倍增求LCA

最近公共祖先（LCA)Hard

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

using namespace std;

constexpr int N = 500010;

int fa[N][21]; // lca fa[i][j]:  表示 第i号点向上跳2^j次方后跳到了哪个节点
vector<int> edges[N]; // 存树
int deep[N]; // 存深度
int n, m, s; // s:根

void dfs(int u, int f){ // u 表示 当前节点 f表示的是u的父亲
    for(auto x : edges[u]){ // 遍历 u的所有儿子是 x
        if(x == f){ // 儿子是父亲 不考虑
            continue;
        }
        deep[x] = deep[u] + 1; //  儿子的深度 = 父亲深度 + 1
        fa[x][0] = u; // fa[儿子][0] 儿子向上跳 2^0（1）是他的父亲
        dfs(x, u); // 当前的 儿子是 x  父亲是 u
    }
}

void init(){
    for(int j = 1; j <= 20; j ++ ){
        for(int i = 1; i <= n; i ++ ){
            if(fa[i][j - 1]){
                fa[i][j] = fa[fa[i][j - 1]][j - 1]; // 先跳到i的2^(j -1)的位置 然后再跳2^(j-1)
                // 跳2^2 （4） = 2^1 + 2^1  = 2 + 2
            }
        }
    }
}

int lca(int u, int v){
    if(deep[u] < deep[v]){ // 始终保证 u的深度 大于 v 的深度
        swap(u, v);
    }

    int t = deep[u] - deep[v]; // 相差的深度
    for(int i = 0; i <= 20; i ++ ){
        if((t >> i) & 1){ // 看 二进制下 t的第i位是不是1  是1的话表示跳2^i
            u = fa[u][i];
        }
    }

    if(u == v){ // 刚好跳到第 v号节点
        return u;
    }
```

```
    for(int i = 20; i >= 0; i -- ){ // 一定要反着枚举
        if(fa[u][i] != fa[v][i]){ // 如果不同就跳 最后一定会跳到lca的儿子
            u = fa[u][i];
            v = fa[v][i];
        }
    }

    return fa[u][0];

}

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