多角度思考 创造性思维

----运用树型动态规划的解题思路和方法的探析

关键字

树结构 动态规划 信息学奥赛

摘要

在近几年信息学竞赛中，需要运用树型动态规划解决的问题频繁出现，这些问题变化繁多、各类思想精华渗透其中，对选手分析问题的能力和解题创造性思维有着较高的要求，因此它在竞赛中占据了重要地位。

本文将分析近几年国际比赛、全国比赛中的树型动态规划问题，重点探讨几种树型动态规划问题的解法，并从这些问题的分析过程中，提炼出解决这类问题的思想方法——多角度思考，创造性思维。旨在论述解决问题的思维过程，而不仅仅是解题方法。

正文

信息学竞赛中通常会出现这样的问题：给一棵树，要求以最少的代价（或取得最大收益）完成给定的操作。有很多问题都是在树和最优性的基础上进行了扩充和加强，从而变成了棘手的问题。

这类问题通常规模较大，枚举算法的效率无法胜任，贪心算法不能得到最优解，因此要用动态规划解决。

和一般动态规划问题一样，这类问题的解决要考虑3步。

1、确立状态

几乎所以的问题都要保存以某结点为根的子树的情况，但是要根据具体问题考虑是否要加维，加几维，如何加维。

2、状态转移

状态转移的变化比较多，要根据具体问题具体分析，这也是本文例题分析的重点。

3、算法实现

由于树的结构，使用记忆化搜索比较容易实现。

由于模型建立在树上，即为树型动态规划。

顾名思义，树型动态规划就是在"树"的数据结构上的动态规划。

大部分动态规划题都是线性的，线性的动态规划有二种方向，既向前和向后，相应的线性的动态规划有二种方法，既顺推与逆推。而树型动态规划是建立在树上的，也相应的有两个方向：

1． 根—>叶：既根传递有用的信息给子节点，完后根得出最优解的过程。

2． 叶－>根：既根的子节点传递有用的信息给根，完后根得出最优解的过程。这类的习题比较的多，应用比较广泛

当然，还有一类问题，同时需要两种遍历方向，本文的第一题就属于这类。

问题描述

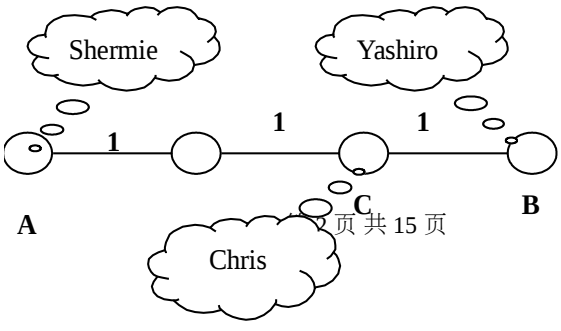
Chris 家的电话铃响起了，里面传出了 Chris 的老师焦急的声音："喂，是 Chris 的家长吗？你们的孩子又没来上课，不想参加考试了吗？"一听说要考试，Chris 的父母就心急如焚，他们决定在尽量短的时间内找到 Chris。他们告诉 Chris 的老师："根据以往的经验，Chris 现在必然躲在朋友 Shermie 或 Yashiro 家里偷玩《拳皇》游戏。现在，我们就从家出发去找 Chris，一但找到，我们立刻给您打电话。"说完砰的一声把电话挂了。

Chris 居住的城市由 N 个居住点和若干条连接居住点的双向街道组成，经过街道 x 需花费 T_x 分钟。可以保证，任两个居住点间有且仅有一条通路。Chris 家在点 C，Shermie 和 Yashiro 分别住在点 A 和点 B。*Chris* 的老师和 *Chris* 的父母都有城市地图，但 *Chris* 的父母知道点 *A*、*B*、*C* 的具体位置而 *Chris* 的老师不知。

为了尽快找到 Chris，Chris 的父母会遵守以下两条规则：

- 如果 A 距离 C 比 B 距离 C 近，那么 Chris 的父母先去 Shermie 家寻找 Chris，如果找不到，Chris 的父母再去 Yashiro 家；反之亦然
- Chris 的父母总沿着两点间唯一的通路行走。

