(1)设计算法以判断无向图G是否是一棵树。

首先遍历图，用一个数组保存结点访问情况，并设置所有值为false，当访问结点后，该结点对应的数组位置的元素改为true。统计访问的边以及结点数量，如果结点访问的数量同图中结点数量一致，并且访问过的边的次数是顶点个数减一的两倍，说明是一棵树

（2） 设计算法在图G的顶点子集U中找出距离顶点v0最近的顶点

（以弧数为单位）, 并要求时间尽可能地少。

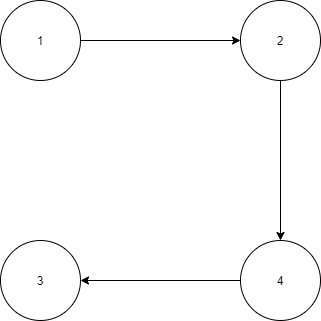
使用广度优先算法依次遍历图中每个顶点，考量将其加入到图中任意两个顶点之间是否会使该两个顶点之间的路径（当前）剪短，若是，则在二维数组中记录下该顶点。依次迭代就可以得到图中任意两顶点之间的最短路径

（3） 对例5－3中的图G（图5－13所示）的邻接表，不用还原

出原图，请执行dfs(1)，写出遍历序列，并构造出相应的

dfs生成树。

对每一个可能的分支路径深入到不能再深入为止，而且每个节点只能访问一次。可以通过图示清晰的说明。

1243

（4）设计算法以判断顶点vi到vj之间是否存在路径？若存在，则返回TRUE，否则返回FALSE。

建立邻接表存储各顶点，边，进行深度优先搜索，用visited[i]作标志，令visited[i]=0，访问过的结点令visited[i]=1，若一个点重复访问则不存在路径。