浅谈树形动态规划的应用

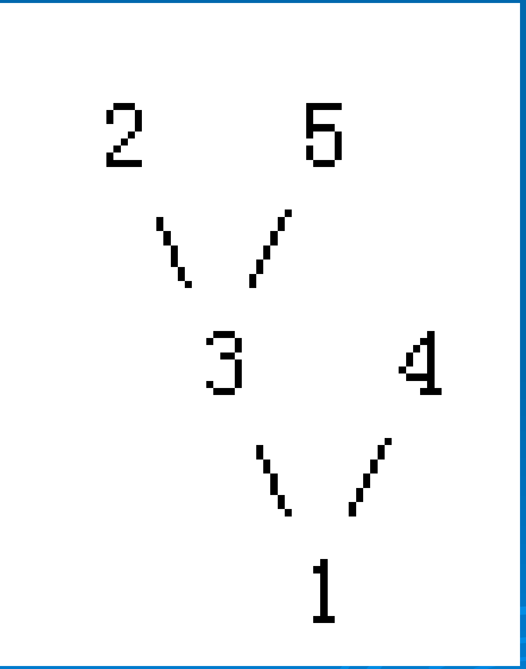
Barty

问题引入

- URAL1018　　苹果二叉树
- 题目大意：设想苹果树很象二叉树，每一枝都是生出两个分支。我们用自然数来数这些枝和根那么必须区分不同的枝（结点），假定树根编号都是定为1，并且所用的自然数为1到N。N为所有根和枝的总数。当一棵树太多枝条时，采摘苹果是不方便的，这就是为什么有些枝要剪掉的原因。现在我们关心的是，剪枝时，如何令到损失的苹果最少。给定苹果树上每条枝的苹果数目，及必须保留的树枝的数目。你的任务是计算剪枝后，能保留多少苹果。

样例数据

- **Sample Input**
- 5 2
- 1 3 1
- 1 4 10
- 2 3 20
- 3 5 20
- **Sample Output**
- 21



引发思考

- 这道题我们不能用以往的线性的动态规划的方法来解决。
- 如果用DFS来做的话时间复杂度会很高。
- 那么有没有高效的算法？
- 有！

树形DP

应用树形动态规划的前提

- 整个图是一个树状的结构或者可以转化为树状的结构。
- 对于每个根节点的状态，跟且仅跟所属的孩子（大多为2个）有牵连关系。也就是说，父亲对孩子没有影响。
- 状态可以简单的表示
- 有重叠子问题(可以没有，不过那样应用dp就没有意义了)

树形DP的优势

- 程序简短，思路清晰
- 空间、时间复杂度都较低
- 容易调试（和单纯DFS相比）

开始解决上面的问题

- 首先我们把问题转化：
- 对于一个二叉树，除根节点外，每个节点都有相应的一个权值。在此基础上，求保留多少个点使得其仍然满足树的性质且权值总和最大。

写出DP方程

- 仿照线性的方程，我们设ch[v,1],ch[v,2]分别存V节点的左右孩子。Dp[v,l]存以V为根的树保留L个节点的最大权和。
- Dp[v,l]=max{dp[ch[v,1],j]+dp[ch[v,2],l-j-1]}
  (0<=j<=l-1)
- 这里特别指出，为了使其仍然为二叉树，我们一定要保留根节点，因此j<=l-1

如何实现？

- 我们借用DFS的模块来实现：
- Procedure dfs(v:integer);
- Var
-   I:integer;
- Begin
-   For i:=1 to n do if father[i]=v then
-   Begin
-     Dfs(i);
-     Dp[v]←Func(dp[i]);
-   End;
- End;

发现缺点

- 这种dfs模块固然简洁，不过对于大量的重复子结构却做了无用功。因此，对于每种状态，我们都要“只算一遍”！

- 方法：我们可以建立一个boolean表，存储visited[v]代表V是否已经得出最优解（即V及其子树是否都已运算过）

修正版

```
Procedure dfs(v:integer);
Var
  I:integer;
Begin
  Visited[v]:=true;
  For i:=1 to n do if father[i]=v then
  Begin
    If not visited[i] then Dfs(i);
    Dp[v]←Func(dp[i]);
  End;
End;
```

