Solution

7039--【11.19 模拟】数据

题解

这题考场想了很久，但还是没有做出来。

正解也不难想，容易发现对于每一个点，紧接在这个点后面的点肯定是他的儿子节点。

然后类比菊花图的情况，不难发现将这个点后面b_i/a_i最大的点接在后面是最优的。

于是做法就出来了，用堆维护b_i/a_i的值，每次找出最大的，将这个点接在他父亲的后面，然后将这两个点合并成一个点。

问题就只剩下合并后的点的权值了，用类似的方法化简一下式子，可以发现如果有一个点z要排在x和y合并成的点前面，必然有：

$$\frac{b_z}{a_z} > \frac{b_x + b_y}{a_x + a_y}$$

所以可以发现，两个合并后的a,b值就是两个点的a,b值之和。

```
#include<bits/stdc++.h>
#define ll long long
#define re register
#define ull unsigned ll
using namespace std;
inline int read()
{
    int s=0,t=1;
    char ch=getchar();
    while(ch<'0'||ch>'9'){if(ch=='-')t=-1;ch=getchar();}
    while(ch>='0'&&ch<='9')s=(s<<3)+(s<<1)+(ch^48),ch=getchar();
    return s*t;
}
const int N=3e5+5;
int n;ll Ans;
int a[N],b[N],f[N],sz[N],prt[N];
struct Node{
    int x,a,b,s;
    friend bool operator<(Node A,Node B){
        return 1ll*A.b*B.a<1ll*B.b*A.a;
    }
};
priority_queue<Node> q;
int Get(int x){return f[x]==x?x:f[x]=Get(f[x]);}
int main()
```

```
{
    n=read();
    for(int i=2;i<=n;i++)prt[i]=read();
    for(int i=1;i<=n;i++)
    {
        f[i]=i,a[i]=read(),b[i]=read(),sz[i]=1;
        q.push((Node){i,a[i],b[i],1});
    }
    while(1)
    {
        while(q.size()&&(q.top().s!=sz[Get(q.top().x)]))q.pop();
        if(q.empty())break;
        int x=q.top().x;q.pop();
        if(prt[x])
        {
            int y=Get(prt[x]);
            Ans+=1ll*b[y]*a[x];
            f[x]=y,sz[y]+=sz[x],a[y]+=a[x],b[y]+=b[x];
            q.push((Node){y,a[y],b[y],sz[y]});
        }
    }
    cout<<Ans;
    return 0;
}
```

7040--【11.19 模拟】路哥

这道题是一道树形dp，设f[i][j]表示到第i个点，此时权值和为j的期望值。

一开始打的是n^3的暴力，但是因为数据水所以拿了60分。

这道题的正解就是暴力优化，树形依赖dp。

因为选每一个点的条件是选了这个点到根节点的所有点，所以可以在dp到这个点的儿子之前，将这个点的dp值传下去，因为此时相当于是只多增加了一个点，所以可以O(k)做。

做完这个点以后，再将这个点的dp值传回去，同时计算断掉这条边的情况，这个也是O(k)的。

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#define N 5010
#define K 5010
#define ll long long
#define mo 998244353
#define er 499122177
using namespace std;
int n,k,i,f[N][K],q[N],sum[N],x,y,tot,g[N][K];
ll temp;
struct edge{int to,next;}e[N*2];
void insert(int x,int y)
{
	tot++;e[tot].to=y;e[tot].next=q[x];
	q[x]=tot;
}
void dfs(int x,int fa)
{	int i,j,l,y;
	for (i=q[x];i;i=e[i].next)
	{	y=e[i].to;
		 if (y!=fa)
		{
			for (j=0;j<=k;j++)
			{
				if (j+sum[y]<=k)
					temp=1ll*f[x][j]*er%mo,f[y][j+sum[y]]=temp;
			}
			dfs(y,x);
			for (j=0;j<=k;j++)
				temp=1ll*(f[y][j]+1ll*f[x][j]*er%mo)%mo,f[x][j]=temp;
		 }
	}
```

```
}
int main()
{
	scanf("%d%d",&n,&k);
	for (i=1;i<=n;i++) scanf("%d",&sum[i]);
	for (i=1;i<n;i++)
	{
		scanf("%d%d",&x,&y);
		insert(x,y);insert(y,x);
	}
	f[1][sum[1]]=1;
	dfs(1,0);
	printf("%d\n",f[1][k]);
	return 0;
}
```

7041--【11.19 模拟】质数

Solution

- 对于虚数的相乘，是模长相乘、极角相加的，最后是质数模长$|p|^2$，所以最多有两个数相乘形成这个质数。
- 不妨考虑对于还未到两个因子的复数暴力乘，由于势能一定，所以时间是可以满足的。
- 又因为有区间赋值，所以我们可以用一个平衡树来进行块的合并（set也可以）。
- 再考虑不贡献模长的四种数，$1,-1,i,-i$，多维护4个东西表示四个方向上的质数有哪些，以及tag还要有当前转了多少度。
- 以上可以用线段树做到两个log，看起来线段树要好打一点（
- 注意一系列tag下放对于答案的影响（至于其他若干弱智错误每一个就要花我十几分钟）。

