树型动态规划

主讲老师：党东

【树型动态规划】

树型动态规划，顾名思议，在树型数据结构上的动态规划，它的状态、阶段、边界、初始值等，都与树有关。

前几讲学习的动态规划都是建立在线性上、坐标上或图结构上，本讲建立在层次分明，有上下关系的树上，它也有顺推和倒推两种方向，对应于树上，则为从根结点到叶结点和从叶结点到根结点。

【树型动态规划】

树形DP的特殊性：没有环，dfs是不会重复，而且具有明显而又严格的层数关系。利用这一特性，我们可以很清晰地根据题目写出一个在树（型结构）上的记忆化搜索的程序。

【树型动态规划】

树的特点与性质：

1、 有n个点，n-1条边的无向图，任意两顶点间可达

2、 无向图中任意两个点间有且只有一条路

3、 一个点至多有一个前趋，但可以有多个后继

4、 无向图中没有环；

【树型动态规划】

对于一道树规题，解题步骤如下：

1.判断是否是一道树规题：即判断数据结构是否是一棵树，然后是否符合动态规划的要求。如果是，那么执行以下步骤，如果不是，那么思考其他方法。

2.建树：通过数据量和题目要求，选择合适的树的存储方式。

如果节点数小于5000，那么我们可以用邻接矩阵存储，如果更大可以用邻接表来存储(注意边要开到2*n，因为是双向的。这是血与泪的教训)。如果是二叉树或者是需要多叉转二叉，那么我们可以用两个一维数组brother[]，child[]来存储）。

3.写出树规方程：通过观察孩子和父亲之间的关系建立方程。我们通常认为，树形DP的写法有两种：

a.根到叶子: 不过这种动态规划在实际的问题中运用的不多。本文只有最后一题提到。

b.叶子到根: 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。

【例1】加分二叉树 --1530

巴蜀中学 BASHU SECONDARY SCHOOL

Description

设一个n个节点的二叉树tree的中序遍历为（l,2,3,…,n），其中数字1,2,3,…,n为节点编号。每个节点都有一个分数（均为正整数），记第i个节点的分数为di，tree及它的每个子树都有一个加分，任一棵子树subtree（也包含tree本身）的加分计算方法如下：

