
中国科学技术大学
1958
University of Science and Technology of China

ACM/ICPC Template

AsZ yuzhou627

October 14, 2015

Contents

Chapter 1

基础

1.1 手工开栈

```cpp
//G++
int size = 256 << 20; // 256MB
char *p = (char*)malloc(size) + size;
__asm__("movl %0, %%esp\n" :: "r"(p));
//C++
#pragma comment(linker, "/STACK:102400000,102400000")
```

Listing 1.1: Stack.cpp

1.2 对拍

```sh
while true; do
    ./gen > tmp.in
    ./std < tmp.in > std.out
    ./my < tmp.in > my.out
    if diff my.out std.out; then
        echo "AC"
    else
        echo "WA"
        exit 0
    fi
done
```

Listing 1.2: checker.sh

1.3 读写优化

```cpp
int in() {
    char c;
    while (c = getchar(), (c < '0' || c > '9') && (c != '-'));
    bool flag = (c == '-');
    if (flag) c = getchar();
    int x = 0;
    while (c >= '0' && c <= '9') {
        x = x * 10 + c - 48;
        c = getchar();
    }
    return flag ? -x : x;
}

void out(int x) { //int
    if (x < 0) putchar('-'), x = -x;
    int len = 0, bit[10]; // LL  -> bit[20]
    while (x) {
        bit[len++] = x % 10;
        x /= 10;
    }
    if (!len) bit[len++] = 0;
    while (len--) putchar(bit[len] + 48);
    putchar('\n');
}
```

Listing 1.3: FastIO.cpp

1.4 vimrc

```
set nu si cin et noswf nobk sw=4 ts=4 sts=4

imap [[ {<cr>}<c-o>O
map<F2> ggVG"+y
map<F3> : !g++ -std=c++11 -g -Wall -DHOME % -o %< <cr>
map<F4> : !./%< < %<.in <cr>
map<F5> : vs %<.in <cr>
```

Listing 1.4: vimrc.acm

1.5 内置位运算函数

1. __builtin_ffs(unsigned int x) 返回右起第一个 1 的位置.x==0, return 0

2. ___builtin_clz(unsigned int x) 返回左起第一个 1 之前 0 的个数.x==0,return unsigned int 的位数

3. ___builtin_ctz(unsigned int x) 返回右起第一个 1 之后的 0 的个数.x==0,return unsigned int 的位数

4. ___builtin_popcount(unsigned int x) 返回 1 的个数

5. ___builtin_parity(unsigned int x) 返回 1 的个数的奇偶性

6. 末尾加上 l unsigned long ll long long

1.6 输出格式控制

1.6.1 基本

1. %: 表示格式说明的起始符号, 不可缺少.

2. -: 有 -表示左对齐输出, 如省略表示右对齐输出.

3. 0: 有 0 表示指定空位填 0, 如省略表示指定空位不填.

4. m.n:m 指域宽, 即对应的输出项在输出设备上所占的字符数 (包括小数点).N 指精度. 用于说明输出的实型数的小数位数. 为指定 n 时, 隐含的精度为 n=6 位.

5. l 或 h:l 对整型指 long 型, 对实型指 double 型.h 用于将整型的格式字符修正为 short 型.

1.6.2 整数

1. d 格式: 用来输出十进制整数.
   %d: 按整型数据的实际长度输出.
   %md:m 为指定的输出字段的宽度. 如果数据的位数小于 m, 则左端补以空格, 若大于 m, 则按实际位数输出.
   %ld: 输出长整型数据.

2. o 格式: 以无符号八进制形式输出整数. 对长整型可以用”%lo” 格式输出. 同样也可以指定字段宽度用“%mo”格式输出.

3. x 格式: 以无符号十六进制形式输出整数. 对长整型可以用”%lx” 格式输出. 同样也可以指定字段宽度用”%mx” 格式输出.

4. u 格式: 以无符号十进制形式输出整数. 对长整型可以用”%lu” 格式输出. 同样也可以指定字段宽度用“%mu”格式输出.

1.6.3 字符串

1. %c 格式: 输出一个字符.

2. %s 格式: 用来输出一个串. 有几中用法

3. %ms: 输出的字符串占 m 列, 如字符串本身长度大于 m, 则突破获 m 的限制, 将字符串全部输出. 若串长小于 m, 则左补空格.

4. %-ms: 如果串长小于 m, 则在 m 列范围内, 字符串向左靠, 右补空格.

5. %m.ns: 输出占 m 列, 但只取字符串中左端 n 个字符. 这 n 个字符输出在 m 列的右侧, 左补空格.

6. %-m.ns: 其中 m,n 含义同上,n 个字符输出在 m 列范围的左侧, 右补空格. 如果 n>m, 则自动取 n 值, 即保证 n 个字符正常输出.

1.6.4 浮点数

1. %f: 不指定宽度, 整数部分全部输出并输出 6 位小数.
   %m.nf: 输出共占 m 列, 其中有 n 位小数, 如数值宽度小于 m 左端补空格.
   %0m.nf: 输出共占 m 列, 其中有 n 位小数, 如数值宽度小于 m 左端补 0.
   %-m.nf: 输出共占 n 列, 其中有 n 位小数, 如数值宽度小于 m 右端补空格.

2. %e 格式: 以指数形式输出实数. 数字部分输出 6 位小数, 指数部分占 5 位或 4 位.
   %m.ne 和%-m.ne:m,n 和”-” 字符含义与前相同. 此处 n 指数据的数字部分的小数位数,m 表示整个输出数据所占的宽度.

3. %g 格式: 自动选 f 格式或 e 格式中较短的一种输出, 且不输出无意义的零.

1.6.5 其他

如果想输出字符”%”, 则应该在“格式控制”字符串中用连续两个%
对于 m.n 的格式还可以用如下方法表示:printf(”%*.*s
n”, m, n, str)
前边的 * 定义的是总的宽度, 后边的定义的是输出的个数. 分别对应外面的参数 m 和 n. 这种方法的好处是可以在语句之外对参数 m 和 n 赋值, 从而控制输出格式.

Chapter 2

数学

2.1 高精

```
const LL BASE = 1e9;

struct BigInteger {
    vector<LL> v; //v中倒序存放数字
    BigInteger () {}
    BigInteger (LL x) { Init(x); }

    void Init(LL x) {
        v.clear();
        if (x == 0) {
            v.push_back(0);
            return ;
        }
        while (x) {
            v.push_back(x % BASE);
            x /= BASE;
        }
    }

    BigInteger operator + (const BigInteger &b) const {
        LL m = max(v.size(), b.v.size()) + 1;
        BigInteger c;
        c.v.resize(m);//全0，注意不能c(m)这样写，因为这样是把c初始化成m
        LL st = 0, tmp, aa, bb;

        REP(i, m) {
            if (i >= (int) v.size()) aa = 0;
            else aa = v[i];
            if (i >= (int) b.v.size()) bb = 0;
            else bb = b.v[i];
            tmp = aa + bb + st;
            if (tmp >= BASE) st = 1, tmp -= BASE;
            else st = 0;
            c.v[i] = tmp;
        }
        while (c.v.size() > 1 && c.v.back() == 0) c.v.pop_back();
        return c;
    }

    BigInteger operator * (const BigInteger &b) const {
        BigInteger c;
        c.v.resize(v.size() + b.v.size()); //max digit a+b 全0
        LL st = 0, tmp;
        REP(i, v.size()) {
            REP(j, b.v.size()) {
                tmp = v[i] * b.v[j] + c.v[i + j] + st; //tmp < base^2
                c.v[i + j] = tmp % BASE;
                st = tmp / BASE; //st < base
            }
            c.v[i + b.v.size()] = st, st = 0;
        }
        while (c.v.size() > 1 && c.v.back() == 0) c.v.pop_back();
        return c;
    }

    BigInteger operator / (const LL &n) const { //n != 0
        if (n == 0) {
            cout << "Error!" << endl;
            return 0;
        }

        BigInteger b;
        b.v.resize(v.size());
        LL st = 0, tmp;
        for (int i = v.size() - 1, j = 0; i >= 0; --i, j++) {
            tmp = st * BASE + v[i];
            b.v[j] = tmp / n;
            st = tmp % n;
        }
        reverse(b.v.begin(), b.v.end());
        while (b.v.size() > 1 && b.v.back() == 0) b.v.pop_back();
```

```
        return b;
    }

    LL operator % (const LL &n) const { //n != 0
        if (n == 0) {            cout << "Error!" << endl;
            return 0;
        }

        LL st = 0, tmp;
        for (int i = v.size() - 1, j = 0; i >= 0; --i, j++) {
            tmp = st * BASE + v[i];
            st = tmp % n;
        }
        return st;
    }
};

void output(const BigInteger &num) {
    cout << num.v.back(); //最前面没有前导0，后面都是有前导零的
    int n = num.v.size() - 1;
    while (n) {
        cout << setw(9) << setfill('0') << num.v[--n]; //这里改变了
    cout对于固定宽度的输出的设置
    }
    cout << endl;
}
```

Listing 2.1: BigInteger.cpp

2.2 位运算

```
//枚举下一个二进制含有N个1的数
int getnext (int x) {
    int b, t, c, r, m;
    b = x & -x;
    t = x + b;
    c = t ^ x;
    m = (c >> 2) / b;
    r = t | m;
    return r;
}

//枚举子集 max --> min
for (int sub = all; sub; sub = all & (sub - 1))
```

Listing 2.2: Bitset.cpp

2.3 归并排序

```
int a[N], b[N]; //b[N] -> tmp a[N] -> data
//Inv1 < Inv2 >

LL Merge_Sort(int l,int r) {
    if(l >= r) return 0;
    int mid = l + r >> 1;
    LL Inv1 = 0, Inv2 = 0;
    if(l <= mid) Inv2 += Merge_Sort(l, mid);
    if(r > mid) Inv2 += Merge_Sort(mid + 1, r);

    for (int i = l, j = mid + 1, k = l; k <= r; k++) {
        if (j > r || i <= mid && a[i] < a[j]) {
            b[k] = a[i++];
            Inv1 += r - j + 1;
        }
        else if (j > r || i <= mid && a[i] == a[j]) {
            b[k] = a[i++];
        }
        else {
            b[k] = a[j++];
            Inv2 += mid - i + 1;
        }
    }

    REPP(i, l, r) a[i] = b[i];
    return Inv2;
}
```

Listing 2.3: MergeSort.cpp

2.4 中国剩余定理

```
LL a[N], mod[N]; //x == a[i] (mod mod[i]), i = 0, 1, ... , n - 1

void exgcd( LL a, LL b, LL &d, LL &x, LL &y){
    if (!b) d = a, x = 1, y = 0;
    else exgcd(b, a % b, d, y, x), y -= (a / b) * x;
```

```
}

LL china (LL n, LL *a, LL *mod){// i = 0, 1, ... , n - 1
    LL re = 0, M = 1;
    REP(i, n) M *= mod[i];
    REP(i, n) {
        LL d, x, y;
        exgcd(M / mod[i], mod[i], d, x, y);        if (x < 0) x += M;
        re = (re + M / mod[i] * x * a[i]) % M; //be careful overflow
    }
    return (re + M) % M;
}
```

Listing 2.4: China.cpp

2.5 矩阵快速幂

```
struct Matrix {
    int a[N][N];

    Matrix (bool ident = 0) {
        MST(a, 0);
        if (ident) {
            REP(i, N) a[i][i] = 1;
        }
    }

    int* operator [] (int i){
        return a[i];
    }

    const int* operator [] (int i) const {
        return a[i];
    }
};

Matrix operator * (const Matrix &a, const Matrix &b) {
    Matrix c;
    REP(i, N) REP(j, N) REP(k, N) {
        add(c[i][j], 1LL * a[i][k] * b[k][j] % MO);
    }
    return c;
}
```

Listing 2.5: Matrix.cpp

2.6 线性求逆元

```
//O(n)
inv[0] = 0, inv[1] = 1;
REPP(i, 2, N - 1) inv[i] = 1LL * (MO - MO / i) * inv[MO % i] % MO;
```

Listing 2.6: InvP.cpp

2.7 Fibonacci 进制数最简表示

```
priority_queue<int> q;

void go(int id) {
    cin >> s[id];
    int len = s[id].size();
    reverse(s[id].begin(), s[id].end());
    REP(i, len) {
        a[id][i] = s[id][i] - '0';
    }
    REP(i, len) if (a[id][i] && a[id][i + 1]) q.push(i);
    while (!q.empty()) {
        int k = q.top(); q.pop();
        if (a[id][k] == 0 || a[id][k + 1] == 0) {
            continue;
        }
        a[id][k] = a[id][k + 1] = 0;
        a[id][k + 2] = 1;
        if (a[id][k + 3]) q.push(k + 2);
    }
}
```

Listing 2.7: Fibonacci.cc

2.8 素数 PE10

时间复杂度:$O(N^{\frac{3}{4}})$
记 dp[i][j] 为 1..j 中除掉以包含前 i 个素数为因子的合数后的乘积，cnt[i][j] 为 1..j 中除掉以包含前 i 个素数为因子的合数后的个数，则有

