浅谈数据的合理组织

四川省绵阳南山中学　何森

【摘要】

信息学是一门高深的学科，它正在高速的发展。随着信息学的发展，其题目中的关系也变得越来越错宗复杂，给我们解题带来困难。对数据进行合理地组织正是我们面对上述题目时的一种有效手段。

本文用几个经典例题从数据的结构和顺序两个方面进行合理组织，达到优化模型或是提升算法效率的目的。介绍了“合理组织数据”在信息学中建立模型和优化算法方面的一些应用，例题包含了动态规划、数据结构、图论类型的题目。目的在于引起读者对于数据的合理组织的关注，并在今后的解题中能积极并灵活地运用这一手段。

【关键字】

组织数据　数据结构　动态规划　图　树　序列

【正文】

【引言】

一个简单的例子：

给出 N 个数字(数字会比较大)，然后给出一些询问，询问一个数字有没有在给出的 N 个数字当中。

当然我们有很多已知的办法：

HASH 表、TRIE、预排序+二分查找……

这些算法都是通过对数据进行合理的组织而起到了减少工作量的作用。

不同的是 HASH 表和 TRIE 是利用数据形式的重新组织，而预排序＋二分查找是通过对数据顺序的重新组织来达到优化算法的目的的。我们组织数据，主要就是通过从“形式”和“顺序”这两个角度来考虑。事实上，这两个方面在实际运用中往往不是独立的，通常需要联合运用。

我们已经学习了很多经典的数据结构，它们都是合理组织数据的表现。在优化算法中有很好表现。对数据组织的合理化，不仅在我们设计算法时能起到优化程序效率的作用，有时，我们在建立解题模型时，合理地组织数据可能给我们提供新的思考角度，从而优化解题模型，例一就是这样的一个例子。

［例一］金明的预算方案及其加强版

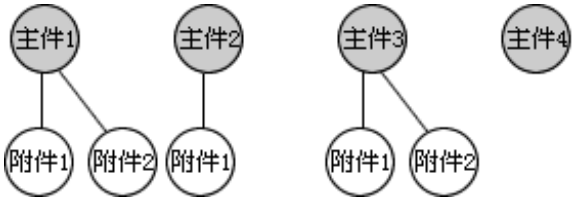
金明的预算方案

【题意描述】

给出 N 个物品，每个物品都有一个权值(<50000)和一个价格(<10000)。我们称可以直接被购买的物品为主件，称不能被直接购买的物品为附件，附件只有

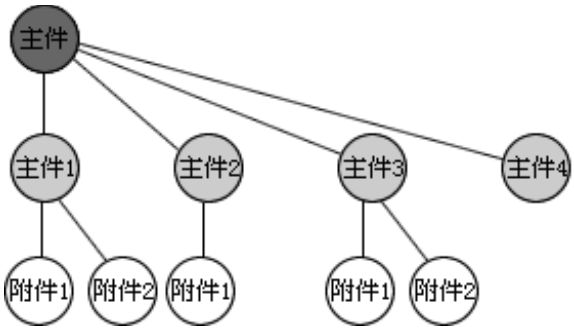
当其主件被购买了才能被购买，一个主件最多有两个附件，附件没有下一级附件。
任务　购买一些物品，总价格不超过 M，使得被购买的物品的权值之和最大。
N<3200 M<60

【简要分析】
我们很容易联想到经典的动态规划之 0-1 背包问题。
但是题目与背包却有一些差别：附件不能被直接购买。
【对数据的初步组织】
主件与附件之间是树形的关系。组织一下数据，如下图：



（图 1）

如图所示：
主件 1 没有附件，主件 2 有两个附件，主件 3 只有一个附件。
【数据组织方案一】
假设我们忽略数据的特殊性，单从树结构考虑，我们容易想到的一个算法是：
给所有主件加上一个“级超主件”，把原来的所有主件都变成“超级主件”的附件，如下图：



(图 2)

【算法一】
这样，在这棵树上，我们可以设计一个动态规划算法：
定义：
cost[a]表示 a 节点所代表的物品的价格
score[a]表示 a 节点所代表的物品的得分
状态 f[a][b]表示以节点 a 为根的子树，总共花费不超过 b 元的最多得分。
状态转移方程：
f[a][b]=Max{f[c1][b1]+f[c2][b2]+f[c3][b3]...f[ck][bk]}+score[a];
其中 ci 为 a 的子节点；∑bi=b-cost[a];
这样枚举的效率显然不高！
我们可以用左儿子右兄弟表示法来表示这一棵树，将原树转化成二叉树，则我们在进行状态转移的时候只用考虑给左儿子分配多少钱。
left[a]表示 a 的左儿子
right[a]表示 a 的右儿子

