简单树型动态规划

——成都七中胡凡

树型DP

- DP求解问题的基本条件
  - 最优化原理（即具有最优子结构性质）
  - 无后效性
  - 子问题的重叠性
- DP求解问题的三要素
  - 阶段
  - 状态
  - 决策
- 在树上做DP

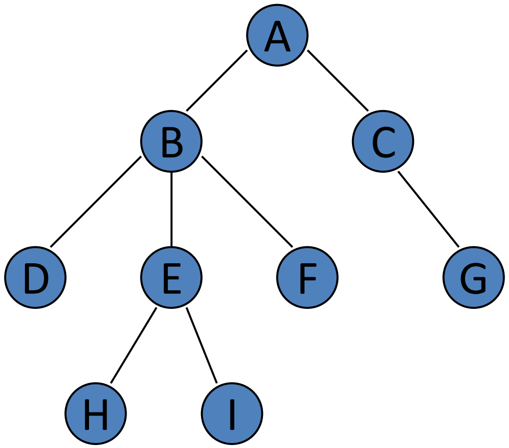
树型DP

- 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）的顺序作为DP的“阶段”。DP的状态表示中，第一维通常是节点的编号（代表以该节点为根的子树）。对于每个节点u，先递归在它的每个子节点v_i上进行DP，在回归时，从子节点v_i向节点u进行状态转移。大多数时候，我们采用递归的方式实现树型DP。

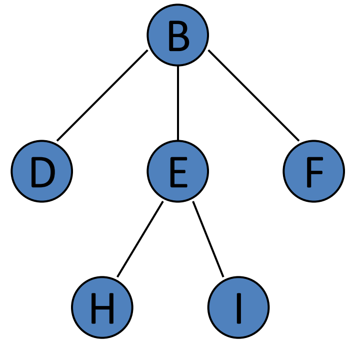
成都七中

目录

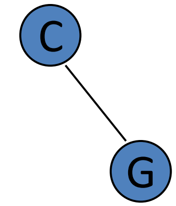
- 基础应用——子树和
- 基础应用——直径
- 覆盖类问题
- 树上背包问题



T

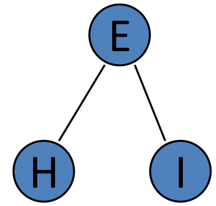


T1



T2

T11



T12

T13

树

树（tree）是树型结构的简称。它是一种重要的非线性数据结构。树——或者是一棵空树，即不含结点的树，或者是一棵非空树，即至少含有一个结点的树。在一棵非空树中，它有且仅有一个称作根（root）的结点，其余的结点可分为m棵（m≥0）互不相交的子树（即称作根的子树），每棵子树（subtree）又同样是一棵树。显然，树的定义是递归的，树是一种递归的数据结构。树的递归定义，将为以后实现树的各种运算提供方便。

树的基本术语

- 1、结点的度和树的度

每个结点具有的子树数或者说后继结点数被定义为该结点的度（degree）。所有结点的度的最大值被定义为该树的度。

- 2、分支结点和叶子结点

度大于0的结点称作分支结点或非终端结点，度等于0的结点称作叶子结点或终端结点。在分支结点中，又把度为1的结点叫做单分支结点，度为2的结点叫做双分支结点，其余以此类推。

成都七中

树的基本术语

- **3、孩子结点、双亲结点和兄弟结点**

每个结点的子树的根，或者说每个结点的后继，被习惯地称作该结点的孩子（child）或儿子，相应地，该结点被称作孩子结点的双亲（parent）或父亲。具有同一双亲的孩子互称兄弟（brothers）。每个结点的所有子树中的结点被称作该结点的子孙。每个结点的祖先则被定义为从树根结点到达该结点的路径上经过的所有结点。

- **4、结点的层数和树的深度**

树既是一种递归结构，也是一种层次结构，树中的每个结点都处在一定的层数上。结点的层数（1evel）从树根开始定义，根结点为第一层，它的孩子结点为第二层，以此类推。树中结点的最大层数称为树的深度（depth）或高度（height）。