$dp[i][j] = dp[i-1][j]/(dp[i-1][j/prime[i]]/dp[i-1][prime[i-1]])$
$/prime[i]^{cnt[i-1][j/prime[i]]-cnt[i-1][prime[i-1]]}$
$cnt[i][j] = cnt[i-1][j]-(cnt[i-1][j/prime[i]]-cnt[i-1][prime[i-1]])$
注意到第一维只要枚举所有小于 $\sqrt{N}$ 的素数，第二维由于转移中只有除法，故只要 n/i 形式的数字即可. 这样的数规模是 $2\sqrt{N}$.

```
unordered_map<int, int> mp;
int dp[2][N], cnt[2][N];
int fac[M], inv[M * M], MO, mod, phi, n;
int prefix[N];
int p[N], vis[N], tot;

void prime() {
    tot = 1;
    for (int i = 2; i < N; i++) {
        if (!vis[i]) p[tot++] = i;
        for (int j = 1; j < tot && i * p[j] < N; j++) {
            vis[i * p[j]] = p[j];
            if (i % p[j] == 0) break;
        }
    }
}

int pow_mod(int a, int b) {
    int ans = 1;
    while (b) {
        if (b & 1) ans = 1LL * ans * a % mod;
        b >>= 1;
        a = 1LL * a * a % mod;
    }
    return ans;
}

void init(int tot) {
    fac[0] = 1;
    REPP(i, 1, MO - 1) {
        fac[i] = 1LL * i * fac[i - 1] % mod;
    }
    inv[0] = 0, inv[1] = 1;
    REPP(i, 2, mod - 1) {
        if (i % MO == 0) {
            inv[i] = 0;
            continue;
        }
        inv[i] = mod - 1LL * mod / i * inv[mod % i] % mod;
    }
    prefix[0] = 1;
    REPP(i, 1, tot) {
        if (p[i] == MO) {
            prefix[i] = prefix[i - 1];
        }
        else {
            prefix[i] = 1LL * prefix[i - 1] * p[i] % mod;
        }
    }
}

int calc(int x) {
    if (x == 0) return 1;
    int k = x / MO, b = x % MO;
    int res = calc(k);
    int p = pow_mod(fac[MO - 1], k);
    p = 1LL * p * fac[b] % mod;
    p = 1LL * p * res % mod;
    return p;
}

int Div(int a, int b) {
    //return 1LL * a * pow_mod(b, phi - 1) % mod;
    return 1LL * a * inv[b] % mod;
}

int main() {
#ifdef HOME
    freopen("4.in", "r", stdin);
#endif

    prime();
    int t, ca = 1;
    scanf("%d", &t);
    while (t--) {
        scanf("%d%d", &n, &MO);
        mod = MO * MO, phi = MO * (MO - 1);
        if (n < MO) {
            int ans = 1;
            REPP(i, 1, tot) {
                if (p[i] > n) break;
                ans = 1LL * ans * p[i] % mod;
            }
            printf("Case #%d: %d\n", ca++, ans / MO % MO);
            continue;
        }
```

```
        int rt = int(sqrt(n + 0.5));
        int tot = upper_bound(p + 1, p + ::tot, rt) - p - 1;
        init(tot);
        mp.clear();
        vector<int> tmp;        int now = 0;
        for (int i = 1; i <= n; i++) {
            int k = n / i;
            tmp.push_back(k);
            dp[0][now] = calc(k);
            cnt[0][now] = k;
            mp[k] = now++;
            i = n / k;
        }
        int cur = 0;
        REPP(i, 1, tot) {
            REP(j, now) {
                int k = tmp[j];
                if (p[i] * p[i] > k) {
                    dp[cur ^ 1][j] = dp[cur][j];
                    cnt[cur ^ 1][j] = cnt[cur][j];
                    continue;
                }
                int id = mp[k / p[i]];
                dp[cur ^ 1][j] = Div(dp[cur][j], Div(dp[cur][id],
    prefix[i - 1]));
                int &ans = dp[cur ^ 1][j];
                if (p[i] != MO) {
                    int cc = pow_mod(p[i], cnt[cur][id] - i) % mod;
                    ans = Div(ans, cc);
                }
                cnt[cur ^ 1][j] = cnt[cur][j] - (cnt[cur][id] - i);
            }
            cur ^= 1;
        }
        printf("Case #%d: %d\n", ca++, dp[cur][mp[n]] % MO);
    }
    return 0;
}
```

Listing 2.8: SumPrime.cpp

2.9 随机素数测试

```
LL mul_mod(LL a, LL b, LL MO) {
    a %= MO;
    LL re = 0;//not 1
    while (b) {
        if (b & 1) add(re, a, MO);
        b >>= 1;
        add(a, a, MO);
    }
    return re;
}

LL pow_mod(LL a, LL b, LL MO) {
    LL re = 1;
    while (b) {
        if (b & 1) re = mul_mod(re, a, MO);
        b >>= 1;
        a = mul_mod(a, a, MO);
    }
    return re;
}

bool test(LL n, LL b) {
    LL m = n - 1, cnt = 0;
    while (~m & 1) {
        m >>= 1;
        cnt++;
    }
    LL ans = pow_mod(b, m, n);
    if (ans == 1 || ans == n - 1) return 1;
    cnt--;
    while (cnt >= 0) {
        ans = mul_mod(ans, ans, n);
        if (ans == n - 1) return 1;
        cnt--;
    }
    return 0;
}

const int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};

bool isPrime(LL n) {
    if (n < 2) return 0;
    if (n < 4) return 1;
    if (n == 3215031751LL) return 0;
    for (int i = 0; i < 12 && BASE[i] < n; ++i) {
        if (!test(n, BASE[i])) return 0;
    }
    return 1;
```

```
}
```

Listing 2.9: MillerRabin.cpp

2.10 随机素数分解

```
bool test(LL n, LL b) {
    LL m = n - 1, cnt = 0;
    while (~m & 1) {
        m >>= 1;
        cnt++;
    }
    LL ans = qp(b, m, n);
    if (ans == 1 || ans == n - 1) return 1;
    cnt--;
    while (cnt >= 0) {
        ans = mul_mod(ans, ans, n);
        if (ans == n - 1) return 1;
        cnt--;
    }
    return 0;
}

const int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};

bool isPrime(LL n) {
    if (n < 2) return 0;
    if (n < 4) return 1;
    if (n == 3215031751LL) return 0;
    for (int i = 0; i < 12 && BASE[i] < n; ++i) {
        if (!test(n, BASE[i])) return 0;
    }
    return 1;
}

LL gcd(LL x, LL y) {return y ? gcd(y, x % y) : x;}

vector<LL> fac;

LL Pollard_Rho(LL n, LL seed) {
    LL x, y, st = 1, ed = 2;
    x = y = rand() % (n - 1) + 1;
    while (1) {
        x = mul_mod(x, x, n);
        add(x, seed, n);
        if (x == y) return n;
        LL d = gcd(abs(x - y), n);
        if (1 < d && d < n) return d;
        st++;
        if (st == ed) {
            y = x;
            ed <<= 1;
        }
    }
}

void factorize(LL n) {
    if (n > 1) {
        if (isPrime(n)) fac.push_back(n);
        else {
            LL d = n;
            while (d >= n) {
                d = Pollard_Rho(n, rand() % (n - 1) + 1);
            }
            factorize(n / d);
            factorize(d);
        }
    }
}
```

Listing 2.10: PollardRho.cpp

2.11 原根 PrimitiveRoot

```
struct Root{
    //p^k, 2*p^k, 2, 4
    int phi, g, p;
    vector<int> fac;

    void getfac(int phi) {
        fac.clear();
        for (int i = 2; i * i <= phi; i ++) {//prime
            if (phi % i == 0) {
                fac.push_back(i);
                while (phi % i == 0) {
                    phi /= i;
                }
            }
        }
        if (phi > 1) fac.push_back(phi);//important
```

```
    }

    int primitive_root(int x) {
        p = x, phi = p - 1, getfac(phi);
        REPP(i, 1, phi) {            bool bad = 0;
            REP(j, fac.size()) {
                if (pow_mod(i, phi / fac[j], p) == 1) {
                    bad = 1;
                    break;
                }
            }
            if (!bad) return g = i;
        }
        return -1;
    }
}go;
```

Listing 2.11: PrimitiveRoot.cpp

2.12 莫比乌斯 Mobius

```
int m[N] = {0, 1}, vis[N], p[N], cnt;

void mobius(int n){
    for (int i = 2; i <= n; ++i){
        if (!vis[i]){
            m[i] = -1;
            p[cnt++] = i;
        }
        for (int j = 0; j < cnt && i * p[j] <= n; ++j){
            if (i % p[j] == 0) {
                vis[i * p[j]] = 1, m[i * p[j]] = 0;
                break;
            }
            else vis[i * p[j]] = 1, m[i * p[j]] = -m[i];
        }
    }
}
```

Listing 2.12: Mobius.cpp

2.13 对数方程 BSGS

```
// a^x == b (mod p)

int BSGS(int a, int b, int p) {
    int m, v, e = 1, i;
    m = (int) sqrt(p + 0.5) + 1;//这里不加1会出错的！
    v = pow_mod(pow_mod(a, m, p), p - 2, p);
    map<int, int> x;
    x[1] = 0;
    for (i = 1; i < m; ++i){
        e = e * a % p;
        if (!x.count(e)) x[e] = i;
    }
    for (i = 0; i < m; ++i){
        if (x.count(b)) return i * m + x[b];
        b = b * v % p;
    }
    return -1; // No solution
}
```

Listing 2.13: BSGS.cpp

2.14 最近点对

```
//double version
struct point{
    double x, y;
}p[N];
//sort all points using cmp at first
inline bool cmp(const point &a, const point &b) {
    if (a.x == b.x) return a.y < b.y;
    return a.x < b.x;
}
inline bool cmpy(const int &a, const int &b) {return p[a].y < p[b].
    y;}
inline double sqr(double x) {return x * x;}
inline double dis(const int &i, const int &j) {
    return sqrt(sqr(p[i].x - p[j].x) + sqr(p[i].y - p[j].y));
}//square of dis

int tmp[N];
//return (dis * dis)
double Closest_Pair(int left, int right) {
    double d = INF; // square of ans
    if (left == right) return d;
    if (left + 1 == right) return dis(left, right);
```

```
    int mid = (left + right) >> 1;
    double  dl = Closest_Pair(left, mid);
    double dr = Closest_Pair(mid + 1, right);
    d = min(dl, dr);
    int cnt = 0;
    REPP(i, left, right) {        if (abs(p[mid].x - p[i].x) <= d)
    {
            tmp[cnt++] = i;
        }
    }
    sort(tmp, tmp + cnt, cmpy);
    REP(i, cnt) {
        for (int j = i + 1; j < cnt && p[tmp[j]].y - p[tmp[i]].y <
    d; ++j) {
            double dd = dis(tmp[i], tmp[j]); //tmp[i] not i
            if (d > dd) d = dd;
        }
    }
    return d;
}
```

Listing 2.14: ClosestPair.cpp

2.15 极角排序

```
struct Point {
    long long x, y;

    int dim() const {
        return x < 0 || (x == 0 && y < 0);
    }
};

long long det(const Point &a, const Point &b) {
    return a.x * b.y - a.y * b.x;
}

bool byAng(const Point &a, const Point &b) {
    if (a.dim() == b.dim()) {
        return det(a, b) < 0;
    }
    else {
        return a.dim() < b.dim();
    }
}
```

Listing 2.15: SortbyAngle.cpp

2.16 凸包

```
struct point{
    int x, y, id;
    point (int x = 0, int y = 0, int id = 0) :x(x), y(y), id(id) {}

    point operator - (const point &a) const {
        point res;
        res.x = x - a.x, res.y = y - a.y;
        return res;
    }
}p[N];

long long det(const point &a, const point &b) {
    return 1LL * a.x * b.y - 1LL * a.y * b.x;
}

bool cmp(const point &a, const point &b) {
    return a.x < b.x || (a.x == b.x && a.y < b.y);
}

vector<point> ConvexHull(vector<point> v) {
    sort(v.begin(), v.end(), cmp);
    int n = v.size(), m;
    vector<point> C;
    for (int i = 0; i < n; i++) {
        while ((m = C.size()) > 1 && det(C[m - 1] - C[m - 2], v[i]
    - C[m - 2]) < EPS) C.pop_back();
        C.push_back(v[i]);
    }
    int k = C.size();
    for (int i = n - 2; i >= 0; i--) {
        while ((m = C.size()) > k && det(C[m - 1] - C[m - 2], v[i]
    - C[m - 2]) < EPS) C.pop_back();
        C.push_back(v[i]);
    }
    if (n > 1) C.pop_back();
    return C;
}
```

