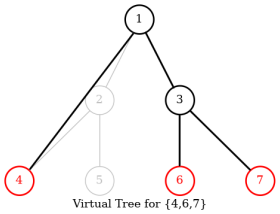
虚树

Virtual Tree

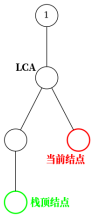
Idea：虚树将特定的 k 个点及它们的 lca 取出建成一个新的树。



构建方法：将指定 k 个点按照 dfs 序排序，依次插入到虚树中，这里先插入树根节点以方便后续操作。插入时，维护一个栈，类似于 dfs 时的系统栈，也即栈内元素连起来是树上的一条路径（当然这里不一定连续）。

考虑如何加入一个点 x，设 $z=\text{lca}(x,\text{stk.top}())$，分类讨论：

- 若 $z=\text{stk.top}()$，说明 x 在 stk.top() 的子树中，直接将 x 入栈；
- 否则，x 不在 stk.top() 的子树中，这时一直弹栈，直到遇到 z（z 本身在栈中时）或 z 的祖先（z 本身不在栈中时，这时把 z 入栈），然后把 x 入栈。



我们在弹栈时将弹出的元素和新的栈顶元素连边，如此便构造出了虚树。所以，所有点插入完成后记得要全部弹出栈。

构建出虚树后就可以进行后续操作了，通常是树形 dp。

Complexity：$O(k\lg n)$ 构建（采用倍增求 lca）。

Code：

```
namespace VT{ // Virtual Tree
    vector<int> edge[N];
    stack<int> stk;
    vector<int> ver; // vertices in the virtual tree
    void build(vector<pii> &sv){
        // sv stores (dfn[x], x) pairs
        // among which x are special vertices
        sort(sv.begin(), sv.end());
        ver.clear();
        stk.push(1), ver.pb(1);
        for(auto &v : sv){
            int x = v.second;
            if(x == 1)  continue;
            int z = LCA::lca(x, stk.top());
            if(z == stk.top())  stk.push(x), ver.pb(x);
            else{
                while(!stk.empty() && LCA::dep[stk.top()] > LCA::dep[z]){
                    int t = stk.top(); stk.pop();
                    if(!stk.empty()){
                        if(LCA::dep[stk.top()] >= LCA::dep[z])
                            edge[t].pb(stk.top()), edge[stk.top()].pb(t);
                        else    edge[t].pb(z), edge[z].pb(t);
                    }
                }
                if(stk.empty() || z != stk.top())   stk.push(z), ver.pb(z);
                stk.push(x), ver.pb(x);
            }
        }
        while(!stk.empty()){
            int t = stk.top(); stk.pop();
```

```
            if(!stk.empty())
                edge[t].pb(stk.top()), edge[stk.top()].pb(t);
        }
    }

    int solve(){
        // ... solve on virtual tree
    }

    void clear(){
        for(auto &v : ver){
            edge[v].clear();
            // ... clear other info
        }
    }
}

int main(){
    // ... input
    LCA::dfs(1, 0, 1);
    LCA::init();
    int q; for(scanf("%d", &q); q; q--){
        int k; scanf("%d", &k);
        vector<pii> sv; // special vertices
        while(k--){
            int a; scanf("%d", &a);
            sv.pb(mp(LCA::dfn[a], a));
            tag[a] = true;
        }

        VT::build(sv);
        printf("%d\n", VT::solve());

        VT::clear();
        for(auto &v : sv)   tag[v.second] = false;
    }
    return 0;
}
```