树的基本术语

- 5、有序树和无序树

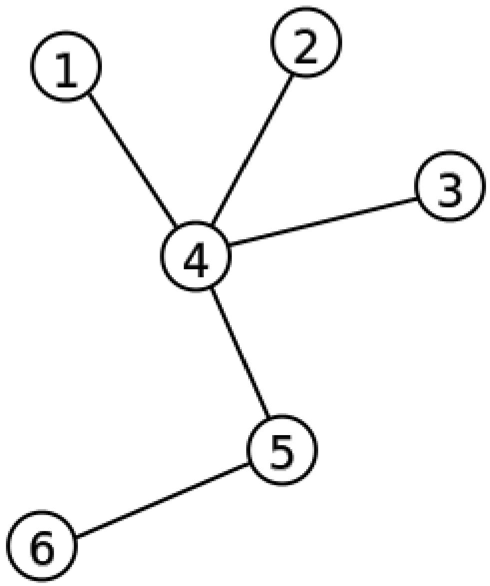
若树中各结点的子树是按照一定的次序从左向右安排的，则称之为有序树，否则称之为无序树。如对于一棵反映父子关系的家族树，兄弟结点之间是按照排行大小有序的，所以它是一棵有序树。以后若不特别指明，均认为树是有序的。

- 6、森林

森林是m（m≥0）棵互不相交的树的集合。例如，对于树中每个分支结点来说，其子树的集合就是森林。在图1的树T中，由A结点的子树所构成的森林为{T1，T2}，由B结点的子树所构成的森林为{T11，T12，T13}，等等。

