[最近公共祖先（LCA）Easy](http://oj.xingyuanoj.top/p/B1161)

```cpp
#include <bits/stdc++.h>

using namespace std;

vector<int> edges[10010];
int fa[10010]; // fa[i]表示 i号节点的父亲是谁
int deep[10010]; // 表示 i号节点深度是多少

void dfs(int u, int f){ // u表示u号节点 f表示u号节点的父亲
    for(auto x : edges[u]){
        if(x == f){ // 儿子是父亲，说明我们就是从父亲来的
            continue;
        }
        fa[x] = u; // x点的父亲是 u
        deep[x] = deep[u] + 1; // 儿子的深度 = 父亲深度 + 1
        dfs(x, u); // 新的儿子是 x  x的父亲是 u
    }
}

int lca(int u, int v){
    if(deep[u] < deep[v]){
        swap(u, v);
    }
    int t = deep[u] - deep[v];
    while(t -- ){ // 保证了 u 和 v 同一深度
        u = fa[u];
    }
    while(u != v) { // 如果这两个点 不相等  同时往上跳
        u = fa[u];
        v = fa[v];
    }
    return u;
}

int main(){
    int n, m, s;
    scanf("%d %d %d", &n, &m, &s);
    for(int i = 1; i <= n - 1; i ++ ){
        int u, v;
        scanf("%d %d", &u, &v);
        edges[u].push_back(v);
        edges[v].push_back(u);
    }
    deep[s] = 0;
    dfs(s, 0);
    while(m -- ){
        int u, v;
        scanf("%d %d", &u, &v);
        printf("%d\n", lca(u, v));
    }
    return 0;
```

```
}
```

[点的距离Easy](http://oj.xingyuanoj.top/p/B1166)

```
#include <bits/stdc++.h>

using namespace std;

vector<int> edges[10010];
int fa[10010]; // fa[i]表示 i号节点的父亲是谁
int deep[10010]; // 表示 i号节点深度是多少

void dfs(int u, int f){ // u表示u号节点 f表示u号节点的父亲
    for(auto x : edges[u]){
        if(x == f){ // 儿子是父亲，说明我们就是从父亲来的
            continue;
        }
        fa[x] = u; // x点的父亲是 u
        deep[x] = deep[u] + 1; // 儿子的深度 = 父亲深度 + 1
        dfs(x, u); // 新的儿子是 x  x的父亲是 u
    }
}

int lca(int u, int v){
    if(deep[u] < deep[v]){ // 为了保证 u 的深度一定大于 v 的深度  保证让u去跳
        swap(u, v);
    }
    int t = deep[u] - deep[v];
    while(t -- ){ // 保证了 u 和 v 同一深度
        u = fa[u];
    }
    while(u != v) { // 如果这两个点 不相等  同时往上跳
        u = fa[u];
        v = fa[v];
    }
    return u;
}

int main(){
    int n, m;
    scanf("%d", &n);
    for(int i = 1; i <= n - 1; i ++ ){
        int u, v;
        scanf("%d %d", &u, &v);
        edges[u].push_back(v);
        edges[v].push_back(u);
    }
    dfs(1, 0);
    scanf("%d", &m);
    while(m -- ){
        int u, v;
        scanf("%d %d", &u, &v);
        printf("%d\n", deep[u] + deep[v] - 2 * deep[lca(u, v)]);
    }
    return 0;
}
```