Listing 2.16: ConvexHull.cpp

2.17 三维向量旋转

```
const int N = 4;
const double pi = acos(-1);

Matrix Init_scale(double a, double b, double c) {
    Matrix ans;
    ans[0][0] = a, ans[1][1] = b, ans[2][2] = c, ans[3][3] = 1.0;
    return ans;
}

Matrix Init_trans(double a, double b, double c) {
    Matrix ans(1);
    ans[3][0] = a, ans[3][1] = b, ans[3][2] = c;
    return ans;
}

double sqr(double x) { return x * x; }

Matrix Init_rotate(double a, double b, double c, double d) {
    d = d * pi / 180.0;//0-360
    double norm = sqrt(sqr(a) + sqr(b) +sqr(c));
    double x = a / norm, y = b / norm, z = c / norm, cosd = cos(d),
     sind = sin(d);
    Matrix ans;
    ans[0][0] = cosd + (1 - cosd) * sqr(x);
    ans[1][0] = (1 - cosd) * x * y - sind * z;
    ans[2][0] = (1 - cosd) * x * z + sind * y;
    ans[0][1] = (1 - cosd) * y * x + sind * z;
    ans[1][1] = cosd + (1 - cosd) * sqr(y);
    ans[2][1] = (1 - cosd) * y * z - sind * x;
    ans[0][2] = (1 - cosd) * z * x - sind * y;
    ans[1][2] = (1 - cosd) * z * y + sind * x;
    ans[2][2] = cosd + (1 - cosd) * sqr(z);
    ans[3][3] = 1.0;
    return ans;
}
```

Listing 2.17: Vector3D.cpp

2.18 高斯消元

```
//运行结束后a.v[i][n]是第i个未知数的值 lrj Page154
//已经加了对精度的优化
void gauss(Matrix& a, int n){
    int i, j, k, r;
    for (i = 0; i < n; ++i){
        r = i;
        for (j = i + 1; j < n; ++j)
            if (fabs(a.v[j][i]) > fabs(a.v[r][i])) r = j;
        if (r != i) for (j = 0; j <= n; ++j) swap(a.v[r][j], a.v[i
    ][j]);

        for (j = n; j >= i; --j)
            for (k = i + 1; k < n; ++k)
                a.v[k][j] -= a.v[k][i] / a.v[i][i] * a.v[i][j];
    }
    for (i = n - 1; i >= 0; --i){
        for (j = i + 1; j < n; ++j){
            a.v[i][n] -= a.v[j][n] * a.v[i][j];
        }
        a.v[i][n] /= a.v[i][i];
    }
}
```

Listing 2.18: Gauss.cpp

Chapter 3

图论

3.1 一般图

3.1.1 强联通分量 Kosaraju

```cpp
//SCC O(N+M)
vector<int> mp[N], g[N], s; // mp 原图邻接表 g 原图转置邻接表
    原图节点从0开始标号 栈
int vis[N], sccno[N], scc_cnt; //i 所属于的SCC的标号 1--> scc_cnt ,
    SCC数量

void dfs1(int x) {
    vis[x] = 1;
    REP(i, mp[x].size()) if (!vis[mp[x][i]]) dfs1(mp[x][i]);
    s.push_back(x);
}

void dfs2(int x) {
    sccno[x] = scc_cnt;
    REP(i, g[x].size()) if (!sccno[g[x][i]]) dfs2(g[x][i]);
}

void find_scc(int n) {
    scc_cnt = 0;
    s.clear();
    MST(vis, 0), MST(sccno, 0);
    REP(i, n) if (!vis[i]) dfs1(i);
    for (int i = n - 1; i >= 0; --i) {
        if (!sccno[s[i]]) {
            scc_cnt++;
            dfs2(s[i]);
        }
    }
}
```

Listing 3.1: Kosaraju.cpp

3.1.2 树同构重心

```cpp
//包括了求树的重心，以及树的最小表示（有根树，
    任选的时候我们可以以重心为根，然后就可以判同构了）

struct tree{
    int son[N], dm[N];//儿子节点数，
    断开该节点后子连通块的节点数最大值
    int fi[N], en[N << 1], ne[N << 1], tot;
    int n;//节点数

    void Init(int x) {//初始化
        MST(son, 0), MST(dm, 0);
        MST(fi, 0), tot = 0;
        n = x;
    }

    void add(int x, int y) {
        ne[++tot] = fi[x], fi[x] = tot, en[tot] = y;
    }

    void dfs1(int x, int p) {//树DP
        for (int go = fi[x]; go; go = ne[go]) {
            if (en[go] != p) {
                dfs1(en[go], x);
                son[x] += son[en[go]];
                dm[x] = max(dm[x], son[en[go]]);
            }
        }
        son[x]++;
    }

    void dfs2(int x, int p) {//树DP
        for (int go = fi[x]; go; go = ne[go]) {
            if (en[go] != p) {
                int y = en[go];
                dm[y] = max(n - son[y], dm[y]);
                dfs2(y, x);
            }
```

```
        }
    }

    void getMess(vector<int>& w) {//得到重心，最多两个
        dfs1(1, 0);         dfs2(1, 0);
        int mi = INF;
        REPP(i, 1, n) {
            if (dm[i] < mi) {
                mi = dm[i];
                w.clear();
                w.push_back(i);
            }
            else if (dm[i] == mi) w.push_back(i);
        }
    }

    string getStr(int x, int p) {//最小表示的括号序列
        int good = 0;
        vector<string> tmp;
        for (int go = fi[x]; go; go = ne[go]) {
            if (en[go] != p) {
                good = 1;
                tmp.push_back(getStr(en[go], x));
            }
        }
        if (!good) return (string) "()";
        else {
            sort(ALL(tmp));//排序
            string ans;
            REP(i, SZ(tmp)) ans += tmp[i];
            ans = '(' + ans + ')';
            return ans;
        }
    }
}t[2];
```

Listing 3.2: Tree.cpp

3.2 二分图

3.2.1 最大匹配

```
int match[N], vis[N];
//vis[x] 右边x有没有被访问过 match[x] 右边x所匹配的边

bool dfs(int x) {//x为来自左边的点
    for (int go = fi[x]; go; go = ne[go]) if (!vis[en[go]]) {
        vis[en[go]] = 1;
        if (!match[en[go]] || dfs(match[en[go]])) {
            match[en[go]] = x;
            return true;
        }
    }
    return false;
}

int Hungarian(){
    int ans = 0;
    MST(match, 0);
    REPP(i, 1, n) {
        MST(vis);
        if (dfs(i)) ans++;
    }
    return ans;
}
```

Listing 3.3: Hungarian.cpp

3.3 网络流

3.3.1 最大流 Dinic

时间复杂度:$O(N^2 * M)$
代码易错点: 无向图存边表的数组要开大两倍.

```
template<int N, int M, typename Type>
struct MaxFlow{
    int edge, source, sink;
    int lvl[N], cur[N];
    int fi[N], ne[M << 1], en[M << 1];
    Type cap[M << 1];
    const Type INF = 0x3f3f3f3f;

    void init(int S, int T) {
        source = S, sink = T;
        edge = 1, MST(fi, 0);
    }

    void _add(int x, int y, Type z) {
        ne[++edge] = fi[x]; fi[x] = edge; en[edge] = y; cap[edge] =
     z;
```

```
    }

    void add(int x, int y, Type z) {
        _add(x, y, z);
        _add(y, x, 0);    }

    bool bfs() {
        queue<int> q;
        MST(lvl, 0);
        q.push(source), lvl[source] = 1;
        while(q.size()) {
            int x = q.front(); q.pop();
            for (int go = fi[x]; go; go = ne[go]) if (cap[go] > 0
    && !lvl[en[go]]) {
                int y = en[go];
                lvl[y] = lvl[x] + 1;
                q.push(y);
            }
        }
        return lvl[sink];
    }

    Type dfs(Type x, Type flow) {
        if (x == sink || flow == 0) {
            return flow;
        }
        Type ans = 0, tmp = 0;
        for (int &go = cur[x]; go; go = ne[go]) if (cap[go] > 0) {
            int y = en[go];
            if (lvl[y] == lvl[x] + 1 && (tmp = dfs(y, min(flow, (
    Type) cap[go]))) > 0) {
                ans += tmp, flow -= tmp;
                cap[go] -= tmp, cap[go ^ 1] += tmp;
                if (flow == 0) {
                    return ans;
                }
            }
        }
        return ans;
    }

    Type dinic() {
        Type ans = 0;
        while (bfs()) {
            memcpy(cur, fi, sizeof(fi));
            ans += dfs(source, INF);
        }
        return ans;
    }
};
MaxFlow<100005, 100005, int> flow;
```

Listing 3.4: Dinic.cpp

3.3.2 费用流

时间复杂度:$O(flow * O(KM))$
代码易错点:SPFA 不要写错即可.

```
const int N = 100005;
const int M = 100005;
const int INF = 0x3f3f3f3f;

struct MinCostFlow{
    int pre[N], dp[N];
    int fi[N], ne[M << 2], en[M << 2], cap[M << 2], cost[M << 2],
    edge;
    bool vis[N];
    int source, sink;

    void init(int S, int T) {
        source = S, sink = T;
        MST(fi, 0), edge = 1;
    }

    void _add(int x, int y, int z, int w) {
        ne[++edge] = fi[x], fi[x] = edge, en[edge] = y, cap[edge] =
     z, cost[edge] = w;
    }

    void add(int x, int y, int z, int w) {
        _add(x, y, z, w);
        _add(y, x, 0, -w);
    }

    bool spfa() {
        MST(dp, 0x3f);
        queue<int> q;
        q.push(source), vis[source] = 1, dp[source] = pre[source] =
     0;

        while (q.size()) {
```

```
            int x = q.front(); q.pop();
            for (int go = fi[x]; go; go = ne[go]) if (cap[go] > 0)
    {
                int y = en[go];
                if (dp[y] > dp[x] + cost[go]) {
    dp[y] = dp[x] + cost[go];
                    pre[y] = go;
                    if (!vis[y]) {
                        vis[y] = 1;
                        q.push(y);
                    }
                }
            }
            vis[x] = 0;
        }
        return dp[sink] != INF;
    }

    pair<int, int> mincost() {
        int cost = 0, flow = 0;
        while (spfa()) {
            int tmp = INF;
            for (int go = pre[sink]; go; go = pre[en[go ^ 1]]) {
                tmp = min(tmp, cap[go]);
            }
            for (int go = pre[sink]; go; go = pre[en[go ^ 1]]) {
                cap[go] -= tmp;
                cap[go ^ 1] += tmp;
            }
            cost += tmp * dp[sink];
            flow += tmp;
        }
        return make_pair(cost, flow);
    }
}flow;
```

Listing 3.5: CostFlow.cpp

3.3.3 全图最小割 StoerWagner

时间复杂度:$O(N^3)$
代码易错点: 无向图, 非负.

```
//O(N^3) 无向图 无负权
int vis[N], wet[N];//weight all edge weight connect i from set A
int combine[N]; //has been merged or not
int mp[N][N]; //map nodes 0based (starts from 0)

int S, T, minCut, n;

void search() {
    int Max, tmp;
    MST(vis, 0), MST(wet, 0);
    S = T = -1;
    REP(i, n) {
        Max = -INF;
        REP(j, n) {
            if (!combine[j] && !vis[j] && wet[j] > Max) {
                tmp = j;
                Max = wet[j];
            }
        }
        if (T == tmp) return ;
        S = T, T = tmp;
        minCut = Max;
        vis[tmp] = 1;
        REP(j, n) {
            if (!combine[j] && !vis[j]) {
                wet[j] += mp[tmp][j];
            }
        }
    }
}

int Stoer_Wagner() {
    MST(combine, 0);
    int ans = INF;
    REP(i, n - 1) {
        search();
        if (minCut < ans) ans = minCut;
        if (ans == 0) return 0;
        combine[T] = 1;
        REP(j, n) if (!combine[j]) {
            mp[S][j] += mp[T][j];
            mp[j][S] += mp[j][T];
        }
    }
    return ans;
}
```

Listing 3.6: StoerWagner.cpp

Chapter 4

字符串

4.1 字符串最小表示

字符串最小表示的算法默认的可以得到最小的那个最小表示的起点, 如果最小表示唯一, 不会有什么问题, 如果不唯一, 那么说明整个串是个循环串, 这时候代码中 c=len, b-a 就是循环节长度.

```cpp
//获得字符串s的最小表示,会将字符串扩展成2倍长.
//返回值是最小表示的起始位置在原串中的特殊位置
int MSR(char *s, int type) {//type 0 最前的位置, 1 最后的位置
    int a = 0, b = 1, c = 0;
    int n = strlen(s);
    REP(i, n) s[i + n] = s[i];
    s[n << 1] = 0;
    while (a < n && b < n && c < n) {
        int tmp = s[a + c] - s[b + c];
        if (!tmp) c++;
        else {
            if (tmp > 0) a += c + 1; //最大表示这里改一下
            else b += c + 1;
            if (a == b) b++;
            c = 0;
        }
    }
    if (type) {
        if (c == n) {
            int d = b - a;
            a %= d;
            a = d - a;
            return n - a;
        }
    }
    return a;
}
```

