CptCodingTemplates

计算几何

Point

```
#include<bits/stdc++.h>
using db = double;

class point{
public:
    db fs , sc;
    point(db x , db y) : fs(x) , sc(y) {}

    friend point operator+(const point& a , const point& b){ return point(a.fs +
b.fs , a.sc + b.sc); }
    friend point operator-(const point& a , const point& b){ return point(a.fs -
b.fs , a.sc - b.sc); }
    friend point operator*(const db& t    , const point& a){ return point(t * a.fs
, t * a.sc); }
    friend point operator*(const point& a , const db& t)   { return point(t * a.fs
, t * a.sc); }
    friend db    operator*(const point& a , const point& b){ return a.fs * b.sc -
a.sc * b.fs; } // cross
    friend db    operator%(const point& a , const point& b){ return a.fs * b.fs +
a.sc * b.sc; } // dot

    db len()        { return sqrt(fs * fs + sc * sc); }
    db ang(point& b){ return acos(*this % b / this -> len() / b.len()); }
    db dis(point& b){ return sqrt(1.0 * (b.fs - fs) * (b.fs - fs) + (b.sc -sc) *
(b.sc - sc)); }
};

point getIntsct(point a , point DA , point b , point DB){ // point ---direction---
>
    db ratio = (a - b) * DB / (DB * DA);
    return a + DA * ratio;
}
```

数据结构

并查集

```
#include<bits/stdc++.h>
#define all(x) (x).begin() , (x).end()
using namespace std;

class DSU{
```

```
private:
    vector<int>fa , rk;
public:
    DSU(int n){
        fa.assign(n , 0); rk.assign(n , 0);
        iota(all(fa) , 0);
    }
    int find(int x){
        if(fa[x] == x)return x;
        else return fa[x] = find(fa[x]);
    }

    void merge(int x , int y){
        x = find(x); y = find(y);
        if(x == y)return;
        if(rk[x] < rk[y]){
            fa[x] = y;
        }else{
            fa[y] = x;
            if(rk[x] == rk[y])rk[x] ++;
        }
    }

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

对顶堆维护中位数

```

#include<bits/stdc++.h>
#define int long long
using namespace std;

class Max_Min_Heap{
private:
    multiset<int>low , high;
    //low : 1 ~ (n + 1) / 2 , high : (n + 1) / 2 + 1 ~ n
    //median : *low.rbegin();
    void balance(){
        while(low.size() > high.size() + 1){
            high.insert(*low.rbegin());
            low.erase(-- low.end());
        }
        while(low.size() < high.size()){
            low.insert(*high.begin());
            high.erase(high.begin());
        }
    }
public:
    void add(int x){
        if(!high.empty() && x >= *high.begin())high.insert(x);
        else low.insert(x);
        balance();
    }
    void del(int x){
        if(high.count(x))high.erase(high.find(x));
        else if(low.count(x))low.erase(low.find(x));
        balance();
    }
    int get(){
        return *low.rbegin();
    }
};
```

线段树

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

#define int long long

const int INF = 1e9+7;
const int N = 100010;
const int MOD=998244353;

typedef pair<int, int>P;

int n, cmd;
int a[N];
int lazy[N * 4], node[N * 4];

inline int ls(int x) {
    return x << 1;//左儿子：乘2
}

inline int rs(int x) {
    return x << 1 | 1;//右儿子：乘2加1
}

inline void pushup(int now) {
    node[now] = node[ls(now)] + node[rs(now)];
}

void build(int now, int l, int r) {
    lazy[now] = 0;
    if (l == r) { node[now] = a[l]; return; }//区间长度为1
    int mid = (l + r) >> 1;
    build(ls(now), l, mid);
    build(rs(now), mid + 1, r);
    pushup(now);
```

```
}

inline void add(int now, int l, int r, int key) {
    lazy[now] += key;//
    node[now] += key * (r - l + 1);
}

inline void pushdown(int fa, int l, int r) {
    int mid = (l + r) >> 1;
    add(ls(fa), l, mid, lazy[fa]);
    add(rs(fa), mid + 1, r, lazy[fa]);
    lazy[fa] = 0;
}

inline void update(int L, int R, int l, int r, int now, int key) {
    if (L <= l && R >= r) {
        node[now] += key * (r - l + 1);
        lazy[now] += key;
        return;
    }
    pushdown(now, l, r);
    int mid = (l + r) >> 1;
    if (L <= mid)update(L, R, l, mid, ls(now), key);
    if (R > mid)update(L, R, mid + 1, r, rs(now), key);
    pushup(now);
}

int ask(int L, int R, int l, int r, int now) {
    int res = 0;
    if (L <= l && R >= r)return node[now];
    int mid = (l + r) >> 1;
    pushdown(now, l, r);
    if (L <= mid)res += ask(L, R, l, mid, ls(now));
    if (R > mid)res += ask(L, R, mid + 1, r, rs(now));
    return res;
}

void solve() {
    int type; cin >> type;
    if (type == 1) {
        int l, r, key; cin >> l >> r >> key; l--, r--;
        update(l, r, 0, n - 1, 1, key);
    }
    else if (type == 2) {
        int l, r; cin >> l >> r; l--, r--;
        cout << ask(l, r, 0, n - 1, 1) << endl;
    }
}

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    cin >> n >> cmd;
    for (int i = 0; i < n; ++i)cin >> a[i];
    build(1, 0, n - 1);
    while (cmd--) {
        solve();
    }
    return 0;
}

// in:
// //n,cmd
// // a[i]
// //type l r (if(type==1)key)
// 5 5
// 1 5 4 2 3
// 2 2 4
// 1 2 3 2
// 2 3 4
// 1 1 5 1
// 2 1 4

// out:
// 11
// 8
// 20
```