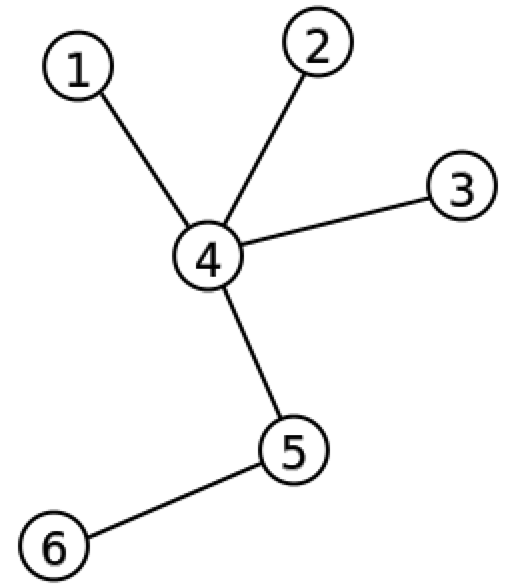
树

- 树是连通且无环的无向图
- 等价条件：
  1. 连通，且含有n个点、n-1条边
  2. 任意两点间恰有一条路径



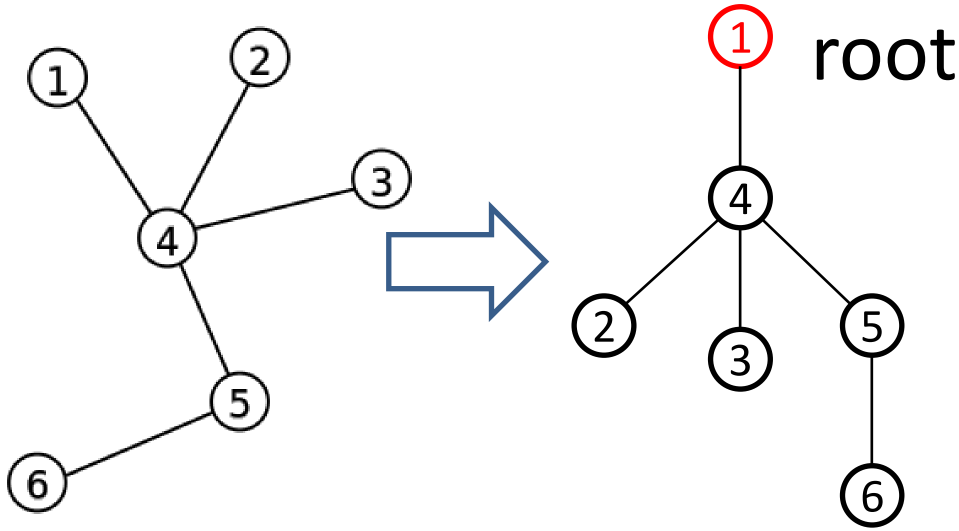
无根树

1. 连通，且含有n个点、n-1条边
2. 任意两点间恰有一条路径
3. 每个节点的地位是相同的



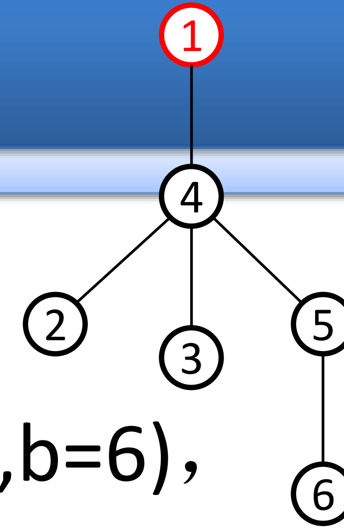
可以当做普通的无向图来处理

有根树

- 在定义的基础上、指定一个节点为“根”
- 

- 相比无根树，有根树更多地利用树的性质，组织结构更加清晰
- 树上的问题一般先转化成有根树再解决。

有根树的结构



- 描述有根树的结构：
- 若a在b到根的路径上(如a=1,b=3; a=4,b=6)，则称a是b的祖先、b是a的子孙
- 特殊地，若a是b的祖先、且a和b相邻(如a=1,b=4; a=5,b=6)，则称a是b的父亲(父节点)，b是a的儿子(子节点)
- 一个节点及其所有子孙节点组成一棵子树
- 深度：根据需要定义根的深度为0或1；每走过一条向下的边深度+1

有根树的存储

- 方法一：除了根没有父亲，所有节点都有唯一的父亲。记录每个节点的父节点即可。缺点是不能从根开始遍历整棵树
  应用：并查集
- 方法二：(同普通的无向图)用邻接表存储。从根开始dfs时得到每个点的父节点，除了父节点外相邻的节点就是子节点。
  应用：(很多OI题中)只知道树边的两个端点、不知道父子关系的情况

基础应用——子树和

- 每个节点u有一个权值 val[u]，统计节点的子树和
  - 令 sum[ ] 为子树和数组，叶子的 sum 即 val
  - 对于每个节点 u 以及它的子节点 v，$sum[u] = val[u] + \sum sum[v]$
- 应用
  - 统计子树大小（把所有节点的 val 设为 1 即可）
  - 找重心

成都七中

树的重心

- 给出一棵有n个节点的无根树。求该树的重心。
- 树的重心：若有一节点，以该节点为根的所有子树中最大子树的节点数最少，则该点就是这棵树的重心。或者说，删除这个节点后最大连通块的节点数最小，那么这个点就是树的重心。
- 一般的树只有一个重心，有些有偶数个节点的树，可能有两个重心。
- 求树的重心方法就是随意确定一个根节点，先把无根树转化为有根树，dfs求出所有点的子树的节点个数。如果有一点满足该点的子树的节点数的二倍大于等于总结点数(size[u]*2>=n)，并且该点的儿子都满足子树的节点数的二倍小于等于总结点数(size[son_u]*2<=n)，这个点就是树的重心。

```
#define N 42000
int n,a,b,next[N],to[N],head[N],num,size[N],father[N],ans=0;
void add(int u,int v){
	next[++num]=head[u];
	to[num]=v;
	head[u]=num;
}
void dfs(int u){
	size[u]=1;
	for(int i=head[u];i;i=next[i]){
		int v=to[i];
		if(father[u]!=v){
			father[v]=u;
			dfs(v);
			size[u]+=size[v];
		}
	}
	if(size[u]*2>=n&&!ans) ans=u;
}
```

```
main(){
	scanf("%d",&n);
	memset(head,0,sizeof(head));
	for(int i=1;i<n;++i){
		scanf("%d%d",&a,&b);
		add(a,b);
		add(b,a);
	}
	memeset(size,0,sizeof(size));
	father[1]=0;
	dfs(1);
	printf("%d",ans);
	return 0;
```

Codeforces 686D - kay and snowflake

- 给出一棵 n 个节点的有根树，有 q 个询问
- 每次询问以编号 v_i 为根的子树的重心

- 数据范围：$n \leq 3 * 10^5$, $q \leq 3 * 10^5$

成都七中

Codeforces 686D - kay and snowflake

- 暴力：
  - 如果询问次数较少，每询问一次就找一次重心

- 多次询问？
- 重心的性质
  - 把两棵树连接起来，新树的重心在原来两棵树的重心连线上（反证法证明）
  - 一棵树添加（或删除）一个节点，重心最多只移动一条边的距离（子树添边验证）