Listing 4.1: MSR.cpp

4.2 表达式求值

```cpp
char s[N];
struct Exp{
    bool error;
    int tot, top, num[N];
    char op[N];

    void init() {
        error = false;
        tot = 0, top = 1;
        op[1] = '(';
    }

    bool priority(char a, char b) {
        if (b == '+' || b == '-') {
            return a != '(';
        }
        return a == '*' || a == '/';
    }

    int calc(char c, int a, int b) {
        if (c == '+') return a + b;
        if (c == '-') return a - b;
        if (c == '*') return a * b;
        if (c == '/' && b != 0) return a / b;
        error = 1;
        return 0;
    }

    int solve(char *s, int len) {//len = strlen(s)
        s[len++] = ')';
        REP(i, len) {
            if (s[i] == '(') op[++top] = s[i];
            else if (s[i] == ')') {
                while (top > 0 && op[top] != '(') {
                    num[tot - 1] = calc(op[top], num[tot - 1], num[tot]);
                    tot--, top--;
```

```
                }
                top--;
            }
            else if (s[i] == '-' && (i == 0 || s[i - 1] == '(')) {
    //处理负号
                num[++tot] = 0;                 op[++top] = '-';
            }
            else if (isdigit(s[i])) {
                int t = s[i] - '0';
                for (i++; isdigit(s[i]); i++) {
                    t = t * 10 + s[i] - '0';
                }
                num[++tot] = t, i--;
            }
            else {
                while (top > 0 && priority(op[top], s[i])) {
                    num[tot - 1] = calc(op[top], num[tot - 1], num[
    tot]);
                    tot--, top--;
                }
                op[++top] = s[i];
            }
        }
        return num[1];
    }
}expre;
```

Listing 4.2: Expression.cpp

4.3 Hash

```
const int X1 = 4243;
const int MOD1 = 1e9 + 7;
const int X2 = 10007;
const int MOD2 = 42424243;
const int LEN = 3e5 + 5;

struct hash {
    int a, b;

    hash() {}
    hash(int a) : a(a), b(a) {}
    hash(int a, int b) : a(a), b(b) {}
};

bool operator < (const hash& a, const hash& b) {
    if (a.a != b.a) {
        return a.a < b.a;
    }
    return a.b < b.b;
}

bool operator == (const hash& a, const hash& b) {
    return a.a == b.a && a.b == b.b;
}

hash operator * (const hash& a, const hash& b) {
    return hash((1LL * a.a * b.a) % MOD1, (1LL * a.b * b.b) % MOD2)
    ;
}

hash operator + (const hash& a, const hash& b) {
    hash c(a.a + b.a, a.b + b.b);
    if (c.a >= MOD1) {
        c.a -= MOD1;
    }
    if (c.b >= MOD2) {
        c.b -= MOD2;
    }
    return c;
}

const hash X(X1, X2);

inline void init() {
    pw[0] = hash(1);
    for (int i = 1; i < LEN; ++i) {
        pw[i] = pw[i - 1] * X;
    }
}
```

Listing 4.3: Hash.cpp

4.4 AC 自动机

```
struct AC_Automation{
    int tr[N * LEN][Z], fail[N * LEN], val[N * LEN];
    int node;

    int New_Node() {
```

```
        mem(tr[node]);
        fail[node] = val[node] = 0;
        return node++;
    }
    void Init() {         node = 0;
        New_Node();
    }

    void insert(char *s, int id) {
        int pos = 0, x, now = 0;
        while (s[pos]) {
            x = str.find(s[pos++]);
            if (!tr[now][x]) {
                tr[now][x] = New_Node();
                val[tr[now][x]] = val[now];
            }
            now = tr[now][x];
        }
        val[now] |= 1 << id;
    }

    void getfail() {
        int x, y;
        queue<int> q;
        REP(i, Z) {
            x = tr[0][i];
            if (x) {
                fail[x] = 0;
                q.push(x);
            }
        }

        while (!q.empty()) {
            x = q.front(); q.pop();
            REP(i, Z) {
                if (!tr[x][i]) {
                    tr[x][i] = tr[fail[x]][i];
                    continue;
                }
                y = tr[x][i];
                q.push(y);
                fail[y] = tr[fail[x]][i];
            }
        }
    }
}AC;
```

Listing 4.4: ACAutomation.cpp

4.5 后缀数组

```
template<typename T = int>
struct SuffixArray {
    int s[N], sa[N], a[N], b[N], c[N], n;
    int rank[N], height[N];
    //sa[i] 0base sa[0] = n 后缀数组排名i的后缀是第几个后缀
    //rank[i] 0base 第i个后缀在后缀数组里面的排名
    //height[i] sa[i]和sa[i-1]的最长公共前缀，height[0] = 0, height[1] = 0;
    //结构体里的n比实际串长大1

    void build(const T *str, int x, int m = 256) {//x 是未处理的str的长度
        n = x;
        REP(i, n) s[i] = str[i];
        s[n++] = int('$'), s[n] = 0;
        getSA(m), getHeight();
    }

    void getSA(int m) {//m 字符集大小
        int *x = a, *y = b;
        fill(c, c + m, 0);
        REP(i, n) c[x[i] = s[i]]++;
        REPP(i, 1, m - 1) c[i] += c[i - 1];
        for (int i = n - 1; i >= 0; --i) sa[--c[x[i]]] = i;
        for(int k = 1; k <= n; k <<= 1) {
            int p = 0;
            for (int i = n - k; i < n; ++i) y[p++] = i;
            REP(i, n) if (sa[i] >= k) y[p++] = sa[i] - k;
            REP(i, m) c[i] = 0;
            REP(i, n) c[x[y[i]]]++;
            REPP(i, 1, m - 1) c[i] += c[i - 1];
            for (int i = n - 1; i >= 0; --i) sa[--c[x[y[i]]]] = y[i];
            swap(x, y);
            p = 1; x[sa[0]] = 0;
            REPP(i, 1, n - 1) {
                x[sa[i]] = y[sa[i - 1]] == y[sa[i]] && y[sa[i - 1] + k] == y[sa[i] + k] ? p - 1 : p++;
            }
```

```
            if (p >= n) break;
            m = p;
        }
    }

    void getHeight(){        REP(i, n) rank[sa[i]] = i;
        int k = 0;
        REP(i, n) {
            if (k) k--;
            int j = sa[rank[i] - 1];
            while (s[i + k] == s[j + k]) k++;
            height[rank[i]] = k;
        }
    }

    void print(){
        REP(i, n) cout << sa[i] << " \n"[i == n - 1];
        REP(i, n) cout << rank[i] << " \n"[i == n - 1];
        REP(i, n) cout << height[i] << " \n"[i == n - 1];
    }
};
SuffixArray<char> SA;

// DC3算法构造后缀数组 O(n)
// sa[i] 表示排名为 i 的后缀，rank[i]表示第i个后缀的排名，height[i]表示 sa[i] 与 sa[i - 1] 最长公共前缀
// 求出的rank, height, sa数组和倍增算法求得的完全一样
// 支持对于任意类型子序列进行求解，注意原序列中所有数据为正，原序列范围较大需要离散化

#define F(x) ((x) / 3 + ((x) % 3 == 1 ? 0 : ty))
#define G(x) ((x) < ty ? (x) * 3 + 1 : ((x) - ty) * 3 + 2)

template<typename T = char>
struct SuffixArray {
    int str[N * 3], sa[N * 3], rank[N], height[N], n;
    int wa[N], wb[N], wv[N], ws[N];

    int &operator [](int k) { return sa[k]; }

    int size() const { return n; }

    bool equal(const int *r, int a, int b) const {
        return r[a] == r[b] && r[a + 1] == r[b + 1] && r[a + 2] == r[b + 2];
    }

    bool cmp(const int *r, int a, int b, int d) const {
        if (d == 1) return (r[a] < r[b]) || (r[a] == r[b] && wv[a + 1] < wv[b + 1]);
        return (r[a] < r[b]) || (r[a] == r[b] && cmp(r, a + 1, b + 1, 1));
    }

    void rsort(const int *r, const int *a, int *b, int n, int m) {
        int i;
        fill(ws, ws + m, 0);
        for (i = 0; i < n; ++i) ++ws[wv[i] = r[a[i]]];
        for (i = 1; i < m; ++i) ws[i] += ws[i - 1];
        for (i = n - 1; ~i; --i) b[--ws[wv[i]]] = a[i];
    }

    void dc3(int *r, int *sa, int n, int m) {
        int i, j, k, *rn = r + n, *san = sa + n, tx = 0, ty = (n + 1) / 3, tz = 0;

        r[n] = r[n + 1] = 0;
        for (i = 0; i < n; ++i) {
            if (i % 3) wa[tz++] = i;
        }
        rsort(r + 2, wa, wb, tz, m);
        rsort(r + 1, wb, wa, tz, m);
        rsort(r, wa, wb, tz, m);
        for (rn[F(wb[0])] = 0, k = i = 1; i < tz; ++i) {
            rn[F(wb[i])] = equal(r, wb[i - 1], wb[i]) ? k - 1 : k++;
        }
        if (k < tz) dc3(rn, san, tz, k);
        else {
            for (i = 0; i < tz; ++i) san[rn[i]] = i;
        }
        for (i = 0; i < tz; ++i) {
            if (san[i] < ty) wb[tx++] = san[i] * 3;
        }
        if (n % 3 == 1) wb[tx++] = n - 1;
        rsort(r, wb, wa, tx, m);
        for (i = 0; i < tz; ++i) wv[wb[i] = G(san[i])] = i;
        for (i = j = k = 0; i < tx && j < tz; ++k) {
            sa[k] = cmp(r, wa[i], wb[j], wb[j] % 3) ? wa[i++] : wb[j++];
        }
        for ( ; i < tx; ++i) sa[k++] = wa[i];
```

```
        for ( ; j < tz; ++j) sa[k++] = wb[j];
    }

    void build(const T *s, int x, int m = 256) {//x
    是原串长度不做任何修改 x==1没有问题
        int i;        for (i = 0; i < x; ++i) str[i] = (int)s[i];
        str[x] = 0; n = x + 1;
        dc3(str, sa, n, m);
        getHeight();
    }

    void getHeight() {
        int i, j, k = 0;
        for (i = 0; i < n; ++i) rank[sa[i]] = i;
        for (i = 0; i < n; height[rank[i++]] = k) {
            for(k ? --k : 0, j = sa[rank[i] - 1]; str[i + k] == str
    [j + k]; ++k);
        }
    }

    void print(){
        REP(i, n) cout << sa[i] << " \n"[i == n - 1];
        REP(i, n) cout << rank[i] << " \n"[i == n - 1];
        REP(i, n) cout << height[i] << " \n"[i == n - 1];
    }
};
SuffixArray<char> SA;
```

Listing 4.5: SA.cpp

4.6 Manacher

```
struct Manacher{
    int p[N << 1], n; //p 回文向两边扩展的长度，p[i]-1
    就是原串对应位置的最长回文长度
    char r[N << 1]; //为了避免边界问题，r是原串的基础上间隔加上'#',
     另外开始的时候加一个'$'. 所以s[i] = r[2 * (i + 1)]

    void init(char *s) {
        n = 0, r[n++] = '$';
        while (*s) {
            r[n++] = '#', r[n++] = *s++;
        }
        r[n++] = '#', r[n] = 0;
    }

    void get() {
        int id = 1, mx = 2;
        p[0] = p[1] = 1;
        REPP(i, 2, n - 1) {
            p[i] = mx > i ? min(p[2 * id - i], mx - i) : 1;
            while (r[i + p[i]] == r[i - p[i]]) p[i]++;
            if (i + p[i] > mx) {
                mx = i + p[i];
                id = i;
            }
        }
    }
}M;
```

Listing 4.6: Manacher.cc

4.7 回文自动机 Palindorme Tree

```
struct PT{
    struct Node{
        Node *ch[C], *suffix;
        int len;
    }bar[N], *foo, *last, *odd, *even;
    char s[N];
    int n, cnt;//cnt = foo - bar = count of palindromes, n the
    number of char added

    void init() {
        odd = bar, even = last = odd + 1, foo = even + 1;
        MST(odd->ch, 0), MST(even->ch, 0);
        odd->suffix = even->suffix = odd;
        odd->len = -1, even->len = 0;
        n = 0, cnt = 2;//root
    }

    Node* New_Node(int x) {
        MST(foo->ch, 0), foo->len = x;
        return foo++;
    }

    int index(char c) {
        return c - 'a';
    }

```

```
    Node* get(Node *p) {
        while (n - p->len - 2 < 0 || s[n - p->len - 2] != s[n - 1]) p = p->suffix;
        return p;
    }
    bool add(char c) {
        int x = index(c); s[n++] = c;
        Node *p = get(last);
        if (!p->ch[x]) {
            last = New_Node(p->len + 2);
            if (last->len == 1) {
                last->suffix = even;
            }
            else last->suffix = get(p->suffix)->ch[x];//guarantee proper suffix
            p->ch[x] = last;
            cnt++;
            return 1;
        }
        else {
            last = p->ch[x];
            return 0;
        }
    }
};
```

Listing 4.7: PalindromicTree.cpp

4.8 后缀自动机 SAM

```
struct SAM{
    struct Node{
        Node *parent, *ch[26];
        int val;
    };

    Node bar[N], *foo, *root, *last;

    Node* NewNode(int x) {
        MST(foo->ch, 0);
        foo->val = x;
        return foo++;
    }

    void extend(int w) {
        Node *p = last;
        Node *np = NewNode(p->val + 1);
        while (p && p->ch[w] == 0) p->ch[w] = np, p = p->parent;
        if (p == 0) np->parent = root;
        else {
            Node *q = p->ch[w];
            if (p->val + 1 == q->val) {
                np->parent = q;
            }
            else {
                Node  *nq = NewNode(p->val + 1);
                memcpy(nq->ch, q->ch, sizeof(q->ch));
                nq->parent = q->parent;
                q->parent = nq;
                np->parent = nq;
                while (p && p->ch[w] == q) p->ch[w] = nq, p = p->parent;
            }
        }
        last = np;
    }
}sam;
```