显然，Chris 的老师知道 Chris 的父母在寻找 Chris 的过程中会遵守以上两条规则，但由于他并不知道 A，B，C 的具体位置，所以现在他希望你告诉他，最坏情况下 Chris 的父母要耗费多长时间才能找到 Chris？



例如上图，这座城市由 4 个居住点和 3 条街道组成，经过每条街道均需花费 1 分钟时间。假设 Chris 住在点 C，Shermie 住在点 A，Yashiro 住在点 B，因为 C 到 B 的距离小于 C 到 A 的距离，所以 Chiris 的父母会先去 Yashiro 家寻找 Chris，一旦找不到，再去 Shermie 家寻找。这样，最坏情况下 Chris 的父母需要花费 4 分钟的时间才能找到 Chris。

输入输出

输入文件 hookey.in 第一行是两个整数 N（$3 \le N \le 200000$）和 M，分别表示居住点总数和街道总数。以下 M 行，每行给出一条街道的信息。第 $i+1$ 行包含整数 U_i、V_i、T_i（$1 \le U_i, V_i \le N$，$1 \le T_i \le 1000000000$），表示街道 i 连接居住点 U_i 和 V_i，并且经过街道 i 需花费 T_i 分钟。街道信息不会重复给出。

输出文件 hookey.out 仅包含整数 T，即最坏情况下 Chris 的父母需要花费 T 分钟才能找到 Chris。

分析

问题抽象

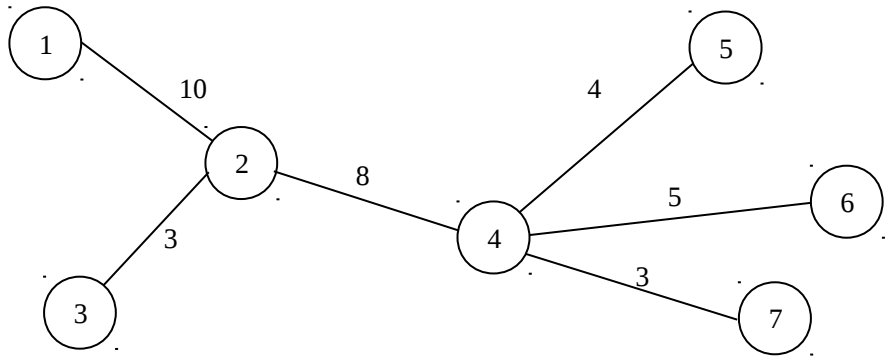
本题题意很明确，即在一棵树中，每条边都有一个长度值，现要求在树中选择 3 个点 X、Y、Z，满足 X 到 Y 的距离不大于 X 到 Z 的距离，且 X 到 Y 的距离与 Y 到 Z 的距离之和最大，求这个最大值。

粗略分析

很显然，该题的结构模型是一棵树，而且数据量很大，很容易把这题的方法向在树形结构上使用动态规划上靠拢。考虑任意节点 a 时，很容易发现，如果以这个点作为题目中要求的节点 Y，那么只需要求出离这点最远的两个节点即可。但是在树形结构上，计算出离某个节点最远的两个节点需要 $O(n)$ 的复杂度，而我们并不知道哪个点是答案中的节点 Y，

所以必须枚举这个点，这样一来，时间复杂度就成了 $O(n^2)$，在 N=200000 时会严重超时，因此这个方法是不可行的。

枚举 Y 点的话，会超时，是否就无法加上枚举的思想呢？可以多尝试一些思路。观察这样一棵树：



可以把点 3 当作点 X，点 1 当作点 Y，点 6 当作点 Z，这样就可以得到最大值了。因为|*XY*|=|*YX*|，故只需考虑 YX 和 YZ 这两条路就可以了。在图中来观察这两条路，可以发现分别是这样走的：YX：1→2→3，YZ：1→2→4→6。来比较这两条路的行程，可以发现，都是从 1 先到 2，然后再分开走各自的路，而且满足从 2 以后的经过的节点，没有重复的节点。为了方便，我们形象地将点 2 称为分叉点。如果枚举分叉点，能不能得到一个高效的算法呢？来尝试分析这种想法。