成都七中

Codeforces 686D - kay and snowflake

- 解法：
  - 根据性质2，可以在短时间内求出所有子树的重心
    - 叶结点的重心就是它自己
    - 对于一个节点 u 以及它的子节点 $v_1, v_2 \ldots$（记节点k的子树大小为 s_k ）
      - 如果不存在 $s_v > \frac{s_u}{2}$ ，则 u 子树的重心即为 u
      - 否则将重心设为 s_v 最大的那个子树的重心，并不断向上跳即可
  - 复杂度
    - 每条边最多被跳一次，$O(n)$

成都七中

Luogu P1122 – 最大子树和

- 给出一棵 n 个节点的无根树，每个节点有一个权值t_i（可能为负）。
- 要求保留树上一个连通块，使得留下的权值和最大
- 数据范围：$n \leq 16000$，答案在 int 范围内

成都七中

Luogu P1122 – 最大子树和

- 解法：
  - 无根树中选择连通块，任意定一个根对答案是没有影响的
  - 考虑到最优策略中，每个子树都是最优决策，所以具有最优子结构
  - 可以使用dp

Luogu P1122 – 最大子树和

解法：

- 定义 f[u]，表示在 u 的子树中，获得的最大收益
- 对于每个叶子节点 leaf，初始化有 $f[\text{leaf}] = \max(0, t_i)$
- 考虑转移，有 $f[u] = \sum f[v] + t_u$（其中 v 代表 u 的子节点），加上 t_u 是因为选了子树则必须选择当前点
- 如果 f[u] 为负，显然不如不选，即 $f[u] = \max(f[u], 0)$
- 答案就是所有 f[u] 的最大值
  - 注意答案不一定必须包含根，所以不能直接取 f [root]

Luogu P1122 – 最大子树和

```
long long ans = - ( 1LL << 60 ) , f[20005] ;
void dfs( int u , int fa ){
    f[u] = val[u] ;
    for( int i = head[u] ; i ; i = p[i].pre ){
        int v = p[i].to ;
        if( v == fa ) continue ;
        dfs( v , u ) ;
        f[u] += f[v] ;
    } f[u] = max( f[u] , 0LL ) ;
    ans = max( f[u] , ans ) ;
}
```

成都七中

基础应用——树的直径

- 树的直径显然可以两遍 dfs 求
- Dp做法？
  - 最长链一定是 从某个子树的根 向下沿伸两条链得到的（链长可以为0）
  - 考虑维护两个数组 f[ ] 和 g[ ]，分别记录 最长链 和 次长链（要求与最长链不同子树）
  - 那么答案就是 max($f[u] + g[u]$)
  - 考虑维护，对u的每个子节点v
    - if(f[v] + len[v] ≥ f[u]) g[u] ← f[u] , f[u] ← f[v] + len[v]
    - else if (f[v] + len[v] > g[u]) g[u] ← f[v] + len[v]

成都七中

树的直径

- **Description**

  给出一棵无根树，求树的直径，即树上两点之间的最长距离

  Input

  第一行为树的节点总数n，第二行至第n行每行两个整数a,b表示树上a点与b点之间有边

  Output

  输出树的直径

  Sample Input

  6

- 5 1

  1 4

  6 3

  2 6

  6 1

  Sample Output

  3

成都七中

树的直径

定义: 一棵树的直径就是这棵树上存在的最长路径。

求法: 两次dfs或bfs。第一次任意选一个点u进行dfs(bfs)找到离它最远的点v，此点就是最长路的一个端点，再以此点进行dfs（bfs），找到离它最远的点，此点就是最长路的另一个端点，于是就找到了树的直径。

证明: 假设此树的最长路径是从s到t,我们选择的点为u。反证法: 假设搜到的点是v。

1、v在这条最长路径上，那么dis[u,v]>dis[u,v]+dis[v,s],显然矛盾。

2、v不在这条最长路径上，我们在最长路径上选择一个点为po，则dis[u,v]>dis[u,po]+dis[po,t]，那么有dis[s,v]=dis[s,po]+dis[po,u]+dis[u,v]>dis[s,po]+dis[po,t]=dis[s,t],即dis[s,v]>dis[s,t],矛盾

