ACM-ICPC 算法模板

icpc International Collegiate Programming Contest

icpc.foundation

fold

图论

并查集

```
struct DisjointSetUnion
{
    static const int __=100005;
    int pre[__];
    DisjointSetUnion() {clear();}
    void un(int x,int y)
    {
        x=fd(x),y=fd(y);
        if(x==y)return;
        if(pre[x]>pre[y])swap(x,y);
        pre[x]+=pre[y],pre[y]=x;
    }
    int fd(int x)
    {
        if(pre[x]<0)return x;
        return pre[x]=fd(pre[x]);
    }
    void clear(){memset(pre,-1,sizeof(pre));}
}dsu;
```

最小生成树

Kruskal

```
struct MinimumSpanningTree
{
    static const int __=100005;
    int edge_num;
    struct edge
    {
        int x,y,z;
        bool operator<(const edge& b)const
        {
            return z<b.z;
        }
        void set(int _x,int _y,int _z)
        {
            x=_x,y=_y,z=_z;
        }
    }e[__<<1];

    //并查集

    void scan()
    {
        fup(i,1,edge_num)
```

```
        {
            int x,y,z;
            sf("%d%d%d",&x,&y,&z);
            e[i].set(x,y,z);
        }
    }

    void solve()
    {
        sort(e+1,e+edge_num+1);
        fup(i,1,edge_num)
        {
            if(dsu.fd(e[i].x)!=dsu.fd(e[i].y))
            {
                lca.add_edge(e[i].x,e[i].y,e[i].z);
                lca.add_edge(e[i].y,e[i].x,e[i].z);
                dsu.un(e[i].x,e[i].y);
            }
        }
    }

    void clear(){mem(dsu.pre,-1);}
}M;
```

堆优化prim

```
struct MinimumSpanningTree
{
    static const int _n_=1005,_e_=1000005;
    struct edge
    {
        int nex,dis,nex_edge;
        edge() {}
        edge(int x,int y,int z=0):
            nex(x),dis(y),nex_edge(z) {}
        bool operator<(const edge& b)const
        {
            return dis>b.dis;
        }
    }e[_e_<<1];
    int head[_n_],dis[_n_],ne;
    bool vis[_n_];
    MinimumSpanningTree(){clear();}
    void add_edge(int x,int y,int dis)
    {
        e[ne]=edge(y,dis,head[x]);
        head[x]=ne++;
    }
    int prim()
    {
        int sum=0;
        memset(vis,false,sizeof(vis));
        memset(dis,0x3f,sizeof(dis));
```

```
        priority_queue<edge>Q;
        dis[1]=0,Q.push(edge(1,0));
        while(!Q.empty())
        {
            edge t=Q.top();
            Q.pop();
            if(vis[t.nex])continue;
            sum+=t.dis;
            vis[t.nex]=true;
            for(int i=head[t.nex];~i;i=e[i].nex_edge)
                if(!vis[e[i].nex] && e[i].dis<dis[e[i].nex])
                {
                    dis[e[i].nex]=e[i].dis;
                    Q.push(edge(e[i].nex,e[i].dis));
                }
        }
        return sum;
    }
    void clear(){ne=0,memset(head,-1,sizeof(head));}
}prim;
```

最短路

Dijkstra

```
struct ShortestPath
{
    static const int _n_=100005,_e_=1000005;
    struct edge
    {
        int nex,dis,nex_edge;
        edge() {}
        edge(int x,int y,int z=0):
            nex(x),dis(y),nex_edge(z) {}
        bool operator<(const edge& b)const
        {
            return dis>b.dis;
        }
    }e[_e_<<1];
    int head[_n_],dis[_n_],ne;
    bool vis[_n_];
    ShortestPath(){clear();}
    void add_edge(int x,int y,int dis)
    {
        e[ne]=edge(y,dis,head[x]);
        head[x]=ne++;
    }
    void dij(int st)
    {
        memset(vis,false,sizeof(vis));
        memset(dis,0x3f3f3f3f,sizeof(dis));
        priority_queue<edge>Q;
```

```
        dis[st]=0,Q.push(edge(st,0));
        while(!Q.empty())
        {
            edge t=Q.top();
            Q.pop();
            if(vis[t.nex])continue;
            vis[t.nex]=true;
            for(int i=head[t.nex];~i;i=e[i].nex_edge)
                if(!vis[e[i].nex] && dis[t.nex]+e[i].dis<dis[e[i].nex])
                {
                    dis[e[i].nex]=dis[t.nex]+e[i].dis;
                    Q.push(edge(e[i].nex,dis[e[i].nex]));
                }
        }
    }
    void clear(){ne=0,memset(head,-1,sizeof(head));}
}dij;
```

最近公共祖先

倍增

```
struct LeastCommonAncestor
{
    static const int __=10005;
    static const int logn=15;
    struct edge
    {
        int x,val;
        edge(int x,int y):
            x(x),val(y) {}
    };

    vector<edge>G[__];
    int n,pre[__][logn],dis[__][logn],dep[__];

    void add_edge(int x,int y,int z)
    {
        G[x].pb(edge(y,z));
    }

    void dfs(int x,int fa,int dist)
    {
        pre[x][0]=fa,dis[x][0]=dist,dep[x]=dep[fa]+1;
        fup(i,0,sz(G[x])-1)
            if(G[x][i].x!=fa)
                dfs(G[x][i].x,x,G[x][i].val);
    }

    void init()
    {
        dfs(1,0,0);
```

```
    fup(i,1,logn-1)
        fup(j,1,n)
        {
            pre[j][i]=pre[pre[j][i-1]][i-1];
            dis[j][i]=dis[j][i-1]+dis[pre[j][i-1]][i-1];
        }
}

int get_lca(int x,int y)
{
    if(dep[x]<dep[y])swap(x,y);
    fdn(i,logn-1,0)
        if(pre[x][i] && dep[pre[x][i]]>=dep[y])
            x=pre[x][i];
    if(x==y)return x;
    fdn(i,logn-1,0)
        if(pre[x][i]!=pre[y][i])
            x=pre[x][i],y=pre[y][i];
    return pre[x][0];
}

int get_dis(int x,int y)
{
    if(dep[x]<dep[y])swap(x,y);
    int sum=0;
    fdn(i,logn-1,0)
        if(pre[x][i] && dep[pre[x][i]]>=dep[y])
            sum+=dis[x][i],x=pre[x][i];
    if(x==y)return sum;
    fdn(i,logn-1,0)
        if(pre[x][i]!=pre[y][i])
            sum+=dis[x][i]+dis[y][i],x=pre[x][i],y=pre[y][i];
    sum+=dis[x][0]+dis[y][0];
    return sum;
}

//x到y路径上第k个节点
int get_kth(int x,int y,int k)
{
    int lca=get_lca(x,y);
    if(k==dep[x]-dep[lca]+1)return lca;
    if(k<dep[x]-dep[lca]+1)
    {
        k--;
        fdn(i,logn-1,0)
            if(pre[x][i] && k>=(1<<i))
                k-=(1<<i),x=pre[x][i];
        return x;
    }
    if(k>dep[x]-dep[lca]+1)
    {
        k=dep[x]+dep[y]-2*dep[lca]-k+1;
        fdn(i,logn-1,0)
```

```
            if(pre[y][i] && k>=(1<<i))
                k-=(1<<i),y=pre[y][i];
        return y;
    }
}

void clear(){fup(i,1,n)G[i].clear();}
}lca;
```

RMQ-ST

```
struct LeastCommonAncestor
{
    static const int __=500005;
    static const int logn=20;

    int n,root,idx;
    int a[__];
    pii minn[__<<1][logn];
    vector<int>G[__];

    LeastCommonAncestor() {clear();}

    void add_edge(int x,int y)
    {
        G[x].pb(y);
    }

    void init(int _n,int rt=1)
    {
        n=_n,root=rt;
        dfs(root,-1,1);
        rmq(idx);
    }

    void dfs(int x,int fa,int dep)
    {
        minn[++idx][0]=mp(x,dep);
        if(!a[x])a[x]=idx;
        for(int y:G[x])
            if(y!=fa)
            {
                dfs(y,x,dep+1);
                minn[++idx][0]=mp(x,dep);
            }
    }

    void rmq(int n)
    {
        for(int j=1;(1<<j)<=n;++j)
            for(int i=1;i+(1<<(j-1))<=n;++i)
                if(minn[i][j-1].se<minn[i+(1<<(j-1))][j-1].se)
                    minn[i][j]=minn[i][j-1];
```

```
                else
                    minn[i][j]=minn[i+(1<<(j-1))][j-1];
    }

    int lca(int x,int y)
    {
        x=a[x],y=a[y];
        if(x>y)swap(x,y);
        int k=(int)log2(y-x+1);
        if(minn[x][k].se<minn[y-(1<<k)+1][k].se)
            return minn[x][k].fi;
        return minn[y-(1<<k)+1][k].fi;
    }

    void clear()
    {
        idx=0;
        for(int i=1;i<=n;++i)
        {
            G[i].clear();
            a[i]=0;
        }
    }
}lca;
```

强连通分量

```
namespace Graph
{
    const int __=2e5+5;
    int n,v[__];//点权
    vector<int>G[__];

    void init(int _n)
    {
        n=_n;
        for(int i=1;i<=n;++i)
            G[i].clear();
    }

    void add_edge(int x,int y)//有向边
    {
        G[x].push_back(y);
    }
}

namespace SCC
{
    const int __=2e5+5;
    vector<int>G[__];//缩点后注意重边有影响使用set
    int n,idx,dfn[__],low[__],bel[__],s[__];//栈
    ll v[__];//缩点后的点权
```

```
    int dfs(int x)
    {
        if(dfn[x])return dfn[x];
        s[++*s]=x;
        dfn[x]=low[x]=++idx;
        for(int y:Graph::G[x])
            if(!bel[y])
                low[x]=min(low[x],dfs(y));
        if(low[x]==dfn[x])
            for(bel[x]=++n,v[n]=0;;--*s)
            {
                v[n]+=Graph::v[s[*s]];
                if(s[*s]==x){--*s;break;}
                else bel[s[*s]]=n;
            }
        return low[x];
    }

    void tarjan()//注意一个点的特判
    {
        idx=n=*s=0;
        for(int i=1;i<=Graph::n;++i)
            if(!dfn[i])dfs(i);
        for(int i=1;i<=Graph::n;++i)
            for(int y:Graph::G[i])
                if(bel[i]!=bel[y])
                    G[bel[i]].push_back(bel[y]);
    }

    void clear()
    {
        for(int i=1;i<=Graph::n;++i)
        {
            dfn[i]=bel[i]=0;
            G[i].clear();
        }
    }

    void print()
    {
        for(int i=1;i<=Graph::n;++i)
            printf("bel[%d]=%d\n",i,bel[i]);
        for(int i=1;i<=n;++i)
        {
            printf("%d:",i);
            for(int x:G[i])
                pf(" %d",x);
            putchar('\n');
        }
    }
}
```

树链剖分

```
struct HeavyLightDecomposition
{
    static const int __=500005;
    int pre[__],siz[__],dep[__];
    int top[__],heavy[__],lson[__],rson[__];
    int n,root,idx;
    ll val[__];
    vector<int>G[__];

    HeavyLightDecomposition():root(1) {}

    void build(int _n)
    {
        n=_n,idx=0;
        dfs(root,0,1),slpf(root,0,root);
        fup(i,1,n)a[lson[i]]=val[i];
        T.build(n);
    }

    void add_edge(int x,int y)
    {
        G[x].pb(y);
    }

    int dfs(int x,int fa,int depth)
    {
        dep[x]=depth,pre[x]=fa;
        int res=1,maxx=0;
        fup(i,0,sz(G[x])-1)
        {
            if(G[x][i]==fa)continue;
            int t=dfs(G[x][i],x,depth+1);
            if(t>maxx)maxx=t,heavy[x]=G[x][i];
            res+=t;
        }
        return siz[x]=res;
    }

    void slpf(int x,int fa,int tp)
    {
        lson[x]=++idx,top[x]=tp;
        if(heavy[x])slpf(heavy[x],x,tp);
        fup(i,0,sz(G[x])-1)
            if(G[x][i]!=fa && G[x][i]!=heavy[x])
                slpf(G[x][i],x,G[x][i]);
        rson[x]=idx;
    }

    ll get_sum(int x,int y)
    {
        ll ans=0;
```

```
        for(;top[x]!=top[y];x=pre[top[x]])
        {
            if(dep[top[x]]<dep[top[y]])swap(x,y);
            ans+=T.get_val(lson[top[x]],lson[x]);
        }
        if(dep[x]>dep[y])swap(x,y);
        ans+=T.get_val(lson[x],lson[y]);
        return ans;
    }

    void add(int x,int y,ll val)
    {
        for(;top[x]!=top[y]; x=pre[top[x]])
        {
            if(dep[top[x]]<dep[top[y]])swap(x,y);
            T.add(lson[top[x]],lson[x],val);
        }
        if(dep[x]>dep[y])swap(x,y);
        T.add(lson[x],lson[y],val);
    }

    //子树加
    void add(int x,ll val){T.add(lson[x],rson[x],val);}

    //子树求和
    ll get_sum(int x){return T.get_val(lson[x],rson[x]);}

    int lca(int x,int y)
    {
        for(;top[x]!=top[y];x=pre[top[x]])
            if(dep[top[x]]<dep[top[y]])swap(x,y);
        if(dep[x]>dep[y])swap(x,y);
        return x;
    }

    void clear(){memset(heavy,0,sizeof(heavy));}
}hld;
```

数学

公式

对数加法

$\log_a x + \log_a y = \log_a (x \cdot y)$

等比数列求和

$$s_n = \begin{cases} n \cdot a_1 & (q = 1) \\ \frac{a_1 \cdot (1 - q^n)}{1 - q} & (q \neq 1) \end{cases}$$

三角函数

和差角

$\sin(a \pm b) = \sin(a) \cdot \cos(b) \pm \cos(a) \cdot \sin(b)$

$\cos(a \pm b) = \cos(a) \cdot \cos(b) \mp \sin(a) \cdot \sin(b)$

$\tan(a \pm b) = \frac{\tan(a) \pm \tan(b)}{1 \mp \tan(a) \cdot \tan(b)}$

和差化积

$\sin(a) + \sin(b) = 2 \cdot \sin(\frac{a+b}{2}) \cos(\frac{a-b}{2})$

$\sin(a) - \sin(b) = 2 \cdot \cos(\frac{a+b}{2}) \sin(\frac{a-b}{2})$

$\cos(a) + \cos(b) = 2 \cdot \cos(\frac{a+b}{2}) \cos(\frac{a-b}{2})$

$\cos(a) - \cos(b) = -2 \cdot \sin(\frac{a+b}{2}) \sin(\frac{a-b}{2})$

$\tan(a) \pm \tan(b) = \frac{\sin(a \pm b)}{\cos(a) \cos(b)}$

积化和差

$\sin(a) \cos(b) = \frac{1}{2} \big(\sin(a+b) + \sin(a-b) \big)$

$\cos(a) \sin(b) = \frac{1}{2} \big(\sin(a+b) - \sin(a-b) \big)$

$\cos(a) \cos(b) = \frac{1}{2} \big(\cos(a+b) + \cos(a-b) \big)$

$\sin(a) \sin(b) = -\frac{1}{2} \big(\cos(a+b) - \cos(a-b) \big)$

常见数列求和

$\sum\limits_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

$\sum\limits_{i=1}^{n} (2i-1)^2 = \frac{n(4n^2-1)}{3}$

$\sum\limits_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}$

$\sum\limits_{i=1}^{n}(2i-1)^3 = n^2(2n^2-1)$

$\sum\limits_{i=1}^{n} n(n+1) = \frac{n(n+1)(n+2)}{3}$

本源勾股数

$a = 2 \cdot n + 1 \quad (n \geq 1)$

那么: $\begin{cases} b = 2 \cdot n^2 + 2 \cdot n \\ c = 2 \cdot n^2 + 2 \cdot n + 1 \end{cases}$

$a = 2 \cdot n \quad (n \geq 2)$

那么: $\begin{cases} b = n^2 - 1 \\ c = n^2 + 1 \end{cases}$

均值不等式

调和平均 $H_n = \frac{n}{\sum\limits_{i=1}^{n} \frac{1}{x_i}}$

几何平均 $G_n = \sqrt[n]{\prod\limits_{i=1}^{n} x_i}$

算术平均 $A_n = \frac{\sum\limits_{i=1}^{n} x_i}{n}$

平方平均 $Q_n = \sqrt{\frac{\sum\limits_{i=1}^{n} x_i^2}{n}}$

那么: $H_n \leq G_n \leq A_n \leq Q_n$

错排

$D_n = n! \cdot \sum\limits_{i=0}^{n} \frac{(-1)^i}{i!} = \left\lfloor \frac{n!}{e} + 0.5 \right\rfloor$

$D_n = (n-1) \cdot \left(D_{n-1} + D_{n-2}\right)$

$D_1 = 0 \quad D_2 = 1$

斯特林公式

$n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$

矩阵树定理

$a[x][y] = \begin{cases} deg(x) & x = y \\ -1 & x \neq y \bigcap x\text{与}y\text{相邻} \\ 0 & \text{其他} \end{cases}$

基姆拉尔森计算公式

$week = (day + 2month + \frac{3(month+1)}{5} + year + \frac{year}{4} - \frac{year}{100} + \frac{year}{400})\%7 + 1$

1, 2月当作上一年的13, 14月

曲线区域分割

n条 直线 最多能把平面分成 $\frac{1}{2}n(n-1)+1$ 部分

n个 三角形 最多能把平面分成 $3n^2-3n+2$ 部分

n个 四边形 最多能把平面分成 $2\cdot(3-2n)^2$ 部分

n个 圆 最多能把平面分成 n^2-n+2 部分

n个 椭圆 最多能把平面分成 $2(n^2-n+1)$ 部分

n个 $d-1$维超平面 最多能把d维空间分成 $\sum\limits_{i=0}^{d} C_n^i$

平面图欧拉公式

顶点数$-$边数$+$面数$=2$

皮克公式

面积$=$内部点个数$+\frac{1}{2}$边界上点个数-1

莫比乌斯反演

$$\begin{cases} f(n) = \sum\limits_{d|n} g(d) \\ g(n) = \sum\limits_{d|n} \mu(d) f(\frac{n}{d}) \end{cases}$$

$$\begin{cases} f(n) = \sum\limits_{n|d} g(d) \\ g(n) = \sum\limits_{n|d} \mu(\frac{d}{n}) f(d) \end{cases}$$

切比雪夫距离和曼哈顿距离

$$\begin{cases} x_{曼} = \frac{x_{切}+y_{切}}{2} \\ y_{曼} = \frac{y_{切}-x_{切}}{2} \end{cases}$$

$$\begin{cases} x_{切} = x_{曼} - y_{曼} \\ y_{切} = x_{曼} + y_{曼} \end{cases}$$

约瑟夫环

```
//n个人编号为1到n, 从1开始报数, 报到k出列, 返回第m(1<=m<=n)个出列的人
ll Josephus(ll n,ll k,ll m)
{
    if(k==1)return m;
    ll x=(k-1)%(n+1-m);
    for(ll i=n+1-m;i<n;)
    {
        ll y=min((i-x+k-2)/(k-1),n-i);
        i+=y,x=(x+y*k)%i;
    }
    return x+1;
}
```

数学

慢速乘

```
ll qmul(ll x,ll y)
{
    ll res=0;
    for(;y;y>>=1,x<<=1)
        if(y&1)res+=x;
    return res;
}
```

快速乘

```
ll qmul(ll x,ll y,ll mod)
{
    x%=mod,y%=mod;
    ll res=x*y-(ll)(((ld)x*y+0.5)/mod)*mod;
    if(res<=-mod || res>=mod)res%=mod;
    if(res<0)res+=mod;
    return res;
}
```

快速幂

```
ll qpow(ll x,ll y)
{
    ll res=1;
    for(;y;y>>=1,x=x*x)
        if(y&1)res=res*x;
    return res;
}
```

矩阵快速幂

```
struct matrix
{
    int n,m;
    ll ma[105][105];
    matrix(int x,int y):n(x),m(y) {clear();}
    void set(int _n,int _m){n=_n,m=_m;}
    ll* operator[](int x){return ma[x];}
    matrix operator*(matrix x)
    {
        assert(m==x.n);
        matrix res(n,x.m);
        for(int i=1;i<=n;i++)
            for(int j=1;j<=x.m;j++)
                for(int k=1;k<=m;k++)
                    (res[i][j]+=ma[i][k]*x[k][j]%mod+mod)%=mod;
        return res;
    }
    matrix operator^(ll y)
    {
        assert(n==m);
        matrix x(n,m);
        for(int i=1;i<=n;i++)
            for(int j=1;j<=m;j++)
                x[i][j]=ma[i][j];
        matrix res(x.n,x.n);
        for(int i=1;i<=x.n;i++)
            res[i][i]=1;
        for(;y;y>>=1,x=x*x)
            if(y&1)res=res*x;
        return res;
    }
    void print()
    {
        for(int i=1;i<=n;++i)
            for(int j=1;j<=m;++j)
                printf("%lld%c",ma[i][j]," \n"[j==m]);
    }
    void clear() {memset(ma,0,sizeof(ma));}
};
```

辛普森积分

已知一个初等函数$f(x)$，求解$\int_a^b f(x)$

先将$f(x)$近似为某二次函数$g(x) = A\cdot x^2 + B\cdot x + C$

那么:

$$
\begin{aligned}
\int_a^b g(x) &= \left(\frac{A}{3}\cdot x^3+\frac{B}{2}\cdot x^2+C\cdot x+D\right)\Big|_a^b \\
&= \frac{A}{3}\cdot(b^3-a^3)+\frac{B}{2}\cdot(b^2-a^2)+C\cdot(b-a) \\
&= \frac{A}{3}\cdot(b-a)\cdot(b^2+b\cdot a+a^2)+\frac{B}{2}\cdot(b-a)\cdot(b+a)+C\cdot(b-a) \\
&= \frac{(b-a)}{6}\left(2A\cdot(b^2+b\cdot a+a^2)+3B\cdot(b+a)+6C\right) \\
&= \frac{(b-a)}{6}\left(A\cdot\left(a^2+b^2+4\cdot(\frac{a+b}{2})^2\right)+B\cdot\left(a+b+4\cdot(\frac{a+b}{2})\right)+6C\right) \\
&= \frac{(b-a)}{6}\cdot\left(g(a)+g(b)+4\cdot g(\frac{a+b}{2})\right)
\end{aligned}
$$

```
struct Simpson
{
    static const double eps=1e-8;
    static double f(double x)
    {
        return (c*x+d)/(a*x+b);
    }
    double simpson(double l,double r)
    {
        return (r-l)*(f(l)+f(r)+4*f((l+r)/2))/6;
    }
    double integral(double l,double r,double ans=1e18)
    {
        double mid=(l+r)/2;
        double le=simpson(l,mid),ri=simpson(mid,r);
        if(fabs(le+ri-ans)<eps)return le+ri;
        return integral(l,mid,le)+integral(mid,r,ri);
    }
}S;
```

