目录

1 数据结构

1.1 主席树

```
int n,m,cnt,root[maxn*40+10],a[maxn*40+10],x,y,k;
struct node
{
	int l,r,sum;
} T[maxn*25+10];
vector<int>v;
///不建议使用，建议开两个数组模拟
int getid(int x)
{
	return lower_bound(v.begin(),v.end(),x)-v.begin()+1;
}
void update(int l,int r,int &x,int y,int pos)
{
	T[++cnt]=T[y],T[cnt].sum++,x=cnt;
	if(l==r)
		return;
	int mid=l+r>>1;
	if(mid>=pos)
		update(l,mid,T[x].l,T[y].l,pos);
	else
		update(mid+1,r,T[x].r,T[y].r,pos);
}
int query(int l,int r,int x,int y,int k)
{
	if(l==r)
		return l;
	int mid=l+r>>1;
	///编号 编号！！！
	int sum=T[T[y].l].sum-T[T[x].l].sum;
	if(sum>=k)
		return query(l,mid,T[x].l,T[y].l,k);
	else
		return query(mid+1,r,T[x].r,T[y].r,k-sum);
}
int main()
{
	int _;
	scanf("%d",&_);
	while(_--)
```

```
    {
        cnt=0,v.clear();
        scanf("%d %d",&n,&m);
        for(int i=1; i<=n; i++)
            scanf("%d",&a[i]),v.push_back(a[i]);

        sort(v.begin(),v.end());
        v.erase(unique(v.begin(),v.end()),v.end());

        for(int i=1; i<=n; i++)
            update(1,n,root[i],root[i-1],getid(a[i]));

        for(int i=1; i<=m; i++)
        {
            scanf("%d %d %d",&x,&y,&k);
            printf("%d\n",v[query(1,n,root[x-1],root[y],k)-1]);
        }
    }
}
```

1.2 分块区间加及求和

```
LL a[maxn+6],pos[maxn+6],L[maxn+6],R[maxn+6],cnt,len,x,y,z,n,q;
LL sum[maxn+6],add[maxn+5];
void update(int x,int y,int z)
{
    int p=pos[x],q=pos[y];
    if(p==q){
        for(int i=x;i<=y;i++) a[i]+=z*1ll,sum[p]+=z*1ll;
        return;
    }

    for(int i=p+1;i<=q-1;i++)sum[i]+=z*1ll*(len),add[i]+=z*1ll;
    int r=R[x],l=L[y];
    for(int i=x;i<=r;i++) a[i]+=z*1ll,sum[p]+=z*1ll;
    for(int i=l;i<=y;i++) a[i]+=z*1ll,sum[q]+=z*1ll;
}
LL query(int x,int y)
{
    int p=pos[x],q=pos[y];
    LL ans=0;
    if(p==q) {
        for(int i=x;i<=y;i++){
            ans=ans+a[i]+add[p];
```

```
            }
            return ans;
        }
        for(int i=p+1;i<=q-1;i++) ans=ans+sum[i];
        int r=R[x],l=L[y];
        for(int i=x;i<=r;i++) ans=ans+a[i]+add[p];
        for(int i=l;i<=y;i++) ans=ans+a[i]+add[q];
        return ans;
}
int main()
{

        scanf("%lld %lld",&n,&q);
        len=sqrt(n);
        for(int i=1;i<=n;i++) {
            scanf("%lld",&a[i]);
            if(i%len==0){
                cnt++;
                for(int j=i-len+1;j<=i;j++){
                    pos[j]=cnt;
                    L[j]=i-len+1;R[j]=i;
                    sum[cnt]+=a[j];
                }
            }
        }
        if(n%len!=0){
            cnt++;
            for(int i=n-n%len+1;i<=n;i++){
                pos[i]=cnt;
                L[i]=n-n%len+1;R[i]=n;
                sum[cnt]+=a[i];
            }
        }

        char op[10];
        while(q--){
            scanf("%s%lld%lld",op,&x,&y);
            if(op[0]=='C') scanf("%d",&z);

            if(op[0]=='C'){
                update(x,y,z);
            }
            else {
                printf("%lld\n",query(x,y));
```

```
            }
        }
}
```

1.3 字典树求前缀串出现次数

```
#include<bits/stdc++.h>
#define pb push_back
using namespace std;
typedef long long L;
char s[200001];
int    trie[200001][58],tot=0;
int ed[200001];
void init()
{
        for(int i=0;i<=tot;i++) for(int j=0;j<58;j++) trie[i][j]=0;
        for(int i=0;i<=tot;i++) ed[i]=0;
}
void add(char a[])
{
        int len=strlen(a),p=0;
        for(int i=0;i<len;i++){
                int c=a[i]-'A';
                if(!trie[p][c]){
                        trie[p][c]=++tot;
                }
                p=trie[p][c];
        }
        ed[p]++;
}
int query(char a[])
{
        int len=strlen(a),p=0,ans=ed[p];
        for(int i=0;i<len;i++){
                int c=a[i]-'A';
                if(!trie[p][c]){
                        return ans;
                }
                p=trie[p][c];
                ans+=ed[p];
        }
        return ans;
}
int main()
```

```
{
        int t;scanf("%d",&t);
        while(t--){
            init();
            int n;scanf("%d",&n);tot=0;
            for(int i=1;i<=n;i++){
                scanf("%s",s);
                add(s);
            }
            int q;scanf("%d",&q);
            while(q--){
                scanf("%s",s);
                printf("%d-----\n",query(s));
            }
        }
}
```

1.4 莫队求区间数字种类数

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 2e5;
int n,m,a[maxn+5],pos[maxn+5],len,L,R,Ans,num[maxn+5],ans[maxn+5];
int ma[maxn+5][30],mi[maxn+5][30],c[maxn+5];

struct node
{
    int l,r,id;
}b[maxn+5];
bool cmp(node a,node b)
{
    return pos[a.l]^pos[b.l]?pos[a.l]<pos[b.l]:((pos[a.l]&1)?a.r<b.r:a.r>b.r);
}
inline int read()
{
    char c = getchar(); int x = 0, f = 1;
    while(c < '0' || c > '9'){if(c == '-') f = -1; c = getchar();}
    while(c >= '0' && c <= '9'){x = x * 10 + (c - '0'); c = getchar();}
    return x * f;
}
void del(int x)
{
```

```
	//num[x]--;
	if(--num[x]==0) Ans--;
}
void add(int x)
{
	if(num[x]++==0) Ans++;
	//num[x]++;
}
int main()
{
	int _;_=read();
	while(_--){
		n=read();m=read();
		len=sqrt(n);
		for(int i=1;i<=n;i++) a[i]=read(),c[i]=a[i],pos[i]=i/len;

		sort(c+1,c+1+n);
		int k=unique(c+1,c+1+n)-c-1;
		for(int i=1;i<=n;i++) a[i]=lower_bound(c+1,c+k+1,a[i])-c;

		for(int i=1;i<=m;i++){
			b[i].l=read();b[i].r=read();
			b[i].id=i;
		}
		sort(b+1,b+1+m,cmp);

		L=1,R=0;Ans=0;
		for(int i=1;i<=m;i++){
			while(L<b[i].l){
				del(a[L++]);
			}
			while(L>b[i].l){
				//L--;
				add(a[--L]);
			}
			while(R<b[i].r){
				//R++;
				add(a[++R]);
			}
			while(R>b[i].r){
				del(a[R--]);
				//R--;
			}
			ans[b[i].id] = Ans;
```

```
            }
            for(int i=1;i<=m;i++){
                puts(ans[i]?"YES":"NO");
            }
        }
}
```

1.5 树状数组 区间加,区间和

```
const int maxn=2e6+6;
const int mod=998244353;
const int oo=0x3fffffff;
LL c1[maxn],c2[maxn],a[maxn],b[maxn];
int n;
void update(int x,LL w)
{
    LL ww=w*x;
    for(int i=x;i<=n;i+=i&(-i)){
        c1[i]+=w;
        c2[i]+=ww;
    }
}
LL ask(int x)
{
    LL a1=0,a2=0,y=x;
    while(x){
        a1=a1+c1[x];
        a2=a2+c2[x];
        x-=x&(-x);
    }
    return (y+1)*a1-a2;
}
LL get(int l,int r)
{
    LL w1=ask(r)-ask(l-1);
    return w1;
}
int main()
{
    int q;scanf("%d%d",&n,&q);
    for(int i=1;i<=n;i++) scanf("%lld",&a[i]);

    for(int i=1;i<=n;i++) b[i]=a[i]-a[i-1];
    for(int i=1;i<=n;i++)    update(i,b[i]);
```

```
    while(q--){
        int op,x,y,l,r;scanf("%d%d%d",&op,&l,&r);
        if(op==1) {
            scanf("%d",&x);
            update(l,x);update(r+1,-x);
        }
        else {
            printf("%lld\n",get(l,r));
        }
    }
}
```

1.6 ST 表

```
void init()
{
    for(int i=0;i<=n;i++) ma[i][0]=mi[i][0]=a[i],num[i]=0;
//注意边界，可把下面 i 从 0 开始， 仅供参考，具体看题
    for(int j=1;(1<<j)<=n;j++){
        for(int i=1;i<=n-(1<<j)+1;i++){
            ma[i][j]=max(ma[i][j-1],ma[i+(1<<(j-1))][j-1]);
            mi[i][j]=min(mi[i][j-1],mi[i+(1<<(j-1))][j-1]);
        }
    }
}
int calu(int x,int y)
{
///查询注意边界，x=0,y=n+1;
    int k=log(y-x+1)/log(2);
    int A=max(ma[x][k],ma[y-(1<<k)+1][k]);
    int B=min(mi[x][k],mi[y-(1<<k)+1][k]);
    return A-B+1;
}

void Build_2D_Sparse_Table(int n, int m){
    int i, j, k1, k2;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            f[0][0][i][j]=a[i][j];
        }
    }
    for(i = 2; i < MAXN ; i++)
        lg[i] = 1 + lg[i/2];
```

```
    for(i = 1; i <= n; i++)
        for(k2 = 1; (1 << k2) <= m; k2++)
            for(j = 1; j <= m - (1 << k2) + 1; j++)
                f[0][k2][i][j] = max(f[0][k2 - 1][i][j], f[0][k2 - 1][i][j + (1 << (k2 - 1))]);
    for(k1 = 1; (1 << k1) <= n; k1++)
        for(i = 1; i <= n - (1 << k1) + 1; i++)
            for(k2 = 0; (1 << k2) <= m; k2++)
                for(j = 1; j <= m - (1 << k2) + 1; j++)
                    f[k1][k2][i][j] = max(f[k1 - 1][k2][i][j], f[k1 - 1][k2][i + (1 << (k1 - 1))][j]);
}

int Query(int x1, int y1, int x2, int y2){
    int k1 = lg[x2 - x1 + 1], k2 = lg[y2 - y1 + 1];
    x2 = x2 - (1 << k1) + 1;
    y2 = y2 - (1 << k2) + 1;
    return max(max(f[k1][k2][x1][y1],f[k1][k2][x1][y2]),max(f[k1][k2][x2][y1],f[k1][k2][x2][y2]));
}
```

1.7 树链剖分

1.7.1 预处理

```
int sz[maxn],fa[maxn],dep[maxn],son[maxn];
vector<int>G[maxn];
void dfs1(int u,int f) {
    sz[u]=1;        dep[u]=dep[f]+1;
    fa[u]=f;        son[u]=0;
    for(auto v:G[u]) if(v!=f) {
        dfs1(v,u);
        sz[u]+=sz[v];
        if(sz[son[u]]<sz[v]) son[u]=v;
    }
}
int top[maxn],pos[maxn],seq[maxn],T;
void dfs2(int u,int tp) {
    //cout<<u<<endl;
    pos[u]=++T;
    seq[T]=u;
    top[u]=tp;
    if(son[u]) dfs2(son[u],tp);
    for(auto v:G[u]) if(v!=fa[u]&&v!=son[u]) {
        dfs2(v,v);
    }
```

```
}
```

1.7.2 点权

```
int QueryMax(int x,int y){
	int re=-inf;
	while(top[x]!=top[y]){
		if(dep[top[x]]<dep[top[y]])
			swap(x,y);
		re=max(re,treeQueryMax(1,1,n,pos[top[x]],pos[x]));
		x=fa[top[x]];
	}
	if(dep[x]<dep[y])
		swap(x,y);
	return re=max(re,treeQueryMax(1,1,n,pos[y],pos[x]));
}
///点剖，长重链
struct Ed{
	int to,w,next;
}edge[maxn];
int tot,head[maxn];
void add(int u,int v,int w=0){
	edge[++tot]={v,w,head[u]};
	head[u]=tot;
}
int n,m,rt,mod,a[maxn];
int sz[maxn],dep[maxn],fa[maxn],son[maxn];
void dfs1(int u){
	 sz[u]=1;dep[u]=dep[fa[u]]+1;
	 for(int i=head[u];i;i=edge[i].next){
		int v=edge[i].to;
		if(v==fa[u]) continue;
		fa[v]=u;
		dfs1(v);
		sz[u]+=sz[v];
		if(sz[son[u]]<sz[v]) son[u]=v;
	 }
}
int pos[maxn],top[maxn],seq[maxn];
void dfs2(int u){//cout<<u<<endl;
	static int T;
	seq[pos[u]=++T]=u;
	if(u==son[fa[u]]) top[u]=top[fa[u]];
	else top[u]=u;
```