f[a][b]=Max{Max{f[left[a]][bleft]+f[right[a]][b-cost[a]-bleft]}+score[a],f[right[a]][b]}
这样我们可以得到一个理论[1]复杂度为 O(NM²)的算法，但是对于本题的数据范围来说，这个复杂度并不太理想。
【数据组织方案二】
上面我们把本题同 0-1 背包进行了类比。发现两道题之间的差别：附件不能被直接购买。显然，如果题目中没有附件，那么本题即为标准的 01 背包问题。
我们回到题目并考虑其特殊性:
1.附件没有附件。
2.每个主件最多只有 2 个附件。
这样，显然对于（图一）中每一组（主件＋附件），可以作为整体考虑。
对于每一组，可能的购买方案最多只有:
1.什么都不买
2.只购买主件
3.购买主件和附件 1
4.购买主件和附件 2
5.购买主件和两个附件
这样，我们可以借鉴经典的 01 背包动态规划，把每一组看作一个对象，取值和花费对应最多五种。
【算法二】
cost[i][k]表示分组后第 i 个对象的第 k 种购买方案的花费。
weight[i][k]表示分组后第 i 个对象的第 k 种购买方案的总权值。
F[i][j]表示前 i 个对象最多花费 j 元，能得到的最大权值。
F[i][j]=max(F[i-1][j-cost[i][k]]+weight[i][k]);
其中 1<=k<=5 且 cost[i][k]<=j
状态总数：O(NM)
转移代价：O(1)
这样，我们得到了一个时间复杂度为 O(NM)的优秀算法。

郁闷的金明

【题意描述】
给出 N 个物品，可以直接被购买的称为主件，而不能直接被购买的称为附件，附件只有当其主件被购买了才能被购买，一个主件可以有任意多个附件，附件没有下一级附件。每个物品都有一个权值(<50000)。
任务　购买一些物品，总价格不超过 M，使得被购买的物品的权值之和最大。
N<60
M<3200

【问题分析】
题目放宽了“一个主件最多可以有两个附件”这个限制，其它条件不变。
自然，我们当然尝试用原题中的算法来套本题。
实际上，这时候原来的算法 1 依然适用，复杂度仍为 O(NM²)。但是对于利用原题条件特殊性的算法 2，一个对象的取值可能达到 N 的组合级别，所以我们大

[1] 参见《算法艺术与信息学竞赛》贪食的九头龙中对算法复杂度的分析

可放弃对于算法 2 的讨论。
我们是否有合理的组织数据的办法呢?

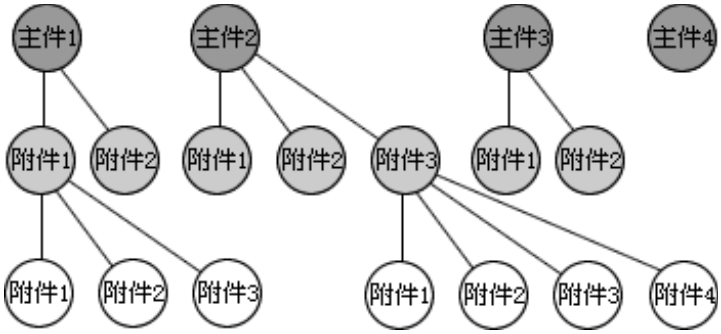
【数据组织方案三】
重新安排这些物品的顺序，使得每个附件都紧跟其主件，保证其左边的第一个主件就是它附属的主件。如下图：

这样做的好处是：一个附件能被购买的必要条件就是在其前面的最近的主件被购买了。
看似和原来的条件没有什么变化，但是实际上我们给节点的位置已经加上了一个限制 。原本树上的问题经过我们“合理地组织数据以后”，成功地转化成了一个序列上的问题。
【算法 **3**】
这样组织数据以后，我们利用前面提到的条件**“**一个附件能被购买的必要条件是其前面的最近的主件被购买了**”**，可以轻松地设计动态规划算法：
定义：
cost[i]表示第 i 个物品的价格
weight[i]表示第 i 个物品的权值
F[i][j][k]表示从第 i 个物品到第 n 个物品，最多花费 j 元，第 i 个物品前的主件有(k=1)没有(k=0)被购买。
分情况进行状态转移：
情况 I：第 i 个物品是主件
F[i][j][k]=Max{F[i+1][j-cost[i]][1]+weight[i] (j>=cost[i]),F[i+1][j][0]}
情况 II：第 i 个物品是附件
如果 k=1 F[i][j][k]= Max{F[i+1][j-cost[i]][1]+weight[i] (j>=cost[i]),F[i+1][j][1]}
如果 k=0 F[i][j][k]= F[i+1][j][0]
状态总数：O(NM)
转移代价：O(1)
时间复杂度同样是 O(NM)。

