Multi-University 2012 #5 A “Capturing the country” Solution

write by Gestalti Lur
2012-09-11
题目链接：http://acm.hdu.edu.cn/showproblem.php?pid=4340

题目大意

在一棵无向图上有 N(N<100)个节点，对于任意节点 i 上都有两个类型的权值 A_i 和 B_i，如果与 i 相邻的节点之前选择了 A 类型的权值那么节点 i 如果选择 A 类型的权值需要的代价就是 A_i/2,对于类型 B 亦然。

问将所有节点的权值选择完毕之后的最小权值和。

算法分析

可以想到可以将无根树通过 DFS 的方式转化为有根树进行决策，为了简便就将根设置在 1.在 DFS 找到的有根树上对于节点 i 可以确定它和它子树的无后效性的所有状态：

考虑到对于所有连通的类型 A 或者类型 B 的节点，最优的方案肯定是这些节点里只有一个节点选择了 A_i 或者 B_i,其他的节点显然都是 A_i/2 或者 B_i/2.

那么对于节点 i 来说，如果选择一种类型的权值就有三种情况：1.i 和 i 的孩子里已经选择了一个完整的某类型的权值，选择的那个完整的权值在 i 上；2.i 和 i 的孩子里已经选择了一个完整的某类型的权值，选择的权值在 i 的某个孩子上；3.i 和 i 的孩子里都只选择了一半的权值；

所以动态规划的状态可以是 f[i , s , t]表示决策到 i 时 i 选择的权值类型为 s 时的最小权值和，t=1 时表示 i 或者 i 的孩子已经选择了一个完整的 s 类型的权值，反之则表示上述情况 3.

对于 f[i , s , 1]是情况 1 和情况 2 的最小值，情况 1 可以从 i 的所有孩子中选择和 i 同样类型的权值的状态和选择和 i 相反类型的状态且已经选择了一个完整的权值的状态（因为这个孩子和 i 的类型不同所以不能够连通）中决策最小值之和，设这个值为 val1,即有：

val1 = sum(min(f[j , s , 0] , f[j , s , 1]) , f[j , ~s , 1] | j ∈ child[i]) + s[i]

对于情况 2，显然他的孩子里必然有一个是已经选择了一个完整的权值，即：

min(val1 - min(f[j , s , 0] , f[j , s , 1] , f[j , ~s , 1] | j ∈ child[i]) + f[j , s , 1]) + s[i]/2

情况 3:

sum(min(f[j , s , 0] , f[j , s , 1]) , f[j , ~s , 1] | j ∈ child[i]) + s[i]/2

边界条件为当 i 为叶子节点时,有：

f[i , s , 0] = s[i] / 2;

f[i , s , 1] = s[i];

最终答案是 min(f[1 , s , 1] , f[1 , s , 0]);

