长链剖分
这是和重链剖分（dsu on tree，http://www.cnblogs.com/zzqsblog/p/6146916.html）一类的 trick，不过它的适用范围与 dsu on tree 不同，它适用于涉及到深度的查询。
例 1 k-th ancestor query
https://zhuanlan.zhihu.com/p/25984772
有一棵 n 个点的有根树，现在要求用 O(nlogn)的时间预处理，O(1)询问某个点的第 k 个祖先。
首先我们来定义一下长链剖分是什么。它与重链剖分十分类似，但是原来重儿子是大小最大的儿子，现在重儿子（其实不是“重”儿子，大概是 preffered child）是往下延伸到叶子节点那条链最长的儿子。
这个长链剖分的过程同重链剖分的过程类似，只要一遍 dfs 即可。
我们先证明一个性质：任意一个点的 k 级祖先所在链的链长一定大于等于 k。这个证明十分简单，因为它到祖先长度就为 k 了，如果它到祖先不是重链，链长肯定不会更短。
求出长链剖分之后我们对于每条重链的链头，设这条重链长度为 len，记录一下这条重链每个深度的点，然后记录链头往上 len 个点是哪些（显然不会影响复杂度，顶多*2）。
我们同样预处理倍增跳的数组，但是我们还要记录一下 1~n 每个数的 high-bit（最高位的 1 在哪）。
我们现在考虑向上跳 k，那么我们先跳 k 的最高位 r，然后还要跳 k-r 这么高。考虑这个点往上 r 的祖先，那么它所在重链链长>=r，注意到 k-r<r，那么我们只要在链头向上向下跳就可以了。
（其实这个例题和下面两个例题没啥关系，只是都用到了长链剖分而已）
例 2 秘术（http://cogs.pro/cogs/problem/problem.php?pid=2652）
题意：给一棵树，每个点有两个权值 a 和 b，现在要求一条长度为 m 的简单路径，使得Σai/Σbi 最小。
先吐槽一句：看到数据范围只有 3w 的时候我是震惊的.....不应该是 20w 吗.....
首先这是一个 01 分数规划的题目，按照套路，考虑二分答案，我们二分一个 p，我们就要判Σai/Σbi<=p 是否可行，那么就是要判∑(ai-pbi)<=0 是否可行。那么我们设 ci=ai-pbi，我们就要找所有长度为 m 的链中 ci 和最小的。
当然我们可以点分治，这是一个十分基础的点分治，这里不谈，但是这样就是两个 log 了（二分还有一个 log），而且也不是很好写。
我们考虑一个暴力 dp 的做法。记 fi 为 i 往下长度为 j 的链 ci 和最小为多少，那么我们可以暴力进行转移，每次考虑一个儿子 p，对于它往下的每个长度 s，可以用 fp+fi 更新答案，然后用 fp+i 的点权来更新 fi。
我们发现第一个儿子并不需要合并，如果进行合并的话太浪费了。考虑每个点 i 的第一个儿子 g，f[i]开始就是 f[g]右移一格，前面补一个 0，然后全部加上 i 的点权。
那么我们可以发现这个第一个儿子肯定是长链剖分里的重儿子最优啊。
在继续之前，首先我们先解决内存分配上的问题，这个 f 数组如果对每个点都开复杂度当然不对。
正确的分配姿势大概是这样： 复制代码
```
void all(int x,int f=0) {  pos[x]=++P;  if(son[x]) all(son[x],x);  for esb(x,e,b)  if(b!=son[x]&&b!=f) all(b,x); }
```
复制代码
我们对每个点 x，先分配一个 fx，然后再对重儿子递归分配，然后再对轻儿子递归分配。
考虑递归 dp 之后重儿子递归上来之后重儿子的 f 就没用了，然后我们可以让这些内存为 f[x]所用，而且重儿子是最深的，内存肯定正好够用，而且正好重儿子的 f 就紧挨着 fx，那么就（自动）实现了“右移一格”这个操作！
“加上 i 的点权”这个操作，如果直接 for 一遍复杂度显然是不对的，那么我们可以在每个点 x 上打一个 tag，让 f[x]+tag[x]表示真实的 f[x]。
接下来的事情就比较简单了，对于所有轻儿子我们暴力更新 f[x]即可。
我们来证明一波这个做法的复杂度，考虑我们每次都用的都是每个儿子的重链更新 dp 数组，那么如果一条向下的重链被用到了（算进了复杂度），那么就说明它不是它的父亲的重链，它的祖先就再也不会用到它了，所以复杂度就是 O(n)的了。（每个点最多被用到 O(1)次）
下面这份代码在 cogs 上跑得最快 qwq 复制代码
```
#include <iostream>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <string>
#include <bitset>
#include <vector>
#include <set>
#include <map>
#include <queue>
#include <algorithm>
#include <sstream>
#include <stack>
#include <iomanip>
using namespace std;
#define pb push_back
#define mp make_pair
typedef pair<int,int> pii;
typedef long long ll;
typedef double ld;
typedef vector<int> vi;
#define fi first
#define se second
#define FO(x) {freopen(#x".in","r",stdin);freopen(#x".out","w",stdout);}
#define Edg int M=0,fst[SZ],vb[SZ],nxt[SZ];void ad_de(int a,int b){++M;nxt[M]=fst[a];fst[a]=M;vb[M]=b;}void adde(int a,int b){ad_de(a,b);ad_de(b,a);}
#define Edgc int M=0,fst[SZ],vb[SZ],nxt[SZ],vc[SZ];void ad_de(int a,int b,int c){++M;nxt[M]=fst[a];fst[a]=M;vb[M]=b;vc[M]=c;}void adde(int a,int b,int
c){ad_de(a,b,c);ad_de(b,a,c);}
