树论

```
/* 预处理 O(1) LCA */
/* LCA 是 dfs 序上两点间深度最小的点 */
int ar[22][CN << 1], dfn[CN << 1], dep[CN << 1], lg[CN << 1], idx;
void bdlca(){
    for(int i = 2; i <= idx; i++) lg[i] = lg[i >> 1] + 1;
    for(int k = 1; k <= 20; k++) for(int i = 1; i <= idx; i++){
        int j = i + (1 << (k - 1)); if(j > idx) continue;
        if(dep[ar[k - 1][i]] < dep[ar[k - 1][j]]) ar[k][i] = ar[k - 1][i];
        else ar[k][i] = ar[k - 1][j];
    }
}
int lca(int l, int r){
    l = dfn[l], r = dfn[r]; if(l > r) swap(l, r);
    int g = lg[r - l + 1], e = 1 << g;
    return dep[ar[g][l]] < dep[ar[g][r - e + 1]] ? ar[g][l] : ar[g][r - e + 1];
}
void dfs(int u, int p){
    dep[u] = dep[p] + 1, ar[0][++idx] = u, dfn[u] = idx;
    for(int l = G[u].size(), i = 0; i < l; i++){
        int v = G[u][i]; if(v == p) continue;
        dfs(v, u), ar[0][++idx] = u;
    }
}

/* 轻重链剖分 HLD */
int dep[CN], fa[CN], top[CN], imp[CN], sz[CN], dfn[CN], idx;
void dfs1(int u, int p){
    dep[u] = dep[p] + 1, fa[u] = p, sz[u] = 1;
    for(int l = G[u].size(), i = 0; i < l; i++){
        int v = G[u][i]; if(v == p) continue;
        dfs1(v, u), sz[u] += sz[v];
        if(sz[imp[u]] < sz[v]) imp[u] = v;
    }
}
void dfs2(int u, int p, int t){
    dfn[u] = ++idx, top[u] = t; if(imp[u]) dfs2(imp[u], u, t);
    for(int l = G[u].size(), i = 0; i < l; i++){
        int v = G[u][i]; if(v == p || v == imp[u]) continue;
        dfs2(v, u, v);
    }
}

/* Virtual Tree */
bool cmp(int i, int j) {return dfn[i] < dfn[j];} vector<int> T[CN];
int stk[CN], top;
void bdvt(int a[]){
    sort(a + 1, a + a[0] + 1, cmp);
    stk[top = 1] = 1, T[1].clear();
    for(int i = 1; i <= a[0]; i++){
        int l = lca(a[i], stk[top]); if(a[i] == 1) continue;
        if(l ^ stk[top]){
```

```
            while(dep[stk[top - 1]] > dep[l]) T[stk[top - 1]].pb(stk[top]), top--;
            if(l ^ stk[top - 1]) T[l].clear(), T[l].pb(stk[top]), stk[top] = l;
            else T[l].pb(stk[top]), top--;
        }
        T[a[i]].clear(), stk[++top] = a[i];
    }
    for(int i = 1; i < top; i++) T[stk[i]].pb(stk[i + 1]);
}
```