枚举分叉点

将某个点 a 当作分叉点时，以其为根构造一棵树，对节点 Y，就有两种情况：Y 就是节点 a；Y 在 a 的某个孩子节点的子树上。对于情况 1，可以把它转化为情况 2，只需给 a 加一个空的孩子节点，认为它和 a 之间的距离是 0 即可。既然 a 是分叉点，那么 X 和 Z 就不能在同一棵子树上，X 和 Y，Y 和 Z 也不能在同一棵子树上。题目中要求的是使|*XY*|+|*YZ*|最大，也就是要求 2|Ya|+|Za|+|Xa|最大。至此，思路已完全明确，对于以 a 为分叉点的情形，只需求出到 a 的最远的 3 个点，而且这 3 个点分别处于 a 的 3 棵不同的子树之中。如果采用枚举分叉点的方法的话，每次都需要 $O(n)$ 的计算才行，时间复杂度就又成了 $O(n^2)$。

两次遍历

这里，需要改变一下思路。以点 1 为根，计算出要求的值后，不去枚举其它的节点，而把这棵树再遍历一遍，进行一次 BFS，深度小的先访问，深度大的后访问，就保证了访问到某一个节点的时候，其父亲节点已经被访问过了。假设我们现在访问到了点 a，我们现在要求的是距点 a 的 3 个最远的点，且这 3 个点到 a 的路径上不经过除 a 外的相同节点。显然，这 3 个点要么位于以 a 为根的子树中，要么位于以 a 为根的子树外。如果在以 a 为根的子树外，那么是一定要通过 a 的父亲节点与 a 相连的。至此，思路逐渐清晰起来。此次遍历时，对于点 a，检查其父亲节点，只需求出到其父亲节点的最远的，且不在以 a 为根的子树中的那点即可，再与第一次遍历求得的值进行比较，就可以求出以该点为分叉点时，|*XY*|+|*YZ*|

的最大值了。具体方法为，每个点记录最大的两个值，并记录这最大的两个值分别是从哪个相邻节点传递来的。当遍历到其某个孩子节点时，只需检查最大值是否是从该孩子节点传递来的，如果是，就取次大值，如果不是，就可以取最大值。这样一来，该算法的时间复杂度和空间复杂度都为 $O(n)$，是个非常优秀的算法。

注意

这里提醒大家注意一点，计算过程中的值和最后的结果可能会超过长整型的范围，所以这里需要使用 int64 或者 double 类型。

对于树必须进行两次遍历，才能求得最终的最大值。该例题的分析中提出了分叉点的想法，是比较具有创造性的，需要从多个角度思考。因此，在平时的练习中，当对一道题目束手无策时，可从多个角度思考，创造新思维，使题目得到解决。此题遍历两次的方法具有普遍性，在许多题目中都可以得到应用：记录最大的两个值，并记录是从哪个节点传递来的思想方法。这种遍历两次和记录最大的多个值的思想是值得学习的，也是动态规划在树上进行使用的经典方法。

本题的树型动态规划复杂度是线形的，但是也有一部分问题，在线形的时间内无法出解。这类问题的变化更多，从状态的确立到状态的转移，都对思考角度有着更高的要求。这里先举 2 个例子来说明。

问题描述

几乎整个 Byteland 王国都被森林和河流所覆盖。小点的河汇聚到一起，形成了稍大点的河。就这样，所有的河水都汇聚并流进了一条大河，最后这条大河流进了大海。这条大河的入海口处有一个村庄——名叫 Bytetown。

在 Byteland 国，有 n 个伐木的村庄，这些村庄都座落在河边。目前在 Bytetown，有一个巨大的伐木场，它处理着全国砍下的所有木料。 木料被砍下后，顺着河流而被运到 Bytetown 的伐木场。Byteland 的国王决定，为了减少运输木料的费用，再额外地建造 k 个伐木场。这 k 个伐木场将被建在其他村庄里。这些伐木场建造后，木料就不用都被送到 Bytetown 了，它们可以在 运输过程中第一个碰到的新伐木场被处理。 显然，如果伐木场座落的那个村子就不用再付运送木料的费用了。它们可以直接被本村的伐木场处理。