Listing 4.8: SAM.cc

Chapter 5

数据结构

5.1 树状数组第 k 大

```cpp
//找第k个空的位置 先全赋值1 然后每次加一个，update -1
int find(int k) {
    int ans = 0;
    for (int i = LOG; i >= 0; i--) {
        ans += 1 << i;
        if (ans > n || k <= s[ans]) ans -= 1 << i;
        else k -= s[ans];
    }
    return ans + 1;
}
```

Listing 5.1: Bit.cc

5.2 矩形面积并

```cpp
struct rectangle{
    int a, b, c, d;
};

struct line{
    int x, up, down, ty;
};

const int N = 65; //N 线段树的长度，因为离散化的缘故是矩形个数的3倍
int sum[N << 2], cnt[N << 2];
int ql, qr, qd, tot, cnt; //tot是竖线的总量， cnt是矩形的个数
vector<int> num; //num是离散化用的数组
line l[2 * N];
rectangle p[N];

void maintain(int x, int l, int r) {
    if (cnt[x]) {
        sum[x] = num[r] - num[l - 1];
    }
    else {
        if (l == r) sum[x] = 0;
        else {
            sum[x] = sum[L] + sum[R];
        }
    }
}

void build(int x, int l, int r) {
    if (l == r) {
        sum[x] = cnt[x] = 0;
    }
    else {
        build(LC), build(RC);
        sum[x] = cnt[x] = 0;
    }
}

void update(int x, int l, int r) {
    if (ql <= l && qr >= r) {
        cnt[x] += qd;
        maintain(x, l, r);
    }
    else {
        if (ql <= MID) update(LC);
        if (qr > MID) update(RC);
        maintain(x, l, r);
    }
}


bool cmp(const line &a, const line &b) {
    if (a.x != b.x) return a.x < b.x;
    return a.ty > b.ty;
}

int solve() {
    tot = 0;
```

```
    REP(i, cnt) {
        l[tot++] = (line) { p[i].a, p[i].b + 1, p[i].d, 1 };
        l[tot++] = (line) { p[i].c, p[i].b + 1, p[i].d, -1 };
        num.push_back(p[i].b + 1);
        num.push_back(p[i].d);        num.push_back(p[i].d + 1);
    }
    sort(num.begin(), num.end());
    num.resize(unique(num.begin(), num.end()) - num.begin());
    sort(l, l + tot, cmp);
    REP(i, tot) {
        l[i].up = lower_bound(num.begin(), num.end(), l[i].up) -
    num.begin() + 1;
        l[i].down = lower_bound(num.begin(), num.end(), l[i].down)
    - num.begin() + 1;
    }
    int n = num.size() - 1;
    build(1, 1, n);
    int ans = 0;
    REP(i, tot) {
        if (i) {
            ans += sum[1] * (l[i].x - l[i - 1].x); //可能需要LL
        }
        ql = l[i].up, qr = l[i].down, qd = l[i].ty;
        update(1, 1, n);
    }
    num.clear(); //清空离散化的数组
    return ans;
}
```

Listing 5.2: Rectangle.cpp

5.3 Treap

5.3.1 Classic

```
int tsize[N], key[N], cnt[N], weight[N] = {INT_MAX}, son[N][2],
    node = 1;

void maintain(int x) {
    tsize[x] = tsize[son[x][0]] + cnt[x] + tsize[son[x][1]];
}

void rotate(int &x, int t) {
    int y = son[x][t];
    son[x][t] = son[y][1 ^ t];
    son[y][1 ^ t] = x;
    maintain(x), maintain(y);
    x = y;
}

void insert(int &x, int y) {
    if (x) {
        if (key[x] == y) {
            cnt[x]++;
        }
        else {
            int t = key[x] < y;
            insert(son[x][t], y);
            if (weight[x] > weight[son[x][t]]) rotate(x, t);
        }
    }
    else {
        x = node++;
        key[x] = y;
        MST(son[x], 0);
        weight[x] = rand();
        cnt[x] = 1;
    }
    maintain(x);
}

void erase(int &x, int y) {
    if (!x) return ;
    if (key[x] == y) {
        if (cnt[x] > 1) {
            cnt[x]--;
        }
        else {
            if (!tsize[son[x][0]] && !tsize[son[x][1]]) {
                x = 0;
                return ;
            }
            rotate(x, weight[son[x][0]] > weight[son[x][1]]);
            erase(x, k);
        }
    }
    else {
        erase(son[x][key[x] < y], y);
    }
    maintain(x);
}
```

```cpp

int select(int &x, int y) {
    if (tsize[son[x][0]] >= y) {
        return select(son[x][0], y);
    }    else if (tsize[son[x][0]] + cnt[x] >= y) {
        return key[x];
    }
    else {
        return select(son[x][1], y - cnt[x] - tsize[son[x][0]]);
    }
}
```

Listing 5.3: Treap.cpp

5.3.2 ftiasch

```cpp
struct Node{
    int value, size;
    Node *l, *r;

    Node* update() {
        size = l->size + 1 + r->size;
        return this;
    }
}bar[N * 15], *foo, *rt[N << 2], *null;

void init() {
    foo = null = bar;
    null->l = null->r = null;
    foo++;
}

Node* New_Node(int x) {
    return new(foo++) (Node) {x, 1, null, null};
}

bool gen(int a, int b) {
    return rand() % (a + b) < a;
}

Node* merge(Node *a, Node *b) {
    if (a == null) return b;
    if (b == null) return a;
    if (gen(a->size, b->size)) {
        a->r = merge(a->r, b);
        return a->update();
    }
    else {
        b->l = merge(a, b->l);
        return b->update();
    }
}

#define PNN pair<Node*, Node*>

PNN split(Node *u, int s) {
    if (u == null) return {null, null};
    Node *l = u->l, *r = u->r;
    if (l->size >= s) {
        PNN res = split(l, s);
        u->l = res.second;
        return {res.first, u->update()};
    }
    else {
        PNN res = split(r, s - (l->size + 1));
        u->r = res.first;
        return {u->update(), res.second};
    }
}

void show(Node *u) {
    if (u == null) return ;
    else {
        show(u->l);
        printf("value: %d\n", u->value);
        show(u->r);
    }
}

int find(Node *u, int x) {
    if (u == null) return 0;
    if (u->value <= x) {
        return u->l->size + 1 + find(u->r, x);
    }
    else return find(u->l, x);
}

Node* erase(Node *&u, int s) {
    int tmp = find(u, s);
    PNN res = split(u, tmp - 1);
    PNN ans = split(res.second, 1);
    u = merge(res.first, ans.second);
```

```
    return ans.first;
}

void insert(Node *a, Node *&u) {
    int ans = find(u, a->value);    PNN res = split(u, ans);
    u = a;
    u = merge(res.first, u);
    u = merge(u, res.second);
}

void dfs(Node *a, Node *&b) {
    if (a == null) return ;
    else {
        dfs(a->l, b);
        dfs(a->r, b);
        a->l = a->r = null;
        insert(a->update(), b);
    }
}

Node* combine(Node *a, Node *b) {
    if (a->size > b->size) {
        swap(a, b);
    }
    dfs(a, b);
    return b;
}
```

Listing 5.4: Treap2.cpp

5.4 主席树 Persistent Segment Tree

```
struct SGT{
    int l, r, cnt;
    SGT(int a = 0, int b = 0, int c = 0) : l(a), r(b), cnt(c) {}
}p[N * 20];

int head[N], n, m, node;

int New_Node() {
    p[node] = SGT();
    return node++;
}

void update(int pre, int &now, int num, int l, int r) {
    now = New_Node();
    if (l == r) {
        p[now].cnt = p[pre].cnt + 1;
    }
    else {
        int mid = l + r >> 1;
        if (num <= mid) {
            update(p[pre].l, p[now].l, num, l, mid);
            p[now].r = p[pre].r;
        }
        else {
            update(p[pre].r, p[now].r, num, mid + 1, r);
            p[now].l = p[pre].l;
        }
        p[now].cnt = p[p[now].l].cnt + p[p[now].r].cnt;
    }
}

int query(int x, int y, int z, int w, int num, int l, int r) {
    if (l == r) {
        return p[x].cnt + p[y].cnt - p[z].cnt - p[w].cnt > 0;
    }
    else {
        int mid = l + r >> 1;
        int tmp = p[x].cnt + p[y].cnt - p[z].cnt - p[w].cnt;
        if (tmp == 0) return 0;
        else {
            if (num <= mid) {
                return query(p[x].l, p[y].l, p[z].l, p[w].l, num, l
    , mid);
            }
            else {
                return query(p[x].r, p[y].r, p[z].r, p[w].r, num,
    mid + 1, r);
            }
        }
    }
}
//New_Node() 0号节点作为空节点，head[0] = 1是空树的根，从1开始标号
//head 每棵线段树的起始点 val  每个节点上存的数字 fa
    每个节点的父亲节点
```

Listing 5.5: PersistentSegmentTree.cpp

5.5 树链剖分

```cpp
int top[N], pos[N], l[N], r[N], pre[N], son[N], size[N], inv[N],
    dep[N], now;

void dfs(int x, int p) {
    pre[x] = p, size[x] = 1, son[x] = 0;
    dep[x] = dep[p] + 1;
    for (int go = fi[x]; go; go = ne[go]) {
        int y = en[go];
        if (y != p) {
            dfs(y, x);
            size[x] += size[y];
            if (size[y] > size[son[x]]) {
                son[x] = y;
            }
        }
    }
}

void divide(int x, int tp) {
    top[x] = tp, pos[x] = l[x] = now, inv[now++] = x;
    if (son[x]) divide(son[x], tp);
    for (int go = fi[x]; go; go = ne[go]) {
        int y = en[go];
        if (y != pre[x] && y != son[x]) {
            divide(y, y);
        }
    }
    r[x] = now - 1;
}

int query(int a, int b) {
    int ans = -INT_MAX;
    while (top[a] - top[b]) {
        if (dep[top[a]] > dep[top[b]]) {
            swap(a, b);
        }
        ql = pos[top[b]], qr = pos[b];
        ans = max(ans, query(1, 1, n));
        b = pre[top[b]];
    }
    if (dep[a] > dep[b]) {
        swap(a, b);
    }
    ql = pos[a], qr = pos[b];
    ans = max(ans, query(1, 1, n));
    return ans;
}
```

Listing 5.6: HeavyLight.cpp

5.6 树分治

```cpp
struct Edge{
    int dist, next, ed;
}e[M];

int edge, head[N], start[N];

void add(int x, int y) {
    e[++edge].next = head[x];
    head[x] = edge;
    e[edge].ed = y;
}

void add(int x, int y, int z) {
    e[++edge].next = start[x];
    start[x] = edge;
    e[edge].ed = y;
    e[edge].dist = z;
}

int n, m;
int vis[N], size[N], maxsub[N], root;
int dist[N];

void dfs_size(int x, int p = 0) {
    size[x] = 1, maxsub[x] = 0;
    for (int go = head[x]; go; go = e[go].next) {
        int y = e[go].ed;
        if (y != p && !vis[y]) {
            dfs_size(y, x);
            size[x] += size[y];
            if (size[y] > maxsub[x]) maxsub[x] = size[y];
        }
    }
}

void dfs_root(int x, int tot, int p = 0) {
    if (tot - size[x] > maxsub[x]) maxsub[x] = tot - size[x];
```

```
38      if (maxsub[x] < maxsub[0]) maxsub[0] = maxsub[x], root = x;
39      for (int go = head[x]; go; go = e[go].next) {
40          int y = e[go].ed;
41          if (y != p && !vis[y]) {            dfs_root(y, tot, x);
42          }
43      }
44  }
45
46  void dfs_dist(int x, int p = 0, int dep = 0) {
47      add(x, root, dep);
48      for (int go = head[x]; go; go = e[go].next) {
49          int y = e[go].ed;
50          if (!vis[y] && y != p) {
51              dfs_dist(y, x, dep + 1);
52          }
53      }
54  }
55
56  void divide(int x) {
57      maxsub[0] = N;
58      dfs_size(x);
59      dfs_root(x, size[x]);
60      dfs_dist(root);
61      vis[root] = 1;
62      for (int go = head[root]; go; go = e[go].next) {
63          int y = e[go].ed;
64          if (!vis[y]) divide(y);
65      }
66  }
```

Listing 5.7: TreeDivide.cc

Chapter 6

杂项

6.1 公式

6.1.1 求和公式

1. $\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6$
2. $\sum_{i=1}^{n} i^3 = (n(n+1)/2)^2$
3. $\sum_{i=1}^{n} i^4 = n(n+1)(2n+1)(3n^2+3n-1)/30$
4. $\sum_{i=1}^{n} i^5 = n^2(n+1)^2(2n^2+2n-1)/12$

不断求导, 可以推导之后的.

