Algorithm Library

Shanghai Jiao Tong University

未至之境
Yet to be Explored

Chengyuan Luo

November, 2021

The template of these templates is based on palayutm/ply-template
Some of the codes are taken from:
FTRobbin/Dreadnought-Standard-Code-Library,
laonahongchen/ICPC-code-template-Quasar-,
laonahongchen/ICPC-code-template-Blazar.

Contents

1 Vim 配置

1.1 vimrc

```
filetype plugin indent on
syntax on
set softtabstop=4 shiftwidth=4 smarttab expandtab
set mouse=a backspace=2
set number relativenumber ruler
set listchars=trail:$ list

let mapleader = " "
inoremap jk <esc>
inoremap bbb {<enter>}<esc>O

nmap <Leader>go :call GoSh()<CR>
func GoSh()
    exec "w"
    exec "! ./go.sh a"
endfunc
```

1.2 VS Code

```
{
  "vim.easymotion": true,
  "vim.incsearch": true,
  "vim.useSystemClipboard": false,
  "vim.useCtrlKeys": true,
  "vim.hlsearch": true,
  "vim.insertModeKeyBindings": [
    {
      "before": ["j", "k"],
      "after": ["<Esc>"]
    },
    {

      "before": ["b", "b", "b"],
      "after": ["{", "<Enter>"]
    }
  ],
  "vim.leader": "<space>"
}
```

2 Z

```
using u32 = unsigned;
using u64 = unsigned long long;
const u32 MOD = 1E9 + 7;

struct Z {
    u32 v;
    Z(u32 v = 0) : v(v) {}
    Z& operator += (const Z &z) {
        v += z.v;
```

```
        if (v >= MOD) v -= MOD;
        return *this;
    }
    Z& operator -= (const Z &z) {
        if (v < z.v) v += MOD;
        v -= z.v;
        return *this;
    }
    Z& operator *= (const Z &z) {
        v = static_cast<u64>(v) * z.v % MOD;
        return *this;
    }
};
Z operator + (const Z &x, const Z &y) {
    return Z(x.v + y.v >= MOD ? x.v + y.v - MOD :
↪   x.v + y.v);
}
Z operator - (const Z &x, const Z &y) {
    return Z(x.v < y.v ? x.v + MOD - y.v : x.v -
↪   y.v);
}
Z operator * (const Z &x, const Z &y) {
    return Z(static_cast<u64>(x.v) * y.v % MOD);
}
Z qpow(Z base, u32 e) {
    Z ret(1);
    for (; e; e >>= 1) {
        if (e & 1) ret *= base;
        base *= base;
    }
    return ret;
}
istream& operator >> (istream &in, Z &x) {
    in >> x.v;
    return in;
}
ostream& operator << (ostream &os, const Z &z) {
    return os << z.v;
}
```

3 随机数生成器

```
using u32 = unsigned;
using u64 = unsigned long long;
const u32 MOD = 1E9 + 7;

struct Z {
    u32 v;
    Z(u32 v = 0) : v(v) {}
    Z& operator += (const Z &z) {
        v += z.v;
        if (v >= MOD) v -= MOD;
        return *this;
    }
    Z& operator -= (const Z &z) {
        if (v < z.v) v += MOD;
        v -= z.v;
        return *this;
    }
    Z& operator *= (const Z &z) {
        v = static_cast<u64>(v) * z.v % MOD;
        return *this;
    }
};
Z operator + (const Z &x, const Z &y) {
    return Z(x.v + y.v >= MOD ? x.v + y.v - MOD :
↪   x.v + y.v);
}
Z operator - (const Z &x, const Z &y) {
    return Z(x.v < y.v ? x.v + MOD - y.v : x.v -
↪   y.v);
}
Z operator * (const Z &x, const Z &y) {
    return Z(static_cast<u64>(x.v) * y.v % MOD);
}
Z qpow(Z base, u32 e) {
    Z ret(1);
    for (; e; e >>= 1) {
        if (e & 1) ret *= base;
        base *= base;
    }
    return ret;
}
istream& operator >> (istream &in, Z &x) {
    in >> x.v;
    return in;
}
ostream& operator << (ostream &os, const Z &z) {
    return os << z.v;
}
```

4 数列与计数

4.1 多项式板子

```
// SZ: size * 4
const size_t SZ = 1 << 19;
using Poly = vector<Z>;
using i64 = long long;

template <typename InputZ, typename Output>
void sp_copy(InputZ begin, InputZ end, Output
↪   output) {
    while (begin != end) *output++ = begin++->v;
}
int get_lg(int x) {
    return 32 - __builtin_clz(x) - ((x & (-x)) ==
↪   x);
}
Z inv[SZ + 5], ww[SZ];
void prep() {
    static bool has_prep = false;
    if (has_prep) return;
    inv[0] = inv[1] = 1;
    for (unsigned i = 2; i <= SZ; ++i)
        inv[i] = MOD - MOD / i * inv[MOD % i];
    ww[0] = 1;
    Z mul = qpow(3, (MOD - 1) / SZ);
    for (unsigned i = 1; i < SZ; ++i)
        ww[i] = ww[i - 1] * mul;
```

```
    has_prep = true;
}
void fft(i64 a[], int lg, bool flag) {
    prep();
    int n = 1 << lg;
    if (flag) reverse(a + 1, a + n);
    static int rev[SZ], rev_lg = -1;
    if (rev_lg != lg) {
        for (int i = 0; i < n; ++i)
            rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << lg >> 1);
        rev_lg = lg;
    }
    for (int i = 0; i < n; ++i)
        if (rev[i] > i) swap(a[i], a[rev[i]]);
    for (int m = 1, l = 2; m < n; m <<= 1, l <<= 1) {
        i64 *x = a, *y = a + m, xx, yy; int *w, mul[SZ];
        for (int i = 0, j = 0, step = SZ / l; i < m; ++i, j += step)
            mul[i] = ww[j].v;
        for (int i = 0; i < n; i += l) {
            w = mul;
            for (int j = 0; j < m; ++j, ++x, ++y, ++w) {
                xx = *x;
                yy = *y % MOD * *w;
                *x = xx + yy;
                *y = xx - yy;
            }
            x += m;
            y += m;
        }
        if (l >> 15 & 1)
            for (int i = 0; i < n; ++i)
                a[i] %= MOD;
    }
    for (int i = 0; i < n; ++i) {
        a[i] %= MOD;
        if (flag) (a[i] *= inv[n].v) %= MOD;
        if (a[i] < 0) a[i] += MOD;
    }
}
void fft(Z a[], int lg, bool flag) {
    static i64 ta[SZ];
    sp_copy(a, a + (1 << lg), ta);
    fft(ta, lg, flag);
    copy(ta, ta + (1 << lg), a);
}
Poly operator += (Poly &f, const Poly &g) {
    if (g.size() > f.size()) f.resize(g.size());
    auto it = f.begin();
    auto jt = g.begin();
    while (jt != g.end()) *it++ += *jt++;
    return f;
}
Poly operator + (const Poly &f, const Poly &g) {
    Poly ret = f; return ret += g;
}
Poly operator -= (Poly &f, const Poly &g) {
    if (g.size() > f.size()) f.resize(g.size());
    auto it = f.begin();
    auto jt = g.begin();
    while (jt != g.end()) *it++ -= *jt++;
    return f;
}
Poly operator - (const Poly &f, const Poly &g) {
    Poly ret = f; return ret -= g;
}
Poly operator * (const Poly &f, const Poly &g) {
    u32 n = f.size() + g.size() - 1;
    if ((i64) f.size() * g.size() <= 2048) {
        static u64 ans[SZ];
        memset(ans, 0, sizeof(u64) * n);
        for (u32 i = 0; i < f.size(); ++i)
            for (u32 j = 0; j < g.size(); ++j)
                if ((ans[i + j] += (u64) f[i].v * g[j].v) >> 62)
                    ans[i + j] %= MOD;
        Poly ret(n);
        for (u32 i = 0; i < n; ++i) ret[i] = ans[i] % MOD;
        return ret;
    }
    Poly ret(f.size() + g.size() - 1);
    static i64 a[SZ], b[SZ];
    int lg = get_lg(n);
    memset(a, 0, sizeof(i64) << lg);
    memset(b, 0, sizeof(i64) << lg);
    sp_copy(f.begin(), f.end(), a);
    sp_copy(g.begin(), g.end(), b);
    fft(a, lg, 0);
    fft(b, lg, 0);
    for (u32 i = 0, _ = 1 << lg; i < _; ++i)
        (a[i] *= b[i]) %= MOD;
    fft(a, lg, 1);
    copy(a, a + n, ret.begin());
    return ret;
}
Poly& operator *= (Poly &f, const Poly &g) {
    return f = f * g;
}
Poly& operator *= (Poly &f, const Z &x) {
    for (Z &c : f) c *= x;
    return f;
}
Poly operator * (const Poly &f, const Z &x) {
    Poly ret = f; return ret *= x;
}
void calc_inv(Z arr[], Z brr[], int n) {
    if (n == 1) {
        brr[0] = qpow(arr[0], MOD - 2);
        return;
    }
    calc_inv(arr, brr, n >> 1);
    int lg = get_lg(n << 1);
    static Z ta[SZ], tb[SZ];
    memset(ta, 0, sizeof(Z) << lg);
    memset(tb, 0, sizeof(Z) << lg);
    copy(arr, arr +  n      , ta);
    copy(brr, brr + (n >> 1), tb);
```

```
    fft(ta, lg, 0);
    fft(tb, lg, 0);
    for (int i = 0, _ = 1 << lg; i < _; ++i)
        ta[i] = (2 - ta[i] * tb[i]) * tb[i];
    fft(ta, lg, 1);
    copy(ta, ta + n, brr);
}
Poly calc_inv(const Poly &f) {
    static Z a[SZ], b[SZ];
    int lg = get_lg(f.size());
    memset(a, 0, sizeof(Z) << lg);
    copy(f.begin(), f.end(), a);
    calc_inv(a, b, 1 << lg);
    return Poly(b, b + f.size());
}
Poly operator / (const Poly &f, const Poly &g) {
    if (f.size() < g.size()) return Poly();
    Poly tf = f; reverse(tf.begin(), tf.end());
    Poly tg = g; reverse(tg.begin(), tg.end());
    tg.resize(f.size() - g.size() + 1);
    Poly ret = tf * calc_inv(tg);
    ret.resize(f.size() - g.size() + 1);
    reverse(ret.begin(), ret.end());
    return ret;
}
Poly& operator /= (Poly &f, const Poly &g) {
    return f = f / g;
}
Poly operator % (const Poly &f, const Poly &g) {
    Poly ret = f - (f / g) * g;
    ret.resize(g.size() - 1);
    return ret;
}
Poly& operator %= (Poly &f, const Poly &g) {
    return f = f % g;
}
Poly calc_der(const Poly &f) {
    Poly ret(f.size() - 1);
    for (u32 i = 1; i < f.size(); ++i) ret[i - 1]
↪   = f[i] * i;
    return ret;
}
Poly calc_pri(const Poly &f) {
    prep();
    Poly ret(f.size() + 1);
    for (u32 i = 1; i <= f.size(); ++i) ret[i] =
↪   f[i - 1] * inv[i];
    return ret;
}
Poly calc_ln(const Poly &f) {
    assert(f[0].v == 1);
    Poly g = calc_der(f) * calc_inv(f);
    g.resize(f.size() - 1);
    return calc_pri(g);
}
Poly calc_exp(int arr[], int n) {
    if (n == 1) {
        assert(arr[0] == 0);
        return Poly{1};
    }
    Poly f = calc_exp(arr, n >> 1);
    Poly tf = f;
    tf.resize(n);
    Poly a = Poly(arr, arr + n);
    Poly g = f * (Poly{1} - calc_ln(tf) + a);
    g.resize(n);
    return g;
}
Poly calc_exp(const Poly &f) {
    static int a[SZ];
    int lg = get_lg(f.size());
    memset(a, 0, sizeof(int) << lg);
    sp_copy(f.begin(), f.end(), a);
    Poly ret = calc_exp(a, 1 << lg);
    ret.resize(f.size());
    return ret;
}
Poly operator ^ (const Poly &f, const int &e) {
    u32 trail = 0;
    for (u32 i = 0; i < f.size(); ++i)
        if (f[i].v) break; else ++trail;
    if ((i64) trail * e >= f.size())
        return Poly(f.size(), 0);
    Z lst = f[trail], inv = qpow(lst, MOD - 2);
    Poly g;
    for (u32 i = trail; i < f.size(); ++i)
        g.emplace_back(f[i] * inv);
    Poly ret = calc_exp(calc_ln(g) * e) *
↪   qpow(lst, e);
    Poly t0 = Poly(trail * e, 0);
    ret.insert(ret.begin(), t0.begin(), t0.end());
    ret.resize(f.size());
    return ret;
}
Poly& operator ^= (Poly &f, const int &e) {
    return f = f ^ e;
}
```

4.2 牛顿迭代

问题描述：给出多项式 $G(x)$，求解多项式 $F(x)$ 满足：

$$G(F(x)) \equiv 0 \pmod{x^n}$$

答案只需要精确到 $F(x) \bmod x^n$ 即可。
实现原理：考虑倍增，假设有：

$$G(F_t(x)) \equiv 0 \pmod{x^t}$$

对 $G(F_{t+1}(x))$ 在模 x^{2t} 意义下进行 Taylor 展开：

$$G(F_{t+1}(x)) \equiv G(F_t(x)) + \frac{G'(F_t(x))}{1!}(F_{t+1}(x) - F_t(x)) \pmod{x^{2t}}$$

那么就有：

$$F_{t+1}(x) \equiv F_t(x) - \frac{G(F_t(x))}{G'(F_t(x))} \pmod{x^{2t}}$$

注意事项：$G(F(x))$ 的常数项系数必然为 0，这个可以作为求解的初始条件。
多项式求逆原理：令 $G(x) = x * A - 1$（其中 A 是一个多项式系数），根据牛顿迭代法有：

$$\begin{aligned} F_{t+1}(x) &\equiv F_t(x) - \frac{F_t(x) * A(x) - 1}{A(x)} \\ &\equiv 2F_t(x) - F_t(x)^2 * A(x) \pmod{x^{2t}} \end{aligned}$$

注意事项：

1. $F(x)$ 的常数项系数必然不为 0，否则没有逆元；
2. 复杂度是 $O(n\log n)$ 但是常数比较大 (10^5 大概需要 0.3 秒左右)；
3. 传入的两个数组必须不同，但传入的次数界没有必要是 2 的次幂；

多项式取指数和对数作用：给出一个多项式 $A(x)$，求一个多项式 $F(x)$ 满足 $e^A(x) - F(x) \equiv 0 \pmod{x^n}$。
原理：令 $G(x) = \ln x - A$（其中 A 是一个多项式系数），根据牛顿迭代法有:

$$F_{t+1}(x) \equiv F_t(x) - F_t(x)(\ln F_t(x) - A(x)) \pmod{x^{2t}}$$

求 $\ln F_t(x)$ 可以用先求导再积分的办法，即:

$$\ln A(x) = \int \frac{F'(x)}{F(x)}\, \mathrm{d}x$$

多项式的求导和积分可以在 $O(n)$ 的时间内完成，因此总复杂度为 $O(n\log n)$。
应用：加速多项式快速幂。
注意事项：

1. 进行 log 的多项式必须保证常数项系数为 1，否则必须要先求出 $\log a[0]$ 是多少；
2. 传入的两个数组必须不同，但传入的次数界没有必要是 2 的次幂；
3. 常数比较大，10^5 的数据求指数和对数分别需要 0.37s 和 0.85s 左右的时间，注意这里 memset 几乎不占用时。

