树型动态规划

树本身就是一个递归的结构，所以在树上进行动态规划或者递推是最合适不过的事情。

必要条件：子树之间不可以相互干扰，如果本来是相互干扰的，那么我们必须添加变量使得他们不相互干扰。

Party at Hali-Bula

题目大意：n 个人形成一个关系树，每个节点代表一个人，节点的根表示这个人的唯一的直接上司，只有根没有上司。要求选取一部分人出来，使得每 2 个人之间不能有直接的上下级的关系，求最多能选多少个人出来，并且求出获得最大人数的选人方案是否唯一。

这是一个经典的树型动态规划，人之间的关系形成树型结构，简单的染色统计是不正确的

DP 部分：用 dp[i][0]表示不选择 i 点时，i 点及其子树能选出的最多人数，dp[i][1]表示选择 i 点时，i 点及其子树的最多人数。

状态转移方程:

对于叶子节点： dp[k][0] = 0, dp[k][1] = 1

对于非叶子节点 i： dp[i][0] = Σmax(dp[j][0], dp[j][1]) (j 是 i 的儿子)

dp[i][1] = 1 + Σdp[j][0] (j 是 i 的儿子)

最多人数即为：max(dp[0][0], dp[0][1])

如何判断最优解是否唯一?

新加一个状态 dup[i][j]，表示相应的 dp[i][j]是否是唯一方案。

对于叶子结点：dup[k][0] = dup[k][1] = 1.

对于非叶子结点：

1：i 的任一儿子 j，若(dp[j][0] > dp[j][1] 且 dup[j][0] == 0) 或 (dp[j][0] < dp[j][1] 且 dup[j][1] == 0) 或 (dp[j][0] == dp[j][1])，则 dup[i][0] = 0

2：i 的任一儿子 j 有 dup[j][0] = 0, 则 dup[i][1] = 0

Strategic game

题目大意：一城堡的所有的道路形成一个 n 个节点的树，如果在一个节点上放上一个士兵，那么和这个节点相连的边就会被看守住，问把所有边看守住最少需要放多少士兵。

典型的树型动态规划：

dproot[i]表示以 i 为根的子树，在 i 上放置一个士兵，看守住整个子树需要多少士兵。

all[i]表示看守住整个以 i 为根的子树需要多少士兵。

状态转移方程：

叶子节点：dproot[k] =1； all[k] = 0；

非叶子节点： dproot[i] = 1 + Σall[j](j 是 i 的儿子)；

all[i] = min(dproot[i], Σdproot[j](j 是 i 的儿子))；