注意：所有的河流都不会分叉，也就是说，每一个村子，顺流而下都只有一条路——到 bytetown。

国王的大臣计算出了每个村子每年要产多少木料，你的任务是决定在哪些村子建设伐木场能获得最小的运费。其中运费的计算方法为：每一块木料每千米 1 分钱。

编一个程序：

1．从文件读入 村子的个数，另外要建设的伐木场的数目，每年每个村子产的木料的块数以及河流的描述。

2．计算最小的运费并输出。

输入输出

第一行 包括两个数 n（2<=n<=100），k（1<=k<=50,且 k<=n）。n 为村庄数，k 为要建的伐木场的数目。除了 bytetown 外，每个村子依次被命名为 1，2，3..... n，bytetown 被命名为 0。

接下来 n 行，每行包涵 3 个整数

wi——每年 i 村子产的木料的块数 0020（0<=wi<=10000）

vi——离 i 村子下游最近的村子（或 bytetown）（0<=vi<=n）

di——vi 到 i 的距离(km)。（1<=di<=10000）

输出包含一行，即最小的运费。

保证每年所有的木料流到 bytetown 的运费不超过 2000，000，000 分

50%的数据中 n 不超过 20。

分析

问题抽象

本题的题意很明确，即建立 k 个伐木厂，使得把所有木材运送到最近的祖先伐木厂的费用最小。

由于题目给定的是一棵树，数据规模又比较大，很容易联想到树型动态规划。

状态的确立

首先必须有的是当前点及以当前点为根的子树中，一共建立了多少伐木厂，但是这显然是不够的，因为这个状态中没有任何与伐木厂位置相关的信息。因此我们还需要再加一维。加上有关伐木厂的位置的信息。综上，状态定为用 $f(from, to\_, kleft)$ 表示把以 from 为根的子树中建立 kleft 个伐木厂，把木材全部运送到最近的祖先伐木厂，所需要的费用，并且 from 有一个祖先伐木厂为 to_。（注意到这里 to_仅仅是 from 的祖先伐木厂，而未必是 from

的最近祖先伐木厂，这是为什么呢？）

这个状态虽然是 3 维的，但是非常清晰，为我们写状态转移方程提供了很大的方便。

状态的转移

状态转移分 2 种情况讨论：在 from 建立伐木厂和不在 from 建立伐木厂。

在 from 建立伐木厂：

即分配 kleft 个伐木厂给 from 的子结点，使得费用最小。这分配的过程，也就相当于背包问题。

$h1(i,k)=h1(i-1,k-j)+f(son,from,j)$ h1 是用来解背包问题的临时数组，son 是 from 的儿子结点。

不在 from 建立伐木厂：

依然是分配 kleft 个伐木厂给 from 的子结点，使得费用最小。不过不同的是，权不是 $f(son,from,j)$ 而是 $f(son,to\_,j)$，因为 from 处不一定有伐木厂。

$h2(i,k)=h2(i-1,k-j)+f(son,to\_,j)$ h2 是用来解背包问题的临时数组，son 是 from 的儿子结点。

最后 $f(from,to\_,kleft)=\min\{h1(son\_num,kleft),h2(son\_num,kleft)\}$

由状态转移也可以看出把 to_设成 from 的祖先伐木厂，而不是 from 最近的祖先伐木厂的好处：转移简便。适当放宽对状态的限制，可以简化转移。

效率分析

由于状态是 3 维的，而转移时需要枚举 k、son、和 j，j 的总数是 n，son 的总数也是 n，所以虽然要枚举 3 个，但是总的运算量是一定的，时间复杂度为 $O(n^3k)$。