ST表

```
#include<bits/stdc++.h>
using namespace std;
const int inf=1e9+7;

class ST{
private:
    vector<vector<int> >st;
    string tp;
public:
    ST(vector<int>a,string type){//1<=i<=n
        tp=type; assert(tp=="max" || tp=="min");
        int n=a.size()-1;
        st.assign(n+1,vector<int>(20,inf));
        for(int i=1;i<=n;++i)st[i][0]=a[i];
        if(tp=="max"){
            for(int j=1;j<=20;++j){
                for(int i=1;i+(1<<j)-1<=n;++i){
                    st[i][j]=max(st[i][j-1],st[i+(1<<(j-1))][j-1]);
                }
            }
        }else if(tp=="min"){
            for(int j=1;j<=20;++j){
                for(int i=1;i+(1<<j)-1<=n;++i){
                    st[i][j]=min(st[i][j-1],st[i+(1<<(j-1))][j-1]);
                }
                }
            }
        }

        int rg(int l,int r){
            int k=(int)log2(r-l+1);
            if(tp=="max")return max(st[l][k],st[r-(1<<k)+1][k]);
            else return min(st[l][k],st[r-(1<<k)+1][k]);
        }
};

void solve(){
    int n;cin >> n;
    vector<int>a(n+1);
    for(int i=1;i<=n;++i)cin >> a[i];
    int q;cin >> q;
    ST st1(a,"max"),st2(a,"min");
    while(q--){
        int t,l,r;cin >> t >> l >> r;
        if(t==0)cout << st1.rg(l,r) <<'\n';
        else if(t==1)cout << st2.rg(l,r) << '\n';
    }
}
```

图论

Tarjan_LCA

```
#include<bits/stdc++.h>
#define all(x) (x).begin() , (x).end()
#define fs first
#define sc second
using namespace std;
using pii = pair<int , int>;

int n , q;
vector<vector<int> >g;
vector<vector<pii> >qry;
vector<int>fa , ans;
vector<bool>vis;

int find(int x){
    if(x == fa[x])return x;
    else return fa[x] = find(fa[x]);
}

void dfs(int u){
    vis[u] = true;
    for(auto v:g[u]){
        if(!vis[v]){
            dfs(v);
```

```
                fa[v] = u;
            }
        }
        for(auto i:qry[u]){
            int v = i.fs , id = i.sc;
            if(vis[v])ans[id] = find(v);
        }
    }

    signed main(){
        cin >> n >> q;
        g.assign(n + 1 , vector<int>());
        qry.assign(n + 1  , vector<pii>());
        for(int i=1;i<=n - 1;++i){
            int u , v;cin >> u >> v;
            g[u].push_back(v);
            g[v].push_back(u);
        }
        for(int i=1;i<=q;++i){
            int u , v;cin >> u >> v;
            qry[u].push_back({v , i});
            qry[v].push_back({u , i});
        }
        ans.assign(q + 1 , 0);
        fa.assign(n + 1  , 0); iota(all(fa) , 0);
        vis.assign(n + 1 , false);
        dfs(1);
        for(int i=1;i<(int)ans.size();++i)cout << ans[i] << '\n';
        return 0;
    }
```