数据结构

- var
-   n,q,i,j,a,b,c:longint;
-   ch:array[1..100,1..2] of longint;
-   dp:array[1..100,0..100] of longint;
-   map:array[1..100,1..100] of longint;
-   num:array[1..100] of longint;

预处理

```
procedure maketree(v:longint);
var
  i:longint;
begin
  for i:=1 to n do
    if map[v,i]>=0 then
    begin
      ch[v,1]:=i;
      num[i]:=map[v,i];
      map[v,i]:=-1;map[i,v]:=-1;
      maketree(i);
      break;
    end;
  for i:=1 to n do
    if map[v,i]>=0 then
    begin
      ch[v,2]:=i;
      num[i]:=map[v,i];
      map[v,i]:=-1;map[i,v]:=-1;
      maketree(i);
      break;
    end;
end;
```

主过程

```
procedure dfs(v,l:longint);
var
  i:longint;
begin
  if (l=0) then dp[v,l]:=0
  else if (ch[v,1]=0)and(ch[v,2]=0) then dp[v,l]:=num[v]
  else begin
    dp[v,l]:=0;
    for i:=0 to l-1 do
    begin
      if dp[ch[v,1],i]=0 then dfs(ch[v,1],i);
      if dp[ch[v,2],l-i-1]=0 then dfs(ch[v,2],l-i-1);
      dp[v,l]:=max(dp[v,l],dp[ch[v,1],i]+dp[ch[v,2],l-i-1]+num[v]);
    end;
  end;
end;
```

主程序

```
begin
  readln(n,q);
  for i:=1 to n do for j:=1 to n do map[i,j]:=-1;
  for i:=1 to n-1 do
  begin
    readln(a,b,c);
    map[a,b]:=c;map[b,a]:=c;
  end;
  maketree(1);
  dfs(1,q+1);
  writeln(dp[1,q+1]);
end.
```

- 通过对上面的题目的分析，我们对树形动态规划有了一定的认识。下面我来具体介绍树形动态规划中的几个重要方法和技巧。
- 大家在阅读后文时可以先将题目思考一下，想出算法，加深理解。

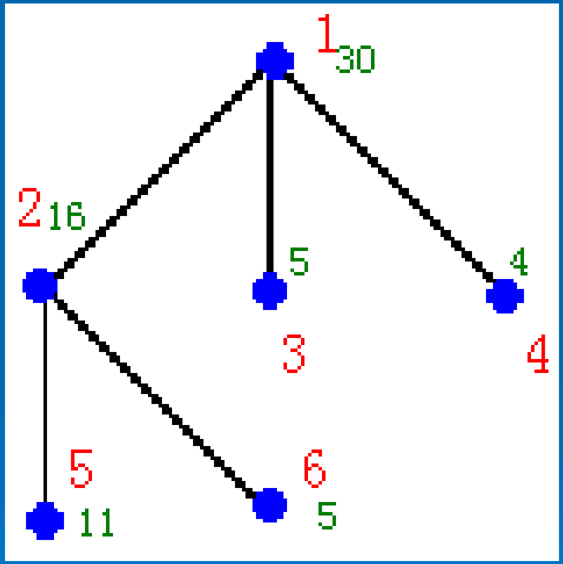
小胖守皇宫

- 描述 **Description**
- huyichen世子事件后，xuzhenyi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xuzhenyi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。
- 帮助xuzhenyi布置侍卫，在看守全部宫殿的前提下，使得花费的经费最少。

- 输入格式 Input Format
- 　输入文件中数据表示一棵树，描述如下：
- 第1行 n，表示树中结点的数目。
- 第2行至第n+1行，每行描述每个宫殿结点信息，依次为：该宫殿结点标号i（0<i<=n），在该宫殿安置侍卫所需的经费k，该边的儿子数m，接下来m个数，分别是这个节点的m个儿子的标号r1，r2，...，rm。
- 对于一个n（0 < n <= 1500）个结点的树，结点标号在1到n之间，且标号不重复。
- 输出格式 **Output Format**
- 　输出文件仅包含一个数，为所求的最少的经费。