subtree的左子树的加分× subtree的右子树的加分＋subtree的根的分数

若某个子树为空，规定其加分为1，叶子的加分就是叶节点本身的分数。不考虑它的空子树。

试求一棵符合中序遍历为（1,2,3,…,n）且加分最高的二叉树tree。要求输出；

（1）tree的最高加分

（2）tree的前序遍历

Input

第1行：一个整数n，为节点个数。

第2行：n个用空格隔开的整数，为每个节点的分数（分数＜100）。

Output

第1行：一个整数，为最高加分（结果不会超过4,000,000,000）。

第2行：n个用空格隔开的整数，为该树的前序遍历。

【例1】加分二叉树 --1530

Sample Input

5

5 7 1 2 10

Sample Output

145

3 1 2 4 5

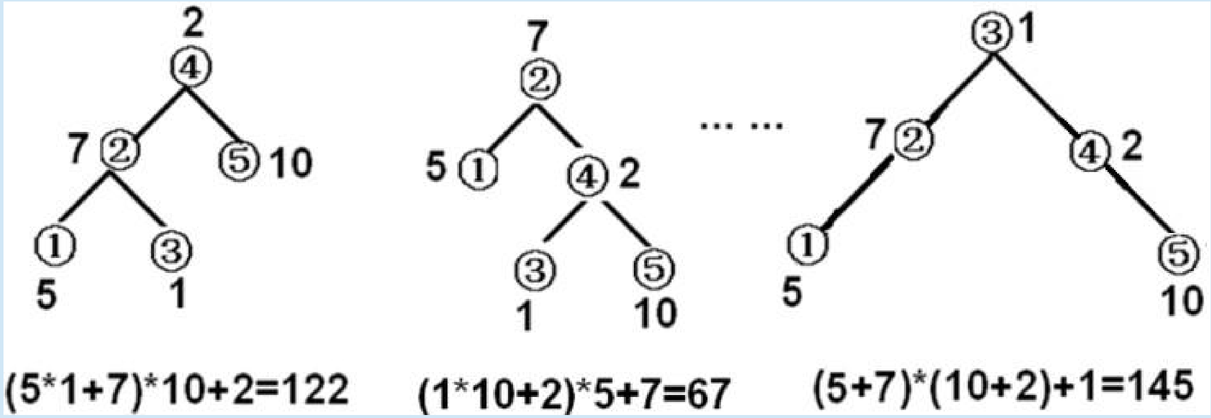
Hint

【数据范围】

$1 \le n < 30$

【例1】加分二叉树 --1530

【思路点拨】本题已经说明了问题的模型是一棵树，而且是一棵中序遍历为1,2,3,…,n的二叉树。而对于一棵中序遍历为1,2,3,…,n的二叉树有很多种形式，对于样例，下图就列出了3种形式，而根据题意可知，第三种形式得到的得分最大。

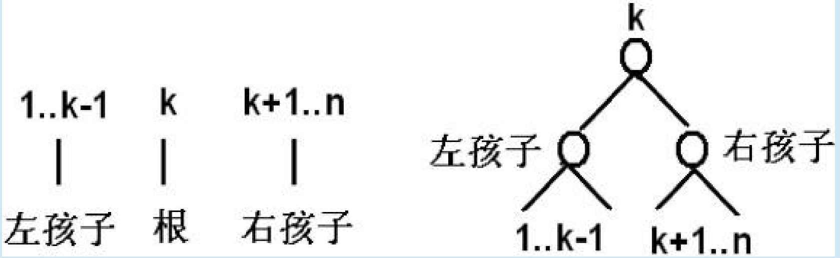


性质：中序遍历是按“左-根-右”方式进行遍历二叉树，因此二叉树左孩子遍历序列一定在根结点的左边，右孩子遍历序列一定在根结点的右边！

【例1】加分二叉树 --1530

【思路点拨】

因此，假设二叉树的根结点为k，那么中序遍历为1,2,…,n的遍历序列，左孩子序列为1,2,…,k-1，右孩子序列为k+1,k+2,…,n，如下图：



我们思考的方式变为，要使得整棵树最优，必须左孩子和右孩子都最优，因此设f[i][j]表示以结点i,i+1,i+2…,j组成的二叉树所得的最大加分；设根为k，则枚举根结点。d[k]表示k结点的最大分值；故有：

f[i][j]=max{f[i][k-1]*f[k+1][j]+d[k]};（1<=i<=k<=j<=n）

初始条件：f[i][i]=d[k]

Answer=f[1][n]

时间复杂度：O(n3)

题目还要求输出最大加分树的前序遍历序列，因此要构造这个树，我们只需记录每次的决策值，令p[i][j]=k，表示中序遍历为i,i+1,…,j的二叉树的取最优决策时的根结点为k，最后前序遍历这个树即可。

【例1】加分二叉树 --1530

【参考代码】

```
int p[31][31],f[31][31]={0},d[31],n,ans;
int DP(int i,int j)//i-j作为一棵树的最高加分值
{  int k,maxx=0,t;
   if(i>j)return 1;  //i到j是一棵空树，返回1
   if(i==j)return d[i]; //i到j只有一个节点，返回本身的值
   if(f[i][j]>0)return f[i][j]; //若备忘录有记载，则直接返回其值
   for(k=i;k<=j;k++)    //在i和j之间枚举k是根
   {  t=DP(i,k-1)*DP(k+1,j)+d[k];  //递归求左右两边
      if(t>f[i][j]){f[i][j]=t;p[i][j]=k;}//记录最优选择的根
   }
   return f[i][j];
}
void Print(int i,int j) //输出方案
{  if(i>j)return;
   if(i==j){cout<<i<<" ";return;}
   cout<<p[i][j]<<" ";
   Print(i,p[i][j]-1);
   Print(p[i][j]+1,j);
}
```

【例2】二叉苹果树 --1529

【问题描述】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。下面是一颗有4个树枝的树：

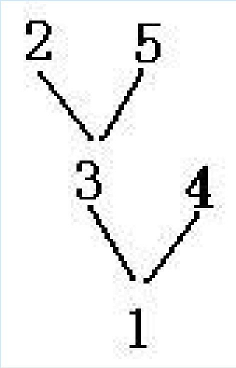
现在这颗树枝条太多了，需要剪枝。但是一些树枝上长有苹果。给定需要保留的树枝数量，求出最多能留住多少苹果。