拓扑排序

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

using namespace std;

#define int long long

const int N=50010;

int n,m;
vector<int>g[N];
int in[N];

bool toposort() {
    vector<int> L;
    queue<int> S;
    for (int i = 1; i <= n; i++)
    if (in[i] == 0) S.push(i);
    while (!S.empty()) {
    int u = S.front();
    S.pop();
    L.push_back(u);
    for (auto v : g[u]) {
        if (--in[v] == 0) {
            S.push(v);
        }
    }
    }
    if (L.size() == n) {
    for (auto i : L) cout << i << ' ';
        return true;
    } else {
        return false;
    }
}


void solve(){
    cin>>n>>m;
    for(int i=0;i<m;++i){
        int u,v;cin>>u>>v;
        g[u].push_back(v);
    }
    if(toposort())cout<<"YES"<<'\n';
    else cout<<"NO"<<'\n';
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    solve();
    return 0;
}
```

BellmanFord

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

using namespace std;

#define int long long

struct edge{
    int from,to,cost;
};

const int MAXE=50010;
const int MAXV=50010;
const int INF=1e9+7;

```

```
edge es[MAXE];
int d[MAXV];
int V,E;

void BellmanFord(int s){
    fill(d,d+V,INF);
    d[s]=0;
    while(1){
        bool update=false;
        for(int i=0;i<E;++i){
            edge e=es[i];
            if(d[e.from]!=INF && d[e.to]>d[e.from]+e.cost){
                d[e.to]=d[e.from]+e.cost;
                update=true;
            }
        }
        if(!update)break;//直到遍历一次没有更新：找到最优解
    }
}

bool FindNegativeLoop(){
    memset(d,0,sizeof(d));
    for(int i=0;i<V;++i){
        for(int j=0;j<E;++j){
            edge e=es[j];
            if(d[e.to]>d[e.from]+e.cost){
                d[e.to]=d[e.from]+e.cost;
                if(i==V-1)return true;//如果第V次仍然更新，则存在负圈
            }
        }
    }
    return false;
}

void solve(){
    cin>>V>>E;
    for(int i=0;i<E;++i){
        int u,v,cost;cin>>u>>v>>cost;
        es[i].cost=cost;es[i].from=u;es[i].to=v;
    }
    int s;cin>>s;//起点
    BellmanFord(s);
    if(FindNegativeLoop())cout<<"ExistNegativeLoop"<<'\n';
    else cout<<"NoNegativeLoop"<<'\n';
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    solve();
    return 0;
}
```

Dijkstra次短路

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

using namespace std;

#define int long long

typedef pair<int,int>P;

const int INF=1e9+7;

int dijkstra(vector<P>g[],int s,int t,int V){
    priority_queue<P>q;
    q.push({0,s});
    vector<int>dis(V,INF);
    while(!q.empty()){
        P now=q.top();q.pop();

    }
    return dis[t];
}

void solve(){
    int V,E;cin>>V>>E;
    int s,t;cin>>s>>t;
    vector<P>g[V];
    for(int i=0;i<V;++i){
        int u,v,c;cin>>u>>v>>c;
        g[u].push_back({c,v});
        g[v].push_back({c,u});
    }
    dijkstra(g,s,t,V);

}
```

Dijkstra路径还原

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

const int N=70000;
const int INF=1e9+7;
const int NIL=-1000000007;
const double pi=acos(-1.0);

//
//Vertex indexes start with 1
//
```

```
typedef pair<int, int>P;//1st:cost , 2nd:to

int V, E, s, t;
vector<P>edge[N];
int pre[N];
int dis[N];

void dijkstra(int s) {
    priority_queue<P, vector<P>, greater<P> >que;
    fill(dis, dis + V + 1, INF); dis[s] = 0;
    fill(pre,pre+V,-1);
    que.push(P(0, s));
    while (!que.empty()) {
        P p = que.top(); que.pop();
        int v = p.second;
        if (dis[v] < p.first)continue;
        for (int i = 0; i < (int)edge[v].size(); ++i) {
            P e = edge[v][i];
            if (dis[e.second] > dis[v] + e.first) {
                dis[e.second] = dis[v] + e.first;
                pre[e.second]=v;
                que.push(P(dis[e.second], e.second));
            }
        }
    }
}

vector<int> GetPath(int t){
    vector<int>path;
    for(;t!=-1;t=pre[t])path.push_back(t);//直到走到s(起点)
    reverse(path.begin(),path.end());
    return path;
}

signed main() {
    cin >> V >> E >> s >> t;
    for (int i = 0; i < E; ++i) {
        int u, v, w; cin >> u >> v >> w;
        edge[u].push_back(P(w, v)); edge[v].push_back(P(w, u));
    }
    dijkstra(s);
    cout<<dis[t]<<'\n';
    vector<int>path=GetPath(t);
    for(auto i:path)cout<<i<<' ';
    cout<<'\n';
    return 0;
}
```

