基础

个人常用头文件与宏定义

```cpp
#include <bits/stdc++.h>

using namespace std;
#define rep(i,a,n) for(int i=a;i<=n;++i)
#define per(i,n,a) for(int i=n;i>=a;--i)
#define pb push_back
#define fi first
#define endl '\n'
#define se second
#define debug(a) cout<<#a<<"="<<a<<endl
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
typedef long long ll;
typedef pair<ll,ll> PII;
const int inf = 0x3f3f3f3f;
const ll mod = 1e9+7;
ll gcd(ll a,ll b){return b?gcd(b,a%b):a;}
ll lcm(ll x,ll y){return x*y/gcd(x,y);}
ll qmi(ll a,ll k,ll p){ll res=1%p;a%=p;while(k)
{if(k&1)res=res*a%p;k>>=1;a=a*a%p;}return res;}

int main()
{
    IOS;

    return 0;
}
```

常用质数

```
113 151 211 281 379 509 683 911 1217 1627 2017 2179 2909 3881 6907 9209 12281
16381 21841 23333 29123 38833 51787 69061 92083 122777 163729 218357 291143
388211 517619 690163 999983 1145141 1226959 1635947 2181271 2908361 3877817
5170427 6893911 9191891 9999991 12255871 16341163 19260817 19491001 21788233
29050993 38734667 51646229 68861641 91815541 122420729 163227661 217636919
290182597 386910137 666623333 469762049 515880193 687840301 917120411 998244353
1000000007 1000000009 1004535809 1222827239 1610612741 2147483647 3221225473
4294967291 100000000000031
```

Z类

```cpp
constexpr int P = 1e9+9;

int norm(int x)
{
    if(x < 0) x += P;
    if(x >= P) x -= P;
    return x;
```

```
}

template<class T>
T power(T a,ll k)
{
    T res = 1;
    while(k){
        if(k&1) res = res*a;
        k >>= 1;
        a = a*a;
    }
    return res;
}

struct Z {
    int x;
    Z(int x = 0) : x(norm(x)) {}
    Z(ll x) : x(norm(x % P)) {}
    int val() const {
        return x;
    }
    Z operator-() const {
        return Z(norm(P - x));
    }
    Z inv() const {
        assert(x != 0);
        return power(*this, P - 2);
    }
    Z &operator*=(const Z &rhs) {
        x = ll(x) * rhs.x % P;
        return *this;
    }
    Z &operator+=(const Z &rhs) {
        x = norm(x + rhs.x);
        return *this;
    }
    Z &operator-=(const Z &rhs) {
        x = norm(x - rhs.x);
        return *this;
    }
    Z &operator/=(const Z &rhs) {
        return *this *= rhs.inv();
    }
    friend Z operator*(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res *= rhs;
        return res;
    }
    friend Z operator+(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res += rhs;
        return res;
    }
    friend Z operator-(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res -= rhs;
        return res;
    }
    friend Z operator/(const Z &lhs, const Z &rhs) {
        Z res = lhs;
        res /= rhs;
        return res;
    }
    friend std::istream &operator>>(std::istream &is, Z &a) {
        ll v;
        is >> v;
        a = Z(v);
        return is;
    }
    friend std::ostream &operator<<(std::ostream &os, const Z &a) {
        return os << a.val();
    }
};
```

算法基础

快速排序

```
void quick_sort(int l,int r)
{
    if(l>=r) return ;
    int i=l-1,j=r+1,x=q[l+r>>1];
    while(i<j){
        do i++; while(q[i]<x);
        do j--; while(q[j]>x);
        if(i<j) swap(q[i],q[j]);
    }
    quick_sort(l,j);
    quick_sort(j+1,r);
}
```

拓展求第k小数：时间复杂度 $O(n)$

判断第k小的数在前半部分还是在后半部分 只递归一边即可

```
int quick_sort(int l,int r,int k)
{
    if(l==r) return q[l];

    int i=l-1,j=r+1,x=q[l+r>>1];
    while(i<j){
        do i++; while(q[i]<x);
        do j--; while(q[j]>x);
        if(i<j) swap(q[i],q[j]);
    }
    int sl = j-l+1;
    if(k<=sl) return quick_sort(l,j,k);
    return quick_sort(j+1,r,k-sl);
}
```

