lca

```
struct edge
{
    int t, next;
} e[maxn << 1];
int tot, head[maxn];
void add(int x, int y)
{
    e[++tot].t = y;
    e[tot].next = head[x];
    head[x] = tot;
}
//倍增的方法ST表的方法实现查询；
//预处理。
int N, M, S;
int depth[maxn];
int fa[maxn][60];
int lg[maxn];
void intit() //常数优化一次查询处理。
{
    lg[1] = 0;
    lg[2] = 1;
    for (int i = 3; i <= N; i++)
        lg[i] = lg[i / 2] + 1;
}
// st表预处理
void dfs(int now, int par) //当前节点，父亲节点。
{
    fa[now][0] = par;
    depth[now] = depth[par] + 1;
    for (int i = 1; i <= lg[depth[now]]; i++)
        fa[now][i] = fa[fa[now][i - 1]][i - 1];
    for (int i = head[now]; i; i = e[i].next) //这个码风也可以的，但是别人难看懂而已。
        if (e[i].t != par)
            dfs(e[i].t, now);
}
int LCA(int x, int y)
{
    //统一形式，简化代码。假设x深度大于y的。统一的处理方式。
    if (depth[x] < depth[y])
        swap(x, y);
    //先转化成同深度；
    while (depth[x] > depth[y])
        x = fa[x][lg[depth[x] - depth[y]]];
    if (x == y)
        return x;
    for (int i = lg[depth[x]]; i >= 0; i--)
        if (fa[x][i] != fa[y][i])
            x = fa[x][i], y = fa[y][i];
    return fa[x][0];
}
```