4.3 MTT

```
// N: size * 4
// MOD
const size_t N = 1 << 18;
const int MOD = 1E9 + 7;
struct Complex {
    double a, b;
    Complex() {}
    Complex(double a, double b) : a(a), b(b) {}
    Complex operator + (const Complex &c) const {
        return Complex(a + c.a, b + c.b);
    }
    Complex operator - (const Complex &c) const {
        return Complex(a - c.a, b - c.b);
    }
    Complex operator * (const Complex &c) const {
        return Complex(a * c.a - b * c.b, a * c.b
↪   + b * c.a);
    }
    Complex conj() const {
        return Complex(a, -b);
    }
} w[N];
void prep() {
    const double PI = acos(-1);
    for (int i = 0; i <= N >> 1; ++i) {
        double ang = 2 * i * PI / N;
        w[i] = Complex(cos(ang), sin(ang));
    }
}
struct _ {
    _() { prep(); }
} __;
void fft(Complex a[], int lg) {
    int n = 1 << lg;
    static int rev[N], rev_lg = -1;
    if (rev_lg != lg) {
        for (int i = 0; i < n; ++i)
            rev[i] = rev[i >> 1] >> 1 | ((i & 1)
↪   << lg >> 1);
        rev_lg = lg;
    }
    for (int i = 0; i < n; ++i)
        if (i < rev[i]) swap(a[i], a[rev[i]]);
    for (int m = 1, l = 2; m < n; m <<= 1, l <<=
↪   1) {
        static Complex ww[N];
        for (int i = 0, j = 0, step = N / l; i <
↪   m; ++i, j += step)
            ww[i] = w[j];
        Complex *xx = a, *yy = a + m, x, y;
        for (int i = 0, j; i < n; i += l) {
            for (j = 0; j < m; ++j, ++xx, ++yy) {
                x = *xx; y = *yy * ww[j];
                *xx = x + y;
                *yy = x - y;
            }
            xx += m;
            yy += m;
        }
    }
}
void mul(int a[], int b[], int c[], int n, int m)
↪   {
    static Complex d[N], e[N], f[N], g[N];
    int lg = 0;
    while ((1 << lg) < n + m) ++lg;
    int tot = 1 << lg;
    for (int i = 0; i < n; ++i)
        d[i] = Complex(a[i] & 32767, a[i] >> 15);
    for (int i = 0; i < m; ++i)
        e[i] = Complex(b[i] & 32767, b[i] >> 15);
    fft(d, lg); fft(e, lg);
    for (int i = 0; i < tot; ++i) {
        int j = i ? tot - i : 0;
        Complex da = (d[i] + d[j].conj()) *
↪   Complex(.5,  0);
        Complex db = (d[i] - d[j].conj()) *
↪   Complex(0, -.5);
        Complex dc = (e[i] + e[j].conj()) *
↪   Complex(.5,  0);
        Complex dd = (e[i] - e[j].conj()) *
↪   Complex(0, -.5);
        f[j] = da * dc + da * dd * Complex(0, 1);
        g[j] = db * dc + db * dd * Complex(0, 1);
    }
    fft(f, lg); fft(g, lg);
    for (int i = 0; i < n + m - 1; ++i) {
```

```
        i64 da = round(f[i].a / tot); da %= MOD;
        i64 db = round(f[i].b / tot); db %= MOD;
        i64 dc = round(g[i].a / tot); dc %= MOD;
        i64 dd = round(g[i].b / tot); dd %= MOD;
        c[i] = (da + ((db + dc) << 15) + (dd << 30)) % MOD;
    }
}
```

4.4 FWT

```
// N: size * 2
const size_t N = 1 << 17;
void div2(Z &x) {
    if (x.v & 1) x.v += MOD;
    x.v >>= 1;
}
void fwt_and(Z a[], int n, bool rev) {
    for (int m = 1, l = 2; m < n; m <<= 1, l <<= 1)
        for (int i = 0; i < n; i += l)
            for (int j = 0; j < m; ++j)
                if (rev) a[i + j] -= a[i + j + m];
                else a[i + j] += a[i + j + m];
}
void fwt_or(Z a[], int n, bool rev) {
    for (int m = 1, l = 2; m < n; m <<= 1, l <<= 1)
        for (int i = 0; i < n; i += l)
            for (int j = 0; j < m; ++j)
                if (rev) a[i + j + m] -= a[i + j];
                else a[i + j + m] += a[i + j];
}
void fwt_xor(Z a[], int n, bool rev) {
    for (int m = 1, l = 2; m < n; m <<= 1, l <<= 1)
        for (int i = 0; i < n; i += l)
            for (int j = 0; j < m; ++j) {
                Z xx = a[i + j], yy = a[i + j + m];
                a[i + j] = xx + yy;
                a[i + j + m] = xx - yy;
                if (rev) {
                    div2(a[i + j]);
                    div2(a[i + j + m]);
                }
            }
}
```

4.5 BM

```
// N: size * 2
const size_t N = 1E4 + 5;
using Poly = vector<Z>;
namespace Rec {
u64 tmp[N];
void mul(Z a[], Z b[], Z c[], int n, int m) {
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j)
            if ((tmp[i + j] += (u64) a[i].v * b[j].v) >> 62)
                tmp[i + j] %= MOD;
    for (int i = 0; i < n + m - 1; ++i) {
        c[i] = tmp[i] % MOD; tmp[i] = 0;
    }
}
void get_mod(Z a[], Z b[], Z c[], int n, int m) {
    static Z tc[N];
    copy(a, a + n, tc);
    Z iv = qpow(b[m - 1], MOD - 2);
    for (int i = n; i-- >= m; ) {
        Z mul = tc[i] * iv;
        for (int j = m, k = i; j--; --k)
            tc[k] -= mul * b[j];
    }
    copy(tc, tc + m - 1, c);
}
void _solve(Z a[], Z b[], i64 n, int m) {
    if (n < m - 1) {
        b[n] = 1; return;
    }
    static Z ta[N], tb[N];
    if (n & 1) {
        _solve(a, b, n - 1, m);
        ta[1] = 1;
        mul(b, ta, tb, m, 2);
        get_mod(tb, a, b, m + 1, m);
    } else {
        _solve(a, b, n >> 1, m);
        mul(b, b, tb, m, m);
        get_mod(tb, a, b, (m << 1) - 1, m);
    }
}
Z solve(const Poly &init, const Poly &a, i64 n) {
    int m = a.size();
    static Z ta[N], b[N];
    for (int i = 0; i < m; ++i)
        ta[i] = 0 - a[m - 1 - i];
    ta[m] = 1;
    _solve(ta, b, n, m + 1);
    Z ans = 0;
    for (int i = 0; i < m; ++i)
        ans += init[i] * b[i];
    return ans;
}
}

namespace BM {
Poly& operator += (Poly &p, const Poly &q) {
    if (q.size() > p.size()) p.resize(q.size());
    for (size_t i = 0; i < q.size(); ++i)
        p[i] += q[i];
    return p;
}
Poly operator * (const Poly &p, Z x) {
    Poly ret(p.size());
    for (size_t i = 0; i < p.size(); ++i)
        ret[i] = p[i] * x;
    return ret;
}
Poly solve(const Poly &a) {
    Poly P, R; int cnt = 1;
```

```
    for (size_t i = 0; i < a.size(); ++i) {
        Poly tmp = P; tmp.insert(begin(tmp), MOD -
↪   1);
        Z delta = 0;
        for (size_t j = 0; j < tmp.size(); ++j)
            delta += tmp[j] * a[i - j];
        if (delta.v) {
            vector<Z> t(cnt);
            R.insert(begin(R), begin(t), end(t));
            P += R * (MOD - delta);
            R = tmp * qpow(delta, MOD - 2);
            cnt = 0;
        } else {
            ++cnt;
        }
    }
    for (size_t i = P.size(); i < a.size(); ++i) {
        Z cur = 0;
        for (size_t j = 0; j < P.size(); ++j)
            cur += a[i - 1 - j] * P[j];
        assert(cur.v == a[i].v);
    }
    return P;
}
}
int main() {
    vector<Z> p(read());
    i64 m = read();
    generate(begin(p), end(p), read);
    Poly P = BM::solve(p);
    for (Z x : P) cout << x << ' ';
    cout << '\n';
    cout << Rec::solve(p, P, m) << '\n';
    return 0;
}
```

4.6 numbers

4.6.1 伯努利数

伯努利数满足

$$B_0 = 1, \sum_{j=0}^{m}\binom{m+1}{j}B_j = 0 \ (m > 0).$$

等式两边同时加上 B_{m+1}，并设 $n = m - 1$，得

$$\sum_{i=0}^{n}\binom{n}{i}B_i = [n = 1] + B_n$$

设 $\hat{B}(x) = \sum_{i=0}^{\infty} B_i \cdot \frac{x^i}{i!}$，则

$$\hat{B}(x)e^x = x + \hat{B}(x) \Rightarrow \hat{B}(x) = \frac{x}{e^x - 1}$$

$$\begin{aligned}
&0^k + 1^k + \ldots + n^k \\
&= k!\left[x^k\right]\frac{e^{(n+1)x} - 1}{x} \cdot \hat{B}(x) \\
&= k!\sum_{i=0}^{k}\frac{B_i}{i!} \cdot \frac{(n+1)^{k-i+1}}{(k-i+1)!} \\
&= \frac{1}{k+1}\sum_{i=0}^{k}\binom{k+1}{i}B_i \cdot (n+1)^{k-i+1}
\end{aligned}$$

4.6.2 第一类斯特林数

记 $S_1(n, k)$ 为将 n 个不同元素分为 k 个环排列的方案数. 由组合意义得,

$$S_1(n, k) = S_1(n-1, k-1) + (n-1)S_1(n-1, k)$$

$$x^{\overline{n}} = \sum_{i=0}^{n} S_1(n, i)x^i$$

$$x^{\underline{n}} = \sum_{i=0}^{n}(-1)^{n-i}S_1(n, i)x^i$$

$$\sum_{i=0}^{n} S_1(n, i)x^i = \prod_{i=0}^{n-1}(x + i)$$

注意最后等式的右半部分,可以使用递增 + 点值平移 $O(n\log n)$ 求出第 n 行斯特林数.

4.6.3 第二类斯特林数

记 $S_2(n, k)$ 为将 n 个不同元素分至 k 个相同的盒子（每个盒子至少一个元素）的方案数. 由组合意义得,

$$S_2(n, k) = S_2(n-1, k-1) + kS_2(n-1, k)$$

$$x^n = \sum_{i=0}^{n} S_2(n, i)x^{\underline{i}}$$

$$S_2(n, k) = \sum_{i=0}^{k}(-1)^{k-i}\binom{k}{i}i^n$$

$$\frac{S_2(n, k)}{k!} = \sum_{i=0}^{k}\frac{i^n}{i!} \cdot \frac{(-1)^{k-i}}{(k-i)!}$$

是一个卷积的形式，可以 FFT 求出某一行第二类斯特林数.

4.6.4 斯特林反演

$$\begin{aligned}
x^n &= \sum_{i=0}^{n} S_2(n, i)x^{\underline{i}} \\
&= \sum_{i=0}^{n} S_2(n, i)\sum_{j=0}^{i}(-1)^{i-j}S_1(i, j)x^j \\
&= \sum_{i=0}^{n} x^i\sum_{j=i}^{n}(-1)^{j-i}S_2(n, j)S_1(j, i)
\end{aligned}$$

设

$$g_n = \sum_{i=0}^{n} S_2(n, i)f_i,$$

则

$$f_n = \sum_{i=0}^{n}(-1)^{n-i}S_1(n, i)g_i.$$

4.6.5 Burnside 引理

设置换群为 G，染色集合为 X.

若染色 $x \in X$ 在置换 f 的作用下得到染色 $y \in X$，则称 x, y 等价. 由置换群的定义，我们可以得到等价类，使得等价类内任意两个染色等价.

设 $X^g (g \in G)$ 表示在置换 g 下的不动点，即

$$X^g = \{x \mid x \in X, gx = x\}.$$

则等价类个数

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|.$$

例 LOJ 6538 烷基计数，对于一棵有根树，每个节点至多三个儿子，且这些儿子排列同构. 求有多少个 n 个节点的等价类.

考虑其生成函数 $f(x)$. 根节点有 3 个儿子（儿子可以为空，因为循环同构，我们不需讨论 $0, 1, 2$ 个儿子的情况），排列的置换群有 6 种，其中 $(1, 2, 3)$ 染色方案数为 $f(x)^3$，$(1, 3, 2), (2, 1, 3), (3, 2, 1)$ 染色方案为 $f(x^2) f(x)$，$(2, 3, 1), (3, 1, 2)$ 染色方案为 $f(x^3)$. 所以

$$f(x) = x \times \frac{f(x)^3 + 3f(x^2)f(x) + 2f(x^3)}{6} + 1.$$

牛顿迭代.