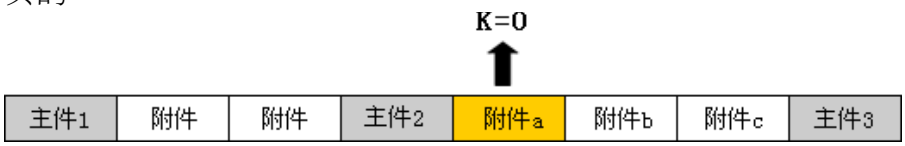
很郁闷的金明

【题意描述】
给出 N 个物品，可以直接被购买的称为主件，而不能直接被购买的称为附件，附件只有当其主件被购买了才能被购买，一个主件可以有任意多个附件，附件可以有多级，也就是说如果某个物品是附件，那么它还有可能有附属于它的下一级附件。每个物品都有一个权值(<50000)。
任务　购买一些物品，总价格不超过 M，使得被购买的物品的权值之和最大。
N<60
M<3200

【问题分析】
现在题目在原题的基础上不仅放宽了附件的个数，还放宽了附件的层数，如图所示：



从上图中，我们可以对本题有一个感性的认识：关系又“宽”又“深”。
我们依然试着从前面的题目中寻找算法：
我们可以直接套用算法 1，因为该算法正好将数据作为树结构来进行处理。而利用了题目特殊条件的算法 2 和算法 3，直接套用算法肯定是行不通的。
但是他们都很有启发性：抛弃树形的结构，重新组织成线形。
现在的题目是不是也可以类似解决呢?
【组织数据方案四】
算法 3 相对来说比较算法 2 更加一般，所以现在我们再回过头来研究一下算法 3，希望在分析过程中找到一些灵感。
回忆算法 3 的思路：
把同在一个组的主件放在附件的前面，利用动态规划“加一维”的思想，顺利地实现了将问题转化到序列上来。
关键字：主件在前　序列　动态规划
我们联想到利用树的先根遍历序，而且正好满足上面的关系。
但是这样有什么好处吗？还能进行动态规划吗？怎样设计状态才能传递父节点的状态呢?
我们再回过去看算法 3 的状态转移：
假设当前状态是 F[i][j][k]，且 k=0。
如果 i 是附件，那么实际上在到达下一个主件以前，i 后面的附件是都不会被购买的。



上图中，对于附件 a，实际上一个 k=0 的状态传递下去是没有意义的，因为附件 b 和附件 c 也必然不能被购买。

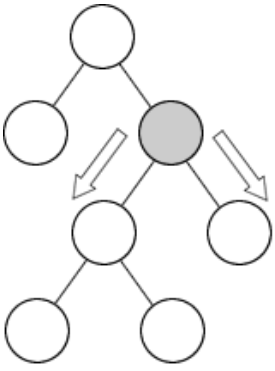
思考并总结上面的结论：
对于一个主件，我们如果不购买的话，那么其附件我们都不用考虑，而直接“跳
”到下一个主件。
我们把它应用到本题中来：

重要结论　我们考虑一棵子树的时候，如果我们不购买其根节点，那么其子树中所有节点我们都不必讨论了。
这一结论似乎很显然，但是我们并不是要在树结构中用这一结论。正如上面提到

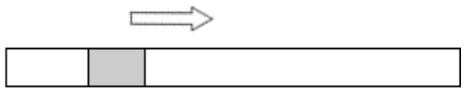
的，我们要在树的先根遍序上进行动态规划，而这一结论正是我们成功的关键。

【算法 **4**】
根据前面的思考，我们先依次求出每棵树的先根遍历序，并保存在同一个序列 list[]中。
为了利用上面的结论，我们还要求出以节点 i 为根的子树的节点总数 count[i]。
现在我们来设计动态规划算法：
定义：
cost[i]表示第 i 个物品的价格
weight[i]表示第 i 个物品的权值
F[i][j]表示从第 i 个物品到第 n 个物品，最多花费 j 元，能得到的最大权值和。
状态转移：
对于一个节点，我们考虑是否购买它：
　　购买：那么我们继续考虑它后面的节点
　　不购买：那么我们跳过它的子孙节点
方程如下：
F[i][j]=Max{F[i+1][j-cost[list[i]]]+weight[list[i]],F[i+count[list[i]]][j]}
这个算法依然是 O(NM)的，很完美地解决了本题。并且，这个算法模型对于以前有很多类似的树形动态规划题目都适用，这是我们在分析本题的过程中的意外收获。

【小结】
这是一道很有启发性的道目。反思这一题的几个不同难度的版本，不难发现我们最终都用线形模型上的动态规划取代了容易想到的树形动态规划算法。我们再次分析前面的算法，试图发现其中内在的一些东西。其实我们这个题主要就是对于树形结构和线形结构的选择，所以我们对比算法 4 和算法 1:不难发现，相比算法 4，算法 1 其实多出的操作就是枚举分配给左儿子多少钱。



树形须进行“分流”