样例数据

- 样例输入 Sample Input
- 6
- 1 30 3 2 3 4
- 2 16 2 5 6
- 3 5 0
- 4 4 0
- 5 11 0
- 6 5 0
- 样例输出 Sample Output
- 25



巧妙地存储状态

- 考虑这道题对于一个节点I的最小值dp[i],必然由i的子节点进行控制。而每个点向下只有三种情况：这个点设置守卫、这个点不设置守卫但下面有一个点控制这个点、这个点不设置守卫且下面没有点控制这个点。对于这三种状态，我们分别存为1,2,3。
- 对于dp[I,1]，我们只要找子节点的最小值的和即可。
- 对于dp[I,3],我们只要求子节点中状态1的和即可。
- 重要的是dp[I,2]，因为这个点如果不设置守卫，只需要在子节点中设置一个守卫就足够了。于是我们先求子节点中状态2、3最小值的和，然后再找子节点中状态2到状态1的最小增量(delta)即可。

程序的核心部分

```
procedure dfs(v:longint);
var
  i,s1,s2,s3,minn:longint;
  leaf:boolean;
begin
  s1:=0;s2:=0;s3:=0;leaf:=true;minn:=maxint;
  for i:=1 to n do
    if fa[i]=v then
      begin
        leaf:=false;
        dfs(i);
        s1:=s1+dp[i,2];
        s2:=s2+min(dp[i,3],dp[i,2]);
        s3:=s3+min(min(dp[i,1],dp[i,2]),dp[i,3]);
        minn:=min(minn,dp[i,3]-min(dp[i,2],dp[i,3]));
      end;
```

程序的核心部分

```
 if leaf then
  begin
    if fa[v]<>0 then dp[v,1]:=0
    else dp[v,1]:=maxint;
    dp[v,2]:=maxint;
    dp[v,3]:=num[v];
  end
  else begin
    dp[v,1]:=s1;
    dp[v,2]:=s2+minn;
    dp[v,3]:=s3+num[v];
  end;
end;
```

选课

- 这道题大家相比都清楚了，我就不再把题目打一遍了
- 对于多叉树，大家也许想不出什么好办法，有的人会考虑枚举根节点的所有孩子的所有情况。这样时间复杂度并不比深搜好多少。
- 于是我们考虑，是否能将多叉树(或者森林)转化为二叉树呢？

