树型dp

1. 树上dp

f[i]:以i为根的子树对于的最优值。

f[i]=max(f[j])+w[i]

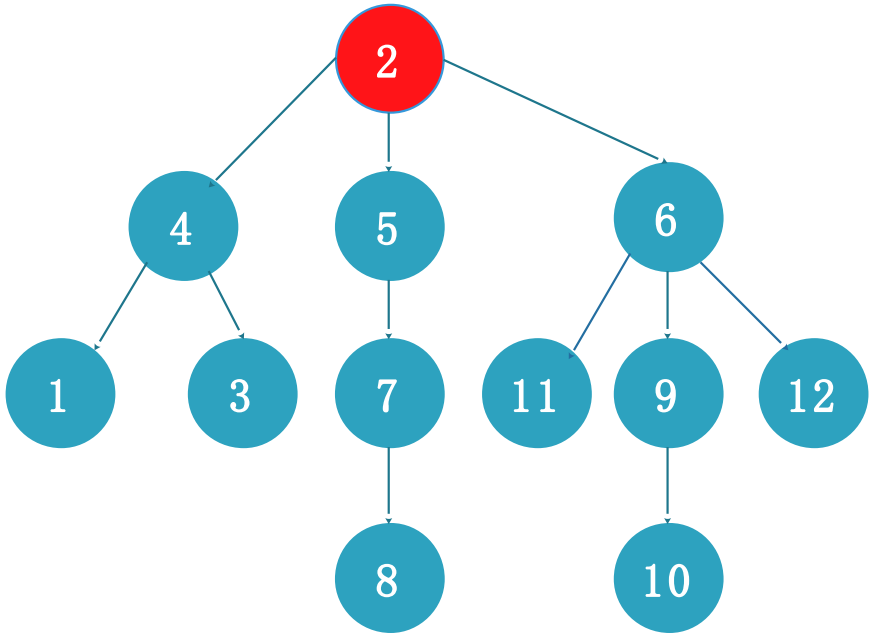
f[i]=∑(f[j])+w[i]; 部分和

j是i的儿子

求解过程:

自上而下的求解（递归）：记忆化搜索

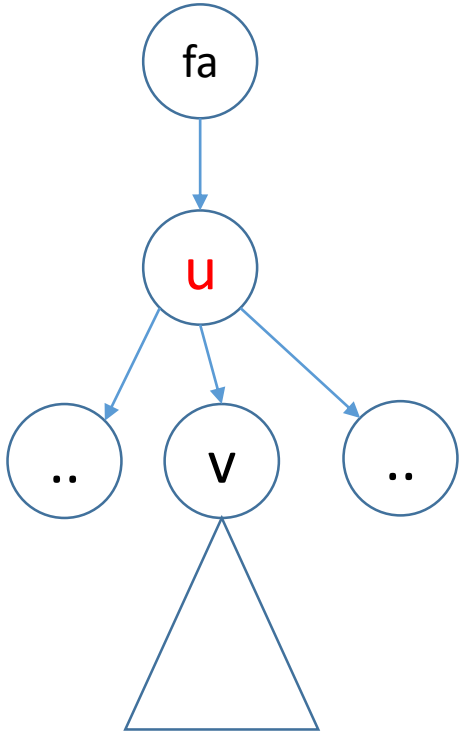
只有求出儿子j，才能求父亲i，最优子结构。



常用操作：预处理节点深度和子树节点个数（或权值和）

```
void dfs(int u,int dep,int fa){
      d[u]=dep;
      siz[u]=1;//sum[u]=a[u];
      for(int i=h[u];i>0;i=e[i].nxt){
            int v=e[i].v;
            if(v==fa)continue;
            dfs(v,dep+1,u);
            siz[u]+=siz[v]; //sum[u]+=sum[u];
      }
}

dfs(1,0,0);
```



一般的模型：

```
void  dp(int u){
      //if(u==0)return;
      //if(f[u]已经求过)return;
      f[u]=初始值;
       for(  u的每一个儿子v) {
            dp(v);
            f[u]=opt(f[v])...;
      }
}
```