#define es(x,e) (int e=fst[x];e;e=nxt[e])
#define esb(x,e,b) (int e=fst[x],b=vb[e];e;e=nxt[e],b=vb[e])
#define VIZ {printf("digraph G{\n"); for(int i=1;i<=n;i++) for es(i,e) printf("%d->%d;\n",i,vb[e]); puts("}");}
#define VIZ2 {printf("graph G{\n"); for(int i=1;i<=n;i++) for es(i,e) if(vb[e]>=i)printf("%d--%d;\n",i,vb[e]); puts("}");}
#define SZ 555555
Edg
int n,l[SZ],r[SZ],fa[SZ],son[SZ],dep[SZ],md[SZ];
int pos[SZ],P=0; ll rs[SZ];
ld a[SZ],b[SZ],g,rst[SZ];
void gs(int x,int f=0)
{
    md[x]=dep[x];
    for esb(x,e,b)if(b!=f)
    {
        dep[b]=dep[x]+1; gs(b,x);
        if(md[b]>md[son[x]])
            son[x]=b;
        md[x]=max(md[x],md[b]);
    }
}
void all(int x,int f=0)
{
    pos[x]=++P;
    if(son[x]) all(son[x],x);
```

```
    for esb(x,e,b)
        if(b!=son[x]&&b!=f) all(b,x);
}
ld minn=1e18,tag[SZ]; int m;
void dfs(int x,int f=0)
{
    if(!son[x])
    {
        rst[pos[x]]=a[x]-b[x]*g;tag[x]=0;
        if(!m) minn=min(minn,rst[pos[x]]);
        return;
    }
    dfs(son[x],x); rst[pos[x]]=-tag[son[x]];
    tag[x]=tag[son[x]]+a[x]-b[x]*g;
    for esb(x,e,b)
    {
        if(b==son[x]||b==f) continue;
        dfs(b,x); int sr=md[b]-dep[b];
        for(int d=1;d-1<=sr&&d<=m;d++)
        {
            if(m-d<=md[x]-dep[x])
                minn=min(minn,rst[pos[b]+d-1]
                +rst[pos[x]+m-d]+tag[x]+tag[b]);
        }
        for(int d=0;d<=sr;d++)
            rst[pos[x]+d+1]=min(rst[pos[x]+d+1],
            rst[pos[b]+d]+tag[b]+a[x]-::b[x]*g-tag[x]);
    }
    if(m<=md[x]-dep[x]) minn=min(minn,rst[pos[x]+m]+tag[x]);
}
char ch,B[1<<15],*S=B,*T=B;
#define getc() (S==T&&(T=(S=B)+fread(B,1,1<<15,stdin),S==T)?0:*S++)
#define isd(c) (c>='0'&&c<='9')
int aa,bb;int F(){
    while(ch=getc(),!isd(ch)&&ch!='-');ch=='-'?aa=bb=0:(aa=ch-'0',bb=1);
    while(ch=getc(),isd(ch))aa=aa*10+ch-'0';return bb?aa:-aa;
}
#define gi F()
int main()
{
    freopen("cdcq_b.in","r",stdin);
    freopen("cdcq_b.out","w",stdout);
    n=gi; m=gi-1;
    for(int i=1;i<=n;i++) a[i]=gi;
    for(int i=1;i<=n;i++) b[i]=gi;
    if(m==-2)
    {
        ld ans=1e18;
        for(int i=1;i<=n;i++) ans=min(ans,a[i]/b[i]);
        printf("%.2lf\n",ans);
        return 0;
    }
    for(int i=1;i<n;i++) adde(gi,gi);
    gs(1); all(1);
    ld l=0,r=5e10;
    while(r-l>5e-4)
    {
        ld p=(l+r)/2; g=p;
        minn=1e18; dfs(1);
        if(minn>0) l=p; else r=p;
    }
    if(r>4.5e10) puts("-1"); else printf("%.2lf\n",l);
}
```

例 3 看门人
这是很久很久以前 wlp 模拟赛里的题（我就是这么了解到长链剖分的）。
一棵 n 个点的有边权的树，对于每个点 i 输入两个数 li 和 ri，那么你需要输出经过点 i 的，不经过 1 到 i 的父亲这条链的，长度在[li,ri]之间的，边权和最长的简单路径。n<=1000000，用特殊技巧输出。
好像这题和上一题没啥区别，只不过 m 变成了[li,ri]，那么每次我们就不能直接暴力更新，必须用一棵线段树维护一发。而且这个题显然不能点分治了。

```
#include <iostream>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <string>
#include <bitset>
#include <vector>
