树状DP简介

主讲人：邓哲也

树形DP

树形动态规划（树形DP）

状态图是一棵树，状态转移也发生在树上。

父节点的值通过所有子节点的值得到，一般在 dfs 的过程中完成 DP。

树形DP

```
void dfs(u){
        for v in u.child:
                dfs(v)
                use dp[v] to update dp[u]
}
```

树上最长链

给定一颗树，每条边有边权。

计算一条最长链。

要求时间复杂度 O(n)

树上最长链

定义 f[i] 是以 i 为根的子树中的最长链。

显然 f[i] = max(f[i], f[j]) （j 是 i 的子节点）

这样只考虑了不经过 i 的路径。

如果要考虑经过 i 的路径，就需要选择两个子节点。

把两个子节点往下走的最长路相加，再加上它们到 i 的路径和，去更新 f[i]。

树上最长链

定义 g[i] 表示从 i 往下走最远能走多少。

g[i] = max(g[i], g[j] + w(i, j)) （j 是 i 的子节点）

对于叶节点 f = g = 0

树上最长链

```
void dfs(int u){
	f[u] = g[u] = 0;
	int maxg1 = -1e9, maxg2 = -1e9;
	for (int i = head[u]; i != -1; i = e[i].next){
		int v = e[i].to;
		dfs(v);
		g[u] = max(g[u], e[i].w + g[v]);
		f[u] = max(f[u], f[v]);
		…
```

树上最长链

```
            …
            if (g[v] + e[i].w > maxg1){
                maxg2 = maxg1;
                maxg1 = g[v] + e[i].w;
            }else if (g[v] + e[i].w > maxg2){
                maxg2 = g[v] + e[i].w;
            }
        }
        f[u] = max(f[u], maxg1 + maxg2);
}
```

dfs(root) 答案就是 f[root]

最大权值和子树

给定一颗树，每个点有权值（可正可负）。

求一个子树，使得权值和最大。

要求时间复杂度 O(n)

最大权值和子树

设 f[i] 表示以 i 为根的子树里的最大权值和子树。

为了方便转移，我们需要用新的一维表示是否选第 i 个点

f[i][1] 表示选，f[i][0] 表示不选

f[i][1] = w[i] + sum(max(0, f[j][1]) | j 是 i 的子节点)

f[i][0] = max(0, max(max(f[j][0], f[j][1]) | j 是 i 的子节点))

下节课再见