4.6.6 五边形数

$\prod_{n=1}^{\infty} (1 - x^n) = \sum_{n=0}^{\infty} (-1)^n (1 - x^{2n+1}) x^{n(3n+1)/2}$

4.7 树的计数

1. 有根树计数：$n + 1$ 个结点的有根树的个数为

$$a_{n+1} = \frac{\sum_{j=1}^{n} j \cdot a_j \cdot S_{n,j}}{n}$$

其中，

$$S_{n,j} = \sum_{i=1}^{n/j} a_{n+1-ij} = S_{n-j,j} + a_{n+1-j}$$

2. 无根树计数：当 n 为奇数时，n 个结点的无根树的个数为

$$a_n - \sum_{i=1}^{n/2} a_i a_{n-i}$$

当 n 为偶数时，n 个结点的无根树的个数为

$$a_n - \sum_{i=1}^{n/2} a_i a_{n-i} + \frac{1}{2} a_{\frac{n}{2}} (a_{\frac{n}{2}} + 1)$$

3. n 个结点的完全图的生成树个数为

$$n^{n-2}$$

4. 矩阵－树定理：图 G 由 n 个结点构成，设 $\boldsymbol{A}[G]$ 为图 G 的邻接矩阵、$\boldsymbol{D}[G]$ 为图 G 的度数矩阵，则图 G 的不同生成树的个数为 $\boldsymbol{C}[G] = \boldsymbol{D}[G] - \boldsymbol{A}[G]$ 的任意一个 $n - 1$ 阶主子式的行列式值。

5 数论

5.1 判素数（miller-rabin）

```
i64 Rand() {
    return (i64) rand() * rand() + rand();
};

i64 mul_mod(i64 a, i64 b, i64 mod) {
    i64 tmp = (long double) a * b / mod;
    i64 ret = a * b - tmp * mod;
    while (ret >= mod) ret -= mod;
    while (ret < 0) ret += mod;
    return ret;
};

i64 pow_mod(i64 base, i64 e, i64 mod) {
    i64 ret = 1;
    for (; e; e >>= 1) {
        if (e & 1) ret = mul_mod(ret, base, mod);
        base = mul_mod(base, base, mod);
    }
    return ret;
};

const int pri[] {
    2, 3, 5, 7, 11, 13, 17, 19, 23, 29
};

bool isp(i64 num) {
    for (int x : pri) if (num == x) return true;
    i64 a = num - 1;
    int b = 0;
    while (!(a & 1)) {
        a >>= 1; ++b;
    }
    for (int p : pri) {
        i64 x = pow_mod(p, a, num), y = x;
        for (int i = 0; i < b; ++i) {
            y = mul_mod(x, x, num);
            if (y == 1 && x != 1 && x != num - 1)
                return false;
            x = y;
        }
        if (y != 1) return false;
    }
    return true;
}

vector<i64> fac;

i64 gcd(i64 a, i64 b) {
    return b ? gcd(b, a % b) : a;
}

void rho(i64 n) {
    if (isp(n)) {
        fac.emplace_back(n);
        return;
    }
    while (true) {
```

```
        i64 x0 = Rand() % n, x1 = x0, d = 1, c =
↪   Rand() % n, cnt = 0;
        while (d == 1) {
            x0 = (mul_mod(x0, x0, n) + c) % n;
            d = gcd(abs(x1 - x0), n);
            ++cnt;
            if (!(cnt & (cnt - 1))) x1 = x0; //
↪   Floyd 倍增判环
        }
        if (d < n) {
            rho(d); rho(n / d); return;
        }
    }
}
```

5.2 二次剩余（Cipolla）

欧拉判定:

$$x^{\frac{p-1}{2}} \equiv \left(\frac{x}{p}\right) \pmod p$$

```
// input mod
// method: cipolla(int n)
int mod;
namespace Cipolla {
int omega;
int sqr(int x) {
    return (i64) x * x % mod;
}
struct Number {
    int x, y;
    Number() {}
    Number(int x, int y = 0) : x(x), y(y) {}
    Number operator * (const Number &n) const {
        Number ret;
        ret.x = ((i64) x * n.x + (i64) y * n.y %
↪   mod * omega) % mod;
        ret.y = ((i64) x * n.y + (i64) y * n.x) %
↪   mod;
        return ret;
    }
    Number& operator *= (const Number &n) {
        return *this = *this * n;
    }
};
Number npow(Number base, int e) {
    Number ret(1);
    for (; e; e >>= 1) {
        if (e & 1) ret *= base;
        base *= base;
    }
    return ret;
}
int get_num(int n) {
    while (true) {
        int x = rand();
        int tmp = (sqr(x) - n) % mod;
        if (tmp < 0) tmp += mod;
        if (qpow(tmp, (mod - 1) / 2) == mod - 1) {
            omega = tmp;
            return x;
        }
    }
}
int cipolla(int n) {
    if (!n) return 0;
    if (qpow(n, (mod - 1) / 2) != 1) {
        return -1;
    }
    int a = get_num(n);
    Number res = npow(Number(a, 1), (mod + 1) /
↪   2);
    assert(!res.y);
    return res.x;
}
}
```

5.3 杜教筛

```
// prep_calc[N]: pre-calculated
map<i64, i64> mp;
i64 calc(i64 n) {
    if (n < N) return pre_calc[n];
    if (mp.count(n)) return mp[n];
    i64 ret = 1LL * n * (n + 1) / 2; // 这里改成
↪   (f * g) 的前缀和
    for (i64 l = 2, r; l <= n; l = r) {
        r = n / (n / l) + 1;
        ret -= (r - l) * calc(n / l); // 这里 r -
↪   l 改成 g 在 [l, r] 的和
    }
    return mp[n] = ret;
}
```

5.4 min_25

```
const size_t N = 2E5 + 5; // 2 * sqrt(N)

i64 n, lim, val[N];
int id1[N], id2[N];
bool npr[N]; int pri[N], pcnt; Z pg0[N], pg1[N];
Z g0[N], g1[N];

void prep() {
    for (int i = 2; i < (int) N; ++i) {
        if (!npr[i]) {
            pri[++pcnt] = i;
            pg0[pcnt] = pg0[pcnt - 1] - 1;
            pg1[pcnt] = pg1[pcnt - 1] + i;
        }
        for (int j = 1, k; j <= pcnt && (k = i *
↪   pri[j]) < (int) N; ++j) {
            npr[k] = true;
            if (i % pri[j] == 0) break;
        }
    }
}

int get_id(i64 x) {
    return x <= lim ? id1[x] : id2[n / x];
}
```

```
Z calc_f(int p, int c) {
    return p ^ c;
}

Z S(i64 n, int x) {
    // 求 \sum f(1 ~ n 中最小质因子 >= pri[x])
    if (n <= 1 || pri[x] > n) return 0;
    Z ret = g0[get_id(n)] + g1[get_id(n)];
    if (x == 1) ret += 2; // #6035 特殊 f(2) = 2 +
↪   1 = 3 != 1
    ret -= pg0[x - 1] + pg1[x - 1];
    // 当前 ret 为 \sum f(1 ~ n 中 >= pri[x] 的质
↪   数)
    for (int k = x; k <= pcnt; ++k) {
        i64 p1 = pri[k], p2 = p1 * pri[k];
        if (p2 > n) break;
        for (int e = 1; p2 <= n; p2 = (p1 = p2) *
↪   pri[k], ++e) {
            ret += S(n / p1, k + 1) *
↪   calc_f(pri[k], e);
            ret += calc_f(pri[k], e + 1);
        }
    }
    return ret;
}

int main() {
    n = read();
    lim = sqrt(n + .5);
    prep();
    int cnt = 0;
    for (i64 i = 1, j; i <= n; i = j + 1) {
        i64 t;
        j = n / (t = val[++cnt] = n / i);
        (t <= lim ? id1[t] : id2[i]) = cnt;
        t %= MOD;
        g0[cnt] = 1 - Z(t);
        g1[cnt] = (t - 1) * (t + 2) / 2 % MOD;
    }
    for (int i = 1; i <= pcnt; ++i) {
        // 筛掉最小质因子为 pri[i] 的数
        i64 bnd = (i64) pri[i] * pri[i];
        if (bnd > n) break;
        for (int j = 1, id; val[j] >= bnd; ++j) {
            id = get_id(val[j] / pri[i]);
            g0[j] -= (g0[id] - pg0[i - 1]);
            g1[j] -= (g1[id] - pg1[i - 1]) *
↪   pri[i];
        }
    }
    // g[i] = \sum 1~val[i] 中质数
    cout << 1 + S(n, 1) << '\n';
    return 0;
}
```

5.5 佩尔方程

```
// Blazar
ULL A,B,p[maxn],q[maxn],a[maxn],g[maxn],h[maxn];
int main() {
  for (int test=1, n;scanf("%d",&n) && n;++test) {
    printf("Case %d: ",test);
    if (fabs(sqrt(n)-floor(sqrt(n)+1e-7))<=1e-7)
↪   {
      int a=(int)(floor(sqrt(n)+1e-7)); printf("%d
↪   %d\n",a,1);
    } else {
      // 求 $x^2-ny^2=1$ 的最小正整数根, n 不是完全
↪   平方数
      p[1]=q[0]=h[1]=1;p[0]=q[1]=g[1]=0;
      a[2]=(int)(floor(sqrt(n)+1e-7));
      for (int i=2;i;++i) {
        g[i]=-g[i-1]+a[i]*h[i-1];
↪   h[i]=(n-sqr(g[i]))/h[i-1];
        a[i+1]=(g[i]+a[2])/h[i];
↪   p[i]=a[i]*p[i-1]+p[i-2];
        q[i]=a[i]*q[i-1]+q[i-2];
        if
↪   (sqr((ULL)(p[i]))-n*sqr((ULL)(q[i]))==1){
          A=p[i];B=q[i];break;
        }
      }
      cout << A << ' ' << B <<endl;
    }
  }
}
```

5.6 直线下整点个数

```
// Quasar
// calc \sum_{i=0}^{n-1} [(a+bi)/m]
// n,a,b,m > 0
LL solve(LL n, LL a, LL b, LL m) {
    if(b == 0)
        return n * (a / m);
    if(a >= m || b >= m)
        return n * (a / m) + (n - 1) * n / 2 * (b
↪   / m) + solve(n, a % m, b % m, m);
    return solve((a + b * n) / m, (a + b * n) % m,
↪   m, b);
}
```

5.7 定理

5.7.1 扩展欧拉定理

$$a^b \equiv \begin{cases} a^{b \bmod \varphi(m)} & (\gcd(a,m)=1) \\ a^b & (\gcd(a,m)\neq 1, b<\varphi(m)) \\ a^{(b \bmod \varphi(m))+\varphi(m)} & (\gcd(a,m)\neq 1, b\geq\varphi(m)) \end{cases} \pmod m$$

5.7.2 卢卡斯定理

$\forall$ 质数 $p, n, m \in \mathbb{N}^+$,

$$\binom{n}{m} \equiv \binom{\lfloor n/p \rfloor}{\lfloor m/p \rfloor}\binom{n \bmod p}{m \bmod p} \pmod p$$

5.7.3 威尔逊定理

对于质数 p，有 $(p-1)! \equiv -1 \pmod p$（证明 $2,3,\ldots,p-2$ 可以逆元两两配对）高斯的扩展：

$$\prod_{1\le k\le m,\gcd(k,m)=1} k \equiv \begin{cases} -1, & \text{if } m=4,p^\alpha,2p^\alpha, \\ 1, & \text{otherwise.} \end{cases} \pmod m$$

5.7.4 莫比乌斯函数

$$\mu(n)=\begin{cases} 1 & 若n=1 \\ (-1)^k & 若n无平方数因子，且n=p_1p_2\ldots p_k \\ 0 & 若n有大于1的平方数因数 \end{cases}$$

$$\sum_{d|n}\mu(d)=\begin{cases} 1 & 若n=1 \\ 0 & 其他情况 \end{cases}$$

$$g(n)=\sum_{d|n}f(d) \Leftrightarrow f(n)=\sum_{d|n}\mu(d)g(\frac{n}{d})$$

$$g(x)=\sum_{n=1}^{[x]}f(\frac{x}{n}) \Leftrightarrow f(x)=\sum_{n=1}^{[x]}\mu(n)g(\frac{x}{n})$$

6 线性代数

6.1 线性基

```cpp
// N: size
const size_t N = 50;
u64 base[N];
void add(u64 val) {
    for (int i = 49; ~i; --i) if (val >> i & 1)
        if (!base[i]) {
            for (int j = 0; j < i; ++j) if (val >> j & 1) val ^= base[j];
            base[i] = val;
            for (int j = i + 1; j < 50; ++j) if (base[j] >> i & 1) base[j] ^= val;
            break;
        } else {
            val ^= base[i];
        }
}
```

6.2 矩阵求逆

```cpp
struct Matrix {
    size_t n, m;
    vector<vector<Z>> a;
    Matrix() {}
    Matrix(size_t n, size_t m) : n(n), m(m) {
        a = vector<vector<Z>>(n, vector<Z>(m));
    }
    void do_diag(Z x) {
        for (size_t i = 0; i < n && i < m; ++i) a[i][i] = x;
    }
    Matrix& operator += (const Matrix &mat) {
        assert(n == mat.n && m == mat.m);
        for (size_t i = 0; i < n; ++i) for (size_t j = 0; j < m; ++j) a[i][j] += mat.a[i][j];
        return *this;
    }
    Matrix operator + (const Matrix &mat) const {
        Matrix ret = *this; return ret += mat;
    }
    Matrix operator * (const Matrix &mat) const {
        assert(m == mat.n);
        Matrix ret(n, mat.m);
        for (size_t i = 0; i < n; ++i)
            for (size_t j = 0; j < mat.m; ++j)
                for (size_t k = 0; k < m; ++k)
                    ret.a[i][j] += a[i][k] * mat.a[k][j];
        return ret;
    }
    Matrix& operator *= (const Z &x) {
        for (size_t i = 0; i < n; ++i) for (size_t j = 0; j < m; ++j) a[i][j] *= x;
        return *this;
    }
    Matrix operator * (const Z &x) const {
        Matrix ret = *this; return ret *= x;
    }
    Matrix& operator *= (const Matrix &mat) { return *this = *this * mat; }
    Matrix get_inv() const {
        assert(n == m);
        Matrix m = *this, r(n, n); r.do_diag(1);
        for (size_t i = 0; i < n; ++i) {
            int pivot = -1;
            for (size_t j = i; j < n; ++j) if (m.a[j][i].v && !~pivot) pivot = j;
            assert(~pivot);
            for (size_t j = i; j < n; ++j) {
                swap(m.a[i][j], m.a[pivot][j]);
                swap(r.a[i][j], r.a[pivot][j]);
            }
            Z mul = qpow(m.a[i][i], MOD - 2);
            for (size_t j = 0; j < n; ++j) { // 矩阵求逆时切勿从 i 开始枚举
                m.a[i][j] *= mul; r.a[i][j] *= mul;
            }
            for (size_t j = 0; j < n; ++j) {
                if (j == i) continue;
                Z mul = m.a[j][i]; if (!mul.v) continue;
                for (size_t k = 0; k < n; ++k) {
                    m.a[j][k] -= mul * m.a[i][k];
                    r.a[j][k] -= mul * r.a[i][k];
                }
                assert(!m.a[j][i].v);
            }
        }
        for (size_t i = 0; i < n; ++i) {
            for (size_t j = 0; j < n; ++j) assert(m.a[i][j].v == (i == j));
        }
```

```cpp
        return r;
    }
};

Matrix qpow(Matrix base, int e) {
    Matrix ret(2, 2); ret.do_diag(1);
    for (; e; e >>= 1) {
        if (e & 1) ret *= base;
        base *= base;
    }
    return ret;
}

ostream& operator << (ostream &os, const Matrix &mat) {
    for (size_t i = 0; i < mat.n; ++i) {
        for (size_t j = 0; j < mat.m; ++j) os << mat.a[i][j] << ' ';
        os << '\n';
    }
    return os;
}
```

6.3 矩阵特征多项式

```cpp
// nflsoj 333
using Poly = vector<Z>;
Poly& operator -= (Poly &p, const Poly &q) {
    if (q.size() > p.size()) p.resize(q.size());
    for (u32 i = 0; i < q.size(); ++i) p[i] -= q[i];
    return p;
}
Poly operator * (const Poly &p, const Z &v) {
    Poly ret(p.size());
    for (u32 i = 0; i < p.size(); ++i) ret[i] = p[i] * v;
    return ret;
}

Poly charac_poly(vector<Poly> mat) {
    int n = (int) mat.size();
    assert(n == (int) mat[0].size());
    for (int j = 1; j < n; ++j) {
        if (!mat[j][j - 1].v) {
            for (int i = j + 1; i < n; ++i) {
                if (mat[i][j - 1].v) {
                    for (int p = 0; p < n; ++p) swap(mat[i][p], mat[j][p]);
                    for (int p = 0; p < n; ++p) swap(mat[p][i], mat[p][j]);
                    break;
                }
            }
        }
        Z inv = qpow(mat[j][j - 1], MOD - 2);
        for (int k = j + 1; k < n; ++k) {
            Z u = mat[k][j - 1] * inv;
            for (int p = 0; p < n; ++p) mat[k][p] -= u * mat[j][p];
            for (int p = 0; p < n; ++p) mat[p][j] += u * mat[p][k];
        }
    }
    vector<Poly> p(1, Poly(1, 1));
    for (int k = 0; k < n; ++k) {
        Poly po = p.back();
        po.insert(begin(po), 0);
        po -= p.back() * mat[k][k];
        for (int i = 0; i < k; ++i) {
            Z mul = mat[i][k];
            for (int j = i; j < k; ++j) mul *= mat[j + 1][j];
            po -= p[i] * mul;
        }
        p.emplace_back(po);
    }
    return p.back();
}
```