```
    if(son[u]) dfs2(son[u]);
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa[u]||v==son[u]) continue;
        dfs2(v);
    }

}
LL sum[maxn<<1],laz[maxn<<1],len[maxn<<1];
void push_up(int rt){
    sum[rt]=(sum[ls]+sum[rs])%mod;
}
void push_down(int rt){
    if(laz[rt]){
        (laz[ls]+=laz[rt])%=mod;
        (laz[rs]+=laz[rt])%=mod;
        (sum[rs]+=1ll*len[rs]*laz[rt])%=mod;
        (sum[ls]+=1ll*len[ls]*laz[rt])%=mod;
        laz[rt]=0;
    }
}
void build(int rt,int l,int r){
     sum[rt]=laz[rt]=0;
     len[rt]=r-l+1;
     if(l==r) {
          sum[rt]=a[seq[l]];return;
     }
     int mid=(l+r)>>1;
     build(ls,l,mid);
     build(rs,mid+1,r);
     push_up(rt);
}
void update(int rt,int l,int r,int L,int R,LL val){
    if(L<=l&&r<=R){
        (laz[rt]+=val)%=mod;
        (sum[rt]+=1ll*(r-l+1)*val)%=mod;return;
    }
    push_down(rt);
    int mid=(l+r)>>1;
    if(L<=mid) update(ls,l,mid,L,R,val);
    if(R>mid) update(rs,mid+1,r,L,R,val);
    push_up(rt);
}
LL ask(int rt,int l,int r,int L,int R){
```

```
        //cout<<l<<','<<r<<endl;
        if(L<=l&&r<=R){
                return sum[rt];
        }
        push_down(rt);
        int mid=(l+r)>>1;
        LL ans=0;
        if(L<=mid) (ans+=ask(ls,l,mid,L,R))%=mod;
        if(R>mid) (ans+=ask(rs,mid+1,r,L,R))%=mod;
        push_up(rt);
        return ans;
}
void updatepath(int x,int y,LL val){
        while(top[x]!=top[y]){
                if(dep[top[x]]<dep[top[y]]) swap(x,y);
                update(1,1,n,pos[top[x]],pos[x],val);
                x=fa[top[x]];
        }
        if(dep[x]>dep[y]) swap(x,y);
        update(1,1,n,pos[x],pos[y],val);
}
LL querypath(int x,int y){
        LL ans=0;
        while(top[x]!=top[y]){
                if(dep[top[x]]<dep[top[y]]) swap(x,y);
                (ans+=ask(1,1,n,pos[top[x]],pos[x]))%=mod;
                x=fa[top[x]];
        }
        if(dep[x]>dep[y]) swap(x,y);
        (ans+=ask(1,1,n,pos[x],pos[y]))%=mod;
        return ans;
}
int op,x,y,z;
int main(){
        scanf("%d%d%d%d",&n,&m,&rt,&mod);
        for(int i=1;i<=n;i++) scanf("%d",&a[i]);
        for(int i=1,u,v;i<n;i++){
                scanf("%d%d",&u,&v);
                add(u,v);add(v,u);
        }
        dfs1(rt);
        dfs2(rt);//printf("**");
        build(1,1,n);
        while(m--){
```

```
            scanf("%d%d",&op,&x);
            if(op==1){
                scanf("%d%d",&y,&z);
                updatepath(x,y,z);
            }
            else if(op==2){
                scanf("%d",&y);
                printf("%lld\n",querypath(x,y));
            }
            else if(op==3){
                scanf("%d",&z);
                update(1,1,n,pos[x],pos[x]+sz[x]-1,z);
            }
            else {
                printf("%lld\n",ask(1,1,n,pos[x],pos[x]+sz[x]-1));
            }
        }
}
```

1.7.3 边权

```
//架空根节点 从节点 2 开始建树，n=1 特判？
//线段树维护的是每个点和其父亲节点的权值
int QueryMax(int x,int y){
    int re=-inf;
    while(top[x]!=top[y]){
        if(dep[top[x]]<dep[top[y]])
            swap(x,y);
        re=max(re,treeQueryMax(2,2,n,pos[top[x]],pos[x]));
        x=fa[top[x]];
    }
    if(x==y) return ;
    if(dep[x]<dep[y]) swap(x,y);
    return re=max(re,treeQueryMax(2,2,n,pos[son[y]],pos[x]));
}

///边剖，路径权值取反，求和，max，min
struct Edge{
    int to,next,w;
}edge[maxn];
int tot,head[maxn];
void add(int u,int v,int w){
    edge[++tot]={v,head[u],w};
    head[u]=tot;
```

```
}
int n,q;
int sz[maxn],fa[maxn],dep[maxn],son[maxn],a[maxn];
void dfs1(int u){
	sz[u]=1;dep[u]=dep[fa[u]]+1;
	for(int i=head[u];i;i=edge[i].next){
		int v=edge[i].to,w=edge[i].w;
		if(v==fa[u]) continue;
		fa[v]=u;
		a[v]=w;
		dfs1(v);
		sz[u]+=sz[v];
		if(sz[son[u]]<sz[v]) son[u]=v;
	}
}
int top[maxn],pos[maxn],seq[maxn];
void dfs2(int u){
	static int T;
	seq[pos[u]=++T]=u;
	if(u==son[fa[u]]) top[u]=top[fa[u]];
	else top[u]=u;
	if(son[u]) dfs2(son[u]);
	for(int i=head[u];i;i=edge[i].next){
		int v=edge[i].to;
		if(v==fa[u]||v==son[u]) continue;
		dfs2(v);
	}
}
int ma[maxn],mi[maxn],sum[maxn],len[maxn],la[maxn];
void push_up(int rt){
	 sum[rt]=sum[ls]+sum[rs];
	 ma[rt]=max(ma[ls],ma[rs]);
	 mi[rt]=min(mi[ls],mi[rs]);
}
void push_down(int rt){
	 if(la[rt]){
		sum[ls]=-sum[ls];
		swap(ma[ls],mi[ls]);
		ma[ls]=-ma[ls];mi[ls]=-mi[ls];
		la[ls]^=1;

		sum[rs]=-sum[rs];
		swap(ma[rs],mi[rs]);
		ma[rs]=-ma[rs];mi[rs]=-mi[rs];
```

```
            la[rs]^=1;

            la[rt]=0;
        }
}

void build(int rt,int l,int r){
        ma[rt]=-inf,mi[rt]=inf,sum[rt]=0;
        len[rt]=r-l+1;
        if(l==r){
            ma[rt]=mi[rt]=sum[rt]=a[seq[l]];
            return ;
        }
        int mid=(l+r)>>1;
        build(ls,l,mid);
        build(rs,mid+1,r);
        push_up(rt);
}

void updNeg(int rt,int l,int r,int ql,int qr){
       if(ql<=l&&r<=qr){
            sum[rt]=-sum[rt];
            swap(ma[rt],mi[rt]);
            ma[rt]=-ma[rt];mi[rt]=-mi[rt];
            la[rt]^=1;
            return ;
       }
       push_down(rt);
       int mid=(l+r)>>1;
       if(ql<=mid) updNeg(ls,l,mid,ql,qr);
       if(qr>mid) updNeg(rs,mid+1,r,ql,qr);
       push_up(rt);
}
int asksum(int rt,int l,int r,int ql,int qr){
        if(ql<=l&&r<=qr) return sum[rt];
        push_down(rt);
        int mid=(l+r)>>1,ans=0;
        if(ql<=mid) ans+=asksum(ls,l,mid,ql,qr);
        if(qr>mid) ans+=asksum(rs,mid+1,r,ql,qr);
        push_up(rt);
        return ans;
}
int askmax(int rt,int l,int r,int ql,int qr){
        if(ql<=l&&r<=qr) return ma[rt];
```

```
        push_down(rt);
        int mid=(l+r)>>1,ans=-inf;
        if(ql<=mid) ans=max(ans,askmax(ls,l,mid,ql,qr));
        if(qr>mid) ans=max(ans,askmax(rs,mid+1,r,ql,qr));
        push_up(rt);
        return ans;
}
int askmin(int rt,int l,int r,int ql,int qr){
        if(ql<=l&&r<=qr) return mi[rt];
        push_down(rt);
        int mid=(l+r)>>1,ans=inf;
        if(ql<=mid) ans=min(ans,askmin(ls,l,mid,ql,qr));
        if(qr>mid) ans=min(ans,askmin(rs,mid+1,r,ql,qr));
        push_up(rt);
        return ans;
}
void updv(int rt,int l,int r,int pos,int val){
      if(l==r&&pos==l){
            ma[rt]=mi[rt]=sum[rt]=val;
            return ;
      }
      push_down(rt);
      int mid=(l+r)>>1;
      if(pos<=mid) updv(ls,l,mid,pos,val);
      else updv(rs,mid+1,r,pos,val);
      push_up(rt);
}

void updPathNeg(int x,int y){
      while(top[x]!=top[y]){
            if(dep[top[x]]<dep[top[y]]) swap(x,y);
            updNeg(2,2,n,pos[top[x]],pos[x]);
            x=fa[top[x]];
      }
      if(x==y) return;
      if(dep[x]>dep[y]) swap(x,y);
      updNeg(2,2,n,pos[x]+1,pos[y]);
}
int askPathSum(int x,int y){
      int ans=0;
      while(top[x]!=top[y]){
            if(dep[top[x]]<dep[top[y]]) swap(x,y);
            ans+=asksum(2,2,n,pos[top[x]],pos[x]);
            x=fa[top[x]];
```

```
        }
        if(x==y) return ans;
        if(dep[x]>dep[y]) swap(x,y);
        ans+=asksum(2,2,n,pos[x]+1,pos[y]);
        return ans;
}
int askPathMax(int x,int y){
        int ans=-inf;
        while(top[x]!=top[y]){
                if(dep[top[x]]<dep[top[y]]) swap(x,y);
                ans=max(ans,askmax(2,2,n,pos[top[x]],pos[x]));
                x=fa[top[x]];
        }
        if(x==y) return ans;
        if(dep[x]>dep[y]) swap(x,y);
        ans=max(ans,askmax(2,2,n,pos[x]+1,pos[y]));
        return ans;
}
int askPathMin(int x,int y){
        int ans=inf;
        while(top[x]!=top[y]){
                if(dep[top[x]]<dep[top[y]]) swap(x,y);
                ans=min(ans,askmin(2,2,n,pos[top[x]],pos[x]));
                x=fa[top[x]];
        }
        if(x==y) return ans;
        if(dep[x]>dep[y]) swap(x,y);
        ans=min(ans,askmin(2,2,n,pos[x]+1,pos[y]));
        return ans;
}
char op[maxn];
int x,y;
int main(){
        scanf("%d",&n);
        for(int i=2;i<=n;i++){
                int u,v,w;scanf("%d%d%d",&u,&v,&w);
                u++;v++;
                add(u,v,w);add(v,u,w);
        }
        dfs1(1);dfs2(1);
        build(2,2,n);
        scanf("%d",&q);
        while(q--){
                scanf("%s",op);
```

```
            if(op[0]=='C'){
                scanf("%d %d",&x,&y);
                int u=edge[x*2].to,v=edge[x*2-1].to;
                if(dep[u]>dep[v]) x=u;
                else x=v;
                updv(2,2,n,pos[x],y);
            }
            else if(op[0]=='N'){
                scanf("%d%d",&x,&y);x++;y++;
                updPathNeg(x,y);
            }
            else if(op[0]=='S'){
                scanf("%d%d",&x,&y);x++;y++;
                printf("%d\n",askPathSum(x,y));
            }
            else if(op[1]=='A'){
                scanf("%d%d",&x,&y);x++;y++;
                printf("%d\n",askPathMax(x,y));
            }
            else {
                scanf("%d%d",&x,&y);x++;y++;
                printf("%d\n",askPathMin(x,y));
            }
        }
}
```

1.8 线段树动态开点

求逆序对

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
const int maxn=2e6+6;
const int inf=0x3f3f3f3f;
const int N=1010;
int root,n,lc[maxn],rc[maxn],c[maxn],cnt;
void update(int &p,int l,int r,int x,int val){
    if(!p) p=++cnt;

    if(l==r) {
        c[p]+=val;return ;
    }

    int mid=l+r>>1;
```

```
        if(x<=mid) update(lc[p],l,mid,x,val);
        else update(rc[p],mid+1,r,x,val);

        c[p]=c[lc[p]]+c[rc[p]];
}
LL ask(int p,int l,int r,int x,int y){
        if(!p) return 0;
        if(x<=l&&r<=y){
            return c[p];
        }
        int mid=l+r>>1;
        LL ans=0;
        if(x<=mid) ans+=ask(lc[p],l,mid,x,y);
        if(y>mid) ans+=ask(rc[p],mid+1,r,x,y);
        return ans;
}
int main(){
    scanf("%d",&n);
    LL ans=0,x;
    for(int i=1;i<=n;i++){
            scanf("%lld",&x);
            ans=ans+ask(root,1,1e9,x+1,1e9);
            update(root,1,1e9,x,1);
    }
    printf("%lld\n",ans);
}
```