Kruskal

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

using namespace std;

#define int long long

const int N=500010;
const int MAXE=500010;

struct edge{
    int u,v,cost;
};

int fa[N];
int rk[N];

void init(int n){
    for(int i=0;i<n;++i){
        fa[i]=i;
        rk[i]=0;
    }
}

int find(int x){
    if(fa[x]==x){
        return x;
    }else{
        return fa[x]=find(fa[x]);
    }
}

void merge(int x,int y){
    x=find(x);
    y=find(y);
    if(x==y)return;
    if(rk[x]<rk[y]){
        fa[x]=y;
    }else {
        fa[y]=x;
        if(rk[x]==rk[y])rk[x]++;
    }
}

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

bool cmp(edge e1,edge e2){
    return e1.cost<e2.cost;
}

edge es[MAXE];
int V,E;
int kruskal(){
    sort(es,es+E,cmp);
    init(V);
    int res=0;
    for(int i=0;i<E;++i){
        edge e=es[i];
        if(!same(e.u,e.v)){
            merge(e.u,e.v);
            res+=e.cost;
        }
    }
    return res;
}

void solve(){
    cin>>V>>E;
    for(int i=0;i<E;++i){
        cin>>es[i].u>>es[i].v>>es[i].cost;
    }
    cout<<kruskal()<<'\n';
}
```

字符串

通用Trie

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

class TrieNode{
public:
    unordered_map<char , TrieNode*>child;
    // map<char,TrieNode*>child;
    int cnt , cntpre;
    bool isend;
    TrieNode(){cnt = 0; cntpre = 0; isend = false;}
};

class Generic_Trie{
private:
    TrieNode* rt;
public:
    Generic_Trie(){
        rt = new TrieNode();
    }
    void insert(string s){
        TrieNode* now = rt;
        for(auto i:s){
            now -> cntpre++;
            if(now -> child.find(i) == now -> child.end()){
                now -> child[i] = new TrieNode();
            }
            now = now -> child[i];
        }
        now -> cnt++; now -> cntpre++; now -> isend = true;
    }
    int askpre(string s){
        TrieNode* now = rt;
        for(auto i:s){
            if(now -> child.find(i) == now -> child.end())return 0;
            now = now -> child[i];
        }
        return now -> cntpre;
    }
    int find(string s){
        TrieNode* now = rt;
        for(auto i:s){
            if(now -> child.find(i) == now -> child.end())return false;
            now = now -> child[i];
        }
        return now -> isend;
    }
};

void solve(){
    Generic_Trie trie;
    int n;cin >> n;
    for(int i=0;i<n;++i){
        string s;cin >> s;
        trie.insert(s);
    }
    int q;cin >> q;
    while(q --){
        string x;cin >> x;
        if(trie.find(x))cout << "Exist" << '\n';
        else cout << "Nope" << '\n';
    }
    cin >> q;
    while(q--){
        string x;cin >> x;
        cout << trie.askpre(x) << '\n';
    }
}
```

KMP

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

#define NO cout<<"NO"<<'\n';
```

```
#define NOO cout<<-1<<'\n';
#define YES cout<<"YES"<<'\n';
#define Yes cout<<"Yes"<<'\n';
#define No cout<<"No"<<'\n';
#define fs first
#define sc second

using db=double;
using ll=long long;
using ull=unsigned long long;
using pii=pair<int,int>;
using pll=pair<ll,ll>;
using pdd=pair<db,db>;
using i128=__int128_t;
using ui128=__uint128_t;

const db eps=1e-10;
const db pi=acos(-1.0);
const int inf=1e9+7;
const int nil=-inf;
const ll INF=1e18+10;
const ll NIL=-INF;
const int MOD=998244353;

