


Java 7

```
import java.util.*;
public class Floyds {
 static int n,i,j,k;
 public void floyd(int n , int[][] cost)
 {
   for(k=1;k<=n;k++)
   {
       for(i=1;i<=n;i++)
       {
           for(j=1;j<=n;j++)
           {
               cost[i][j]=min(cost[i][j],cost[i][k]+cost[k][j]);
           }
       }
  }
  System.out.println("all pair shortest paths matrix.");
  for(i=1;i<=n;i++)
  {
      for(j=1;j<=n;j++)
```

```
      {
         System.out.print(cost[i][j]+" ");
      }
     System.out.println();
  }
 }
 public int min(int i,int j)
 {
    if(i<j)
        return i;
    else
        return j;
 }
 public static void main(String[] args)
 {
    Scanner sc=new Scanner(System.in);
    //System.out.println("Enter the no of vertices: ");
    n=sc.nextInt();
    int cost[][]=new int[n+1][n+1];
    //System.out.println("Enter the cost matrix:");
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            cost[i][j]=sc.nextInt();
    Floyds f = new Floyds();
          f.floyd(n,cost);
  }
}
```

Line: 1 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
5
0 5 999 2 999
999 0 2 999 999
3 999 0 999 7
999 999 4 0 1
1 3 999 999 0
```

Your Output (stdout)

```
all pair shortest paths matrix.
0 5 6 2 3
5 0 2 7 8
3 8 0 5 6
2 4 4 0 1
1 3 5 3 0
```

Expected Output

```
all pair shortest paths matrix.
0 5 6 2 3
5 0 2 7 8
3 8 0 5 6
2 4 4 0 1
1 3 5 3 0
```