1.9 扫描线

```
const int maxn=2e6+5;
int yy[maxn],cnt[maxn],n;
LL len[maxn];
struct node{
    int x,y,f,xx;
}a[maxn];
bool cmp(node a,node b){
    return a.xx<b.xx;
}
void push_up(int rt,int l,int r){
        if(cnt[rt]) len[rt]=yy[r+1]-yy[l];
        else len[rt]=len[rt<<1]+len[rt<<1|1];
}

void update(int rt,int l,int r,int x,int y,int val){
```

```
        if(x<=l&&r<=y) {
            cnt[rt]+=val;
            push_up(rt,l,r);
            return ;
        }
        int mid=l+r>>1;
        if(x<=mid) update(rt<<1,l,mid,x,y,val);
        if(y>mid) update(rt<<1|1,mid+1,r,x,y,val);
        push_up(rt,l,r);
}

int main(){
    scanf("%d",&n);
    for(int i=1,x1,x2,y1,y2;i<=n;i++){
        scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
        a[i].x=y1;a[i].y=y2;a[i].f=1;
        a[i+n].x=y1;a[i+n].y=y2;a[i+n].f=-1;

        a[i].xx=x1,a[i+n].xx=x2;
        yy[i]=y1,yy[i+n]=y2;
    }
    sort(yy+1,yy+1+2*n);
    int m=unique(yy+1,yy+1+2*n)-yy-1;
    sort(a+1,a+1+2*n,cmp);

    LL ans=0;
    for(int i=1;i<=2*n;i++){
        ans+=len[1]*(a[i].xx-a[i-1].xx);
        int y1=lower_bound(yy+1,yy+1+m,a[i].x)-yy;
        int y2=lower_bound(yy+1,yy+1+m,a[i].y)-yy;
        update(1,1,m,y1,y2-1,a[i].f);
    }
    printf("%lld\n",ans);
}
```

1.10 单调队列

单调队列处理固定滑窗极值

```
int n,k,a[maxn],q[maxn];
int main(){
    scanf("%d%d",&n,&k);
    for(int i=1;i<=n;i++){
        scanf("%d",&a[i]);
    }
```

```
    int l=1,r=0;
    for(int i=1;i<=n;i++){
        while(l<=r&&q[l]+k<=i) l++;
        while(l<=r&&a[q[r]]>a[i]) r--;
        q[++r]=i;
        if(i>=k) printf("%d ",a[q[l]]);
    }
    printf("\n");
    l=1;r=0;
    for(int i=1;i<=n;i++){
        while(l<=r&&q[l]+k<=i) l++;
        while(l<=r&&a[q[r]]<a[i]) r--;
        q[++r]=i;
        if(i>=k) printf("%d ",a[q[l]]);
    }
    return 0;
}
```

1.11 虚树

```
///栈中维护的一直是一条链
///不要怕麻烦 写之前考虑如何清空每次虚树所需的数组等变量
P2495 消耗战：给一颗带权树，每次询问 k 个点，问割掉一些边使这 k 个点都与根断开的最小代价，sigma（Ki）<1e5
int n,m,lgN,cnt,dfn[maxn],fa[maxn][30],h[maxn],dep[maxn],col[maxn],top,stk[maxn];
LL mi[maxn];
///mi 表示 i~跟节点中权值最小的边，

vector<pair<int,LL> >G[maxn];///原树
VI v[maxn];///虚树
bool cmp(int a,int b){
    return dfn[a]<dfn[b];///dfn 小在前
}
void dfs(int u,int f){
    fa[u][0]=f;dfn[u]=++cnt;
    dep[u]=dep[f]+1;
    for(int i=1;i<=lgN;i++){
        fa[u][i]=fa[fa[u][i-1]][i-1];///倍增
    }
    for(auto v:G[u]) {
        if(v.fi==f) continue;
        mi[v.fi]=min(mi[u],v.se);
        dfs(v.fi,u);
    }
```

```
}
int Lca(int x,int y){
    if(dep[x]<dep[y]) swap(x,y);
    for(int i=lgN;i>=0;i--){
        if(dep[fa[x][i]]>=dep[y]) x=fa[x][i];
    }
    if(x==y) return x;

    for(int i=lgN;i>=0;i--){
        if(fa[x][i]!=fa[y][i]) {
            x=fa[x][i];
            y=fa[y][i];
        }
    }
    return fa[x][0];
}
void append(int u){
    if(top<=1) {stk[++top]=u;return ;}///当前只有根节点
    int lca=Lca(u,stk[top]);///求当前 dfn 最大和次大的 lca
    if(lca==stk[top]) {stk[++top]=u;return ;}///u 为 stk[top]子节点
    while(top>1&&dfn[lca]<=dfn[stk[top-1]]) {///lca 子树下无新节点
        v[stk[top-1]].pb(stk[top]);top--;///连边
    }
    if(lca!=stk[top]){///lca 不在 stk 中 要将他加进去，并连边
        v[lca].pb(stk[top]);
        stk[top]=lca;
    }
    stk[++top]=u;///添加当前节点
}
LL calu(int u){
    LL c=0;///u 下面的标记节点与根断开的最小代价
    for(int to:v[u]){
        c+=min(calu(to),mi[to]);///
    }
    v[u].clear();///用完即清空
    if(col[u]) return mi[u];///u 为标记节点：u 上面的节点必须断开
    else return min(c,mi[u]);///否则不用必须断开，可选择
}
int main(){
    scanf("%d",&n);lgN=(int)log(n)/log(2)+2;
    for(int i=1;i<n;i++){
        LL u,v,w;scanf("%lld%lld%lld",&u,&v,&w);
        G[u].pb(mkp(v,w));///make_pair
        G[v].pb(mkp(u,w));
```

```
    }
    mi[1]=inf;
    dfs(1,0);
    scanf("%d",&m);
    while(m--){
        int k;scanf("%d",&k);
        for(int i=1;i<=k;i++) scanf("%d",&h[i]),col[h[i]]=1;
        sort(h+1,h+1+k,cmp);

        top=0;stk[++top]=1;///当根节点入栈
        for(int i=1;i<=k;i++){
            append(h[i]);///更新
        }
        while(top>1) v[stk[top-1]].pb(stk[top]),top--;///剩余节点在一条链上
        printf("%lld\n",calu(1));
        for(int i=1;i<=k;i++) col[h[i]]=0;
    }
}
```

1.12 单调栈

```
a[0]=maxn,stk[tp=1]=0;
for(int i=1;i<=n;i++){
    while(a[stk[tp]]<=a[i])r[stk[tp--]]=i-1;    ///注意等号的影响，如果相等可以扩展的话要去掉等号从跑一遍
    l[i]=stk[tp]+1,stk[++tp]=i;
}
while(tp)r[stk[tp--]]=n;
/// 作为最大值的区间 栈中维护递减
/// 注意数据范围和作为最大还是最小值的区间修改 a[0]=? 和 while 的符号即可
```

1.13 CDQ 分治

当三维都相等时会出现少算的情况，所以要先离散化，本题求的是对于 0<=d<n, F(i)=d 出现的次数 P3810

```
int n,k,a[maxn],b[maxn],c[maxn],p[maxn],q[maxn],w[maxn],v[maxn],ans[maxn],cnt[maxn];
bool cmp(int x,int y) {
    if(a[x]!=a[y]) return a[x]<a[y];
    if(b[x]!=b[y]) return b[x]<b[y];
    return c[x]<c[y];
}
```

```
void upd(int x,int val) {
	for(;x<=k;x+=x&(-x)) w[x]+=val;
}
int ask(int x) {
	int res=0;
	for(;x>0;x-=x&(-x)) res+=w[x];
	return res;
}
void cdq(int l,int r) {
	if(l>=r) return ;
	int mid=l+r>>1;
	cdq(l,mid);cdq(mid+1,r);
	for(int i=l;i<=r;i++) q[i]=p[i];
	int i=l,j=mid+1,k=l;
	for(;i<=mid&&j<=r;k++) {
		int x=q[i],y=q[j];
		if(b[x]<=b[y]) upd(c[p[k]=x],v[x]),i++;
		else cnt[y]+=ask(c[p[k]=y]),j++;
	}
	for(;j<=r;j++,k++) cnt[p[k]=q[j]]+=ask(c[q[j]]);
	for(;l<i;l++) upd(c[q[l]],-v[q[l]]);
	for(;i<=mid;i++,k++) p[k]=q[i];
}

int main() {
	//freopen("RACE input.in","r",stdin);
	scanf("%d%d",&n,&k);
	for(int i=1;i<=n;i++) {
		scanf("%d%d%d",&a[i],&b[i],&c[i]);
		p[i]=i;
	}
	sort(p+1,p+1+n,cmp);
	int m=0;
	for(int i=1;i<=n;i++) {
		int x=p[i],y=p[m];
		if(a[x]^a[y]||b[x]^b[y]||c[x]^c[y]) p[++m]=x,v[x]=1;
		else v[y]++;
	}
	cdq(1,m);
	//for(int i=1;i<=m;i++) cnt[p[i]]+=v[p[i]]-1;

	for(int i=1;i<=m;i++) ans[cnt[p[i]]+v[p[i]]-1]+=v[p[i]];
	for(int i=0;i<n;i++) printf("%d\n",ans[i]);
}
```

1.14 点分治

///树上点对距离小于等于 K
注意判断条件 v^fa&&!vis[v] 为了防止回到父亲，重心在 dfs 过程中是动态改变的
递归中 memset，和重心的选择都是有可能导致 TLE 的原因

```
struct node{int v,w; };
int n,k,ans,zx,mn,tot,vis[maxn],sz[maxn],dis[maxn];
vector<node>G[maxn];
void dfs1(int u,int fa) {///实际上是求子树大小，下面求重心时要用到
	sz[u]=1;
	for(auto v:G[u]) if(v.v^fa&&!vis[v.v]) {
		dfs1(v.v,u);sz[u]+=sz[v.v];
	}
}
void dfs2(int u,int fa,int nn) {///求重心
	int mx=nn-sz[u];
	for(auto v:G[u]) if(v.v^fa&&!vis[v.v]) {
		dfs2(v.v,u,nn);mx=max(mx,sz[v.v]);
	}
	if(mx<mn) mn=mx,zx=u;
}
void dfs3(int u,int fa,int d) {///将每个点到根的距离存储下来，另一种写法是 dis[d]++;
	dis[++tot]=d;
	for(auto v:G[u]) if(v.v^fa&&!vis[v.v]) {
		dfs3(v.v,u,d+v.w);
	}
}
int kk(int u,int fa,int d) {
	tot=0;dfs3(u,fa,d);
	sort(dis+1,dis+tot+1);
	int ans=0,i=1,j=tot;
	while(i<j) {
		if(dis[i]+dis[j]<=k) ans+=j-i,i++;
		else j--;
	}
	return ans;
}
void dfs(int u ,int fa) {
	mn=INT_MAX;
	dfs1(u,fa);dfs2(u,fa,sz[u]);
	vis[zx]=1;ans+=kk(zx,0,0);
	int t=zx;///zx 是动态的要及时存储
```

```
    for(auto v:G[t]) if(v.v^fa&&!vis[v.v]){
        ans-=kk(v.v,t,v.w);/// 容斥
        dfs(v.v,t);
    }
}
void init(int n) {
    for(int i=0;i<=n;i++) G[i].clear(),vis[i]=0;
}
int main() {
    while(scanf("%d%d",&n,&k)&&n) {
        init(n);
        for(int i=1;i<n;i++) {
            int u,v,w;scanf("%d%d%d",&u,&v,&w);
            G[u].pb({v,w});G[v].pb({u,w});
        }
        ans=0;
        dfs(1,0);
        printf("%d\n",ans);
    }
}
```