class KMP{
private:
    string text;
    vector<int> getnx(string p){
        int n=p.size();
        p=' '+p; vector<int>nx(p.size());
        nx[1]=0;
        for(int i=2,j=0;i<=n;++i){
            while(j && p[i]!=p[j+1])j=nx[j];
            if(p[i]==p[j+1])j++;
            nx[i]=j;
        }
        return nx;
    }
public:
    KMP(string s){
        text=s; text=' '+text;
    }
    vector<int> match(string p){
        int n=p.size(),m=text.size();
        vector<int>nx=getnx(p);
        p=' '+p;
        vector<int>pos;
        for(int i=1,j=0;i<=m;++i){
            while(j && text[i]!=p[j+1])j=nx[j];
            if(text[i]==p[j+1])j++;
            if(j==n)pos.push_back(i-n+1);
        }
        return pos;
    }
};
```

字符串哈希

```
#include<bits/stdc++.h>
using namespace std;
using ll = long long;
using ull = unsigned long long;

class StringHash{
private:
    const ll P = 13331;
    vector<ull>poly , hash_val;
public:
    StringHash(string s){
        int n = s.size();
        s = ' ' + s;
        poly.assign(n + 1 , 1); hash_val.assign(n + 1 , 0);
        for(int i=1;i<=n;++i){
            poly[i] = poly[i - 1] * P;
            hash_val[i] = hash_val[i - 1] * P + s[i];
        }
    }
    ull cal(int l , int r){
        return hash_val[r] - hash_val[l - 1] * poly[r - l + 1];
    }
    bool same(int l1 , int r1 , int l2 , int r2){
        return cal(l1 , r1) == cal(l2 , r2);
    }
};
```

数学

FFT

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
using db = double;
const db PI = acos(-1);

void butterfly(vector<complex<db> >& A , int n){
    vector<int>dp(n, 0);
    for(int i=0;i<n;++i)dp[i] = dp[i / 2] / 2 + ((i & 1) ? n / 2 : 0);
    for(int i=0;i<n;++i)if(i < dp[i])swap(A[i] , A[dp[i]]);
}
```

```
void FFT(vector<complex<db> >& A , int n){
    butterfly(A , n);
    for(int m=2;m<=n;m <<= 1){
        complex<db> spin{cos(2 * PI / m) , sin(2 * PI / m)};
        for(int i=0;i<n;i += m){
            complex<db> wk = {1 , 0};
            for(int j=0;j<m / 2;++j){
                complex<db> x = A[i + j] , y = A[i + j + m / 2] * wk;
                A[i + j] = x + y;
                A[i + j + m / 2] = x - y;
                wk *= spin;
            }
        }
    }
}


void IFFT(vector<complex<db> >& A , int n){
    butterfly(A , n);
    for(int m=2;m<=n;m <<= 1){
        complex<db> spin{cos(2 * PI / m) , -sin(2 * PI / m)};
        for(int i=0;i<n;i += m){
            complex<db> wk = {1 , 0};
            for(int j=0;j<m / 2;++j){
                complex<db> x = A[i + j] , y = A[i + j + m / 2] * wk;
                A[i + j] = x + y;
                A[i + j + m / 2] = x - y;
                wk *= spin;
            }
        }
    }
    for(int i=0;i<n;++i)A[i] /= n;
}
```

线性筛

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

#define NO cout<<"NO"<<'\n';
#define NOO cout<<-1<<'\n';
#define YES cout<<"YES"<<'\n';
#define Yes cout<<"Yes"<<'\n';
#define No cout<<"No"<<'\n';
#define fs first
#define sc second

using db=double;
using ll=long long;
using ull=unsigned long long;
using pii=pair<int,int>;
using pll=pair<ll,ll>;
using pdd=pair<db,db>;
using i128=__int128_t;
using ui128=__uint128_t;

const db eps=1e-10;
const db pi=acos(-1.0);
const int inf=1e9+7;
const int nil=-inf;
const ll INF=1e18+10;
const ll NIL=-INF;
const int MOD=998244353;

vector<int> EulerSieve(int n){
    vector<int>pri;
    vector<bool>not_pri;
    not_pri.assign(n+1,false);
    for(int i=2;i<=n;++i){
        if(!not_pri[i])pri.push_back(i);
        for(auto j:pri){
            if(j*i>n)break;
            not_pri[i*j]=true;
            if(i%j==0)break;
        }
    }
    return pri;
}