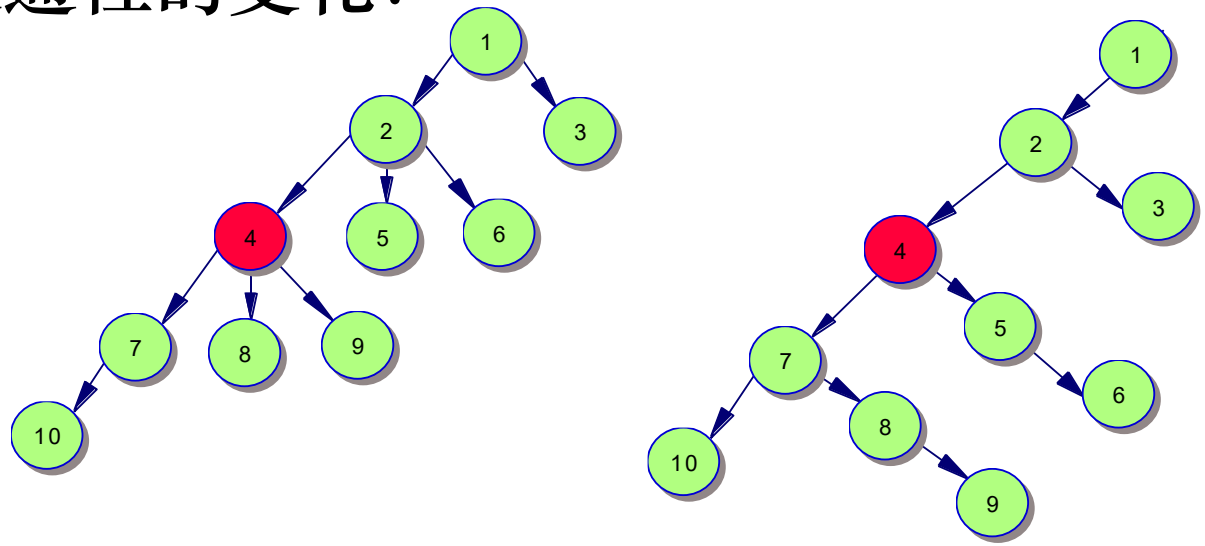
```
int  dp(int u){
      //if(u==0)return 0;
      //if(f[u]已经求过)return f[u];
      f[u]=初始值;
       for(  u的每一个儿子v) {
            f[u]=opt(f[u],dp[v])
      }
      return f[u];
}
```