高斯消元

浮点数

```
int guass(int n,int m)
{
    int i=1;
    for(int j=1;i<=n && j<=m;j++)
    {
        int x=i;
        for(int k=i+1;k<=n;k++)
            if(!zero(a[k][j]) && fabs(a[k][j])>fabs(a[x][j]))
               x=k;
        if(zero(a[x][j]))continue;
        for(int k=j;k<=m;k++)
            swap(a[i][k],a[x][k]);
        for(int k=1;k<=n;k++)
        {
            if(k==i || zero(a[k][j]))continue;
```

```
                double f=a[k][j]/a[i][j];
                for(int l=j;l<=m;l++)
                    a[k][l]-=a[i][l]*f;
            }
            for(int k=m;k>=j;k--)
                a[i][k]/=a[i][j];
            i++;
        }
        return i-1;
    }
```

模意义

```
int guass(int n,int m)
{
    int i=1;
    for(int j=1;i<=n && j<=m;++j)
    {
        int x=i;
        for(int k=i;k<=n;++k)
            if(a[k][j])
            {
                x=k;
                break;
            }
        if(!a[x][j])continue;
        for(int k=j;k<=m;++k)
            swap(a[i][k],a[x][k]);
        for(int k=1;k<=n;++k)
        {
            if(k==i || !a[k][j])continue;
            ll f=a[k][j]*qpow(a[i][j],mod-2)%mod;
            for(int l=j;l<=m;++l)
            {
                a[k][l]-=a[i][l]*f%mod;
                if(a[k][l]<=-mod)a[k][l]%=mod;
                if(a[k][l]<0)a[k][l]+=mod;
            }
        }
        for(int k=m;k>=j;--k)
            (a[i][k]*=qpow(a[i][j],mod-2))%=mod;
        ++i;
    }
    return i-1;
}
```

异或

```
int guass(int n,int m)
{
    int i=1;
    for(int j=1; i<=n && j<=m; j++)
```

```
    {
        int x=0;
        for(int k=i; k<=n; k++)
            if(b[k][j])
            {
                x=k;
                break;
            }
        if(!x)continue;
        for(int k=j;k<=m;k++)
            swap(b[i][k],b[x][k]);
        for(int k=1;k<=n;k++)
        {
            if(k==i || !b[k][j])continue;
            for(int l=j;l<=m;l++)
                b[k][l]^=b[i][l];
        }
        i++;
    }
    return i-1;
}
```

线性基

```
struct XorBasis
{
    typedef ll type;

    vector<type>G;
    bool zero;

    XorBasis() {clear();}

    void insert(type x)
    {
        for(int i=sz(G)-1;~i && x;--i)
        {
            type t=x^G[i];
            if(t<x)
            {
                if(t>G[i])break;
                x^=G[i];
            }
        }
        if(!x){zero=true;return;}
        G.pb(x);
        for(int i=sz(G)-1;i && G[i]<G[i-1];--i)
            swap(G[i],G[i-1]);
    }

    void build()
    {
        for(int i=0;i<sz(G);++i)
```

```
            for(int j=i+1;j<sz(G);++j)
                if((G[j]^G[i])<G[j])
                    G[j]^=G[i];
    }

    type get_max()
    {
        type res=0;
        for(int i=sz(G)-1;i>=0;--i)
            if((res^G[i])>res)
                res^=G[i];
        return res;
    }

    type kth(ll k)
    {
        if(zero && !--k)return 0;
        if(k>=(1ll<<sz(G)))return -1;
        type ans=0;
        for(int i=sz(G)-1;i>=0;--i)
        {
            type x=(1ll<<i);
            if(k>=x)k-=x,ans^=G[i];
            if(!k)return ans;
        }
        return ans;
    }

    int size() {return G.size()+zero;}

    void clear()
    {
        zero=false;
        G.clear();
    }
}X;
```

可持久化线性基

```
struct PersistentXorBasis
{
    static const int __=5e5+5;
    typedef int type;

    struct XorBasis
    {
        vector<pair<type,int> >G;
        int zero;   //最后出现的位置

        XorBasis() {clear();}

        void operator=(const XorBasis &b)
        {
```

```
        for(int i=0;i<sz(b.G);++i)
            G.push_back(b.G[i]);
        zero=b.zero;
    }

    int size() {return G.size()+bool(zero);}

    void clear()
    {
        zero=0;
        G.clear();
    }
}X[__];

int n;

void insert(pair<type,int>x)
{
    vector<pair<type,int> >&G=X[x.se].G;
    for(int i=sz(G)-1;~i && x.fi;--i)
    {
        type t=x.fi^G[i].fi;
        if(t<x.fi)
        {
            if(t>G[i].fi)break;
            if(x.se>G[i].se)swap(G[i],x);
            x.fi^=G[i].fi;
        }
    }
    if(!x.fi){X[x.se].zero=x.se;return;}
    G.pb(x);
    for(int i=sz(G)-1;i && G[i].fi<G[i-1].fi;--i)
        swap(G[i],G[i-1]);
}

void build(type a[],int _n)
{
    n=_n;
    for(int i=1;i<=n;++i)
    {
        X[i]=X[i-1];
        insert(make_pair(a[i],i));
    }
}

type get_max(int l,int r)
{
    vector<pair<type,int> >&G=X[r].G;

    type res=0;
    for(int i=sz(G)-1;~i;--i)
        if(G[i].se>=l && (res^G[i].fi)>res)
            res^=G[i].fi;
```

```
        return res;
    }

    void clear()
    {
        for(int i=1;i<=n;++i)
            x[i].clear();
    }
}X;
```

行列式求值

```
ll a[205][205],mod;

ll det(int n)
{
    ll ans=1,f=1;
    for(int i=1; i<=n; i++)
    {
        for(int j=i+1; j<=n; j++)
        {
            int x=i,y=j;
            while(a[y][i])
            {
                ll t=a[x][i]/a[y][i];
                for(int k=i; k<=n; k++)
                    a[x][k]=(a[x][k]-a[y][k]*t%mod)%mod;
                swap(x,y);
            }
            if(x!=i)
            {
                for(int k=1; k<=n; k++)
                    swap(a[i][k],a[j][k]);
                f=-f;
            }
        }
        ans=ans*a[i][i]%mod;
    }
    return (ans*f+mod)%mod;
}
```

拉格朗日插值

给出$n+1$个x互不相同点(x_i, y_i)，可以确定一个最高次数至多为n的多项式

构造多项式: $l_i(x)=\begin{cases}1 & (x=x_i)\\ 0 & (x=x_j \wedge i\neq j)\end{cases}$

那么: $f(x)=\sum\limits_{i=0}^{n} l_i(x)\cdot y_i$

令: $l_i(x)=\frac{(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}=\frac{\prod\limits_{j=0\wedge j\neq i}^{n}(x-x_j)}{\prod\limits_{j=0\wedge j\neq i}^{n}(x_i-x_j)}$

```
//(x[0],y[0]),……,(x[n],y[n])    n+1个点
//插出n次多项式，返回带入x0的值
ll lagrange(int x[],int y[],int n,int x0)
{
    ll res=0;
    for(int i=0;i<=n;++i)
    {
        ll fz=1,fm=1;
        for(int j=0;j<=n;++j)
        {
            if(j==i)continue;
            fz=md(fz*(x0-x[j]));
            fm=md(fm*(x[i]-x[j]));
        }
        (res+=y[i]*fz%mod*qpow(fm,mod-2)%mod)%=mod;
    }
    return res;
}
```

自然幂和

求: $\sum\limits_{i=1}^{n} i^k$

```
//快速幂

const int __=1e5+5;
int pk[__+5],num[__+5],sum[__+5];

//线性筛: pk[i]=qpow(i,k)%mod
void powk(int k)
{
    num[0]=0,pk[1]=sum[1]=1;
    for(int i=2;i<=k+1;++i)pk[i]=0;
    for(int i=2;i<=k+1;++i)
    {
        if(!pk[i])
        {
            num[++num[0]]=i;
            pk[i]=qpow(i,k);
        }
        for(int j=1;j<=num[0] && i*num[j]<=__;++j)
        {
            int &p=num[j];
            pk[i*p]=1ll*pk[i]*pk[p]%mod;
            if(!(i%p))break;
        }
        sum[i]=(sum[i-1]+pk[i])%mod;
    }
```

```
}

//预处理: ll fac[]阶乘, ll inv[]逆元, ll facinv[]阶乘逆元

//(x[i]=x+i,y[i])    n+1个x连续的点
//插出n次多项式, 返回带入x0的值
ll pre[__],saf[__];

ll lagrange(int x,int y[],int n,int x0)
{
    pre[0]=x0-x,saf[n]=x0-x-n;
    for(int i=1;i<=n;++i)
    {
        pre[i]=md(pre[i-1]*(x0-x-i));
        saf[n-i]=md(saf[n-i+1]*(x0-x-n+i));
    }
    ll res=0;
    for(int i=0;i<=n;++i)
    {
        ll fz=1;
        if(i!=0)fz=md(fz*pre[i-1]);
        if(i!=n)fz=md(fz*saf[i+1]);
        ll fm=md(facinv[i]*facinv[n-i]);
        if((n-i)&1)fm=mod-fm;
        (res+=md(y[i]*md(fz*fm)))%=mod;
    }
    return res;
}

int main()
{
    //预处理
    int _;for(sf("%d",&_);_;--_)
    {
        ll n;int k;
        sf("%lld%d",&n,&k);
        powk(k);
        pf("%lld\n",lagrange(0,sum,k+1,n%mod));
    }
    return 0;
}
```

复杂度: $O(q \cdot k)$

快速数论变换

```
namespace NTT
{
    const int mod=119<<23|1; //998244353
    const int g=3;           //原根
    int wn[25];              //顺时针旋转因子
    int wr[25];              //逆时针旋转因子
    int inv[25];             //2^i的逆元
```

```
    void init()
    {
        inv[0]=wr[1]=1;
        inv[1]=mod-mod/2;
        for(int i=2;i<=23;++i)
            inv[i]=md(1ll*inv[i-1]*inv[1]);
        wn[23]=15311432;    //qpow(3,119)
        wr[23]=469870224;   //inv[wn[23]]
        for(int i=22;i>=1;--i)
        {
            wn[i]=1ll*wn[i+1]*wn[i+1]%mod;
            wr[i]=1ll*wr[i+1]*wr[i+1]%mod;
        }
    }

    void ntt(ll xs[],int m,int logm,bool dft=true)
    {
        int invm=inv[logm];
        for(int i=0,j=m>>1,k=0;i<m;++i)
        {
            if(k>i)swap(xs[k],xs[i]);
            while(k&j)k^=j,j>>=1;
            k|=j,j=m>>1;
        }
        for(int i=1,rat=2;rat<=m;++i,rat<<=1)
        {
            int l=rat>>1,w=dft?wn[i]:wr[i];
            for(int j=0,wx=1;j<l;++j,wx=md(1ll*wx*w))
                for(int k=j;k<m;k+=rat)
                {
                    int t=md(xs[k]-md(wx*xs[k+l]));
                    xs[k]=md(xs[k]+md(wx*xs[k+l]));
                    xs[k+l]=t;
                    if(m==rat && !dft)
                    {
                        xs[k]=md(xs[k]*invm);
                        xs[k+l]=md(xs[k+l]*invm);
                    }
                }
        }
    }
};

NTT::init();
```

字符串匹配

长度为n文本串a中查找长度为m的模式串b　　(下标均从0开始)

假设成功匹配的位置为x

那么: $0=\sum\limits_{i=0}^{m-1}(a[x-m+1+i]-b[i])^2$

展开这个等式:

$$\sum_{i=0}^{m-1}(a[x-m+1+i])^2+(b[i])^2-2\cdot a[x-m+1+i]\cdot b[i]$$

翻转字符串b (即: $b[m-1-i]=b[i]$)

$$\sum_{i=0}^{m-1}\Big((a[x-m+1+i])^2+(b[m-1-i])^2-2\cdot a[x-m+1+i]\cdot b[m-1-i]\Big)$$

分成三部分考虑

$$\sum_{i=0}^{m-1}(a[x-m+1+i])^2=\sum_{i=x-m+1}^{x}(a[i])^2$$

$$\sum_{i=0}^{m-1}(b[m-1-i])^2=\sum_{i=0}^{m-1}(b[i])^2$$

$\sum_{i=0}^{m-1}a[x-m+1+i]\cdot b[m-1-i]=\sum_{i=0}^{m-1}a[x-i]\cdot b[i]$**是卷积的形式**

```
const int __=2.1e6+5;

int a[__],b[__];
ll sum1[__],sum2,na[__],nb[__];

int main()
{
    NTT::init();
    int _;for(sf("%d",&_);_;--_)
    {
        int n,m;sf("%d%d",&n,&m);
        fup(i,0,n-1)sf("%d",&a[i]);
        fup(i,0,m-1)sf("%d",&b[i]);
        reverse(b,b+m);
        sum1[0]=md(1ll*a[0]*a[0]);
        na[0]=a[0];
        fup(i,1,n-1)
        {
            sum1[i]=md(sum1[i-1]+1ll*a[i]*a[i]);
            na[i]=a[i];
        }
        fdn(i,n-1,m)
            sum1[i]=md(sum1[i]-sum1[i-m]);
        sum2=0;
        fup(i,0,m-1)
        {
            sum2=md(sum2+1ll*b[i]*b[i]);
            nb[i]=b[i];
        }
        int x=1,logx=0;
        for(;x<n+m-1;x<<=1,++logx);
        NTT::ntt(na,x,logx);
        NTT::ntt(nb,x,logx);
        for(int i=0;i<x;++i)
```

```
            na[i]=md(na[i]*nb[i]);
        NTT::ntt(na,x,logx,false);
        fup(i,m-1,n-1)
            if(!md(sum1[i]+sum2-2*na[i]))
                //匹配成功
    }
    return 0;
}
```

组合数学

$P_n^k=\frac{n!}{(n-k)!}=nP_{n-1}^{k-1}=P_{n-1}^k+kP_{n-1}^{k-1}$

$C_n^k=\frac{n!}{k!(n-k)!}=C_{n-1}^{k-1}+C_{n-1}^k$

组合数奇偶性

C_n^k中如果$n\&k=k$，那么C_n^k为奇数，否则为偶数

卢卡斯定理

```
struct Combination
{
    static const int __=1e6+5;
    int mod;    //mod<=1e6且为质数
    ll fac[__],inv[__],facinv[__];

    void init(int p)
    {
        mod=p;
        fac[0]=inv[0]=facinv[0]=1;
        fac[1]=inv[1]=facinv[1]=1;
        for(int i=2;i<mod;++i)
        {
            fac[i]=fac[i-1]*i%mod;
            inv[i]=mod-(mod/i*inv[mod%i]%mod);
            facinv[i]=facinv[i-1]*inv[i]%mod;
        }
    }

    ll c(int n,int k)
    {
        if(k>n)return 0;
        return fac[n]*facinv[k]%mod*facinv[n-k]%mod;
    }

    ll lucas(ll n,ll k)
    {
        if(!k)return 1;
        return c(n%mod,k%mod)*lucas(n/mod,k/mod)%mod;
    }
}C;
```

二项式反演

已知$f(n)=\sum_{i=0}^{n}C_n^i\cdot g(i)$，求$g(n)$

二项式定理:

$(a+b)^n=\sum_{i=0}^{n}C_n^i\cdot a^i\cdot b^{n-i}$

令: $a=-1$**,** $b=1 \quad (n \neq 0) \quad (0^0$**无意义**$)$

$$0=\sum_{i=0}^{n}(-1)^i \cdot C_n^i \quad (n \neq 0)$$

当$n=0$**时,** $(a+b)^0=1$

所以:

$$\sum_{i=0}^{n}(-1)^i \cdot C_n^i=\begin{cases}0 & (n \neq 0)\\ 1 & (n=0)\end{cases}$$

$$g(n)=\sum_{i=0}^{n} C_n^i \cdot \begin{cases}0 & (i \neq n)\\ g(i) & (i=n)\end{cases}$$

即:

$$g(n)=\sum_{i=0}^{n} C_n^i \cdot \begin{cases}0 & (n-i \neq 0)\\ g(i) & (n-i=0)\end{cases}$$

所以:

$$g(n)=\sum_{i=0}^{n}\sum_{j=0}^{n-i}(-1)^j \cdot C_{n-i}^j \cdot C_n^i \cdot g(i)$$

组合恒等式:

$$C_n^i \cdot C_{n-i}^j=C_n^j \cdot C_{n-j}^i$$

证明: 从n**个中选**i**个再从剩下**$n-i$**个中选**j**个**$=$**从**n**个中选**j**个再从剩下**$n-j$**个中选**i**个**

$$\begin{aligned}g(n)&=\sum_{i=0}^{n}\sum_{j=0}^{n-i}(-1)^j \cdot C_{n-j}^i \cdot C_n^j \cdot g(i)\\&=\sum_{j=0}^{n}\sum_{i=0}^{n-j}(-1)^j \cdot C_{n-j}^i \cdot C_n^j \cdot g(i)\\&=\sum_{j=0}^{n}(-1)^j \cdot C_n^j \sum_{i=0}^{n-j} C_{n-j}^i \cdot g(i)\end{aligned}$$

$$f(n)=\sum_{i=0}^{n} C_n^i \cdot g(i) \rightarrow f(n-j)=\sum_{i=0}^{n-j} C_{n-j}^i \cdot g(i)$$

$$
\begin{aligned}
g(n) &= \sum_{j=0}^{n}(-1)^j\cdot C_n^j\cdot f(n-j)\\
&= \sum_{i=0}^{n}(-1)^i\cdot C_n^i\cdot f(n-i)\\
&= \sum_{i=0}^{n}(-1)^{n-i}\cdot C_n^{n-i}\cdot f(i)\\
&= \sum_{i=0}^{n}(-1)^{n-i}\cdot C_n^i\cdot f(i)
\end{aligned}
$$

所以: $g(n)=\sum_{i=0}^{n}(-1)^{n-i}\cdot C_n^i\cdot f(i)$

差分

给定一个离散序列: $a_0, a_2, \cdots, a_n, \cdots$

零阶差分(原序列): $\Delta^0 h_n = a_n$

一阶差分: $\Delta^1 h_n = a_{n+1} - a_n$

二阶差分: $\Delta^2 h_n = \Delta^1 h_{n+1} - \Delta^1 h_n = a_{n+2} - 2\cdot a_{n+1} + a_n$

三阶差分: $\Delta^3 h_n = \Delta^2 h_{n+1} - \Delta^2 h_n = a_{n+3} - 3\cdot a_{n+2} + 3\cdot a_{n+1} - a_n$

p阶差分: $\Delta^p h_n = \Delta^{p-1} h_{n+1} - \Delta^{p-1} h_n$

观察发现: $\Delta^p h_n = \sum_{i=0}^{p}(-1)^{p-i}\cdot C_p^i\cdot a_{n+i}$

如果a_x是关于x的k次多项式, 即: $a_x = \sum_{i=0}^{k} b_i\cdot x^i$

$$
\begin{aligned}
\Delta^1 h_x &= a_{x+1} - a_x\\
&= \sum_{i=0}^{k} b_i\cdot(x+1)^i - \sum_{i=0}^{k} b_i\cdot x^i\\
&= \sum_{i=0}^{k} b_i \sum_{j=0}^{i} C_i^j\cdot x^j - \sum_{i=0}^{k} b_i\cdot x^i\\
&= \sum_{j=0}^{k} x^j \sum_{i=j}^{k} b_i\cdot C_i^j - \sum_{i=0}^{k} b_i\cdot x^i\\
&= \sum_{i=0}^{k} x^i \sum_{j=i}^{k} b_j\cdot C_j^i - \sum_{i=0}^{k} b_i\cdot x^i\\
&= \sum_{i=0}^{k} x^i\cdot\left(\sum_{j=i}^{k}(b_j\cdot C_j^i) - b_i\right)\\
&= \sum_{i=0}^{k-1} x^i \sum_{j=i+1}^{k} C_j^i\cdot b_j
\end{aligned}
$$

定理1: k次多项式差分后为$k-1$次多项式

差分的线性叠加性: $h_n = \alpha\cdot f_n + \beta\cdot g_n$

$$\begin{aligned}\Delta h_n &= \left(\alpha\cdot f_{n+1}+\beta\cdot g_{n+1}\right)-\left(\alpha\cdot f_n+\beta\cdot g_n\right)\\ &= \alpha\cdot\left(f_{n+1}-f_n\right)+\beta\cdot\left(g_{n+1}-g_n\right)\\ &= \alpha\cdot\Delta f_n+\beta\cdot\Delta g_n\end{aligned}$$

对于一个k次多项式$a_x=\sum_{i=0}^{k} b_i\cdot x^i$

已知它的$\Delta^0 h_0,\Delta^1 h_0,\Delta^2 h_0,\cdots,\Delta^k h_0$

根据二项式反演: 已知: $f(n)=\sum_{i=0}^{n} C_n^i\cdot g(i)$**, 则:** $g(n)=\sum_{i=0}^{n}(-1)^{n-i}\cdot C_n^i\cdot f(i)$

令: $g(n)=\Delta^n h_0=\begin{cases}\sum_{i=0}^{n}(-1)^{n-i}\cdot C_n^i\cdot a_i & (0\le n\le k)\\ 0 & (n>k)\end{cases}$

反演得到: $f(n)=a_n=\sum_{i=0}^{n} C_n^i\cdot\Delta^i h_0=\sum_{i=0}^{k} C_n^i\cdot\Delta^i h_0$

考虑k次多项式a_i的前n项和$s_n=\sum_{i=0}^{n} a_i$ $s_n=\sum_{i=0}^{n} a_i=\sum_{i=0}^{n}\sum_{j=0}^{k} C_i^j\cdot\Delta^j h_0=\sum_{j=0}^{k}\Delta^j h_0\sum_{i=0}^{n} C_i^j$

$\sum_{i=0}^{n} C_i^m=C_{n+1}^{m+1}$ **(数学归纳法可证明)**

$s_n=\sum_{j=0}^{k}\Delta^j h_0\sum_{i=0}^{n} C_i^j=\sum_{j=0}^{k}\Delta^j h_0\cdot C_{n+1}^{j+1}=\sum_{i=0}^{k} C_{n+1}^{i+1}\cdot\Delta^i h_0$

数论

质因数分解

```
ll pri_div[16];
void div(ll x)
{
    int y=(int)sqrt(x+0.1);
    for(int i=2;i<=y;i++)
        if(x%i==0)
            for(pri_div[++pri_div[0]]=i;x%i==0;x/=i);
    if(x!=1)pri_div[++pri_div[0]]=x;
}
```

线性求逆元

$1^{-1}=1(\%p)$

$\lfloor\frac{p}{x}\rfloor\cdot x+p\%x=p\equiv 0(\%p)$

$x^{-1}\equiv -\frac{\lfloor\frac{p}{x}\rfloor}{p\%x}=p-\lfloor\frac{p}{x}\rfloor\cdot(p\%x)^{-1}(\%p)$

```
for(int i=2;i<=n;++i)
    inv[i]=p-(1ll*p/i*inv[p%i])%p;
```