6.1.2 常用积分公式

含有 $ax+b$ 的积分 $(a \neq 0)$

1. $\int \frac{\mathrm{d}x}{ax+b} = \frac{1}{a}\ln|ax+b| + C$
2. $\int (ax+b)^{\mu}\mathrm{d}x = \frac{1}{a(\mu+1)}(ax+b)^{\mu+1} + C(\mu \neq 1)$
3. $\int \frac{x}{ax+b}\mathrm{d}x = \frac{1}{a^2}(ax+b-b\ln|ax+b|) + C$
4. $\int \frac{x^2}{ax+b}\mathrm{d}x = \frac{1}{a^3}\left(\frac{1}{2}(ax+b)^2 - 2b(ax+b) + b^2\ln|ax+b|\right) + C$
5. $\int \frac{\mathrm{d}x}{x(ax+b)} = -\frac{1}{b}\ln\left|\frac{ax+b}{x}\right| + C$
6. $\int \frac{\mathrm{d}x}{x^2(ax+b)} = -\frac{1}{bx} + \frac{a}{b^2}\ln\left|\frac{ax+b}{x}\right| + C$
7. $\int \frac{x}{(ax+b)^2}\mathrm{d}x = \frac{1}{a^2}\left(\ln|ax+b| + \frac{b}{ax+b}\right) + C$
8. $\int \frac{x^2}{(ax+b)^2}\mathrm{d}x = \frac{1}{a^3}\left(ax+b-2b\ln|ax+b| - \frac{b^2}{ax+b}\right) + C$
9. $\int \frac{\mathrm{d}x}{x(ax+b)^2} = \frac{1}{b(ax+b)} - \frac{1}{b^2}\ln\left|\frac{ax+b}{x}\right| + C$

含有 $\sqrt{ax+b}$ 的积分

1. $\int \sqrt{ax+b}\mathrm{d}x = \frac{2}{3a}\sqrt{(ax+b)^3} + C$
2. $\int x\sqrt{ax+b}\mathrm{d}x = \frac{2}{15a^2}(3ax-2b)\sqrt{(ax+b)^3} + C$
3. $\int x^2\sqrt{ax+b}\mathrm{d}x = \frac{2}{105a^3}(15a^2x^2 - 12abx + 8b^2)\sqrt{(ax+b)^3} + C$
4. $\int \frac{x}{\sqrt{ax+b}}\mathrm{d}x = \frac{2}{3a^2}(ax-2b)\sqrt{ax+b} + C$
5. $\int \frac{x^2}{\sqrt{ax+b}}\mathrm{d}x = \frac{2}{15a^3}(3a^2x^2 - 4abx + 8b^2)\sqrt{ax+b} + C$
6. $\int \frac{\mathrm{d}x}{x\sqrt{ax+b}} = \begin{cases} \frac{1}{\sqrt{b}}\ln\left|\frac{\sqrt{ax+b}-\sqrt{b}}{\sqrt{ax+b}+\sqrt{b}}\right| + C & (b>0) \\ \frac{2}{\sqrt{-b}}\arctan\sqrt{\frac{ax+b}{-b}} + C & (b<0) \end{cases}$
7. $\int \frac{\mathrm{d}x}{x^2\sqrt{ax+b}} = -\frac{\sqrt{ax+b}}{bx} - \frac{a}{2b}\int \frac{\mathrm{d}x}{x\sqrt{ax+b}}$
8. $\int \frac{\sqrt{ax+b}}{x}\mathrm{d}x = 2\sqrt{ax+b} + b\int \frac{\mathrm{d}x}{x\sqrt{ax+b}}$
9. $\int \frac{\sqrt{ax+b}}{x^2}\mathrm{d}x = -\frac{\sqrt{ax+b}}{x} + \frac{a}{2}\int \frac{\mathrm{d}x}{x\sqrt{ax+b}}$

含有 $x^2 \pm a^2$ 的积分

1. $\int \frac{\mathrm{d}x}{x^2+a^2} = \frac{1}{a}\arctan\frac{x}{a} + C$
2. $\int \frac{\mathrm{d}x}{(x^2+a^2)^n} = \frac{x}{2(n-1)a^2(x^2+a^2)^{n-1}} + \frac{2n-3}{2(n-1)a^2}\int \frac{\mathrm{d}x}{(x^2+a^2)^{n-1}}$
3. $\int \frac{\mathrm{d}x}{x^2-a^2} = \frac{1}{2a}\ln\left|\frac{x-a}{x+a}\right| + C$

含有 $ax^2+b(a>0)$ 的积分

1. $\int \frac{\mathrm{d}x}{ax^2+b}=\begin{cases}\frac{1}{\sqrt{ab}}\arctan\sqrt{\frac{a}{b}}x+C & (b>0)\\ \frac{1}{2\sqrt{-ab}}\ln\left|\frac{\sqrt{a}x-\sqrt{-b}}{\sqrt{a}x+\sqrt{-b}}\right|+C & (b<0)\end{cases}$
2. $\int \frac{x}{ax^2+b}\mathrm{d}x=\frac{1}{2a}\ln\left|ax^2+b\right|+C$
3. $\int \frac{x^2}{ax^2+b}\mathrm{d}x=\frac{x}{a}-\frac{b}{a}\int\frac{\mathrm{d}x}{ax^2+b}$
4. $\int \frac{\mathrm{d}x}{x(ax^2+b)}=\frac{1}{2b}\ln\frac{x^2}{|ax^2+b|}+C$
5. $\int \frac{\mathrm{d}x}{x^2(ax^2+b)}=-\frac{1}{bx}-\frac{a}{b}\int\frac{\mathrm{d}x}{ax^2+b}$
6. $\int \frac{\mathrm{d}x}{x^3(ax^2+b)}=\frac{a}{2b^2}\ln\frac{|ax^2+b|}{x^2}-\frac{1}{2bx^2}+C$
7. $\int \frac{\mathrm{d}x}{(ax^2+b)^2}=\frac{x}{2b(ax^2+b)}+\frac{1}{2b}\int\frac{\mathrm{d}x}{ax^2+b}$

含有 $ax^2+bx+c(a>0)$ 的积分

1. $\frac{\mathrm{d}x}{ax^2+bx+c}=\begin{cases}\frac{2}{\sqrt{4ac-b^2}}\arctan\frac{2ax+b}{\sqrt{4ac-b^2}}+C & (b^2<4ac)\\ \frac{1}{\sqrt{b^2-4ac}}\ln\left|\frac{2ax+b-\sqrt{b^2-4ac}}{2ax+b+\sqrt{b^2-4ac}}\right|+C & (b^2>4ac)\end{cases}$
2. $\int \frac{x}{ax^2+bx+c}\mathrm{d}x=\frac{1}{2a}\ln|ax^2+bx+c|-\frac{b}{2a}\int\frac{\mathrm{d}x}{ax^2+bx+c}$

含有 $\sqrt{x^2+a^2}(a>0)$ 的积分

1. $\int \frac{\mathrm{d}x}{\sqrt{x^2+a^2}}=\operatorname{arsh}\frac{x}{a}+C_1=\ln(x+\sqrt{x^2+a^2})+C$
2. $\int \frac{\mathrm{d}x}{\sqrt{(x^2+a^2)^3}}=\frac{x}{a^2\sqrt{x^2+a^2}}+C$
3. $\int \frac{x}{\sqrt{x^2+a^2}}\mathrm{d}x=\sqrt{x^2+a^2}+C$
4. $\int \frac{x}{\sqrt{(x^2+a^2)^3}}\mathrm{d}x=-\frac{1}{\sqrt{x^2+a^2}}+C$
5. $\int \frac{x^2}{\sqrt{x^2+a^2}}\mathrm{d}x=\frac{x}{2}\sqrt{x^2+a^2}-\frac{a^2}{2}\ln(x+\sqrt{x^2+a^2})+C$
6. $\int \frac{x^2}{\sqrt{(x^2+a^2)^3}}\mathrm{d}x=-\frac{x}{\sqrt{x^2+a^2}}+\ln(x+\sqrt{x^2+a^2})+C$
7. $\int \frac{\mathrm{d}x}{x\sqrt{x^2+a^2}}=\frac{1}{a}\ln\frac{\sqrt{x^2+a^2}-a}{|x|}+C$
8. $\int \frac{\mathrm{d}x}{x^2\sqrt{x^2+a^2}}=-\frac{\sqrt{x^2+a^2}}{a^2x}+C$
9. $\int \sqrt{x^2+a^2}\mathrm{d}x=\frac{x}{2}\sqrt{x^2+a^2}+\frac{a^2}{2}\ln(x+\sqrt{x^2+a^2})+C$
10. $\int \sqrt{(x^2+a^2)^3}\mathrm{d}x=\frac{x}{8}(2x^2+5a^2)\sqrt{x^2+a^2}+\frac{3}{8}a^4\ln(x+\sqrt{x^2+a^2})+C$
11. $\int x\sqrt{x^2+a^2}\mathrm{d}x=\frac{1}{3}\sqrt{(x^2+a^2)^3}+C$
12. $\int x^2\sqrt{x^2+a^2}\mathrm{d}x=\frac{x}{8}(2x^2+a^2)\sqrt{x^2+a^2}-\frac{a^4}{8}\ln(x+\sqrt{x^2+a^2})+C$
13. $\int \frac{\sqrt{x^2+a^2}}{x}\mathrm{d}x=\sqrt{x^2+a^2}+a\ln\frac{\sqrt{x^2+a^2}-a}{|x|}+C$
14. $\int \frac{\sqrt{x^2+a^2}}{x^2}\mathrm{d}x=-\frac{\sqrt{x^2+a^2}}{x}+\ln(x+\sqrt{x^2+a^2})+C$

含有 $\sqrt{x^2-a^2}(a>0)$ 的积分

1. $\int \frac{\mathrm{d}x}{\sqrt{x^2-a^2}}=\frac{x}{|x|}\operatorname{arch}\frac{|x|}{a}+C_1=\ln\left|x+\sqrt{x^2-a^2}\right|+C$
2. $\int \frac{\mathrm{d}x}{\sqrt{(x^2-a^2)^3}}=-\frac{x}{a^2\sqrt{x^2-a^2}}+C$
3. $\int \frac{x}{\sqrt{x^2-a^2}}\mathrm{d}x=\sqrt{x^2-a^2}+C$
4. $\int \frac{x}{\sqrt{(x^2-a^2)^3}}\mathrm{d}x=-\frac{1}{\sqrt{x^2-a^2}}+C$
5. $\int \frac{x^2}{\sqrt{x^2-a^2}}\mathrm{d}x=\frac{x}{2}\sqrt{x^2-a^2}+\frac{a^2}{2}\ln|x+\sqrt{x^2-a^2}|+C$
6. $\int \frac{x^2}{\sqrt{(x^2-a^2)^3}}\mathrm{d}x=-\frac{x}{\sqrt{x^2-a^2}}+\ln|x+\sqrt{x^2-a^2}|+C$
7. $\int \frac{\mathrm{d}x}{x\sqrt{x^2-a^2}}=\frac{1}{a}\arccos\frac{a}{|x|}+C$
8. $\int \frac{\mathrm{d}x}{x^2\sqrt{x^2-a^2}}=\frac{\sqrt{x^2-a^2}}{a^2x}+C$
9. $\int \sqrt{x^2-a^2}\mathrm{d}x=\frac{x}{2}\sqrt{x^2-a^2}-\frac{a^2}{2}\ln|x+\sqrt{x^2-a^2}|+C$
10. $\int \sqrt{(x^2-a^2)^3}\mathrm{d}x=\frac{x}{8}(2x^2-5a^2)\sqrt{x^2-a^2}+\frac{3}{8}a^4\ln|x+\sqrt{x^2-a^2}|+C$
11. $\int x\sqrt{x^2-a^2}\mathrm{d}x=\frac{1}{3}\sqrt{(x^2-a^2)^3}+C$
12. $\int x^2\sqrt{x^2-a^2}\mathrm{d}x=\frac{x}{8}(2x^2-a^2)\sqrt{x^2-a^2}-\frac{a^4}{8}\ln|x+\sqrt{x^2-a^2}|+C$
13. $\int \frac{\sqrt{x^2-a^2}}{x}\mathrm{d}x=\sqrt{x^2-a^2}-a\arccos\frac{a}{|x|}+C$
14. $\int \frac{\sqrt{x^2-a^2}}{x^2}\mathrm{d}x=-\frac{\sqrt{x^2-a^2}}{x}+\ln|x+\sqrt{x^2-a^2}|+C$

