树形DP

BZOJ1369.Gem

给出一棵树，要求你为树上的结点标上权值，权值可以是任意 的正整数，唯一的限制条件是相邻的两个结点不能标上相同的 权值，要求一种方案，使得整棵树的总价值最小。

结论：logn种颜色就可以染完所有的点。

```
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
using namespace std;
const int N=10007;

int n,lim,dp[N][20]; //dp[i][j]:i为根的子树，i颜色为j的权值和
vector<int>G[N];

void dfs(int x,int f){
    for(int i=1;i<=lim;++i) dp[x][i]=i;
    for(auto to:G[x]){
        if(to==f) continue;
        dfs(to,x);
        for(int j=1;j<=lim;++j){ //枚举x的颜色
            int mi=inf;
            for(int k=1;k<=lim;++k){ //枚举儿子的颜色
                if(j!=k) mi=min(mi,dp[to][k]);
            }
            dp[x][j]+=mi;
        }
    }
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n;
    lim=log2(n)+1;
    for(int i=1,u,v;i<n;++i){
        cin>>u>>v;
        G[u].emplace_back(v);
        G[v].emplace_back(u);
    }
    dfs(1,0);
    int ans=dp[1][1];
```

```
    for(int i=2;i<=lim;++i) ans=min(ans,dp[1][i]);
    cout<<ans<<"\n";
    return 0;
}
```

Codefest 16.The Chocolate Spree

在n个点的点权树上选两条不相交的路径，使路径上的点权和最大。

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int N=1e5+7;

vector<int>G[N];
int n,dp[N][5],fa[N],a[N];
//dp0:最大两链和
//dp1:最大一条链
//dp2:到根的链+一条不相交链的最大值
//dp3:儿子结点dp1的最大值
//dp4:到根的最大链

void dfs(int x){
    dp[x][0]=dp[x][1]=dp[x][2]=dp[x][4]=a[x];
    dp[x][3]=0;
    for(auto to:G[x]){
        if(fa[x]==to) continue;
        fa[to]=x;
        dfs(to);

        dp[x][0]=max(dp[x][0],dp[x][1]+dp[to][1]); //自身的最大链加上儿子最大链
        dp[x][0]=max(dp[x][0],dp[to][0]); //儿子子树内的两条链
        dp[x][0]=max(dp[x][0],dp[x][4]+dp[to][2]); //到根最大链穿过x
        dp[x][0]=max(dp[x][0],dp[x][2]+dp[to][4]);

        dp[x][1]=max(dp[x][1],dp[to][1]);
        dp[x][1]=max(dp[x][1],dp[x][4]+dp[to][4]); //穿过根

        dp[x][2]=max(dp[x][2],dp[to][2]+a[x]); //与儿子到根的链组合
        dp[x][2]=max(dp[x][2],dp[to][4]+a[x]+dp[x][3]); //与另一个儿子链接
        dp[x][2]=max(dp[x][2],dp[x][4]+dp[to][1]);

        dp[x][3]=max(dp[x][3],dp[to][1]);
        dp[x][4]=max(dp[x][4],dp[to][4]+a[x]);
    }
}

signed main(){
    ios::sync_with_stdio();
    cin.tie(0);cout.tie(0);
    cin>>n;
    for(int i=1;i<=n;++i) cin>>a[i];
    for(int i=1,u,v;i<n;++i){
        cin>>u>>v;
        G[u].emplace_back(v);
```