- 可以！

树、森林到二叉树的转换

- （1）将树转换为二叉树

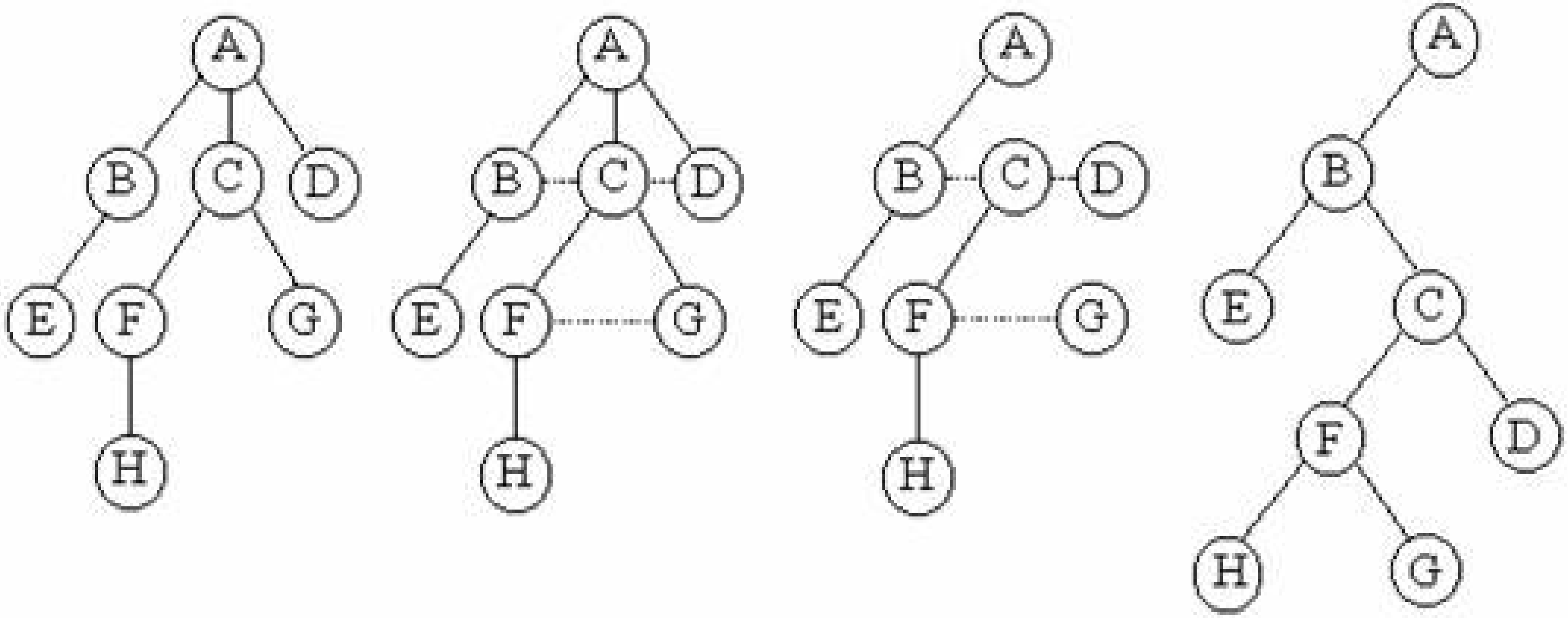
树中每个结点最多只有一个最左边的孩子(长子)和一个右邻的兄弟。按照这种关系很自然地就能将树转换成相应的二叉树。

将一般树转化为二叉树的思路，主要根据树的孩子-兄弟存储方式而来，步骤是：

(1) 加线：在各兄弟结点之间用虚线相链。可理解为每个结点的兄弟指针指向它的一个兄弟。

(2) 抹线：对每个结点仅保留它与其最左一个孩子的连线，抹去该结点与其它孩子之间的连线。可理解为每个结点仅有一个孩子指针，让它指向自己的长子。

(3) 旋转：把虚线改为实线从水平方向向下旋转 45 ℃ ，成右斜下方向。原树中实线成左斜下方向。这样就树的形状成呈现出一棵二叉树。



(a) 一般树　　(b) 加线　　(c) 抹线　　(d) 旋转整理

图 6.15 一般树转换为二叉树

将一个森林转换为二叉树

- 森林是树的有限集合，如图 6.17(a) 所示。由上节可知，一棵树可以转换为二叉树（没有右子树），一个森林就可以转换为二叉树（没有右子树）的森林。将森林转换为二叉树的一般步骤为：
  （１）将森林中每棵子树转换成相应的二叉树。形成有若干二叉树的森林，如图 6.17(ｂ) 所示。
  （２）按森林图形中树的先后次序，依次将后边一棵二叉树作为前边一棵二叉树根结点的右子树，这样整个森林就生成了一棵二叉树，实际上第一棵树的根结点便是生成后的二叉树的根结点。图 6.17 是将 一个森林转化为一棵二叉树的示例。图 6.17(d) 是转化后的一棵二叉树。