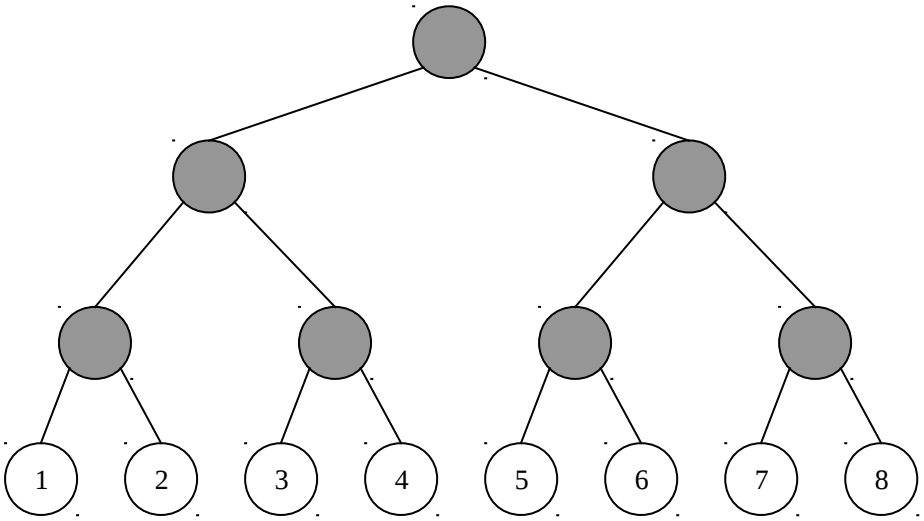
本题的树型动态规划的解法与上题有不同，它是多维的，要通过分析建立状态，而在兄弟结点之间又要通过类似背包问题的思想进行第二次动态规划，而不是单一的求取最大值或求和等操作，难度又上了一层。本题所运用的思想，如动态规划加一维、兄弟结点之间进行第二次动态规划等，在信息学奥赛这都有广泛的运用。

由于树结构的特殊性，使得一些时空复杂度看上去比较高的问题，经过均摊分析，降低了时空复杂度，从而得到解决 ◦时空复杂度的均摊分析在树型结构中有着广泛应用，来看这样一个例题。

问题描述

网络已经成为当今世界不可或缺的一部分。每天都有数以亿计的人使用网络进行学习、科研、娱乐等活动。然而，不可忽视的一点就是网络本身有着庞大的运行费用。所以，向使用网络的人进行适当的收费是必须的，也是合理的。

MY 市 NS 中学就有着这样一个教育网络。网络中的用户一共有 2^N 个，编号依次为 1, 2, 3, …, 2^N。这些用户之间是用路由点和网线组成的。用户、路由点与网线共同构成一个满二叉树结构。树中的每一个叶子结点都是一个用户，每一个非叶子结点（灰色）都是一个路由点，而每一条边都是一条网线（见下图，用户结点中的数字为其编号）。



MY 网络公司的网络收费方式比较奇特，称为"配对收费"。即对于每两个用户 i, j ($1 \le i < j \le 2^N$) 进行收费。由于用户可以自行选择两种付费方式 A、B 中的一种，所以网络公司向学校收取的费用与每一位用户的付费方式有关。该费用等于每两位不同用户配对产生费用之和。

为了描述方便，首先定义这棵网络树上的一些概念：

祖先：根结点没有祖先，非根结点的祖先包括它的父亲以及它的父亲的祖先；

管辖叶结点：叶结点本身不管辖任何叶结点，非叶结点管辖它的左儿子所管辖的叶结点与它的右儿子所管辖的叶结点；

距离：在树上连接两个点之间的用边最少的路径所含的边数。

对于任两个用户 i, j $(1 \le i < j \le 2^N)$，首先在树上找到与它们距离最近的公共祖先：路由点 P，然后观察 P 所管辖的叶结点（即用户）中选择付费方式 A 与 B 的人数，分别记为 nA 与 nB，接着按照网络管理条例第 X 章第 Y 条第 Z 款进行收费（如下表），其中 $F_{i,j}$ 为 i 和 j 之间的流量，且为已知量。

I 付费方式	J 付费方式	nA 与 nB 大小关系	付费系数 k	实际付费
A	A	$nA<nB$	2	$k * F_{i,j}$
A	B		1	
B	A		1	
B	B		0	
A	A	$nA \ge nB$	0	
A	B		1	
B	A		1	
B	B		2	

由于最终所付费用与付费方式有关，所以 NS 中学的用户希望能够自行改变自己的付费方式以减少总付费。然而，由于网络公司已经将每个用户注册时所选择的付费方式记录在案，所以对于用户 i，如果他/她想改变付费方式（由 A 改为 B 或由 B 改为 A），就必须支付 C_i 元给网络公司以修改档案（修改付费方式记录）。