2 图论

2.1 DIJ 队列优化

```
const int oo=0x3fffffff;
int vis[2020],d[2020];
vector<pii>G[maxn];
priority_queue<PII>q;
int main()
{
    int n,m,u,v,w;scanf("%d%d",&m,&n);
    for(int i=1;i<=m;i++){
        scanf("%d %d %d",&u,&v,&w);
        G[v].push_back(make_pair(u,w));
        G[u].push_back(make_pair(v,w));
    }
    for(int i=1;i<=n;i++) d[i]=oo;
    q.push(make_pair(0,1));
    d[1]=0;
    while(!q.empty()){
        int f=q.front().second;
```

```
            q.pop();
            if(vis[f]) continue;
            vis[f]=1;
            for(int i=0;i<G[f].size();i++){
                int v=G[f][i].first;
                if(d[v]>d[f]+G[f][i].second){
                    d[v]=d[f]+G[f][i].second;
                    q.push(make_pair(-d[v],v));
                }
            }
        }
        printf("%d\n",d[n]);
}
```

2.2 普通 DIJ

```
    for(int i=1;i<=2000;i++) for(int j=1;j<=2000;j++)
        a[i][j]=(i==j?0:oo);
        for(int i=1;i<=n;i++) dis[i]=a[1][i];
    dis[1]=0;
    for(int i=1;i<n;i++){
        int c=oo,p=-1;
        for(int j=1;j<=n;j++){
            if(!vis[j]&&c>dis[j])
                c=dis[j],p=j;
        }
        if(p==-1) break;
        vis[p]=1;
        for(int j=1;j<=n;j++){
            int c=dis[p]+a[p][j];
            if(dis[j]>c) dis[j]=c;
        }
    }
```

2.3 匈牙利算法

```
//hungry 返回最大匹配
bool dfs(int x){
     for(int i=1;i<=l;i++){
        if(mp[x][i]&&!vis[i]){
            vis[i]=1;
            if(!match[i]||dfs(match[i])) {
```

```
                    match[i]=x;return true;
                }
            }
        }
        return false;
}
int hungary(){
    int ans=0;
    memset(match,0,sizeof match);
    for(int i=1;i<=h;i++){
        memset(vis,0,sizeof vis);
        ans=ans+dfs(i);
    }
    return ans;
}
```

2.4 KM

```
///hdu-2255 完备匹配的最大权匹配 O（n^3）
struct KM
{
    int n;
    vector<int> G[maxn];
    long long W[maxn][maxn];
    long long lx[maxn], ly[maxn];
    int left[maxn];
    bool S[maxn], T[maxn];
    void init(int n)
    {
        this->n = n;
        for (int i = 0; i < =n; ++i)
            G[i].clear();
        for (int i = 0; i <= n; ++i)
            for (int j = 0; j <= n; ++j)
                W[i][j] = 0;
    }
    void add_edge(int u, int v, int w)
    {
        G[u].push_back(v);
        W[u][v] = w;
    }
    bool match(int u)
    {
        S[u] = true;
```

```
	for (unsigned i = 0; i < G[u].size(); ++i)
	{
		int v = G[u][i];
		if (lx[u] + ly[v] == W[u][v] && !T[v])
		{
			T[v] = true;
			if (left[v] == -1 || match(left[v]))
			{
				left[v] = u;
				return true;
			}
		}
	}
	return false;
}
void update()
{
	long long a = inf;
	for (int u = 1; u < =n; ++u) if (S[u])
	{
		for (unsigned i = 0; i < G[u].size(); ++i)
		{
			int v = G[u][i];
			if (!T[v]) a = min(a, lx[u] + ly[v] - W[u][v]);
		}
	}
	for (int i = 1; i < =n; ++i)
	{
		if (S[i]) lx[i] -= a;
		if (T[i]) ly[i] += a;
	}
}
long long solve()
{
	for (int i = 1; i < =n; ++i)
	{
		lx[i] = *max_element(W[i] + 1, W[i] + 1 + n);
		left[i] = -1;
		ly[i] = 0;
	}
	for (int u = 1; u < = n; ++u)
	{
		for (;;)
		{
```

```
                for (int i = 1; i < =n; ++i) S[i] = T[i] = false;
                if (match(u)) break; else update();
            }
        }
        long long ans = 0;
        for (int i = 1; i <= n; ++i) if (left[i] != -1)
            ans += W[left[i]][i];
        return ans;
    }
}km;
```

2.4.2 KM2

```
LL lx[maxn],ly[maxn],slack[maxn];
int f[maxn],pre[maxn];                          // f[y]表示右边的 y 匹配了左边的谁
bool vis[maxn];
LL KM()
{
    fo(i,1,n)
        fo(j,1,n) lx[i]=max(lx[i],mp[i][j]);    // 初始化顶标
    fo(i,1,n)
    {
        memset(slack,127,sizeof(slack));
        memset(vis,0,sizeof(vis));
        f[0]=i;                                 // 给 i 一个假的原配，统一格式
        int py=0, nextpy;
        for(; f[py]; py=nextpy)                 // 当前的左结点为 px，原配 py
        {
            int px=f[py];
            LL d=inf;
            vis[py]=1;
            fo(j,1,n) if (!vis[j])              // 松弛 slack，及找最小的 slack
            {
                if (lx[px]+ly[j]-mp[px][j]<slack[j]) slack[j]=lx[px]+ly[j]-mp[px][j], pre[j]=py;
                if (slack[j]<d) d=slack[j], nextpy=j;
            }
            fo(j,0,n) if (vis[j]) lx[f[j]]-=d, ly[j]+=d;    // 修改顶标
                else slack[j]-=d;                           // 用于松弛它的顶标已经改了，所以它也要改
        }
        for(; py; py=pre[py]) f[py]=f[pre[py]];     // 修改增广路
    }
    LL re=0;
    fo(i,1,n) re+=lx[i]+ly[i];
```

```
    return re;
}

int main()
{
    // n=max(nl,nr)，即左右点数的最大值
    // (i,j)边权为 mp[i][j]
    printf("%lld\n",KM());
}
```

2.5spfa

```
bool spfa(){
    memset(dis,inf,sizeof dis);
    dis[1]=0;
    q.push(1);is[1]=1;
    while(!q.empty()){
        int f=q.front();q.pop();
        is[f]=0;
        if(cnt[f]++>n-1) return 0;
        for(int i=0;i<vec[f].size();i++){
            int y=vec[f][i].to,w=vec[f][i].w;
            if(dis[f]+w<dis[y]){
                dis[y]=dis[f]+w;
                if(!is[y]){
                    is[y]=1;
                    q.push(y);
                }
            }
        }
    }
    return 1;
}
```

2.6 tarjin

```
///有向图，
int n,m,dfn[maxn],low[maxn],stk[maxn],vis[maxn],cnt[maxn];
int col[maxn],sum=0,dep=0,top=0,flag[maxn];
vector<int>G[maxn];
void dfs(int u){
     dfn[u]=low[u]=++dep;
```

```
        vis[u]=1;
        stk[++top]=u;
        for(int i=0;i<G[u].size();i++){
            int v=G[u][i];
            if(!dfn[v]){
                dfs(v);
                low[u]=min(low[u],low[v]);
            }
            else {
                if(vis[v]){
                    low[u]=min(low[u],dfn[v]);
                }
            }
        }
        if(dfn[u]==low[u]){
            col[u]=++sum;
            vis[u]=0;
            while(stk[top]!=u){
                col[stk[top]]=sum;
                vis[stk[top--]]=0;
            }
            top--;
        }
}
```

2.7 最大流

注意 N,M

```
struct Edge{
    int to,next; ll cap,flow;
};
struct Dinic{

    int n,s,t,cur[N],d[N],head[N],tot;
    bool vis[N]; Edge edge[M];
    void init(int _n,int _s,int _t){///点个数（从 1 开始），源点，汇点
        n=_n; s=_s; t=_t; tot=0;
        for(int i=1;i<=n;++i) head[i]=-1;
    }
    void addedge(int u,int v,ll c){
        edge[tot].to=v; edge[tot].cap=c; edge[tot].flow=0;
        edge[tot].next=head[u]; head[u]=tot++;

        edge[tot].to=u; edge[tot].cap=0; edge[tot].flow=0;
```

```
        edge[tot].next=head[v]; head[v]=tot++;
    }

    int que[N],hd,tail;
    bool bfs(){
        for(int i=1;i<=n;++i) vis[i]=false;
        hd=0; tail=-1;
        que[++tail]=s; vis[s]=true; d[s]=0;
        while(hd<=tail){
            int x=que[hd++];
            for(int i=head[x];~i;i=edge[i].next){
                Edge& e=edge[i];
                if(!vis[e.to]&&e.cap>e.flow){
                    vis[e.to]=true; d[e.to]=d[x]+1;
                    que[++tail]=e.to;
                }
            }
        }
        return vis[t];
    }

    ll dfs(int x,ll a){
        if(x==t||a==0) return a;
        ll flow=0,f;
        for(int& i=cur[x];~i;i=edge[i].next){
            Edge& e=edge[i];
            if(d[x]+1==d[e.to]&&(f=dfs(e.to,min(a,e.cap-e.flow)))>0){
                e.flow+=f; edge[i^1].flow-=f;
                flow+=f; a-=f;
                if(a==0) break;
            }
        }
        return flow;
    }

    ll Maxflow(){
        ll flow=0;
        while(bfs()){
            for(int i=1;i<=n;++i) cur[i]=head[i];
            flow+=dfs(s,inf);
        }
        return flow;
    }
}dinic;
```

3 字符串

3.1 Kmp 匹配次数

字符串数组从 1 开始 next 到 n+1 均为有效值

```
int _,n,m,nxt[maxn];
char s[maxn],p[maxn];
void kmp_pre(char x[],int m,int nxt[]) {
	nxt[1]=0;
	int i=1,j=0;
	while(i<=m) {
		while(j&&x[j]!=x[i]) j=nxt[j];
		if(x[++i]==x[++j]) nxt[i]=nxt[j]; ///有效匹配
		else nxt[i]=j;

//		nxt[++i]=++j; 找循环节写法 把上面两行删掉
	}
}
int main() {
	for(scanf("%d",&_);_;_--) {
		scanf("%s",p+1);
		scanf("%s",s+1);
		m=strlen(p+1);n=strlen(s+1);
		kmp_pre(p,m,nxt);
		int i=1,j=1,ans=0;
		while(i<=n) {
			while(j&&s[i]!=p[j]) j=nxt[j];
			i++;j++;
			if(j>m) ans++,j=nxt[j];
		}
		printf("%d\n",ans);
//		所有前缀出现次数
//		for(int i=2;i<=n+1;i++) {
//			cnt[i-1]=(cnt[nxt[i]-1]+1)%mod;
//			ans=(ans+cnt[i-1])%mod;
//		}
//		(i-1)/(i-nxt[i]) 为 1~i 的最小循环长度 (不含本身)
//		for(int i=3;i<=n+1;i++) {
//			if(nxt[i]>1&&(i-1)%(i-1-(nxt[i]-1))==0) printf("%d %d\n",i-1,(i-1)/(i-1-(nxt[i]-1)));
//		}
	}
}
```

3.2 AC 自动机

！！！指向最长相同后缀
！！！注意内存大小，字符集大小，计数的顺序(自下向上，自上向下)，以及初始化

3.2.1
自下向上

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int SZ = 26;
const int maxn = 1e6 + 5;
const int mod    = 998244353;
struct ACAM {
    int trans[maxn][SZ],fail[maxn],ed[maxn];
    int tot,root;
    vector<int>pos[maxn],G[maxn];
    int newnode() {
        for(int i=0;i<SZ;i++) trans[tot][i]=-1;
        ed[tot]=0;
        return tot++;
    }
    void init() {
        tot=0;root=newnode();
    }
    void append(char *s,int id) {
        int l=strlen(s),now=root;
        for(int i=0;i<l;i++) {
            char ch=s[i]-'a';
            if(trans[now][ch]==-1) {
                trans[now][ch]=newnode();
            }
            now=trans[now][ch];
            pos[now].push_back(id);
        }
        ed[now]++;
    }
    queue<int>q;
    void build() {
        fail[root]=root;
        for(int i=0;i<SZ;i++) {
            if(trans[root][i]==-1) {
                trans[root][i]=root;
            } else {
```

```
                fail[trans[root][i]]=root;
                q.push(trans[root][i]);
            }
        }
        while(!q.empty()) {
            int f=q.front();q.pop();
            for(int i=0;i<SZ;i++) {
                if(trans[f][i]==-1) {
                    trans[f][i]=trans[fail[f]][i];
                } else {
                    fail[trans[f][i]]=trans[fail[f]][i];
                    q.push(trans[f][i]);
                }
            }
        }
        for(int i=0;i<tot;i++) G[i].clear();
        for(int i=1;i<tot;i++) G[fail[i]].push_back(i);
    }
}ac;
int n;
char s[maxn];
int main() {
    scanf("%d",&n);
    ac.init();
    for(int i=1;i<=n;i++) {
        scanf("%s",s);
        ac.append(s,i);
    }
    ac.build();