#include <set>
#include <map>
#include <queue>
#include <algorithm>
#include <sstream>
#include <stack>
#include <iomanip>
using namespace std;
#define pb push_back
#define mp make_pair
typedef pair<int,int> pii;
typedef long long ll;
```

```
typedef double ld;
typedef vector<int> vi;
#define fi first
#define se second
#define FO(x) {freopen(#x".in","r",stdin);freopen(#x".out","w",stdout);}
#define Edg int M=0,fst[SZ],vb[SZ],nxt[SZ];void ad_de(int a,int b){++M;nxt[M]=fst[a];fst[a]=M;vb[M]=b;}void adde(int a,int b){ad_de(a,b);ad_de(b,a);}
#define Edgc int M=0,fst[SZ],vb[SZ],nxt[SZ],vc[SZ];void ad_de(int a,int b,int c){++M;nxt[M]=fst[a];fst[a]=M;vb[M]=b;vc[M]=c;}void adde(int a,int b,int
c){ad_de(a,b,c);ad_de(b,a,c);}
#define es(x,e) (int e=fst[x];e;e=nxt[e])
#define esb(x,e,b) (int e=fst[x],b=vb[e];e;e=nxt[e],b=vb[e])
#define VIZ {printf("digraph G{\n"); for(int i=1;i<=n;i++) for es(i,e) printf("%d->%d;\n",i,vb[e]); puts("}");}
#define VIZ2 {printf("graph G{\n"); for(int i=1;i<=n;i++) for es(i,e) if(vb[e]>=i)printf("%d--%d;\n",i,vb[e]); puts("}");}
#define SZ 1234567
Edg
int n,l[SZ],r[SZ],fa[SZ],fv[SZ],son[SZ],dep[SZ],md[SZ];
int pos[SZ],P=0; ll rs[SZ];
const int R=1048576;
ll seg[SZ+SZ],*vv=seg+R;
void upd(int x)
{
	for(x+=R,x>>=1;x;x>>=1)
		seg[x]=max(seg[x+x],seg[x+x+1]);
}
ll query(int l,int r)
{
	ll a=-1e18;
	for(l+=R-1,r+=R+1;l^r^1;l>>=1,r>>=1)
	{
		if(~l&1) a=max(a,seg[l^1]);
		if(r&1) a=max(a,seg[r^1]);
	}
	return a;
}
void gs(int x)
{
	md[x]=dep[x];
	for esb(x,e,b)
	{
		dep[b]=dep[x]+1;
		rs[b]=rs[x]+fv[b];
		gs(b);
		if(md[b]>md[son[x]])
			son[x]=b;
		md[x]=max(md[x],md[b]);
	}
}
void all(int x)
{
	pos[x]=++P;
	vv[P]=rs[x]; upd(P);
	if(son[x]) all(son[x]);
	for esb(x,e,b)
	{
		if(b!=son[x]) all(b);
	}
}
ll ml[SZ];
void dfs(int x)
{
	if(!son[x]) return; //gg
	//先考虑重链的作用，之后把重链与自己放在一起考虑
	{
	dfs(son[x]);
	int L=l[x],R=min(r[x],md[x]-dep[x]);
	if(L<=R) //用重链更新答案
		ml[x]=max(ml[x],query(pos[x]+L,pos[x]+R)-rs[x]);
	}
	//现在重链与自己已经一体了
	//所占的位置是[pos[x],pos[x]+md[x]-dep[x]]
	for esb(x,e,b)
	{
		if(b==son[x]) continue;
		dfs(b);
		int sr=md[b]-dep[b];
		for(int d=0;d<=sr;d++)
		{
			int L=max(l[x]-d-1,0),R=min(r[x]-d-1,md[x]-dep[x]);
			if(L>R) continue;
			ml[x]=max(ml[x],vv[pos[b]+d]+query(pos[x]+L,pos[x]+R)-rs[x]*2);
		}
		for(int d=0;d<=sr;d++)
		{
			int tg=pos[x]+d+1;
			vv[tg]=max(vv[tg],vv[pos[b]+d]);
			upd(tg);
		}
	}
}
int main()
{
	FO(watchdog)
	scanf("%d",&n);
	for(int i=1;i<=n;i++)
```

```
        scanf("%d%d",l+i,r+i), ml[i]=-1;
    for(int i=2;i<=n;i++)
        scanf("%d%d",fa+i,fv+i),
        ad_de(fa[i],i);
    gs(1); all(1); dfs(1);
    ll ans=0,MOD=998244353;
    for(int i=1;i<=n;i++)
        ans=(ans*23333+ml[i]%MOD)%MOD;
    ans=(ans%MOD+MOD)%MOD;
    printf("%d\n",int(ans));
}
```

最后总结一波：树上涉及到深度的维护题，用重链剖分就可以轻松解决。祝大家省选顺利！