树的直径

```
int num_edge,head[N],dis[N],n,a,b,y;
int add_edge(int from,int to){
    edge[++num_edge].next=head[from];
    edge[num_edge].to=to;
    head[from]=num_edge;
}
int dfs(int x){
    for(int i=head[x];i;i=edge[i].next)
        if(!dis[edge[i].to]){
            dis[edge[i].to]=dis[x]+1;
            dfs(edge[i].to);
        }
}
```

```
scanf("%d",&n);
for(int i=1;i<n;++i){
    scanf("%d%d",&a,&b);
    add_edge(a,b);
    add_edge(b,a);
}
memset(dis,0,sizeof(dis)
dfs(1);
for(int i=y=1;i<=n;i++)
    if(dis[i]>dis[y])
        y=i;
memset(dis,0,sizeof(dis)
dfs(y);
for(int i=y=1;i<=n;i++)
    if(dis[i]>dis[y])
        y=i;
printf("%d",dis[y]);
```

• POJ 2631 Roads in the North

- 题目大意：给你一棵树，求树上最远的两点距离为多少
- **Sample Input**
- 5 1 6
- 1 4 5
- 6 3 9
- 2 6 8
- 6 1 7
- **Sample Output**
- 22

POJ 2631 Roads in the North

找树的最长链，只需要求出以每个节点为根的子树中的最长链，取其中的最大值即可。

对于每个节点我们都要记录两个值：f[u]表示以u为根的子树中，u到叶子节点的距离最大值；g[u]表示以u为根的子树中，除距离最大值所在子树，u到叶子节点的距离最大值（即次大值）。假设v是u的儿子，则有：

1.若f[v]+dis[u][v]>f[u]，则g[u]=f[u]，f[u]=f[v]+dis[u][v]；

2.若f[v]+dis[u][v]>g[u]，则g[u]=f[v]+ dis[u][v]

扫描所有的节点，找最大的f[u]+g[u]的值。

POJ 2631 Roads in the North

```
int f[200005] , g[200005] , ans ;
void dfs( int u , int fa ){
    for( int i = head[u] ; i ; i = p[i].pre ){
        int v = p[i].to ;
        if( v == fa ) continue ;
        dfs( v , u ) ;
        if( f[v] + p[i].len >= f[u] )
            g[u] = f[u] , f[u] = f[v] + p[i].len ;
        else if ( f[v] + p[i].len > g[u] )
            g[u] = f[v] + p[i].len ;
    } ans = max( ans , f[u] + g[u] ) ;
}
```

覆盖类问题

- 相邻点选择限制（例：独立集）
- 相邻点覆盖限制（例：覆盖集，支配集）
- 链覆盖限制

- 前两种用贪心也可以做，但dp更易于理解

Luogu P1352 - 没有上司的舞会

- 【题目描述】
- 某公司有N个职员，编号为1~N。他们之间有从属关系，也就是说他们的关系就像一棵以董事长为根的树，父结点就是子结点的直接上司。现在有个周年庆宴会，宴会每邀请来一个职员都会增加一定的快乐指数Ri，但是呢，如果某个职员的上司来参加舞会了，那么这个职员就无论如何也不肯来参加舞会了。所以，请你编程计算，邀请哪些职员可以使快乐指数最大，求最大的快乐指数。

- 【输入格式】
- 第一行一个整数N。(1<=N<=6000)
- 第二行有N个数，每个数之间用一个空格隔开，第i个数表示i号职员的快乐指数Ri。(-128<=Ri<=127)
- 接下来N-1行，每行输入一对整数L,K。表示K是L的直接上司。

【输出格式】
- 输出最大的快乐指数。

成都七中

Luogu P1352 - 没有上司的舞会

- 解法：
  - 设根节点为root
  - 不难发现，只考虑父亲对儿子的限制 可以包含所有情况
  - 考虑如何用子节点的信息f[v]来推算f[u]
  - 状态信息还需要包括：某个节点是否被选
    - 不知道点的选择状态则无法转移