```
#include<cstdio>
#include<cmath>
#include<algorithm>
#include<cstring>
#include<ctime>
#define maxn 200005
#define maxm 50000005
using namespace std;
int n,q,i,j,k;
struct Z{int x,y,c;Z(int _x=0,int _y=0,int _c=0){x=_x,y=_y,c=_c;}} fx[4];
Z operator*(Z a,Z b){
```

```
    if (a.c+b.c>2) return Z(0,0,3);
    return Z(a.x*b.x-a.y*b.y,a.x*b.y+a.y*b.x,a.c+b.c);
}
Z a[maxn],t[maxn*4]; int c[maxn*4],bz[maxn*4],s[maxn*4][4],tag[maxn*4];
void read(int &x)
{
    x=0; char ch=getchar(); int t=1;
    for(;(ch<'0'||ch>'9')&&ch!='-';ch=getchar());
    if (ch=='-') t=-1,ch=getchar();
    for(;ch>='0'&&ch<='9';ch=getchar()) x=x*10+ch-'0';
    x=x*t;
}
int tot,pri[maxm],isp[maxm];
void getpri()
{   isp[0]=isp[1]=1;
    for(i=2;i<maxm;i++)
    {
        if (!isp[i]) pri[++tot]=i;
        for(j=1;j<=tot&&i*pri[j]<maxm;j++)
        {
            isp[i*pri[j]]=1;
            if (i%pri[j]==0) break;
        }
    }
}
void check(int x,int l,int r)
{
    Z a=t[x]*fx[tag[x]];
    s[x][0]=s[x][1]=s[x][2]=s[x][3]=0;
    if (a.x==0&&!isp[abs(a.y)]) s[x][(a.y>0)?1:3]+=(r-l+1);
    if (a.y==0&&!isp[abs(a.x)]) s[x][(a.x>0)?0:2]+=(r-l+1);
}
void upd(int x)
{
    int l=x<<1,r=x<<1^1;
    c[x]=c[l]+c[r];
    s[x][0]=s[l][0]+s[r][0],s[x][1]=s[l][1]+s[r][1];
    s[x][2]=s[l][2]+s[r][2],s[x][3]=s[l][3]+s[r][3];
}
int tmp[4];
void downtag(int x,int l,int r)
{   if (bz[x])
    {
        int mid=(l+r)>>1;
```

```
            t[x<<1]=t[x],bz[x<<1]=1,tag[x<<1]=tag[x],c[x<<1]=(c[x]>0)*(mid-l+1);

        t[x<<1^1]=t[x],bz[x<<1^1]=1,tag[x<<1^1]=tag[x],c[x<<1^1]=(c[x]>0)*(r-mid);
            for(int k=0;k<4;k++)
            {
                s[x<<1][k]=(s[x][k]>0)*(mid-l+1);
                s[x<<1^1][k]=(s[x][k]>0)*(r-mid);
            }
            bz[x]=0,t[x]=Z(0,0,0),tag[x]=0;
        } else
        if (tag[x])
        {
            for(int k=0;k<4;k++) tmp[(k+tag[x])%4]=s[x<<1][k];
            memcpy(s[x<<1],tmp,sizeof(tmp));
            for(int k=0;k<4;k++) tmp[(k+tag[x])%4]=s[x<<1^1][k];
            memcpy(s[x<<1^1],tmp,sizeof(tmp));
            (tag[x<<1]+=tag[x])%=4,(tag[x<<1^1]+=tag[x])%=4,tag[x]=0;
        }
    }
    void maketree(int x,int l,int r)
    {    if (l==r)
        {     bz[x]=1;
            if (!a[l].x&&!a[l].y) c[x]=0;
            else{
                c[x]=1;
                if (!a[l].x&&abs(a[l].y)==1) t[x]=Z(1,0,0),tag[x]=(a[l].y<0)?3:1;
                else if (!a[l].y&&abs(a[l].x)==1) t[x]=Z(1,0,0),tag[x]=(a[l].x>0)?0:2;
                else t[x]=a[l],t[x].c=1,tag[x]=0;
            }
            check(x,l,r);
            return;
        }
        int mid=(l+r)>>1;
        maketree(x<<1,l,mid),maketree(x<<1^1,mid+1,r);
        upd(x);
    }
    void add(int x,int l,int r,int L,int R,int d)
    {
        if (l>R||r<L) return;
        if (L<=l&&r<=R)
        {
            for(int k=0;k<4;k++) tmp[(k+d)%4]=s[x][k];
            memcpy(s[x],tmp,sizeof(tmp)),(tag[x]+=d)%=4;
            return;
```