Miller-Rabin

```
struct MillerRabin
{
    const int base[14]=
    {0,2,3,5,7,11,13,17,19,23,29,31,37,41};

    static ll qmul(ll x,ll y,ll mod)
    {
        if(x>=mod)x%=mod;
        if(y>=mod)y%=mod;
        ll res=0;
        for(;y;y>>=1)
        {
            if((y&1) && (res+=x)>=mod)
                res-=mod;
            if((x+=x)>=mod)x-=mod;
        }
        return res;
    }

    static ll qpow(ll x,ll y,ll mod)
    {
        if(x>=mod)x%=mod;
        ll res=1;
        for(;y;y>>=1,x=qmul(x,x,mod))
```

```
            if(y&1)res=qmul(res,x,mod);
        return res;
    }

    bool test(ll n)
    {
        if(n==2)return true;
        if(n<=1 || !(n&1))return false;
        ll u;for(u=n-1;!(u&1);u>>=1);
        for(int i=1;i<=13 && base[i]<n;++i)
        {
            ll x=qpow(base[i],u,n),y;
            for(ll v=u;v<=n;x=y,v<<=1)
            {
                y=qmul(x,x,n);
                if(y==1 && x!=1 && x!=n-1)
                    return false;
            }
            if(x!=1)return false;
        }
        return true;
    }
}MR;
```

原根

```
int pri_root(ll x)
{
    div(x-1);
    for(int i=2;i<=x-1;i++)
    {
        bool flag=true;
        for(int j=1;j<=pri_div[0] && flag;j++)
            if(qpow(i,(x-1)/pri_div[j],x)==1)
                flag=false;
        if(flag)return i;
    }
}
```

欧拉降幂

$$x^y\%m = \begin{cases} x^y\%m & y < \varphi(m) \\ x^{y\%\varphi(m)+\varphi(m)}\%m & y \geq \varphi(m) \end{cases}$$

$$a_l^{a_{l+1}^{a_{l+2}^{\cdot^{\cdot^{a_r}}}}}\%m$$

```
ll qpow(ll x,ll y,ll m)
{
    ll res=1;
    for(x=x<m?x:x%m+m;y;y>>=1)
    {
        if(y&1)
```

```
        {
            res=res*x;
            res=res<m?res:res%m+m;
        }
        x=x*x;
        x=x<m?x:x%m+m;
    }
    return res;
}

//求phi(x)值模板(记忆化)

ll solve(int l,int r,int m)
{
    if(l==r || m==1)
        return (a[l]<m)?a[l]:a[l]%m+m;
    int phim=get_phi(m);
    ll p=solve(l+1,r,phim);
    return qpow(a[l],p,m);
}
```

数论分块

$\lfloor \frac{n}{i} \rfloor \quad (0 < i \le n)$**的不同取值不超过:** $2\sqrt{n}$**种**

令: $\lfloor \frac{n}{i} \rfloor = x$

那么: $x \cdot i \le n \le (x+1) \cdot i - 1$

得: $\lceil \frac{n+1}{x+1} \rceil \le i \le \lfloor \frac{n}{x} \rfloor$

```
pii cal(int n,int x)
{
    int l=(n+x+1)/(x+1),r=n/x;
    return mp(l,r);
}
```

枚举$\lfloor \frac{n}{i} \rfloor$**的所有取值　复杂度:** $O(\sqrt{n})$

```
for(int i=1;i<=n;i=n/(n/i)+1)
    //  n/i
```

$\sum\limits_{i=1}^{n} \lfloor \frac{n}{i} \rfloor$**所有不同取值个数为:** $2 \cdot \lfloor \sqrt{n} \rfloor - \left[\lfloor \sqrt{n} \rfloor \cdot (\lfloor \sqrt{n} \rfloor + 1) > n \right]$

$\lfloor \frac{n}{1} \rfloor, \lfloor \frac{n}{2} \rfloor, \cdots, \lfloor \frac{n}{n} \rfloor$**从大到小排序并去重第**$k$**个　复杂度:** $O(1)$

```
ll kth(ll n,ll k)
{
    ll sqn=(ll)sqrt(n+0.1);
    if(k<=sqn)return n/k;
    if(sqn*sqn+sqn>n)return 2*sqn-k;
    return 2*sqn+1-k;
}
```

线性筛

筛质数

```
const int __=1e6+5;
bool pri[__];
int num[__];
void prime()
{
    pri[0]=pri[1]=true;
    for(int i=2;i<__;++i)
    {
        if(!pri[i])num[++num[0]]=i;
        for(int j=1;j<=num[0] && i*num[j]<__;++j)
        {
            pri[i*num[j]]=true;
            if(!(i%num[j]))break;
        }
    }
}
```

筛欧拉函数$\varphi(x)$

```
const int __=1e6+5;
int phi[__],num[__];
void euler()
{
    phi[1]=1;
    for(int i=2;i<__;++i)
    {
        if(!phi[i])
        {
            num[++num[0]]=i;
            phi[i]=i-1;
        }
        for(int j=1;j<=num[0] && i*num[j]<__;++j)
        {
            int &p=num[j];
            if(!(i%p))
            {
                phi[i*p]=phi[i]*p;
                break;
            }
            phi[i*p]=phi[i]*(p-1);
```

```
        }
    }
}
```

筛莫比乌斯函数$\mu(x)$

```
const int __=5e6+5;
bool pri[__];
int mu[__],num[__];
void mobius()
{
    mu[1]=1,pri[1]=true;
    for(int i=2;i<__;++i)
    {
        if(!pri[i])
        {
            num[++num[0]]=i;
            mu[i]=-1;
        }
        for(int j=1;j<=num[0] && i*num[j]<__;++j)
        {
            int &p=num[j];
            pri[i*p]=true;
            if(!(i%p)){mu[i*p]=0;break;}
            mu[i*p]=-mu[i];
        }
    }
}
```

欧拉函数

定义: $\varphi(n)=\sum\limits_{i=1}^{n}\left[\gcd(i,n)=1\right]$

定理1:$\varphi(1)=1$

定理2:$\varphi(p)=p-1\qquad(p\textbf{是素数})$

定理3:$\varphi(p^k)=p^k-p^{k-1}=p^k(1-\frac{1}{p})\qquad(p\textbf{是素数})$

定理4:$\varphi(p\cdot q)=\varphi(p)\cdot\varphi(q)\qquad\big(\gcd(p,q)=1\big)$

证明:

设: $m\in[1,p\cdot q)$ **且** $\gcd(m,p\cdot q)=1$

那么: $\begin{cases}m\%p=x\\m\%q=y\end{cases}$ **即:** $\begin{cases}m=x(\%p)\\m=y(\%q)\end{cases}$ **且:** $\gcd(m,p)=\gcd(m,q)=1$

根据中国剩余定理: 该方程在模$(p\cdot q)$**下有唯一解**

即: $m=\Big(x\cdot q\cdot\big(q^{-1}(\%p)\big)+y\cdot p\cdot\big(p^{-1}(\%q)\big)\Big)\%(p\cdot q)$

定理5: $\varphi(n)=n(1-\frac{1}{p_1})\cdot(1-\frac{1}{p_2})\cdots(1-\frac{1}{p_k})=n\prod\limits_{i=1}^{k}(1-\frac{1}{p_i})$

$(n=p_1^{e_1}\cdot_2^{e_2}\cdots p_k^{e_k})$**其中**$p_1,p_2,\cdots,p_k$**为素数**

根据定理4可证明

定理6:$a^{\varphi(p)}\equiv 1(\%p)\qquad\big(\gcd(a,p)=1\big)$

证明:

设: 与p**互质且小于**p**的**$\varphi(p)$**分别为**$b_1,b_2,\cdots,b_{\varphi(p)}$

那么推理: $a\cdot b_1\%p,a\cdot b_2\%p,\cdots,a\cdot b_{\varphi(p)}\%p$**两两不同, 且分别等于**$b_1,b_2,\cdots,b_{\varphi(p)}$

推理证明:

> **因为:** a**与**b_k**都与**p**互质, 所以**$a\cdot b_k$**也与**p**互质**
>
> **假设:** $a\cdot b_i=a\cdot b_j(\%p)$
>
> **因为:** $\gcd(a,p)=1$ **且** $1\le b_i<p$ **且** $1\le b_j<p$
>
> **那么:** $b_i=b_j$

所以: $(a\cdot b_1\%p)\cdot(a\cdot b_2\%p)\cdots(a\cdot b_{\varphi(p)}\%p)=(b_1\cdot b_2\cdots b_{\varphi(p)})\%p$

即: $a^{\varphi(p)}\equiv 1(\%p)\quad\big(\gcd(a,p)=1\big)$

引理7: 若$\gcd(x,n)=1\big(x\in[1,n)$ **且** $n>2\big)$**, 那么**$\gcd(n-x,n)=1$ **且** $x!=n-x$

证明:

设: $\gcd(n-x,n)=d$

那么: $n-x=k_1\cdot d$ **且** $n=k_2\cdot d$

则: $x=(k_2-k_1)\cdot d$

因为: $\gcd(x,n)=1$

所以: $\gcd(n-x,n)=d=1$

假设: $x=n-x$

那么: $\gcd(x,n)=\gcd(x,2x)=x=1$

因为: $n>2$**, 所以:** $n-x\neq x$**与假设不符**

定理8: 当$n>2$**时,** $\varphi(n)$**为偶数**

根据引理7可证明: 与n**互素的数总是成对出现**

定理9: $\sum\limits_{d=1}^{n-1}d=\frac{\varphi(n)\cdot n}{2}\qquad\big(\gcd(d,n)=1\big)$

根据引理7,定理8可证明

定理10: $\varphi(k\cdot p)=\varphi(k)\cdot p\qquad(p$**是质数 且** $k\%p=0)$

根据定理5可证明

定理11: $\varphi(k\cdot p)=\varphi(k)\cdot\varphi(p)=\varphi(k)\cdot(p-1)\quad(p$**是质数 且** $k\%p\neq 0)$

根据定理4可证明

定理12: $\varphi(n\cdot m)=\varphi(m)\cdot\varphi(n)\cdot\frac{\gcd(n,m)}{\varphi(\gcd(n,m))}$

```
ll phi(ll x)
{
    ll y=x,p=x;
    for(ll i=2;i*i<=x;i++)
        if(y%i==0)for(p-=p/i;y%i==0;y/=i);
    if(y!=1)p-=p/y;
    return p;
}
```

扩展欧几里得

求解: $a\cdot x+b\cdot y=\gcd(a,b)$

设: $c=b,\ d=a\%b=a-\lfloor\frac{a}{b}\rfloor\cdot b$

$$\begin{cases}a\cdot x+b\cdot y=\gcd(a,b)\\ c\cdot x'+d\cdot y'=\gcd(c,d)=\gcd(b,a\%b)=\gcd(a,b)\end{cases}$$

得:

$$\begin{aligned}a\cdot x+b\cdot y&=c\cdot x'+d\cdot y'\\&=b\cdot x'+(a-\lfloor\frac{a}{b}\rfloor\cdot b)\cdot y'\\&=a\cdot y'+b\cdot(x'-\lfloor\frac{a}{b}\rfloor\cdot y')\end{aligned}$$

$$\begin{cases}x=y'\\ y=x'-\lfloor\frac{a}{b}\rfloor\cdot y'\end{cases}$$

当且仅当: $d=0$**时**$\begin{cases}x'=1\\ y'=0\end{cases}$

设: $a\cdot x+b\cdot y=\gcd(a,b)$**通过Exgcd解得得解为**$(x_0,y_0)$

```
struct eg{ll x,y,r;eg(ll x,ll y,ll r):x(x),y(y),r(r){}};

eg exgcd(ll a,ll b)
{
    if(!b)return eg(1,0,a);
    eg t=exgcd(b,a%b);
    return eg(t.y,t.x-a/b*t.y,t.r);
}
```

那么:

$a\cdot x+b\cdot y=a\cdot x_0+b\cdot y_0$

$a\cdot(x-x_0)=b\cdot(y_0-y)$

$\frac{a}{\gcd(a,b)}\cdot(x-x_0)=\frac{b}{\gcd(a,b)}\cdot(y_0-y)$

显然: $\frac{a}{\gcd(a,b)}$**与**$\frac{b}{\gcd(a,b)}$**互质**

那么: $\begin{cases} x - x_0 = \frac{b}{\gcd(a,b)} \cdot k \\ y_0 - y = \frac{a}{\gcd(a,b)} \cdot k \end{cases}$

得: $\begin{cases} x = x_0 + \frac{b}{\gcd(a,b)} \cdot k \\ y = y_0 - \frac{a}{\gcd(a,b)} \cdot k \end{cases}$

线性同余方程

求解: $a \cdot x \equiv b(\%p)$

设: $a \cdot x = p \cdot (-y) + b$

得: $a \cdot x + p \cdot y = b$

根据拓展欧几里得: $a \cdot x + p \cdot y = \gcd(a,p)$**得到一个解为**$x_0'$

那么: $a \cdot x + p \cdot y = b$**的一个解为**$x_0 = \frac{b}{\gcd(a,p)} \cdot x_0'$

根据拓展欧几里得: $x = x_0 + \frac{p}{\gcd(a,p)} \cdot k$

即: $x = \frac{b}{\gcd(a,p)} \cdot x_0' + \frac{p}{\gcd(a,p)} \cdot k \qquad k \in \big[0, \gcd(a,p)\big)$

中国剩余定理

$$\begin{cases} x_1 \equiv r_1(\%m_1) \\ x_2 \equiv r_2(\%m_2) \\ \cdots \\ x_i \equiv r_i(\%m_i) \\ \cdots \\ x_n \equiv r_n(\%m_n) \end{cases}$$

$\gcd(m_i, m_j) = 1 \qquad (1 \le i < j \le n)$

设: $m = m_1 \times m_2 \times \cdots \times m_n = \prod\limits_{i=1}^{n} m_i$

$\begin{cases} y_1 \equiv 1(\%m_1) \\ y_2 \equiv 1(\%m_2) \\ \cdots \\ y_i \equiv 1(\%m_i) \\ \cdots \\ y_n \equiv 1(\%m_n) \end{cases}$ **且** $\begin{cases} y_1 = k_1 \cdot \frac{m}{m_1} \\ y_2 = k_2 \cdot \frac{m}{m_2} \\ \cdots \\ y_i = k_i \cdot \frac{m}{m_i} \\ \cdots \\ y_n = k_n \cdot \frac{m}{m_n} \end{cases}$

那么其中的一个解为: $x_0 = r_1 \cdot y_1 + r_2 \cdot y_2 + \cdots + r_n \cdot y_n = \sum\limits_{i=1}^{n} (r_i \cdot y_i)$

考虑其中一个等式: $\frac{m}{m_i} \cdot k_i \equiv 1(\%m_i)$

因为: m_i**与** $\frac{m}{m_i}$ **互质**

得: $k_i = \left(\frac{m}{m_i}\right)^{-1}(\%m_i)$

所以: $y_i = \frac{m}{m_i} \cdot \left(\left(\frac{m}{m_i}\right)^{-1}(\%m_i)\right)$

所以: $x_0 = \sum_{i=1}^{n} \left(r_i \cdot \frac{m}{m_i} \cdot \left(\left(\frac{m}{m_i}\right)^{-1}(\%m_i)\right)\right)$

线性同余方程组

$$\begin{cases} x_1 \equiv r_1(\%m_1) \\ x_2 \equiv r_2(\%m_2) \\ \dots \\ x_i \equiv r_i(\%m_i) \\ \dots \\ x_n \equiv r_n(\%m_n) \end{cases}$$

考虑其中任意2个方程: $\begin{cases} x_i \equiv r_i(\%m_i) \\ x_j \equiv r_j(\%m_j) \end{cases}$

等价于: $\begin{cases} x_i = m_i \cdot k_i + r_i \\ x_j = m_j \cdot k_j + r_j \end{cases}$

假设x同时满足上述2个方程

那么: $x = m_i \cdot k_i + r_i = m_j \cdot k_j + r_j$

得: $m_i \cdot k_i - m_j \cdot k_j = r_j - r_i$

即: $m_i \cdot k_i \equiv r_j - r_i(\%m_j)$

若: $(r_j - r_i)\% \gcd(m_i, m_j) \neq 0$**则无解**

通过拓展欧几里得解得: $k_i \in [0, \frac{m_j}{\gcd(m_i,m_j)})$

得: $x = m_i \cdot k_i + r_i + lcm(m_i, m_j) \cdot k$

即: $x \equiv m_i \cdot k_i + r_i\big(\%lcm(m_i, m_j)\big)$

```
ll lce(ll r[],ll m[],int n)
{
    for(int i=2;i<=n;i++)
    {
        eg t=exgcd(m[1],m[i]);
        if((r[i]-r[1])%t.r)return -1;
        ll md=m[i]/t.r;
        t.x=((r[i]-r[1])/t.r*t.x%md+md)%md;
        r[1]+=m[1]*t.x,m[1]=m[1]/t.r*m[i];
    }
    return r[1];
}
```

狄利克雷卷积

定义: $f(n)$, $g(n)$**是两个数论函数(定义域均为正整数)**

定义卷积运算"$\times$**":** $(f \times g)(n) = \sum\limits_{i \cdot j = n} f(i) \cdot g(j) = \sum\limits_{n\%d=0} f(d) \cdot g(\frac{n}{d})$

交换律: $(f \times g)(n) = (g \times f)(n)$

结合律: $((f \times g) \times h)(n) = (f \times (g \times h))(n)$

单位元:

设: $e(n)$**为卷积运算单位元**

那么有: $(f \times e)(n) = f(n)$

$\sum\limits_{n\%d=0} f(d) \cdot e(\frac{n}{d}) = f(n)$

所以: $e(\frac{n}{d}) = \begin{cases} 1 & (d = n) \\ 0 & (d \neq n) \end{cases}$

即: $e(n) = \begin{cases} 1 & (n = 1) \\ 0 & (n \neq 1) \end{cases}$

定义函数: $l(n) = 1$

那么: $f(n) = \sum\limits_{n\%d=0} g(d)$**可以写成:** $f(n) = (g \times l)(n)$

定义: 莫比乌斯函数$\mu(n)$**为函数**$l(n)$**的逆元**

那么: $(\mu \times l)(n) = e(n)$

得: $\sum\limits_{n\%d=0} \mu(n) = e(n)$

那么:

$$
\begin{aligned}
g(n) &= (f \times \mu)(n) \\
&= \sum_{n\%d=0} \mu(\frac{n}{d}) \cdot f(d)
\end{aligned}
$$

杜教筛

作用: 在低于线性复杂度内解决某一类函数的前缀和

求数论函数$f(x)$**的前缀和**$s(n)$**, 即:** $s(n) = \sum\limits_{i=1}^{n} f(i)$

找到一个数论函数$g(x)$**, 使得**$g(x)$**与**$f(x)$**的狄利克雷卷积**$(f \times g)(n) = \sum\limits_{d|n} g(d) \cdot f(\frac{n}{d})$**的前缀和易求得**

$$
\begin{aligned}
& \sum_{i=1}^{n} (f \times g)(i) \\
&= \sum_{i=1}^{n} \sum_{d|i} g(d) \cdot f(\frac{i}{d}) \\
&= \sum_{d=1}^{n} \sum_{d|i}^{n} g(d) \cdot f(\frac{i}{d}) \\
&= \sum_{d=1}^{n} \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} g(d) \cdot f(i) \\
&= \sum_{d=1}^{n} g(d) \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} f(i) \\
&= \sum_{d=1}^{n} g(d) \cdot s(\lfloor \frac{n}{d} \rfloor) \\
&= g(1) \cdot s(n) + \sum_{d=2}^{n} g(d) \cdot s(\lfloor \frac{n}{d} \rfloor)
\end{aligned}
$$

即: $g(1)\cdot s(n)=\sum\limits_{i=1}^{n}(f\times g)(i)-\sum\limits_{d=2}^{n}g(d)\cdot s(\lfloor\frac{n}{d}\rfloor)$

莫比乌斯函数$\mu(x)$前缀和

$$\begin{cases}f(x)=\mu(x)\\ g(x)=1\end{cases}\quad\Rightarrow\quad(f\times g)(x)=\sum_{d|x}\mu(d)=e(x)=\begin{cases}1 & (x=1)\\ 0 & (x\neq 1)\end{cases}$$

即: $s(n)=1-\sum\limits_{i=2}^{n}s(\lfloor\frac{n}{i}\rfloor)$

```
const int __=5e6+5;     //n^(2/3)项

bool pri[__];
int mu[__],num[__];
ll sum[__];     //莫比乌斯函数前缀和

//线性筛莫比乌斯函数

unordered_map<ll,ll>dp;

ll cal(ll x)
{
    if(x<__)return sum[x];
    if(dp[x])return dp[x];
    ll res=1;
    for(ll l=2,r;l<=x;l=r+1)
    {
        r=x/(x/l);
        res-=(r-l+1)*cal(x/l);
    }
    return dp[x]=res;
}
```

欧拉函数$\varphi(x)$前缀和

$$\begin{cases}f(x)=\varphi(x)\\ g(x)=1\end{cases}\quad\Rightarrow\quad(f\times g)(x)=\sum_{d|x}\varphi(d)=x$$

即: $s(n)=\frac{(n+1)\cdot n}{2}-\sum\limits_{i=2}^{n}s(\lfloor\frac{n}{i}\rfloor)$

```
const int __=5e6+5;
const int mod=1e9+7;
const int inv2=(mod+1)/2;

ll md(ll x)
{
    if(x<=-mod || x>=mod)
        x%=mod;
    if(x<0)x+=mod;
    return x;
}
```

```
int phi[__],num[__];
ll sum[__];     //欧拉函数前缀和

//线性筛欧拉函数(取模)

unordered_map<ll,ll>dp;

ll cal(ll x)
{
    if(x<__)return sum[x];
    if(dp[x])return dp[x];
    ll res=md(md((md(x)+1ll)*md(x))*inv2);
    for(ll l=2,r;l<=x;l=r+1)
    {
        r=x/(x/l);
        res=md(res-md(md(r-l+1)*cal(x/l)));
    }
    return dp[x]=res;
}
```

min_25筛

```
namespace math
{
    const int __=5e5+5;

    bool pri[__];
    int num[__];

    //线性筛质数

    ll f[__<<1],v[__<<1],n;int sqn,m;

    int get(ll x)
    {
        if(x<=sqn)return x;
        return m+1-n/x;
    }

    ll min_25(ll _n)
    {
        n=_n;m=0;
        sqn=sqrt(n+0.1);
        for(ll i=n;i>=1;i=n/(n/i+1))
            ++m,f[m]=(v[m]=n/i)-1;
        for(int i=1;num[i]<=sqn;++i)
            for(int j=m;;--j)
            {
                int x=get(v[j]/num[i]);
                if(v[x]<num[i])break;
                f[j]-=f[x]-i+1;
            }
```

```
            return f[m];
        }
    }
```

