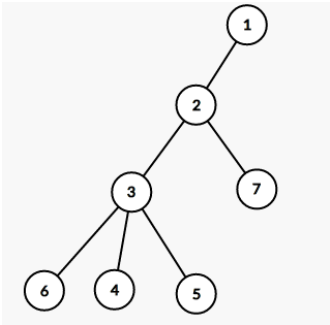
树上莫队

Mo's Algorithm on Tree

Idea：使用欧拉序将树转化为序列。欧拉序的构建方法为：dfs 遍历整棵树，当一个节点入栈或出栈时将其加入欧拉序。例如，下图的一个欧拉序为：12366445537721.



每个节点会在欧拉序中出现两次。记节点 i 的开始时间戳（第 1 次出现位置）为 st_i，结束时间戳（第 2 次出现位置）为 ed_i，那么在询问 u 到 v 的路径时，假设 $st_u < st_v$，有两种情况：

- 若 u 是 v 的祖先，则 $[st_u, st_v]$ 中仅出现一次的点就是 u, v 路径上的点；
- 否则，$[ed_u, st_v]$ 中仅出现一次的点外加上 $\text{lca}(u, v)$ 就是 u, v 路径上的点。（注意 $\text{lca}(u, v)$ 不在区间内，要单独算）

这样，我们就把树上的问题转化为了序列问题，然后使用序列莫队即可。

ATT：欧拉序的长度为 $2n$，做莫队时记得循环到 $2n$.

Code：

```cpp
// ========== calculate lca and euler sequence ========== //
int st[N], ed[N], tot, fa[N][25], dep[N], euler[N<<1];
void dfs(int x, int f, int depth){
    euler[++tot] = x, st[x] = tot;
    fa[x][0] = f, dep[x] = depth;
    for(auto &to : edge[x]){
        if(to == f) continue;
        dfs(to, x, depth+1);
    }
    euler[++tot] = x, ed[x] = tot;
}
inline void init(){
    for(int j = 1; (1 << j) <= n; j++)
        for(int i = 1; i <= n; i++)
            if(fa[i][j-1])
                fa[i][j] = fa[fa[i][j-1]][j-1];
}
inline int lca(int x, int y){
    if(dep[x] < dep[y]) swap(x, y);
    for(int i = 20; i >= 0; i--)
        if(dep[x] - (1 << i) >= dep[y])
            x = fa[x][i];
    if(x == y)  return x;
    for(int i = 20; i >= 0; i--)
        if(fa[x][i] && fa[x][i] != fa[y][i])
            x = fa[x][i], y = fa[y][i];
    return fa[x][0];
}
// ======================================================== //

int nowAnswer, cnt[N];
bool vis[N];
inline void update(int x){
    cnt[color[x]] += !vis[x] ? 1 : -1;
    if(vis[x] && cnt[color[x]] == 0)    nowAnswer--;
    if(!vis[x] && cnt[color[x]] == 1)   nowAnswer++;
    vis[x] = !vis[x];
}

```

```cpp
int main(){
    scanf("%d%d", &n, &m);
    sq = (n << 1) / sqrt(m);
    for(int i = 1; i <= (n << 1); i++)  belong[i] = (i - 1) / sq + 1;

    for(int i = 1; i <= n; i++) scanf("%d", &color[i]);
    disc(color);
    for(int i = 1; i < n; i++){
        int u, v; scanf("%d%d", &u, &v);
        edge[u].pb(v), edge[v].pb(u);
    }
    dfs(1, 0, 1);
    init();
    for(int i = 1; i <= m; i++){
        int u, v, l; scanf("%d%d", &u, &v); l = lca(u, v);
        if(st[u] > st[v])   swap(u, v);
        q[i] = {i, l == u ? st[u] : ed[u], st[v], l == u ? 0 : l};
    }
    sort(q+1, q+m+1);
    for(int i = 1, l = 1, r = 0; i <= m; i++){
        for(; l > q[i].l; l--)  update(euler[l-1]);
        for(; r < q[i].r; r++)  update(euler[r+1]);
        for(; l < q[i].l; l++)  update(euler[l]);
        for(; r > q[i].r; r--)  update(euler[r]);
        if(q[i].lca)    update(q[i].lca);
        q[i].ans = nowAnswer;
        if(q[i].lca)    update(q[i].lca);
    }
    sort(q+1, q+m+1, [&](const Query &A, const Query &B){ return A.id < B.id; } );
    for(int i = 1; i <= m; i++) printf("%d\n", q[i].ans);
    return 0;
}
```