```
        }
        downtag(x,l,r);
        int mid=(l+r)>>1;
        add(x<<1,l,mid,L,R,d),add(x<<1^1,mid+1,r,L,R,d);
        upd(x);
}
void mul(int x,int l,int r,int L,int R,Z v)
{
        if (l>R||r<L) return;
        if (!c[x]) return;
        if (L<=l&&r<=R&&bz[x])
        {
                t[x]=t[x]*v,c[x]=(t[x].c<=2)*(r-l+1),check(x,l,r);
                return;
        }
        downtag(x,l,r);
        int mid=(l+r)>>1;
        mul(x<<1,l,mid,L,R,v),mul(x<<1^1,mid+1,r,L,R,v);
        upd(x);
}
void cover(int x,int l,int r,int L,int R,Z v)
{
        if (l>R||r<L) return;
        if (L<=l&&r<=R)
        {
                bz[x]=1,t[x]=v,tag[x]=0,c[x]=(t[x].c<=2)*(r-l+1);
                check(x,l,r);
                return;
        }
        downtag(x,l,r);
        int mid=(l+r)>>1;
        cover(x<<1,l,mid,L,R,v);
        cover(x<<1^1,mid+1,r,L,R,v);
        upd(x);
}
int find(int x,int l,int r,int L,int R)
{
        if (l>R||r<L) return 0;
        if (L<=l&&r<=R) return s[x][0];
        downtag(x,l,r);
        int mid=(l+r)>>1;
        return find(x<<1,l,mid,L,R)+find(x<<1^1,mid+1,r,L,R);
}
int main()
```

```
{
    read(n),read(q),getpri();
    fx[0]=Z(1,0,0),fx[1]=Z(0,1,0),fx[2]=Z(-1,0,0),fx[3]=Z(0,-1,0);
    for(i=1;i<=n;i++) read(a[i].x),read(a[i].y);
    maketree(1,1,n);
    while (q--)
    {   int opt,l,r;
        Z v;
        read(opt),read(l),read(r);
        if (opt==1)
        {
            read(v.x),read(v.y);
            if (!v.x&&!v.y) v.c=3;
            else if (!v.x&&abs(v.y)==1||!v.y&&abs(v.x)==1) v.c=0;
            else v.c=1;
            cover(1,1,n,l,r,v);
        }
        else if(opt==2)
        {
            read(v.x),read(v.y);
            if (!v.x&&!v.y) v.c=3,mul(1,1,n,l,r,v);
            else if (!v.x&&abs(v.y)==1) add(1,1,n,l,r,(v.y>0)?1:3);
            else if (!v.y&&abs(v.x)==1) add(1,1,n,l,r,(v.x>0)?0:2);
            else v.c=1,mul(1,1,n,l,r,v);
        }
        else printf("%d\n",find(1,1,n,l,r));
    }
}
```

7042--【11.19 模拟】组合

题目大意

- 给出一棵初始大小为n的树，可以如下操作：
- 选择一个选择一个点v，再新增一个点v'，将v'连向所有与v相连的点。
- 求最少的操作次数及方案使得图中存在一条哈密顿回路。
- $n \leq 100$

题解

- 哈密顿回路需要把每个点都经过一遍且只能经过一遍，除非是一条链，否则在树上都是不存在的。
- 可以发现操作相当于把每个点复制一遍， 等同于给允许这个点多经过一次，有了这个结论就容易了许多。
- 在树上DFS，每次返回到父亲就需要操作一次，但这样不能保证操作最少（当然，最后不需要回到根节点）
- 不需要返回的点构成了一条链，剩下的每个点都需要返回操作一次，那么显然当这条链最长也就是直径的时候，操作次数最少，
- 直接找出直径模拟即可。

```
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
#define N 110
int dis[N][N],p[N];
int ans[N * 2],sum,n;
void dfs(int k,int t)
{    int x=0;
     ans[++ans[0]]=k,p[k]=1;
     for(int i=1;i<=n;i++)
          if(dis[k][i]==1&&!p[i])
          {
               if(dis[t][i]+1==dis[t][k]){x=i;continue;}
              dfs(i,t);
              printf("%d ",k);
              ans[++ans[0]]=++sum;
          }
      if(x)dfs(x,t);
}
```

```
int main()
{   int i,j,k,x,y;
    scanf("%d",&n);
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            if(i==j)dis[i][j]=0;else dis[i][j]=N;
    for(i=1;i<=n-1;i++)
    {
        scanf("%d%d",&x,&y);
       dis[x][y]=dis[y][x]=1;
    }
    for(k=1;k<=n;k++)
        for(i=1;i<=n;i++)
          for(j=1;j<=n;j++)
              dis[i][j]=min(dis[i][k]+dis[j][k],dis[i][j]);
    int s=1,t=1;
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            if(dis[i][j]>dis[s][t])s=i,t=j;
    printf("%d\n",n-dis[s][t]-1);
    sum=n;
    dfs(s,t);
    printf("\n");
    for(i=1;i<=ans[0];i++)printf("%d ",ans[i]);
    return 0;
}
```