最大公约数求和

$$\sum_{i=1}^{n}\gcd(n,i)$$

设: $\gcd(n,i)=d$

显然: $n\%d=0$

原问题转换为:

$$\sum_{d|n}d\cdot\sum_{i=1}^{n}\left[\gcd(n,i)=d\right]$$

$$\sum_{d|n}d\cdot\sum_{i=1}^{\frac{n}{d}}\left[\gcd(\tfrac{n}{d},i)=1\right]$$

$$\sum_{d|n}d\cdot\varphi(\tfrac{n}{d})$$

不难发现$\sum\limits_{d|n}d\cdot\varphi(\frac{n}{d})$**是狄利克雷卷积形式**

$f(n)=n$ **与** $\varphi(n)$**均为积性函数**

所以$g(n)=\sum\limits_{d|n}d\cdot\varphi(\frac{n}{d})$**为积性函数**

若: $n=p_1^{e_1}\cdot p_2^{e_2}\cdots p_k^{e_k}\qquad p_1,p_2,\cdots,p_k$**均为质数**

那么: $g(n)=g(p_1^{e_1})\cdot g(p_2^{e_2})\times\cdots\times g(p_k^{e_k})$

考虑: $g(p^k)=\sum\limits_{d|p^k}d\cdot\varphi(\frac{p^k}{d})\qquad p$**为质数**

即: $g(p^k)=\sum\limits_{i=0}^{k}p^{k-i}\cdot\varphi(p^i)=\sum\limits_{i=1}^{k}p^{k-i}\cdot\varphi(p^i)+p^k\cdot\varphi(1)$

$$g(p^k)=p^k+\sum_{i=1}^{k}p^{k-i}\cdot(p^i-p^{i-1})=p^k+\sum_{i=1}^{k}(p^k-p^{k-1})$$

$$g(p^k)=(k+1)\cdot p^k-k\cdot p^{k-1}$$

复杂度: $O(\sqrt{n})$

$$\sum_{i=1}^{m}\gcd(n,i)$$

$$\sum_{d|n}d\cdot\sum_{i=1}^{m}\left[\gcd(n,i)=d\right]$$

$$\sum_{d|n}d\cdot\sum_{i=1}^{\lfloor\frac{m}{d}\rfloor}\left[\gcd(\tfrac{n}{d},i)=1\right]$$

设: $$\begin{aligned} f(x,y) &= \sum_{i=1}^{x}\left[\gcd(y,i)=1\right] \\ &= \sum_{d|y}\mu(d)\cdot\lfloor\frac{x}{d}\rfloor \end{aligned}$$

$$\begin{aligned} \sum_{d|n} d\cdot\sum_{i=1}^{\lfloor\frac{m}{d}\rfloor}\left[\gcd(\frac{n}{d},i)=1\right] &= \sum_{d|n} d\cdot\left(\sum_{g|\frac{n}{d}}\mu(g)\cdot\lfloor\frac{\lfloor\frac{m}{d}\rfloor}{g}\rfloor\right) \\ &= \sum_{g|n}\mu(g)\cdot\left(\sum_{d|\frac{n}{g}} d\cdot\lfloor\frac{\lfloor\frac{m}{g}\rfloor}{d}\rfloor\right) \end{aligned}$$

$\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}\gcd(i,j)$

解法1:

枚举: $\gcd(i,j)=d\in[1,n]$

原问题转换为:

$\sum\limits_{d=1}^{n}d\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}[\gcd(i,j)=d]$

$\sum\limits_{d=1}^{n}d\sum\limits_{i=1}^{\lfloor\frac{n}{d}\rfloor}\sum\limits_{j=1}^{\lfloor\frac{n}{d}\rfloor}[\gcd(i,j)=1]$

设: $f(n)=\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}[\gcd(i,j)=1]$

设: $g(n)=\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{i}[\gcd(i,j)=1]=\sum\limits_{i=1}^{n}\varphi(i)$

不难发现: $f(n)=2\cdot g(n)-1$

$\sum\limits_{d=1}^{n}d\sum\limits_{i=1}^{\lfloor\frac{n}{d}\rfloor}\sum\limits_{j=1}^{\lfloor\frac{n}{d}\rfloor}[\gcd(i,j)=1]=\sum\limits_{d=1}^{n}d\cdot f(\lfloor\frac{n}{d}\rfloor)=\sum\limits_{d=1}^{n}d\cdot(2\cdot g(\lfloor\frac{n}{d}\rfloor)-1)$

显然: $\sum\limits_{d=1}^{n}d\cdot(2\cdot g(\lfloor\frac{n}{d}\rfloor)-1)$**可以分块**

复杂度: $O(n+q\cdot\sqrt{n})$

解法2:

$\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}\gcd(i,j)=2\cdot\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{i}\gcd(i,j)-\frac{(n+1)\cdot n}{2}$

设: $g(n)=\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{i}\gcd(i,j)=\sum\limits_{i=1}^{n}\sum\limits_{d|i}d\cdot\varphi(\frac{i}{d})$

$\sum\limits_{d|n}d\cdot\varphi(\frac{n}{d})$**是积性函数, 可以线性筛, 同时维护它的前缀和就可以**$O(1)$**回答询问**

复杂度: $O(n+q)$

$\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{m}\gcd(i,j)$

解法1:

枚举: $\gcd(i,j)=d\in[1,n]$

原问题转换为:

$\sum\limits_{d=1}^{n}d\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{m}[\gcd(i,j)=d]$

设: $g(x)=\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{m}[\gcd(i,j)=x]$

设: $f(x)=\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{m}[x\big|\gcd(i,j)]=\lfloor\frac{n}{x}\rfloor\cdot\lfloor\frac{m}{x}\rfloor$

莫比乌斯反演: $f(x)=\sum\limits_{x|d}g(d)\quad\Rightarrow\quad g(x)=\sum\limits_{x|d}\mu(\frac{d}{x})f(d)$

$g(x)=\sum\limits_{x|d}\mu(\frac{d}{x})\cdot\lfloor\frac{n}{d}\rfloor\cdot\lfloor\frac{m}{d}\rfloor$

$\sum\limits_{d=1}^{n}d\sum\limits_{d|i}\mu(\frac{i}{d})\cdot\lfloor\frac{n}{i}\rfloor\cdot\lfloor\frac{m}{i}\rfloor$

复杂度: $O(n\log n)$

解法2:

$\sum\limits_{d=1}^{n}d\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{m}[\gcd(i,j)=d]\quad\Rightarrow\quad\sum\limits_{d=1}^{n}d\sum\limits_{i=1}^{\lfloor\frac{n}{d}\rfloor}\sum\limits_{j=1}^{\lfloor\frac{m}{d}\rfloor}[\gcd(i,j)=1]$

莫比乌斯反演得: $\sum\limits_{i=1}^{\lfloor\frac{n}{d}\rfloor}\sum\limits_{j=1}^{\lfloor\frac{m}{d}\rfloor}[\gcd(i,j)=1]=\sum\limits_{i=1}^{\lfloor\frac{n}{d}\rfloor}\mu(i)\cdot\lfloor\frac{\lfloor\frac{n}{d}\rfloor}{i}\rfloor\cdot\lfloor\frac{\lfloor\frac{m}{d}\rfloor}{i}\rfloor$

得: $\sum\limits_{d=1}^{n}d\sum\limits_{i=1}^{\lfloor\frac{n}{d}\rfloor}\mu(i)\cdot\lfloor\frac{\lfloor\frac{n}{d}\rfloor}{i}\rfloor\cdot\lfloor\frac{\lfloor\frac{m}{d}\rfloor}{i}\rfloor$

令: $f(x,y)=\sum\limits_{i=1}^{x}\mu(i)\cdot\lfloor\frac{x}{i}\rfloor\cdot\lfloor\frac{y}{i}\rfloor$

利用数论分块可以在$O(\sqrt{x}+\sqrt{y})$得到函数值

同样的, 已知$f(x,y)$的值, $\sum\limits_{d=1}^{n}d\cdot f(\lfloor\frac{n}{d}\rfloor,\lfloor\frac{m}{d}\rfloor)$可以在$O(\sqrt{n}+\sqrt{m})$内得到答案

所以可以分块套分块, 复杂度: $O\big((\sqrt{n}+\sqrt{m})^2\big)$

解法3:

$\sum\limits_{d=1}^{n}d\sum\limits_{i=1}^{\lfloor\frac{n}{d}\rfloor}\mu(i)\cdot\lfloor\frac{\lfloor\frac{n}{d}\rfloor}{i}\rfloor\cdot\lfloor\frac{\lfloor\frac{m}{d}\rfloor}{i}\rfloor=\sum\limits_{d=1}^{n}d\sum\limits_{i=1}^{\lfloor\frac{n}{d}\rfloor}\mu(i)\cdot\lfloor\frac{n}{i\cdot d}\rfloor\cdot\lfloor\frac{m}{i\cdot d}\rfloor$

令: $k=i\cdot d\qquad k\in[1,n]$

枚举 k: $\sum\limits_{k=1}^{n}\sum\limits_{d|k}d\cdot\mu(\frac{k}{d})\cdot\lfloor\frac{n}{k}\rfloor\cdot\lfloor\frac{m}{k}\rfloor=\sum\limits_{k=1}^{n}\lfloor\frac{n}{k}\rfloor\cdot\lfloor\frac{m}{k}\rfloor\sum\limits_{d|k}d\cdot\mu(\frac{k}{d})$

不难发现: $\sum\limits_{d|k} d\cdot \mu(\frac{k}{d})$**是狄利克雷卷积形式**

$f(n)=n$ **与** $\mu(n)$**均为积性函数**

所以$g(n)=\sum\limits_{d|n} d\cdot \mu(\frac{n}{d})$**为积性函数**

若: $n=p_1^{e_1}\cdot p_2^{e_2}\cdots p_k^{e_k} \qquad p_1,p_2,\cdots,p_k$**均为质数**

那么: $g(n)=g(p_1^{e_1})\cdot g(p_2^{e_2})\times\cdots\times g(p_k^{e_k})$

考虑: $g(p^k)=\sum\limits_{d|p^k} d\cdot \mu(\frac{p^k}{d}) \qquad p$**为质数**

$$g(p^k)=\sum_{i=0}^{k} p^{k-i}\cdot \mu(p^i)=p^k-p^{k-1}=\varphi(p^k)$$

所以: $g(n)=\varphi(n)$

$$\sum_{k=1}^{n}\lfloor\frac{n}{k}\rfloor\cdot\lfloor\frac{m}{k}\rfloor\sum_{d|k} d\cdot \mu(\frac{k}{d})=\sum_{k=1}^{n}\lfloor\frac{n}{k}\rfloor\cdot\lfloor\frac{m}{k}\rfloor\cdot\varphi(k)$$

复杂度: $O\big(n+q\cdot(\sqrt{n}+\sqrt{m})\big)$

动态规划

最长上升子序列

```
int len[__];

int LIS(int a[],int n)
{
    int lis=0;
    for(int i=1;i<=n;++i)
    {
        int x=lower_bound(len+1,len+lis+1,a[i])-len;
        len[x]=a[i],lis=max(x,lis);
    }
    return lis;
}
```

最长公共子序列

```
for(int i=1;i<=lena;++i)
    for(int j=1;j<=lenb;++j)
        if(a[i]==b[j])dp[i][j]=dp[i-1][j-1]+1;
        else dp[i][j]=max(dp[i-1][j],dp[i][j-1]);
```

区间dp

```
for(int l=2;l<=n;l++)
    for(int i=1,j=i+l-1;j<=n;i++,j++)
        for(int k=i;k<j;k++)
            dp[i][j]=min(dp[i][j],dp[i][k]+dp[k+1][j]+w[i][j]);
```

决策单调优化

```
for(int l=2;l<=n;l++)
    for(int i=1,j=i+l-1;j<=n;i++,j++)
        for(int k=s[i][j-1];k<=min(j-1,s[i+1][j]);k++)
        {
            int x=dp[i][k]+dp[k+1][j]+w[i][j];
            if(x<=dp[i][j])s[i][j]=k,dp[i][j]=x;
        }
```

数位dp

回文数个数

```
struct DigitalDynamicProgramming
{
```

```
    int bit[20],a[20];
    ll dp[20][20];

    DigitalDynamicProgramming() {memset(dp,-1,sizeof(dp));}

    ll dfs(int len,int sum,bool lim)
    {
        if(!len)return 1;
        if(!lim && ~dp[len][sum])return dp[len][sum];
        int r=lim?bit[len]:9;
        if(len<=(sum-1)/2)
            if(a[sum-len]>r)return 0;
            else return dfs(len-1,sum,lim && a[sum-len]==r);
        ll res=0;
        for(int i=(len==sum-1);i<=r;++i)
        {
            a[len]=i;
            res+=dfs(len-1,sum,lim && i==bit[len]);
        }
        if(lim)return res;
        return dp[len][sum]=res;
    }

    ll cal(ll x)
    {
        if(x<0)return 0;
        int idx=0;
        for(;x;x/=10)bit[++idx]=x%10;
        ll res=0;
        for(int i=1;i<=idx;++i)
            res+=dfs(i,i+1,i==idx);
        return res+1;
    }
}dp;
```

决策单调队列

```
const int __=50005;
//id点决策[l,r]区间
struct des
{
    int id,l,r;
    des() {}
    des(int x,int y,int z):
        id(x),l(y),r(z) {}
};

int a[__];
fdeque<des>Q;
ll sum[__],dp[__],l;

ll sq(ll x){return x*x;}
```

```
//i点经k点决策
ll cal(int i,int k)
{
    return dp[k]+sq(i-k-1+sum[i]-sum[k]-l);
}

//二分
int bs(const des &t,int i)
{
    int l=max(t.l,i+1),r=t.r,ans=r;
    while(l<=r)
    {
        int mid=(l+r)>>1;
        if(cal(mid,t.id)<=cal(mid,i))
            l=mid+1,ans=mid;
        else r=mid-1;
    }
    return ans;
}

int main()
{
    int n;sf("%d%lld",&n,&l);
    fup(i,1,n)
    {
        sf("%d",&a[i]);
        sum[i]=sum[i-1]+a[i];
    }
    Q.pb(des(0,1,n));
    fup(i,1,n)
    {
        dp[i]=cal(i,Q.front().id);
        if(Q.front().r==i)Q.pop_front();
        if(i==n)break;
        Q.front().l=i+1;
        while(!Q.empty() && cal(Q.back().l,i)<cal(Q.back().l,Q.back().id))
            Q.pop_back();
        if(Q.empty())
        {
            Q.pb(des(i,i+1,n));
            continue;
        }
        des &t=Q.back();
        int r=bs(t,i);
        Q.pop_back();
        Q.pb(des(t.id,t.l,r));
        if(r!=n)Q.pb(des(i,r+1,n));
    }
    pf("%lld\n",dp[n]);
    return 0;
}
```

字符串

最小表示法

```
int min_string(char* s,int len)//将s拼接在s后面
{
    int i=1,j=2,k=0;
    while(i<=len && j<=len && k<=len)
    {
        if(s[i+k]==s[j+k])k++;
        else if(s[i+k]<s[j+k])j+=k+1,k=0;
        else if(s[i+k]>s[j+k])i+=k+1,k=0;
        if(i==j)j++;
    }
    return min(i,j);
}
```

字典树

1. 添加操作

   添加操作格式为 `insert barty 8` , 意思为插入 `barty` 这个单词, 这个单词词频为8次

   注意如果再次添加 `insert barty 8` 操作时, 就会将词频增加为16次(不会出现词频 ≤ 0的情况)

2. 删除操作

   删除操作格式为 `delete barty` , 意思为删除所有 `barty` 这个单词

   如果当前没有删除的词汇, 输出 `"Empty"`

3. 查询操作

   查询操作格式为 `query ty` , 意思为查询当前版本以 `ty` 结尾的单词词频总和

4. 修改操作

   修改操作格式为 `update ty tied` , 意思为将所有结尾是 `ty` 的单词更新为 `tied` 结尾, 比如 `barty` 会变为 `bartied`

   如果不存在 `ty` 结尾的单词, 输出 `Empty`

   如果已经存在 `tied` 结尾的单词, 那么说明存在conflict, 不做合并, 输出 `Conflict`

   如果既不存在 `ty` 结尾的单词, 也已经存在以 `tied` 结尾的单词, 则输出 `Empty`

```
struct Trie
{
    struct node
    {
        int nex[27];ll pfx,wd;
        void clear(){mem(nex,0);pfx=wd=0;}
    }t[1000000];

    struct memory
    {
```

```
    static const int __=1e5+5;
    int idx,trash[__];

    int get()
    {
        if(trash[0])return trash[trash[0]--];
        return ++idx;
    }

    void del(int x){trash[++trash[0]]=x;}

    void clear(){idx=trash[0]=0;}
}M;

Trie() {M.clear();t[0].clear();}

int get_idx(char *s)
{
    int x=0;
    for(int i=1;s[i];i++)
    {
        int k=s[i]-'a'+1;
        if(!t[x].nex[k])return -1;
        x=t[x].nex[k];
    }
    return x;
}

int insert(char *s,ll val,bool prefix=false)
{
    int x=0;
    for(int i=1;s[i];i++)
    {
        int k=s[i]-'a'+1;
        if(!t[x].nex[k])
        {
            int idx=M.get();
            t[x].nex[k]=idx;
            t[idx].clear();
            t[idx].nex[0]=x;
        }
        x=t[x].nex[k];
        t[x].pfx+=val;
    }
    if(!prefix)t[x].wd+=val;
    return x;
}

void erase(char *s,ll val,bool prefix=false)
{
    int x=0,i;
    for(i=1;s[i];i++)
    {
```

```
            x=t[x].nex[s[i]-'a'+1];
            t[x].pfx-=val;
        }
        if(!prefix)t[x].wd-=val;
        for(--i;x && !t[x].pfx;i--)
        {
            M.del(x),x=t[x].nex[0];
            t[x].nex[s[i]-'a'+1]=0;
        }
    }

    ll search(char *s,bool prefix=false)
    {
        int x=0;
        for(int i=1;s[i];i++)
        {
            int k=s[i]-'a'+1;
            if(!t[x].nex[k])return 0;
            x=t[x].nex[k];
        }
        return prefix?t[x].pfx:t[x].wd;
    }

    void update(char *pre,char *now)
    {
        int x=get_idx(pre);
        if(!~x)
        {
            puts("Empty");
            return;
        }
        int y=get_idx(now);
        if(~y)
        {
            puts("Conflict");
            return;
        }
        node p=t[x];
        erase(pre,p.pfx,true);
        int z=insert(now,p.pfx,true);
        t[z].wd=p.wd;
        for(int i=1;i<=26;i++)
        {
            t[z].nex[i]=p.nex[i];
            if(p.nex[i])t[p.nex[i]].nex[0]=z;
        }
    }

    void clear(){M.clear();t[0].clear();}
}T;
```

KMP

```
template<class T>
void get_next(T a[],int lena,T b[],int lenb,int nex[],int res[])
{
    if(a==b)nex[1]=1;
    for(int i=(a==b?2:1),j=1;i<=lena;++i)
        for(res[i]=1;;j=nex[j-1])
            if(a[i]==b[j]){res[i]=++j;break;}
            else if(j==1)break;
}

template<class T>
int kmp(T ys[],int lenys,T pp[],int lenpp,int nex[])
{
    int ans=0;
    for(int i=1,j=1;i<=lenys;++i)
        for(;;j=nex[j-1])
            if(ys[i]==pp[j])
            {
                if(j++==lenpp)//[i-lenpp+1,i]匹配
                    pf("%d\n",i-lenpp+1);
                break;
            }
            else if(j==1)break;
    return ans;
}
```

KMP自动机

```
struct KMPAutomaton
{
    static const int __=1e4+5;
    static const int alp=26;

    static int to_idx(char ch)
    {
        return ch-'A'+1;
    }

    int nex[__][alp+1],n;

#define fail(x) nex[x][0]

    void build(char s[],int len)
    {
        n=len;
        mem(nex[0],0);
        for(int i=0;i<n;++i)
        {
            int k=to_idx(s[i+1]);
```

```
                nex[i][k]=i+1;
                if(i)fail(i+1)=nex[fail(i)][k];
                for(int j=1;j<=alp;++j)
                    if(j!=k)nex[i][j]=nex[fail(i)][j];
            }
            for(int i=1;i<=alp;++i)
                nex[n][i]=nex[fail(n)][i];
        }

        int KMP(char s[],int len)
        {
            int x=0,ans=0;
            for(int i=1;i<=len;++i)
            {
                x=nex[x][to_idx(s[i])];
                ans+=(x==n);
            }
            return ans;
        }
}K;
```

exKMP

```
template<class T>
//res[i]: a[i…n]与b[1…m]的LCP (已知b串exkmp的nex)
void exkmp(T a[],int lena,T b[],int lenb,int nex[],int res[])
{
    int wz=1,maxx=0;
    for(int i=1;a[i]==b[i] && i<=min(lena,lenb);++i)
        maxx=i;
    res[1]=maxx;
    if(!maxx)maxx=1;
    for(int i=2;i<=lena;++i)
        if(i+nex[i-wz+1]-1>=maxx)
        {
            res[i]=maxx-i+1;
            while(i+res[i]<=lena && res[i]+1<=lenb
                  && a[i+res[i]]==b[res[i]+1])
                ++res[i];
            maxx=max(i+res[i]-1,i);
            wz=i;
        }
        else res[i]=nex[i-wz+1];
}

exkmp(b+1,lenb-1,b,lenb,nex,nex+1);
exkmp(a,lena,b,lenb,nex,ext);
```

manacher

```
template<class T>
```

```
int manacher(T s[],int len,int pr[])
{
    int n=len<<1|1,res=1;
    for(int i=n,j=len;i>=1;--i)
        if(i&1)s[i]='#';
        else s[i]=s[j--];
    s[n+1]=0,pr[1]=1;
    int wz=1,maxx=1;
    for(int i=2;i<=n;res=max(res,pr[i++]))
        if(i<=maxx && i+pr[2*wz-i]-1!=maxx)
            pr[i]=min(pr[2*wz-i],maxx-i+1);
        else
        {
            pr[i]=max(maxx-i+1,1);
            while(pr[i]+1<=min(n-i+1,i)
                  && s[i+pr[i]]==s[i-pr[i]])
            ++pr[i];
            maxx=i+pr[i]-1,wz=i;
        }
    return res-1;
}
```

