Standard Code Library

Your TeamName

Your School

March 18, 2021

Contents

一切的开始

宏定义

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
typedef unsigned u32;
typedef unsigned long long u64;
typedef long double LD;
#define il inline
#define pln putchar('\n')
#define For(i,a,b)  for(int i=(a),(i##i)=(b);i<=(i##i);++i)
#define Rep(i,n)    for(int i=0,(i##i)=(n);i<(i##i);++i)
#define Fodn(i,a,b) for(int i=(a),(i##i)=(b);i>=(i##i);--i)
const int M=1000000007,INF=0x3f3f3f3f;
const long long INFLL=0x3f3f3f3f3f3f3f3fLL;
const int N=1000010;
// ----------------------------------------------------------------------------
```

数学

快速幂

- 注意乘法溢出

```cpp
inline LLL fp(LLL a, LLL b, LLL Mod) {
    LLL res = (Mod != 1);
    for (a %= Mod; b; b >>= 1, a = a * a % Mod)
        if (b & 1) res = res * a % Mod;
    return res;
}
```

GCD

```cpp
template <typename T>
inline T gcd(T a, T b) {
    while (b){
        T t = b;
        b = a % b;
        a = t;
    }
    return a;
}

template <typename T>
inline T lcm(T a, T b) { return a / gcd(a, b) * b; }

template <typename T>
inline T exgcd(T a, T b, T &x, T &y) {
    T m = 0, n = 1, t;
    x = 1, y = 0;
    while (b){
        t = m, m = x - a / b * m, x = t;
        t = n, n = y - a / b * n, y = t;
        t = b, b = a % b, a = t;
    }
    return a;
}
```

CRT

- 同余方程合并
- 返回最小正数解或最小非负解无解则返回-1

```cpp
inline LL Crt(LL a1, LL a2, LL mod1, LL mod2) {
    LL u, v;
    LL g = exgcd(mod1, mod2, u, v);
```

```cpp
    if ((a2 - a1) % g)
        return -1;
    LL m12 = abs(lcm(mod1, mod2));
    LLL res = (((LLL)mod1 * ((LLL)u * ((a2 - a1) / g) % m12) % m12) + a1) % m12;
    return res <= 0 ? res + m12 : res; /* 求最小正数解还是非负解 */
}
```

线性筛

```cpp
struct primenumberlist{
#define MAXN (100000000)
    int cnt, pri[10000000];
    bool np[MAXN + 10];
    primenumberlist(){
        np[1] = 1; cnt = 0;
        for (int i = 2; i <= MAXN; ++i) {
            if (!np[i]) pri[++cnt] = i;
            for (int j = 1; j <= cnt; ++j) {
                LL t = pri[j] * i;
                if (t > MAXN) break;
                np[t] = 1;
                if (!(i % pri[j])) break;
            }
        }
    }
} prime;
```

φ 欧拉函数

```cpp
template <typename T>
inline T phi(T x) {
    T res = x;
    for (T i = 2; i * i <= x; ++i)
        if ((x % i) == 0) {
            res = res / i * (i - 1);
            while ((x % i) == 0) x /= i;
        }
    if (x > 1) res = res / x * (x - 1);
    return res;
}
```

Miller-Rabin 素性测试

- $n <= 10^{18}$

```cpp
namespace MillerRabin {
    const LLL test[]={2ll,325ll,9375ll,28178ll,450775ll,9780504ll,1795265022ll};

    inline bool isprime(LLL n) {
        if (n==13||n==19||n==73||n==193||n==407521||n==299210837ll)return 1;
        if (n <= 3) return n > 1;
        if (n <= 6) return n == 5;
        if (!(n & 1) || !(n % 3) || !(n % 5)) return 0;

        LLL d = n - 1; int t = 0;
        while (!(d & 1)) d >>= 1, ++t;
        for (LLL ai = 0, a = test[0]; ai < 7; ++ai, a = test[ai]) {
            if (a % n == 0) return 0;
            LLL v = fp(a, d, n); if (v == 1 || v == n - 1) continue;
            LLL pre = v;
            for (int i = 1; i <= t; ++i) {
                v = v * v % n;
                if (v == 1)
                    if (pre != 1 && pre != (n - 1)) return 0; else break;
                pre = v;
            }
            if (v != 1) return 0;
        }
        return 1;
```

```
    }
}
```

Pollard-Rho 分解质因数

- 求 n 的一个非平凡因子
- 调用 `pollard_rho()` 前先判断 n 的素性