序列直接分配到后面

而在线形的序列上，没有用的钱自然地被分配给后面的元素。也就是说：树的形态决定了在状态转移的时候要进行额外的枚举。这正是树形动态规划算法的瓶颈所在！而我们利用树的先根遍历序将转树形转化为线形序列，成功地避免了树形形态的限制，正是合理地组织数据。
我们得到的启示：凭第一感觉想出来的模型不一定是最好的，对于一个题目，我们充分挖掘其数据关系并加以利用，合理地组织数据并且尝试用已有的知识来解决，推陈出新，才能不断地进步。

前面我们在树据的组织结构上进行了合理地安排，成功地对于每一次加强的题目都设计出了优秀的算法，下面，我们来看一看“顺序”的合理安排的例子：

［例二］树的果实

【题意描述】

给出一棵有 N 个节点的有根树（根为 1 号节点），每个节点有权值。

要求对于每一个节点，求：

1.其子树中权值比该节点大的节点总数

2.树上除其子孙节点外比该节点大的节点总数

3.从根节点到该节点路径中比该节点大的节点总数

其中$(1<=N<=10^5)$

【问题分析】

对于要求的后面两个值，我们很容易想到$O(N\log_2(N))$的算法：

树上除其子孙节点外比该节点大的节点总数：直接排序，在待统计节点前的与该节点权值不同的个数再减去问题 1 的答案即为所求。

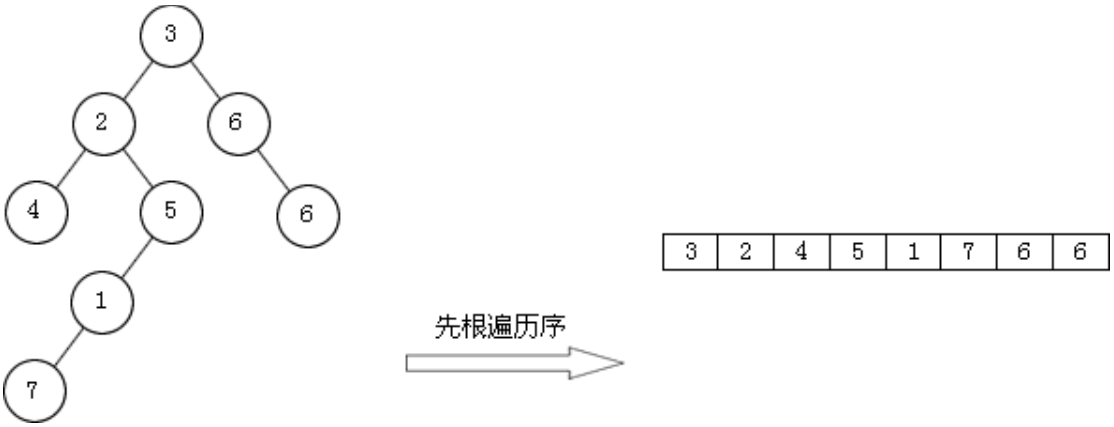
从根节点到该节点路径中比该节点大的节点总数：以权值为关键字构造线段树(若权值大可行离散化处理)，深度优先遍历树上节点，用栈记录下到节点的路径，并把当前节点插入线段树，在线段树中我们记录区间的元素个数，当前节点权值到最大权值这个区间中元素个数即为所求，我们再递归处理子树，在子树访问完毕后还须把该节点从线段树中删除。

我们最大的困难在于求：其子树中权值比该节点大的节点总数

$O(N^2)$的朴素统计方法是很容易想到的，但是本题的数据规模达到 10^5，$O(N^2)$的复杂度显然太高。我们自然想到利用线段树、树状数组这些优秀的统计数据结构来进行题目中要求的统计任务。但是这些数据结构都是用于线型序列统计的，并且似乎没有改造版本用于树形结构。

既然没有办法改造数据结构，那么我们转换数据形态——把树转化为序列再进行统计，先根遍历序即是我们转换后的理想形态。

我们给出一个例子：



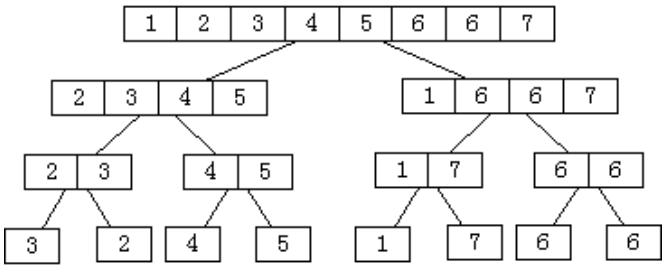
同一棵子树构成一个连续的区间，这正方便了我们的统计。

我们定义：

一个元素所在子树在遍历序中构成的区间叫作元素所在区间。
元素相比较都指其权值大小相比较。

现在问题已经转化成为：
给出一个序列，每个元素有权值。对于每一个元素，统计一个区间中有多少元素比该元素大。

这正是我们比较熟悉的序列上的统计问题。下面我们研究转化后的问题：
【数据组织方案一】
我们不对数据进行更深入的组织，直接利用先根遍历序，强制用数据结构来进行统计。当然我们可以构造出一种比较有效的嵌套数据结构——以有序表为元素的线段树，如图：