AC自动机

```
struct AhoCorasickAutomaton
{
    static const int alp=26;

    static int to_idx(char ch)
    {
        return ch-'a'+1;
    }

    struct Trie
    {
        static const int __=1000005;
        struct node
        {
            int nex[alp+1],last,num;
            bool add[alp+1];
            void clear()
            {
                num=last=0;
                mem(nex,0);
                mem(add,false);
            }
        }t[__];

        Trie() {clear();}

        node& operator[](int x){return t[x];}
```

```
        int idx;

        void insert(char s[],int len)
        {
            int x=0;
            for(int i=1;i<=len;++i)
            {
                int k=to_idx(s[i]);
                if(!t[x].nex[k])
                {
                    t[x].nex[k]=++idx;
                    t[idx].clear();
                }
                x=t[x].nex[k];
            }
            //标记结尾
        }

        void clear(){idx=0;t[0].clear();}
    }t;

    AhoCorasickAutomaton() {clear();}

#define nex(x) t[x].nex[i]
#define fail(x) t[x].nex[0]

    void get_fail()
    {
        queue<int>Q;Q.push(0);
        while(!Q.empty())
        {
            int x=Q.front();Q.pop();
            for(int i=1;i<=alp;++i)
                if(nex(x))
                {
                    Q.push(nex(x));
//                  for(int y=x;y;y=fail(y))
//                      if(nex(fail(y)))
//                      {
//                          fail(nex(x))=nex(fail(y));
//                          break;
//                      }
                    if(x)fail(nex(x))=nex(fail(x));
                }
                else
                {
                    nex(x)=nex(fail(x));
                    t[x].add[i]=true;
                }
            if(t[fail(x)].num)t[x].last=fail(x);
            else t[x].last=t[fail(x)].last;

        }
```

```
    }

    int ac(char s[],int len)
    {
        int ans=0;
        for(int i=1,x=0;i<=len;++i)
        {
            int k=to_idx(s[i]);
//              while(x && !t[x].nex[k])x=fail(x);
            x=t[x].nex[k];
            for(int y=x;y;y=t[y].last)
                ;//统计答案
        }
        return ans;
    }

    void debug()
    {
        for(int i=0;i<=t.idx;++i)
        {
            pf("t[%d]: fail:%d last:%d\n",i,fail(i),t[i].last);
            for(int j=1;j<=26;++j)
                if(t[i].nex[j])
                    printf("%d(%c) ",t[i].nex[j],j-1+'a');
            puts("\n");
        }
    }

    void clear(){t.clear();}
}aca;
```

后缀数组

```
struct SuffixArray
{
    int id;ll rk;
    bool operator<(const SuffixArray &b)const
    {
        return rk<b.rk;
    }
}sa[__];

//rk[i]: s[i…n]的排名
//sa[i].id: 第i名为s[sa[i].id…n]
template<class T>
void get_sa(T s[],int n,int rk[])
{
    for(int i=1;i<=n;i++)
        sa[i].id=i,sa[i].rk=s[i];
    for(int i=1;;i<<=1)
    {
        sort(sa+1,sa+1+n);
```

```
        for(int j=1;j<=n;j++)
        {
            rk[sa[j].id]=rk[sa[j-1].id];
            if(sa[j].rk!=sa[j-1].rk)
                rk[sa[j].id]++;
        }
        if(rk[sa[n].id]==n)break;
        for(int j=1;j<=n;j++)
        {
            sa[j].id=j;
            sa[j].rk=1ll*rk[j]*__;
            if(j+i<=n)
                sa[j].rk+=rk[j+i];
        }
    }
}

//he[i]: 第i名后缀与第i-1名后缀的LCP
template<class T>
void get_he(T s[],int n,int rk[],int he[])
{
    for(int i=1,cp=0;i<=n;i++)
    {
        if(cp)--cp;
        int j=sa[rk[i]-1].id,maxx=max(i,j);
        while(maxx+cp<=n && s[i+cp]==s[j+cp])
            cp++;
        he[rk[i]]=cp;//h[i]=cp;
    }
}
```

回文树

```
struct PalindromicTree
{
    static const int __=1e5+5;
    static const int alp=26;

    static int to_idx(char ch)
    {
        return ch-'a'+1;
    }

#define fail(x) t[x].nex[0]

    struct node
    {
        int len,times,dep,nex[alp+1];
        void set(int l,int fa,int d)
        {
            len=l,nex[0]=fa,dep=d;
        }
```

```
    void clear()
    {
        len=times=0;
        mem(nex,0);
    }
}t[__];

char s[__<<1];

int idx,pre,saf,l,r;

PalindromicTree() {clear();}

void push_back(char c)
{
    s[++r]=c;
    for(int x;x=r-t[saf].len-1;saf=fail(saf))
        if(x>=l && s[x]==s[r])break;
    int k=to_idx(s[r]);
    if(t[saf].nex[k])saf=t[saf].nex[k];
    else
    {
        int y=fail(saf);
        for(int x;x=r-t[y].len-1;y=fail(y))
            if(x>=l && s[x]==s[r])break;
        y=t[y].nex[k];
        t[++idx].clear();
        t[idx].set(t[saf].len+2,y,t[y].dep+1);
        t[saf].nex[k]=idx,saf=idx;
    }
    if(t[saf].len==r-l+1)pre=saf;
    ++t[saf].times;
    ans+=t[saf].dep;
}

void push_front(char c)
{
    s[--l]=c;
    for(int x;x=l+t[pre].len+1;pre=fail(pre))
        if(x<=r && s[x]==s[l])break;
    int k=to_idx(s[l]);
    if(t[pre].nex[k])pre=t[pre].nex[k];
    else
    {
        int y=fail(pre);
        for(int x;x=l+t[y].len+1;y=fail(y))
            if(x<=r && s[x]==s[l])break;
        y=t[y].nex[k];
        t[++idx].clear();
        t[idx].set(t[pre].len+2,y,t[y].dep+1);
        t[pre].nex[k]=idx,pre=idx;
    }
    if(t[pre].len==r-l+1)saf=pre;
```

```
        ++t[pre].times;
        ans+=t[pre].dep;
    }

    ll solve()
    {
        ll ans=0;
        for(int i=idx;i>=2;--i)
        {
            t[fail(i)].times+=t[i].times;
            ans=max(ans,1ll*t[i].times*t[i].len);
        }
        return ans;
    }

    void clear()
    {
        idx=1,pre=saf=0,l=__,r=l-1;
        t[0].clear(),t[1].clear();
        t[0].set(0,1,0),t[1].set(-1,1,0);
    }
}PT;
```

数据结构

RMQ

```
int dp[logn][__];
void rmq(int n)
{
    for(int j=1;(1<<j)<=n;j++)
        for(int i=1;i+(1<<(j-1))<=n;i++)
            dp[j][i]=min(dp[j-1][i],dp[j-1][i+(1<<(j-1))]);
}
int get_min(int l,int r)
{
    int k=(int)log2(r-l+1);
    return min(dp[k][l],dp[k][r-(1<<k)+1]);
}
rmq(n);
```

树状数组

```
struct BinaryIndexTree
{
    typedef int type;
    type c[__];int n;

    void build(int _n)
    {
        n=_n;
        for(int i=1;i<=n;++i)
            c[i]=0;
    }

    void add(int x,type val)
    {
        for(int i=x;i<=n;i+=i&-i)
            c[i]+=val;
    }

    type sum(int x)
    {
        type res=0;
        for(int i=x;i;i-=i&-i)
            res+=c[i];
        return res;
    }
}B;
```

差分与树状数组

长度为n的序列: $a_1, a_2, \cdots, a_n$

一阶差分与前缀和

一阶差分数组: $d_i = a_i - a_{i-1} \qquad (a_0 = 0)$

考虑序列a第m项:

$$a_m = \sum_{i=1}^{m} d_i$$

考虑序列a前m项和:

$$\sum_{i=1}^{m} a_i = \sum_{i=1}^{m}\sum_{j=1}^{i} d_j = \sum_{i=1}^{m}(m+1-i)\cdot d_i = (m+1)\sum_{i=1}^{m} d_i - \sum_{i=1}^{m} i\cdot d_i$$

树状数组优化上面的代码, 复杂度: $O(n\log n)$

```
struct BinaryIndexTree
{
    static const int __=1e5+5;
    typedef ll type;
    int n;type a[2][__];

    void add(int x,type v)
    {
        for(int i=x;i<=n;i+=i&-i)
        {
            a[0][i]+=v;
            a[1][i]+=x*v;
        }
    }

    void add(int l,int r,type v)
    {
        if(l>r)return;
        add(l,v),add(r+1,-v);
    }

    type sum(int x)
    {
        type b[2]={0};
        for(int i=x;i;i-=i&-i)
        {
            b[0]+=a[0][i];
            b[1]+=a[1][i];
        }
        return (x+1)*b[0]-b[1];
    }

    type sum(int l,int r)
    {
        if(l>r)return 0;
        return sum(r)-sum(l-1);
    }
}B;
```

二阶差分与前缀和

二阶差分数组: $e_i = d_i - d_{i-1} = a_i - 2 \cdot a_{i-1} + a_{i-2} \qquad (a_{-1} = a_0 = 0)$

考虑序列a第m项:

$$a_m = \sum_{i=1}^{m} d_i = \sum_{i=1}^{m}\sum_{j=1}^{i} e_j = (m+1)\sum_{i=1}^{m} e_i - \sum_{i=1}^{m} i \cdot e_i$$

考虑序列a前m项和:

$$\begin{aligned}
\sum_{i=1}^{m} a_i &= \sum_{i=1}^{m}\left((i+1)\sum_{j=1}^{i} e_j - \sum_{j=1}^{i} j \cdot e_j\right) \\
&= \left(\sum_{i=1}^{m} e_i \sum_{j=i+1}^{m+1} j\right) - \left(\sum_{i=1}^{m} e_i \cdot ((m+1-i) \cdot i)\right) \\
&= \left(\sum_{i=1}^{m} e_i \cdot \frac{(m+2+i) \cdot (m+1-i)}{2}\right) - \left(\sum_{i=1}^{m} e_i \cdot ((m+1-i) \cdot i)\right) \\
&= \sum_{i=1}^{m} e_i \cdot \frac{(m+2-i) \cdot (m+1-i)}{2} \\
&= \sum_{i=1}^{m} e_i \cdot \frac{(m^2+3m+2) - (2m+3) \cdot i + i^2}{2} \\
&= \frac{(m+1)(m+2)}{2}\sum_{i=1}^{m} e_i - \frac{2m+3}{2}\sum_{i=1}^{m} i \cdot e_i + \frac{1}{2}\sum_{i=1}^{m} i^2 \cdot e_i
\end{aligned}$$

树状数组优化以上代码, 复杂度: $O(n \log n)$

```
struct BinaryIndexTree
{
    static const int __=1e5+5;
    typedef ll type;
    int n;type a[3][__];

    void add(int x,type v)
    {
        for(int i=x;i<=n;i+=i&-i)
        {
            a[0][i]+=v;
            a[1][i]+=x*v;
            a[2][i]+=x*x*v;
        }
    }
    //a[i]+=b+k*(i-l)
    void add(int l,int r,type k,type b)
    {
        if(l>r)return;
        add(l,b);
        add(l+1,k-b);
        add(r+1,-b-(r-l+1)*k);
        add(r+2,b+(r-l)*k);
    }

    type sum(int x)
```

```
    {
        type b[3]={0};
        for(int i=x;i;i-=i&-i)
        {
            b[0]+=a[0][i];
            b[1]+=a[1][i];
            b[2]+=a[2][i];
        }
        return ((x+1)*(x+2)*b[0]-(2*x+3)*b[1]+b[2])/2;
    }

    type sum(int l,int r)
    {
        if(l>r)return 0;
        return sum(r)-sum(l-1);
    }
}B;
```

k阶差分与前缀和

不难看出: 在维护原数组a_i的k阶差分数组b_i时, a_i的前m项和都是如下几个相同形式的部分求和

$$\sum f(m)\cdot \sum_{i=1}^{m} i^c\cdot b_i \qquad \big(f(x)\text{是关于}x\text{的多项式}\big)$$

考虑一个部分的前m项和:

$$\begin{aligned}\sum_{i=1}^{m} f(i)\sum_{j=1}^{i} j^c\cdot b_j &= \sum_{i=1}^{m} i^c\cdot b_i\sum_{j=i}^{m} f(j) = \sum_{i=1}^{m} i^c\cdot b_i\cdot\left(\sum_{j=1}^{m} f(j)-\sum_{j=1}^{i-1} f(j)\right)\\ &= \sum_{i=1}^{m} i^c\cdot b_i\cdot\big(g(m)-g(i-1)\big) \qquad \left(g(n)=\sum_{i=1}^{n} f(i)\right)\end{aligned}$$

运用上述方法不难求得三阶差分数组f_i与前缀和的关系

$$\sum_{i=1}^{m} a_i = \frac{(m+1)(m+2)(m+3)}{6}\sum_{i=1}^{m} f_i - \frac{3m^2+12m+11}{6}\sum_{i=1}^{m} i\cdot f_i + \frac{m+2}{2}\sum_{i=1}^{m} i^2\cdot f_i - \frac{1}{6}\sum_{i=1}^{m} i^3\cdot f_i$$

树状数组优化以上代码, 复杂度: $O(n\log n)$

```
struct BinaryIndexTree
{
    static const int __=1e5+5;
    typedef ll type;
    int n;type a[4][__];

    void add(int x,type v)
    {
        for(int i=x;i<=n;i+=i&-i)
        {
            a[0][i]+=v;
            a[1][i]+=x*v;
            a[2][i]+=x*x*v;
            a[3][i]+=x*x*x*v;
        }
```

```
    }
    //a[i]+=x*(i-l)*(i-l)+y*(i-l)+z;
    void add(int l,int r,type x,type y,type z)
    {
        if(l>r)return;
        add(l,z);
        add(l+1,x+y-2*z);
        add(l+2,x-y+z);
        type p=r-l+1,q=r-l;
        add(r+1,-p*p*x-p*y-z);
        add(r+2,(p*p+q*q-2)*x+(p+q)*y+2*z);
        add(r+3,-q*q*x-q*y-z);
    }

    type sum(int x)
    {
        type b[4]={0};
        for(int i=x;i;i-=i&-i)
        {
            b[0]+=a[0][i];
            b[1]+=a[1][i];
            b[2]+=a[2][i];
            b[3]+=a[3][i];
        }
        return ((x+1)*(x+2)*(x+3)*b[0]-(3*x*x+12*x+11)*b[1]+(3*x+6)*b[2]-b[3])/6;
    }

    type sum(int l,int r)
    {
        if(l>r)return 0;
        return sum(r)-sum(l-1);
    }
}B;
```

离散化树状数组

```
struct BinaryIndexTree
{
    const static int __=4e5+5;

    ll a[__];int c[__],idx,siz;

    BinaryIndexTree() {clear();}

    void push_back(ll x){a[++idx]=x;}

    int size() {return siz;}

    void build()
    {
        sort(a+1,a+1+idx);
        idx=unique(a+1,a+1+idx)-a-1;
    }
```

```
int get(ll x)
{
    return lower_bound(a+1,a+1+idx,x)-a;
}

void insert(ll x)
{
    ++siz;
    for(int i=get(x);i<=idx;i+=i&-i)
        ++c[i];
}

void erase(ll x)
{
    --siz;
    for(int i=get(x);i<=idx;i+=i&-i)
        --c[i];
}

int sum(int p)
{
    int res=0;
    for(int i=p;i;i-=i&-i)
        res+=c[i];
    return res;
}

//x数的排名
int rank(ll x)
{
    int res=1;
    for(int i=get(x)-1;i;i-=i&-i)
        res+=c[i];
    return res;
}

//第x个数
ll operator[](int x)
{
    int p=idx;
    for(int l=1,r=idx;l<=r;)
    {
        int mid=(l+r)>>1,s=0;
        if(sum(mid)>=x)
            p=mid,r=mid-1;
        else l=mid+1;
    }
    return a[p];
}

//>x的最小数
ll greater(ll x)
```

```
    {
        int p=idx,l=get(x);
        for(int y=sum(l++),r=idx;l<=r;)
        {
            int mid=(l+r)>>1;
            if(sum(mid)>y)
                p=mid,r=mid-1;
            else l=mid+1;
        }
        return a[p];
    }

    //<x的最大数
    ll less(ll x)
    {
        int p=1,r=get(x)-1;
        for(int y=sum(r),l=1;l<=r;)
        {
            int mid=(l+r)>>1;
            if(sum(mid-1)<y)
                p=mid,l=mid+1;
            else r=mid-1;
        }
        return a[p];
    }

    //>x的数的个数
    int upper(ll x){return siz-sum(get(x));}

    //<x的数的个数
    int lower(ll x){return sum(get(x)-1);}
    void clear() {idx=siz=0;mem(c,0);}
}bit;
```

线段树

```
struct SegmentTree
{
    static const int __=100005;

    ll val;int n,ql,qr;
    struct node
    {
        ll val,st,ad;
        node() {}
        void set(ll v,ll s,ll a)
        {
            val=v,st=s,ad=a;
        }
        //改
        void putset(int tl,int tr,ll v)
        {
```

```
        set((tr-tl+1)*v,v,0);
    }
    //改
    void putadd(int tl,int tr,ll v)
    {
        val+=v*(tr-tl+1);
        if(st)st+=v;
        else ad+=v;
    }
}t[__<<2];

//改
void pushup(int x,int tl,int tm,int tr)
{
    t[x].val=t[x<<1].val+t[x<<1|1].val;
}

void build(int _n){n=_n;build(1,1,n);}

void build(int x,int tl,int tr)
{
    if(tl==tr)t[x].set(a[tl],0,0);
    else
    {
        int tm=(tl+tr)>>1;
        build(x<<1,tl,tm);
        build(x<<1|1,tm+1,tr);
        pushup(x,tl,tm,tr);
    }
}

void pushdown(int x,int tl,int tm,int tr)
{
    if(t[x].st)
    {
        t[x<<1].putset(tl,tm,t[x].st);
        t[x<<1|1].putset(tm+1,tr,t[x].st);
        t[x].st=0;
    }
    if(t[x].ad)
    {
        t[x<<1].putadd(tl,tm,t[x].ad);
        t[x<<1|1].putadd(tm+1,tr,t[x].ad);
        t[x].ad=0;
    }
}

bool cut(int x,int tl,int tr)
{
    return ql>tr || qr<tl;
}

bool check(int x,int tl,int tr)
```

```
    {
        return ql<=tl && tr<=qr;
    }

    void set(int _ql,int _qr,ll _val)
    {
        ql=_ql,qr=_qr,val=_val;
        _set(1,1,n);
    }

    void _set(int x,int tl,int tr)
    {
        if(cut(x,tl,tr))return;
        if(check(x,tl,tr))
        {
            t[x].putset(tl,tr,val);
            return;
        }
        int tm=(tl+tr)>>1;
        pushdown(x,tl,tm,tr);
        _set(x<<1,tl,tm);
        _set(x<<1|1,tm+1,tr);
        pushup(x,tl,tm,tr);
    }

    void add(int _ql,int _qr,ll _val)
    {
        ql=_ql,qr=_qr,val=_val;
        _add(1,1,n);
    }

    void _add(int x,int tl,int tr)
    {
        if(cut(x,tl,tr))return;
        if(check(x,tl,tr))
        {
            t[x].putadd(tl,tr,val);
            return;
        }
        int tm=(tl+tr)>>1;
        pushdown(x,tl,tm,tr);
        _add(x<<1,tl,tm);
        _add(x<<1|1,tm+1,tr);
        pushup(x,tl,tm,tr);
    }

    ll get_val(int _ql,int _qr)
    {
        ql=_ql,qr=_qr;
        return _get_val(1,1,n);
    }

    //改
```

```
    ll _get_val(int x,int tl,int tr)
    {
        if(cut(x,tl,tr))return 0;
        if(check(x,tl,tr))return t[x].val;
        ll res=0;int tm=(tl+tr)>>1;
        pushdown(x,tl,tm,tr);
        res+=_get_val(x<<1,tl,tm);
        res+=_get_val(x<<1|1,tm+1,tr);
        pushup(x,tl,tm,tr);
        return res;
    }
}T;
```

高维标记

区间$[l,r]$**加首项为**f_1,f_2,f_3**的线性递推**$f_n=a\cdot f_{n-1}+b\cdot f_{n-2}+c\cdot f_{n-3}$

查询区间和

```
ll md(ll x)
{
    if(0<=x)
    {
        if(x<mod)return x;
        if(x<(mod<<1))return x-mod;
        return x%mod;
    }
    if(x<=-mod)x%=mod;
    return x+mod;
}

struct Matrix
{
    static const int n=3;
    int a[n+1][n+1];
    int* operator[](int x){return a[x];}

    Matrix& operator=(const Matrix &b)//赋值
    {
        for(int i=1;i<=n;++i)
            for(int j=1;j<=n;++j)
                a[i][j]=md(b.a[i][j]);
        return *this;
    }

    Matrix operator*(const Matrix &b)const//矩阵乘法
    {
        Matrix c;c.clear();
        for(int i=1;i<=n;++i)
            for(int k=1;k<=n;++k)//Cache
                for(int j=1;j<=n;++j)
                    c.a[i][j]=md(c.a[i][j]+md(1ll*a[i][k]*b.a[k][j]));
        return c;
```

```
    }

    Matrix operator+(const Matrix &b)const//矩阵加法
    {
        Matrix c;c.clear();
        for(int i=1;i<=n;++i)
            for(int j=1;j<=n;++j)
                c.a[i][j]=md(a[i][j]+b.a[i][j]);
        return c;
    }

    void print()
    {
        for(int i=1;i<=n;++i)
            for(int j=1;j<=n;++j)
                pf("%d%c",a[i][j]," \n"[j==n]);
    }

    void clear()
    {
        for(int i=1;i<=n;++i)
            for(int j=1;j<=n;++j)
                a[i][j]=0;
    }
}m[__],sm[__];
//m[i]=pow(M,i),sm[i]=sm[i-1]+m[i];

struct Vector
{
    static const int n=Matrix::n;
    int a[n+1];

    int& operator[](int x){return a[x];}

    Vector& operator=(const Vector &b)//赋值
    {
        for(int i=1;i<=n;++i)
            a[i]=md(b.a[i]);
        return *this;
    }

    Vector operator*(const Matrix &b)const//向量乘矩阵
    {
        Vector c;c.clear();
        for(int j=1;j<=n;++j)
            for(int i=1;i<=n;++i)//Cache
                c.a[i]=md(c.a[i]+md(1ll*a[j]*b.a[j][i]));
        return c;
    }

    Vector& operator+=(const Vector &b)//向量加法
    {
        for(int i=1;i<=n;++i)
```

```
            a[i]=md(a[i]+b.a[i]);
        return *this;
    }

    bool empty()
    {
        for(int i=1;i<=n;++i)
            if(a[i])return false;
        return true;
    }

    void clear() {for(int i=1;i<=n;++i)a[i]=0;}
};

void init(int ___,Vector x)//系数向量{0,f[i-1],f[i-2],……f[i-n]}
{
    const int n=Matrix::n;
    Matrix M;//系数阵
    for(int i=1;i<=n;++i)
    {
        M[i][1]=md(x[i]);
        for(int j=2;j<=n;++j)
            if(j==i+1)M[i][j]=1;
            else M[i][j]=0;
    }

    //单位阵
    for(int i=1;i<=n;++i)
        for(int j=1;j<=n;++j)
            if(i==j)m[0][i][j]=sm[0][i][j]=1;
            else m[0][i][j]=sm[0][i][j]=0;

    for(int i=1;i<=___;++i)
    {
        m[i]=m[i-1]*M;
        sm[i]=sm[i-1]+m[i];
    }
}

int a[__];

struct SegmentTree
{
    Vector val;int n,ql,qr;
    struct node
    {
        int val;Vector ad;

        void putadd(int tl,int tr,Vector v)
        {
            ad+=v;
            Vector x=v*sm[tr-tl];
            val=md(val+x[Matrix::n]);
```

```
    }

    void clear(){val=0;ad.clear();}
}t[__<<2];

void pushup(int x)
{
    t[x].val=md(t[x<<1].val+t[x<<1|1].val);
}

void build(int _n){n=_n;build(1,1,n);}

void build(int x,int tl,int tr)
{
    t[x].clear();
    if(tl==tr){t[x].val=a[tl];return;}
    int tm=(tl+tr)>>1;
    build(x<<1,tl,tm);
    build(x<<1|1,tm+1,tr);
    pushup(x);
}

void pushdown(int x,int tl,int tm,int tr)
{
    if(!t[x].ad.empty())
    {
        t[x<<1].putadd(tl,tm,t[x].ad);
        t[x<<1|1].putadd(tm+1,tr,t[x].ad*m[tm+1-tl]);
        t[x].ad.clear();
    }
}

void add(int _ql,int _qr,Vector _val)//{f[n],f[n-1],……,f[1]}
{
    ql=_ql,qr=_qr,val=_val;
    _add(1,1,n);
}

void _add(int x,int tl,int tr)
{
    if(ql>tr || qr<tl)return;
    if(ql<=tl && tr<=qr)
    {
        t[x].putadd(tl,tr,val*m[tl-ql]);
        return;
    }
    int tm=(tl+tr)>>1;
    pushdown(x,tl,tm,tr);
    _add(x<<1,tl,tm);
    _add(x<<1|1,tm+1,tr);
    pushup(x);
}
```

```
    int get_val(int _ql,int _qr)
    {
        ql=_ql,qr=_qr;
        return _get_val(1,1,n);
    }

    int _get_val(int x,int tl,int tr)
    {
        if(ql>tr || qr<tl)return 0;
        if(ql<=tl && tr<=qr)return t[x].val;
        int res=0,tm=(tl+tr)>>1;
        pushdown(x,tl,tm,tr);
        res=md(_get_val(x<<1,tl,tm)+_get_val(x<<1|1,tm+1,tr));
        pushup(x);
        return res;
    }
}T;
```

