#### **Теоретическая часть https://e-maxx.ru/algo/export\_dijkstra**

#### **Задача**

Дан взвешенный ориентированный граф из N вершин. Требуется найти в нем величину кратчайшего пути между каждой парой вершин.

**Входные данные:**

В первой строке записано натуральное число (1<=N<=100). В каждой из следующих N строк записано по N чисел — матрица весов дуг графа. Все веса представляют собой целые неотрицательные числа, не превосходящие 1000. Если в матрице в i -й строке j-м столбце стоит 0, то это означает, что дуги из вершины в вершину нет.

**Выходные данные:**

В выходной файл надо вывести матрицу кратчайших путей между каждой парой вершин графа (т.е. матрицу у которой в i-й строке j-м столбце столбце стоит длина кратчайшего пути из вершины в вершину или 0 — если пути из в не существует).

import java.io.PrintWriter;  
import java.util.Scanner;  
  
 public class Solution {  
 *// В качестве условной бесконечности   
 // выберем достаточно большое число 10^9* private static final int *INF* = 1000 \* 1000 \* 1000;  
  
 public static void main(String[] args) {  
 Solution solution = new Solution();  
  
 *// Вызываем процедуру решения задачи* solution.solve();  
 }  
  
 private void solve() {  
 *// Для считывания воспользуемся классом Scanner  
 // Для вывода - классом PrintWriter* Scanner scanner = new Scanner(System.*in*);  
 PrintWriter printWriter = new PrintWriter(System.*out*);  
  
 *// Считываем число вершин графа* int vertexCount = scanner.nextInt();  
  
 *// Граф будем хранить в матрице смежности* int[][] g = new int[vertexCount][vertexCount];  
  
 for (int i = 0; i < vertexCount; i++) {  
 for (int j = 0; j < vertexCount; j++) {  
 *// Считываем вес ребра между вершинами  
 // i и j соответственно* g[i][j] = scanner.nextInt();  
  
 *// По условию если g[i][j] = 0, то это  
 // означает, что дуги из i в j нет;  
 // в этом случае расстояние между этими  
 // вершинами бесконечно велико* if (g[i][j] == 0) {  
 g[i][j] = *INF*;  
 }  
 }  
 }  
  
 *// Согласно алгоритму будем обновлять  
 // ответ для каждой пары вершин i и j,  
 // перебирая промежуточную вершину k* for (int k = 0; k < vertexCount; k++) {  
 for (int i = 0; i < vertexCount; i++) {  
 for (int j = 0; j < vertexCount; j++) {  
 g[i][j] = Math.*min*(g[i][j], g[i][k] + g[k][j]);  
 }  
 }  
 }  
*// Для каждой пары вершин выведем величину  
 // кратчайшего пути от i до j, или 0,  
 // если j не достижима из i* for (int i = 0; i < vertexCount; i++) {  
 for (int j = 0; j < vertexCount; j++) {  
 if (g[i][j] == *INF*) {  
 printWriter.print(0 + " ");  
 } else {  
 printWriter.print(g[i][j] + " ");  
 }  
 }  
  
 printWriter.println();  
 }  
  
 *// После выполнения программы необходимо закрыть  
 // потоки ввода и вывода* scanner.close();  
 printWriter.close();  
 }  
 }

#### **Задача1. Города и дороги**

В галактике "Milky Way" на планете "Neptune" есть N городов, некоторые из которых соединены дорогами. Император "Maximus" галактики "Milky Way" решил провести инвентаризацию дорог на планете "Neptune". Но, как оказалось, он не силен в математике, поэтому он просит вас сосчитать количество дорог.

**Входные данные**

В первой строке задается число N (0 ≤ N ≤ 100). В следующих N строках содержится по N чисел, каждое из которых является единичкой или ноликом. Причем, если в позиции (i,j) квадратной матрицы стоит единичка, то i-ый и j-ый города соединены дорогами, а если нолик, то не соединены.

**Выходные данные**

Выведите одно число – количество дорог на планете "Neptune".

**Примеры**