7 数据结构

7.1 左偏树

```cpp
// N
struct Node {
    int lc, rc, val, dis;
    Node() {}
} t[N];
int arr[N], rt[N];
bool del[N];
int merge(int x, int y) {
    if (!x || !y) return x | y;
    if (arr[y] < arr[x]) swap(x, y);
    t[x].rc = merge(t[x].rc, y);
    if (t[t[x].lc].dis < t[t[x].rc].dis)
        swap(t[x].lc, t[x].rc);
    t[x].dis = t[t[x].rc].dis + 1;
    return x;
}
```

7.2 LCT

```cpp
// N
const size_t N = 1E5 + 5;
int pa[N], ch[N][2], siz[N], val[N];
bool tag[N];
void update(int x) {
    swap(ch[x][0], ch[x][1]);
    tag[x] ^= 1;
}
void pushdown(int x) {
    if (tag[x]) {
        if (ch[x][0]) update(ch[x][0]);
        if (ch[x][1]) update(ch[x][1]);
        tag[x] = 0;
    }
}
void pushup(int x) {
    siz[x] = siz[ch[x][0]] + val[x] + siz[ch[x][1]];
```

```
}
int getd(int x) {
    return ch[pa[x]][0] == x ? 0 : ch[pa[x]][1] ==
↪   x ? 1 : -1;
}
void rotate(int x) {
    int y = pa[x], z = pa[y], k = getd(x);
    if (~getd(y)) ch[z][getd(y)] = x;
    pa[x] = z; pa[y] = x;
    ch[y][k] = ch[x][k ^ 1];
    ch[x][k ^ 1] = y;
    if (ch[y][k]) pa[ch[y][k]] = y;
    pushup(y);
}
void splay(int x) {
    static int stk[N];
    int y = x, tp = 0;
    stk[++tp] = y;
    while (~getd(y)) stk[++tp] = y = pa[y];
    while (tp) pushdown(stk[tp--]);
    while (~getd(x)) {
        y = pa[x];
        if (~getd(y))
            rotate(getd(x) ^ getd(y) ? x : y);
        rotate(x);
    }
    pushup(x);
}
void access(int x) {
    for (int y = 0; x; x = pa[y = x]) {
        splay(x);
        val[x] += siz[ch[x][1]];
        ch[x][1] = y;
        val[x] -= siz[ch[x][1]];
        pushup(x);
    }
}
void makeroot(int x) {
    access(x);
    splay(x);
    update(x);
}
void link(int x, int y) {
    makeroot(x);
    access(y); splay(y);
    pa[x] = y;
    val[y] += siz[x];
    pushup(y);
}
i64 split(int x, int y) {
    makeroot(y);
    access(x); splay(x);
    // x -> y is now a link from the root
    return (i64) (siz[x] - siz[y]) * siz[y];
}
```

7.3 KD-Tree

```
// N
using P = pair<int, int>;
#define fi first
#define se second
const size_t N = 2E5 + 5;
struct Node {
    int xl, yl, xm, ym, xr, yr;
    int lc, rc, pa;
    i64 sum, val, tag;
    int cnt; bool exist;
    Node() {}
} t[N];
int tot;
P point[N];
map<P, int> mp;
int build(int l, int r, bool d = 0, int pa = 0) {
    if (l > r) return 0;
    int x = ++tot;
    t[x].pa = pa;
    int mid = (l + r) >> 1;
    nth_element(point + l, point + mid, point + r
↪   + 1,
            [&](const P &p, const P &q) {
        P a = p, b = q;
        if (d) swap(a.fi, a.se), swap(b.fi, b.se);
        return a < b;
    });
    mp[point[mid]] = x;
    t[x].xl = t[x].xm = t[x].xr = point[mid].fi;
    t[x].yl = t[x].ym = t[x].yr = point[mid].se;
    if ((t[x].lc = build(l, mid - 1, d ^ 1, x))) {
        int y = t[x].lc;
        chkmin(t[x].xl, t[y].xl); chkmax(t[x].xr,
↪   t[y].xr);
        chkmin(t[x].yl, t[y].yl); chkmax(t[x].yr,
↪   t[y].yr);
    }
    if ((t[x].rc = build(mid + 1, r, d ^ 1, x))) {
        int y = t[x].rc;
        chkmin(t[x].xl, t[y].xl); chkmax(t[x].xr,
↪   t[y].xr);
        chkmin(t[x].yl, t[y].yl); chkmax(t[x].yr,
↪   t[y].yr);
    }
    return x;
}
void pushup(int x) {
    t[x].sum = t[t[x].lc].sum + t[t[x].rc].sum;
    if (t[x].exist) t[x].sum += t[x].val;
}
void update(int x, i64 v) {
    t[x].sum += v * t[x].cnt;
    t[x].val += v;
    t[x].tag += v;
}
void pushdown(int x) {
    if (t[x].tag) {
        if (t[x].lc) update(t[x].lc, t[x].tag);
        if (t[x].rc) update(t[x].rc, t[x].tag);
        t[x].tag = 0;
    }
}
void link_pd(int x) {
    static int stk[N];
```

```
    int tp = 0;
    for (; x; x = t[x].pa) stk[++tp] = x;
    while (tp) pushdown(stk[tp--]);
}
void modify(int x, int a, int b, int val) {
    if (!x || t[x].xr < a || t[x].yr < b) return;
    if (t[x].xl >= a && t[x].yl >= b) return
↪   update(x, val);
    pushdown(x);
    if (t[x].xm >= a && t[x].ym >= b) t[x].val +=
↪   val;
    modify(t[x].lc, a, b, val);
    modify(t[x].rc, a, b, val);
    pushup(x);
}
void doit(int x, int y, int d) {
    int u = mp[{x, y}];
    link_pd(u);
    i64 e = t[u].val * d;
    t[u].exist ^= 1;
    for (; u; u = t[u].pa) {
        t[u].cnt += d;
        t[u].sum += e;
    }
    modify(1, x + 1, y + 1, d);
}
void query(int x, int a, int b, i64 &sum, int
↪   &cnt) {
    if (!x || t[x].xl > a || t[x].yl > b) return;
    if (t[x].xr <= a && t[x].yr <= b) {
        sum += t[x].sum;
        cnt += t[x].cnt;
        return;
    }
    pushdown(x);
    if (t[x].xm <= a && t[x].ym <= b &&
↪   t[x].exist) {
        sum += t[x].val;
        cnt += 1;
    }
    query(t[x].lc, a, b, sum, cnt);
    query(t[x].rc, a, b, sum, cnt);
}
```

8 图论

8.1 点双

```
void dfs1(int u, int p = 0) {
    static int tme = 0, stk[N], tp;
    dfn[u] = low[u] = ++tme;
    stk[++tp] = u;
    int child = 0;
    for (int v: g[u]) {
        if (!dfn[v]) {
            dfs1(v, u); ++child;
            low[u] = min(low[u], low[v]);
            if (low[v] >= dfn[u]) {
                cut[u] = true;
                ++cc;
                do bcc[cc].emplace_back(stk[tp]);
                while (stk[tp--] != v);
                bcc[cc].emplace_back(u);
            }
        } else
            low[u] = min(low[u], dfn[v]);
    }
    if (!child) {
        cut[u] = true;
        bcc[++cc].emplace_back(u);
    }
}
```

8.2 全局平衡二叉树

```
vector<int> g[];
int siz[], son[], lsiz[];
int pa[], ch[][2];
T val[], sum[];
void dfs1(int u, int p = 0) {
    siz[u] = 1;
    for (int v : g[u]) {
        if (v == p) continue;
        dfs1(v, u);
        siz[u] += siz[v];
        if (siz[v] > siz[son[u]]) son[u] = v;
    }
}
void dfs2(int u, int p = 0) {
    for (int v : g[u]) {
        if (v == p) continue;
        dfs2(v, u);
        if (v == son[u]) continue;
        lsiz[u] += siz[v];
        // val[v] -> val[u]
    }
    sum[u] = val[u];
}
int build(vector<int> &vc, int l, int r) {
    if (l > r) return 0;
    int tot = 0;
    for (int i = l; i <= r; ++i) tot +=
↪   lsiz[vc[i]];
    for (int i = l, sum = 0; i <= r; ++i)
        if ((sum += lsiz[vc[i]]) * 2 >= tot) {
            int x = vc[i];
            if ((ch[x][0] = build(vc, l, i - 1)))
↪   pa[ch[x][0]] = x;
            if ((ch[x][1] = build(vc, i + 1, r)))
↪   pa[ch[x][1]] = x;
            return x;
        }
}

int build(int u) {
    static bool vis[N];
    vector<int> stk;
    for (int v = u; v; v = son[v]) {
        vis[v] = true;
        stk.emplace_back(v);
    }
```

```
    int x = build(stk, 0, (int) stk.size() - 1);
    for (int v = u; v; v = son[v])
        for (int w : g[v])
            if (!vis[w]) pa[build(w)] = v;
    return x;
}
int rt;
int build() { rt = build(1); }
void pushup(x) {
    sum[x] = val[x];
    if (ch[x][0]) sum[x] = sum[ch[x][0]] + sum[x];
    if (ch[x][1]) sum[x] = sum[x] + sum[ch[x][1]];
}
void modify(int x) {
    int y;
    while ((x = pa[y = x])) {
        if (ch[x][0] != y && ch[x][1] != y)
            // del sum[y] -> val[x]
        pushup(y);
        if (ch[x][0] != y && ch[x][1] != y)
            // add sum[y] -> val[x]
    }
    pushup(y);
}
```

8.3 求欧拉回路

```
// input: N, k, graph
// output: print_ans (an euler tour whose length
↪  is \geq k)

int k;
bool vis[N];
vector<int> g[N];
vector<int> ans1, ans2;
void print_ans(const vector<int> &vc) {
    for (int x : vc) cout << x << ' ';
    exit(0);
}
void dfs(int u) {
    vis[u] = true;
    if (ans1.size() >= k) print_ans(ans1);
    for (int v : g[u]) {
        if (vis[v]) continue;
        ans1.emplace_back(u);
        dfs(v);
        ans1.pop_back(); ans2.emplace_back(u);
        if (ans2.size() >= k) {
            reverse(begin(ans2), end(ans2));
            print_ans(ans2);
        }
    }
}
```

8.4 SPFA

```
// input: N, n - number of vertices
// output: dis - distance, return - no negative
↪  loops
int dis[N], cnt[N];
bool inque[N];
bool spfa(int n) {
    memset(dis, 0x3f, sizeof dis);
    queue<int> que;
    que.emplace(0);
    dis[0] = 0; inque[0] = true; cnt[0] = 1;
    while (!que.empty()) {
        int u = que.front(); que.pop();
        inque[u] = false;
        for (auto [v, w] : g[u]) {
            if (chkmin(dis[v], dis[u] + w) &&
↪  !inque[v]) {
                que.emplace(v);
                inque[v] = true;
                if (++cnt[v] > n) return false;
            }
        }
    }
    return true;
}
```

8.5 虚树

```
// 需要快速求 lca (LCA::get_lca)

void add_edge(int u, int v) {
    // 虚树中一条 u -> v 的边
}

void build(vector<int> &vc) {
    vc.emplace_back(1);
    sort(vc.begin(), vc.end(), [](int x, int y) {
        return dfn[x] < dfn[y];
    });
    vc.erase(unique(vc.begin(), vc.end()),
↪  vc.end());
    static int stk[N];
    int tp = 1;
    stk[tp] = 1;
    for (unsigned i = vc[0] == 1; i < vc.size();
↪  ++i) {
        int u = vc[i], lca = LCA::get_lca(u,
↪  stk[tp]);
        while (tp > 1 && dfn[stk[tp - 1]] >=
↪  dfn[lca]) {
            add_edge(stk[tp - 1], stk[tp]); --tp;
        }
        if (dfn[lca] < dfn[stk[tp]]) {
            add_edge(lca, stk[tp]); --tp;
        }
        if (!tp || dfn[stk[tp]] < dfn[lca]) {
            stk[++tp] = lca;
        }
        stk[++tp] = u;
    }
    for (; tp > 1; --tp) {
        add_edge(stk[tp - 1], stk[tp]);
    }
}
```

8.6 2-SAT

```
// Quasar
int stamp, comps, top;
int dfn[N], low[N], comp[N], stack[N];

void add(int x, int a, int y, int b) {
    edge[x << 1 | a].push_back(y << 1 | b);
}

void tarjan(int x) {
    dfn[x] = low[x] = ++stamp;
    stack[top++] = x;
    for (int i = 0; i < (int)edge[x].size(); ++i)
 ↪  {
        int y = edge[x][i];
        if (!dfn[y]) {
            tarjan(y);
            low[x] = std::min(low[x], low[y]);
        } else if (!comp[y]) {
            low[x] = std::min(low[x], dfn[y]);
        }
    }
    if (low[x] == dfn[x]) {
        comps++;
        do {
            int y = stack[--top];
            comp[y] = comps;
        } while (stack[top] != x);
    }
}

bool solve() {
    int counter = n + n + 1;
    stamp = top = comps = 0;
    std::fill(dfn, dfn + counter, 0);
    std::fill(comp, comp + counter, 0);
    for (int i = 0; i < counter; ++i) {
        if (!dfn[i]) {
            tarjan(i);
        }
    }
    for (int i = 0; i < n; ++i) {
        if (comp[i << 1] == comp[i << 1 | 1]) {
            return false;
        }
        answer[i] = (comp[i << 1 | 1] < comp[i <<
 ↪  1]); // DAG 序更大的 SCC 编号更小
    }
    return true;
}
```

8.7 有根树同构

```
const unsigned long long MAGIC = 4423;

unsigned long long magic[N];
std::pair<unsigned long long, int> hash[N];

void solve(int root) {
    magic[0] = 1;
    for (int i = 1; i <= n; ++i) {
        magic[i] = magic[i - 1] * MAGIC;
    }
    std::vector<int> queue;
    queue.push_back(root);
    for (int head = 0; head < (int)queue.size();
 ↪  ++head) {
        int x = queue[head];
        for (int i = 0; i < (int)son[x].size();
 ↪  ++i) {
            int y = son[x][i];
            queue.push_back(y);
        }
    }
    for (int index = n - 1; index >= 0; --index) {
        int x = queue[index];
        hash[x] = std::make_pair(0, 0);

        std::vector<std::pair<unsigned long long,
 ↪  int> > value;
        for (int i = 0; i < (int)son[x].size();
 ↪  ++i) {
            int y = son[x][i];
            value.push_back(hash[y]);
        }
        std::sort(value.begin(), value.end());

        hash[x].first = hash[x].first * magic[1] +
 ↪  37;
        hash[x].second++;
        for (int i = 0; i < (int)value.size();
 ↪  ++i) {
            hash[x].first = hash[x].first *
 ↪  magic[value[i].second] + value[i].first;
            hash[x].second += value[i].second;
        }
        hash[x].first = hash[x].first * magic[1] +
 ↪  41;
        hash[x].second++;
    }
}
```