含有 $\sqrt{a^2-x^2}(a>0)$ 的积分

1. $\int \frac{\mathrm{d}x}{\sqrt{a^2-x^2}} = \arcsin\frac{x}{a} + C$
2. $\frac{\mathrm{d}x}{\sqrt{(a^2-x^2)^3}} = \frac{x}{a^2\sqrt{a^2-x^2}} + C$
3. $\int \frac{x}{\sqrt{a^2-x^2}}\mathrm{d}x = -\sqrt{a^2-x^2} + C$
4. $\int \frac{x}{\sqrt{(a^2-x^2)^3}}\mathrm{d}x = \frac{1}{\sqrt{a^2-x^2}} + C$
5. $\int \frac{x^2}{\sqrt{a^2-x^2}}\mathrm{d}x = -\frac{x}{2}\sqrt{a^2-x^2} + \frac{a^2}{2}\arcsin\frac{x}{a} + C$
6. $\int \frac{x^2}{\sqrt{(a^2-x^2)^3}}\mathrm{d}x = \frac{x}{\sqrt{a^2-x^2}} - \arcsin\frac{x}{a} + C$
7. $\int \frac{\mathrm{d}x}{x\sqrt{a^2-x^2}} = \frac{1}{a}\ln\frac{a-\sqrt{a^2-x^2}}{|x|} + C$
8. $\int \frac{\mathrm{d}x}{x^2\sqrt{a^2-x^2}} = -\frac{\sqrt{a^2-x^2}}{a^2x} + C$
9. $\int \sqrt{a^2-x^2}\mathrm{d}x = \frac{x}{2}\sqrt{a^2-x^2} + \frac{a^2}{2}\arcsin\frac{x}{a} + C$
10. $\int \sqrt{(a^2-x^2)^3}\mathrm{d}x = \frac{x}{8}(5a^2-2x^2)\sqrt{a^2-x^2} + \frac{3}{8}a^4\arcsin\frac{x}{a} + C$
11. $\int x\sqrt{a^2-x^2}\mathrm{d}x = -\frac{1}{3}\sqrt{(a^2-x^2)^3} + C$
12. $\int x^2\sqrt{a^2-x^2}\mathrm{d}x = \frac{x}{8}(2x^2-a^2)\sqrt{a^2-x^2} + \frac{a^4}{8}\arcsin\frac{x}{a} + C$
13. $\int \frac{\sqrt{a^2-x^2}}{x}\mathrm{d}x = \sqrt{a^2-x^2} + a\ln\frac{a-\sqrt{a^2-x^2}}{|x|} + C$
14. $\int \frac{\sqrt{a^2-x^2}}{x^2}\mathrm{d}x = -\frac{\sqrt{a^2-x^2}}{x} - \arcsin\frac{x}{a} + C$

含有 $\sqrt{\pm ax^2+bx+c}(a>0)$ 的积分

1. $\int \frac{\mathrm{d}x}{\sqrt{ax^2+bx+c}} = \frac{1}{\sqrt{a}}\ln|2ax+b+2\sqrt{a}\sqrt{ax^2+bx+c}| + C$
2. $\int \sqrt{ax^2+bx+c}\mathrm{d}x =$
   $\frac{2ax+b}{4a}\sqrt{ax^2+bx+c} + \frac{4ac-b^2}{8\sqrt{a^3}}\ln|2ax+b+2\sqrt{a}\sqrt{ax^2+bx+c}| + C$
3. $\int \frac{x}{\sqrt{ax^2+bx+c}}\mathrm{d}x = \frac{1}{a}\sqrt{ax^2+bx+c} - \frac{b}{2\sqrt{a^3}}\ln|2ax+b+2\sqrt{a}\sqrt{ax^2+bx+c}| + C$
4. $\int \frac{\mathrm{d}x}{\sqrt{c+bx-ax^2}} = -\frac{1}{\sqrt{a}}\arcsin\frac{2ax-b}{\sqrt{b^2+4ac}} + C$
5. $\int \sqrt{c+bx-ax^2}\mathrm{d}x = \frac{2ax-b}{4a}\sqrt{c+bx-ax^2} + \frac{b^2+4ac}{8\sqrt{a^3}}\arcsin\frac{2ax-b}{\sqrt{b^2+4ac}} + C$
6. $\int \frac{x}{\sqrt{c+bx-ax^2}}\mathrm{d}x = -\frac{1}{a}\sqrt{c+bx-ax^2} + \frac{b}{2\sqrt{a^3}}\arcsin\frac{2ax-b}{\sqrt{b^2+4ac}} + C$

含有 $\sqrt{\pm\frac{x-a}{x-b}}$ 或 $\sqrt{(x-a)(x-b)}$ 的积分

1. $\int \sqrt{\frac{x-a}{x-b}}\mathrm{d}x = (x-b)\sqrt{\frac{x-a}{x-b}} + (b-a)\ln(\sqrt{|x-a|} + \sqrt{|x-b|}) + C$
2. $\int \sqrt{\frac{x-a}{b-x}}\mathrm{d}x = (x-b)\sqrt{\frac{x-a}{b-x}} + (b-a)\arcsin\sqrt{\frac{x-a}{b-x}} + C$
3. $\int \frac{\mathrm{d}x}{\sqrt{(x-a)(b-x)}} = 2\arcsin\sqrt{\frac{x-a}{b-x}} + C\ (a<b)$
4. $$\int \sqrt{(x-a)(b-x)}\mathrm{d}x = \frac{2x-a-b}{4}\sqrt{(x-a)(b-x)} + \frac{(b-a)^2}{4}\arcsin\sqrt{\frac{x-a}{b-x}} + C, (a<b) \quad (6.1)$$

含有三角函数的积分

1. $\int \sin x\mathrm{d}x = -\cos x + C$
2. $\int \cos x\mathrm{d}x = \sin x + C$
3. $\int \tan x\mathrm{d}x = -\ln|\cos x| + C$
4. $\int \cot x\mathrm{d}x = \ln|\sin x| + C$
5. $\int \sec x\mathrm{d}x = \ln\left|\tan\left(\frac{\pi}{4}+\frac{x}{2}\right)\right| + C = \ln|\sec x + \tan x| + C$
6. $\int \csc x\mathrm{d}x = \ln\left|\tan\frac{x}{2}\right| + C = \ln|\csc x - \cot x| + C$
7. $\int \sec^2 x\mathrm{d}x = \tan x + C$
8. $\int \csc^2 x\mathrm{d}x = -\cot x + C$
9. $\int \sec x\tan x\mathrm{d}x = \sec x + C$
10. $\int \csc x\cot x\mathrm{d}x = -\csc x + C$
11. $\int \sin^2 x\mathrm{d}x = \frac{x}{2} - \frac{1}{4}\sin 2x + C$
12. $\int \cos^2 x\mathrm{d}x = \frac{x}{2} + \frac{1}{4}\sin 2x + C$
13. $\int \sin^n x\mathrm{d}x = -\frac{1}{n}\sin^{n-1}x\cos x + \frac{n-1}{n}\int \sin^{n-2}x\mathrm{d}x$
14. $\int \cos^n x\mathrm{d}x = \frac{1}{n}\cos^{n-1}x\sin x + \frac{n-1}{n}\int \cos^{n-2}x\mathrm{d}x$
15. $\frac{\mathrm{d}x}{\sin^n x} = -\frac{1}{n-1}\frac{\cos x}{\sin^{n-1}x} + \frac{n-2}{n-1}\int \frac{\mathrm{d}x}{\sin^{n-2}x}$

16. $\frac{\mathrm{d}x}{\cos^n x}=\frac{1}{n-1}\frac{\sin x}{\cos^{n-1}x}+\frac{n-2}{n-1}\int\frac{\mathrm{d}x}{\cos^{n-2}x}$

17.

$$\begin{aligned}&\int\cos^m x\sin^n x\mathrm{d}x\\=&\frac{1}{m+n}\cos^{m-1}x\sin^{n+1}x+\frac{m-1}{m+n}\int\cos^{m-2}x\sin^n x\mathrm{d}x\\=&-\frac{1}{m+n}\cos^{m+1}x\sin^{n-1}x+\frac{n-1}{m+1}\int\cos^m x\sin^{n-2}x\mathrm{d}x\end{aligned}$$

18. $\int\sin ax\cos bx\mathrm{d}x=-\frac{1}{2(a+b)}\cos(a+b)x-\frac{1}{2(a-b)}\cos(a-b)x+C$

19. $\int\sin ax\sin bx\mathrm{d}x=-\frac{1}{2(a+b)}\sin(a+b)x+\frac{1}{2(a-b)}\sin(a-b)x+C$

20. $\int\cos ax\cos bx\mathrm{d}x=\frac{1}{2(a+b)}\sin(a+b)x+\frac{1}{2(a-b)}\sin(a-b)x+C$

21. $\int\frac{\mathrm{d}x}{a+b\sin x}=\begin{cases}\frac{2}{\sqrt{a^2-b^2}}\arctan\frac{a\tan\frac{x}{2}+b}{\sqrt{a^2-b^2}}+C & (a^2>b^2)\\\frac{1}{\sqrt{b^2-a^2}}\ln\left|\frac{a\tan\frac{x}{2}+b-\sqrt{b^2-a^2}}{a\tan\frac{x}{2}+b+\sqrt{b^2-a^2}}\right|+C & (a^2<b^2)\end{cases}$

22. $\int\frac{\mathrm{d}x}{a+b\cos x}=\begin{cases}\frac{2}{a+b}\sqrt{\frac{a+b}{a-b}}\arctan\left(\sqrt{\frac{a-b}{a+b}}\tan\frac{x}{2}\right)+C & (a^2>b^2)\\\frac{1}{a+b}\sqrt{\frac{a+b}{a-b}}\ln\left|\frac{\tan\frac{x}{2}+\sqrt{\frac{a+b}{b-a}}}{\tan\frac{x}{2}-\sqrt{\frac{a+b}{b-a}}}\right|+C & (a^2<b^2)\end{cases}$

23. $\int\frac{\mathrm{d}x}{a^2\cos^2x+b^2\sin^2x}=\frac{1}{ab}\arctan\left(\frac{b}{a}\tan x\right)+C$

24. $\int\frac{\mathrm{d}x}{a^2\cos^2x-b^2\sin^2x}=\frac{1}{2ab}\ln\left|\frac{b\tan x+a}{b\tan x-a}\right|+C$

25. $\int x\sin ax\mathrm{d}x=\frac{1}{a^2}\sin ax-\frac{1}{a}x\cos ax+C$

26. $\int x^2\sin ax\mathrm{d}x=-\frac{1}{a}x^2\cos ax+\frac{2}{a^2}x\sin ax+\frac{2}{a^3}\cos ax+C$

27. $\int x\cos ax\mathrm{d}x=\frac{1}{a^2}\cos ax+\frac{1}{a}x\sin ax+C$

28. $\int x^2\cos ax\mathrm{d}x=\frac{1}{a}x^2\sin ax+\frac{2}{a^2}x\cos ax-\frac{2}{a^3}\sin ax+C$

含有反三角函数的积分 (其中 $a>0$)

1. $\int\arcsin\frac{x}{a}\mathrm{d}x=x\arcsin\frac{x}{a}+\sqrt{a^2-x^2}+C$
2. $\int x\arcsin\frac{x}{a}\mathrm{d}x=(\frac{x^2}{2}-\frac{a^2}{4})\arcsin\frac{x}{a}+\frac{x}{4}\sqrt{x^2-x^2}+C$
3. $\int x^2\arcsin\frac{x}{a}\mathrm{d}x=\frac{x^3}{3}\arcsin\frac{x}{a}+\frac{1}{9}(x^2+2a^2)\sqrt{a^2-x^2}+C$
4. $\int\arccos\frac{x}{a}\mathrm{d}x=x\ arccos\frac{x}{a}-\sqrt{a^2-x^2}+C$
5. $\int x\arccos\frac{x}{a}\mathrm{d}x=(\frac{x^2}{2}-\frac{a^2}{4})\arccos\frac{x}{a}-\frac{x}{4}\sqrt{a^2-x^2}+C$
6. $\int x^2\arccos\frac{x}{a}\mathrm{d}x=\frac{x^3}{3}\arccos\frac{x}{a}-\frac{1}{9}(x^2+2a^2)\sqrt{a^2-x^2}+C$
7. $\int\arctan\frac{x}{a}\mathrm{d}x=x\arctan\frac{x}{a}-\frac{a}{2}\ln(a^2+x^2)+C$
8. $\int x\arctan\frac{x}{a}\mathrm{d}x=\frac{1}{2}(a^2+x^2)\arctan\frac{x}{a}-\frac{a}{2}x+C$
9. $\int x^2\arctan\frac{x}{a}\mathrm{d}x=\frac{x^3}{3}\arctan\frac{x}{a}-\frac{a}{6}x^2+\frac{a^3}{6}\ln(a^2+x^2)+C$

含有指数函数的积分

1. $\int a^x\mathrm{d}x=\frac{1}{\ln a}a^x+C$
2. $\int\mathrm{e}^{ax}\mathrm{d}x=\frac{1}{a}a^{ax}+C$
3. $\int x\mathrm{e}^{ax}\mathrm{d}x=\frac{1}{a^2}(ax-1)a^{ax}+C$
4. $\int x^n\mathrm{e}^{ax}\mathrm{d}x=\frac{1}{a}x^n\mathrm{e}^{ax}-\frac{n}{a}\int x^{n-1}\mathrm{e}^{ax}\mathrm{d}x$
5. $\int xa^x\mathrm{d}x=\frac{x}{\ln a}a^x-\frac{1}{(\ln a)^2}a^x+C$
6. $\int x^na^x\mathrm{d}x=\frac{1}{\ln a}x^na^x-\frac{n}{\ln a}\int x^{n-1}a^x\mathrm{d}x$
7. $\int\mathrm{e}^{ax}\sin bx\mathrm{d}x=\frac{1}{a^2+b^2}\mathrm{e}^{ax}(a\sin bx-b\cos bx)+C$
8. $\int\mathrm{e}^{ax}\cos bx\mathrm{d}x=\frac{1}{a^2+b^2}\mathrm{e}^{ax}(b\sin bx+a\cos bx)+C$
9. $\int\mathrm{e}^{ax}\sin^n bx\mathrm{d}x=$
$\frac{1}{a^2+b^2n^2}\mathrm{e}^{ax}\sin^{n-1}bx(a\sin bx-nb\cos bx)+\frac{n(n-1)b^2}{a^2+b^2n^2}\int\mathrm{e}^{ax}\sin^{n-2}bx\mathrm{d}x$
10. $\int\mathrm{e}^{ax}\cos^n bx\mathrm{d}x=$
$\frac{1}{a^2+b^2n^2}\mathrm{e}^{ax}\cos^{n-1}bx(a\cos bx+nb\sin bx)+\frac{n(n-1)b^2}{a^2+b^2n^2}\int\mathrm{e}^{ax}\cos^{n-2}bx\mathrm{d}x$