段树(扫描线)

```
struct SegmentTree
{
    static const int __=1e5+5;
    typedef double type;
    type a[__];int n;

    struct seg
    {
        type val,len;
        int times;
        seg() {}
        seg(type l,type v,int t) {set(l,v,t);}
        void set(type l,type v,int t)
        {
            len=l,val=v,times=t;
        }
    }t[__<<2];

    void pushup(int x)
    {
        if(t[x].times)
            t[x].val=t[x].len;
        else
            t[x].val=t[x<<1].val+t[x<<1|1].val;
    }

    void build(int x=1,int tl=1,int tr=-1)
    {
        if(tr==-1)tr=n;
        if(tl+1==tr)t[x].set(a[tr]-a[tl],0,0);
        else
        {
            int tm=(tl+tr)>>1;
```

```
            build(x<<1,tl,tm);
            build(x<<1|1,tm,tr);
            t[x].len=t[x<<1].len+t[x<<1|1].len;
            pushup(x);
        }
    }

    void add(type ql,type qr,int v)
    {
        int l=lower_bound(a+1,a+1+n,ql)-a;
        int r=lower_bound(a+1,a+1+n,qr)-a;
        add(l,r,v,1,1,n);
    }

    void add(int ql,int qr,int v,int x,int tl,int tr)
    {
        if(ql>=tr || qr<=tl)return;
        if(ql<=tl && tr<=qr)
        {
            t[x].times+=v;
            pushup(x);
            return;
        }
        int tm=(tl+tr)>>1;
        add(ql,qr,v,x<<1,tl,tm);
        add(ql,qr,v,x<<1|1,tm,tr);
        pushup(x);
    }
    type get_sum() {return t[1].val;}
}T;
```

点树(扫描线)

```
struct SegmentTree
{
    static const int __=1e5+5;
    typedef ll type;
    type a[__];int n;

    struct seg
    {
        type val,len;
        ll times;
        seg() {}
        seg(type l,type v,ll t) {set(l,v,t);}
        void set(type l,type v,ll t)
        {
            len=l,val=v,times=t;
        }
    }t[__<<2];

    void pushup(int x)
    {
```

```
        if(!t[x].times)t[x].val=0;
        else t[x].val=t[x].times;
        t[x].val+=max(t[x<<1].val,t[x<<1|1].val);
    }

    void build(int x=1,int tl=1,int tr=-1)
    {
        if(tr==-1)tr=n;
        if(tl==tr)t[x].set(0,0,0);
        else
        {
            int tm=(tl+tr)>>1;
            build(x<<1,tl,tm);
            build(x<<1|1,tm+1,tr);
            t[x].len=t[x<<1].len+t[x<<1|1].len;
            pushup(x);
        }
    }

    void add(type ql,type qr,ll v)
    {
        int l=lower_bound(a+1,a+1+n,ql)-a;
        int r=lower_bound(a+1,a+1+n,qr)-a;
        add(l,r,v,1,1,n);
    }

    void add(int ql,int qr,ll v,int x,int tl,int tr)
    {
        if(ql>tr || qr<tl)return;
        if(ql<=tl && tr<=qr)
        {
            t[x].times+=v;
            pushup(x);
            return;
        }
        int tm=(tl+tr)>>1;
        add(ql,qr,v,x<<1,tl,tm);
        add(ql,qr,v,x<<1|1,tm+1,tr);
        pushup(x);
    }
    type get_sum() {return t[1].val;}
}T;
```

区间合并

```
struct SegmentTree
{
    const static int __=1e5+5;
    int n,ql,qr;
    struct node
    {
        int tl,tr,v,lv,rv;
        node() {}
```

```
    void set(int l,int r){tl=l,tr=r;}
    void set(int _v,int _lv,int _rv)
    {
        v=_v,lv=_lv,rv=_rv;
    }
    int length(){return tr-tl+1;}
    void clear() {v=lv=rv=0;}
}t[__<<2];

node pushup(node &ls,node &rs)
{
    if(!ls.v)return rs;
    if(!rs.v)return ls;
    int lv=ls.lv,rv=rs.rv,v=max(ls.v,rs.v);
    if(a[ls.tr]==a[rs.tl])
    {
        if(ls.lv==ls.length())
            lv+=rs.lv;
        if(rs.rv==rs.length())
            rv+=ls.rv;
        v=max(v,ls.rv+rs.lv);
    }
    node res;
    res.set(ls.tl,rs.tr);
    res.set(v,lv,rv);
    return res;
}

void build(int _n){build(1,1,n=_n);}

void build(int x,int tl,int tr)
{
    t[x].set(tl,tr);
    if(tl==tr){t[x].set(1,1,1);return;}
    int tm=(tl+tr)>>1;
    build(x<<1,tl,tm);
    build(x<<1|1,tm+1,tr);
    t[x]=pushup(t[x<<1],t[x<<1|1]);
}

int get_val(int _ql,int _qr)
{
    ql=_ql,qr=_qr;
    return get_val(1).v;
}

node get_val(int x)
{
    if(ql<=t[x].tl && t[x].tr<=qr)
        return t[x];
    int tm=(t[x].tl+t[x].tr)>>1;
    node l,r;
    l.clear(),r.clear();
```

```
        if(ql<=tm)l=get_val(x<<1);
        if(qr>tm)r=get_val(x<<1|1);
        return pushup(l,r);
    }
}T;
```

可持久化线段树

区间第k小数

```
//离散化模板

//权值主席树
struct PersistentSegmentTree
{
    #define ls(x) t[x].lson
    #define rs(x) t[x].rson

    const static int __=1e5+5;
    const static int nlogn=__*((int)log2(__)+2);

    struct node
    {
        int lson,rson,val;
        void clear()
        {
            val=lson=rson=0;
        }
    }t[nlogn];

    int n,idx,pos,val;
    int root[__],rt;

    PersistentSegmentTree() {t[0].clear();clear();}

    void build(int _n)
    {
        clear();n=_n;D.clear();
        for(int i=1;i<=n;++i)
            D.pb(a[i]);
        D.build();
        for(int i=1;i<=n;++i)
            add(D.get(a[i]),1);
    }

    void pushup(int x)
    {
        t[x].val=t[ls(x)].val+t[rs(x)].val;
    }

    //位置x的值+v
    void add(int x,int v)
    {
```

```
        pos=x,val=v;
        root[++rt]=++idx;
        t[idx].clear();
        _add(root[rt-1],root[rt],1,n);
    }

    //pn:前一个节点  nn:当前节点
    void _add(int pn,int nn,int tl,int tr)
    {
        int tm=(tl+tr)>>1;
        if(tl==pos && pos==tr)
        {
            t[nn].val=t[pn].val+val;
            return;
        }
        if(pos<=tm)
        {
            rs(nn)=rs(pn),ls(nn)=++idx;
            t[idx].clear();
            _add(ls(pn),idx,tl,tm);
        }
        else
        {
            ls(nn)=ls(pn),rs(nn)=++idx;
            t[idx].clear();
            _add(rs(pn),idx,tm+1,tr);
        }
        pushup(nn);
    }

    //区间[l,r]第k大的值
    ll kth(int l,int r,int k)
    {
        return D[_kth(root[l-1],root[r],1,n,k)];
    }

    int _kth(int ln,int rn,int tl,int tr,int k)
    {
        if(tl==tr)return tl;
        int lv=t[ls(rn)].val-t[ls(ln)].val;
        int tm=(tl+tr)>>1;
        if(k<=lv)
            return _kth(ls(ln),ls(rn),tl,tm,k);
        return _kth(rs(ln),rs(rn),tm+1,tr,k-lv);
    }

    void clear() {idx=rt=0;}
}pst;
```

可持久化数组

```
struct PersistentArray
{
```

```
const static int __=1e6+5;

#define ls(x) t[x].lson
#define rs(x) t[x].rson

struct node
{
    int val,lson,rson;
    void clear()
    {
        val=lson=rson=0;
    }
}t[__*20];

int *a,n,idx;
int root[__*20],rt;

PersistentArray() {clear();}

void build(int _a[],int _n)
{
    a=_a,n=_n;
    root[0]=++idx;
    t[idx].clear();
    build(root[0],1,n);
}

void build(int x,int tl,int tr)
{
    if(tl==tr)
    {
        t[x].val=a[tl];
        return;
    }
    int tm=(tl+tr)>>1;
    t[ls(x)=++idx].clear();
    build(ls(x),tl,tm);
    t[rs(x)=++idx].clear();
    build(rs(x),tm+1,tr);
}

//基于版本y，x位置的值改为v，返回版本号
int set(int y,int x,int v)
{
    y=root[y];
    t[root[++rt]=++idx].clear();
    _set(y,root[rt],x,v);
    return rt;
}

void _set(int pn,int nn,int pos,int val)
{
    int tl=1,tr=n;
```

```
        while(tl!=tr)
        {
            int tm=(tl+tr)>>1;
            if(pos<=tm)
            {
                rs(nn)=rs(pn),ls(nn)=++idx;
                pn=ls(pn),nn=idx,tr=tm;
            }
            else
            {
                ls(nn)=ls(pn),rs(nn)=++idx;
                pn=rs(pn),nn=idx,tl=tm+1;
            }
        }
        t[idx].val=val;
    }

    //版本y中位置pos的元素
    int get(int y,int pos)
    {
        root[++rt]=root[y];     //可持久化并查集:删掉这一行
        int tl=1,tr=n,x=root[y];
        while(tl!=tr)
        {
            int tm=(tl+tr)>>1;
            if(pos<=tm)
                x=ls(x),tr=tm;
            else
                x=rs(x),tl=tm+1;
        }
        return t[x].val;
    }

    void clear() {idx=rt=0;}
}pa;
```

可持久化并查集

```
struct PersistentDisjointSetUnion
{
    static const int __=1e6+5;
    int pre[__],n;
    int root[__],t;      //root[t]=x:时间t对应数组的版本为x
    PersistentDisjointSetUnion() {}

    void init(int _n)
    {
        n=_n,root[0]=t=0;
        for(int i=1;i<=n;++i)
            pre[i]=-1;
        pa.clear();
        pa.build(pre,n);
    }
```

```
    //判断x和y是否在同一集合
    bool same(int x,int y)
    {
        x=fd(x),y=fd(y);
        ++t,root[t]=root[t-1];
        return x==y;
    }

    //合并x和y
    void un(int x,int y)
    {
        x=fd(x),y=fd(y);
        if(x!=y)
        {
            int sx=pa.get(root[t],x);
            int sy=pa.get(root[t],y);
            if(sx>sy)swap(x,y);
            int z=pa.set(root[t],x,sx+sy);
            root[++t]=pa.set(z,y,x);
        }
        else ++t,root[t]=root[t-1];
    }

    //回到y版本
    void back(int y)
    {
        root[++t]=root[y];
    }

    int fd(int x)
    {
        int y=pa.get(root[t],x);
        if(y<0)return x;
        return fd(y);
    }

    void print()
    {
        pf("t:%d\n",t);
        for(int i=1;i<=n;++i)
            pf("%d%c",pa.get(root[t],i)," \n"[i==n]);
    }
}pdsu;
```

分块

区间加法/询问区间小于某个数的个数

```
struct Block
{
    static const int __=50005;
    static const int _b_=300;
```

```
    ll a[__];int n,bsz,bel[__];
    ll ad[_b_];
    vector<ll>ord[_b_];

    ll operator[](int x){return a[x]+ad[bel[x]];}

    void build()
    {
        bsz=(int)sqrt(n);
        for(int i=1;i<=n;i++)
            ord[bel[i]=(i-1)/bsz+1].pb(a[i]);
        for(int i=1;i<=bel[n];i++)
            sort(ord[i].begin(),ord[i].end());
    }

    void rebuild(int x)
    {
        int r=(bel[n]==x)?n:(x*bsz);
        ord[x].clear();
        for(int i=(x-1)*bsz+1;i<=r;i++)
            ord[x].pb(a[i]);
        sort(ord[x].begin(),ord[x].end());
    }

    void add(int l,int r,ll val)
    {
        for(int i=l;i<=min(r,bel[l]*bsz);i++)
            a[i]+=val;
        rebuild(bel[l]);
        if(bel[l]==bel[r])return;
        for(int i=bel[l]+1;i<bel[r];i++)
            ad[i]+=val;
        for(int i=(bel[r]-1)*bsz+1;i<=r;i++)
            a[i]+=val;
        rebuild(bel[r]);
    }

    int get_min(int l,int r,ll val)
    {
        int res=0;
        for(int i=l;i<=min(r,bel[l]*bsz);i++)
            if(a[i]+ad[bel[i]]<val)res++;
        if(bel[l]==bel[r])return res;
        for(int i=bel[l]+1;i<bel[r];i++)
            res+=lower_bound(ord[i].begin(),ord[i].end(),val-ad[i])-ord[i].begin();
        for(int i=(bel[r]-1)*bsz+1;i<=r;i++)
            if(a[i]+ad[bel[i]]<val)res++;
        return res;
    }

    void clear(){memset(ad,0,sizeof(ad));}
}B;
```

单点插入/询问单点值

```
struct Block
{
    static const int __=200005;
    static const int _b_=1000;
    int a[__];
    vector<int>blo[_b_];
    int n,bsz,bel[__];

    int operator[](int x)
    {
        int idx=1;
        for(;x>blo[idx].size();idx++)
            x-=blo[idx].size();
        return blo[idx][x-1];
    }

    void build()
    {
        bsz=(int)sqrt(n);
        for(int i=1;i<=n;i++)
            blo[bel[i]=(i-1)/bsz+1].pb(a[i]);
    }

    void rebuild()
    {
        n=0;
        for(int i=1;blo[i].size();blo[i++].clear())
            for(int j=0;j<blo[i].size();j++)
                a[++n]=blo[i][j];
        build();
    }

    void insert(int x,int val)
    {
        int idx=1;
        for(;x>blo[idx].size();idx++)
            x-=blo[idx].size();
        blo[idx].insert(blo[idx].begin()+x-1,val);
        if(blo[idx].size()>bsz*17)
            rebuild();
    }
}B;
```

非旋式Treap

```
struct Treap
{
    #define fa(x) t[x].nex[0]
    #define ls(x) t[x].nex[1]
    #define rs(x) t[x].nex[2]
```

```
const static int __=2e5+5;
static int rd()
{
    static int seed=2333;
    return seed=seed*482711ll%2147483647;
}

struct node
{
    ll val,minn,ad;
    int nex[3],siz,key;
    bool rev;

    void set(ll v)
    {
        val=minn=v;
    }

    void operator+=(const node &b)
    {
        siz+=b.siz;
        minn=min(minn,b.minn);
    }

    void putrev(){rev=!rev;}

    void putadd(ll v)
    {
        val+=v,minn+=v,ad+=v;
    }

    void clear()
    {
        key=rd();
        rev=false;
        siz=1,ad=0;
        mem(nex,0);
    }
}t[__];

int root;

void pushup(int x)
{
    t[x].siz=1,t[x].minn=t[x].val;
    if(ls(x))t[x]+=t[ls(x)];
    if(rs(x))t[x]+=t[rs(x)];
}

void pushdown(int x)
{
    if(t[x].rev)
```

```
        {
            swap(ls(x),rs(x));
            if(ls(x))t[ls(x)].putrev();
            if(rs(x))t[rs(x)].putrev();
            t[x].rev=0;
        }
        if(t[x].ad)
        {
            if(ls(x))t[ls(x)].putadd(t[x].ad);
            if(rs(x))t[rs(x)].putadd(t[x].ad);
            t[x].ad=0;
        }
    }

    struct memory
    {
        static const int __=1e5+5;
        int idx,trash[__];

        int get()
        {
            if(trash[0])return trash[trash[0]--];
            return ++idx;
        }

        void del(int x){trash[++trash[0]]=x;}

        void clear(){idx=trash[0]=0;}
    }M;

    Treap() {clear();}

    void up(int x)
    {
        for(;x;x=fa(x))pushup(x);
    }

    pii split(int x,int p)
    {
        int rt[3]={0},now[3]={0};
        for(int siz1=0;x;)
        {
            pushdown(x);
            int k=(t[ls(x)].siz+1+siz1<=p)?1:2;
            if(!rt[k])rt[k]=x;
            else fa(t[now[k]].nex[3-k]=x)=now[k];
            if(k==1)siz1+=t[ls(x)].siz+1;
            now[k]=x,x=t[x].nex[3-k];
        }
        rs(now[1])=0,up(now[1]);
        ls(now[2])=0,up(now[2]);
        return mp(rt[1],rt[2]);
    }
```

```
int merge(int x,int y)
{
    if(!x || !y)return x?x:y;
    int rt[3]={0,x,y},z=0,d=0;
    while(rt[1] && rt[2])
    {
        int k=(t[rt[1]].key<=t[rt[2]].key)?1:2;
        if(!rt[1] || !rt[2])k=(rt[1]?1:2);
        pushdown(rt[k]);
        if(!rt[0])rt[0]=rt[k];
        else fa(t[z].nex[d]=rt[k])=z;
        z=rt[k],rt[k]=t[rt[k]].nex[d=3-k];
    }
    fa(t[z].nex[d]=rt[1]?rt[1]:rt[2])=z;
    up(z);
    return rt[0];
}

//a[p]后插入一个数p
void insert(int p,ll v)
{
    pii y=split(root,p);
    int x=M.get();
    t[x].clear();
    t[x].set(v);
    root=merge(merge(y.fi,x),y.se);
}

//删除a[p]
void erase(int p)
{
    pii x=split(root,p-1);
    pii y=split(x.se,1);
    root=merge(x.fi,y.se);
}

//a[l]......a[r] +val
void add(int l,int r,ll v)
{
    pii x=split(root,r);
    pii y=split(x.fi,l-1);
    t[y.se].putadd(v);
    root=merge(merge(y.fi,y.se),x.se);
}

//a[l]......a[r] -> a[r]......a[l]
void reversal(int l,int r)
{
    pii x=split(root,r);
    pii y=split(x.fi,l-1);
    t[y.se].putrev();
    root=merge(merge(y.fi,y.se),x.se);
```

```
    }

    //min(a[l]......a[r])
    ll get_min(int l,int r)
    {
        pii x=split(root,r);
        pii y=split(x.fi,l-1);
        ll v=t[y.se].minn;
        root=merge(merge(y.fi,y.se),x.se);
        return v;
    }

    //a[l]......a[r] -> a[r-k+1]......a[r]a[l]......a[r-k]
    void revolve(int l,int r,int k)
    {
        k=(k%(r-l+1)+(r-l+1))%(r-l+1);
        if(!k)return;
        pii x=split(root,r);
        pii y=split(x.fi,l-1);
        pii z=split(y.se,r-l+1-k);
        root=merge(merge(y.fi,merge(z.se,z.fi)),x.se);
    }

    void clear()
    {
        root=0;
        M.clear();
    }
}T;
```

平衡树

全局定义

宏定义

```
#define fa(x) t[x].nex[0]
#define ls(x) t[x].nex[1]
#define rs(x) t[x].nex[2]
```

内存池

```
struct memory
{
    static const int __=1e5+5;
    int idx,trash[__];

    int get()
    {
        if(trash[0])return trash[trash[0]--];
        return ++idx;
    }
```

```
    void del(int x){trash[++trash[0]]=x;}

    void clear(){idx=trash[0]=0;}
}M;
```

rotate

```
void rotate(int x)
{
    int f=fa(x),k=(x==ls(f))?1:2;
    if(fa(f))
        t[fa(f)].nex[f==ls(fa(f))?1:2]=x;
    else root=x;
    t[f].nex[k]=t[x].nex[3-k];
    t[x].nex[3-k]=f;
    fa(x)=fa(f),fa(f)=x;
    if(t[f].nex[k])fa(t[f].nex[k])=f;
    pushup(f),pushup(x);
}
```

AVL

```
struct AVL
{
    //宏定义

    const static int __=1e5+5;

    struct node
    {
        int val;
        int nex[3],cont,siz,h;

        void set(int pre,int v)
        {
            nex[0]=pre,val=v;
        }
        void clear()
        {
            cont=siz=h=1;
            mem(nex,0);
        }
    }t[__];

    int root;

    void pushup(int x)
    {
        t[x].siz=t[x].cont+t[ls(x)].siz+t[rs(x)].siz;
        t[x].h=max(t[ls(x)].h,t[rs(x)].h)+1;
    }
```

```
//内存池

AVL() {clear();}

//rotate

void up(int x)
{
    for(;x;x=fa(x))
    {
        pushup(x);
        int dh=t[ls(x)].h-t[rs(x)].h;
        if(dh<=-2)
        {
            x=rs(x);
            if(t[ls(x)].h<=t[rs(x)].h)rotate(x);
            else x=ls(x),rotate(x),rotate(x);
        }
        if(dh>=2)
        {
            x=ls(x);
            if(t[ls(x)].h>=t[rs(x)].h)rotate(x);
            else x=rs(x),rotate(x),rotate(x);
        }
    }
}

void insert(int v)
{
    int x=root,y=0;
    while(x && t[x].val!=v)
        if(v<t[y=x].val)x=ls(x);
        else x=rs(x);
    if(x)++t[x].cont;
    else
    {
        x=M.get();
        t[x].clear();
        t[x].set(y,v);
        if(!y)root=x;
        else if(v<t[y].val)ls(y)=x;
        else rs(y)=x;
    }
    up(x);
}

void erase(int v)
{
    int x=root;
    while(x && t[x].val!=v)
        if(v<t[x].val)x=ls(x);
        else x=rs(x);
```

```
        if(!x)return;
        --t[x].cont;
        if(!t[x].cont)
        {
loop:
            int k=(x==ls(fa(x)))?1:2;
            if(!ls(x) || !rs(x))
            {
                int y=ls(x)?ls(x):rs(x);
                if(x==root)root=y;
                else t[fa(x)].nex[k]=y;
                fa(y)=fa(x);
            }
            else
            {
                int y=x;
                for(x=ls(x);rs(x);x=rs(x));
                t[y].val=t[x].val;
                t[y].cont=t[x].cont;
                goto loop;
            }
            M.del(x),x=fa(x);
        }
        up(x);
    }

    void clear()
    {
        root=0;
        M.clear();
    }
}T;
```