8.8 支配树

```
// Dreadnought
// 半支配点：u 是 v 的半支配点，当且仅当存在一条 u
 ↪  -> v 的路径，使得这条路径上除了 u 以外 num >
 ↪  num[v]
// sem: num 最小的半支配点
vector<int> prec[N], succ[N];
vector<int> ord;
int stamp, vis[N];
int num[N];
int fa[N];
void dfs(int u) {
  vis[u] = stamp;
  num[u] = ord.size();
  ord.push_back(u);
  for (int i = 0; i < (int)succ[u].size(); ++i) {
    int v = succ[u][i];
    if (vis[v] != stamp) {
```

```
      fa[v] = u;
      dfs(v);
    }
  }
}
int fs[N], mins[N], dom[N], sem[N];
int find(int u) {
  if (u != fs[u]) {
    int v = fs[u];
    fs[u] = find(fs[u]);
    if (mins[v] != -1 && num[sem[mins[v]]] <
↪   num[sem[mins[u]]]) {
      mins[u] = mins[v];
    }
  }
  return fs[u];
}
void merge(int u, int v) { fs[u] = v; }
vector<int> buf[N];
int buf2[N];
void mark(int source) {
  ord.clear();
  ++stamp;
  dfs(source);
  for (int i = 0; i < (int)ord.size(); ++i) {
    int u = ord[i];
    fs[u] = u, mins[u] = -1, buf2[u] = -1;
  }
  for (int i = (int)ord.size() - 1; i > 0; --i) {
    int u = ord[i], p = fa[u];
    sem[u] = p;
    for (int j = 0; j < (int)prec[u].size(); ++j)
↪   {
      int v = prec[u][j];
      if (use[v] != stamp) continue;
      if (num[v] > num[u]) {
        find(v); v = sem[mins[v]];
      }
      if (num[v] < num[sem[u]]) {
        sem[u] = v;
      }
    }
    buf[sem[u]].push_back(u);
    mins[u] = u;
    merge(u, p);
    while (buf[p].size()) {
      int v = buf[p].back();
      buf[p].pop_back();
      find(v);
      if (sem[v] == sem[mins[v]]) {
        dom[v] = sem[v];
      } else {
        buf2[v] = mins[v];
      }
    }
  }
  dom[ord[0]] = ord[0];
  for (int i = 0; i < (int)ord.size(); ++i) {
    int u = ord[i];
    if (~buf2[u]) {
      dom[u] = dom[buf2[u]];
    }
  }
}
```

8.9 MCS 求 PEO

```
// 一个图是弦图当且仅当它有 PEO
// input: N
// n: number of vertices
// g: edges

const size_t N = 1E4 + 5;

int n; vector<int> g[N];

int label[N], pos[N], peo[N];
vector<int> que[N];

int main() {
    for (int i = 1; i <= n; ++i) {
        que[0].emplace_back(i);
    }
    int j = 0;
    for (int i = n; i >= 1; --i) {
        int u;
        while (j >= 0) {
            while (!que[j].empty()) {
                u = que[j].back();
                if (pos[u]) {
                    que[j].pop_back();
                } else {
                    break;
                }
            }
            if (!que[j].empty()) break;
            --j;
        }
        assert(j >= 0);
        pos[u] = i; peo[i] = u;
        for (int v : g[u]) {
            if (!pos[v]) {
                ++label[v];
                que[label[v]].emplace_back(v);
                if (label[v] > j) j = label[v];
            }
        }
    }
}
```

8.10 最大团

```
// Dreadnought
// Super Fast Maximum Clique
// To Build Graph: Maxclique(Edges, Number of
↪  Nodes)
// To Get Answer: mcqdyn(AnswerNodes Index Array,
↪  AnswserLength)
typedef bool BB[N];
struct Maxclique {
  const BB* e; int pk, level; const float Tlimit;
```

```
  struct Vertex{ int i, d; Vertex(int
→   i):i(i),d(0){} };
  typedef vector<Vertex> Vertices; typedef
→   vector<int> ColorClass;
  Vertices V; vector<ColorClass> C; ColorClass
→   QMAX, Q;
  static bool desc_degree(const Vertex &vi, const
→   Vertex &vj){
    return vi.d > vj.d;
  }
  void init_colors(Vertices &v){
    const int max_degree = v[0].d;
    for(int i = 0; i < (int)v.size(); i++) v[i].d
→   = min(i, max_degree) + 1;
  }
  void set_degrees(Vertices &v){
    for(int i = 0, j; i < (int)v.size(); i++)
      for(v[i].d = j = 0; j < int(v.size()); j++)
        v[i].d += e[v[i].i][v[j].i];
  }
  struct StepCount{ int i1, i2;
→   StepCount():i1(0),i2(0){} };
  vector<StepCount> S;
  bool cut1(const int pi, const ColorClass &A){
    for(int i = 0; i < (int)A.size(); i++) if
→   (e[pi][A[i]]) return true;
    return false;
  }
  void cut2(const Vertices &A, Vertices &B){
    for(int i = 0; i < (int)A.size() - 1; i++)
      if(e[A.back().i][A[i].i])
        B.push_back(A[i].i);
  }
  void color_sort(Vertices &R){
    int j = 0, maxno = 1, min_k =
→   max((int)QMAX.size() - (int)Q.size() + 1, 1);
    C[1].clear(), C[2].clear();
    for(int i = 0; i < (int)R.size(); i++) {
      int pi = R[i].i, k = 1;
      while(cut1(pi, C[k])) k++;
      if(k > maxno) maxno = k, C[maxno +
→   1].clear();
      C[k].push_back(pi);
      if(k < min_k) R[j++].i = pi;
    }
    if(j > 0) R[j - 1].d = 0;
    for(int k = min_k; k <= maxno; k++)
      for(int i = 0; i < (int)C[k].size(); i++)
        R[j].i = C[k][i], R[j++].d = k;
  }
  void expand_dyn(Vertices &R){// diff -> diff
→   with no dyn
    S[level].i1 = S[level].i1 + S[level - 1].i1 -
→   S[level].i2;//diff
    S[level].i2 = S[level - 1].i1;//diff
    while((int)R.size()) {
      if((int)Q.size() + R.back().d >
→   (int)QMAX.size()){
        Q.push_back(R.back().i); Vertices Rp;
→   cut2(R, Rp);
        if((int)Rp.size()){
          if((float)S[level].i1 / ++pk < Tlimit)
→   degree_sort(Rp);//diff
          color_sort(Rp);
          S[level].i1++, level++;//diff
          expand_dyn(Rp);
          level--;//diff
        }
        else if((int)Q.size() > (int)QMAX.size())
→   QMAX = Q;
        Q.pop_back();
      }
      else return;
      R.pop_back();
    }
  }
  void mcqdyn(int* maxclique, int &sz){
    set_degrees(V); sort(V.begin(),V.end(),
→   desc_degree); init_colors(V);
    for(int i = 0; i < (int)V.size() + 1; i++)
→   S[i].i1 = S[i].i2 = 0;
    expand_dyn(V); sz = (int)QMAX.size();
    for(int i = 0; i < (int)QMAX.size(); i++)
→   maxclique[i] = QMAX[i];
  }
  void degree_sort(Vertices &R){
    set_degrees(R); sort(R.begin(), R.end(),
→   desc_degree);
  }
  Maxclique(const BB* conn, const int sz, const
→   float tt = 0.025) \

→   : pk(0), level(1), Tlimit(tt){

→   for(int i = 0; i < sz; i++)
→   V.push_back(Vertex(i));

→   e = conn, C.resize(sz + 1), S.resize(sz + 1);

→   }
};
```

8.11 最小树形图

```
// oi-wiki
// tarjan dmst - O(n + m \log m)
#define maxn 102
#define INF 0x3f3f3f3f
struct UnionFind {
  int fa[maxn << 1];
  UnionFind() { memset(fa, 0, sizeof(fa)); }
  void clear(int n) { memset(fa + 1, 0,
→   sizeof(int) * n); }
  int find(int x) { return fa[x] ? fa[x] =
→   find(fa[x]) : x; }
  int operator[](int x) { return find(x); }
};
struct Edge {
  int u, v, w, w0;
};
struct Heap {
  Edge *e;
```

```cpp
  int rk, constant;
  Heap *lch, *rch;
  Heap(Edge *_e) : e(_e), rk(1), constant(0), lch(nullptr), rch(nullptr) {}
  void push() {
    if (lch) lch->constant += constant;
    if (rch) rch->constant += constant;
    e->w += constant;
    constant = 0;
  }
};
Heap *merge(Heap *x, Heap *y) {
  if (!x) return y;
  if (!y) return x;
  if (x->e->w + x->constant > y->e->w + y->constant) swap(x, y);
  x->push();
  x->rch = merge(x->rch, y);
  if (!x->lch || x->lch->rk < x->rch->rk) swap(x->lch, x->rch);
  if (x->rch)
    x->rk = x->rch->rk + 1;
  else
    x->rk = 1;
  return x;
}
Edge *extract(Heap *&x) {
  Edge *r = x->e;
  x->push();
  x = merge(x->lch, x->rch);
  return r;
}

vector<Edge> in[maxn];
int n, m, fa[maxn << 1], nxt[maxn << 1];
Edge *ed[maxn << 1];
Heap *Q[maxn << 1];
UnionFind id;

void contract() {
  bool mark[maxn << 1];
  // 将图上的每一个结点与其相连的那些结点进行记录。
  for (int i = 1; i <= n; i++) {
    queue<Heap *> q;
    for (int j = 0; j < in[i].size(); j++) q.push(new Heap(&in[i][j]));
    while (q.size() > 1) {
      Heap *u = q.front();
      q.pop();
      Heap *v = q.front();
      q.pop();
      q.push(merge(u, v));
    }
    Q[i] = q.front();
  }
  mark[1] = true;
  for (int a = 1, b = 1, p; Q[a]; b = a, mark[b] = true) {
    //寻找最小入边以及其端点，保证无环。
    do {
      ed[a] = extract(Q[a]);
      a = id[ed[a]->u];
    } while (a == b && Q[a]);
    if (a == b) break;
    if (!mark[a]) continue;
    // 对发现的环进行收缩，以及环内的结点重新编号，总权值更新。
    for (a = b, n++; a != n; a = p) {
      id.fa[a] = fa[a] = n;
      if (Q[a]) Q[a]->constant -= ed[a]->w;
      Q[n] = merge(Q[n], Q[a]);
      p = id[ed[a]->u];
      nxt[p == n ? b : p] = a;
    }
  }
}

i64 expand(int x, int r);
i64 expand_iter(int x) {
  i64 r = 0;
  for (int u = nxt[x]; u != x; u = nxt[u]) {
    if (ed[u]->w0 >= INF)
      return INF;
    else
      r += expand(ed[u]->v, u) + ed[u]->w0;
  }
  return r;
}
i64 expand(int x, int t) {
  i64 r = 0;
  for (; x != t; x = fa[x]) {
    r += expand_iter(x);
    if (r >= INF) return INF;
  }
  return r;
}
void link(int u, int v, int w) { in[v].push_back({u, v, w, w}); }

int main() {
  int rt;
  scanf("%d %d %d", &n, &m, &rt);
  for (int i = 0; i < m; i++) {
    int u, v, w;
    scanf("%d %d %d", &u, &v, &w);
    link(u, v, w);
  }
  // 保证强连通
  for (int i = 1; i <= n; i++) link(i > 1 ? i - 1 : n, i, INF);
  contract();
  i64 ans = expand(rt, n);
  if (ans >= INF)
    puts("-1");
  else
    printf("%lld\n", ans);
  return 0;
}
```

8.12 二分图最大权匹配（KM）

```
int n;
// n, N 两侧点数
// 需定义 INF
namespace KM {
i64 arr[N][N];
bool visl[N], visr[N];
int matchl[N], matchr[N], matcht[N];
i64 slack[N], expl[N], expr[N];
void change_match(int v) {
    for (; v; swap(v, matchl[matcht[v]])) {
        matchr[v] = matcht[v];
    }
}
void find_path(int s) {
    queue<int> que;
    que.emplace(s);
    visl[s] = true;
    while (true) {
        while (!que.empty()) {
            int l = que.front();
            que.pop();
            for (int r = 1; r <= n; ++r) {
                if (visr[r]) continue;
                i64 gap = expl[l] + expr[r] -
↪   arr[l][r];
                if (gap > slack[r]) continue;
                matcht[r] = l;
                if (gap == 0) {
                    if (!matchr[r]) return
↪   change_match(r);
                    que.emplace(matchr[r]);
                    visl[matchr[r]] = visr[r] =
↪   true;
                } else {
                    slack[r] = gap;
                }
            }
        }
        int v = -1;
        for (int r = 1; r <= n; ++r) {
            if (!visr[r] && (!~v || slack[r] <
↪   slack[v])) {
                v = r;
            }
        }
        assert(~v);
        i64 delta = slack[v];
        for (int i = 1; i <= n; ++i) {
            if (visl[i]) expl[i] -= delta;
            if (visr[i]) expr[i] += delta; else
↪   slack[i] -= delta;
        }
        if (!matchr[v]) return change_match(v);
        que.emplace(matchr[v]);
        visl[matchr[v]] = visr[v] = true;
    }
}
i64 km() {
    for (int l = 1; l <= n; ++l) {
        for (int r = 1; r <= n; ++r) {
            expl[l] = max(expl[l], arr[l][r]);
        }
    }
    for (int l = 1; l <= n; ++l) {
        fill(slack + 1, slack + n + 1, INF);
        memset(visl, 0, sizeof(bool) * (n + 1));
        memset(visr, 0, sizeof(bool) * (n + 1));
        memset(matcht, 0, sizeof(int) * (n + 1));
        find_path(l);
    }
    i64 ans = 0;
    for (int i = 1; i <= n; ++i) ans +=
↪   arr[i][matchl[i]];
    return ans;
}
}
```

8.13 一般图最大匹配（随机，非完全正确）

```
// #79. skyline 97
using namespace std;
int n,m,a[505],p[505],ans,u[505],U;
vector<int> v[505];
bool dfs(int x){
    u[x]=U;
    random_shuffle(v[x].begin(),v[x].end());
    for(int i=0;i<v[x].size();++i){
        int y=v[x][i];
        if(!a[y]){
            a[x]=y,a[y]=x;
            return 1;
        }
    }
    for(int i=0;i<v[x].size();++i){
        int y=v[x][i],z=a[y];
        if(u[z]==U) continue;
        a[x]=y,a[y]=x,a[z]=0;
        if(dfs(z)) return 1;
        a[x]=0,a[y]=z,a[z]=y;
    }
    return 0;
}
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;++i){
        int x,y;
        scanf("%d%d",&x,&y);
        v[x].pb(y),v[y].pb(x);
    }
    for(int i=1;i<=n;++i)p[i]=i;
    for(int _=5;_;--_){
        random_shuffle(p+1,p+n+1);
        for(int i=1;i<=n;++i){
            if(a[p[i]]) continue;
            ++U;
            dfs(p[i]);
        }
    }
    for(int i=1;i<=n;++i)if(a[i]>i)++ans;
    printf("%d\n",ans);
```

```
    for(int i=1;i<n;++i)printf("%d ",a[i]);
    printf("%d\n",a[n]);
    //system("pause");
    return 0;
}
```

