最近公共祖先
——倍增算法代码实现

主讲人：邓哲也

倍增代码算法实现

- 树的存储结构：
- struct edge{
- int v, next;
- }e[M];
- int h[N], etot;

- 采用邻接链表来存树，其中h[]是头数组，e[]是边数组

倍增代码算法实现

- 添加边
- void add_edge(int x, int y) {
- e[etot].v = y;
- e[etot].next = h[x];
- h[x] = etot ++;
- }

- h[]初始化为-1，etot初始化为0

倍增代码算法实现

- dfs，获得每个节点的父节点信息
- void dfs(int u, int fa) {
- up[u][0] = fa;
- for (int i = h[u]; i != -1; i = e[i].next) {
- if (e[i].v != fa)
- dfs(e[i].v, u);
- }
- }

倍增代码算法实现

- 计算up数组
- void init() {
- for (int k = 1; (1 << k) <= n; k ++)
- for (int i = 1; i <= n; i ++)
- up[i][k] = up[up[i][k - 1]][k - 1];
- }

倍增代码算法实现

- 计算LCA

```
int lca(int u, int v) {
    if (dep[u] < dep[v])
        swap(u, v);
    if (dep[u] != dep[v]) {
        for (int k = 20; k >= 0; k --)
            if (dep[up[u][k]] >= dep[v])
                u = up[u][k];
    }
    if (u == v)
        return u;
    for (int k = 20; k >= 0; k --)
        if (up[u][k] != up[v][k])
            u = up[u][k], v = up[v][k];
    return up[u][0];
}
```

下节课再见