参考代码

```
/*
 MULTI UNIVERSITY #5 A
 gestapolur
 2012-09-10
 ACCEPTED
```

```
  hint: DFS + tree DP
*/
#include<cstdio>
#define MAXN 102
#define INF 1000000003

int n , m;//verx & total edges( doubling ) on tree
int head1[ MAXN ] , next1[ MAXN * 2 ] , nv[ MAXN * 2 ];//edges on oringal tree , use forward star
int head2[ MAXN ] , next2[ MAXN ];//edges selected after DFS
int f[ MAXN ][ 2 ][ 2 ] , a[ MAXN ] , b[ MAXN ] , cnt[ MAXN ];
/* f[ i , s1=A(1),B(0) , s2=0,1 ] , minimum value sum of i and i's subtree.*/
bool vis[ MAXN ];

inline void addedge1( int u , int v )
{
  ++ m;
  next1[ m ] = head1[ u ];
  nv[ m ] = v;
  head1[ u ] = m;
  return ;
}

void dfs( int u )
{
  for( int v = head1[ u ] ; v ; v = next1[ v ] )
    if( not vis[ nv[ v ] ] )
      {
      next2[ nv[ v ] ] = head2[ u ];
      head2[ u ] = nv[ v ];
      ++ cnt[ u ];
      vis[ nv[ v ] ] = true;
      dfs( nv[ v ] );
      }
  return ;
}

inline int min( int a , int b ){ return a < b ? a : b;}

void dp( int u )
{
  if( not head2[ u ] )
    {
     f[ u ][ 1 ][ 0 ] = a[ u ] / 2;
     f[ u ][ 1 ][ 1 ] = a[ u ];
     f[ u ][ 0 ][ 0 ] = b[ u ] / 2;
     f[ u ][ 0 ][ 1 ] = b[ u ];
    }
```

```
 else
   {
     int v , val1 , val2;
     for( v = head2[ u ] ; v ; v = next2[ v ] ) dp( v );
     //A
     val2 = 0;
     for( v = head2[ u ] ; v ; v = next2[ v ] )
     val2 += min( min( f[ v ][ 1 ][ 1 ] , f[ v ][ 1 ][ 0 ] ) , f[ v ][ 0 ][ 1 ] );
     f[ u ][ 1 ][ 1 ] =  val2 + a[ u ];
     for( v = head2[ u ] ; v ; v = next2[ v ] )
     {
       val1 = val2 - min( min( f[ v ][ 1 ][ 1 ] , f[ v ][ 1 ][ 0 ] ) , f[ v ][ 0 ][ 1 ] ) + f[ v ]
[ 1 ][ 1 ];
       f[ u ][ 1 ][ 1 ] = min( val1 + a[ u ] / 2 , f[ u ][ 1 ][ 1 ] );
     }
     f[ u ][ 1 ][ 0 ] = val2 + a[ u ] / 2;
     //B
     val2 = 0;
     for( v = head2[ u ] ; v ; v = next2[ v ] )
     val2 += min( f[ v ][ 1 ][ 1 ] , min( f[ v ][ 0 ][ 1 ] , f[ v ][ 0 ][ 0 ] ) );
     f[ u ][ 0 ][ 1 ] = val2 + b[ u ];
     for( v = head2[ u ] ; v ; v = next2[ v ] )
     {
       val1 = val2 - min( min( f[ v ][ 0 ][ 1 ] , f[ v ][ 0 ][ 0 ] ) , f[ v ][ 1 ][ 1 ] ) + f[ v ]
[ 0 ][ 1 ];
       f[ u ][ 0 ][ 1 ] = min( val1 + b[ u ] / 2 , f[ u ][ 0 ][ 1 ] );
     }
     f[ u ][ 0 ][ 0 ] = val2 + b[ u ] / 2;
   }
 return ;
}

void init()
{
 int i , u , v;
 m = 0;
 for( i = 1 ; i <= n ; ++ i )
   {
     f[ i ][ 0 ][ 0 ] = INF;
     f[ i ][ 0 ][ 1 ] = INF;
     f[ i ][ 1 ][ 0 ] = INF;
     f[ i ][ 1 ][ 1 ] = INF;
     head1[ i ] = 0;
     head2[ i ] = 0;
     vis[ i ] = false;
     cnt[ i ] = 0;
     scanf("%d" , &a[ i ] );
   }
```

```
 for( i = 1 ; i <= n ; ++ i )
  scanf("%d" , &b[ i ] );

 for( i = 1 ; i < n ; ++ i )
  {
   scanf("%d %d" , &u , &v );
   addedge1( u , v );
   addedge1( v , u );
  }
 vis[ 1 ] = true;
 return ;
}

int main()
{
 while( scanf( "%d" , &n ) not_eq EOF )
  {
   init();
   dfs( 1 );
   dp( 1 );
   printf( "%d\n" , min( f[ 1 ][ 1 ][ 1 ] , f[ 1 ][ 0 ][ 1 ] ) );
  }
 return 0;
}
```