Luogu P1352 - 没有上司的舞会

- 解法：
  - 定义 f[u][0/1] 表示，不选中/选中 u 点时，获得的最大收益
  - 考虑转移：
    - $f[u][0] = \sum \max(f[v][0], f[v][1])$
    - $f[u][1] = \sum f[v][0]$
    - 注意 f[u][1] 需要加上 r_{u}（因为是将 u 点选中）
  - 答案即 $\max(f[root][0], f[root][1])$

- 把此题中所有点的权值设为1，即树的最大独立集

Luogu P1352 - 没有上司的舞会

```
vector <int> G[6010];
int n,x,y;
int dp[6010][2];
int r[6010];
int boss[6010];
void dfs(int u)
{
    for(int j=0;j<G[u].size();j++)
    {
        int v=G[u][j];
        dfs(v);
        dp[u][0]+=max(dp[v][1],dp[v][0]);
        dp[u][1]+=dp[v][0];
    }
    dp[u][1]+=r[u];
}
```

成都七中

皇宫看守（jzoj 2005）

- 太平王世子事件后，陆小凤成了皇上特聘的御前一品侍卫。
- 皇宫以午门为起点，直到后宫嫔妃们的寝宫，呈一棵树的形状；某些宫殿间可以互相望见。大内保卫森严，三步一岗，五步一哨，每个宫殿都要有人全天候看守，在不同的宫 殿安排看守所需的费用不同。可是陆小凤手上的经费不足，无论如何也没法在每个宫殿都安置留守侍卫。
- 编程任务：帮助陆小凤布置侍卫，在看守全部宫殿的前提下，使得花费的经费最少。

皇宫看守（jzoj 2005）

- **解法：**
- f[i][0]表示i节点在父节点可看到时，以i为根的子树需要安排的最少士兵数；
- f[i][1]表示i节点在子节点可看到时，以i为根的子树需要安排的最少士兵数；
- f[i][2]表示i节点安置士兵时，以i为根的子树需要安排的最少士兵数。
- f[i][0]=min{f[son][1],f[son][2]}+d
- f[i][1]=min{f[son][1],f[son][2]}+d
- d=min{f[son][2]−min(f[son][1],f[son][2])}
- f[i][2]=min{f[son][0],f[son][1],f[son][2]}+cost[i]

权值最少的覆盖集

成都七中

BZOJ1907 - 树的路径覆盖

- 给出一个 n 个节点的无根树，求出树的最小链覆盖
- 要求 不同的链 不能有公共点

- 数据范围：
  - t 组数据，$t \le 10$
  - 对于每组数据，$n \le 1 * 10^4$

成都七中

BZOJ1907 - 树的路径覆盖

- 解法：
  - 可以发现，如果直接通过搜索的方式找链是非常麻烦的，而且还要求最优
  - 尝试按照一定的次序寻找答案
    - 任意定根，答案不变
    - 对于一棵以 u 为根的子树，u 只有两种情况
      - 链的拐点（u与两个子节点相连）
      - 链的端点（u自成一条链 或 与一个子节点相连）
    - 从子节点的信息 可以得出 当前子树的信息
  - 可以使用dp

BZOJ1907 - 树的路径覆盖

- 解法：
  - 定义 f[u][0/1]，表示对于u的子树，当u是 端点/拐点的答案
  - 初值
    - 所有f[u][0/1]的初值为1（只考虑u点）
  - 将子节点信息合并，考虑转移
    - $f[u][1] = \min(f[u][1] + f[v][1],\ f[u][0] + f[v][0] - 1)$
    - $f[u][0] = \min(f[u][0] + \min(f[v][1], f[v][0]), tmp + f[v][0])$
    - $tmp \mathrel{+}= f[v][1]$

成都七中

BZOJ1907 - 树的路径覆盖

- 解法：
- $f[u][0] = \min(f[u][0] + \min(f[v][1], f[v][0]), tmp + f[v][0])$

对于第二个方程，可以发现：如果u和某个子节点v都是端点，那么连接起来肯定不劣

  - 和 u与父节点相连等价，但此情况占用了父节点一个度数，如果u存在两个及以上的兄弟端点，答案就会变得不优

因此可以进一步优化为贪心

树形背包

二叉苹果树（洛谷P2015）

- 有一棵苹果树，如果树枝有分叉，一定是分2叉（就是说没有只有1个儿子的结点）这棵树共有N个结点（叶子点或者树枝分叉点），编号为1-N,树根编号一定是1。我们用一根树枝两端连接的结点的编号来描述一根树枝的位置。现在这颗树枝条太多了，需要剪枝。但是一些树枝上长有苹果。给定需要保留的树枝数量，求出最多能留住多少苹果。