```cpp
namespace PollardRho{
    mt19937 mt(20011224);//19491001

    inline LLL pollard_rho(LLL n, LLL c) {
        LLL x = uniform_int_distribution<LL>(1, n - 1)(mt), y = x;
        LLL val = 1;
        for (int dep = 1;; dep <<= 1, x = y, val = 1) {
            for (int stp = 1; stp <= dep; ++stp) {
                y = (y * y + c) % n;
                val = val * abs(x - y) % n;
                if ((stp & 127) == 0) {
                    LLL d = gcd(val, n);
                    if (d > 1) return d;
                }
            }
            LLL d = gcd(val, n);
            if (d > 1) return d;
        }
    }

    //接口根据题意重写
    vector<LLL> factor;
    void getfactor(LLL x, LLL c = 19260817) {
        if (MillerRabin::isprime(x)) {factor.emplace_back(x); return;}
        LLL p = x;
        while (p == x) p = pollard_rho(x, c--);
        getfactor(p); getfactor(x / p);
    }
    inline LLL ask(LLL x) {
        factor.clear();
        while (!(x & 1)) x >>= 1, factor.emplace_back(2);
        if (x > 1) getfactor(x);
        return factor.size();
    }
}
```

组合数

- 数较小模数为较大质数求逆元
-
    - 如果模数固定可以 O(n) 预处理阶乘的逆元
- 数较大模数为较小质数用 $Lucas$ 定理
-
    -

$$C_n^m \equiv C_{\lfloor \frac{n}{p} \rfloor}^{\lfloor \frac{m}{p} \rfloor} * C_{n\ mod\ p}^{m\ mod\ p} (mod\ p)$$

- 数较大模数较小用 $exLucas$ 定理求 $C_n^m mod\ P$

exLucas

- O(P log P)
- 不要求 P 为质数

```cpp
namespace EXLUCAS {
    inline LL idxp(LL n, LL p) {
        LL nn = n;
        while (n > 0) nn -= (n % p), n /= p;
```

```
        return nn / (p - 1);
    }

    LL facp(LL n, LL p, LL pk) {
        if (n == 0) return 1;
        LL res = 1;
        if (n >= pk) {
            LL t = n / pk, k = 1, els = n - t * pk;
            for (LL i = 1; i <= els; ++i) if (i % p) k = k * i % pk;
            res = k;
            for (LL i = els + 1; i < pk; ++i) if (i % p) k = k * i % pk;
            res = res * fp(k, n / pk, pk) % pk;
        }
        else for (LL i = 1; i <= n; ++i) if (i % p) res = res * i % pk;
        return res * facp(n / p, p, pk) % pk;
    }

    inline LL exlucas(LL n, LL m, LL p, LL pk, LL k) {
        LL a = facp(n, p, pk) * fp(facp(n - m, p, pk) * facp(m, p, pk) % pk, pk / p * (p - 1) - 1, pk) % pk;
        LL b = idxp(n, p) - idxp(m, p) - idxp(n - m, p);
        if (b >= k) return 0;
        while (b--) a *= p;
        return a % pk;
    }

    /* 接口 */ inline LL exlucas(LL n, LL m, LL p) {
        LL a = 0, b = 1;
        for (LL i = 2; i * i <= p; ++i) {
            if (p % i) continue;
            LL t = 0, pk = 1;
            while (p % i == 0) ++t, p /= i, pk *= i;
            a = Crt(a, exlucas(n, m, i, pk, t), b, pk);
            b *= pk;
        }
        return (p > 1) ? Crt(a, exlucas(n, m, p, p, 1), b, p) : a;
    }
}
```

图论

LCA

- 倍增

```
void dfs(int u, int fa) {
    pa[u][0] = fa; dep[u] = dep[fa] + 1;
    FOR (i, 1, SP) pa[u][i] = pa[pa[u][i - 1]][i - 1];
    for (int& v: G[u]) {
        if (v == fa) continue;
        dfs(v, u);
    }
}