其中，线段树的每一个节点是对应区间的元素以权值为关键字的有序表。
这样，预处理可以用一个归并排序，求得树上所有区间的有序表。时间复杂度为 $O(N\log_2(N))$。
假设现在我们要统计一个区间(长度为 L)。
那么我们可以用 $\log_2(L)$的时间找到该区间的所有分解区间(不超过 $2\log_2(L)$个)。
然后在对每个分解区间进行处理：二分查找在该区间中有多少元素的权值比指定的元素的权值大。
然后把所有分解区间中比给定元素大的元素个数加起来，即为所求。
这样每统计一个元素的复杂度为 $O(\log_2^2(N))$。总时间复杂度为 $O(N\log_2^2(N))$，空间复杂度为 $O(N\log_2(N))$。

【数据组织方案二】
我们从特殊情况考虑：假设我们在先根遍历序中，需要统计元素 k，并且 k 所在区间里的元素都比它大。
显然，这时比 k 大的元素个数就是 k 所在区间中的元素个数。统计区间元素个数我们可以直接利用线段树和树状数组。
那么我们如何保证当前列表中的元素权值都比 **k** 的权值大呢?
我们重新组织数据：所有元素按从大到小的顺序排序。然后依次处理每一个元素先取得所在区间的元素个数，再将该元素插入。
我们一个很巧妙的方法：从大到小地向线段树里面加入元素，然后统计区间个数。
从大到小保证了现有的所有元素都比待插入的元素大。所以区间中的元素个数即为比待插入元素大的元素个数。
按照从大到小的顺序之前先对其区间进行统计，利用线段树或树状数组。
这样，我们得到了复杂度为 O(Nlog2(N))的算法。

WC2005 何林同学的论文中介绍了此题的另一解法，复杂度也为 O(Nlog2(N))。主要思想是也是利用树的前根遍历序，不同的是他的算法是基于容斥原理，需要正反两次遍历树，而我们这里介绍的算法是利用了“组织数据的操作顺序”这一手段来实现的。有兴趣的同学可以参见何林同学 2005 年的论文。

“形态”和“顺序”这两种数据组织对象在上面的两个例题中分别对我们进行了表演。下面我们再来分析一个更经典的题目：

［例三］航线规划

【题意描述】

给出一个有 N 个点 M 条边的无向图，两点之间可能有多条边，然后给出 Q 个命令，命令共有如下两种：

1 A B

表示删除一条 A 到 B 的边

2 A B

表示询问 AB 间共有多少条关键边（即删除改边后使得 AB 不连通)

数据保证任意时刻图都是连通的。

$1 \le N \le 3 * 10^4$

$1 \le M \le 10^5$

$1 \le Q \le 10^5$

【问题分析】

显然，我们可以轻松地设计出一个朴素的算法：

用队列保存所有边，当遇到删边操作时加上删除标记（利用 HASH 我们可以做到 O(1))，遇到询问操作时则枚举删边然后用并查集判断 AB 是否连通。这个算法处理删边的复杂度为 O(1)，处理询问的复杂度为 $O(M^2)$，空间复杂度为 O(M+N)。