二叉苹果树（洛谷P2015）

【输入格式】

第1行2个数，N和Q(1<=Q<= N,1<N<=100)。 N表示树的结点数，Q表示要保留的树枝数量。

接下来N-1行描述树枝的信息。每行3个整数，前两个是它连接的结点的编号。第3个数是这根树枝上苹果的数量。每根树枝上的苹果不超过30000个。

【输出格式】

剩余苹果的最大数量。

【样例输入】

```
5 2
1 3 1
1 4 10
2 3 20
3 5 20
```

【样例输出】

```
21
```

二叉苹果树（洛谷P2015）

分析：我们需要保留的树枝数目为Q，保留节点j=Q+1。分三种情况讨论保留苹果的最大数量。

- 1.树根的左子树为空，全部保留右子树，右子树保留j-1个节点；
- 2.树根的右子树为空，全部保留左子树，左子树保留j-1个节点；
- 3.树根的两棵子树都非空，设左子树保留k个节点，则右子树保留j-1个节点。

设dp[i][j]表示i为根的树上保留j个节点的最大权值和。

二叉苹果树（洛谷P2015）

```
void dfs (int u,int father){
	for (int i=last[u];i!=0;i=pre[i]){
		int v=next[i],value=apple[i];
		if(v == father)continue;
		dfs(v,u);
		for(int j=m;j>=1;--j){
			for(int k=j;k>=1;--k){
				dp[u][j]=max(dp[u][j],dp[u][j-k]+dp[v][k-1]+value);
			}
		}
	}
}
```

选课（Codevs1378）

学校实行学分制，每门课都有固定的学分。学校开设了N（N<300）门的选修课程，每个学生可选课程的数量M是给定的。学生选修了这M门课并考核通过就能获得相应的学分。在选修课程中，有些课程可以直接选修，有些课程必须在选了其它的一些课程后才能选修。例如《数据结构》必须在选修了《高级语言程序设计》之后才能选修。我们称《高级语言程序设计》是《数据结构》的先修课。每门课的直接先修课最多只有一门。两门课也可能存在相同的先修课。每门课都有一个课号，依次为1，2，3，…。 例如:

课号	先修课号	学分
1	无	1
2	1	1
3	2	3
4	无	3
5	2	4

表中1是2的先修课，2是3、4的先修课。如果要选3，那么1和2都一定已被选修过。你的任务是为自己确定一个选课方案，使得你能得到的学分最多，并且必须满足先修课优先的原则。假定课程之间不存在时间上的冲突。

选课（Codevs1378）

Sample Input
7 4
2 2
0 1
0 4
2 1
7 1
7 6
2 2

Sample Output
13

成都七中

[HAOI2010]软件安装

- 有 n 个物品，每个物品有体积w_i以及价值v_i
  - 一个物品最多依赖另一个物品
  - 物品贡献价值，当且仅此物品及以上的整个依赖链都被选中
- 要选出一些物品，使得他们的体积和不超过m，并且价值尽可能大

- 数据范围：
  - $0 \le n \le 100$, $0 \le w_i \le m \le 500$, $0 \le v_i \le 1000$

成都七中

[HAOI2010]软件安装

- 解法：
  - 这是一道典型的树上背包问题
  - 不妨定义 f[i][j] 表示在以i号节点为根的子树中，选择体积和为j的物品，得到的最大价值
  - 考虑如何转移
    - 树上背包与普通背包略有不同
    - 每个节点都是一个泛化物品（即分配不同体积可能得到不同价值）
    - 不过大体是类似的

[HAOI2010]软件安装

解法：

- f[i][j] 表示在以i号节点为根的子树中，选择体积和为j的物品，得到的最大价值
- 对于每个节点，初始化$for(\ i = w[u]\ \rightarrow\ i = m\ )\ f[u][i] = v[u]$