8.14 一般图最大权匹配

```
// uoj #81 claris
#include<bits/stdc++.h>
#define DIST(e) (lab[e.u]+lab[e.v]-g[e.u][e.v].w*2)
using namespace std;
typedef long long ll;
const int N=1023,INF=1e9;
struct Edge{
  int u,v,w;
} g[N][N];
int n,m,n_x,lab[N],match[N],slack[N],st[N],pa[N],flower_from[N][N],S[N],vis[N];
vector<int> flower[N];
deque<int> q;
void update_slack(int u,int x){
  if(!slack[x]||DIST(g[u][x])<DIST(g[slack[x]][x]))slack[x]=u;
}
void set_slack(int x){
  slack[x]=0;
  for(int u=1; u<=n; ++u)
    if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)update_slack(u,x);
}
void q_push(int x){
  if(x<=n)return q.push_back(x);
  for(int i=0; i<flower[x].size(); i++)q_push(flower[x][i]);
}
void set_st(int x,int b){
  st[x]=b;
  if(x<=n)return;
  for(int i=0; i<flower[x].size(); ++i)set_st(flower[x][i],b);
}
int get_pr(int b,int xr){
  int pr=find(flower[b].begin(),flower[b].end(),xr)-flower[b].begin();
  if(pr%2==1){
    reverse(flower[b].begin()+1,flower[b].end());
    return (int)flower[b].size()-pr;
  }
  else return pr;
}
void set_match(int u,int v){
  match[u]=g[u][v].v;
  if(u<=n)return;
  Edge e=g[u][v];
  int xr=flower_from[u][e.u],pr=get_pr(u,xr);
  for(int i=0; i<pr; ++i)set_match(flower[u][i],flower[u][i^1]);
  set_match(xr,v);
  rotate(flower[u].begin(),flower[u].begin()+pr,flower[u].end());
}
void augment(int u,int v){
  int xnv=st[match[u]];
  set_match(u,v);
  if(!xnv)return;
  set_match(xnv,st[pa[xnv]]);
  augment(st[pa[xnv]],xnv);
}
int get_lca(int u,int v){
  static int t=0;
  for(++t; u||v; swap(u,v)){
    if(u==0)continue;
    if(vis[u]==t)return u;
    vis[u]=t;
    u=st[match[u]];
    if(u)u=st[pa[u]];
  }
  return 0;
}
void add_blossom(int u,int lca,int v){
  int b=n+1;
  while(b<=n_x&&st[b])++b;
  if(b>n_x)++n_x;
  lab[b]=0,S[b]=0;
  match[b]=match[lca];
  flower[b].clear();
  flower[b].push_back(lca);
  for(int x=u,y; x!=lca; x=st[pa[y]])
    flower[b].push_back(x),flower[b].push_back(y=st[match[x]]),q_push(y);
  reverse(flower[b].begin()+1,flower[b].end());
  for(int x=v,y; x!=lca; x=st[pa[y]])
    flower[b].push_back(x),flower[b].push_back(y=st[match[x]]),q_push(y);
  set_st(b,b);
  for(int x=1; x<=n_x; ++x)g[b][x].w=g[x][b].w=0;
  for(int x=1; x<=n; ++x)flower_from[b][x]=0;
  for(int i=0; i<flower[b].size(); ++i){
    int xs=flower[b][i];
    for(int x=1; x<=n_x; ++x)
      if(g[b][x].w==0||DIST(g[xs][x])<DIST(g[b][x]))
        g[b][x]=g[xs][x],g[x][b]=g[x][xs];
    for(int x=1; x<=n; ++x)
      if(flower_from[xs][x])flower_from[b][x]=xs;
  }
  set_slack(b);
}
void expand_blossom(int b){
  for(int i=0; i<flower[b].size(); ++i)
    set_st(flower[b][i],flower[b][i]);
  int xr=flower_from[b][g[b][pa[b]].u],pr=get_pr(b,xr);
  for(int i=0; i<pr; i+=2){
    int xs=flower[b][i],xns=flower[b][i+1];
    pa[xs]=g[xns][xs].u;
    S[xs]=1,S[xns]=0;
    slack[xs]=0,set_slack(xns);
    q_push(xns);
```

```cpp
  }
  S[xr]=1,pa[xr]=pa[b];
  for(int i=pr+1; i<flower[b].size(); ++i){
    int xs=flower[b][i];
    S[xs]=-1,set_slack(xs);
  }
  st[b]=0;
}
bool on_found_Edge(const Edge &e){
  int u=st[e.u],v=st[e.v];
  if(S[v]==-1){
    pa[v]=e.u,S[v]=1;
    int nu=st[match[v]];
    slack[v]=slack[nu]=0;
    S[nu]=0,q_push(nu);
  }
  else if(S[v]==0){
    int lca=get_lca(u,v);
    if(!lca)return augment(u,v),augment(v,u),1;
    else add_blossom(u,lca,v);
  }
  return 0;
}
bool matching(){
  fill(S,S+n_x+1,-1),fill(slack,slack+n_x+1,0);
  q.clear();
  for(int x=1; x<=n_x; ++x)
    if(st[x]==x&&!match[x])pa[x]=0,S[x]=0,q_push(x);
  if(q.empty())return 0;
  for(;;){
    while(q.size()){
      int u=q.front();
      q.pop_front();
      if(S[st[u]]==1)continue;
      for(int v=1; v<=n; ++v)
        if(g[u][v].w>0&&st[u]!=st[v]){
          if(DIST(g[u][v])==0){
            if(on_found_Edge(g[u][v]))return 1;
          }
          else update_slack(u,st[v]);
        }
    }
    int d=INF;
    for(int b=n+1; b<=n_x; ++b)
      if(st[b]==b&&S[b]==1)d=min(d,lab[b]/2);
    for(int x=1; x<=n_x; ++x)
      if(st[x]==x&&slack[x]){
        if(S[x]==-1)d=min(d,DIST(g[slack[x]][x]));
        else if(S[x]==0)d=min(d,DIST(g[slack[x]][x])/2);
      }
    for(int u=1; u<=n; ++u){
      if(S[st[u]]==0){
        if(lab[u]<=d)return 0;
        lab[u]-=d;
      }
      else if(S[st[u]]==1)lab[u]+=d;
    }
    for(int b=n+1; b<=n_x; ++b)
      if(st[b]==b){
        if(S[st[b]]==0)lab[b]+=d*2;
        else if(S[st[b]]==1)lab[b]-=d*2;
      }
    q.clear();
    for(int x=1; x<=n_x; ++x)
      if(st[x]==x&&slack[x]&&st[slack[x]]!=x&&DIST(g[slack[x]][x])==0)
        if(on_found_Edge(g[slack[x]][x]))return 1;
    for(int b=n+1; b<=n_x; ++b)
      if(st[b]==b&&S[b]==1&&lab[b]==0)expand_blossom(b);
  }
  return 0;
}
pair<ll,int> weight_blossom(){
  fill(match,match+n+1,0);
  n_x=n;
  int n_matches=0;
  ll tot_weight=0;
  for(int u=0; u<=n; ++u)st[u]=u,flower[u].clear();
  int w_max=0;
  for(int u=1; u<=n; ++u)
    for(int v=1; v<=n; ++v){
      flower_from[u][v]=(u==v?u:0);
      w_max=max(w_max,g[u][v].w);
    }
  for(int u=1; u<=n; ++u)lab[u]=w_max;
  while(matching())++n_matches;
  for(int u=1; u<=n; ++u)
    if(match[u]&&match[u]<u)
      tot_weight+=g[u][match[u]].w;
  return make_pair(tot_weight,n_matches);
}
int main(){
  cin>>n>>m;
  for(int u=1; u<=n; ++u)
    for(int v=1; v<=n; ++v)
      g[u][v]=Edge {u,v,0};
  for(int i=0,u,v,w; i<m; ++i){
    cin>>u>>v>>w;
    g[u][v].w=g[v][u].w=w;
  }
  cout<<weight_blossom().first<<'\n';
  for(int u=1; u<=n; ++u)cout<<match[u]<<' ';
}
```

8.15 无向图最小割

```cpp
// Quasar
int cost[maxn][maxn],seq[maxn],len[maxn],n,m,pop,ans;
bool used[maxn];
void Init(){
  int i,j,a,b,c;
  for(i=0;i<n;i++) for(j=0;j<n;j++) cost[i][j]=0;
  for(i=0;i<m;i++){
    scanf("%d %d %d",&a,&b,&c); cost[a][b]+=c; cost[b][a]+=c;
  }
  pop=n; for(i=0;i<n;i++) seq[i]=i;
```

```
}
void Work(){
  ans=inf; int i,j,k,l,mm,sum,pk;
  while(pop > 1){
    for(i=1;i<pop;i++) used[seq[i]]=0; used[seq[0]]=1;
    for(i=1;i<pop;i++) len[seq[i]]=cost[seq[0]][seq[i]];
    pk=0; mm=-inf; k=-1;
    for(i=1;i<pop;i++) if(len[seq[i]] > mm){ mm=len[seq[i]]; k=i; }
    for(i=1;i<pop;i++){
      used[seq[l=k]]=1;
      if(i==pop-2) pk=k;
      if(i==pop-1) break;
      mm=-inf;
      for(j=1;j<pop;j++) if(!used[seq[j]])
        if((len[seq[j]]+=cost[seq[l]][seq[j]]) > mm)
          mm=len[seq[j]], k=j;
    }
    sum=0;
    for(i=0;i<pop;i++) if(i != k) sum+=cost[seq[k]][seq[i]];
    ans=min(ans,sum);
    for(i=0;i<pop;i++)
      cost[seq[k]][seq[i]]=cost[seq[i]][seq[k]]+=cost[seq[pk]][seq[i]];
    seq[pk]=seq[--pop];
  }
  printf("%d\n",ans);
}
```

8.16 网络流（ISAP）

```
// N: vertices, M: edges
// method: add_edge(int u, int v, int cap), maxflow(int s, int t)
struct Maxflow {
    struct Edge {
        int to, cap, nxt;
        Edge(int to = 0, int cap = 0, int nxt = 0):
            to(to), cap(cap), nxt(nxt) {}
    } e[M];
    int head[N], cur[N], d[N], f[N], tot = 1;
    int n, s, t;
    void add_edge(int u, int v, int cap) {
        e[++tot] = Edge(v, cap, head[u]); head[u] = tot;
        e[++tot] = Edge(u,  0,  head[v]); head[v] = tot;
    }
    int dfs(int v, int fl = INF) {
        if (v == t) return fl;
        int ret = 0;
        for (int &i = cur[v]; i; i = e[i].nxt) {
            if (e[i].cap && d[e[i].to] + 1 == d[v]) {
                int tmp = dfs(e[i].to, min(fl, e[i].cap));
                ret += tmp; fl -= tmp;
                e[i].cap -= tmp;
                e[i ^ 1].cap += tmp;
                if (!fl) return ret;
            }
        }
        cur[v] = head[v];
        if (!(--f[d[v]])) d[s] = n;
        ++f[++d[v]];
        return ret;
    }
    int maxflow(int _s, int _t) {
        n = _n; s = _s; t = _t;
        memset(cur, 0, sizeof cur);
        memset(d, 0, sizeof d);
        memset(f, 0, sizeof f);
        f[0] = n;
        int ret = 0;
        while (d[s] < n) ret += dfs(s);
        return ret;
    }
} flow;
```

8.17 网络流（HLPP）

```
// N: vertices, M: edges
// method: add_edge(int u, int v, i64 cap), maxflow(int s, int t)
struct Maxflow {
    int n;
    struct Edge {
        int to; i64 cap; int nxt;
        Edge() {}
        Edge(int to, i64 cap, int nxt) : to(to), cap(cap), nxt(nxt) {}
    } e[M << 1];
    int tot_e, head[N], cur[N], deg[N];
    Maxflow() {
        memset(this, 0, sizeof *this);
        tot_e = 1;
    }
    void add_edge(int u, int v, i64 cap) {
        e[++tot_e] = {v, cap, head[u]}; head[u] = tot_e;
        e[++tot_e] = {u,  0 , head[v]}; head[v] = tot_e;
        ++deg[u]; ++deg[v];
    }
    int cnt_upd_h, max_h, h[N], cnt[N]; i64 rest[N];
    vector<int> vc1[N], vc2[N];
    void update_h(int v, int nh) {
        ++cnt_upd_h;
        if (h[v] < INF) --cnt[h[v]];
        h[v] = nh;
        if (h[v] == INF) return;
        ++cnt[h[v]];
        max_h = h[v];
        vc1[h[v]].emplace_back(v);
        if (rest[v]) vc2[h[v]].emplace_back(v);
    }
```

```cpp
    void relabel(int t) {
        cnt_upd_h = max_h = 0;
        fill(h, h + n + 1, INF);
        fill(cnt, cnt + n + 1, 0);
        for (int i = 0; i <= max_h; ++i) {
            vc1[i].clear();
            vc2[i].clear();
        }
        queue<int> que;
        que.emplace(t);
        update_h(t, 0);
        while (!que.empty()) {
            int u = que.front(); que.pop();
            for (int i = head[u]; i; i = e[i].nxt) {
                int v = e[i].to;
                if (h[u] + 1 < h[v] && e[i ^ 1].cap) {
                    update_h(v, h[u] + 1);
                    que.emplace(v);
                }
            }
        }
    }
    void push(int i) {
        int u = e[i ^ 1].to, v = e[i].to;
        i64 w = min((i64) rest[u], e[i].cap);
        if (!w) return;
        if (!rest[v]) vc2[h[v]].emplace_back(v);
        e[i].cap -= w; e[i ^ 1].cap += w;
        rest[u] -= w; rest[v] += w;
    }
    void push_flow(int u) {
        int nh = INF;
        for (int &i = cur[u], j = 0; j < deg[u]; i = e[i].nxt, ++j) {
            if (!i) i = head[u];
            int v = e[i].to;
            if (e[i].cap) {
                if (h[u] == h[v] + 1) {
                    push(i);
                    if (!rest[u]) return;
                } else if (nh > h[v] + 1) {
                    nh = h[v] + 1;
                }
            }
        }
        if (cnt[h[u]] > 1) {
            update_h(u, nh);
        } else {
            for (int i = h[u]; i <= max_h; ++i) {
                for (int v : vc1[i]) update_h(v, INF);
                vc1[i].clear();
            }
        }
    }
    int maxflow(int s, int t, int lim = 10000) {
        rest[s] = 1E18;
        relabel(t);
        for (int i = head[s]; i; i = e[i].nxt) push(i);
        for (int &i = max_h; ~i; --i) {
            while (!vc2[i].empty()) {
                int u = vc2[i].back();
                vc2[i].pop_back();
                if (h[u] != i) continue;
                push_flow(u);
                if (cnt_upd_h > lim) relabel(t);
            }
        }
        return rest[t];
    }
} flow;
```

