prime detection

miller-rabin

```
ll qmul(ll a,ll b,ll mod)//快速乘
{
    ll c=(long double)a/mod*b;
    ll res=(unsigned long long)a*b-(unsigned long long)c*mod;
    return (res+mod)%mod;
}
ll qpow(ll a,ll n,ll mod)//快速幂
{
    ll res=1;
    while(n)
    {
        if(n&1) res=qmul(res,a,mod);
        a=qmul(a,a,mod);
        n>>=1;
    }
    return res;
}
bool millerrabin(ll n)//Miller Rabin Test
{
    if(n<3||n%2==0) return n==2;//特判
    ll u=n-1,t=0;
    while(u%2==0) u/=2,++t;
    ll ud[]={2,325,9375,28178,450775,9780504,1795265022};
    for(ll a:ud)
    {
        ll v=qpow(a,u,n);
        if(v==1||v==n-1||v==0) continue;
        for(int j=1;j<=t;j++)
        {
            v=qmul(v,v,n);
            if(v==n-1&&j!=t){v=1;break;}//出现一个n-1，后面都是1，直接跳出
            if(v==1) return 0;//这里代表前面没有出现n-1这个解，二次检验失败
        }
        if(v!=1) return 0;//Fermat检验
    }
    return 1;
}
```

1e18 fac decompose

```
using ll = long long;
ll max_factor, n;
std::map<ll, int> MP;

ll gcd(ll a, ll b) {
    while (b) {
        ll t = a % b;
        a = b;
        b = t;
    }
    return a;
}

ll qpow(ll x, ll p, ll mod) {
    ll ans = 1;
    x%=mod;
    while (p) {
        if (p & 1) ans = ans * x % mod;
        x = x * x % mod;
        p >>= 1;
    }
    return ans;
}

bool Miller_Rabin(ll p) {  // 判断素数
    if (p < 2) return false;
    if (p == 2 || p == 3) return true;
    ll d = p - 1, r = 0;
    while (!(d & 1)) ++r, d >>= 1;  // 将d处理为奇数
    for (ll k = 0; k < 10; ++k) {
        ll a = rand() % (p - 2) + 2;
        ll x = qpow(a, d, p);
        if (x == 1 || x == p - 1) continue;
        for (int i = 0; i < r - 1; ++i) {
            x = (__int128) x * x % p;
            if (x == p - 1) break;
        }
        if (x != p - 1) return false;
    }
    return true;
}

ll Pollard_Rho(ll x) {
    ll s = 0, t = 0;
    ll c = (ll) rand() % (x - 1) + 1;
    int step = 0, goal = 1;
    ll val = 1;
    for (goal = 1;; goal *= 2, s = t, val = 1) {  // 倍增优化
        for (step = 1; step <= goal; ++step) {
            t = ((__int128) t * t + c) % x;
            val = (__int128) val * abs(t - s) % x;
            if ((step % 127) == 0) {
                ll d = gcd(val, x);
                if (d > 1) return d;
            }
        }
        ll d = gcd(val, x);
        if (d > 1) return d;
    }
}

void fac(ll x) {
    if (x < 2) return;
    if (Miller_Rabin(x)) {              // 如果x为质数
        max_factor = std::max(max_factor, x);  // 更新答案
        MP[x]++;
        fac(n /= x);
        return;
    }
    ll p = x;
    while (p >= x) p = Pollard_Rho(x);  // 使用该算法
    while ((x % p) == 0) x /= p;
    //fac(x);
    fac(p);  // 继续向下分解x和p
}

ll cul(ll x, ll y, ll k, ll mod) {
    ll ans = 1;

    ans = qpow(x%mod, (y+ 1), mod) ;
    ll ans2 = qpow(ans, k, mod);
    ans = (ans2 - 1 + mod)%mod;
    int inv = (qpow(x%mod, k, mod) - 1 + mod) % mod;
    inv = qpow(inv, mod - 2, mod);

    return ans * inv % mod;
}
int main()
{
    fac(....)
}
```