2.树上选点问题（P2014 选课）

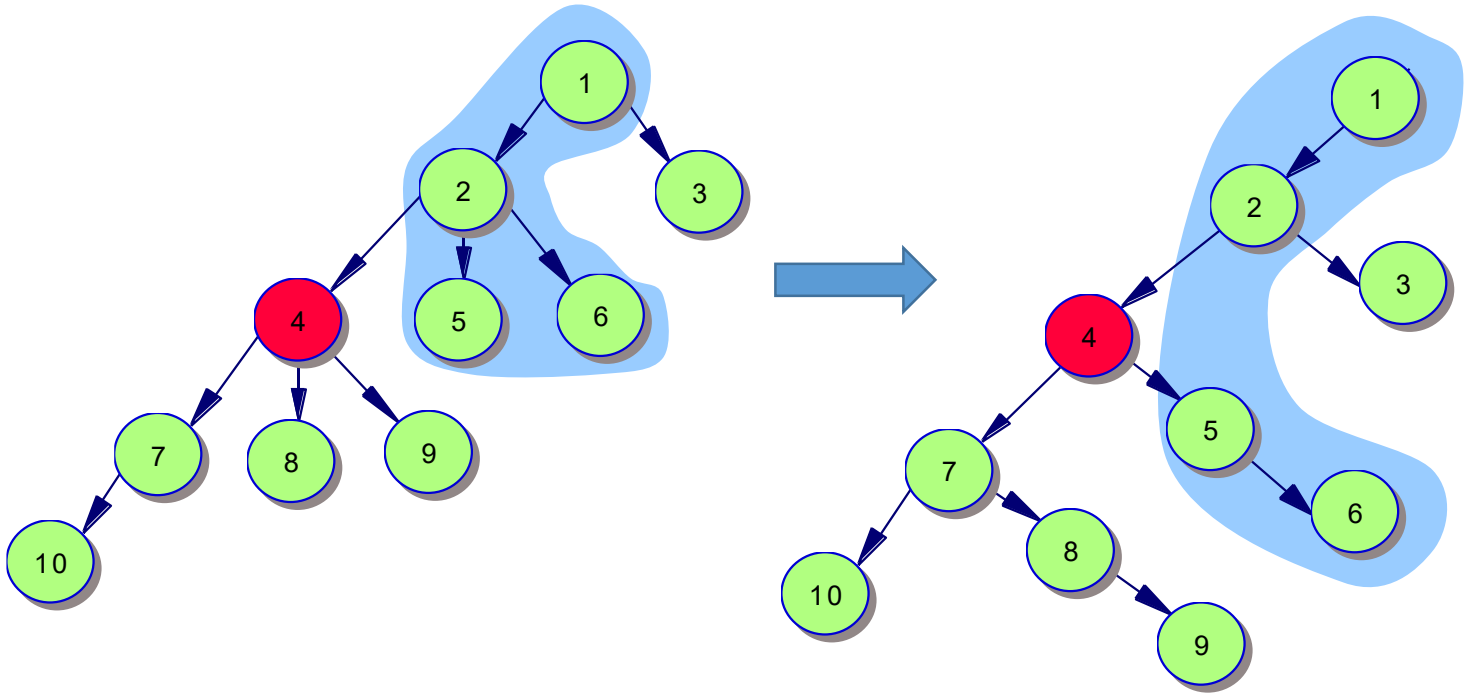
多叉树转为为二叉树

左孩子右兄弟表示法

树的结构和连通性的变化：



“缺点” 转化后的树形态上发生了变化；本质关系没变



father ,son

2 11

R[11]=L[2]

L[2]=11

R[son]=L[father];

L[father]=son

转化为二叉树后f [u][x]:以u为根的子树，包含x个结点的最优值，不一定必须有结点v:

f[u][x]=max{f[brother[u][x]:

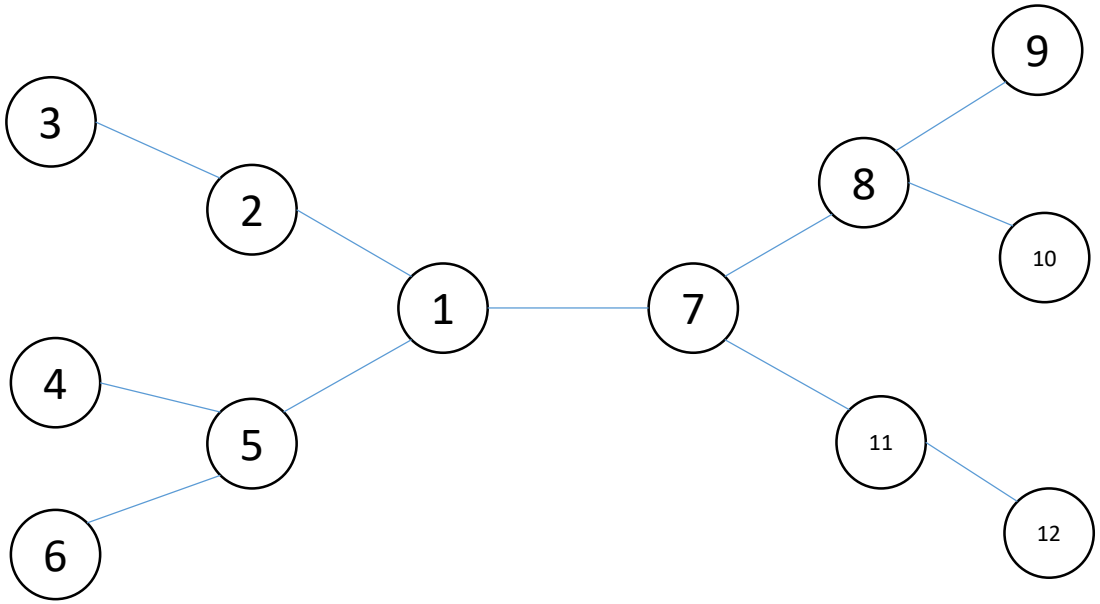
如果不包含结点u（显然也不含有左孩子），则把x全分给u的右孩子

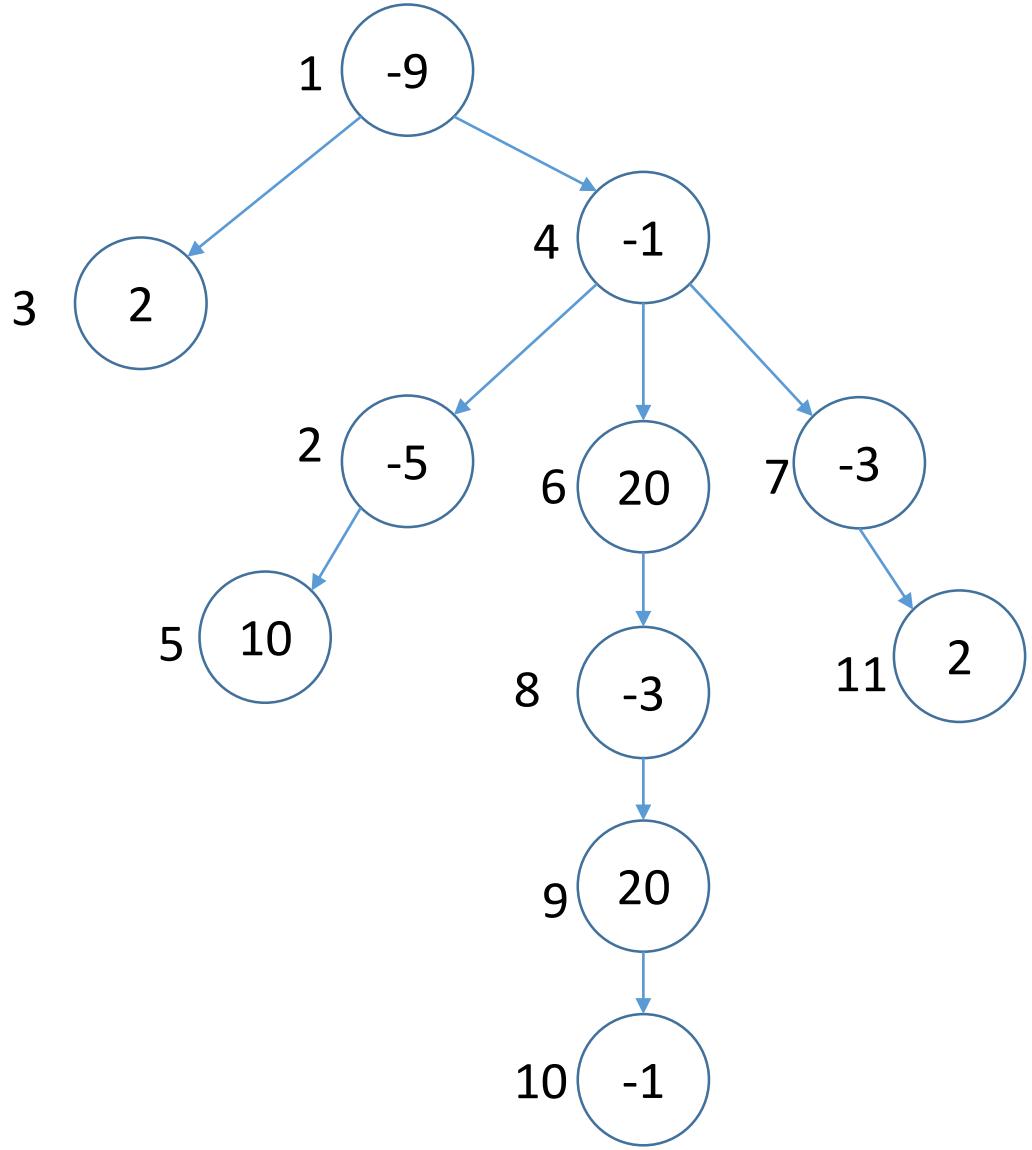
f[son[u]][i]+u[i]+f[brother[u]][x-i-1]