}
```

3.2.2

Cf1237G

n 个模式串，每个串有对应的权值，

求主串的最大权值大小，主串中有 小于 14 个问号可以填互不相同的字母

Dp[j][S]:当前在 ac 自动机上 j 节点，已选字符集的状态为 S

并对每一段加速：sum[i][j]在第 j 个问号从 i 节点走到下一个问号的权值，

net[i][j]：在第 j 个问号从 i 节点走到下一个问号到达 ac 自动机上的点

```
char s[maxn+5];
LL fail[maxn+5],trie[maxn+5][15],val[maxn],tot;
queue<int>q;

struct Ac_auto
```

```
{
    void add(char *s,LL w){
        int len=strlen(s),p=0;
        for(int i=0;i<len;i++){
            int c=s[i]-'a';
            if(!trie[p][c]) trie[p][c]=++tot;
            p=trie[p][c];
        }
        val[p]+=w;
    }
    void build_fail(){
        while(!q.empty()) q.pop();
        for(int i=0;i<14;i++)
            if(trie[0][i]) q.push(trie[0][i]),vec[0].push_back(trie[0][i]);

        while(!q.empty()){
            int f=q.front();q.pop();

            for(int i=0;i<14;i++){
                if(trie[f][i]){
                    fail[trie[f][i]]=trie[fail[f]][i];
                    vec[fail[trie[f][i]]].push_back(trie[f][i]);

                    val[trie[f][i]]+=val[fail[trie[f][i]]];？///自顶向下

                    q.push(trie[f][i]);
                }
                else trie[f][i]=trie[fail[f]][i];
            }
        }
    }
}ac;
int n;
LL ans,dp[1010][(1<<14)+10],sum[1010][(1<<14)+10],net[1010][(1<<14)+10],pos[20],cnt,limit;
LL calu(int x){
    int res=0;
    while(x){
        res+=x%2;
        x/=2;
    }
    return res;
}
int main(){
    scanf("%d",&n);
```

```
    for(int i=1,w;i<=n;i++){
        scanf("%s%d",s,&w);
        ac.add(s,w);
    }
    ac.build_fail();
    scanf("%s",s);
    n=strlen(s);
    cnt=0;
    for(int i=0;i<n;i++){
        if(s[i]=='?') pos[++cnt]=i;
    }
    pos[0]=-1;pos[cnt+1]=n;

    for(int i=0;i<=tot;i++){
        for(int j=0;j<=cnt;j++){
            LL p=i,w=val[i];
            for(int k=pos[j]+1;k<=pos[j+1]-1;k++){
                p=trie[p][s[k]-'a'];
                w+=val[p];
            }
            net[i][j]=p;
            sum[i][j]=w;
        }
    }


    memset(dp,-inf,sizeof dp);
    dp[net[0][0]][0]=sum[0][0];

    ///dp p S
    limit=1<<14;
    for(int i=1;i<=cnt;i++){

        for(int S=0;S<limit;S++){
            int c1=calu(S);
            if(c1!=i-1) continue;

            for(int ch=0;ch<14;ch++) if(!((1<<ch)&S)){
                for(int pf=0;pf<=tot;pf++){

dp[net[trie[pf][ch]][i]][S|(1<<ch)]=max(dp[net[trie[pf][ch]][i]][S|(1<<ch)],dp[pf][S]+sum[trie[pf][ch]][i]);
                }
            }
```

```
            }
        }
        ans=-inf;
        for(int pf=0;pf<=tot;pf++)
            for(int S=0;S<limit;S++) if(calu(S)==cnt)
                ans=max(ans,dp[pf][S]);
        printf("%lld\n",ans);
}
```

3.3 哈希？

```
#include<bits/stdc++.h>
using namespace std;

typedef unsigned long long ull;
const int maxn = 1e5 + 5;
const int seed = 135;
const int p1 = 999999937, p2 = 19260817;
ull xp1[maxn], xp2[maxn], xp[maxn];

struct Hash {
    static void init() {
        xp1[0] = xp2[0] = xp[0] = 1;
        for (int i = 1; i < maxn; ++i) {
            xp1[i] = xp1[i - 1] * seed % p1;
            xp2[i] = xp2[i - 1] * seed % p2;
            xp[i] = xp[i - 1] * seed;
        }
    }

    ull h[maxn];
    ull build(int n, const char* s) {
        ull r1 = 0, r2 = 0;
        for (int i = 1; i <= n; i++) {
            r1 = (r1 * seed + s[i]) % p1;
            r2 = (r2 * seed + s[i]) % p2;
            h[i] = (r1 << 32) | r2;
        }
        return h[n];
    }
```

```
    ull query(int l, int r) {
        int len = r - l + 1;
        unsigned int mask32 = ~(0u);
        ull left1 = h[l - 1] >> 32, right1 = h[r] >> 32;
        ull left2 = h[l - 1] & mask32, right2 = h[r] & mask32;
        return (((right1 - left1 * xp1[len] % p1 + p1) % p1) << 32) |
               (((right2 - left2 * xp2[len] % p2 + p2) % p2));
    }
} h;

int n;
char s[maxn];
int main(){
        scanf("%d",&n);
        scanf("%s",s+1);
        Hash::init();
        h.build(n,s);
        ull c=h.query(1,1+n-1);
}
```

3.5 Manacher 前后缀最长回文长度

```
int l=0,an=0,mx=0,p=0;
ma[l++]='$';
ma[l++]='#';
for(int i=0; i<len; i++)
        ma[l++]=s[i],ma[l++]='#';
ma[l]='\0';
for(int i=0; i<l; i++){
        r[i]=mx>i?min(r[2*p-i],mx-i):1;
        for(; ma[i+r[i]]==ma[i-r[i]]; r[i]++);
        if(i+r[i]>mx)
                mx=i+r[i],p=i;
}
int ans1=0,ans2=0;
for(int i=0; i<l; i++){
        if(r[i]-i==0)
                ans1=max(ans1,r[i]-1);
        if(r[i]+i==l)
                ans2=max(ans2,r[i]-1);
}
```

3.6 PAM

len 表示当前节点对应回文子串长度
num 表示前 pam 节点(即读入 s[i]之后,s[1,…,i]的最长回文后缀对应节点 u)对应回文子串的所有不同回文后缀的个数（包括自身）
sz 表示当前节点对应回文子串在主串中出现次数(需要处理完整个主串后倒着对 failDP)
别忘初始化！！！！

```
const int SZ = 26;///字符集
const int maxn = 1e6 + 6;
struct PAM {
    struct PamNode{int fail,trans[SZ],sz,len,num;}pam[maxn];
    int tot;char s[maxn];     ll ans;///  保存的是读取到第 i 个字符时一共有多少个回文子串
    void init() {
        tot=1;
        pam[0].fail=1;pam[0].len=0;
        pam[1].fail=1;pam[1].len=-1;
        memset(pam[0].trans,0,SZ*sizeof (int));
        memset(pam[1].trans,0,SZ*sizeof (int));
    }
    inline int newnode(int len) {
        tot++;
        memset(pam[tot].trans,0,SZ*sizeof (int));
        pam[tot].len=len;pam[tot].fail=0;
        pam[tot].sz=0;pam[tot].num=0;
        return tot;
    }
    inline int getfail(int i,int u) {
        while(s[i-pam[u].len-1]^s[i]) u=pam[u].fail;
        return u;
    }
    inline int append(int i,int u) {
        int c=s[i]-'a';
        int fa=getfail(i,u);
        u=pam[fa].trans[c];
        if(!u) {
            int z=newnode(pam[fa].len+2);
            int w=getfail(i,pam[fa].fail);
            pam[z].fail=pam[w].trans[c];
            u=pam[fa].trans[c]=z;///注意这里要后更新，否则上面 getfail 时可能导致死循环
            pam[z].num=pam[pam[z].fail].num+1;
/*还有一点我们可以指出，这里属于 z 代表的回文子串第一次出现，之前读入 s[1,..,i-1]都是没有出现过该回文子串
的，所以可以维护出每种回文子串第一次出现的索引.只是这里没做而已.
```

```
*/
            }
            ans=ans+pam[u].num;
            pam[u].sz++;return u;
        }
        void calu() {
            for(int i=tot,fail;i>1;i--) {
                fail=pam[i].fail;
                pam[fail].sz+=pam[i].sz;
            }
        }
}pa;
```

前后同时插入

```
const int SZ = 26;///字符集
const int maxn = 2e5 + 6; ///开两倍
struct PAM {
    struct PamNode{int fail,trans[SZ],sz,len,num;}pam[maxn];
    int tot,n,l,r;char s[maxn];    ll ans;
    void init() {
        tot=1;          ans=0;          l=100003,r=100002;
        memset(s,-1,sizeof s);    ///特别注意
        pam[0].fail=1;    pam[0].len=0;
        pam[1].fail=1;    pam[1].len=-1;
        memset(pam[0].trans,0,SZ*sizeof (int));
        memset(pam[1].trans,0,SZ*sizeof (int));
    }
    inline int newnode(int len) {
        tot++;
        memset(pam[tot].trans,0,SZ*sizeof (int));
        pam[tot].len=len;      pam[tot].fail=0;
        pam[tot].sz=0;          pam[tot].num=0;
        return tot;
    }
    inline int L_getfail(int i,int u) {
        while(s[i+pam[u].len+1]^s[i]) u=pam[u].fail;
        return u;
    }
    inline int R_getfail(int i,int u) {
        while(s[i-pam[u].len-1]^s[i]) u=pam[u].fail;
        return u;
    }
    inline int L_append(char c,int u) {
```

```
            s[--l]=c; c=c-'a';
            int fa=L_getfail(l,u);
            u=pam[fa].trans[c];
            if(!u) {
                int z=newnode(pam[fa].len+2);
                int w=L_getfail(l,pam[fa].fail);
                pam[z].fail=pam[w].trans[c];
                u=pam[fa].trans[c]=z;///注意这里要后更新，否则上面 getfail 时可能导致死循环
                pam[z].num=pam[pam[z].fail].num+1;
            }
            ans=ans+pam[u].num;
            pam[u].sz++;return u;
        }
        inline int R_append(char c,int u) {
            s[++r]=c; c=c-'a';
            int fa=R_getfail(r,u);
            u=pam[fa].trans[c];
            if(!u) {
                int z=newnode(pam[fa].len+2);
                int w=R_getfail(r,pam[fa].fail);
                pam[z].fail=pam[w].trans[c];
                u=pam[fa].trans[c]=z;///注意这里要后更新，否则上面 getfail 时可能导致死循环
                pam[z].num=pam[pam[z].fail].num+1;
            }
            ans=ans+pam[u].num;
            pam[u].sz++;return u;
        }
        void calu() {
            for(int i=tot,fail;i>1;i--) {
                fail=pam[i].fail;
                pam[fail].sz+=pam[i].sz;
            }
        }
}pa;
int n,k;
int main() {
    while(~scanf("%d",&n)) {
        pa.init(); char op[3];
        for(int i=1,L_last=0,R_last=0;i<=n;i++) {
            scanf("%d",&k);
            if(k==1) {
                scanf("%s",op);
                L_last=pa.L_append(op[0],L_last);
                if(pa.pam[L_last].len==pa.r-pa.l+1) R_last=L_last;
```

