树形 DP

李淳风

长郡中学

2024 年 9 月 24 日

前言

树形 DP 就是动态规划在树形结构上的实现。给定一棵有 n 个节点的树（通常是无根树，有 $n-1$ 条无向边），我们可以任选一个节点为根节点，从而定义出每个节点的深度和每棵子树的根。
在树上设计动态规划算法时，一般就以节点从深到浅（子树从小到大）的顺序来作为 DP 的阶段。DP 的状态标识中，第一维通常是节点编号（代表以该节点为根的子树）。大多数时候，我们采用递归的方式实现树形动态规划。对于每个节点 x，先递归在它的每个子节点上进行 DP，在回溯时，从子节点向节点 x 进行状态转移。

例题

没有上司的舞会

Ural 大学有 n 名职员，编号为 $1 \sim n$。他们的关系就像一棵以校长为根的树，父节点就是子节点的直接上司。每个职员有一个快乐指数，用整数 H_i 给出，其中 $1 \leq i \leq n$。现在要召开一场周年庆宴会，不过，没有职员愿意和直接上司一起参会。在满足这个条件的前提下，主办方希望邀请一部分职员参会，使得所有参会职员的快乐指数总和最大，求这个最大值。

例题

没有上司的舞会

Ural 大学有 n 名职员，编号为 $1 \sim n$。他们的关系就像一棵以校长为根的树，父节点就是子节点的直接上司。每个职员有一个快乐指数，用整数 H_i 给出，其中 $1 \le i \le n$。现在要召开一场周年庆宴会，不过，没有职员愿意和直接上司一起参会。在满足这个条件的前提下，主办方希望邀请一部分职员参会，使得所有参会职员的快乐指数总和最大，求这个最大值。

根据前言部分，我们就以节点编号（子树的根）作为 DP 状态的第一维。因为职员是否愿意参加只和它的直接上司是否参加有关，所以我们在每棵子树递归完成时，保留两个“代表信息”——根节点不参加时整棵子树的最大快乐指数和，以及根节点参加时整棵子树的最大快乐指数和，我们可以分别用 $f[x][0], f[x][1]$ 来表示。

没有上司的舞会

那么，根据我们的定义，我们可以试着写出转移方程。
$f[x][0]$ 表示 x 节点不参加舞会时，x 子树内的最大快乐指数和，那么 x 的所有子节点（直接下属）可以参加，也可以不参加：

$$f[x][0] = \sum_{s \in Son(x)} \max(f[s][0], f[s][1])$$

$f[x][1]$ 表示 x 节点参加舞会时，x 子树内的最大快乐指数和，那么 x 的所有子节点（直接下属）都不能参加（别忘了 H_x）：

$$f[x][1] = H_x + \sum_{s \in Son(x)} \max(f[s][0])$$

其中，$Son(x)$ 表示 x 的子节点集合。
本题一棵有根树，所以我们需要先找出没有上司的节点 $root$ 作为根。
最终的目标值是 $\max(f[root][0], f[root][1])$，时间复杂度为 $O(n)$。

例题

选课

在大学里每个学生，为了达到一定的学分，必须从很多课程里选择一些课程来学习，在课程里有些课程必须在某些课程之前学习，如高等数学总是在其它课程之前学习。现在有 n 门功课，第 i 门课有 s_i 个学分，每门课有一门或没有直接先修课（若课程 a 是课程 b 的先修课即只有学完了课程 a，才能学习课程 b)。一个学生要从这些课程里选择 m 门课程学习，问他能获得的最大学分是多少？

$1 \le n \le 300, 1 \le m \le 300, 1 \le s_i \le 30$

例题

选课

在大学里每个学生，为了达到一定的学分，必须从很多课程里选择一些课程来学习，在课程里有些课程必须在某些课程之前学习，如高等数学总是在其它课程之前学习。现在有 n 门功课，第 i 门课有 s_i 个学分，每门课有一门或没有直接先修课（若课程 a 是课程 b 的先修课即只有学完了课程 a，才能学习课程 b）。一个学生要从这些课程里选择 m 门课程学习，问他能获得的最大学分是多少？