8.18 网络流（dengyaotriangle）

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

namespace flow { template<typename edg> class base_flowgraph {
public:
    int n, s, t;
    vector<vector<edg>> adj;
    base_flowgraph(int n, int s, int t): n(n), s(s), t(t) {
        adj.resize(n, vector<edg>());
    }
    vector<edg> &operator[](unsigned x) {
        return adj[x];
    }
};

template<typename FT> struct flow_edg {
    int v;
    FT w;
    int b;
    flow_edg(int v, FT w, int b): v(v), w(w), b(b) {}
};

template<typename FT> class dinic_flowgraph: public base_flowgraph<flow_edg<FT>> {
public:
    vector<int> cur, dis;
    dinic_flowgraph(int n, int s, int t): base_flowgraph<flow_edg<FT>>(n, s, t) {
        cur.resize(n);
        dis.resize(n);
    }
    void addedge(int u, int v, FT w) {
        flow_edg<FT> eu(v, w, (int)this->adj[v].size()), ev(u, 0, (int)this->adj[u].size());
        this->adj[u].push_back(eu);
        this->adj[v].push_back(ev);
    }
    bool bfs() {
        fill(dis.begin(), dis.end(), -1);
```

```
        fill(cur.begin(), cur.end(), 0);
        dis[this->s] = 0;
        queue<int> q;
        q.push(this->s);

        while (!q.empty()) {
            int u = q.front();
            q.pop();

            for (int i = 0; i <
↪   (int)this->adj[u].size(); i++) {
                flow_edg<FT> v = this->adj[u][i];

                if (v.w && dis[v.v] == -1) {
                    dis[v.v] = dis[u] + 1;
                    q.push(v.v);
                }
            }
        }

        return dis[this->t] != -1;
    }
    FT dfs(int u, FT fl) {
        if (!fl || u == this->t)
            return fl;

        FT ret(0);

        for (int &i = cur[u]; i <
↪   (int)this->adj[u].size(); i++) {
            flow_edg<FT> v = this->adj[u][i];

            if (v.w && dis[v.v] == dis[u] + 1) {
                FT nw = dfs(v.v, min(fl, v.w));
                this->adj[u][i].w -= nw;
                this->adj[v.v][v.b].w += nw;
                ret += nw;

                if (!(fl -= nw))
                    return ret;
            }
        }

        return ret;
    }
    FT maxflow() {
        FT ans(0);

        while (bfs())
            ans += dfs(this->s,
↪   numeric_limits<FT>::max());

        return ans;
    }
};
template<typename FT, typename CT> struct cost_edg
↪   {
    int v;
    FT w;
    CT c;
    int b;
    cost_edg(int v, FT w, CT c, int b): v(v),
↪   w(w), c(c), b(b) {}
};

template<typename FT, typename CT> class
↪   ek_flowgraph: public
↪   base_flowgraph<cost_edg<FT, CT>> {
public:
    vector<CT> dis;
    vector<int> lst, lsi;
    vector<bool> inq;
    ek_flowgraph(int n, int s, int t):
↪   base_flowgraph<cost_edg<FT, CT>>(n, s, t) {
        dis.resize(n);
        lst.resize(n);
        lsi.resize(n);
        inq.resize(n);
    }
    void addedge(int u, int v, FT w, CT c) {
        cost_edg<FT, CT> eu(v, w, c,
↪   (int)this->adj[v].size()), ev(u, 0, -c,
↪   (int)this->adj[u].size());
        this->adj[u].push_back(eu);
        this->adj[v].push_back(ev);
    }
    template<class comp>bool spfa(pair<FT, CT>
↪   &ans, CT mxv, comp cmp) {
        queue<int> q;
        fill(dis.begin(), dis.end(), mxv);
        fill(inq.begin(), inq.end(), 0);
        dis[this->s] = 0;
        q.push(this->s);

        while (!q.empty()) {
            int u = q.front();
            q.pop();
            inq[u] = 0;

            for (int i = 0; i <
↪   (int)this->adj[u].size(); i++) {
                cost_edg<FT, CT> v =
↪   this->adj[u][i];

                if (v.w && cmp(dis[v.v], dis[u] +
↪   v.c)) {
                    dis[v.v] = dis[u] + v.c;
                    lst[v.v] = u;
                    lsi[v.v] = i;

                    if (!inq[v.v]) {
                        q.push(v.v);
                        inq[v.v] = 1;
                    }
                }
            }
        }

        if (dis[this->t] == mxv)
            return 0;

        int cu = this->t;
```

```
        FT fl = numeric_limits<FT>::max();
        CT fc(0);

        while (cu != this->s) {
            fl = min(fl, this->adj[lst[cu]][lsi[cu]].w);
            fc += this->adj[lst[cu]][lsi[cu]].c;
            cu = lst[cu];
        }

        ans.first += fl, ans.second += fl * fc;
        cu = this->t;

        while (cu != this->s) {
            this->adj[lst[cu]][lsi[cu]].w -= fl;
            this->adj[cu][this->adj[lst[cu]][lsi[cu]].b].w += fl;
            cu = lst[cu];
        }

        return 1;
    }
    pair<FT, CT> mincostmaxflow() {
        pair<FT, CT> ans(0, 0);

        while (spfa(ans, numeric_limits<CT>::max(), greater<CT>()));

        return ans;
    }
    pair<FT, CT> maxcostmaxflow() {
        pair<FT, CT> ans(0, 0);

        while (spfa(ans, numeric_limits<CT>::min(), less<CT>()));

        return ans;
    }
};


template<typename FT, typename CT> class zkw_flowgraph: public base_flowgraph<cost_edg<FT, CT>> {
public:
    vector<int> cur;
    vector<CT> dis;
    vector<bool> vis, inq;
    zkw_flowgraph(int n, int s, int t): base_flowgraph<cost_edg<FT, CT>>(n, s, t) {
        dis.resize(n);
        cur.resize(n);
        vis.resize(n);
        inq.resize(n);
    }
    void addedge(int u, int v, FT w, CT c) {
        cost_edg<FT, CT> eu(v, w, c, (int)this->adj[v].size()), ev(u, 0, -c, (int)this->adj[u].size());
        this->adj[u].push_back(eu);
        this->adj[v].push_back(ev);
    }
    template<class comp> bool spfa(CT mxv, comp cmp) {
        queue<int> q;
        fill(dis.begin(), dis.end(), mxv);
        fill(inq.begin(), inq.end(), 0);
        fill(cur.begin(), cur.end(), 0);
        fill(vis.begin(), vis.end(), 0);
        dis[this->s] = 0;
        q.push(this->s);

        while (!q.empty()) {
            int u = q.front();
            q.pop();
            inq[u] = 0;

            for (int i = 0; i < (int)this->adj[u].size(); i++) {
                cost_edg<FT, CT> v = this->adj[u][i];

                if (v.w && cmp(dis[v.v], dis[u] + v.c)) {
                    dis[v.v] = dis[u] + v.c;

                    if (!inq[v.v]) {
                        q.push(v.v);
                        inq[v.v] = 1;
                    }
                }
            }
        }

        return dis[this->t] != mxv;
    }
    FT dfs(int u, FT fl, CT &cst) {
        if (!fl || u == this->t)
            return fl;

        vis[u] = 1;
        FT ret(0);

        for (int &i = cur[u]; i < (int)this->adj[u].size(); i++) {
            cost_edg<FT, CT> v = this->adj[u][i];

            if (v.w && dis[v.v] == dis[u] + v.c && !vis[v.v]) {
                FT nw = dfs(v.v, min(fl, v.w), cst);
                this->adj[u][i].w -= nw;
                this->adj[v.v][v.b].w += nw;
                cst += nw * v.c;
                ret += nw;

                if (!(fl -= nw)) {
                    vis[u] = 0;
                    return ret;
                }
            }
```

```
        }

        return ret;
    }
    pair<FT, CT> mincostmaxflow() {
        pair<FT, CT> ans(0, 0);

        while (spfa(numeric_limits<CT>::max(),
↪   greater<CT>()))
            ans.first += dfs(this->s,
↪   numeric_limits<FT>::max(), ans.second);

        return ans;
    }
    pair<FT, CT> maxcostmaxflow() {
        pair<FT, CT> ans(0, 0);
        fill(vis.begin(), vis.end(), 0);

        while (spfa(numeric_limits<CT>::min(),
↪   less<CT>()))
            ans.first += dfs(this->s,
↪   numeric_limits<FT>::max(), ans.second);

        return ans;
    }
};

template<typename FT, template<typename _FT> class
↪   based_graph> class bounded_flow : public
↪   based_graph<FT> {
public:
    vector<FT> pot;
    bounded_flow(int n): based_graph<FT>(n + 2, n,
↪   n + 1) {
        pot.resize(n + 2);
    }
    void addboundedge(int u, int v, FT wl, FT wr)
↪   {
        this->addedge(u, v, wr - wl);
        pot[u] -= wl;
        pot[v] += wl;
    }
    bool feasibleflow() {
        FT tot(0);

        for (int i = 0; i < this->n; i++) {
            if (i != this->s && i != this->t) {
                if (pot[i] > 0)
                    this->addedge(this->s, i,
↪   pot[i]), tot += pot[i];
                else if (pot[i] < 0)
                    this->addedge(i, this->t,
↪   -pot[i]);
            }
        }

        FT ans = this->maxflow();
        return ans == tot;
    }
    void clearaugedge() {
        this->adj[this->s].clear();
        this->adj[this->t].clear();

        for (int i = 0; i < this->n; i++) {
            while (!this->adj[i].empty() &&
↪   (this->adj[i].back().v == this->s ||
↪   this->adj[i].back().v == this->t))
                this->adj[i].pop_back();
        }
    }
};

template<typename FT, template<typename _FT> class
↪   based_graph> class sourced_bounded_flow:
↪   public
    bounded_flow<FT, based_graph> {
public:
    int fs, ft;
    sourced_bounded_flow(int n, int s, int t):
↪   bounded_flow<FT, based_graph>(n), fs(s), ft(t)
↪   {}
    pair<bool, FT> feasibleflow() {
        this->addedge(ft, fs,
↪   numeric_limits<FT>::max());
        bool ret = bounded_flow<FT,
↪   based_graph>::feasibleflow();
        FT ans = this->adj[fs].back().w;
        this->adj[fs].pop_back();
        this->adj[ft].pop_back();
        return make_pair(ret, ans);
    }
    pair<bool, FT> maxfeasibleflow() {
        pair<bool, FT> x = feasibleflow();
        this->clearaugedge();

        if (!x.first)
            return x;

        this->addedge(this->s, fs,
↪   numeric_limits<FT>::max());
        this->addedge(ft, this->t,
↪   numeric_limits<FT>::max());
        x.second += this->maxflow();
        return x;
    }
    pair<bool, FT> minfeasibleflow() {
        pair<bool, FT> x = feasibleflow();
        this->clearaugedge();

        if (!x.first)
            return x;

        this->addedge(this->s, ft,
↪   numeric_limits<FT>::max());
        this->addedge(fs, this->t,
↪   numeric_limits<FT>::max());
        x.second -= this->maxflow();
        return x;
    }
};
```

```cpp
template<typename FT, typename CT, template<typename _FT, typename _CT> class based_graph> class
    cost_bounded_flow : public based_graph<FT, CT> {
public:
    vector<FT> pot;
    CT cs;
    cost_bounded_flow(int n): based_graph<FT, CT>(n + 2, n, n + 1), cs(0) {
        pot.resize(n + 2);
    }
    void addboundedge(int u, int v, FT wl, FT wr, CT c) {
        this->addedge(u, v, wr - wl, c);
        pot[u] -= wl;
        pot[v] += wl;
        cs += c * wl;
    }
    pair<bool, CT> mincostfeasibleflow() {
        FT tot(0);

        for (int i = 0; i < this->n; i++) {
            if (i != this->s && i != this->t) {
                if (pot[i] > 0)
                    this->addedge(this->s, i, pot[i], 0), tot += pot[i];
                else if (pot[i] < 0)
                    this->addedge(i, this->t, -pot[i], 0);
            }
        }

        pair<FT, CT> ans = this->mincostmaxflow();
        return make_pair(ans.first == tot, ans.second + cs);
    }
    void clearaugedge() {
        this->adj[this->s].clear();
        this->adj[this->t].clear();

        for (int i = 0; i < this->n; i++) {
            while (!this->adj[i].empty() && (this->adj[i].back().v == this->s || this->adj[i].back().v == this->t))
                this->adj[i].pop_back();
        }
    }
};

template<typename FT, typename CT, template<typename _FT, typename _CT> class based_graph> class
    sourced_cost_bounded_flow : public
        cost_bounded_flow<FT, CT, based_graph> {
public:
    int fs, ft;
    sourced_cost_bounded_flow(int n, int s, int t): cost_bounded_flow<FT, CT, based_graph>(n), fs(s), ft(t) {}
    pair<bool, pair<FT, CT>> mincostfeasibleflow() {
        this->addedge(ft, fs, numeric_limits<FT>::max(), 0);
        pair<bool, CT> ret = cost_bounded_flow<FT, CT, based_graph>::mincostfeasibleflow();
        FT ans = this->adj[fs].back().w;
        this->adj[fs].pop_back();
        this->adj[ft].pop_back();
        return make_pair(ret.first, make_pair(ans, ret.second));
    }
    pair<bool, pair<FT, CT>> mincostmaxfeasibleflow() {
        pair<bool, pair<FT, CT>> x = mincostfeasibleflow();
        this->clearaugedge();

        if (!x.first)
            return x;

        this->addedge(this->s, fs, numeric_limits<FT>::max(), 0);
        this->addedge(ft, this->t, numeric_limits<FT>::max(), 0);
        pair<FT, CT> g = this->mincostmaxflow();
        x.second.first += g.first;
        x.second.second += g.second;
        return x;
    }
};

}

// 115. 无源汇有上下界可行流
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    int n, m;
    cin >> n >> m;
    flow::bounded_flow<int, flow::dinic_flowgraph> x(n + 1);
    vector<pair<int, int>> idx(m + 1);
    vector<int> lb(m + 1);
    for (int i = 1; i <= m; i++) {
        int s, t, l, r;
        cin >> s >> t >> l >> r;
        lb[i] = l;
        idx[i] = make_pair(t, x[t].size());
        x.addboundedge(s, t, l, r);
    }
    bool ans = x.feasibleflow();
    if (!ans)
        cout << "NO";
    else {
        cout << "YES\n";

        for (int i = 1; i <= m; i++) {
            cout << x[idx[i].first][idx[i].second].w + lb[i] << '\n';
```

```cpp
        }
    }
    return 0;
}

// 116. 有源汇有上下界最大流
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    int n, m, s, t;
    cin >> n >> m >> s >> t;
    flow::sourced_bounded_flow<int,
↪   flow::dinic_flowgraph> x(n + 1, s, t);

    for (int i = 1; i <= m; i++) {
        int s, t, l, r;
        cin >> s >> t >> l >> r;
        x.addboundedge(s, t, l, r);
    }
    pair<bool, int> ans = x.maxfeasibleflow();
    if (!ans.first)
        cout << "please go home to sleep";
    else {
        cout << ans.second;
    }
    return 0;
}