```
            }
            if(k==2) {
                scanf("%s",op);
                R_last=pa.R_append(op[0],R_last);
                if(pa.pam[R_last].len==pa.r-pa.l+1) L_last=R_last;
            }
            if(k==3) printf("%d\n",pa.tot-1);///本质不同回文串
            if(k==4) printf("%lld\n",pa.ans);///总回文串
        }
    }
}
```

3.7 SAM

每个节点代表一系列后缀 (abcd,bcd…)
每个节点 slink 发生质变。endpos 集合发生变化
slink 树 在区分前缀节点后 可以计算每个节点所代表的 endpos 子串出现次数(不同子树 endpos 交集为空，同一子树中深度最低的节点的 endpos 为其所有子节点 endpos 集合加上有可能出现的前缀。)

trans 函数本身构成 DAG 结构。在每个字符有所带权值情况下，可以用于求一字符串所有不同子串(相同也可以先计算每个节点出现次数)的权值和。 也可以求出每个节点所代表的有效字符串种类

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int SZ = 26;
const int maxn = 1e6+5;

char s[maxn];
/*
一切和节点个数有关的数组都要开字符串长度的两倍
*/
struct SAM {
    int trans[maxn<<1][SZ],slink[maxn<<1],mi[maxn<<1],mx[maxn<<1],tot;
    int dp[maxn<<1];
    int newnode(int _mi,int _mx,int *tran,int _slink) {
        mi[tot]=_mi;            mx[tot]=_mx;
        slink[tot]=_slink;
        tran?memcpy(trans[tot],tran,SZ*sizeof(int)):memset(trans[tot],-1,SZ*sizeof(int));
        return tot++;
    }
    int init() {
```

```
    tot=0;
    return newnode(0,0,0,-1);
}
int append(int ch,int u) {
    int c=s[ch]-'a';
    int z=newnode(-1,mx[u]+1,0,-1);
    int v=u;
    while(v!=-1&&trans[v][c]==-1) {
        trans[v][c]=z;
        v=slink[v];
    }
    if(v==-1) {
        slink[z]=0;
        mi[z]=1;
    } else {
        int x=trans[v][c];
        if(mx[v]+1==mx[x]) {
            mi[z]=mx[x]+1;
            slink[z]=x;
        } else {
            int y=newnode(-1,mx[v]+1,trans[x],slink[x]);
            slink[x]=y;          slink[z]=y;
            mi[x]=mx[y]+1;    mi[z]=mx[y]+1;


            while(v!=-1&&trans[v][c]==x) {
                trans[v][c]=y;
                v=slink[v];
            }
            mi[y]=mx[slink[y]]+1;
        }
    }
    return z;
}
/// 遍历后缀
void slink_tree() {
    for(int i=1;i<tot;i++) G[i].clear();
    for(int i=1;i<tot;i++) G[slink[i]].push_back(i);
}
void dfs(int u) {
    for(int v:G[u]) {
        dfs(v);
    }
}
```

```
    /// 遍历所有子串
    void top() {
        for(int i=0;i<tot;i++) {
            for(int j=0;j<SZ;j++) if(trans[i][j]!=-1)
                in[trans[i][j]]++;
        }
        while(!q.empty()) q.pop();
        q.push(0) ;
        while(!q.empty()) {
            int f=q.front();q.pop();
            for(int i=0;i<SZ;i++) {
                int v=trans[f][j];
                if(v==-1) continue;
                in[v]--;
                if(!in[v]) q.push(v);
            }
        }
    }

}sam;

int main() {
    scanf("%s",s+1);
    int n=strlen(s+1),u=sam.init();
    for(int i=1;i<=n;i++) {
        u=sam.append(i,u);
    }

    return 0;
}}
```

3.8 最小表示法

```
int k = 0, i = 0, j = 1;
while (k < n && i < n && j < n) {
    if (sec[(i + k) % n] == sec[(j + k) % n]) {
        k++;
    } else {
        sec[(i + k) % n] > sec[(j + k) % n] ? i = i + k + 1 : j = j + k + 1;
        if (i == j) i++;
        k = 0;
    }
}
i = min(i, j);
```

4 DP

4.1 三维前缀和

```
for(int i=1;i<=n;++i)
   for(int j=1;j<=m;++j)
      for(int k=1;k<=p;++k)
         a[i][j][k]+=a[i-1][j][k];
for(int i=1;i<=n;++i)
   for(int j=1;j<=m;++j)
      for(int k=1;k<=p;++k)
         a[i][j][k]+=a[i][j-1][k];
for(int i=1;i<=n;++i)
   for(int j=1;j<=m;++j)
      for(int k=1;k<=p;++k)
         a[i][j][k]+=a[i][j][k-1];
```

4.2 子集求和

```
for(int i=0;i<w;++i)//依次枚举每个维度
{
      for(int j=0;j<(1<<w);++j)//求每个维度的前缀和
      {
            if(j&(1<<i))s[j]+=s[j^(1<<i)];
      }
}
```

4.3 数位 dp

int dfs(int pos,int pre,int sta,bool limit)//pos 当前数位,pre 上一个数字,sta：pre 是否为 6，limit 代表是否有至少有一个高位小于数位限制，如果小于的话，后面所有数位均无限制条件。

```
{
      if(pos==-1)return 1;                    //搜索到最后一位时，就说明找到了一个符合条件数
      if(!limit&&dp[pos][sta]!=-1)return dp[pos][sta];
//非 limit 表示后面的数位无特殊限制，这里产生了大量重复子问题，而 limit 条件下后面每一位都认为是特殊的，不会被重复枚举，记忆化无用。
      int up=limit?a[pos]:9;                                  //limit 条件下对当前位增加特殊限制
      int temp=0;                                                  //计数
      for(int i=0; i<=up; i++)
```

```
    {
        if(i==4)continue;
        if(pre==6&&i==2)continue;                    //跳过非法状态。
        temp+=dfs(pos-1,i,i==6,limit&&i==a[pos]);   //记忆化递归
    }
    if(!limit)dp[pos][sta]=temp;
    return temp;
}
```

4.4 位运算

Next, find the sum of numbers in which index is in each segment. For divided segment [l,r]in which length is 2^k, the values l^x,(l+1)^x,⋯(r−1)^x,r^x are, when sorted, form another segment [l^(x&~(2k − 1)),r^(x&~(2k − 1))] which x&~(2k − 1)x&~(2k − 1) means the largest of multiples of 2^knot greater than x. So you can find the sum of numbers in each divided segment ∑ ri=l(current ai) as ∑ i=l^(x&~(2k − 1))r^(x&~(2k − 1))(initial ai). Repeat this and we can find the answer for each Sum query.

x|y-y==x&(~y)
x+y=x^y+2*(x&y)

判断一个数是不是 2 的正整数次幂

```
bool isPowerOfTwo(int n) { return n > 0 && (n & (n - 1)) == 0; }
```

判断符号是否相同

```
bool isSameSign(int x, int y) {   // 有 0 的情况例外
  return (x ^ y) >= 0;
}
```

交换两个数

```
void swap(int &a, int &b) { a ^= b ^= a ^= b; }
```

遍历 u 的非空子集

```
for (int s = u; s; s = (s - 1) & u) {
```

```
    // s 是 u 的一个非空子集
}
```

内建函数

__builtin_popcount(unsigned int n) ：计算 的二进制中有多少个 1。

__builtin_parity(unsigned int n) ：判断 的二进制中 1 的个数的奇偶性。

__builtin_ffs(unsigned int n) ：计算 的二进制末尾最后一个 的位置，位置的编号从 1 开始（最低位编号为 1）。

__builtin_ctz(unsigned int n) ：计算 的二进制末尾连续 0 的个数。

__builtin_clz(unsigned int x) ：返回前导 0 的个数。

这些函数都可以在函数名末尾添加 ll （如 __builtin_popcountll ）来使参数类型变为 unsigned long long 。
例如，我们有时候希望求出一个数以二为底的对数，如果不考虑 0 的特殊情况，
就相当于这个数二进制的位数 -1 ，而一个数 n 的二进制表示的位数可以使用 32-__builtin_clz(n) 表示，
因此 31-__builtin_clz(n) 就可以求出 n 以二为底的对数。

由于这些函数是内建函数，经过了编译器的高度优化，运行速度十分快（有些甚至只需要一条指令）。

4.5 斜率優化

///FZU - 2302

```
int _, n, k;
ll a[N], b[N], s[N], dp[N][N], q[N];
ll Y(int i, int j, int k) {
    return dp[i][k] + s[i] * s[i] - (dp[j][k] + s[j] * s[j]);
}
ll X(int i, int j) {
    return s[i] - s[j];
}
int main() {
    //freopen("input.in","r",stdin);
    ios::sync_with_stdio(0);
```

```
    cin.tie(0); cout.tie(0);
    for (cin >> _; _; _--) {
        cin >> n >> k;
        for (int i = 1; i <= n; i++) cin >> b[i], b[i + n] = b[i];
        ll ans = inf;
        for (int st = 1; st <= n; st++) {
            for (int i = 1; i <= n; i++) s[i] = s[i - 1] + b[i + st - 1];

            memset(dp, 0x3f, sizeof dp);

            for (int i = 1; i <= n; i++)  dp[i][1] = s[i] * s[i];
            for (int j = 2; j <= k; j++) {
                int l = 1, r = 1;
                q[l] = j - 1;
                for (int i = j; i <= n; i++) {
                    while (l < r && Y(q[l + 1], q[l], j - 1) <= 2 * s[i] * X(q[l + 1], q[l])) l++;
                    dp[i][j] = dp[q[l]][j - 1] + (s[i] - s[q[l]]) * (s[i] - s[q[l]]);
                    while (l < r && X(q[r], q[r - 1])*Y(i, q[r - 1], j - 1) <= Y(q[r], q[r - 1], j -
1)*X(i, q[r - 1])) r--;
                    q[++r] = i;
                }
            }
            ans = min(ans, dp[n][k]);
        }
        cout << ans << '\n';
    }
    return 0;
}
```

5 数学

5.1 欧拉降幂

$$a^b \equiv \begin{cases} a^{b\%\phi(p)} & gcd(a,p)=1 \\ a^b & gcd(a,p)\neq 1, b<\phi(p) \\ a^{b\%\phi(p)+\phi(p)} & gcd(a,p)\neq 1, b\geq \phi(p) \end{cases}$$

※二三不能混用 ，eg：2^2%1e9 ，取模时必须判断大小，
函数：MOD(x,mod){return x<mod?x:x%mod+mod;} 快速幂里也必须用
w[l]^w[l+1]^....w[r]

```
ll sol(ll l,ll r,ll m) {
    if(l==r||m==1) return MOD(w[l],m);
    return qpow(w[l],sol(l+1,r,phi(m)),m);
}
```

5.2 Lucas 与组合数

```
const int MOD = 1000003;
int fact[MOD], inv[MOD];
long long pow(long long a, long long n) {
	long long res = 1;
	for (; n; n >>= 1, a = a * a % MOD) if (n & 1) res = res * a % MOD;
	return res;
}
void prepare() {
	fact[0] = 1;
	for (int i = 1; i < MOD; i++) fact[i] = (long long) fact[i - 1] * i % MOD;

	inv[MOD - 1] = pow(fact[MOD - 1], MOD - 2);
	for (int i = MOD - 2; i; i--) inv[i] = (long long) inv[i + 1] * (i + 1) % MOD;
}
int combi(int n, int m) {
	if (m > n) return 0;
	if (n < MOD) return (long long) fact[n] * inv[m] % MOD * inv[n - m] % MOD;
	return (long long) combi(n / MOD, m / MOD) * combi(n % MOD, m % MOD) % MOD;
}
int main() {
	prepare();
	return 0;
}
```

5.3 欧几里得

```
int gcd(int a, int b) {
	return b ? gcd(b, a % b) : a;
```

```
}
void exgcd(int a, int b, int &g, int &x, int &y) {
	if (b == 0) x = 1, y = 0, g = a;
	else exgcd(b, a % b, g, y, x), y -= a / b * x;
}
int main() {
	return 0;
}
```

5.4 Miller-Rabin

```
long long pow(long long a, long long n, long long mod) {
	long long res = 1;
	for (; n; n >>= 1, a = mul(a, a, mod)) if (n & 1) res = mul(res, a, mod);
	return res;
}

bool isPrime(long long n) {
	const static int primes[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};

	long long s = 0, d = n - 1;
	while (d % 2 == 0) d /= 2, s++;
	if (s == 0) return n == 2;

	for (int i = 0; i < 12 && primes[i] < n; i++) {
		long long a = primes[i];
		if (pow(a, d, n) != 1) {
			bool flag = true;
			for (int r = 0; r < s; r++)
				if (flag && pow(a, d * (1 << r), n) == n - 1) flag = false;
			if (flag) return false;
		}
	}

	return true;
}
```

5.5 海伦公式

海伦公式

$$S = \sqrt{p(p-a)(p-b)(p-c)}$$

公式描述： 公式中a，b，c分别为三角形三边长，p为半周长，S为三角形的面积。

5.6 组合

不定方程 x1+x2+...+xm=n (n,m 为正整数)的非负整数解的个数 C(n+m-1,m-1)
圆排列(n-1)!
项链数(n-1)!/2

5.7 整除分块

Sigma(n/i)

```
for(int l = 1, r; l <= n; l = r + 1)
{
	r = n / (n / l);
	ans += (r - l + 1) * (n / l);
}
```

5.8 逆元+组合数

```
	f[0]=nf[0]=nf[1]=1;
	for(int i=1;i<maxn;i++) f[i]=f[i-1]*i%mod;
	for(int i=2;i<maxn;i++) nf[i]=(LL)(mod-mod/i)*nf[mod%i]%mod;
	for(int i=1;i<maxn;i++) (nf[i]*=nf[i-1])%=mod;

	ll fac[maxn], fav[maxn];
	void init() {
		fac[0] = fav[0] = 1;
		for (int i = 1; i <= 2e5; i++) fac[i] = fac[i - 1] * i % mod;
		fav[200000] = qpow(fac[200000], mod - 2);
		for (int i = 200000 - 1; i >= 1; i--) fav[i] = fav[i + 1] * (i + 1) % mod;
	}
	ll C(int n,int m) {
		if(n<m) return 0;
		return fac[n]*fav[n-m]%mod*fav[m]%mod;
```

```
    }
```