$1 \le n \le 300, 1 \le m \le 300, 1 \le s_i \le 30$

由于每门课的先修课最多只有一门（对应着树中每个节点至多只有 1 个父亲），所以这 n 门课程构成了森林结构（由若干棵树组成，可能有不止一门课没有先修课）。为了方便状态设计，我们可以新建一个“虚拟课程”——0 号节点，作为“实际上没有先修课的课程”的先修课，把包含 n 个点的森林转化为包含 $n+1$ 个节点的树，其中节点 0 为根节点。

选课

之后我们就开始划分阶段，设计状态了。由于还有选择 m 门课程的限制，我们可以设 $f[x][t]$ 表示在以 x 为根的子树中选择 t 门课程能够获得的最高学分。设 x 的子节点集合为 $Son(x)$，子节点个数 $p = |Son(x)|$。修完 x 这门课后，对于 x 的每个子节点 $y_i \in Son(x)$，我们可以在以 y_i 为根的子树中选修若干门课（记为 c_i），在满足 $\sum c_i = t - 1$ 的基础上获得尽量多的学分。

当 $t = 0$ 时，显然 $f[x][t] = 0$。当 $t > 0$ 时，根据以上分析，我们可以得到状态转移方程：

$$f[x][t] = \max_{(\sum_{i=1}^{p} c_i)=t-1} \left\{ \sum_{i=1}^{p} f[y_i][c_i] \right\} + s_x$$

选课

之后我们就开始划分阶段，设计状态了。由于还有选择 m 门课程的限制，我们可以设 $f[x][t]$ 表示在以 x 为根的子树中选择 t 门课程能够获得的最高学分。设 x 的子节点集合为 $Son(x)$，子节点个数 $p = |Son(x)|$。修完 x 这门课后，对于 x 的每个子节点 $y_i \in Son(x)$，我们可以在以 y_i 为根的子树中选修若干门课（记为 c_i），在满足 $\sum c_i = t - 1$ 的基础上获得尽量多的学分。
当 $t = 0$ 时，显然 $f[x][t] = 0$。当 $t > 0$ 时，根据以上分析，我们可以得到状态转移方程：

$$f[x][t] = \max_{(\sum_{i=1}^{p} c_i)=t-1} \left\{ \sum_{i=1}^{p} f[y_i][c_i] \right\} + s_x$$

这个方程实际上是一个分组背包模型。有 $p = |Son(x)|$ 组物品，每组物品都有 $t - 1$ 个，其中第 i 组的第 j 个物品的体积为 j，价值为 $f[y_i][j]$，背包的总容积为 $t - 1$。每组物品中至多能选择一个物品（每个子节点 y 只能选一个状态转移到 x），使得物品体积不超过 $t - 1$ 的前提下（最多选修 $t - 1$ 门课程），物品价值总和最大。当然，$x = 0$ 是一个特例，虚拟的根节点不需要被选修，此时背包容积为 t。

选课

关键部分代码如下:

```
void dp(int x){
    f[x][0]=0;
    for(int i=0;i<son[x].size();i++){
        int y=son[x][i];
        dp(y);//先递归计算子问题
        for(int t=m;t>=0;t--)//枚举当前背包体积
            for(int j=0;j<=t;j++)//枚举选一组物品中的哪一个
                f[x][t]=max(f[x][t],f[x][t-j]+f[y][j]);
    }
    if(x!=0)//选修 x 本身
        for(int t=m;t>0;t--)
            f[x][t]=f[x][t-1]+s[x];
}
```

这类题目被称为背包类树形 DP，又称有树形依赖的背包问题，实际上是背包与树形 DP 的结合。通常我们把当前背包的“体积”作为第二位状态，转移时我们实际上就是在处理分组背包问题。