归并排序

```
void merge_sort(int l,int r)
{
    if(l >= r) return ;
    int mid = l+r>>1;
    merge_sort(l,mid);
    merge_sort(mid+1,r);
    int k=0, i=l,j=mid + 1;
    while(i<=mid && j<=r){
        if(q[i]<q[j]) temp[k++] = q[i++];
        else temp[k++] = q[j++];
    }
    while(i<=mid) temp[k++] = q[i++];
    while(j<=r) temp[k++] = q[j++];

    for(i=l,j=0;i<=r;i++,j++)
        q[i] = temp[j];
}
```

高精度

数组的低位代表最后一位 因为不好修改和增加 数组的最后一位代表数的高位

高精度加法

```
vector<int> add(vector<int> &a,vector<int> &b)
{
    vector<int> c;
    int t = 0;
    for(int i=0;i<a.size()||i<b.size();i++){
        if(i<a.size()) t += a[i];
        if(i<b.size()) t += b[i];
        c.push_back(t%10);
        t /= 10;
    }
    if(t) c.push_back(1);
    return c;
}
```

高精度减法

```
//需要事先判断大小，模板只适用于A>=B这种情况

bool com(string a,string b)
{
    if(a.size() != b.size()) return a.size()<b.size();
    else{
        for(int i=0;a[i];i++){
            if(a[i] != b[i]) return a[i]<b[i];
        }
    }
    return false;
}
```

```
vector<int> sup(vector<int> &a,vector<int> &b)
{
    vector<int> c;
    int t=0;
    for(int i=0;i<a.size();i++){
        t = a[i] - t;
        if(i<b.size()) t -= b[i];
        if(t>=0){
            c.push_back(t%10);
            t = 0;
        }
        else{
            c.push_back((t+10)%10);
            t = 1;
        }
    }
    while(c.size()>1 && c.back() == 0)
        c.pop_back();
    return c;
}
```

高精度乘法

第一种情况 大乘小

```
vector<int> mul(vector<int> a,int b)
{
    vector<int> c;
    int t = 0;
    for(int i=0;i < a.size();i++){
        t += a[i] * b;
        c.push_back(t%10);
        t /= 10;
    }
    while(t){
        C.push_back(t % 10);
        t /= 10;
    }
    while (c.size() > 1 && c.back() == 0)
        c.pop_back();

    return c;
}
```

第二种情况 大乘大

```
void mul(vector<int> &a,vector<int> &b)
{
    for(int i=0;i<a.size();i++){
        for(int j=0;j<b.size();j++){
            c[i+j] += a[i]*b[j];
            c[i+j+1] += c[i+j]/10;
            c[i+j] %= 10;
        }
    }
    len = a.size()+b.size();
    while(len>0 && c[len] == 0)
        len--;
}

vector<int> mul(vector<int> a, vector<int> b) {
    vector<int> c(a.size()+b.size()+10,0);
    for(int i=0;i<a.size();i++){
        for(int j=0;j<b.size();j++)
            c[i+j] += a[i]*b[j];
    }
    for(int i=0;i<c.size()-1;i++){
        c[i+1] += c[i]/10;
        c[i] %= 10;
    }
    while (c.size() > 1 && c.back() == 0)
        c.pop_back();
    return c;
}
```

高精度除法

```
//r为余数
vector<int> div(vector<int> &a,int b,int &r)
{
    vector<int> c;
    r = 0;
    for(int i=a.size()-1;i>=0;i--){
        r = r * 10 + a[i];
        c.push_back(r / b);
        r = r % b;
    }
    reverse(c.begin(),c.end());
    while(c.size()>1&&c.back()==0)
        c.pop_back();
    return c;
}
```

位运算

相关函数:

```
__builtin_popcount(x)   //二进制中1的个数
__builtin_popcountll(x)     //long long数据类型中 二进制中1的个数
__builtin_parity(x)     //二进制1的个数的奇偶性
__builtin_ctz(x)  // x二进制末尾0的个数
__builtin_clz(x)  // x二进制末尾1的个数
log2(x) = 31-__builtin_clz(x); // 等价转换
```