# [LCA题目总结](http://www.cnblogs.com/scau20110726/archive/2013/06/14/3135095.html)

好像LCA的题目并不多，现在就做了10道左右，但是找不到别的了，在此做个小总结，将来有更好的题目会不断更新

解决LCA问题，一般用3种方法

1.朴素方法：两个点都一直沿路径往上走，直到有某一个节点被经过两次并且是第一次出现这样的点，那么这个就是LCA

此方法最好理解，但是用得不多，但不代表没作用，有些题目还是需要用到的

2.LCA转RMQ（在线算法）：一般是将LCA转为RMQ问题，用ST算法求解，当然求解RMQ问题有很多方法，不过ST比较常用而已，这样做能及时回答每一个询问

3.Tarjan算法（离线算法）：利用Tarjan算法，不过要先读入所有询问再一并回答，建议认真学习Tarjan算法本质

\*\*\*\*\*\*具体问题具体分析，但是可以使用Tarjan算法的时候推荐使用Tarjan算法，代码量少速度快，另外Tarjan算法容易产生变形，值得深究\*\*\*\*\*\*\*\*\*\*\*\*\*

[**hdu 2586 How far away ？**](http://www.cnblogs.com/scau20110726/archive/2013/05/26/3100265.html)

模板题，直接求LCA，可以在线，离线算法均可解，可以测试一下模板

[**poj 1986 Distance Queries**](http://www.cnblogs.com/scau20110726/archive/2013/05/27/3102068.html)

模板题，直接求LCA

[**hdu 2874 Connections between cities**](http://www.cnblogs.com/scau20110726/archive/2013/05/30/3107462.html)

模板题，不过不是树是森林，所以某些点不存在LCA，要做判断

[**zoj 3195 Design the city**](http://www.cnblogs.com/scau20110726/archive/2013/05/26/3100270.html)

任然算是模板题，上面的题要求两点间的最短距离，这里要求3点间的最短距离，其实就是两两之间求一次LCA并计算出距离，然后相加除以2即可

[**hdu 3078 Network**](http://www.cnblogs.com/scau20110726/archive/2013/06/10/3130991.html)

LCA + 修改点权值 + 排序：每个点有初始的权值，一边查询一边伴随着修改某些点的权值，查询是从a到b路径中第k大的权值是多少。不需要太多的技巧，修改操作就直接修改，查询操作先求LCA，然后从a走到b保存下所有的权值，排序，然后直接输出即可

[**poj 2763 Housewife Wind**](http://www.cnblogs.com/scau20110726/archive/2013/06/01/3111715.html)

LCA + 修改边权：一边查询两点间的距离，一边修改某些边权。对于修改了某些边的边权，就要从此开始遍历下面的子孙后代更改他们的dir值（点到根的距离）。也不需要太多技巧，直接按题意实现即可，不过时间比较糟糕，用线段树或树状数组可以对修改操作进行优化，时间提升很多

[**poj 3694 Network**](http://www.cnblogs.com/scau20110726/archive/2013/05/29/3106073.html)

连通分量 + LCA ： 先缩点，再求LCA，并且不断更新，这题用了朴素方法来找LCA，并且在路径上做了一些操作

[**poj 3417 Network**](http://www.cnblogs.com/scau20110726/archive/2013/05/31/3110666.html)

LCA + Tree DP  :  在运行Tarjan处理完所有的LCA询问后，进行一次树DP，树DP不难，但是要想到用树DP并和这题结合还是有点难度

[**poj 3728 The merchant**](http://www.cnblogs.com/scau20110726/archive/2013/05/31/3110663.html)

LCA + 并查集的变形，优化：好题，难题，思维和代码实现都很有难度，需要很好地理解Tarjan算法中并查集的本质然后灵活变形，需要记录很多信息（有点dp的感觉）

[**hdu 3830 Checkers**](http://www.cnblogs.com/scau20110726/archive/2013/06/14/3135024.html)

LCA + 二分：好题，有一定思维难度。先建立出二叉树模型，然后将要查询的两个点调整到深度一致，然后二分LCA所在的深度，然后检验