我们经过思考后发现，事实上所谓的关键边都是图上的桥(由题目中的描述我们很容易想到)。而桥的数量是 O(N)级别的。

利用上面的结论，我们显然可以先用 O(E)的时间求出图中所有的桥，然后再用 $O(N^2)$的时间求出 AB 间的关键边的数量。

然而，我们所优化后的程序依然有很高的时间复杂度，根本不能胜任此题。

我们继续思考：

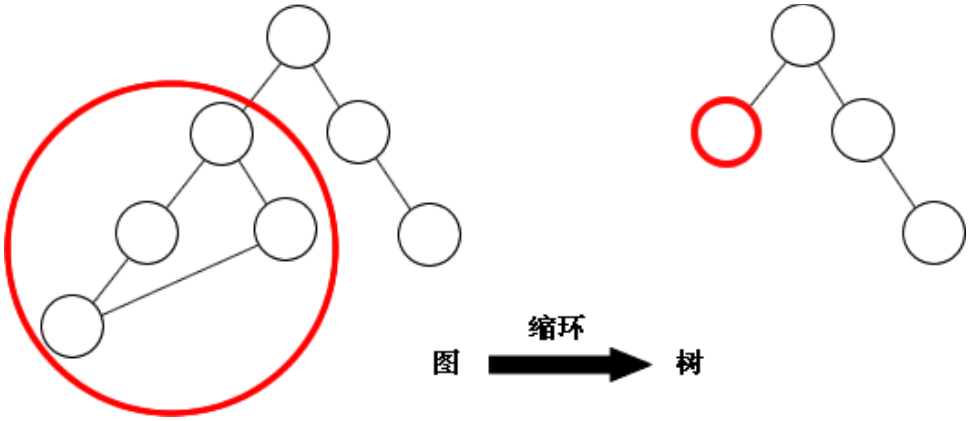
树上的任意两点间只有一条路径。

也就是说树上的任意一条边都是关键边。这跟我们的题目有什么关系呢？显然，同一个重连通分量(块)中的任意两点之间都没有关键边。

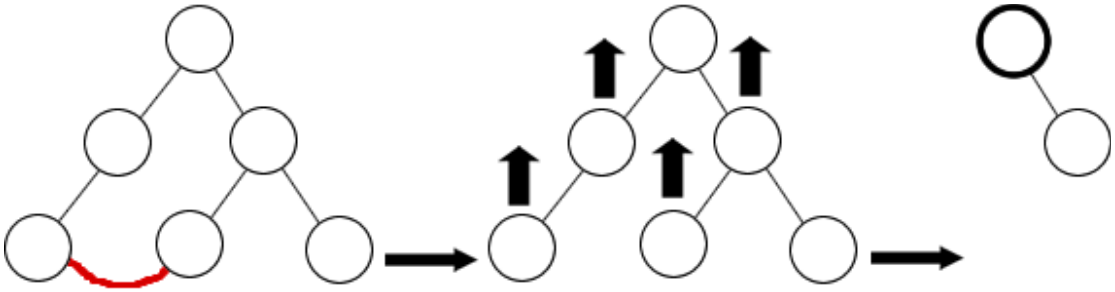
并且，对于两个不同的重连通分量 M1,M2：在进行删边操作以前，询问任意分属这两个分量的两点 $A \in M1, B \in M2$，询问的结果都是一样的，即结果只跟分量间的边有关系。也就是说，一个重连通分量可以当作整体来考虑。

【初步组织数据】

由前面的思考，我们把图中的重连分量都“缩”成一个点。构成一个新图，显然，新图是一棵树。如下：



这样，对于 AB 的询问：
若 AB 属于同一个重连通分量，则没有关键边。
若 AB 属于不同的重连通分量，则转化为求两个树上节点的距离。
求树上两个节点的距离我们有现成的办法，定义：
Depth[A]为节点 **A** 的深度
LCA(A,B)为节点 **A** 和节点 **B** 的最近公共祖先。
那么 **AB** 间的距离 **Dis(A,B)=Depth[A]+Depth[B]-2*Depth[LCA(A,B)]**
注意一个细节，即我们把一个重连通分量“缩”成一个节点时，事实上是把分量里面的所有点的深度都设为它们中最小的那个深度，即往上提升(在同一个重连通分量中以深度最小的点作其它点的代表)。
如此一来，对于一个现成的图，我们可以很快地求出两点间的关键边数量了。
预处理即为一个求重连通分量的操作，时间复杂度为 O(M)。
而对于每一个询问我们都可以 O(1)完成回答。
但事实上这道题目中的图是随时变化的(有删边操作)，这样我们就不太好处理了。
如果每次都求一次块的话，复杂度会很高。我们思考怎么处理这个问题：删边操作会导致块的分裂。我们当然可以只对被删边所在的块进行处理，但是最坏情况下还是和每次求一次块是一样的。
【进一步组织数据】
现在的问题是我们需要快速地将一个块进行重新求块，似乎是没有现成的办法。
但是如果操作不是删边，而是加边呢?
显然，在一棵树上加上一条边，必然产生环，伴随着的就是新的重连通分量产生。
我们只需要将几个有关的块进行合并。换句话说，就是把一些点的位置抬高，并把它们合并成一个块。如下图：



加上红边

比如我们加入一条边 AB，T=LCA(A,B)，那么我们的环上的节点即为 A 到 T 的路径中和 B 到 T 的路径中的节点。我们需要把环上的节点的深度都减小到 Depth[T]，并且，我们提升一个节点，其子孙节点也要一同被提升相同的高度。

