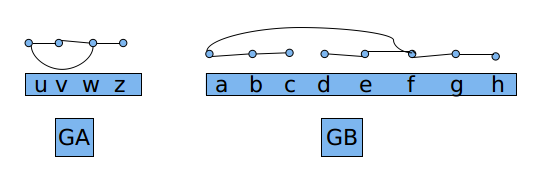
<http://www.4byte.cn/question/641545/is-there-any-simple-example-to-explain-ullmann-algorithm.html>

我们想知道GB是否有一个子图与GA同构,我们尝试吧GA的顶点映射到GB的顶点, 以便把GA的边映射的边映射到GB的边.



注意, 原始的(paper)和(the slides)都使用同一个矩阵M, 但是代码不是.

那是因为矩阵代表的和代码中possible\_assignments[i]代表的相同.

M[i][j]是确定的, 如果平[GA的]第i个顶点可以映射到GB的第j个顶点.

我将使用一个GB的顶点集candidates(u) etc.这个GB的u可以被映射.

首先, 观察GA的一个顶点是否只能映射到GB的一个顶点(at least as large degree), 所以最初candidates(u)=candidates(v)={a, b, e, f, g}, candidates(w)={f}, candidates(z)是GB的所有顶点.

现在我们第一次做”细化M”的过程, 这是Ullmann算法的主要部分.我们可以检查每当GB的一个顶点x在GA的candidates(y)之中时, y的每个邻居至少有一个x的邻居中, 如果这个检查失败, 我们从candidates(y)中删除x, 我们检查并除去直到再也不能除去.

例如, h在candidates(z)中, 然而, w是z的邻居, 但是在candidates(w)={f}中, 没有h的邻居, 因为z无法映射到h, 因为边{w, z}无法映射到一条边, 所以可以把h从candidates(z)中删除. “细化”后的结果是candidates(u)=candidates(v)=candidates(z)={a, e, g}, candidates(w)={f}.

现在我们开始回溯.

我们首先尝试把u映射到z, 我们设置candidates={a}并从其他candidates中删除a. “细化”后发现e, g不是a的邻居, 所以我们把e, g从candidates(v)中删除.candidates(v)这”叶”为空时我们从这个分支返回. 取消对candidates的改变.

现在我们映射u到e, 同样, candidates(v)到空结束.

最后, 映射u到g, 得到同样的结果.

我们推断GA不是GB的一个子图, 不用尝试所有8\*7\*6\*5此分配.