segtree

维护区间最值为例：

```
struct T
{
    int l,r,mid;
    int lazy,sum;
}tree[N<<2];
void build(int rt,int l,int r)
{
    tree[rt].lazy=0;
    tree[rt].l=l;
    tree[rt].r=r;
    if(l==r)
    {
        tree[rt].sum=a[l];//依据初始化类型具体而定，置0可以看为空树，但是l,r均初始化。
        return ;
    }
    int mid=tree[rt].mid=l+r>>1;
    build(rt<<1,l,mid);
    build(rt<<1|1,mid+1,r);
}
void push_down(int rt)
{
    if(tree[rt].lazy)
    {
        tree[rt<<1].sum+=tree[rt].lazy;
        tree[rt<<1].lazy+=tree[rt].lazy;
        tree[rt<<1|1].sum+=tree[rt].lazy;
        tree[rt<<1|1].lazy+=tree[rt].lazy;
        tree[rt].lazy=0;
    }
}
void push_up(int rt){
    tree[rt].sum = max(tree[rt<<1].sum,tree[rt<<1|1].sum);//看情况而定
}
void update(int rt,int l,int r,int v)
{
    if(tree[rt].r<l||tree[rt].l>r) return ;
    if(tree[rt].l>=l&&tree[rt].r<=r)
    {
        tree[rt].sum+=v;
        tree[rt].lazy+=v;
        return ;
    }
    push_down(rt);
    if(tree[rt].mid>=l) update(rt<<1,l,r,v);
    if(tree[rt].mid<r) update(rt<<1|1,l,r,v);
    push_up(rt);
}
int query(int rt,int l,int r)
{
    if(tree[rt].r<l||tree[rt].l>r) return 0;
    if(tree[rt].l>=l&&tree[rt].r<=r) return tree[rt].sum;
    push_down(rt);
    if(tree[rt].mid<l) return query(rt<<1|1,l,r);
    if(tree[rt].mid>=r) return query(rt<<1,l,r);
    return max(query(rt<<1|1,l,r), query(rt<<1,l,r));
}
```

BIT (Fenwick)

template from jiangly 注意 add 当下， sum （x) = x-1的sum 有超时风险，因为vector

```
template <class T>
struct Fenwick {
    int n;
    vector<T> a;

    Fenwick(int n = 0) {
        this->n = n;
        a.assign(n, T());
    }

    void add(int x, T v) {
        for (int i = x + 1; i <= n; i += i & -i) {
            a[i - 1] += v;
        }
    }
    T sum(int x) {
        auto ans = T();
        for (int i = x; i > 0; i -= i & -i) {
            ans += a[i - 1];
        }
        return ans;
    }

    T rangeSum(int l, int r) {
        return sum(r) - sum(l);
    }

};
```

exgcd

求解：ax+by = gcd(a,b)

```
ll exgcd(ll a,ll b,ll &x,ll &y)
{
    if(!b)
    {
        x=1;y=0;
        return a;
    }
    else
    {
        ll tx,ty;
        ll d=exgcd(b,a%b,tx,ty);
        x=ty;y=tx-(a/b)*ty;
        return d;
    }
}
```

DSU

```
struct DSU {
    vector<int> f, siz;

    DSU() {}
    DSU(int n) {
        init(n);
    }

    void init(int n) {
        f.resize(n);
        iota(f.begin(), f.end(), 0);
        siz.assign(n, 1);
    }

    int find(int x) {
        while (x != f[x]) {
            x = f[x] = f[f[x]];
        }
        return x;
    }

    bool same(int x, int y) {
        return find(x) == find(y);
    }

    bool merge(int x, int y) {
        x = find(x);
        y = find(y);
        if (x == y) {
            return false;
        }
        siz[x] += siz[y];
        f[y] = x;
        return true;
    }

    int size(int x) {
        return siz[find(x)];
    }
};
```

tarjan

使用tarjan算法求强连通分量。

有向图（对于无向图还需要加一个int fa(若需要求环))

```
vector<int>adj[N];
int dfn[N],low[N],stk[N],top,dn,vis[N];
int nsz,nl[N];
void tarjan(int u)
{
    vis[u] = 1;
    dfn[u] = low[u] = ++dn;
    stk[++top] = u;
    for(auto v:adj[u]){
        if(!dfn[v]){
            tarjan(v);
            low[u] = min(low[u],low[v]);
        }
        else if(vis[v]){
            low[u] = min(low[u],dfn[v]);
        }
    }
    if(low[u]==dfn[u]){//强连通分量
        nl[u] = ++nsz;
        while(stk[top]!=u)nl[stk[top]] = nl[u],vis[stk[top]] = 0,top--;
        vis[u] = 0,top--;
    }
}
```