int lca(int u, int v) {
    if (dep[u] < dep[v]) swap(u, v);
    int t = dep[u] - dep[v];
    FOR (i, 0, SP) if (t & (1 << i)) u = pa[u][i];
    FORD (i, SP - 1, -1) {
        int uu = pa[u][i], vv = pa[v][i];
        if (uu != vv) { u = uu; v = vv; }
    }
    return u == v ? u : pa[u][0];
}
```

计算几何

二维几何：点与向量

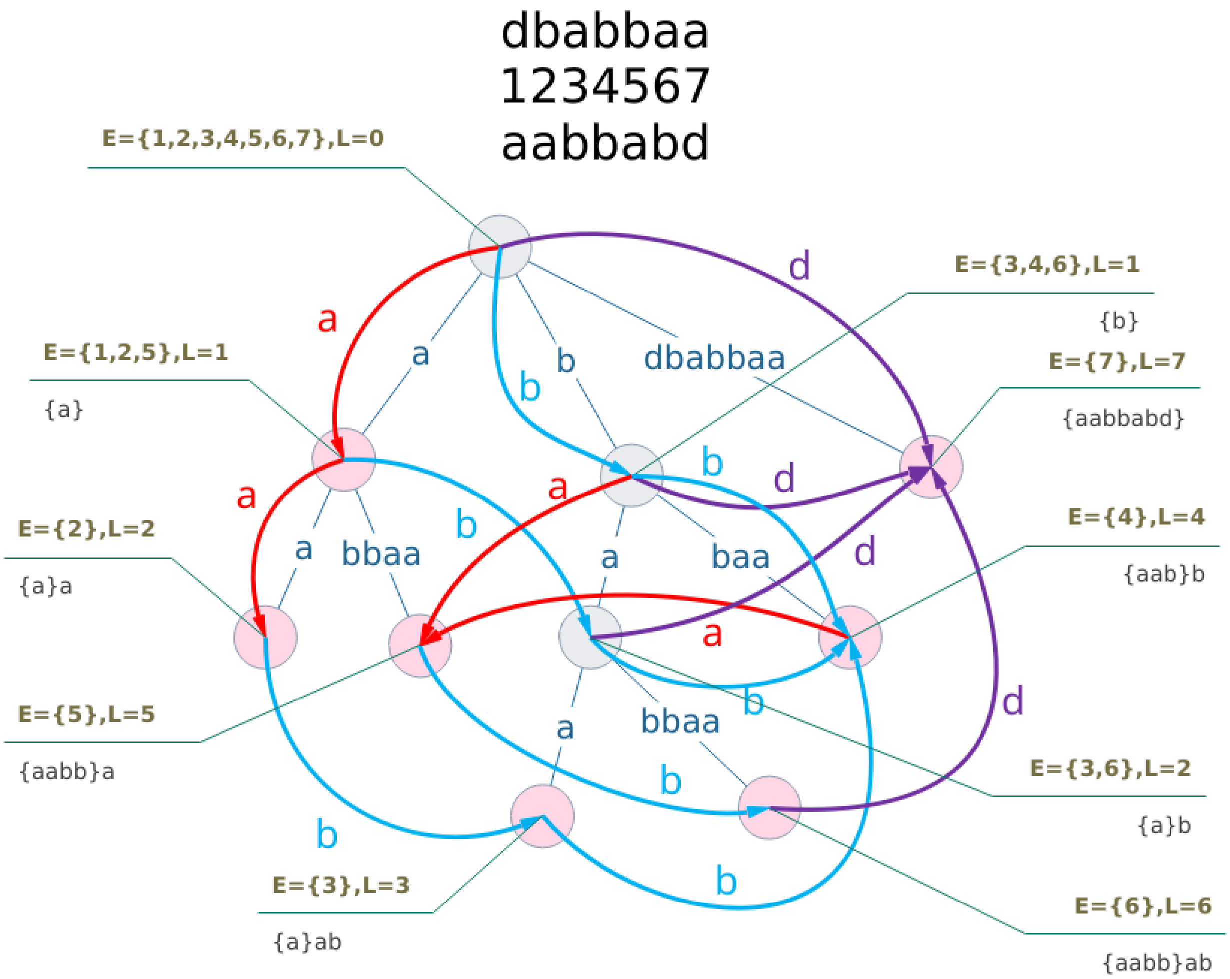
```cpp
#define y1 yy1
#define nxt(i) ((i + 1) % s.size())
typedef double LD;
const LD PI = 3.14159265358979323846;
const LD eps = 1E-10;
int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
struct L;
struct P;
typedef P V;
struct P {
    LD x, y;
    explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
    explicit P(const L& l);
};
struct L {
    P s, t;
    L() {}
    L(P s, P t): s(s), t(t) {}
};

P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
inline bool operator < (const P& a, const P& b) {
    return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
}
bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
P::P(const L& l) { *this = l.t - l.s; }
ostream &operator << (ostream &os, const P &p) {
    return (os << "(" << p.x << "," << p.y << ")");
}
istream &operator >> (istream &is, P &p) {
    return (is >> p.x >> p.y);
}

LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
// --------------------------------------------
```

字符串

后缀自动机



杂项

STL

- copy

```cpp
template <class InputIterator, class OutputIterator>
  OutputIterator copy (InputIterator first, InputIterator last, OutputIterator result);
```