例题

Accumulation Degree

有一个树形的水系，由 $n-1$ 条河道和 n 个交叉点组成。我们可以把交叉点看作树中的节点，编号为 $1 \sim n$，河道则看作树中的无向边。每条河道都有一个容量，连接 x 和 y 的河道的容量记为 $c(x, y)$。河道中单位时间流过的水量不能超过河道的容量。

有一个节点是整个水系的发源地，可以源源不断地流出水，我们称之为源点。除了源点之外，树中所有度数为 1 的节点都是入海口，可以吸收无限多的水，我们称之为汇点。也就是说，水系中的水从源点出发，沿着每条河道，最终流向各个汇点。

在整个水系稳定时，每条河道中的水都以单位时间固定的水量流向固定的方向。除源点和汇点之外，其余各点不贮存水，也就是流入该点的河道水量之和等于从该点流出的河道水量之和。整个水系的流量就定义为源点单位时间发出的水量。

在流量不超过河道容量的前提下，求哪个点作为源点时，整个水系的流量最大，输出这个最大值。

$n \leq 2 * 10^5$。

Accumulation Degree

本题是一个“不定根”的树形 DP 问题。这类题目的特点是，给定一个树形结构，需要以每个节点为根进行一系列统计。我们一般通过两次扫描来解决这类题目：

- 第一次扫描时，任选一个点为根，在“有根树”上执行一次树形 DP，在回溯时进行自底向上的状态转移。
- 第二次扫描时，从刚才选的根出发，对整棵树执行一次深度优先遍历，在每次递归前进行自顶向下的推导。

Accumulation Degree

我们先来考虑本题的朴素做法——枚举源点 s，然后计算水系流量。把源点作为树根，整个水系就变成了一棵有根树，每条河道的方向也可以直接得出。在这种情况下，每个节点只会从父亲节点获得水源，并流向它的子节点，每个节点的“流域”就是以该点为根的子树。这样就非常符合树形 DP 的运用场景——每棵子树都是一个子问题。
设 $D_s[x]$ 表示在以 x 为根的子树中，把 x 作为源点，从 x 出发流向子树的流量最大是多少。

$$D_s[x] = \sum_{y \in Son(x)} \begin{cases} \min(D_s[y], c(x, y)) & y \text{ 的度数大于 } 1 \\ c(x, y) & y \text{ 的度数等于 } 1 \end{cases}$$

对于枚举的每个源点 s，可以用树形 DP 在 $O(n)$ 的时间内求出 D_s 数组，并用 $D_s[s]$ 更新答案。所以，最终的时间复杂度为 $O(n^2)$。

Accumulation Degree

用“二次扫描与换根法”代替源点的枚举，可以在 $O(n)$ 的时间内解决整个问题。首先，我们任选一个源点作为根节点，记为 $root$，然后采用上面的代码进行一次树形 DP，求出 D_{root} 数组，简单记为 D 数组。
设 $f[x]$ 表示把 x 作为源点时的最大流量，显然有 $f[root] = D[root]$。
假设 $f[x]$ 已经被正确求出，考虑其子节点 y，$f[y]$ 尚未计算。而 $f[y]$ 由两部分组成。

- 从 y 流向以 y 为根的子树的流量，已经计算并被保存在了 $D[y]$ 中。
- 从 y 沿着到父节点 x 的河道，进而流向水系中其它部分的流量。

因为把 x 作为源点的总流量为 $f[x]$，从 x 流向 y 的流量为 $\min(D[y], c(x, y))$，所以 x 流向 y 以外其他部分的流量就是二者之差。
于是，把 y 作为源点时，先流到 x，再流向其他部分的流量，就是把这个“差”再与 $c(x, y)$ 取最小值后的结果。
当 x 的度数大于 1 时，有：

$$f[y] = D[y] + \begin{cases} \min(f[x] - \min(D[y], c(x,y)), c(x,y)) & y \text{ 的度数大于 } 1 \\ \min(f[x] - c(x,y), c(x,y)) & y \text{ 的度数等于 } 1 \end{cases}$$

x 度数为 1 时：

$$f[y] = D[y] + c(x, y)$$