现在的问题是，给定每个用户注册时所选择的付费方式以及 C_i，试求这些用户应该如何选择自己的付费方式以使得 NS 中学支付给网络公司的总费用最少（更改付费方式费用+配对收费的费用）。

输入输出

输入文件中第一行有一个正整数 N。

第二行有 2^N 个整数，依次表示 1 号，2 号，…，2^N 号用户注册时的付费方式，每一个数字若为 0，则表示对应用户的初始付费方式为 A，否则该数字为 1，表示付费方式为 B。

第三行有 2^N 个整数，表示每一个用户修改付费方式需要支付的费用，依次为 $C_1, C_2, \ldots, C_M$。$(M=2^N)$

以下 2^N-1 行描述给定的两两用户之间的流量表 F，总第$(i+3)$行第 j 列的整数为 $F_{i,j+i}$。（$1 \le i < 2^N$，$1 \le j \le 2^N - i$）

所有变量的含义可以参见题目描述。

你的程序只需要向输出文件输出一个整数，表示 NS 中学支付给网络公司的最小总费用。（单位：元）

数据规模和约定

40%的数据中 $N \leq 4$；

80%的数据中 $N \leq 7$；

100%的数据中 $N \leq 10$，$0 \leq F_{i,j} \leq 500$，$0 \leq C_i \leq 500\,000$

分析

问题转化

由这题的数据范围可以联想到树型动态规划。但是一时无法下手，因为收费规则是配对的，仔细观察发现，这题的收费规则很特殊：系数分别为 2、1、1、0 和 0、1、1、2。

把配对收费规则转化一下，对于任两个用户 I,j，设它们的最近公共祖先(LCA)是 p

如果在 p 上 na<nb，则 i,j 中谁是 A 谁支付 F[i,j]的费用

如果在 p 上 na≥nb，则 i,j 中谁是 B 谁支付 F[i,j]的费用

这样的转化让点对交费转移到单点交费，为 dp 提供了前提，且最后的总费用不变。

整个模型是一棵完全二叉树，设计树型动态规划。

状态的确立

状态的设计应该与 i 点中 A 的个数有关，但仅仅这样是不够的，因为这些点是按 A 收费还是 B 收费还与以它每个祖先为根的子树中，A 多还是 B 多有关。因此，这也是需要记录的。

用 $f(i, j, k)$ 在点 i 所管辖的所有用户中，有 j 个用户为 A，在 i 的每个祖先 u 上，如果 na<nb，则标 0，否则标 1，这个数列可以用二进制表示，用 k 记录，在这种情况下的最少花费。

状态的转移

状态转移方程，其实就是将 j 个用户分配给 i 的左右子节点，并更改 k

$$f(i, j, k) = \min\{f(left, u, k+s) + w1, f(right, j-u, k+s) + w2\}$$

k+s 表示在数列上加入节点 i 的标号，left 和 right 分别表示 i 的左右子节点

边界

当 i 是叶节点时，可以根据 k 算出 i 与其它用户配对收费时所要交的费用，再根据 i 的初始情况加上 AB 的转化费用。

效率分析

设 $m=2^n$ 粗略看来，空间复杂度是 $O(m^3)$，时间复杂度是 $O(m^4)$，对于本题的数据可以同时超空间和超时间了。

不过这只是很粗略的分析，这些状态中有很多是取不到的。

把根节点看做第 0 层，叶节点就是第 n 层，对于任意的第 i 层，A 用户的个数最多只有 2^{n-i}，而祖先结点只有 i 个，即 k 最大只有 2^i，把 $f(i,j,k)$ 的后两维合并成一维，空间复杂度其实只有 $O(m^2)$。

再看时间复杂度，对于第 i 层，有 2^i 个节点，每个节点的状态数是 O(m)的，状态转移的复杂度是 $O(2^{n-i})$，所以每层的复杂度是 $O(m^2)$。