6 其他

6.1 矩阵快速幂

```
struct MUL
{
    ll a[4][4];
    MUL operator*(const MUL &k2) const{
        MUL p;
        memset(p.a,0,sizeof p.a);
        for(int i=1;i<=3;i++){
            for(int j=1;j<=3;j++){
                for(int k=1;k<=3;k++){
                    p.a[i][j]=(p.a[i][j]+a[i][k]*k2.a[k][j]%mod)%mod;
                }
            }
        }
        return p;
    }
};


MUL qp(MUL x,ll y)
{
    MUL mu;
    for(int i=1;i<=3;i++){
        for(int j=1;j<=3;j++){
            if(i==j) mu.a[i][j]=1;
            else mu.a[i][j]=0;
        }
    }
    while(y){
        if(y&1) mu=mu*x;
        x=x*x;
        y>>=1;
    }
    return mu;
}
```

6.2 快读

```
inline char nc() {
	static char buf[100000], *p1 = buf, *p2 = buf;
	return p1 == p2 && (p2 = (p1 = buf) + fread(buf, 1, 100000, stdin), p1 == p2) ? EOF : *p1++;
}
inline int read() {
	char ch = nc(); int sum = 0;
	while (!(ch >= '0' && ch <= '9')) ch = nc();
	while (ch >= '0' && ch <= '9') sum = sum * 10 + ch - 48, ch = nc();
	return sum;
}
```

```
inline int read(){
	int x=0,f=1;char ch=getchar();
	while(ch<'0'||ch>'9') {if(ch=='-')f=-1;ch=getchar();}
	while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}
	return x*f;
}
```

6.3 MAP

```
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/hash_policy.hpp>
using namespace __gnu_pbds;

cc_hash_table<int,int> ma;
gp_hash_table<int,int> ma;///略快
```

6.4 随机数

```
mt19937 rng32(chrono::steady_clock::now().time_since_epoch().count());
mt19937 mrand(random_device{}());

int rnd(int x) { return mrand() % x;}
```

```
}
mt19937 rnd(time(NULL));
int main(){
	col[i]=rnd()%2;
```

```
{
    srand((unsigned)time(NULL));
    scanf("%d",&n);
    for(int i=1; i<=n*n; i++)
        p[i]=i;
    int cnt=0;
    while(cnt<=3)
    {
        random_shuffle(p+1,p+n*n+1);
        for(int i=1; i<=n; i++)
        {
            for(int j=1; j<=n; j++)
                printf("%d ",p[(i-1)*n+j]);
            printf("\n");
        }printf("\n");
        cnt++;
    }
}
```

6.5 Java 高精度

```
class Sqrt
{///大数开根号
    static BigDecimal ONE = BigDecimal.ONE ;
    static BigDecimal TWO = ONE.add(ONE) ;
    static BigDecimal ZERO = BigDecimal.ZERO ;
    static BigDecimal calc(BigDecimal x,int scale)
    {
        BigInteger pows = BigInteger.TEN.pow(scale) ;
        BigDecimal EPS = ONE.divide(new
BigDecimal(pows.toString()),scale+10,RoundingMode.DOWN) ;
        BigDecimal mid=ZERO,l = ZERO,r = x ;
        while(r.subtract(l).compareTo(EPS)>0) {
            mid = l.add(r).divide(TWO,scale+1,RoundingMode.DOWN) ;
            if(mid.multiply(mid).compareTo(x)>0) r = mid ;
            else l = mid ;
        }
        return mid ;
    }
};
```

```
import java.math.BigInteger;
import java.util.Scanner;

public class Main {
	public static void main(String args[]) {
		// read
		Scanner in = new Scanner(System.in);
		BigInteger a = in.nextBigInteger();

		// 3 special numbers
		a = BigInteger.ZERO;
		a = BigInteger.ONE;
		a = BigInteger.TEN;

		// convert an int to BigInteger
		BigInteger b = BigInteger.valueOf(2333);
		BigInteger p = BigInteger.valueOf(998244353);

		// convert a BigInteger to int
		int i = b.intValue();
		// convert a BigInteger to long
		long l = b.longValue();

		// operations:
		// a + b;                                   BigInteger add(BigInteger b);
		a.add(b);
		// a - b;                                   BigInteger subtract(BigInteger b);
		a.subtract(b);
		// a * b;                                   BigInteger multiply(BigInteger b);
		a.multiply(b);
		// a / b;                                   BigInteger divide(BigInteger b);
		a.divide(b);
		// a % b;                                   BigInteger mod(BigInteger b);
		a.mod(b);
		// -a;                                      BigInteger negate();
		a.negate();
		// a < 0 ? -1 : (a > 0 ? 1 : 0);            int signum();
		a.signum();
		// signum(a - b);                           int compareTo(BigInteger b);
		a.compareTo(b);
		// a == b;                                  boolean equals(BigInteger b);
		a.equals(b);

		// abs(a);                                  BigInteger abs();
```

```
a.abs();
// max(a, b);                                   BigInteger max(BigInteger b);
a.max(b);
// min(a, b);                                   BigInteger min(BigInteger b);
a.min(b);
// gcd(a, b);                                   BigInteger gcd(BigInteger b);
a.gcd(b);
// pow(a, i);                                   BigInteger pow(int i);
a.pow(i);

// modPow(a, b, p);                 BigInteger modPow(BigInteger b, BigInteger p);
a.modPow(b, p);
// modPow(a, p - 2, p);                     BigInteger modInverse(BigInteger p);
a.modInverse(p);
// isPrime(a); (probability:1 - 0.5^i)   boolean isProbablePrime(int certainty);
a.isProbablePrime(i);

// a << i;                                      BigInteger shiftLeft(int i);
a.shiftLeft(i);
// a >> i;                                      BigInteger shiftRight(int i);
a.shiftRight(i);
// a ^ b;                                       BigInteger xor(BigInteger b);
a.xor(b);
// a | b;                                       BigInteger or(BigInteger b);
a.or(b);
// a & b;                                       BigInteger and(BigInteger b);
a.and(b);
// ~a;                                          BigInteger not();
a.not();
// a & ~b;                                      BigInteger andNot(BigInteger b);
a.andNot(b);

// ((a >> i) & 1) == 1;                    BigInteger testBit(int i);
a.testBit(i);
// a | (1 << i);                              BigInteger setBit(int i);
a.setBit(i);
// a & ~(1 << i);                             BigInteger clearBit(int i);
a.clearBit(i);
// a ^ (1 << i);                              BigInteger flipBit(int i);
a.flipBit(i);
// a & -a;                                      BigInteger getLowerSetBit();
a.getLowestSetBit();
// __builtin_popcount(a);                      int bitCount();
a.bitCount();
```

```
            // ceil(log2(this < 0 ? -this : this+1)) int bitLength();
            a.bitLength();
        }
}
```

6.6 O（1）取模乘

```
long long mul(long long a, long long b, long long p) {
    return (a * b - (long long) (a / (long double) p * b + 1e-3) * p + p) % p;
}
```

6.7 二分图

二分图的最大匹配：匈牙利算法
二分图的最小点覆盖数 = 二分图最大匹配数
二分图最大独立集 = 总点数 - 二分图最大匹配数
（有向无环图）不可重叠最少路径覆盖数=原图点数 - 二分图的最大匹配

二分图多重匹配就是 在匈牙利算法上加上一维 limit 的维度（可以引入源点和汇点 进而转化成最大流问题）
二分图最大权匹配 是 条件 好像是 在完备匹配的基础上？ KM 算法 （也可通过网络流解决）
最小割=最大流：去掉一些边使源点与汇点不联通的最小花费
网格图最大流->对偶图 最短路

最大权闭合子图：有向无环图中的一个闭合子图的点权值之和最大的那个。求法是建 S,T .S 向所有点权为正的点连边权为该点点权的边，点权为负的点向 T 连接点权的相反数的边，图中原先的边增加边权为正无穷。那么最大权闭合子图的最大权为所有正权之和减去最小割（最大流）

6.8 重心

(一)
树中所有点到某个点的距离和中，到重心的距离和是最小的；如果有两个重心，那么他们的距离和一样。
(二)
把两个树通过一条边相连得到一个新的树，那么新的树的重心在连接原来两个树的重心的路径上。

(三)
把一个树添加或删除一个叶子，那么它的重心最多只移动一条边的距离。

6.9 文件

```
freopen("a.in", "r", stdin);
freopen("a.out", "w", stdout);
```

6.10 __int128

```
__uint128_t
__int128 n=1,m=1,ans=0,a,b;
inline __int128 read(){
	__int128 x=0,f=1;
	char ch=getchar();
	while(ch<'0'||ch>'9'){ if(ch=='-') f=-1; ch=getchar(); }
	while(ch>='0'&&ch<='9'){ x=x*10+ch-'0'; ch=getchar(); }
	return x*f;
}
inline void print(__int128 x) {
	if(x<0){putchar('-'); x=-x;}
	if(x>9) print(x/10);
	putchar(x%10+'0');
}
int main() {
	//freopen("RACE input.in","r",stdin);
	  a=read();
	  print(a);
}
```

6.11 杜教 BM

```
#include<bits/stdc++.h>
using namespace std;

#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())

typedef long long ll;
```

```
typedef vector<ll> VI;
typedef pair<int,int> PII;
const ll mod=998244353;
const int N=1e5+10;

ll F[N],f[N];
ll qpow(ll x,ll y){ll ans=1;x%=mod;while(y){ if(y&1) ans=ans*x%mod; x=x*x%mod; y>>=1;}return ans;}
namespace BM
{
    const int N=10010;
    ll res[N],base[N],_c[N],_md[N];
    vector<ll> Md;
    void mul(ll *a,ll *b,int k)
    {
        for(int i=0;i<k+k;i++) _c[i]=0;
        for(int i=0;i<k;i++) if (a[i]) for(int j=0;j<k;j++) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
        for (int i=k+k-1; i>=k; i--) if (_c[i])
                for(int j=0;j<Md.size();j++) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
        for(int i=0;i<k;i++) a[i]=_c[i];
    }
    int solve(ll n,VI a,VI b)
    {
        ll ans=0,pnt=0;
        int k=SZ(a);
        assert(SZ(a)==SZ(b));
        for(int i=0;i<k;i++) _md[k-1-i]=-a[i];
        _md[k]=1;
        Md.clear();
        for(int i=0;i<k;i++) if (_md[i]!=0) Md.push_back(i);
        for(int i=0;i<k;i++) res[i]=base[i]=0;
        res[0]=1;
        while ((1ll<<pnt)<=n) pnt++;
        for (int p=pnt; p>=0; p--)
        {
            mul(res,res,k);
            if ((n>>p)&1)
            {
                for (int i=k-1; i>=0; i--) res[i+1]=res[i];
                res[0]=0;
                for(int j=0;j<Md.size();j++) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
            }
        }
```

```
        for(int i=0;i<k;i++) ans=(ans+res[i]*b[i])%mod;
        if (ans<0) ans+=mod;
        return ans;
    }
    VI BM(VI s)
    {
        VI C(1,1),B(1,1);
        int L=0,m=1,b=1;
        for(int n=0;n<s.size();n++)
        {
            ll d=0;
            for(int i=0;i<=L;i++) d=(d+(ll)C[i]*s[n-i])%mod;
            if (d==0) ++m;
            else if (2*L<=n)
            {
                VI T=C;
                ll c=mod-d*qpow(b,mod-2)%mod;
                while (SZ(C)<SZ(B)+m) C.pb(0);
                for(int i=0;i<B.size();i++) C[i+m]=(C[i+m]+c*B[i])%mod;
                L=n+1-L;B=T;b=d;m=1;
            }
            else
            {
                ll c=mod-d*qpow(b,mod-2)%mod;
                while (SZ(C)<SZ(B)+m) C.pb(0);
                for(int i=0;i<B.size();i++) C[i+m]=(C[i+m]+c*B[i])%mod;
                ++m;
            }
        }
        return C;
    }
    int work(VI a,ll n)
    {
        VI c=BM(a);
        c.erase(c.begin());
        for(int i=0;i<c.size();i++) c[i]=(mod-c[i])%mod;
        return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
    }
};

int main()
{

    ll k,n;
```

```
    cin>>n>>k;
    vector<ll> tmp;
    tmp.push_back(F[1]);
    tmp.push_back(F[2]);
    for(ll i=3;i<=10000;++i) {
        ///前几项
        F[i]=?
        tmp.push_back(F[i]);
    }
    ///n<1e18
    cout<<BM::work(tmp, n - 1)<<'\n';
}
```

6.12 汉诺塔

```
Function Hanoi(n,a,b,c)
    if n==1 then
        print(a+'->'+c)
    else
        Hanoi(n-1,a,c,b)
        print(a+'->'+c)
        Hanoi(n-1,b,a,c)
    end if
end Function
```

