类edge用来存放每条边中两个节点的信息。将输入的边的信息都存储在E[]数组中。每个点可达的多个点的信息存在po容器中。

首先使用tarjan算法，找出整个图所有的连通分量。将每个连通分量都看成一棵树。对图中的每个节点遍历，为了避免重复遍历，使用instack数组标记节点是否遍历过。用Stap数组来存储已经遍历过的节点序号。遍历时使用深度优先遍历。对每个点计算它的DFN和LOW。节点的DFN是遍历到该点的时间戳，LOW是节点或其子树所能够到达的已经遍历过的节点。当一个节点的DFN和LOW相等时，已经遍历的节点中形成了一个连通分量，将该节点取出，然后查找前一步遍历的节点直到完成该棵子树。

在tarjan中Bcnt为连通分量的个数，Belong数组记录了每个节点属于哪些连通分量，Bnum数组存储了每个连通分量中节点个数。完成tarjan后，遍历所有的边，如果节点A到节点B有边，且节点A和B不属于同一个连通分量，若节点A在编号为x的连通分量中，则将节点B加入TT[x]容器中。接着对每个连通分量使用dfs函数，在to[][]中记录该连通分量可达的点，使用vi[]数组标记该点是否遍历过以避免重复。最后to[][]中记录了所有连通分量各自可达的点。

最后遍历每条边，每次遍历时，如果两点不在一个连通分量中，则用cnt记录连通的节点个数之和，根据to[][]中的数字算出如果这条边变成双向则可增加的连通节点格式，得到最后额cnt。比较每条边的cnt大小。获得最大的那个输出。得到最终结果。