【文件输入】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。

【文件输出】一个数，最多能留住的苹果的数量。

【样例输入】

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



【样例输出】21

【例2】二叉苹果树 --1529

【思路点拨】由题意可知，需要保留的树枝数量为Q条，即保留结点t=Q+1个。树根必须保留，可以分三种情况讨论保留苹果的最大数：

①树根的左子树为空，全部保留右子树，右子树中保留t-1个结点；

②树根的右子树为空，全部保留左子树，左子树中保留t-1个结点；

③树根的两棵子树非空，设左子树保留k个结点，则右子树保留t-k-1个结点。

要得到保留树根时的苹果最大数，只需要上述三个方案中的最大值。设树根为V，左儿子为ch[v][1],右儿子为ch[v][2]，对于①方案，要取得该方案的最大值，需要取得以ch[v][2]为根，保留t-1个结点的最大值。这时同样具有上述三种方案。其他②③情况同理；由此可以看出，该问题具有明显的最优子结构性质，每个问题都与左右儿子结点有关系，但不与孙子结点发生关系，具备无后效性；且计算方案时，搜索子结构时具备重叠性，可以用动态规划来解决。

阶段和状态：f[v][t]：表示以V为根的树上保留t个节点的最大权值和。设ch[v][1],ch[v][2]分别存V节点的左右孩子。

状态转移方程：

f[v][t]=max{f[ch[v][1]][i]+f[ch[v][2]][t-i-1]+num[v]}(0<=i<=t-1)

初始化：f[v][t]=0,(t==0)；

f[v][t]=num[v]；(ch[v][1]==0且ch[v][2]==0)

Answer=f[1][q+1]；

注意：本题知道根为1；若不知道根结点需要枚举根节点，再建立树；

【例2】二叉苹果树 --1529

【参考代码】

```
int n,q,i,j,a,b,c,ch[101][3],f[101][101]={0},map[101][101],num[101];
void MakeTree(int v)
{  int i;
   for(i=1;i<=n;i++)
      if(map[v][i]>=0)
      {  ch[v][1]=i; num[i]=map[v][i];
         map[v][i]=-1;map[i][v]=-1;
         MakeTree(i);
         break;
      }
   for(i=1;i<=n;i++)
       if(map[v][i]>=0)
       {  ch[v][2]=i;num[i]=map[v][i];
          map[v][i]=-1;map[i][v]=-1;
          MakeTree(i);
          break;
       }
}
```

【例2】二叉苹果树 --1529

【参考代码】

```
int DP(int v,int t)
{  int i;
   if(t==0)return 0;
   if((ch[v][1]==0)&&(ch[v][2]==0))return num[v];
   if(f[v][t]>0)return f[v][t];
   for(i=0;i<=t-1;i++)
      f[v][t]=max(f[v][t],DP(ch[v][1],i)+DP(ch[v][2],t-i-1)+num[v]);
   return f[v][t];
}
int main()
{  cin>>n>>q; q++;
   for(i=1;i<=n;i++)
     for(j=1;j<=n;j++)map[i][j]=-1;
   for(i=1;i<=n-1;i++){cin>>a>>b>>c;map[a][b]=c;map[b][a]=c;}
   MakeTree(1);
   cout<<DP(1,q)<<endl;
}
```

【例3】选课 --1531

【问题描述】大学里实行学分。每门课程都有一定的学分，学生只要选修了这门课并考核通过就能获得相应的学分。学生最后的学分是他选修的各门课的学分的总和。
每个学生都要选择规定数量的课程。其中有些课程可以直接选修，有些课程需要一定的基础知识，必须在选了其他的一些课程的基础上才能选修。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。为便于表述每门课都有一个课号，课号依次为1,2,3,…。下面举例说明：

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

上例中1是2的先修课，即如果要选修2，则1必定已被选过。同样，如果要选修3，那么1和2都一定已被选修过。
学生不可能学完大学所开设的所有课程，因此必须在入学时选定自己要学的课程。每个学生可选课程的总数是给定的。现在请你找出一种选课方案，使得你能得到学分最多，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。