```
        G[v].emplace_back(u);
    }
    dfs(1);
    cout<<dp[1][0]<<"\n";
    return 0;
}
```

P2515 [HAOI2010]软件安装（缩点+树形DP）

给定n个点代价、价值和依赖的父亲，要求选取之前必须选择其父亲，求容量M能得到的最大价值。

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e4+7;

int n,m,w[N],v[N],d[N],mp[205][205];
int dp[N][505],tmp[N][505],deg[N];

struct E{
    int u,v,nxt;
}e[N];
int head[N],cnt=1;
inline void addedge(int u,int v){
    e[++cnt]=(E){u,v,head[u]};head[u]=cnt;
}

int dfn[N],low[N],idx=0,bel[N];
int stk[N],top=0,vis[N];

inline void tarjan(int x,int ex){
    dfn[x]=low[x]=++idx;
    stk[++top]=x;
    vis[x]=1;
    for(int i=head[x];i;i=e[i].nxt){
        int to=e[i].v;
        if(i==(ex^1)) continue;
        if(!dfn[to]){
            tarjan(to,i);
            low[x]=min(low[x],low[to]);
        }else if(vis[to]) low[x]=min(low[x],dfn[to]);
    }
    if(dfn[x]==low[x]){
        int y;
        while(y=stk[top--]){
            bel[y]=x;
            vis[y]=0;
            if(x==y) break;
            w[x]+=w[y];
            v[x]+=v[y];
        }
    }
}

void dfs(int x){
    for(int j=w[x];j<=m;++j) dp[x][j]=v[x];
```

```
    for(int i=1;i<=n;++i){
        if(mp[x][i]){
            dfs(i);
            for(int j=m-w[x];j>=0;--j){ //子树一共j容量
                for(int k=0;k<=j;++k){  //给儿子分配k重量
                    dp[x][j+w[x]]=max(dp[x][j+w[x]],dp[x][j+w[x]-k]+dp[i][k]);
                }
            }
        }
    }
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n>>m;
    for(int i=0;i<=n;++i) bel[i]=i;
    for(int i=1;i<=n;++i) cin>>w[i];
    for(int i=1;i<=n;++i) cin>>v[i];
    for(int i=1;i<=n;++i){
        cin>>d[i];
        if(!d[i]) continue;
        addedge(d[i],i);
        addedge(i,d[i]);
    }
    for(int i=1;i<=n;++i){
        if(!dfn[i]){
            tarjan(i,0);
        }
    }
    for(int i=1;i<=n;++i){ //rebuild Graph
        if(bel[i]!=bel[d[i]]){
            mp[bel[d[i]]][bel[i]]=1;
            ++deg[bel[i]];
        }
    }
    for(int i=1;i<=n;++i) if(!deg[bel[i]]) mp[0][bel[i]]=1;  //dp on tree
    dfs(0);
    cout<<dp[0][m]<<"\n";
    return 0;
}
```

题解区存在利用dfs序性质的$O(nm)$做法。

```
#include<bits/stdc++.h>
#define N 110
using namespace std;
int n,m,K,sl,ans,w[N],s[N],d[N],siz[N],pre[N],sum[N],val[N],dp[N][510];
int tim,top,id[N],stk[N],vis[N],dfn[N],low[N];
int t,h[N];
struct E{
    int fr,to,nxt;
}e[N];

inline void add(int u,int v) {
```

```
    e[++t]=(E){u,v,h[u]};
    h[u]=t;
}
void tarjan(int u) {
    dfn[u]=low[u]=++tim;
    stk[++top]=u;
    vis[u]=1;
    int v;
    for(int i=h[u];i;i=e[i].nxt){
        if(!dfn[v=e[i].to]) tarjan(v),low[u]=min(low[u],low[v]);
        else if(vis[v]) low[u]=min(low[u],dfn[v]);
    }
    if(dfn[u]==low[u]){
        ++K;
        do{
            v=stk[top--];
            sum[K]+=w[v];
            val[K]+=s[v];
            vis[v]=0;
            id[v]=K;
        }while(u!=v);
    }
}
void dfs(int u){
    dfn[++tim]=u;
    siz[u]=1;
    for(int i=h[u],v;i;i=e[i].nxt){
        pre[v=e[i].to]=pre[u]+sum[u];
        dfs(v);
        siz[u]+=siz[v];
    }
}
inline void upd(int &x,int y) { if(y>x) x=y;}
signed main() {
    cin>>n>>m;
    for(int i=1;i<=n;++i) cin>>w[i];
    for(int i=1;i<=n;++i) cin>>s[i];
    for(int i=1,x;i<=n;++i){
        cin>>x;
        if(x) add(x,i);
    }
    for(int i=1;i<=n;++i) if(!dfn[i]) tarjan(i);
    top=t;
    t=0;
    fill(h+1,h+K+1,0);
    for(int i=1; i<=top; ++i)
        if(id[e[i].fr]!=id[e[i].to])
            d[id[e[i].to]]++,add(id[e[i].fr],id[e[i].to]);
    for(int i=1;i<=K;++i) if(!d[i]) add(0,i);
    tim=0;
    dfs(0);
    for(int i=1; i<=tim; ++i) {
        for(int j=pre[dfn[i]]; j<=m-sum[dfn[i]]; ++j)
            upd(dp[i+1][j+sum[dfn[i]]],dp[i][j]+val[dfn[i]]);
        for(int j=pre[dfn[i]]; j<=m; ++j)
            upd(dp[i+siz[dfn[i]]][j],dp[i][j]);
    }
    printf("%d\n",dp[tim+1][m]);
```

```
    return 0;
}
```