i:=0..x-1：包含结点v的情况。}

3.二次扫描与换根法

- 在一类无根树上问题中，需要以每个结点为根统计一些信息。如果我们暴力枚举每个点为根，假设统计复杂度是O(n) 的，那么总复杂度会达到 $O(n^2)$ 的级别，这样显然太慢了。
- 这种问题我们一般通过两次扫描、换根的方法来优化复杂度，具体分为两步：
- 第一次扫描，任选一个结点为根(一般选1)进行树形 DP，自底向上统计信息到根。d[u]:统计子树u的信息。
- 第二次扫描，从刚才选择的根1出发，进行一次 DFS 。
- f[u]是以u为树根的信息，f[1]=d[1],然后从树根1开始，父亲u，儿子v，根据f[u]求f[v] ，需要找出正确的转移方程。自上而下进行的。
- f[v]=子树v的贡献d[v]+子树v以外的节点根据f[u]转化为对对v的贡献（v变为树根）

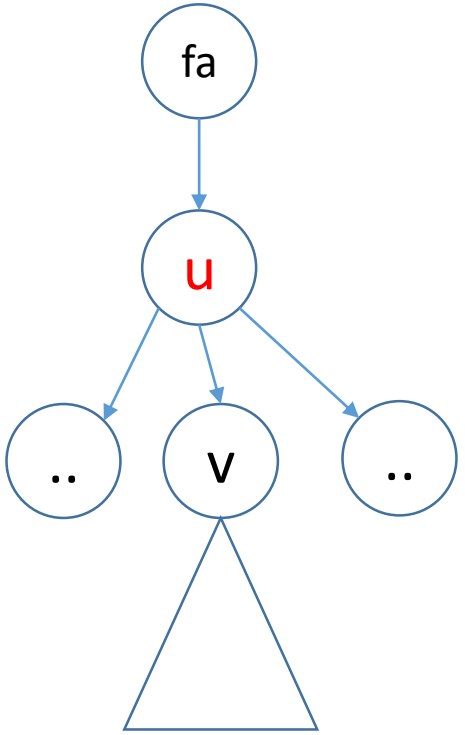

3
2
1
7
9
8
10
4
5
11
6
12


1
-9
3
2
4
-1
2
-5
6
20
7
-3
5
10
8
-3
11
2
9
20
10
-1

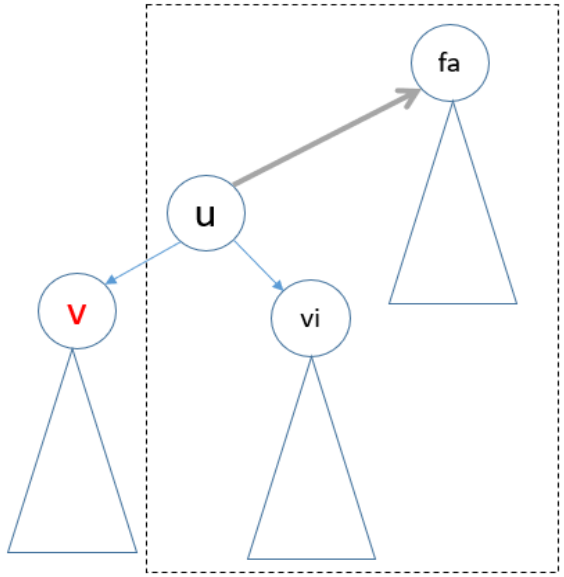
第一遍dp：自下而上求d[u]

```
void dp(int u, int fa){
     d[u]=0;siz[u]=0;//sum[u]=0;
     for(int i=h[u];i>0;i=e[i].nxt){
          int v=e[i].v;
          if(v==fa)continue;
          dp(v,u);
          siz[u]+=siz[v]; //sum[u]+=sum[u];
          d[u]+=d[v];
     }
     siz[u]++;//sum[u]+=a[u];
}

dp(1,0);
```



第二遍dfs：自上而下求f[u]




```
//一般情况f[1]=d[1]
void dfs(int u, int fa  for(int
i=h[u];i>0;i=e[i].nxt){
            int v=e[i].v;
            if(v==fa)continue;
            f[v]=? ){//根据f[u]求f[v],v变为根了
            dfs(v,u);
      }
}

  dfs(1,0);
```