void solve(){
    vector<int>pri=EulerSieve(300);
    for(auto i:pri)cout << i << ' ';
    cout << '\n';
}
```

乘法逆元/快速幂

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

#define NO cout<<"NO"<<'\n';
#define NOO cout<<-1<<'\n';
#define YES cout<<"YES"<<'\n';
#define Yes cout<<"Yes"<<'\n';
#define No cout<<"No"<<'\n';
#define fs first
#define sc second

using db=double;
using ll=long long;
using ull=unsigned long long;
using pii=pair<int,int>;
using pll=pair<ll,ll>;
using pdd=pair<db,db>;
using i128=__int128_t;
using ui128=__uint128_t;

const db eps=1e-10;
const db pi=acos(-1.0);
const int inf=1e9+7;
const int nil=-inf;
const ll INF=1e18+10;
const ll NIL=-INF;
const int MOD=998244353;

int fpowMOD(int a,int n,int p){
    int res=1;
    while(n){
        if(n&1){
            res=res*a%p;
        }
        a=a*a%p;
        n>>=1;
    }
    return res;
}

void solve(){
    int a,p;cin >> a >> p;
    if(a%p)cout << fpowMOD(a,p-2,p) << '\n';
}
```

杂项

i128

```
#include<bits/stdc++.h>
#define all(x) (x).begin() , (x).end()
using namespace std;
using i128 = __int128_t;

istream &operator>>(istream &is , i128 &n){
    n = 0;
    string s;
    is >> s;
    for(auto c:s)n = 10 * n + c - '0';
    return is;
}

ostream &operator<<(ostream &os , i128 &n){
    if(n == 0)return os << 0;
    string s;
    while(n > 0){
        s += '0' + n % 10;
        n /= 10;
    }
    reverse(all(s));
    return os << s;
}
```

高精度

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
using ll = long long;

class BigInt{
private:
    static const int BASE = 100000000;
    static const int WIDTH = 8;
    vector<int>a;
    bool sign;

    // 去除前导零 : 许多东西的依赖
    void trim() {
        while(a.size() > 1 && a.back() == 0) a.pop_back();
        if(a.size() == 1 && a[0] == 0) sign = false;
    }

    // 无符号比较 : 许多东西的依赖
    bool abs_less(const BigInt& b) const {
        if(a.size() != b.a.size()) return a.size() < b.a.size();
        for(int i=a.size() - 1;i>=0;i--){
            if(a[i] != b.a[i]) return a[i] < b.a[i];
        }
        return false;
    }

    // 无符号加减法 : 加减法 , 无符号带余数除法的依赖
    BigInt abs_add(const BigInt& b) const {
        BigInt res;
        res.a.clear();
        int cry = 0;
        for(int i=0;i<a.size() || i < b.a.size() || cry;i++){
            if(i < a.size()) cry += a[i];
            if(i < b.a.size()) cry += b.a[i];
            res.a.push_back(cry % BASE);
            cry /= BASE;
        }
        res.trim();
        return res;
    }
    BigInt abs_sub(const BigInt& b) const {
        BigInt res = *this;
        for(int i=0, brw=0;i < b.a.size() || brw;i++){
            res.a[i] -= (i < b.a.size() ? b.a[i] : 0) + brw;
            brw = (res.a[i] < 0);
            if (brw) res.a[i] += BASE;
        }
        res.trim();
        return res;
    }

    // 无符号带余数（通过r传出）除法 ： 除法/取模的依赖 ，同时它依赖乘法重载
    BigInt abs_mod_div(const BigInt& b, BigInt& r) const {
        BigInt q(0);
        r = *this;
        if(r.abs_less(b)) return q;

        BigInt tmpr = *this;
        q.a.resize(a.size() - b.a.size() + 1);

        for(int i=a.size() - b.a.size();i>=0;--i){
            ll low = 0, high = BASE, ansq = 0;

            while(low <= high){
                ll mid = low + (high - low) / 2;
                BigInt tmpp = b.abs_add(BigInt(0)) * BigInt(mid);
                if(!tmpr.abs_less(tmpp)){
                    ansq = mid;
                    low = mid + 1;
                }else{
                    high = mid - 1;
                }
            }
            q.a[i] = ansq;
            tmpr = tmpr.abs_sub(b.abs_add(BigInt(0)) * BigInt(ansq));
        }
        q.trim();
        r = tmpr;
        r.sign = false;
        return q;
    }