总共有 n 层，所以状态转移的时间复杂度是 $O(m^2 n)$。

至此空间复杂度和时间复杂度都降到了可以承受的地步，在具体实现时，要用记忆化搜索和二进制的位运算。

本题除了树型动态规划的设计之外，还有 2 个难点：一个是对收费规则的转化，另一个是对时间复杂度的分析。第一点不是本文讨论的重点，第二点却在树型动态规划中有着广泛的运用。很多问题，粗略的分析可能会超时，但是仔细分析之后，也许会发现时间复杂度比粗略的分析少一维甚至更多。因此，要学会对时间复杂度进行均摊分析，如果一个问题，想到了正确的方法，却因为对时间效率分析的错误而放弃，实在是太可惜了。

上 2 题的难度主要体现在状态的转移和效率分析，状态基本上是很容易确定的。但是还有一类问题，如果按照固有的思路建立状态，是无法写成状态转移方程的。这就需要我们从多角度分析问题，需要我们的创造性思维，不断挖掘问题的特殊性。这里举一个例子。

问题描述

S 国有一个山洞，它由 n 个房间和若干走廊组成。对于任意的两个房间，总有唯一的路径相连。A 在这些房间中的一个藏了很多财宝，但是他不告诉 B 到底是哪一个房间。B 很想知道财宝藏在哪里，所以 B 猜若干次，每次猜一个房间有财宝。如果 B 猜的对，那么 A 告诉 B 他猜对了；如果 B 没猜对，那么 A 告诉 B 从他猜的房间开始经过哪一条走廊可以找到宝藏（即只告诉 B 从他猜的房间开始的第一条走廊）。

写一个程序，从 jas.in 中读入山洞的描述，计算 B 在最坏情况下为了知道宝藏位置所需要猜的最少次数，然后将答案输出到 jas.out。

输入输出

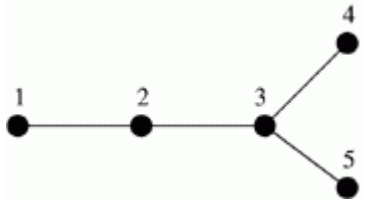
在输入数据的第一行有一个正整数 n(*1<= n <= 50000*)，表示山洞的数目。山洞从 1 到 n 标号。在接下来的 n-1 行中，每行包含两个正整数 a 和 b(*1<= a,b <= n*)，用一个空格分隔，表示在房间 a 和房间 b 之间有一条走廊。

你的程序应该只输出一个整数，即 B 在最坏情况下为了知道宝藏位置所需要猜的最少次数

样例

jas.in

```
5
1 2
2 3
4 3
5 3
```



jas.out

```
2
```

分析

首先容易想到应该和树的 dp 有关。

状态的确立

需要记录的，不仅是 i 为根的子树，最少需要 z_i 次询问；还需记录另一些参数！

每次询问过后，我们会被告知宝藏在某个连通块。如果宝藏所在的连通块，不包括 i 这个结点，那么 z_i 次询问显然足够。否则，如果宝藏仍然在子树上，却不一定需要 z_i -1 次询问，可能出现浪费。因此我们让询问的位置尽可能靠近 i，就是说如果连通块包括 i，将需要尽可能少的多余询问。这就是突破口！将前面的 z_i 定义为 $z_{i,0}$ 。

i 结点为根的子树，若第一次讯问后，宝藏所在连通块包含 i，如果我们知道宝藏仍在子树上，此时至少还需要 $z_{i,1}$ 次询问类似地，第二次讯问后，如果宝藏还在子树上，最少需要 $z_{i,2}$ 次询问..... 就是说， $z_{i,1}$ 尽可能小。当 $z_{i,1}$ 最小的时候， $z_{i,2}$ 尽可能小.....

注意到可能出现 $z_{i,1}$ = 5, $z_{i,2}$ = 2 和 $z_{i,1}$ = 4, $z_{i,2}$ = 3 的两种情况， 但是我们选择后者，因为 $z_{i,1}$ 更小，关于这一点后文中讲给出具体讲解。

状态的转移

为了具体理解这个突破口，我们看两个例子。

例如：对当前 A 为根的子树，它有两个儿子 i 和 j

$z_{i,0}$ =5, $z_{i,1}$ =3, $z_{i,2}$ =1; $z_{j,0}$ =4, $z_{j,1}$ =2

我们可以先在 i 子树上询问，如果宝藏在 i 内部，那么 $z_{i,0}$ = 5 次讯问足以。

否则转问 j 子树，如果在 j 内部，需要 $z_{j,0}$ = 4 次，再加上前面一次共 5 次足以。

否则转问 i 子树，如果在 i 内部，需要 $z_{i,1}$ = 3 次，再加上前面 2 次共 5 次。

.....