6.13 C++ 高精度

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
struct Wint:vector<ll>
{
    bool f;
    //将低精度转高精度的初始化，可以自动被编译器调用
    //因此无需单独写高精度数和低精度数的运算函数，十分方便
    Wint(ll n=0)//默认初始化为 0，但 0 的保存形式为空
    {
        f=n>=0?0:1;
        push_back(abs(n));
        check();
    }
```

```
    Wint& check()//在各类运算中经常用到的进位小函数，不妨内置
    {
        while(!empty()&&!back()) pop_back();//去除最高位可能存在的 0
        if(empty())return *this;
        for(int i=1; i<size(); ++i)
        {
            (*this)[i]+=(*this)[i-1]/10;
            (*this)[i-1]%=10;
        }
        while(back()>=10)
        {
            push_back(back()/10);
            (*this)[size()-2]%=10;
        }
        return *this;//为使用方便，将进位后的自身返回引用
    }
};
//输入输出
istream& operator>>(istream &is,Wint &n)
{
    string s;
    is>>s;
    n.clear();
    if(s[0]=='-') {n.f=1;s.erase(0,1);}
    for(int i=s.size()-1; i>=0; --i)n.push_back(s[i]-'0');
    return is;
}
ostream& operator<<(ostream &os,const Wint &n)
{
    if(n.empty()) {os<<0;return os;}
    if(n.f) os<<'-';
    for(int i=n.size()-1; i>=0; --i)os<<n[i];
    return os;
}
//比较，只需要写两个，其他的直接代入即可
bool operator!=(const Wint &a,const Wint &b)
{
    if(a.empty()&&b.empty()) return 0;
    if(a.size()!=b.size()||a.f!=b.f)return 1;
    for(int i=a.size()-1; i>=0; --i)
        if(a[i]!=b[i])return 1;
    return 0;
}
bool operator==(const Wint &a,const Wint &b)
```

```
{
	return !(a!=b);
}
bool cmp(const Wint &a,const Wint &b) ///a 的绝对值是否小于 b
{
	if(a.size()!=b.size())return a.size()<b.size();
	for(int i=a.size()-1; i>=0; --i)
		if(a[i]!=b[i])return a[i]<b[i];
	return 0;
}
bool operator<(const Wint &a,const Wint &b)
{
	if(a.empty()&&b.empty()) return 0;
	if(a.f==1&&b.f==0) return 1;
	if(a.f==0&&b.f==1) return 0;
	return a.f==0?cmp(a,b):(!cmp(a,b));
}
bool operator>(const Wint &a,const Wint &b)
{
	return b<a;
}
bool operator<=(const Wint &a,const Wint &b)
{
	return !(a>b);
}
bool operator>=(const Wint &a,const Wint &b)
{
	return !(a<b);
}
//加法，先实现+=，这样更简洁高效
Wint add(Wint a, Wint b,bool fu)///绝对值之和
{
	a.f=fu;
	if(a.size()<b.size())a.resize(b.size());
	for(int i=0; i!=b.size(); ++i)a[i]+=b[i];
	return a.check();
}
Wint Minus(Wint a,Wint b,bool fu) ///绝对值之差
{
	if(cmp(a,b))swap(a,b);
	for(int i=0; i!=b.size(); a[i]-=b[i],++i)
		if(a[i]<b[i])//需要借位
		{
			int j=i+1;
```

```
                while(!a[j])++j;
                while(j>i)
                {
                    --a[j];
                    a[--j]+=10;
                }
            }
        a.f=fu;
        return a.check();
}
Wint operator+(Wint a,Wint b)
{
        if(a.f==b.f) return add(a,b,a.f);
        return    Minus(a,b,cmp(a,b)?b.f:a.f);

}
//减法，返回差的绝对值，由于后面有交换，故参数不用引用
Wint operator-(Wint a,Wint b)
{
        b.f^=1;
        return a+b;
}
Wint& operator+=(Wint &a,const Wint &b)
{
        return a=a+b;
}
Wint& operator-=(Wint &a,const Wint &b)
{
        return a=a-b;
}
//乘法不能先实现*=，原因自己想
Wint operator*(const Wint &a,const Wint &b)
{
        Wint n;
        n.f=a.f^b.f;
        n.assign(a.size()+b.size()-1,0);
        for(int i=0; i!=a.size(); ++i)
            for(int j=0; j!=b.size(); ++j)
                n[i+j]+=a[i]*b[j];
        return n.check();
}
Wint& operator*=(Wint &a,const Wint &b)
{
        return a=a*b;
```

```
}
//除法和取模先实现一个带余除法函数
Wint divmod(Wint &a,   Wint &b)///未修改！！！，符号
{
	Wint ans;
	bool af=a.f,bf=b.f;
   a.f=0,b.f=0;
	for(int t=a.size()-b.size(); a>=b; --t)
	{
		Wint d;
		d.assign(t+1,0);
		d.back()=1;
		Wint c=b*d;
		while(a>=c)
		{
			a-=c;
			ans+=d;
		}
	}
	ans.f=af^bf;
	a.f=af,b.f=bf;
	return ans;
}
Wint operator/(Wint a, Wint b)
{
	return divmod(a,b);
}
Wint& operator/=(Wint &a, Wint b)
{
	return a=a/b;
}
Wint& operator%=(Wint &a, Wint b)
{
	divmod(a,b);
	return a;
}
Wint operator%(Wint a, Wint b)
{
	return a%=b;
}
///快速幂
Wint quick_pow( Wint n, Wint k,Wint mod)
{
	k.f=0;
```

```
    if(k.empty())return 1;
    Wint ans=1,base=n;
    while(k>0)
    {
        if(k[0]%2==0) ans=ans%mod;
        else ans=ans*base%mod;
        base=base*base%mod;
        k/=2;
        //cout<<"k="<<k<<endl;
    }
    return ans;
}
```

7 博弈

7.1K 倍动态减法

```
#include<iostream>
#include<cstdio>
#include<cstring>
#define N 2000000
using namespace std;
int a[N],b[N];
int n,k,t,cas=0;
int main(){
    scanf("%d",&t);
    while(t--){
        scanf("%d%d",&n,&k);
        int i=0,j=0;
        a[0]=b[0]=1;
        while(a[i]<n){
            i++;
            a[i]=b[i-1]+1;
            while(a[j+1]*k<a[i])
                j++;
            if(a[j]*k<a[i])
                b[i]=b[j]+a[i];
            else
                b[i]=a[i];
        }
```

```
            printf("Case %d: ",++cas);
            if(a[i]==n)
                puts("lose");
            else{
                int ans;
                while(n){
                    if(n>=a[i]){
                        n-=a[i];
                        ans=a[i];
                    }
                    i--;
                }
                printf("%d\n",ans);
            }
        }
        return 0;
}
```

8 STL

8.1 vector

内部实现: 数组 // 就是没有固定大小的数组，vector 直接翻译是向量的意思
支持操作:
begin(), //取首个元素，返回一个 iterator
end(), //取末尾（最后一个元素的下一个存储空间的地址）
size(), //就是数组大小的意思
clear(), //清空
empty(), //判断 vector 是否为空
push_back(), pop_back() //从末尾插入或弹出
insert() O(N) //插入元素，O(n)的复杂度
erase() O(N) //删除某个元素，O(n)的复杂度

8.2 deque

类似 //双端队列，两头都支持进出
支持 push_front()和 pop_front()
是的精简版:) //栈，只支持从末尾进出
支持 push(), pop(), top()
是的精简版 //单端队列，就是我们平时所说的队列，一头进，另一头出
支持 push(), pop(), front(), back()

8.3 list

内部实现: 双向链表 //作用和 vector 差不多，但内部是用链表实现
支持操作:
begin(), end(), size(), clear(), empty()
push_back(), pop_back() //从末尾插入或删除元素
push_front(), pop_front()
insert() O(1) //链表实现，所以插入和删除的复杂度的 O(1)
erase() O(1)
sort() O(nlogn)
//不支持[ ]操作！

8.4 set

内部实现: 红黑树 //Red-Black Tree，一种平衡的二叉排序树
//又是一个 Compare 函数，类似于 qsort 函数里的那个 Compare 函数，作为红黑树在内部实现的比较方式
insert() O(logn)
erase() O(logn)
find() O(logn) 找不到返回 a.end()
lower_bound() O(logn) 查找第一个不小于 k 的元素
upper_bound() O(logn) 查找第一个大于 k 的元素
equal_range() O(logn) 返回 pair

8.6 map

内部实现: pair 组成的红黑树 //map 中文意思：印射！！
//就是很多 pair 组成一个红黑树
insert() O(logn)
erase() O(logn)
find() O(logn) 找不到返回 a.end()
lower_bound() O(logn) 查找第一个不小于 k 的元素
upper_bound() O(logn) 查找第一个大于 k 的元素
equal_range() O(logn) 返回 pair
[key]运算符 O(logn) *** //这个..太猛了，怎么说呢，数组有一个下标，如 a[i],这里 i 是 int 型的。数组可以认为是从 int 印射到另一个类型的印射，而 map 是一个任意的印射，所以 i 可以是任何类型的！

8.8 priority_queue

内部实现: 堆　　//优先队列
支持操作:
push() O(n)
pop()　O(n)
top()　O(1)

8.9 hash_map

内部实现: Hash 表//内部用哈希表实现的 map
重载 HashFcn 和 EqualKey
支持操作:
insert(); O(1)
earse();　O(1)
[ ];　　O(1)

sort()系列:

stable_sort(first,last,cmp); //稳定排序
partial_sort(first,middle,last,cmp);//部分排序
int A[12] = {7, 2, 6, 11, 9, 3, 12, 10, 8, 4, 1, 5};
partial_sort(A, A + 5, A + 12);
前 k 小，可自定义 cmp，排后无序，复杂度均摊线性。

```
bool cmp(int a,int b){
    return a<b;
}
int _,a[11];
int main(){
    for(int i=1;i<=10;i++) a[i]=10-i+1;
    nth_element(a+1,a+1+3,a+10+1,cmp);
    for(int i=1;i<=10;i++) cout<<a[i]<<',';
}
```

8.10 erase 使用

///list、set 或 map

```
std::list< int> List;
std::list< int>::iterator itList;
for( itList = List.begin(); itList != List.end(); )
{
    if( FIND( *itList) )
    {
        List.erase( itList++);
    }
    else
        itList++;
}
///vector、deque
std::vector< int> Vec;
std::vector< int>::iterator itVec;
for( itVec = Vec.begin(); itVec != Vec.end(); )
{
    if( FIND( *itVec) )
    {
          itVec = Vec.erase( itVec);
    }
    else
        itList++;
}
```

9 几何

9.1 三角形外心

三点确定一圆
已知三角形三点，求外心并计算 n 个点哪个在圆上

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
typedef vector<LL> VL;
const int maxn=2e6+6;
const double eps=1e-8;
LL _,n,a[maxn],cnt=0;
struct P{
    double x,y;
    P (double x=0,double y=0):x(x),y(y){}
    double norm2(){return x*x+y*y;}
}p[maxn],O;
double R;
```

```
double sqr(double x) {return x*x;}
int sgn(double x){return x<-eps?-1:x>eps;}
P operator + (P a,P b){return P(a.x+b.x,a.y+b.y);}
P operator - (P a,P b){return P(a.x-b.x,a.y-b.y);}
P operator * (P a,double b){return P(a.x*b,a.y*b);}
P operator / (P a,double b){return P(a.x/b,a.y/b);}
double cj(P a,P b){return a.x*b.y-a.y*b.x;}
double dis(P a,P b){return sqrt(sqr(a.x-b.x)+sqr(a.y-b.y));}
P waixin(P a,P b,P c){
	P p=b-a,q=c-a;
	P s(p.norm2()/2,q.norm2()/2);
	double d=cj(p,q);
	return a+P(cj(s,P(p.y,q.y)),cj(P(p.x,q.x),s))/d;
}

mt19937 rnd(time(NULL));
int main(){
	scanf("%lld",&_);
	while(_--){
		scanf("%lld",&n);
		for(int i=0;i<n;i++){
			scanf("%lf%lf",&p[i].x,&p[i].y);
		}
		if(n<=2){
			printf("%.6f %.6f 0\n",p[0].x,p[0].y);
		}
		else if(n<=4){
			O=(p[1]+p[2])/2;
			R=dis(O,p[1]);
			printf("%.6f %.6f %.6f\n",O.x,O.y,R);
		}
		else {
			for(;;){
				int x=rnd()%n,y=rnd()%n,z=rnd()%n;
				if(x==y||x==z||y==z) continue;
				O=waixin(p[x],p[y],p[z]);
				R=dis(O,p[x]);
				cnt=0;
				if(fabs(O.x)>1e9||fabs(O.y)>1e9) continue;
				for(int i=0;i<n;i++) if(!sgn(R-dis(O,p[i]))) cnt++;
				if(cnt>=(n+1)/2) break;
			}
			printf("%.6f %.6f %.6f\n",O.x,O.y,R);
		}
```

```
        }
    }
```