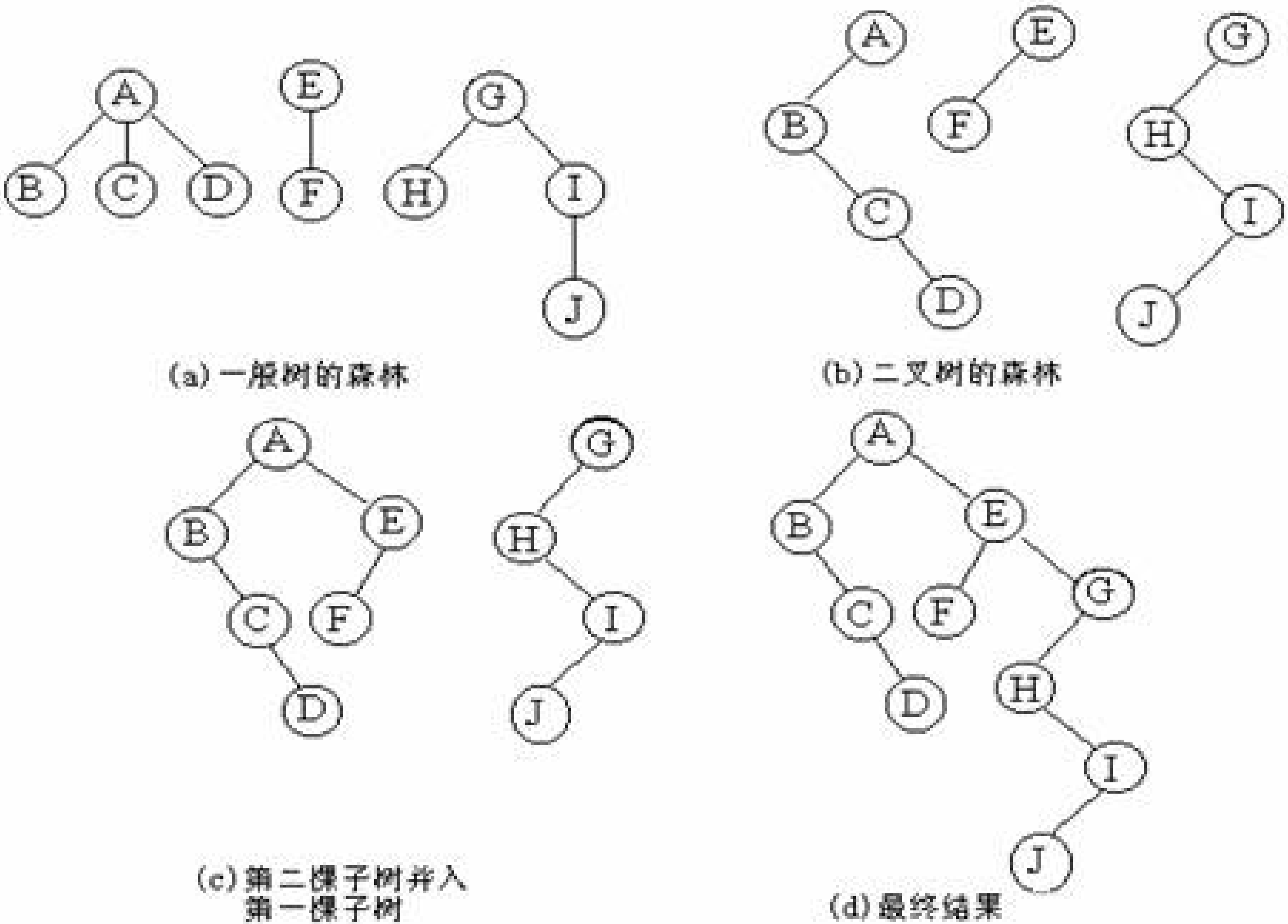


图6.17 森林转换为二叉树

- 当本题转化为二叉树以后，相信聪明的读者一定能够想出解决的方法吧。
- Dp[v,l]=max{max{dp[ch[v,1],j]+num[v]+dp[ch[v,2],l-j-1]},dp[ch[v,2],l]}
- 具体实现同上题，我就不再罗嗦了。

- 通过对3道例题的理解和掌握，相信大家对树形动态规划一定有了一个清晰的理解和认识。树形动态规划虽然以往NOIp不常考到，但是对于OIer还是一个很重要的知识点，大家一定要牢牢掌握！

- 预祝大家在NOIp取得优异成绩！

The End.

Thanks for your attention!