树上莫队

总结时用到的博客：

https://blog.csdn.net/qq_39759315/article/details/88553210

https://www.cnblogs.com/RabbitHu/p/MoDuiTutorial.html

分块方式

首先需要解决$bzoj1086$王室联邦，此题的解法即为分块的方法

下只简述过程，每块内的节点个数在$[B,3B]$之间，操作方式为维护一个栈，从根节点开始dfs对于某一点x

1. 记录栈底的位置flg
2. 先访问所有子树，若访问完某棵子树后$top-flg\ge B$,则将$stack[flg+1]$至$stack[top]$中的点分为一块；
3. 再将x压入栈。

最后栈中剩余的点加入最后一个块中

树上莫队的具体操作

- 带修改问题对时间的排序方式与普通莫队相同
    - 在序列上操作
        - 欧拉序
        - dfs遍历序列，当访问到结点i时加入序列，回溯至i时再将i加入序列
        - 定义$in[i]$表示在欧拉序A中第一次出现的位置$out[i]$表示i第二次的位置

          对于路径x到y上的询问，不妨设$in[x]<in[y]$
        - 若$lca(x,y)=x$,统计欧拉序中子序列$A[in[x]],\ldots,A[in[y]]$中的出现过一次的节点的值即可回答询问

可以简单画图证明若是路径上的点当且仅当只其在子序列中只出现一次

- 若$lca(x,y)\neq x$,统计欧拉序中子序列$A[out[x]],\ldots,A[in[y]]$中的出现过一次的节点的值以及$lca(x,y)$的值即可回答询问

  通过画图会发现$lca(x,y)$并不在序列中，需要额外统计

◦ 在树上操作

- 直接在树上操作的问题在于并不能类似序列中一样只在左右移动，因而考虑记录一个vis数组，表示结点是否在当前处理的路径上，对于路径(u_1,v_1)到(u_2,v_2)的转移，只要将路径(u_1,u_2)和(v_1,v_2)上的点的vis值全部取反后在统计答案即可

- 需要注意在维护路径上点是并不考虑路径两端点的lca需要将其单独统计，上述方法的可行性以及不维护lca的原因如下(引用VFleaKing的证明)

> T(v, u) = S(root, v) xor S(root, u)
>
> 观察将curV移动到targetV前后T(curV, curU)变化：
>
> T(curV, curU) = S(root, curV) xor S(root, curU)
>
> T(targetV, curU) = S(root, targetV) xor S(root, curU) 取对称差：
>
> T(curV, curU) xor T(targetV, curU)= (S(root, curV) xor S(root, curU)) xor (S(root, targetV) xor S(root, curU))

> 由于对称差的交换律、结合律：
>
> T(curV, curU) xor T(targetV, curU)= S(root, curV) xor S(root, targetV)
>
> 两边同时xor T(curV, curU)
>
> T(targetV, curU)= T(curV, curU) xor S(root, curV) xor S(root, targetV)
>
> 发现最后两项很爽……哇哈哈
>
> T(targetV, curU)= T(curV, curU) xor T(curV, targetV)

- 总结一下树上的操作和序列的不同点

  1. 分块方式不同
  2. 无法直接类比普通莫队序列上的操作，需要通过一些辅助使树上操作变成欧拉序上的操作，或者直接在树上移动，移动是都是通过vis数组判断节点的值是否需要统计(添加或删除)

From:
https://wiki.cvbbacm.com/ - **CVBB ACM Team**

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:mian:gary:mos_algorithm_tree&rev=1590674312

Last update: **2020/05/28 21:58**