总知共须 5 次即可完成询问，而不需要在 A 处询问。

例如：对当前 A 为根的子树，它有两个儿子 i 和 j

$z_{i,0}$ =4, $z_{i,1}$ =2, $z_{i,2}$ =1; $z_{j,0}$ =2

先在 i 子树上询问，如果在 i 内部，我们需要 $z_{i,0}$ = 4 次讯问。

否则，我们在 A 处作一次询问，可以知道究竟在 i 内部还是 j 内部。

这时候，无论在哪一个内部，都只需要 $z_{i,1} = z_{j,0} = 2$ 次讯问，共 4 次。

因此 4 次讯问足够，这时候需要在 A 处作一次询问，但是并不是一开始就问 A。

算法的实现

以上是想到这个算法的缘由，再直接说说算法的实现。

程序对每个节点开设一个长度为 $\log n$ 的表（可以证明 $\log n$ 次询问足够），$Z_{i,0..\log n}$，对叶节点，显然只有 $Z_{i,0} = 1$，其余均为 0。

对于当前结点 A，把所有子节点 i 的 Z_i 累加（$\log n$ 维向量的加法）存入 Z_A，对于 Z_A 序列中，末尾的 0 和 1，我们可以按照前面第一个例子的方法， 哪一个子树这个位置是 1，我就对那个子树作一次询问。设 x 是最大的满足 $Z_{A,x}$ >1 的数，那么可以在剩下 y(y>x)次询问的时候，询问一次根，这样可以保证 y 次询问足够。

因而在所有的 y(y>x)中，寻找到最小的 y 使得 $Z_{A,y}$ =0。将 $Z_{A,y}$ 修改为 1（此时询问一次 A）所有的 $Z_{A,0..y-1}$ 修改为 0（倘若这次对 A 的询问，返回说宝藏在通往父节点的路上，那么就肯定不在 A 为根的子树上了，不需要多余的询问）这就是 A 节点的 Z 序列。

前面提到了(5,2)和(4,3)的同等情况下选择后者的问题。将 Z_A 按照二进制数表示， Z_0 是最低位，那么我们是在求 Z_A 的最小值。这就使(5,2)=(100100) > (011000)=(4,3)的更形象的表示。为什么要选择最小呢？比如一个 Z_i 如果稍大些，刚好和另一个 Z_j 错开，构成全

1 状态，后来发现如不修改 Z_i 会出现 0，那样不如直接在根节点询问，结果肯定更优。

本题的难度比上三题都大。它的状态的确立不是显而易见的。这类问题的解决，依赖于平时的积累和总结。不仅仅要从理性角度分析，还可以去感性的感受一下，由例子入手，从而解决复杂的问题。

本文总结

本文 4 个例题从不同方面阐述了树型动态规划的解题方法，如：对于树进行两次遍历；分叉点的思路；记录最大的两个值，并记录是从哪个节点传递来的思想方法；多维动态规划；兄弟结点之间通过类似背包的思想进行第二次动态规划；对复杂度的均摊分析等，这些方法在比赛中有着广泛的运用。

在此过程中，我认识到：面对陌生的问题，一方面要深入分析问题的属性，挖掘问题的本质，另一方面要多角度思考，抓住问题的特殊性，从而创造出正确的解题思路。

不过，这类问题变化繁多，本文只是提供了一些解题的方法和思路，抛砖引玉，重要的是平时的不断积累和总结。

参考资料

《实用算法的分析与程序设计》 吴文虎、王建德 电子工业出版社

《算法与数据结构》 傅清祥、王晓东 电子工业出版社

《数据结构》（第二版） 严蔚敏、吴伟民编著，清华大学出版社

"*Introduction to algorithms*" Thomas H. Cormen Charles E. Leiserson Ronald L. Rivest Clifford Stein ，Higher Education Press The MIT Press