Splay

```
struct Splay
{
    //宏定义

    const static int __=1e5+5;

    struct node
    {
        int val;
        int nex[3],cont,siz;

        void set(int pre,int v)
        {
            nex[0]=pre,val=v;
        }
        void clear()
        {
            cont=siz=1;
```

```
        mem(nex,0);
    }
}t[__];

int root;

void pushup(int x)
{
    t[x].siz=t[x].cont+t[ls(x)].siz+t[rs(x)].siz;
}

//内存池

Splay() {clear();}

//rotate

void splay(int x,int y=0)
{
    while(fa(x)!=y)
    {
        int f=fa(x),ff=fa(f);
        if(ff==y)rotate(x);
        else
            if((x==ls(f))==(f==ls(ff)))
                rotate(f),rotate(x);
            else rotate(x),rotate(x);
    }
}

void insert(int v)
{
    int x=root,y=0;
    while(x && t[x].val!=v)
        if(v<t[y=x].val)x=ls(x);
        else x=rs(x);
    if(x)++t[x].cont;
    else
    {
        x=M.get();
        t[x].clear();
        t[x].set(y,v);
        if(!y)root=x;
        else if(v<t[y].val)ls(y)=x;
        else rs(y)=x;
    }
    splay(x);
}

void erase(int v)
{
    int x=root,y=0;
    while(x && t[x].val!=v)
```

```
            if(v<t[y=x].val)x=ls(x);
            else x=rs(x);
        if(!x){splay(y);return;}
        splay(x);
        --t[x].cont;
        if(!t[x].cont)
        {
            if(!ls(x) || !rs(x))
            {
                root=ls(x)?ls(x):rs(x);
                fa(root)=0;
            }
            else
            {
                int y=rs(x);
                root=ls(x),fa(ls(x))=0;
                M.del(x);
                for(x=root;rs(x);x=rs(x));
                splay(x);
                rs(x)=y,fa(y)=x;
                pushup(x);
            }
        }
    }

    void clear()
    {
        root=0;
        M.clear();
    }
}T;
```

Scapegoat Tree

```
struct ScapegoatTree
{
    //宏定义

    const static int __=1e5+5;
    constexpr static double alp=0.75;
    struct node
    {
        int val;
        int nex[3],cont,siz,num;

        void set(int pre,int v,int c=1)
        {
            nex[0]=pre,val=v,cont=c;
        }
        void clear()
        {
            cont=siz=num=1;
            mem(nex,0);
```

```
    }
}t[__];

int root,c[__],cont[__];

void pushup(int x)
{
    t[x].num=1+t[ls(x)].num+t[rs(x)].num;
    t[x].siz=t[x].cont+t[ls(x)].siz+t[rs(x)].siz;
}

//内存池

ScapegoatTree() {clear();}

void dfs(int x)
{
    if(ls(x))dfs(ls(x));
    c[++c[0]]=t[x].val;
    cont[c[0]]=t[x].cont;
    M.del(x);
    if(rs(x))dfs(rs(x));
}

int build(int f,int l,int r)
{
    if(l>r)return 0;
    int x=M.get(),m=(l+r)>>1;
    t[x].clear();
    t[x].set(f,c[m],cont[m]);
    ls(x)=build(x,l,m-1);
    rs(x)=build(x,m+1,r);
    pushup(x);
    return x;
}

void rebuild(int x)
{
    int f=fa(x);
    c[0]=0,dfs(x);
    int y=build(f,1,c[0]);
    if(!f)root=y;
    else t[f].nex[(x==ls(f))?1:2]=y;
}

bool check(int x)
{
    double k=t[x].num*alp;
    if(t[ls(x)].num>k || t[rs(x)].num>k)
        return true;
    return false;
}
```

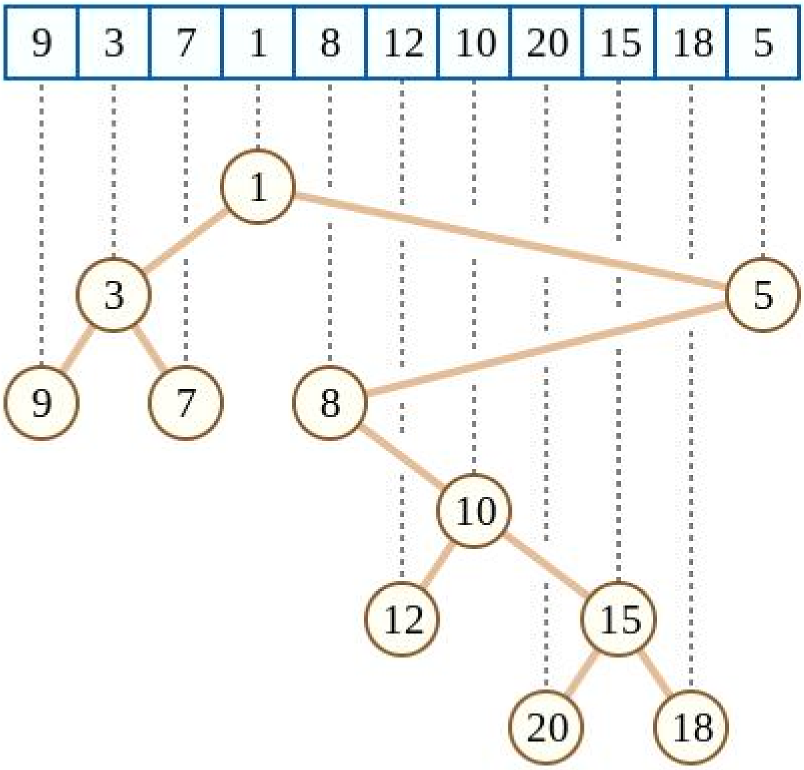
```
    void up(int x)
    {
        int y=0;
        for(;x;x=fa(x))
        {
            pushup(x);
            if(check(x))y=x;
        }
        if(y)rebuild(y);
    }

    //AVL::insert();

    //AVL::erase();

    void clear()
    {
        root=0;
        M.clear();
    }
}T;
```

功能

```
//查询x数的排名(若有多个相同的数，因输出最小的排名)
int kth(int v)
{
    int res=1,x=root;
    while(x)
        if(t[x].val>v)x=ls(x);
        else
        {
            if(ls(x))res+=t[ls(x)].siz;
            if(t[x].val==v)return res;
            res+=t[x].cont;
            x=rs(x);
        }
    return res;
}

//查询排名为x的数
int rank(int k)
{
    int x=root,y=0;
    while(x)
    {
        y=x;
        if(ls(x) && k<=t[ls(x)].siz)
            x=ls(x);
        else
        {
            int z=t[x].cont;
            if(ls(x))
```

```
                z+=t[ls(x)].siz;
            if(k<=z)break;
            k-=z,x=rs(x);
        }
    }
    return t[y].val;
}

//求x的前驱(前驱定义为小于x，且最大的数)
int less(int v)
{
    int x=root,y=0;
    while(x)
    {
        if(v>t[x].val && (y==0 ||
        (y && t[x].val>t[y].val)))
            y=x;
        if(v<=t[x].val)x=ls(x);
        else x=rs(x);
    }
    return t[y].val;
}

//求x的后继(后继定义为大于x，且最小的数)
int greater(int v)
{
    int x=root,y=0;
    while(x)
    {
        if(v<t[x].val && (y==0 ||
        (y && t[x].val<t[y].val)))
            y=x;
        if(v<t[x].val)x=ls(x);
        else x=rs(x);
    }
    return t[y].val;
}
```

笛卡尔树




```
struct CartesianTree
{
    static const int __=50005;

    struct node
    {
        int val,key,id,fa,lson,rson;

        void clear(){fa=lson=rson=0;}

        bool operator<(const node &b)const
        {
            return val<b.val;
        }
    }t[__];

    CartesianTree() {clear();}

    node& operator[](int x){return t[x];}

    int root;
    stack<int>S;

    void build(int n)
    {
        sort(t+1,t+n+1);
```

```
        for(int i=1;i<=n;++i)
        {
            while(!S.empty() && t[S.top()].key>t[i].key)
                S.pop();
            t[i].clear();
            if(S.empty())
            {
                t[root].fa=i;
                t[i].lson=root;
                root=i,S.push(i);
                continue;
            }
            t[i].lson=t[S.top()].rson;
            t[t[S.top()].rson].fa=i;
            t[S.top()].rson=i;
            t[i].fa=S.top();
            S.push(i);
        }
    }

    void clear(){for(root=0;!S.empty();S.pop());}
}ct;
```

计算几何

```
typedef double db;
const db eps=1e-6;
```

点/向量

```
struct point
{
    db x,y,len2,angle;
    point() {}
    point(db _x,db _y){set(_x,_y);}
    void set(db _x,db _y)
    {
        x=_x,y=_y;
        len2=x*x+y*y;
        angle=atan2(y,x);
    }
    void print(){printf("(%.6f,%.6f)",(double)x,(double)y);}
    void println(){print();puts("");}
    //逆时针旋转
    point rotate(db alpha)
    {
        db _x=x*cos(alpha)-y*sin(alpha);
        db _y=x*sin(alpha)+y*cos(alpha);
        return point(_x,_y);
    }
    //向量夹角
    db get_angle(point &p)
    {
        return acos((*this)*p/length()/p.length());
    }
    db length(){return sqrt(len2);}
    point operator+(const point &p){return point(x+p.x,y+p.y);}
    point operator-(const point &p){return point(x-p.x,y-p.y);}
    point operator*(const db k){return point(k*x,k*y);}
    point operator/(const db k){return point(x/k,y/k);}
    db operator*(const point &p){return x*p.x+y*p.y;}
    db operator^(const point &p){return x*p.y-y*p.x;}
    bool operator==(const point &p){return fabs(x-p.x)<eps && fabs(y-p.y)<eps;}
};
```

线

```
struct line
{
    point st,ed,vec;
    db len2,angle;
    line() {}
```

```
    line(point s,point e) {set(s,e);}
    void set(point s,point e)
    {
        st=s,ed=e,vec=e-s;
        len2=vec.len2;
        angle=atan2(vec.y,vec.x);
    }
    //延长k倍
    line operator*(db k){return line(st,st+vec*k);}
    //靠近st的1/k分点
    point operator/(db k){return st+vec/k;}
    db length(){return sqrt(len2);}
    //点在直线左侧
    bool on_left(point p){return (vec^(p-st))>eps;}
    //点在直线上
    bool on_line(point p){return fabs(vec^(p-st))<eps;}
    //点在线段上
    bool on_segment(point p)
    {
        if(p.x+eps<min(st.x,ed.x) || p.x-eps>max(st.x,ed.x))
            return false;
        if(p.y+eps<min(st.y,ed.y) || p.y-eps>max(st.y,ed.y))
            return false;
        return on_line(p);
    }
    //直线平行
    bool parallel(line l){return fabs(vec^l.vec)<eps;}
    //直线重合
    bool coincidence(line l){return on_line(l.st) && on_line(l.ed);}
    //直线交点
    point get_intersection(line l)
    {
        db x1=st.x,y1=st.y,x2=ed.x,y2=ed.y;
        db x3=l.st.x,y3=l.st.y,x4=l.ed.x,y4=l.ed.y;
        db k1=(x4-x3)*(y2-y1),k2=(x2-x1)*(y4-y3);
        db x=(k1*x1-k2*x3+(y3-y1)*(x2-x1)*(x4-x3))/(k1-k2);
        db y=(k2*y1-k1*y3+(x3-x1)*(y2-y1)*(y4-y3))/(k2-k1);
        return point(x,y);
    }
    //线段相交
    bool segment_intersection(line l)
    {
        if(coincidence(l))
            return on_segment(l.st) || on_segment(l.ed)
                || l.on_segment(st) || l.on_segment(ed);
        return ((l.st-st)^vec)*((l.ed-st)^vec)<=0
            && ((st-l.st)^l.vec)*((ed-l.st)^l.vec)<=0;
    }
    //线段中垂线
    line vertical_bisector()
    {
        db x1=st.x,y1=st.y,x2=ed.x,y2=ed.y;
        db xm=(x1+x2)/2.0,ym=(y1+y2)/2.0;
```

```
        return line(point(xm+ym-y1,ym-xm+x1),point(xm-ym+y1,ym+xm-x1));
    }
    //直线旋转
    line rotate(db alpha)
    {
        point v=vec.rotate(alpha);
        return line(st,st+v);
    }
    //垂线
    line vertical(point p)
    {
        point l(vec.y,-vec.x);
        l=l*(distance(p)/l.length());
        if(on_line(p+l))return line(p,p+l);
        else return line(p,p-l);
    }
    //直线夹角
    db get_angle(line l){return vec.get_angle(l.vec);}
    //点到直线的距离
    db distance(point p){return fabs((p-st)^(p-ed))/(length());}
    //中点
    point midpoint(){return point((st.x+ed.x)/2,(st.y+ed.y)/2);}
    void print(){st.print();putchar('-');ed.print();}
    void println(){print();puts("");}
};
```

三角形

面积

$S_{\Delta} = \sqrt{p(p-a)(p-b)(p-c)} \qquad p = \frac{a+b+c}{2}$

外心

1. **三边垂直平分线交于外心**

2. $|OA| = |OB| = |OC|$

重心

1. **三边中线交于重心**

2. $\vec{OA} + \vec{OB} + \vec{OC} = \vec{0}$

垂心

1. **三边垂线交于垂心**

2. $\vec{OA} \cdot \vec{OB} = \vec{OB} \cdot \vec{OC} = \vec{OC} \cdot \vec{OA}$

内心

1. **三角角平分线交于内心**

2. $a \cdot \overrightarrow{OA} + b \cdot \overrightarrow{OB} + c \cdot \overrightarrow{OC} = \vec{0}$

```
//三角形
struct triangle
{
    point v[4];line e[4];
    triangle(point p1,point p2,point p3)
    {
        point p[4]={point(),p1,p2,p3};
        set(p);
    }
    triangle(point p[]) {set(p);}
    //构造逆时针三角形(注意特判三点共线)
    void set(point p[])
    {
        for(int i=1;i<=3;++i)v[i]=p[i];
        if(!line(v[1],v[2]).on_left(v[3]))
            swap(v[2],v[3]);
        for(int i=1;i<=2;++i)e[i].set(v[i],v[i+1]);
        e[3].set(v[3],v[1]);
    }
    bool inside(point p)
    {
        return e[1].on_left(p)==e[2].on_left(p)
            && e[1].on_left(p)==e[3].on_left(p);
    }
    //重心(中线交点)
    point gravity(){return (v[1]+v[2]+v[3])/3;}
    //外心
    point circum_center()
    {
        line l1=e[1].vertical_bisector();
        line l2=e[2].vertical_bisector();
        return l1.get_intersection(l2);
    }
    //内心
    point inscribed_center()
    {
        db a=e[2].length(),b=e[3].length(),c=e[1].length();
        db x=(a*v[1].x+b*v[2].x+c*v[3].x)/(a+b+c);
        db y=(a*v[1].y+b*v[2].y+c*v[3].y)/(a+b+c);
        return point(x,y);
    }
    //垂心
    point orthocenter()
    {
        line l1=e[1].vertical(v[3]);
        line l2=e[2].vertical(v[1]);
        return l1.get_intersection(l2);
    }
    //周长
    db perimeter()
    {
```

```
        db sum=0.0;
        for(int i=1;i<=3;++i)sum+=e[i].length();
        return sum;
    }
    //面积
    db area(){return fabs((v[3]-v[1])^(v[2]-v[1]))/2;}
    void print(){for(int i=1;i<=3;++i){v[i].print();if(i!=3)putchar('-');}}
    void println(){print();puts("");}
};
```

矩形

```
//矩形
struct rectangle
{
    point v[5];line e[5];
    rectangle(point p1,point p2)
    {
        point p[5]={point(),point(p1.x,p1.y),point(p1.x,p2.y),
                    point(p2.x,p1.y),point(p2.x,p2.y)};
        set(p);
    }
    rectangle(point p1,point p2,point p3,point p4)
    {
        point p[5]={point(),p1,p2,p3,p4};
        set(p);
    }
    rectangle(point p[]) {set(p);}
    //构造出顺时针或逆时针的矩形(注意特判四点共线)
    void set(point p[])
    {
        for(int i=1;i<=4;++i)v[i].set(p[i].x,p[i].y);
        if(!line(v[1],v[2]).parallel(line(v[3],v[4])))
            swap(v[2],v[3]);
        if(!line(v[1],v[4]).parallel(line(v[2],v[3])))
            swap(v[3],v[4]);
        for(int i=1;i<=3;++i)e[i].set(v[i],v[i+1]);
        e[4].set(v[4],v[1]);
    }
    //点在矩形内或矩形上
    bool inside(point p)
    {
        bool flag=true;
        for(int i=1;i<=4;++i)
        {
            if(e[i].on_segment(p))return true;
            if(e[i].on_left(p)!=e[1].on_left(p))
                flag=false;
        }
        return flag;
    }
    bool inside(line l){return inside(l.st) && inside(l.ed);}
```

```
    bool intersection(line l)
    {
        for(int i=1;i<=4;++i)
            if(e[i].segment_intersection(l))
                return true;
        return false;
    }
    void print(){for(int i=1;i<=4;++i){v[i].print();if(i!=4)putchar('-');}}
    void println(){print();puts("");}
};
```

圆

```
//圆
struct circle
{
    point c;db r;
    circle() {}
    circle(point p,db _r){set(p,_r);}
    void set(point p,db _r){c=p,r=_r;}
    //外接圆
    static circle circum_circle(triangle t)
    {
        point p=t.circum_center();
        return circle(p,line(p,t.v[1]).length());
    }
    //内接圆
    static circle inscribed_circle(triangle t)
    {
        point p=t.inscribed_center();
        return circle(p,t.area()*2/t.perimeter());
    }
    //点在圆内或圆上
    bool inside(point p){return line(p,c).len2<=r*r;}
    void print(){c.print(),printf(" r:%.2f",(double)r);}
    void println(){print();puts("");}
};
```

多边形

```
struct polygon
{
    const static int __=1e3+5;
    point v[__];int n;
    polygon() {}
    polygon(point p[],int n)
    {
        for(int i=1;i<=n;++i)v[i]=p[i];
        v[0]=p[n],v[n+1]=p[1];
        this->n=n;
```

```
    }
    db area()
    {
        db res=0.0;
        for(int i=1;i<=n;++i)
            res+=v[i]^v[i+1];
        return fabs(res)/2;
    }
};
```

二维凸包

```
struct ConvexHull
{
    const static int __=1e5+5;

    point ch[__];int n;

    static bool cmp(const point &x,const point &y)
    {
        if(x.x==y.x)
            return x.y<y.y;
        return x.x<y.x;
    }

    point& operator[](int x){return ch[x];}

    void add(point &p,int lim)
    {
        while(n>=lim && ((p-ch[n-1])^(ch[n]-ch[n-1]))>0)
            --n;
        ch[++n].set(p.x,p.y);
    }

    ConvexHull() {clear();}

    void get_ConvexHull(point p[],int np)
    {
        sort(p+1,p+1+np,cmp);
        n=0;
        for(int i=1;i<=np;++i)
            add(p[i],2);
        for(int i=np-1,j=n;i>=1;--i)
            add(p[i],j+1);
        --n;
    }

    //周长(若凸包为直线需要除以2)
    double circumference()
    {
        ch[n+1]=ch[1];
        double res=0.0;
```

```
        for(int i=1;i<=n;++i)
            res+=line(ch[i],ch[i+1]).length();
        return res;
    }

    void print()
    {
        for(int i=1;i<=n;++i)
            printf("%.2f %.2f\n",(double)ch[i].x,(double)ch[i].y);
    }

    void clear(){n=0;}
}C;
```

模拟退火

最小圆覆盖

```
const double eps=1e-3;
const double pi=acos(-1);

double random(void)
{
    return (rand()%10000+1)/10000.0;
}

struct node
{
    double x,y;
} a[1005];

int n;

double getmax(double x,double y)
{
    double maxx=0.0;
    for(int i=1; i<=n; i++)
    {
        double res=sqrt((x-a[i].x)*(x-a[i].x)+(y-a[i].y)*(y-a[i].y));
        if(res>maxx)maxx=res;
    }
    return maxx;
}

int main()
{
    srand(time(0));
    double max_x,max_y;
    while(~scanf("%lf%lf%d",&max_x,&max_y,&n))
    {
        memset(a,0,sizeof(a));
        for(int i=1; i<=n; i++)
            scanf("%lf%lf",&a[i].x,&a[i].y);
```

```
        double x=random()*max_x,y=random()*max_y;
        double step=sqrt(max_x*max_x+max_y*max_y)/2.0;
        double ans_x=x,ans_y=y,minn=1e18,t=50;
        double last_res=getmax(x,y);
        while(step>eps)
        {
            double nexx=x+cos(random()*2*pi)*step;
            double nexy=y+sin(random()*2*pi)*step;
            if(nexx<0 || nexx>max_x)continue;
            if(nexy<0 || nexy>max_y)continue;
            double now_res=getmax(nexx,nexy);
            double res=now_res-last_res;
            if(res<=0)
            {
                ans_x=x=nexx,ans_y=y=nexy;
                minn=last_res=now_res;
            }
            else if(exp(-res/t)>random())
            {
                x=nexx,y=nexy;
                last_res=now_res;
            }
            step*=0.9999,t*=0.97;
        }
        printf("(%.1f,%.1f).\n%.1f\n",ans_x,ans_y,minn);
    }
    return 0;
}
```

球

球体积交/并

```
ld pow2(ld x){return x*x;}
ld pow3(ld x){return x*x*x;}

ld cos(ld a,ld b,ld c){return (b*b+c*c-a*a)/(2*b*c);}

ld cap(ld r,ld h){return pi*(r*3-h)*h*h/3;}//球缺体积公式

//2球体积交
ld sphere_intersect(ld r1,ld r2,ld d)//两球半径与(球心距离的平方)
{
    //相离
    if(d>=pow2(r1+r2))return 0;
    //包含
    if(d<=pow2(r1-r2))return pow3(min(r1,r2))*4*pi/3;
    //相交
    ld h1=r1-r1*cos(r2,r1,sqrt(d)),h2=r2-r2*cos(r1,r2,sqrt(d));
    return cap(r1,h1)+cap(r2,h2);
}
```

```
//2球体积并
ld sphere_union(ld r1,ld r2,ld d)//两球半径与(球心距离的平方)
{
    //相离
    if(d>=pow2(r1+r2))return (pow3(r1)+pow3(r2))*4*pi/3;
    //包含
    if(d<=pow2(r1-r2))return pow3(max(r1,r2))*4*pi/3;
    //相交
    ld h1=r1+r1*cos(r2,r1,sqrt(d)),h2=r2+r2*cos(r1,r2,sqrt(d));
    return cap(r1,h1)+cap(r2,h2);
}
```