我们研究发现，如果操作是加边的话，我们似乎可以很高效地处理。
那么我们当然可以把操作反过来处理(先处理后面的操作)，这样就可以实现我们所要达到我们所期望的结果——操作变为只有加边和询问。
现在我们来考虑细节实现:
我们需要用到 LCA，当然可以用中序遍历+RMQ 实现。而且，加边操作并不影响 LCA。
然后我们还有一个提升一棵子树的高度的操作。即把一棵子树的所有节点的 Detph 都减去同一个数。
显然，我们可以求出树的先根遍历序。这样，同一棵子树构成一个连续区间。利用线段树或树状数组我们就可以用 $O(\log_2(N))$的时间完成这项操作。
【小结】
这是一道很经典的题目。我们最初利用“收缩”的思想，把图整理成为一棵树，然后又巧妙地将数据从后往前处理，把原题中的“删边操作”操作变成了“加边操作”。既有“形态”，又有“顺序”上的考虑。在细节实现中，我们又利用了树的两大遍历序——中序遍历和前序遍历，把树上的求 LCA 操作和提升子树的操作变成了序列上的求 RMQ 操作和给一个区间所有元素减去一个值的操作。无处不体现出“对数据的合理组织”。

【总结】

“对数据的合理组织”无处不在，它不仅仅是一种手段，更是竞赛的一种思考方向。在数据关系越来越复杂，解题模型越来越不明显的信息学竞赛中，合理地组织了数据，就可以说离成功只有一步之遥了。

我们在被告知一个很巧妙的算法时，感兴趣的除了算法本身之外，还有就是算法的设计者到底是怎么想到这个算法的。甚至，我们往往对后者所产生的兴趣超过前者。这正是我们前进的动力，思想的源泉。

多思考、多总结、勇于探索、不断创新！

【参考资料】

《算法艺术与信息学竞赛》 刘汝佳 黄亮 著

【感谢】

感谢叶诗富老师对我的指导和帮助。

感谢古楠同学和王晓珂同学对我的论文提出了很好的建议。

【附录】

金明的预算方案[2]

【题目描述】

金明今天很开心，家里购置的新房就要领钥匙了，新房里有一间金明自己专用的很宽敞的房间。更让他高兴的是，妈妈昨天对他说：“你的房间需要购买哪些物品，怎么布置，你说了算，只要不超过 N 元钱就行”。今天一早，金明就开始做预算了，他把想买的物品分为两类：主件与附件，附件是从属于某个主件的，下表就是一些主件与附件的例子：

主件	附件
电脑	打印机，扫描仪
书柜	图书
书桌	台灯，文具
工作椅	无

如果要买归类为附件的物品，必须先买该附件所属的主件。每个主件可以有 0 个、1 个或 2 个附件。附件不再有从属于自己的附件。金明想买的东西很多，肯定会超过妈妈限定的 N 元。于是，他把每件物品规定了一个重要度，分为 5 等：用整数 1~5 表示，第 5 等最重要。他还从因特网上查到了每件物品的价格（都是 10 元的整数倍）。他希望在不超过 N 元（可以等于 N 元）的前提下，使每件物品的价格与重要度的乘积的总和最大。

设第 j 件物品的价格为 v[j]，重要度为 w[j]，共选中了 k 件物品，编号依次为 j1，j2，……，jk，则所求的总和为：

v[j1]*w[j1]+v[j2]*w[j2]+ …+v[jk]*w[jk]。（其中*为乘号）

请你帮助金明设计一个满足要求的购物单。

【输入文件】

输入文件 budget.in 的第 1 行，为两个正整数，用一个空格隔开：

N m

（其中 N（<32000）表示总钱数，m（<60）为希望购买物品的个数。）

从第 2 行到第 m+1 行，第 j 行给出了编号为 j-1 的物品的基本数据，每行有 3 个非负整数

v p q

[2] NOIP2006

（其中 v 表示该物品的价格（v<10000），p 表示该物品的重要度（1~5），q 表示该物品是主件还是附件。如果 q=0，表示该物品为主件，如果 q>0，表示该物品为附件，q 是所属主件的编号）

【输出文件】

输出文件 budget.out 只有一个正整数，为不超过总钱数的物品的价格与重要度乘积的总和的最大值（<200000）。

【输入样例】

1000 5
800 2 0
400 5 1
300 5 1
400 3 0
500 2 0

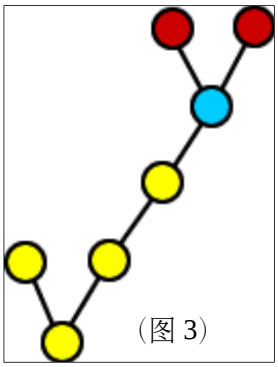
【输出样例】

2200

树的果实[3]

【题目描述】

森林中生长着许多奇特的果树，它们不仅外形独特，其果实更是可口。这天，两只小虫 Nileh 和 Nixed 决定一起分享一棵果树。他们从一直辛勤工作到下午，终于把这棵果树锯倒了。