含有对数函数的积分

1. $\int\ln x\mathrm{d}x=x\ln x-x+C$
2. $\int\frac{\mathrm{d}x}{x\ln x}=\ln\left|\ln x\right|+C$
3. $\int x^n\ln x\mathrm{d}x=\frac{1}{n+1}x^{n+1}(\ln x-\frac{1}{n+1})+C$
4. $\int(\ln x)^n\mathrm{d}x=x(\ln x)^n-n\int(\ln x)^{n-1}\mathrm{d}x$
5. $\int x^m(\ln x)^n\mathrm{d}x=\frac{1}{m+1}x^{m+1}(\ln x)^n-\frac{n}{m+1}\int x^m(\ln x)^{n-1}\mathrm{d}x$

6.2 常数

6.2.1 梅森素数

2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203,
2281, 3217, 4253, 4423, 9689, 9941, 11213, 19937, 21701, 23209,
44497, 86243, 110503, 132049, 216091, 756839, 859433, 1257787,
1398269, 2976221, 3021377, 6972593, 13466917, 20996011, 24036583,
25964951, 30402457, 32582657, 37156667, 42643801, 43112609, 57885161

6.2.2 完美数

6, 28, 496, 8128, 33550336, 8589869056, 137438691328
2305843008139952128, 2658455991569831744654692615953842176

6.2.3 欧拉常数

$$\gamma = \lim_{n\to\infty} S_n = \lim_{n\to\infty} \sum_{k=1}^{n} \frac{1}{k} - \ln n = 0.57721566490153286060651209008240243104 2$$

6.3 反演

二项式反演:

$$a_n = \sum_{k=1}^{n} C_n^k b_k \Longleftrightarrow b_n = \sum_{k=1}^{n} (-1)^{n-k} C_n^k a_k$$

莫比乌斯反演:

$$F(d) = \sum_{d|n}^{\infty} f(n) \Longleftrightarrow f(d) = \sum_{d|n}^{\infty} F(n) * \mu(\frac{n}{d})$$

$$F(n) = \sum_{d|n} f(d) \Longleftrightarrow f(n) = \sum_{d|n} F(d) * \mu(\frac{n}{d})$$

6.4 Prüfer 编码与 Cayley 公式

Cayley 公式: 一个完全图 K_n 有 n^{n-2} 棵生成树，换句话说 n 个节点的带标号的无根树有 n^{n-2} 个.
n 个节点的度依次为 $D_1, D_2, ..., D_n$ 的无根树共有 $\frac{(n-2)!}{[(D_1-1)!(D_2-1)!..(D_n-1)!]}$ 个，因为此时 Prüfer 编码中的数字 i 恰好出现 $D_i - 1$ 次.

6.5 结论

1. 杨氏矩阵和钩子公式, 行列均递增 (假设左下角是最小角, 如果上面有格子, 则格子数更大, 右边也一样) 的矩阵. 1-n 组成的所有形状可能的杨氏矩阵的个数可以递推:

   $$F_1 = 1, F_2 = 2$$

   $$F_{n+2} = F_{n+1} + (n+1) * F_n \ (n \geq 1)$$

   对于给定形状的杨氏矩阵, 个数可以用钩子公式计算: $n!$ 除以每个格子的钩子长度。其中钩子长度定义为该格子朝向最小角的箭头长度 (行 5 列 3, 长度 7)。

2. 匹克公式 顶点坐标均是整点的简单多边形：面积 = 内部格点数目 + 边上格点数目/2-1

3. 赌徒输光定理: 在“公平”的赌博中，任一个拥有有限赌本的赌徒，只要长期赌下去，必然有一天会输光. A 和 B 赌博, A 有赌本 n_1, B 有赌本 n_2, 每局有 p 的概率 A 胜, $q = 1 - p$ 的概率 B 胜. 输家给赢家 1 块钱. 问两个人自己先输光的概率. 则有:$P(A) + P(B) = 1$.
   $P(A) = (1 - (\frac{p}{q})^{n_2})/(1 - (\frac{p}{q})^{n_1+n_2})$, $P(B) = (1 - (\frac{q}{p})^{n_1})/(1 - (\frac{q}{p})^{n_1+n_2})$.
   特别的, 如果 $p = q$, $P(A) = \frac{n_2}{n_1+n_2}, P(B) = \frac{n_1}{n_1+n_2}$. 这意味着, 击鼓传花的游戏, 如果每个人向左右相邻传的概率是一样的, 最后一个人 (其他人都接到过花) 获胜, 那么除去一开始传的人, 其他人获胜的概率是一样的!

4. 字符串的不同子串个数是 $O(N^2)$ 的, 对于每个后缀 i, 有 $n - sa[i] - height[i]$ 个子串是他独有的. 求和就是所有不同子串的个数.

5. 字符串的一个重复出现的子串一定是两个不同后缀的 lcp 的一部分. 那么和一个后缀的前缀相同的另一个后缀一定在后缀数组中他排在一起. 所以如果我们想知道一个后缀的最长前缀, 在原串中至少出现 2 次, 那么就是这个后缀在后缀数组中左右相邻的两个后缀的 lcp 取较大值. 也意味着, 比 lcp 还长的前缀就是之出现过一次的子串了.

6. 字符串的回文子串个数是 $O(N)$ 的, 严格小于等于 N.

7. 一棵树的 bfs 序列, 任何前缀都是一个联通块.

8. 判断无根树同构, 可以先求直径, 长度不一样, 就不同构, 否则, 取直径中点为根,(直径长为奇数的时候, 我们断开中间的边, 加一个中间点.) 转化成有根叔之后求树的最小表示, 判断即可. 也可以选树的重心做根, 但是重心最多也会有两个. 所以先判重心是否一样多, 然后如果一样多, 同是两个, 那么加个点, 选这个点做根. 然后求最小表示. 判断即可.

9. 定义 $f(n)$ 等于 n 除以他的最大的 $squarefree$ 的因子, 那么 $1 \to n$ 的范围内, 不同的 $f(i)$ 的个数不会太多.$1e6$ 只有 2000,$1e7$ 只有 6500,$1e8$ 只有 21000.

10. Counting Squarefree Numbers 也就是算 $\mu(x)=0$ 的个数, 可以做到 $O(\sqrt{N})$(利用莫比乌斯反演). 但是结果的规模至少比质数多, 也就是 $\geq \pi(n)$.$OEIS$ 告诉我们,10^n 以内的 $Squarefree Numbers$ 的数目大约是 $10^n/(\pi^2/6)$. 极限就是 $\pi^2/6$. 粗略估计的话,608/1000 的比例的样子.

11. 动态换根求 LCA,u,v,$root$, 先求 u,v 在原树的 $LCA=z$, 那么现在的根要么不在原叔 u,v 的 z 的子树里, 这时新 LCA 是 z, 否则, 就是 u 和老 z 的 LCA 和 v 和 z 的 LCA 里面深度较深的那个.

```
int getLca(int a, int b, int rt) {
    int c = get(a, b);
    if (get(c, rt) != c) return c;
    int x = get(a, rt), y = get(b, rt);
    if (dep[x] > dep[y]) return x;
    return y;
}
```

Listing 6.1: DynamicLCA.cc

12. 树的直径, 两棵树合并, 新的直径可以由之前树的直径的端点, 枚举组合得到, 每棵树只要保存一个直径的两个端点即可.

13. 给定边的顺序, 园内接多边形面积最大, 这时候边的顺序也就没有关系了.

14. hash 要用 unsigned long long, 首先保证没有本质性错误, 0111 和 111 是不一样的. 然后注意生日悖论, 只要有 $O(\sqrt{N})$ 个数, 那么就几乎会冲突 (随机 5 个人, 基本就有两个同一个月出生).

15. $\max\{a,b,c\}-\min\{a,b,c\}=(abs(a-b)+abs(b-c)+abs(c-a))/2$

16. Stirling 公式逼近阶乘, $n!=\sqrt{2\pi n}(\frac{n}{e})^n$

17. 树相关的结论:

    (a) 树的重心最多两个, 如果有两个, 那么他们相邻.

    (b) 树中所有点到某个点的距离和中，到重心的距离和是最小的；如果有两个重心，那么他们的距离和一样.

    (c) 把两个树通过一条边相连得到一个新的树，那么新的树的重心在连接原来两个树的重心的路径上.

    (d) 把一个树添加或删除一个叶子，那么它的重心最多只移动一条边的距离.

    (e) 树的直径可以又很多, 但是直径中点只有一个, (偶数的话, 直径中间那条边是固定的), 合并两棵树, 如果之前树的某个直径是 AB, 和 CD, 那么新树的直径是这 4 个点两两组合中的某个.

6.6 数论

1. 10^9 以内素数的最大间距是 282.

2. 10^9 以内的因子数最多的高合成数是 $735134400=2^6*3^3*5^2*7*11*13*17$, 有 1344 个约数.

3. 一个 $4k+1$ 型质数可以表示成两个完全平方数的和. 事实上有 $wilson$ 定理,$1+(2k!)^2$ 是 p 的倍数, 然后用无穷递降法让他们变得最小.$4k+3$ 型质数不能被表示.

4. 一个整数能被表示成 2 个完全平方数之和的充要条件是: 素因子中 $4k+3$ 型因子的指数是偶数.

5. 一个整数能被表示成 3 个完全平方数之和的充要条件是:n 不是 $4^m(8k+7)$ 的形式.

6. 给定两个互素的正整数 a 和 b，那么它们最大不能组合的数为 $ab-a-b$，不能组合的数的个数为 $(a-1)(b-1)/2$.

7. $\gcd(a^m-1,a^n-1)=a^{\gcd(m,n)}-1$, $\gcd(F_m,F_n)=F_{\gcd(m,n)}$

8. $(a^2+kb^2)(c^2+kd^2)=(ac+kbd)^2+k(ad-bc)^2=(ac-kbd)^2+k(ad+bc)^2$

9. $Lagrange's\ four-square\ theorem$:
$(a^2+b^2+c^2+d^2)(x^2+y^2+z^2+w^2)=$
$(ax+by+cz+dw)^2+(ay-bx+cw-dz)^2+(az-bw-cx+dy)^2+(aw+bz-cy-dx)^2$

10. $\sum_{i=1}^{N}\gcd(i,N)=\sum_{d|N}d\phi(\frac{N}{d})$.

11. $a>b,(a,b)=1,\gcd(a^m-b^m,a^n-b^n)=a^{\gcd(m,n)}-b^{\gcd(m,n)}$.

12. $d=\gcd(C_N^1,C_N^2,\ldots,C_N^N)$, 则有:

    (a) N 是质数, 那么 $d=N$.

    (b) N 有多个质因子, 那么 $d=1$.

    (c) N 只有一个质因子, 那么 d 就是这个质因子.

13. $x^2+y^2=n$ 的整数解的个数 $f(n)=4\sum_{d|n}H(d)$, 其中 d 是偶数, 则 $H(d)=0$, d 为奇数, 则 $H(d)=(-1)^{\frac{d-1}{2}}$.

14. 求 $x^2=n(mod\ p)$, 其中 p 是奇质数. 首先根据 $Euler$ 判别法直接判断 n 是不是二次剩余, 如果是我们任意取 $\omega=a^2-n$, 且 ω 不是 p 的二次剩余. 那么 $x=(a+\sqrt{\omega})^{(p+1)/2}$ 是原方程的解. 因为非二次剩余占一半, 直接随机, 然后欧拉判别法检测即可.

15. $Fibnacci$ 数列模一个数的周期. 显然模任何数是周期的, 而且可以证明周期的规模是 $O(N)$ 的. 设 $n = \Pi_{i=1}^{k} p_i^{a_i}$, 先对每个 $p_i^{a_i}$ 求出循环节 x_i, 答案就是所有 x_i 的最小公倍数.$Fibnacci$ 数模 p^a 的循环节是 $G(p)p^{a-1}$, 其中 $G(p)$ 是模 p 的循环节. 求 $G(p)$ 有如下定理:
如果 $(\frac{5}{p}) = 1$ 则 $G(p)$ 是 $p-1$ 的因子, 否则是 $2(p+1)$ 的因子. 对于 ≤ 5 的质数,$G(2) = 3$,$G(3) = 8$,$G(5) = 20$.

16. 广义的 $Fibnacci$ 数列模一个奇质数 p 的循环节,$f_{n+2} = af_{n+1} + bf_n$. 设 $c = a^2 + 4b$, 当 $(\frac{c}{p}) = 1$ 时, 循环节是 $p-1$ 的因子. 否则是 $(p+1)(p-1)$ 的因子.

17. 组合数的最小公倍数
$LCM(\binom{0}{N}, \binom{1}{N}, \ldots, \binom{N}{N}) = LCM(1, 2, \ldots, N, N+1)/(N+1)$