转移如下：

- 对于点 u 以及它的子节点 v，将体积分配给已经处理过的部分和未处理部分
- $for(\ j = m\ \rightarrow j = 0\ )$
  - $for(\ k = 0\ \rightarrow k = m\ )$
    - » $if(\ j \geq\ k\ +\ w[u])\ f[u][j] = \max(\ f[u][j],\ f[v][k] + f[u][j-k]\ )$
- 转移中$j \geq k + w[u]$是为了保证选择子树的过程中必须包含根节点（保证依赖关系）

复杂度 $O(nm^2)$

成都七中

[HAOI2010]软件安装

解法：

- 上面忽略了一些问题
- 如果依赖关系形成了环，如何处理？
  - 显然，要么整个环都选，要么都不选，tarjan 缩点即可
- 如果依赖关系是一个森林，如何处理？
  - 建立$w_i = v_i = 0$的虚拟根，向所有树的根连边，从虚拟根开始dp即可

Luogu P1273 - 有线电视网

- 给出一棵 n 个节点的树，树上有 m 个叶结点，根为1号节点
  - 每个叶节点有权值，每条边有花费
- 如果选择一个叶结点，就需要选择该叶结点到根的所有边
  - 点的收益 和 边的花费 最多贡献一次
- 要求代价为正的情况下，选中尽量多的叶结点

- 数据范围：$\mathrm{m} < n \leq 3000$

成都七中

Luogu P1273 - 有线电视网

- 解法：
  - 此题存在资源分配限制，同时要求选中最多叶结点
  - 根据前一题的经验，我们仍然尝试使用dp解决这个问题
  - 定义状态 f[i][j] 表示在i的子树花费j元，选中叶结点个数的最大值
    - 是否可行？
    - 代价&收益 均不确定范围，第二维不知道大小

Luogu P1273 - 有线电视网

- 解法：
  - 注意到，叶结点少于 n (3000)个
  - 可以把dp中的节点数和花费换一个位置
    - 把节点数放进状态，把花费最优化
  - 定义 f[i][j] 表示在i的子树选j个叶节点，最大收益是多少
  - 剩下的部分就和上题相似
  - 计算节点的dp值，将处理过的子节点信息 与 新加入的子节点信息合并即可

成都七中

Luogu P1273 - 有线电视网

- 解法：
  - f[i][j] 表示在i的子树选j个叶节点的最大收益
  - 转移：
    - 对于当前节点u和一个新加入的子节点v，通过边e相连
    - $f[u][j] = \max(f[u][j], f[u][j-k] + f[v][k] - cost[e])$
    - j 的枚举上限是 已处理部分与v子树 的叶结点个数和
    - 答案即 k， k为最大的满足 $f[root][k] \geq 0$ 的数
    - 在每个节点合并的复杂度是（子节点个数 * 子数下叶结点个数）
    - 复杂度应该在$n^2 log$级别

[HAOI2015]树上染色

- 有一棵节点数为N的树，树边有边权
- 给你一个在0~N之内的正整数K，你要在这棵树中选择K个点，将其染成黑色，并将其他的N-K个点染成白色
- 将所有点染色后，你会获得黑点两两之间的距离加上白点两两之间距离的和的收益
- 问收益最大值是多少

- 数据规模：$N \le 2000$

成都七中

[HAOI2015]树上染色

- 解法：
  - 此题的难点在于如何计算贡献
  - 可以发现，使用上面类似的dp方法，只知道点的信息，是很难计算贡献的
    - 贡献还和边权有关，和点之间的距离有关
  - 那么我们不妨将贡献转化到边上
    - 一条边将树分成两部分
    - 这条边对答案的贡献即：（两侧黑点个数乘积＋两侧白点个数乘积）× 边权

成都七中

[HAOI2015]树上染色

- 解法：
  - 不难想到，定义 f[i][j] 表示i的子树内染了j个黑点，贡献最大是多少
  - 对于一个节点u和它的子节点v，考虑分配给子节点黑点的个数
  - 转移：
    - cnt为点对的数量，j是u的黑点数量，k是分配给v的黑点数量，val是边权
    - $cnt = k * (K - k) + (size[v] - k) * (N - K - (size[v] - k))$
    - $f[u][j] = \max(f[u][j], f[u][j-k] + f[\mathrm{v}][\mathrm{k}] + cnt * \mathrm{val}[\mathrm{e}])$
  - 答案即 f[root][K]

成都七中

谢 谢!