public:
    BigInt(ll x = 0) : sign(false){
        if(x < 0) sign = true, x = -x;
        if(x == 0) a.push_back(0);
        else while(x > 0){
            a.push_back(x % BASE);
            x /= BASE;
        }
        trim();
    }
    BigInt(const string& s) : sign(false){
        int st = 0;
        if(s[0] == '-') sign = true, st = 1;

        for(int i=s.length();i>st;i-=WIDTH){
            if(i - WIDTH < st) a.push_back(stoi(s.substr(st, i - st)));
            else a.push_back(stoi(s.substr(i - WIDTH, WIDTH)));
        }
        if(s.length() == st) a.push_back(0);
        trim();
    }

    // <<<<<<<<<<<<<<<<<<<<<< 比较运算符 >>>>>>>>>>>>>>>>>>>>>>
    bool operator<(const BigInt& b) const {
        if(sign != b.sign) return sign;
        if(!sign) return abs_less(b);
        return !abs_less(b) && *this != b;
    }
    bool operator>(const BigInt& b) const { return b < *this; }
    bool operator<=(const BigInt& b) const { return !(*this > b); }
    bool operator>=(const BigInt& b) const { return !(*this < b); }
    bool operator==(const BigInt& b) const { return sign == b.sign && a == b.a; }
    bool operator!=(const BigInt& b) const { return !(*this == b); }

    // +++++++++++++++++++++++++ 加法 +++++++++++++++++++++++++
    BigInt operator+(const BigInt& b) const {
        if(sign == b.sign){
            BigInt res = abs_add(b);
            res.sign = sign;
            return res;
        }else{
            if(abs_less(b)){
                BigInt tmp = b.abs_sub(*this);
                tmp.sign = b.sign;
                return tmp;
            }else{
                BigInt tmp = abs_sub(b);
                tmp.sign = sign;
                return tmp;
            }
        }
    }

    // ---------------------- 减法(依赖加法) ----------------------
    BigInt operator-(const BigInt& b) const {
        BigInt tmp = b;
        tmp.sign = !b.sign;
        return *this + tmp;
    }

    // ********************** 乘法 **********************
    BigInt operator*(const BigInt& b) const {
        BigInt res;
        res.a.assign(a.size() + b.a.size(), 0);
        for(int i=0;i<a.size();++i){
            ll cry = 0;
            for(int j=0;j<b.a.size() || cry;++j){
                ll cur = res.a[i + j] + cry;
                if(j < b.a.size()) cur += (ll)a[i] * b.a[j];
                res.a[i + j] = cur % BASE;
                cry = cur / BASE;
            }
        }
        res.trim();
        res.sign = (res.a.size() != 1 || res.a[0] != 0) && (sign != b.sign);
        return res;
    }

    // ////////////////////// 除法 //////////////////////
    BigInt operator/(const BigInt& b) const {
        BigInt r;
        BigInt q = abs_mod_div(b, r);
        bool signres = (q.a.size() != 1 || q.a[0] != 0) && (sign != b.sign);
        q.sign = signres;
        return q;
    }

    BigInt operator%(const BigInt& b) const {
        BigInt r;
        abs_mod_div(b, r);
        bool signres = (r.a.size() != 1 || r.a[0] != 0) && sign;
        r.sign = signres;
        return r;
    }

    // <<<<<<<<<<<<<<<<<<<<<< 输入输出 >>>>>>>>>>>>>>>>>>>>>>
    friend istream& operator>>(istream& is, BigInt& n){
        string s;
        is >> s;
        n = BigInt(s);
        return is;
    }
    friend ostream& operator<<(ostream& os, const BigInt& n){
        if(n.a.empty() || (n.a.size() == 1 && n.a[0] == 0)) return os << 0;
        if(n.sign) os << '-';
        os << n.a.back();
        for(int i=n.a.size() - 2;i>=0;--i){
            os << setfill('0') << setw(WIDTH) << n.a[i];
        }
        return os;
    }
};
```

CMakelist

```
cmake_minimum_required(VERSION 3.25)
project(ICPC)

set(CMAKE_C_STANDARD 14)

set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/bin)

file(GLOB_RECURSE SOURCES "*.cpp")

foreach(SRC ${SOURCES})
    get_filename_component(FILENAME ${SRC} NAME_WE)
    add_executable(${FILENAME} ${SRC})
endforeach()
```