【例3】选课 --1531

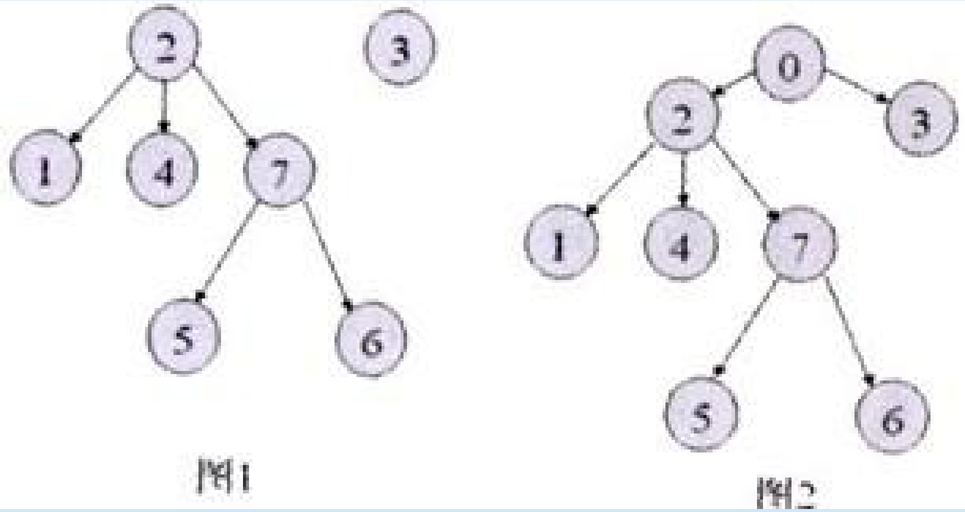
【文件输入】输入文件的第一行包括两个正整数M，N(中间用一个空格隔开)，其中M表示待选课程总数(1<=M<=1000)，N表示学生可以选的课程总数(1<=N<=M)。接下来M行，每行代表一门课，课号依次为1,2,…,M。每行有两个数(用一个空格隔开)，第一个数为这门课的先修课的课号(若不存在先修课则该项为0)，第二个数为这门课的学分。学分不超过20的正整数。

【文件输出】输出文件只有一行为一个数，即实际所选课程的学分总数。

【样例输入】

7 4
2 2
0 1
0 4
2 1
7 1
7 6
2 2

【样例输出】13



【例3】选课 --1531

【思路点拨】

每门课程最多只有1门直接先修课，如果我们把课程看成结点，也就是说每个结点最多只一个前驱结点。

如果把前驱结点看成父结点，换句话说，每个结点只有一个父结点。显然具有这种结构的模型是树结构，要么是一棵树，要么是一个森林。

这样，问题就转化为在一个M个结点的森林中选取N个结点，使得所选结点的权值之和最大。同时满足每次选取时，若选儿子结点，必选根结点的条件。

- 如图1，为两棵树，我们可以虚拟一个结点，将这些树连接起来，那么森林就转会为了1棵树，选取结点时，从每个儿子出发进行选取。显然选M=4时，选3，2，7，6几门课程最优。