球冠

```
ld pow2(ld x){return x*x;}

ld cos(ld a,ld b,ld c){return (b*b+c*c-a*a)/(2*b*c);}

ld crown(ld r,ld h){return 2*pi*r*h;}//球冠体积公式

//2球表面积
ld sphere_union(ld r1,ld r2,ld d)//两球半径与(球心距离的平方)
{
    //相离
    if(d>=pow2(r1+r2))return (pow2(r1)+pow2(r2))*4*pi;
    //包含
    if(d<=pow2(r1-r2))return pow2(max(r1,r2))*4*pi;
    //相交
    ld h1=r1+r1*cos(r2,r1,sqrt(d)),h2=r2+r2*cos(r1,r2,sqrt(d));
    return crown(r1,h1)+crown(r2,h2);
}
```

其他

头文件

```
//💭💡🎈fold
#include<bits/stdc++.h>
#define fi first
#define sf scanf
#define se second
#define pf printf
#define pb push_back
#define mp make_pair
#define sz(x) ((int)(x).size())
#define all(x) (x).begin(),(x).end()
#define mem(x,y) memset((x),(y),sizeof(x))
#define fup(i,x,y) for(int i=(x);i<=(y);++i)
#define fdn(i,x,y) for(int i=(x);i>=(y);--i)
typedef long long ll;
typedef long double ld;
typedef unsigned long long ull;
typedef std::pair<int,int> pii;
using namespace std;
```

扩栈

```
register char *_sp __asm__("rsp");

int main()
{
    const int size=64*1024*256;
    static char *sys,*mine(new char[size]+size-4096);
    sys=_sp;
    _sp=mine;

//  solve();

    _sp=sys;
    return 0;
}
```

三分

```
for(int l=1,r=2e5;l<=r;)
{
    int len=(r-l)/3;
    ll x=cal(l+len),y=cal(r-len);
    if(x<y)//如果x==y: 判断期望答案更大还是更小
        r=r-len-1,ans=x;
    else
        l=l+len+1,ans=y;
}
```

读入外挂

cpp

```
template<class T>
inline void read(T &x)
{
    x=0;
    int 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();
    }
    x*=f;
}

void print(ll x)
{
    if(x>9)print(x/10);
    putchar(x%10+'0');
}
```

Java

```
static class InputReader
{
    public BufferedReader reader;
    public StringTokenizer tokenizer;

    public InputReader(InputStream stream)
    {
```

```
        reader=new BufferedReader(new InputStreamReader(stream),32768);
        tokenizer=null;
    }

    public String next()
    {
        while(tokenizer==null || !tokenizer.hasMoreTokens())
        {
            try{tokenizer=new StringTokenizer(reader.readLine());}
            catch(IOException e){throw new RuntimeException(e);}
        }
        return tokenizer.nextToken();
    }

    public int nextInt(){return Integer.parseInt(next());}

    public long nextLong(){return Long.parseLong(next());}

    public double nextDouble(){return Double.parseDouble(next());}
}
```

离散化

```
struct Discretization
{
    const static int __=1e5+5;

    ll a[__],b[__];int idx;

    Discretization() {clear();}

    ll operator[](int x){return b[x];}

    void push_back(ll x){a[++idx]=x;}

    void build()
    {
        sort(a+1,a+1+idx);
        idx=unique(a+1,a+1+idx)-a-1;
    }

    int get(ll x)
    {
        int y=lower_bound(a+1,a+1+idx,x)-a;
        b[y]=x;
        return y;
    }

    void clear() {idx=0;}
}D;
```

堆

```
template<class T>
struct heap
{
    T hp[1000005];
    int idx;
    heap() {clear();}
    void push(T x)
    {
        hp[++idx]=x;
        int t=idx;
        while(t!=1 && hp[t]<hp[t>>1])
            swap(hp[t>>1],hp[t]),t>>=1;
    }
    void pop()
    {
        hp[1]=hp[idx--];
        int t=1,y=1;
        while(1)
        {
            if((t<<1)<=idx && hp[t<<1]<hp[y])
                y=t<<1;
            if((t<<1|1)<=idx && hp[t<<1|1]<hp[y])
                y=t<<1|1;
            if(t==y)return;
            swap(hp[t],hp[y]),t=y;
        }
    }
    T top()
    {
        return hp[1];
    }
    void clear() {idx=0;}
};
```

莫队算法

```
namespace mo
{
    int n,q,blo;
    ll a[__],ans[__],res;
    struct query{int l,r,id;}qj[__];

    void sfd(){fup(i,1,n)scanf("%lld",a+i);}
    void sfq(){fup(i,1,q)scanf("%d%d",&qj[i].l,&qj[i].r),qj[i].id=i;}
    void pf(){fup(i,1,q)printf("%lld\n",ans[i]);}
    void add(int x)
    {
    }
```

```
    void cut(int x)
    {
    }
    bool cmp(const query& x,const query& y)
    {
        if(x.l/blo==y.l/blo)return x.r<y.r;
        return x.l<y.l;
    }
    void solve()
    {
        blo=sqrt(n+0.1);
        sort(qj+1,qj+q+1,cmp);
        int l=1,r=0;res=0;
        fup(i,1,q)
        {
            while(r<qj[i].r)++r,add(r);
            while(l>qj[i].l)--l,add(l);
            while(r>qj[i].r)cut(r),--r;
            while(l<qj[i].l)cut(l),++l;
            ans[qj[i].id]=res;
        }
    }
};
```

带修改莫队

单点修改/询问区间不同数字个数

```
const int __=10005;
int a[__],b[__],n,qdx=0,mdx=0,bsz;
int ans[__],times[1000005];

struct query
{
    int l,r,md,id;
    void set(int _l,int _r,int _md,int _id)
    {
        l=_l,r=_r,md=_md,id=_id;
    }
    bool operator<(const query &b)const
    {
        if(l/bsz!=b.l/bsz)return l<b.l;
        if(r/bsz!=b.r/bsz)return r<b.r;
        return id<b.id;
    }
}que[__];

struct modfiy
{
    int wz,x,y;
    void set(int _wz,int _x,int _y)
    {
        wz=_wz,x=_x,y=_y;
```

```
    }
}mod[__];

int res=0;

void add(int x)
{
    if(!times[x])++res;
    ++times[x];
}

void cut(int x)
{
    if(times[x]==1)--res;
    --times[x];
}

void upd(int l,int r,int t)
{
    if(l<=mod[t].wz && mod[t].wz<=r)
        cut(mod[t].x),add(mod[t].y);
    a[mod[t].wz]=mod[t].y;
}

void del(int l,int r,int t)
{
    if(l<=mod[t].wz && mod[t].wz<=r)
        add(mod[t].x),cut(mod[t].y);
    a[mod[t].wz]=mod[t].x;
}

void work()
{
    bsz=(int)pow(n,2.0/3);
    sort(que+1,que+1+qdx);
    int l=1,r=0,t=0;
    fup(i,1,qdx)
    {
        while(t<que[i].md)++t,upd(l,r,t);
        while(t>que[i].md)del(l,r,t),--t;
        while(l<que[i].l)cut(a[l]),++l;
        while(l>que[i].l)--l,add(a[l]);
        while(r<que[i].r)++r,add(a[r]);
        while(r>que[i].r)cut(a[r]),--r;
        ans[que[i].id]=res;
    }
}

int main()
{
    int q;sf("%d%d",&n,&q);
    fup(i,1,n)sf("%d",&a[i]),b[i]=a[i];
    fup(i,1,q)
```

```
    {
        char op[2];int l,r;
        sf("%s%d%d",op,&l,&r);
        if(op[0]=='Q')
            ++qdx,que[qdx].set(l,r,mdx,qdx);
        if(op[0]=='R')
            mod[++mdx].set(l,0,r);
    }
    fup(i,1,mdx)
    {
        mod[i].x=b[mod[i].wz];
        b[mod[i].wz]=mod[i].y;
    }
    work();
    fup(i,1,qdx)
        pf("%d\n",ans[i]);
    return 0;
}
```

回滚莫队

询问区间$\max(v \times v$**在区间出现次数)**

```
//离散化模板
const int __=1e5+5;
int a[__],times[__];
int blo;
ll ans[__];

struct query
{
    int l,r,id;
    int lb,rb;

    void scan(int _id)
    {
        sf("%d%d",&l,&r);
        lb=(l-1)/blo+1;
        rb=(r-1)/blo+1;
        id=_id;
    }

    bool operator<(const query& b)const
    {
        if(lb==b.lb)return r<b.r;
        return l<b.l;
    }

    ll cal()
    {
        ll res=0;
        for(int i=l;i<=r;++i)
        {
```

```
            int t=++times[a[i]];
            res=max(res,1ll*t*D[a[i]]);
        }
        for(int i=l;i<=r;++i)
            --times[a[i]];
        return res;
    }
}que[__];

int main()
{
    int n,q;sf("%d%d",&n,&q);
    blo=sqrt(n+0.1);
    fup(i,1,n)
    {
        sf("%d",&a[i]);
        D.pb(a[i]);
    }
    D.build();
    fup(i,1,n)a[i]=D.get(a[i]);
    int idx=0;
    fup(i,1,q)
    {
        que[++idx].scan(i);
        if(que[idx].lb==que[idx].rb)
        {
            ans[i]=que[idx].cal();
            --idx;
        }
    }
    sort(que+1,que+1+idx);
    int r=0;
    ll rres=0;
    fup(i,1,idx)
    {
        if(que[i].lb!=que[i-1].lb)
        {
            for(;r>que[i-1].lb*blo;--r)
                --times[a[r]];
            rres=0,r=que[i].lb*blo;
        }
        for(;r<que[i].r;)
        {
            int t=++times[a[++r]];
            rres=max(rres,1ll*t*D[a[r]]);
        }
        ll res=rres;
        for(int l=que[i].lb*blo;l>=que[i].l;--l)
        {
            int t=++times[a[l]];
            res=max(res,1ll*t*D[a[l]]);
        }
        for(int l=que[i].lb*blo;l>=que[i].l;--l)
```

```
            --times[a[l]];
        ans[que[i].id]=res;
    }
    fup(i,1,q)pf("%lld\n",ans[i]);
    return 0;
}
```

CDQ分治

子矩阵加/查询子矩阵和

```
//一阶差分树状数组模板，并定义B[2]变量

const int __=800005;

//ans[q.id]+=q.ans*q.mul
struct query
{
    int id,mul,x,y;ll ans;

    bool operator<=(const query &b)const
    {
        return x<=b.x;
    }
}q[__],t[__];

ll ans[__];

void cdq(int l,int r)
{
    if(l==r)return;
    int mid=(l+r)>>1;
    cdq(l,mid),cdq(mid+1,r);
    int x=l,y=mid+1,z=l;
    while(x!=mid+1 || y!=r+1)
    {
        if(y==r+1 || (x!=mid+1 && q[x]<=q[y]))
        {
            if(!q[x].mul)
            {
                B[0].add(q[x].y,q[x].id);
                B[1].add(q[x].y,q[x].id*q[x].x);
            }
            t[z++]=q[x++];
        }
        else
        {
            if(q[y].mul)
            {
                ll b[2]={B[0].sum(q[y].y),B[1].sum(q[y].y)};
                q[y].ans+=q[y].mul*((q[y].x+1)*b[0]-b[1]);
                ans[q[y].id]+=q[y].mul*((q[y].x+1)*b[0]-b[1]);;
            }
```

```
            t[z++]=q[y++];
        }
    }
    fup(i,l,mid)
        if(!q[i].mul)
        {
            B[0].add(q[i].y,-q[i].id);
            B[1].add(q[i].y,-q[i].id*q[i].x);
        }
    fup(i,l,r)q[i]=t[i];
}

int main()
{
    int n,m;sf("%*s%d%d",&n,&m);
    B[0].n=m,B[1].n=m;
    char op[2];int x1,y1,x2,y2,idx=0,qdx=0;
    while(~sf("%s%d%d%d%d",op,&x1,&y1,&x2,&y2))
    {
        if(op[0]=='L')//子矩阵加
        {
            int v;sf("%d",&v);
            ++x2,++y2;
            ++idx;q[idx]={v,0,x1,y1};
            ++idx;q[idx]={v,0,x2,y2};
            ++idx;q[idx]={-v,0,x1,y2};
            ++idx;q[idx]={-v,0,x2,y1};
        }
        if(op[0]=='k')//查询子矩阵和
        {
            --x1,--y1,ans[++qdx]=0;
            ++idx;q[idx]={qdx,1,x1,y1};
            ++idx;q[idx]={qdx,1,x2,y2};
            ++idx;q[idx]={qdx,-1,x1,y2};
            ++idx;q[idx]={qdx,-1,x2,y1};
        }
    }
    cdq(1,idx);
    fup(i,1,qdx)
        pf("%lld\n",ans[i]);
    return 0;
}
```

整体二分

区间第k小/单点修改

```
//树状数组定义B;
int a[__],ans[__];

struct query
{
    int id,l,r,v;
```

```
}que[__],t[__];//数组个数+操作个数*2

void BinarySearch(int vl,int vr,int ql,int qr)
{
    if(ql>qr)return;
    if(vl==vr)
    {
        for(int i=ql;i<=qr;++i)
            if(que[i].id>0)
                ans[que[i].id]=vl;
        return;
    }
    int vm=(vl+vr)>>1,x=ql-1,y=0;
    for(int i=ql;i<=qr;++i)
    {
        if(que[i].id<0)//插入操作
            if(que[i].v<=vm)//向左划分
            {
                B.add(que[i].l,que[i].id+2);
                que[++x]=que[i];
            }
            else t[++y]=que[i];//向右划分
        else
        {   //查询操作
            int sum=B.sum(que[i].r)-B.sum(que[i].l-1);
            if(que[i].v>sum)//向右划分
            {
                que[i].v-=sum;
                t[++y]=que[i];
            }
            else que[++x]=que[i];//向左划分
        }
    }
    //树状数组清零
    for(int i=ql;i<=x;++i)
        if(que[i].id<0)
            B.add(que[i].l,-que[i].id-2);
    for(int i=1,j=x+1;i<=y;++i,++j)
        que[j]=t[i];
    BinarySearch(vl,vm,ql,x);
    BinarySearch(vm+1,vr,x+1,qr);
}

int main()
{
    int _;for(sf("%d",&_);_;--_)
    {
        int n,q;sf("%d%d",&n,&q);
        int idx=0,qdx=0,minn=1e9+7,maxx=-minn;
        for(int i=1;i<=n;++i)
        {
            sf("%d",&a[i]);
            minn=min(minn,a[i]);
```

```
            maxx=max(maxx,a[i]);
            que[++idx]={-1,i,0,a[i]};//insert(l,v);
        }
        for(int i=1;i<=q;++i)
        {
            char op[2];sf("%s",op);
            if(op[0]=='Q')//查询区间[l,r]第v小数
            {
                int l,r,v;sf("%d%d%d",&l,&r,&v);
                minn=min(minn,v);
                maxx=max(maxx,v);
                que[++idx]={++qdx,l,r,v};
            }
            else//a[x]=v;
            {
                int x,v;sf("%d%d",&x,&v);
                que[++idx]={-3,x,0,a[x]};//delete(l,a[x]);
                que[++idx]={-1,x,0,v};//insert(l,v);
                a[x]=v;
            }
        }
        B.build(n);
        BinarySearch(minn,maxx,1,idx);
        fup(i,1,qdx)
            pf("%d\n",ans[i]);
    }
    return 0;
}
```

表达式求值

```
typedef double type;

int idx;
stack<type>Sn;
type num[__];

struct Operator
{
    //定义符号
    static bool is(char c)
    {
        return c=='+' || c=='-' || c=='*'
            || c=='/' || c=='(' || c==')';
    }

    char o;
    Operator() {}

    void operator=(const char c){o=c;}

    bool operator==(const char c){return o==c;}
```

```
    //定义符号优先级
    bool operator<=(const Operator &b)const
    {
        if(o=='*' || o=='/')
            return b.o=='*' || b.o=='/';
        if(o=='+' || o=='-')
            return b.o!='(';
        if(o==')')return true;
        return false;
    }

    //定义符号运算
    type fun()
    {
        type x=Sn.top();Sn.pop();
        type y=Sn.top();Sn.pop();
        if(o=='+')return y+x;
        if(o=='-')return y-x;
        if(o=='*')return y*x;
        if(o=='/')return y/x;
    }
}op[__];

stack<Operator>So;

//读浮点数
int read_num(int x)
{
    ll fz=0,fm=0;
    for(;;++x)
    {
        if(fm)fm*=10;
        if(a[x]=='.')fm=1;
        else fz=fz*10+(a[x]-'0');
        if(!a[x+1] || Operator::is(a[x+1]))
            break;
    }
    if(!fm)fm=1;
    num[++idx]=fz*1.0/fm;
    return x;
}

void compare(Operator c)
{
    while(!So.empty() && c<=So.top())
    {
        if(So.top()=='('){So.pop();return;}
        Sn.push(So.top().fun()),So.pop();
    }
    So.push(c);
}
```

```
type calculate()
{
    for(int i=1;i<=idx;++i)
        if(op[i]==0)Sn.push(num[i]);
        else compare(op[i]);
    for(;!So.empty();So.pop())
        Sn.push(So.top().fun());
    type res=Sn.top();Sn.pop();
    return res;
}

int main()
{
    sf("%s",a+1);
    int n=strlen(a+1);
    for(int i=1;i<=n;++i)
    {
        if(Operator::is(a[i]))
            op[++idx]=a[i];
        else i=read_num(i),op[idx]=0;
    }
    pf("%.2f\n",calculate());
    return 0;
}
```

Dancing Links

精准覆盖: 从01矩阵中选择一些行, 使得每列有且仅有一个数字1

```
struct node
{
    int x,y;
    int l,r,u,d;
    node(int x=0,int y=0,int l=0,
         int r=0,int u=0,int d=0):
        x(x),y(y),l(l),r(r),u(u),d(d) {}
} dl[10105];

int a[105][105],id[105][105];
int idx,one[105];

void build(int n,int m)
{
    dl[0]=node(0),idx=m;
    for(int j=1; j<=m; j++)
    {
        dl[j]=node(0,j,j-1,dl[j-1].r,j,j);
        dl[dl[j-1].r].l=j,dl[j-1].r=j;
        int pre=j;
        for(int i=1; i<=n; i++)
            if(a[i][j]==1)
            {
                int x=(id[i][j]=++idx);
```

```
                dl[idx]=node(i,j,idx,idx);
                dl[x].d=dl[pre].d,dl[x].u=pre;
                dl[dl[pre].d].u=x,dl[pre].d=x;
                pre=x,one[j]++;
            }
    }
    for(int i=1; i<=n; i++)
    {
        int pre=0;
        for(int j=1; j<=m; j++)
            if(a[i][j]==1)
                if(!pre)pre=id[i][j];
                else
                {
                    int x=id[i][j];
                    dl[x].r=dl[pre].r,dl[x].l=pre;
                    dl[dl[pre].r].l=x,dl[pre].r=x;
                    pre=x;
                }
    }
}

void change(int x,bool op)
{
    dl[dl[x].r].l=op?x:dl[x].l;
    dl[dl[x].l].r=op?x:dl[x].r;
    for(int i=dl[x].d;i!=x;i=dl[i].d)
        for(int j=dl[i].r;j!=i;j=dl[j].r)
        {
            dl[dl[j].d].u=op?j:dl[j].u;
            dl[dl[j].u].d=op?j:dl[j].d;
            if(op)one[dl[j].y]++;
            else one[dl[j].y]--;
        }
}

int ans[105];

bool dancing(int depth)
{
    int x=dl[0].r;
    if(x==0)return true;
    if(dl[x].d==x)return false;
    for(int i=x;i;i=dl[i].r)
        if(one[i]<one[x])x=i;
    change(dl[x].y,0);
    for(int i=dl[x].d;i!=x;i=dl[i].d)
    {
        ans[depth]=dl[i].x;
        for(int j=dl[i].r;j!=i;j=dl[j].r)
            change(dl[j].y,0);
        if(dancing(depth+1))return true;
        for(int j=dl[i].l;j!=i;j=dl[j].l)
```

```
            change(dl[j].y,1);
        }
        change(dl[x].y,1);
        return false;
    }

void init(void)
{
    memset(id,0,sizeof(id));
    memset(dl,0,sizeof(dl));
    memset(one,0,sizeof(one));
}

int main()
{
    int T;
    scanf("%d",&T);
    while(T--)
    {
        init();
        int n,m;
        scanf("%d%d",&n,&m);
        for(int i=1; i<=n; i++)
            for(int j=1; j<=m; j++)
                scanf("%d",&a[i][j]);
        build(n,m);
        if(dancing(1))
            printf("Yes\n");
        else printf("No\n");
    }
    return 0;
}
```

Berlekamp-Massey

```
ll pv[105]={0,4,12,33,88,232,609,1596}; //改前几项，前面用一个0占位
const int num=7;                        //前几项个数
double fn[num+1],wa[num+1];

int bm(void)
{
    int f=1,w=1,last=1;
    fn[1]=0,wa[1]=1.0/pv[1];
    for(int i=2; i<=num; i++)
    {
        double sum=-pv[i];
        for(int j=1; j<=f;++j)
            sum+=fn[j]*pv[i-j];
        if(fabs(sum)<eps)continue;
        double t[num+1]={0,-1/sum};
        for(int j=1;j<=f;++j)
            t[j+1]=fn[j]/sum;
```

```
        for(int j=1;j<=w;++j)
            fn[i-last+j-1]+=-sum*wa[j];
        memcpy(wa,t,sizeof(wa));
        last=i,++f,++w;

    }
    while(fabs(fn[f])<eps)f--;
    return f;
}
```

STL in Java

ArrayList(vector)

```
//vector<int>G;
ArrayList<Integer>G=new ArrayList<Integer>();
//G[i];
G.get(i);
//G.push_back(x);
G.add(x);
```

Queue(queue)

```
//queue<int>Q;
Queue<Integer>Q=new LinkedList<Integer>();
//Q.push(x);
Q.add(x);
//int t=Q.front();Q.pop();
int t=Q.remove();
//Q.empty()
Q.peek()!=null
```

TreeMap(map)

```
//map<int,int>vis;
TreeMap<Integer,Integer>map=new TreeMap<Integer,Integer>();
//int x=vis[y];
Integer x=map.get(y);
//vis[x]=y;
map.put(x,y);
//auto x=vis.lower_bound(y);
Integer x=map.ceilingKey(A[i]);
```