（图 3）

他们观察着这棵果树，果树开始端　是露出地面的根部，接着像其他果树一样，有着诸多分叉（如图 3 所示就是一棵果树），在每个分叉处生长着果实，自然 Nileh 和 Nixedd 的食物就是这些果实了！他们准备把果树分成两部分，每个虫虫得到各自的一部分，两分果树的方法就是选择一个分叉点，虫虫将他们咬断（自然分叉点上的果实也被扔掉了），这样果树就被分成两部分（每个部分不一定是连在一起的）：分叉点上面的部分和分叉点的下面部分。Nileh 和 Nixed 就会各自选一个部分吃啦！比如对于左边这棵树，如果他们咬掉蓝色的果子，那么就被分为红色和黄色的两个部分。

考虑到被咬掉的果子会被浪费，他们想尽可能地减少浪费，于是虫虫给每个果子一个美味值，对于每个果子，他们决定计算如果咬掉这个果子，上面部分、下面部分和从树根到这个分叉点的路径中比这个果子更美味的果子各有多少个。他们以此来选择最终要被咬掉的果子是哪一个。遗憾的是果树可能很庞大，而小虫几乎是不会计算的，身为程序员的你帮帮他们吧。

【输入格式】

输入文件第一行是一个整数 n(n<=105)表示树的分叉数（包括树根）

输入文件的第 i 行一个数 pi，表示分叉 i 的上一级分叉的编号(pi<i)。

[3] NOI2004 浙江省队选拔赛题目

(1 号分叉即树根，它没有上级分叉点)

输入文件的第 n+i(1<=i<=n)行一个正数 ai,表示生长在 i 号分叉上的果实的美味值。（每个果子的美味值不相等）

【输出格式】

输出共 n 行，每行三个数，分别表示咬掉第 i 个果实后上面部分、下面部分、从树根到这个分叉点的路径中比它美味的果实数。

【输入样例】

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

【输出样例】

```
2 0 0
0 0 0
0 3 1
0 1 1
```

航线规划[4]

【题目描述】

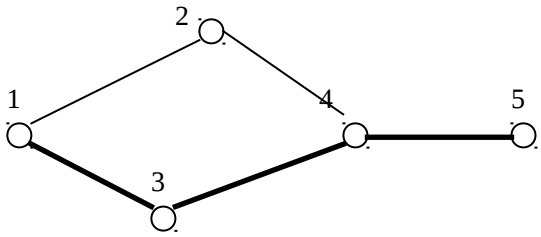
对 Samuel 星球的探险已经取得了非常巨大的成就，于是科学家们将目光投向了 Samuel 星球所在的星系——一个巨大的由千百万星球构成的 Samuel 星系。

星际空间站的 Samuel II 巨型计算机经过长期探测，已经锁定了 Samuel 星系中许多星球的空间坐标，并对这些星球从 1 开始编号 1、2、3……。

一些先遣飞船已经出发，在星球之间开辟探险航线。

探险航线是双向的，例如从 1 号星球到 3 号星球开辟探险航线，那么从 3 号星球到 1 号星球也可以使用这条航线。

例如下图所示:



在 5 个星球之间，有 5 条探险航线。

A、B 两星球之间，如果某条航线不存在，就无法从 A 星球抵达 B 星球，我们则称这条航线为关键航线。

[4] NOI2005 安徽省队选拔赛题目

显然上图中，1 号与 5 号星球之间的关键航线有 1 条：即为 4-5 航线。

然而，在宇宙中一些未知的磁暴和行星的冲撞，使得已有的某些航线被破坏，随着越来越多的航线被破坏，探险飞船又不能及时回复这些航线，可见两个星球之间的关键航线会越来越多。

假设在上图中，航线 4-2（从 4 号星球到 2 号星球）被破坏。此时，1 号与 5 号星球之间的关键航线就有 3 条：1-3，3-4，4-5。

小联的任务是，不断关注航线被破坏的情况，并随时给出两个星球之间的关键航线数目。现在请你帮助完成。

输入：第一行有两个整数 N，M。表示有 N 个星球（1< N < 30000），初始时已经有 M 条航线（1 < M < 100000）。随后有 M 行，每行有两个不相同的整数 A、B 表示在星球 A 与 B 之间存在一条航线。接下来每行有三个整数 C、A、B。C 为 1 表示询问当前星球 A 和星球 B 之间有多少条关键航线；C 为 0 表示在星球 A 和星球 B 之间的航线被破坏，当后面再遇到 C 为 1 的情况时，表示询问航线被破坏后，关键路径的情况，且航线破坏后不可恢复； C 为-1 表示输入文件结束，这时该行没有 A,B 的值。被破坏的航线数目与询问的次数总和不超过 40000。
输出：对每个 C 为 1 的询问，输出一行一个整数表示关键航线数目。

注意：我们保证无论航线如何被破坏，任意时刻任意两个星球都能够相互到达。在整个数据中，任意两个星球之间最多只可能存在一条直接的航线。

【输入样例】

```
5 5
1 2
1 3
3 4
4 5
4 2
1 1 5
0 4 2
1 5 1
-1
```

【输出样例】

```
1
3
```