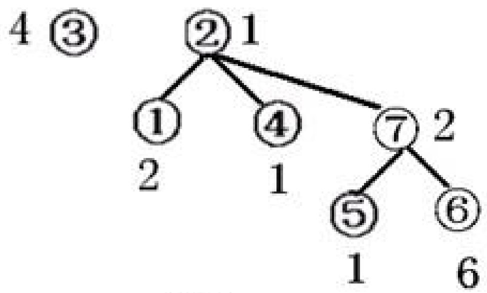


图1

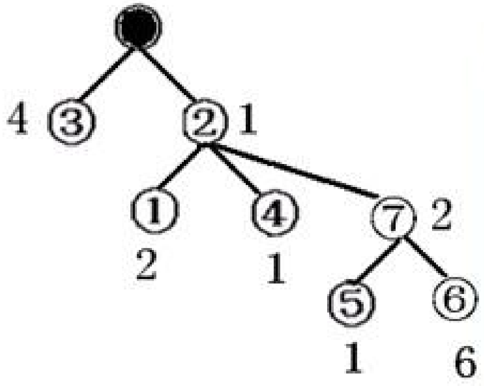


图2

【例3】选课 --1531

【思路点拨】

转化为二叉树

- 如果该问题仅仅只是一棵二叉树，我们对儿子的分配就仅仅只需考虑左右孩子即可，问题就变得很简单了。因此我们试着将该问题转化为二叉树求解。

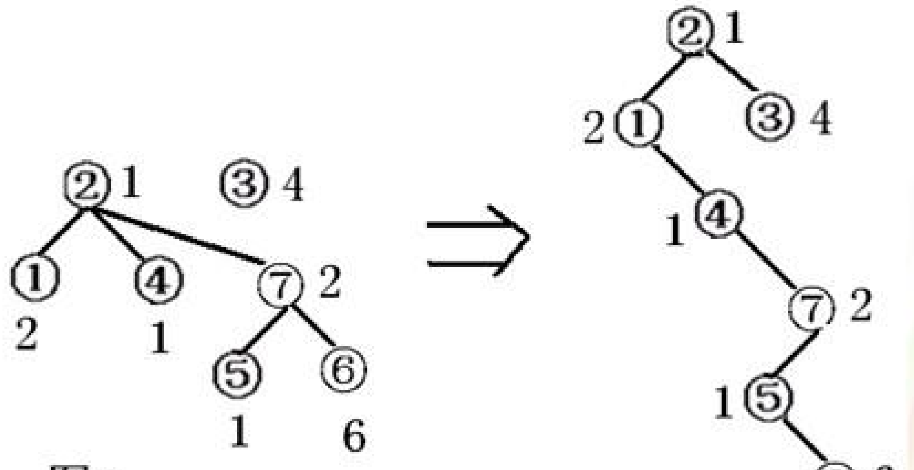


图1　图2

- 图2就是对图1采用孩子兄弟表示法所得到的二叉树

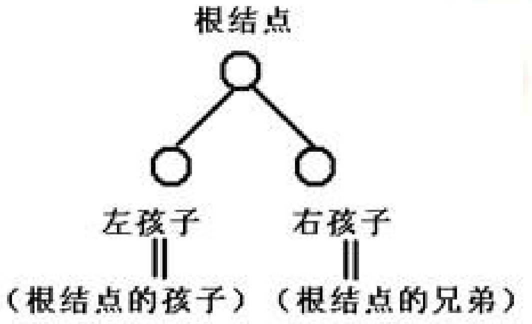
```
cin>>n>>m;
for(i=1;i<=n;i++)
{  cin>>k>>v;
  a[i].value=v;
   if(son[k]==0)a[k].left=i;
   else a[son[k]].right=i;
   son[k]=i;
}
```

【例3】选课 --1531

【思路点拨】

动态规划

- 仔细理解左右孩子的意义（如右图）：
  左孩子：原根结点的孩子
  右孩子：原根结点的兄弟
- 也就是说，左孩子分配选课资源时，
  根结点必须要选修，而与右孩子无关。



- 因此，我们设f(i,j)表示以i为根结点的二叉树选j门课程的所获得的最大学分，则有，

$$f(i,j)=\max\begin{cases} f(i_l,k)+f(i_r,j-k-1)+a[i], \text{根结点选修} \\ f(i_r,j), \text{根结点不选修} \end{cases}$$

- $0<=k<j<n,\ i\in(1..m)$
- 时间复杂度$O(mn^2)$

【例3】选课 --1531

【参考代码】

```
#include<iostream>
using namespace std;
struct tree{int left,right,value;}a[500]={0};
int n,m,f[500][500]={0},son[500]={0};
void dp(int i,int j)
{  int k,tem,ans;
   if(f[i][j]>0)return;
   if(i==0||j==0)return;
   if(a[i].right!=0)dp(a[i].right,j);
   f[i][j]=f[a[i].right][j];
   for(k=0;k<j;k++)
   {  if(a[i].left!=0)dp(a[i].left,k);
      if(a[i].right!=0)dp(a[i].right,j-k-1);
      f[i][j]=max(f[i][j],f[a[i].left][k]+f[a[i].right][j-k-1]+a[i].value);
   }
}
```

【例3】选课 --1531

巴蜀中学 BASHU SECONDARY SCHOOL

【参考代码】

```
int main()
{  int i,k,v;
   cin>>n>>m;
   for(i=1;i<=n;i++)
   {  scanf("%d%d",&k,&v);
      a[i].value=v;
      if(son[k]==0)a[k].left=i;else a[son[k]].right=i;
      son[k]=i;
   }
   dp(a[0].left,m);
   cout<<f[a[0].left][m]<<endl;
}
```