// 117. 有源汇有上下界最小流
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    int n, m, s, t;
    cin >> n >> m >> s >> t;
    flow::sourced_bounded_flow<long long,
↪   flow::dinic_flowgraph> x(n + 1, s, t);
    for (int i = 1; i <= m; i++) {
        int s, t;
        long long l, r;
        cin >> s >> t >> l >> r;
        x.addboundedge(s, t, l, r);
    }
    pair<bool, long long> ans =
↪   x.minfeasibleflow();
    if (!ans.first)
        cout << "please go home to sleep";
    else {
        cout << ans.second;
    }
    return 0;
}
```

8.19 最小费用流

```cpp
// dreadnought
// Q is a priority_queue<PII, vector<PII>,
↪  greater<PII> >
// for an edge(s, t): u is the capacity, v is the
↪  cost, nxt is the next edge,
// op is the opposite edge
// this code can not deal with negative cycles
typedef pair<int,int> PII;
struct edge{ int t,u,v; edge *nxt,*op;
↪  }E[MAXE],*V[MAXV];
int D[MAXN], dist[MAXN], maxflow, mincost; bool
↪  in[MAXN];
bool modlabel(){
  while(!Q.empty()) Q.pop();
  for(int i=S;i<=T;++i) if(in[i])
↪  D[i]=0,Q.push(PII(0,i)); else D[i]=inf;
  while(!Q.empty()){
    int x=Q.top().first,y=Q.top().second; Q.pop();
    if(y==T) break; if(D[y]<x) continue;
    for(edge *ii=V[y];ii;ii=ii->nxt) if(ii->u)
      if(x+(ii->v+dist[ii->t]-dist[y])<D[ii->t]){
        D[ii->t]=x+(ii->v+dist[ii->t]-dist[y]);
        Q.push(PII(D[ii->t],ii->t));
      }
  }
  if(D[T]==inf) return false;
  for(int i=S;i<=T;++i) if(D[i]>D[T])
↪  dist[i]+=D[T]-D[i];
  return true;
}
int aug(int p,int limit){
  if(p==T) return
↪  maxflow+=limit,mincost+=limit*dist[S],limit;
  in[p]=1; int kk,ll=limit;
  for(edge *ii=V[p];ii;ii=ii->nxt) if(ii->u){
    if(!in[ii->t]&&dist[ii->t]+ii->v==dist[p]){
      kk=aug(ii->t,min(ii->u,ll));
↪  ll-=kk,ii->u-=kk,ii->op->u+=kk;
      if(!ll) return in[p]=0,limit;
    }
  }
  return limit-ll;
}
PII mincostFlow(){
  for(int i=S;i<=T;++i) dist[i]=i==T?inf:0;
  while(!Q.empty()) Q.pop(); Q.push(PII(0,T));
  while(!Q.empty()){
    int x=Q.top().first,y=Q.top().second;
↪  Q.pop(); if(dist[y]<x) continue;
    for(edge *ii=V[y];ii;ii=ii->nxt)
↪  if(ii->op->u&&ii->v+x<dist[ii->t]
        dist[ii->t]=ii->v+x,Q.push(PII(dist[ii->t
↪  ],ii->t));
        }
        maxflow=mincost=0;
        do{
        do{
        memset(in,0,sizeof(in));
        }while(aug(S,maxflow));
        }while(modlabel());
        return PII(maxflow,mincost);
        }
```

9 字符串

9.1 后缀树组

```
// input: n, s
// output: sa, rnk, hei
// method: init(const string&); calc_sa();
↪  calc_hei();
struct GetSa {
    int n;
    string s;
    vector<int> sa, rnk, hei;
    GetSa() {}
    void init(const string &_s) {
        s = _s; n = _s.size();
    }
    void calc_sa() {
        sa.resize(n);
        rnk.resize(n);
        vector<int> x(n), y(n);
        for (int i = 0; i < n; ++i) x[i] = s[i];
        int tot = *max_element(ALL(x)) + 1;
        vector<int> cnt(tot);
        for (int i = 0; i < n; ++i) ++cnt[x[i]];
        partial_sum(ALL(cnt), begin(cnt));
        for (int i = 0; i < n; ++i)
            sa[--cnt[x[i]]] = i;
        for (int l = 1; ; l <<= 1) {
            vector<int> cnt(tot);
            int p = n;
            for (int i = n - l; i < n; ++i) y[--p]
↪   = i;
            for (int i = 0; i < n; ++i)
                if (sa[i] >= l) y[--p] = sa[i] -
↪   l;
            for (int i = 0; i < n; ++i)
↪   ++cnt[x[y[i]]];
            partial_sum(ALL(cnt), begin(cnt));
            for (int i = 0; i < n; ++i)
                sa[--cnt[x[y[i]]]] = y[i];
            y[sa[0]] = 0;
            for (int i = 1; i < n; ++i)
                y[sa[i]] = y[sa[i-1]] +
                    (x[sa[i-1]] < x[sa[i]] ||
↪   (sa[i]+l < n && (sa[i-1]+l >= n ||
↪   x[sa[i-1]+l] < x[sa[i]+l])));
            tot = y[sa.back()] + 1;
            x.swap(y);
            if (tot == n) break;
        }
        copy(ALL(x), begin(rnk));
    }
    void calc_hei() {
        hei.resize(n);
        for (int i = 0, j = 0; i < n; ++i) {
            if (!rnk[i]) continue;
            int ii = sa[rnk[i]-1];
            if (j) --j;
            while (ii+j < n && i+j < n && s[ii+j]
↪   == s[i+j]) ++j;
            hei[rnk[i]] = j;
        }
    }
};
```

9.2 后缀自动机

```
// N: length of string
// AL: alphabet size
// method: add(), build()
namespace Sam {
const size_t V = N << 1;
const size_t AL = 26;
int ch[V][AL], par[V], len[V], pos[V], tot = 1,
↪  lst = 1, s[N];
bool ed[V];
void add(int po, int c) {
    int p = lst, np = ++tot;
    s[po] = c;
    len[np] = len[lst] + 1;
    pos[np] = po;
    ed[np] = true;
    for (; p && !ch[p][c]; p = par[p])
        ch[p][c] = np;
    if (p) {
        int q = ch[p][c];
        if (len[p] + 1 == len[q]) {
            par[np] = q;
        } else {
            int nq = ++tot;
            len[nq] = len[p] + 1;
            par[nq] = par[q];
            pos[nq] = pos[q];
            memcpy(ch[nq], ch[q], sizeof ch[q]);
            for (; p && ch[p][c] == q; p = par[p])
                ch[p][c] = nq;
            par[q] = par[np] = nq;
        }
    } else {
        par[np] = 1;
    }
    lst = np;
}
int fch[V][AL], cnt;
void dfs(int u = 1) {
    if (!u) return;
    if (ed[u]) {
        ++cnt;
        sa[cnt] = pos[u];
        rnk[pos[u]] = cnt;
    }
    for (int v : fch[u]) dfs(v);
}
void build() {
    for (int i = 2; i <= tot; ++i)
        fch[par[i]][s[pos[i] + len[par[i]]]] = i;
    dfs();
}
```

9.3 Manacher

```
void manacher(int n, char s[], int f[]) {
    int id = 0, r = 0;
    for (int i = 1; i < n; ++i) {
        f[i] = r > i ? min(f[2 * id - i], r - i) :
↪   1;
        while (f[i] <= i && i + f[i] < n && s[i +
↪   f[i]] == s[i - f[i]])
            ++f[i];
        if (i + f[i] > r) { id = i; r = i + f[i];
↪   }
    }
}
```

9.4 回文自动机

```
// N: length of string
// method: prep, add
namespace PAM {
const size_t AL = 26;
int n, s[N];
int tot, lst, ch[N][AL], par[N], len[N], dep[N];
void prep() {
    par[0] = par[1] = 1;
    s[0] = len[1] = -1;
    lst = tot = 1;
}
int get_link(int x) {
    for (; s[n] != s[n - len[x] - 1]; x = par[x])
↪   {}
    return x;
}
int add(int c) {
    s[++n] = c;
    int p = get_link(lst);
    if (!ch[p][c]) {
        int np = ++tot;
        len[np] = len[p] + 2;
        par[np] = ch[get_link(par[p])][c];
        dep[np] = dep[par[np]] + 1;
        ch[p][c] = np;
    }
    return dep[lst = ch[p][c]];
}
}
```

9.5 Lyndon 分解

```
// input: n, s[]
void lyndon() {
    for (int i = 0; i < n; ) {
        int j = i, k = i + 1;
        for (; k < n && s[j] <= s[k]; ++k)
            j = s[j] < s[k] ? i : j + 1;
        while (i <= j) i += k - j; // right pos
    }
    return 0;
}
```

9.6 Z Function

```
void z_func(string s, int f[]) {
    int l = 0, r = 0;
    for (int i = 1; i < (int) s.size(); ++i) {
        f[i] = i < r ? min(r - i, f[i - l]) : 0;
        while (i + f[i] < (int) s.size() &&
            s[f[i]] == s[i + f[i]]) ++f[i];
        if (i + f[i] > r) r = (l = i) + f[i];
    }
}
```

10 计算几何

10.1 基本操作

```
// Dreadnought
struct Point {
  Point rotate(const double ang) { // 逆时针旋转
↪   ang 弧度
    return Point(cos(ang) * x - sin(ang) * y,
↪   cos(ang) * y + sin(ang) * x);
  }
  Point turn90() { // 逆时针旋转 90 度
    return Point(-y, x);
  }
};
Point isLL(const Line &l1, const Line &l2) {
  double s1 = det(l2.b - l2.a, l1.a - l2.a),
         s2 = -det(l2.b - l2.a, l1.b - l2.a);
  return (l1.a * s2 + l1.b * s1) / (s1 + s2);
}
bool onSeg(const Line &l, const Point &p) { // 点
↪   在线段上
  return sign(det(p - l.a, l.b - l.a)) == 0 &&
↪   sign(dot(p - l.a, p - l.b)) <= 0;
}
Point projection(const Line &l, const Point &p) {
↪   // 点到直线投影
  return l.a + (l.b - l.a) * (dot(p - l.a, l.b -
↪   l.a) / (l.b - l.a).len2());
}
double disToLine(const Line &l, const Point &p) {
  return abs(det(p - l.a, l.b - l.a) / (l.b -
↪   l.a).len());
}
double disToSeg(const Line &l, const Point &p) {
↪   // 点到线段距离
  return sign(dot(p - l.a, l.b - l.a)) *
↪   sign(dot(p - l.b, l.a - l.b)) != 1 ?
    disToLine(l, p) : min((p - l.a).len(), (p -
↪   l.b).len());
}
Point symmetryPoint(const Point a, const Point b)
↪   { // 点 b 关于点 a 的中心对称点
  return a + a - b;
}
Point reflection(const Line &l, const Point &p) {
↪   // 点关于直线的对称点
  return symmetryPoint(projection(l, p), p);
```

```cpp
}
// 求圆与直线的交点
bool isCL(Circle a, Line l, Point &p1, Point &p2) {
  double x = dot(l.a - a.o, l.b - l.a),
         y = (l.b - l.a).len2(),
         d = x * x - y * ((l.a - a.o).len2() - a.r * a.r);
  if (sign(d) < 0) return false;
  d = max(d, 0.0);
  Point p = l.a - ((l.b - l.a) * (x / y)), delta = (l.b - l.a) * (sqrt(d) / y);
  p1 = p + delta, p2 = p - delta;
  return true;
}
// 求圆与圆的交面积
double areaCC(const Circle &c1, const Circle &c2) {
  double d = (c1.o - c2.o).len();
  if (sign(d - (c1.r + c2.r)) >= 0) {
    return 0;
  }
  if (sign(d - abs(c1.r - c2.r)) <= 0) {
    double r = min(c1.r, c2.r);
    return r * r * PI;
  }
  double x = (d * d + c1.r * c1.r - c2.r * c2.r) / (2 * d),
         t1 = acos(x / c1.r), t2 = acos((d - x) / c2.r);
  return c1.r * c1.r * t1 + c2.r * c2.r * t2 - d * c1.r * sin(t1);
}
// 求圆与圆的交点，注意调用前要先判定重圆
bool isCC(Circle a, Circle b, Point &p1, Point &p2) {
  double s1 = (a.o - b.o).len();
  if (sign(s1 - a.r - b.r) > 0 || sign(s1 - abs(a.r - b.r)) < 0) return false;
  double s2 = (a.r * a.r - b.r * b.r) / s1;
  double aa = (s1 + s2) * 0.5, bb = (s1 - s2) * 0.5;
  Point o = (b.o - a.o) * (aa / (aa + bb)) + a.o;
  Point delta = (b.o - a.o).unit().turn90() * newSqrt(a.r * a.r - aa * aa);
  p1 = o + delta, p2 = o - delta;
  return true;
}
// 求点到圆的切点，按关于点的顺时针方向返回两个点
bool tanCP(const Circle &c, const Point &p0, Point &p1, Point &p2) {
  double x = (p0 - c.o).len2(), d = x - c.r * c.r;
  if (d < EPS) return false; // 点在圆上认为没有切点
  Point p = (p0 - c.o) * (c.r * c.r / x);
  Point delta = ((p0 - c.o) * (-c.r * sqrt(d) / x)).turn90();
  p1 = c.o + p + delta;
  p2 = c.o + p - delta;
  return true;
}
// 求圆到圆的外共切线，按关于 c1.o 的顺时针方向返回两条线
vector<Line> extanCC(const Circle &c1, const Circle &c2) {
  vector<Line> ret;
  if (sign(c1.r - c2.r) == 0) {
    Point dir = c2.o - c1.o;
    dir = (dir * (c1.r / dir.len())).turn90();
    ret.push_back(Line(c1.o + dir, c2.o + dir));
    ret.push_back(Line(c1.o - dir, c2.o - dir));
  } else {
    Point p = (c1.o * -c2.r + c2.o * c1.r) / (c1.r - c2.r);
    Point p1, p2, q1, q2;
    if (tanCP(c1, p, p1, p2) && tanCP(c2, p, q1, q2)) {
      if (c1.r < c2.r) swap(p1, p2), swap(q1, q2);
      ret.push_back(Line(p1, q1));
      ret.push_back(Line(p2, q2));
    }
  }
  return ret;
}
// 求圆到圆的内共切线，按关于 c1.o 的顺时针方向返回两条线
vector<Line> intanCC(const Circle &c1, const Circle &c2) {
  vector<Line> ret;
  Point p = (c1.o * c2.r + c2.o * c1.r) / (c1.r + c2.r);
  Point p1, p2, q1, q2;
  if (tanCP(c1, p, p1, p2) && tanCP(c2, p, q1, q2)) { // 两圆相切认为没有切线
    ret.push_back(Line(p1, q1));
    ret.push_back(Line(p2, q2));
  }
  return ret;
}
bool contain(vector<Point> polygon, Point p) { // 判断点 p 是否被多边形包含，包括落在边界上
  int ret = 0, n = polygon.size();
  for(int i = 0; i < n; ++ i) {
    Point u = polygon[i], v = polygon[(i + 1) % n];
    if (onSeg(Line(u, v), p)) return true;
    if (sign(u.y - v.y) <= 0) swap(u, v);
    if (sign(p.y - u.y) > 0 || sign(p.y - v.y) <= 0) continue;
    ret += sign(det(p, v, u)) > 0;
  }
  return ret & 1;
}
vector<Point> convexCut(const vector<Point>&ps, Line l) { // 用半平面 (q1,q2) 的逆时针方向去切凸多边形
  vector<Point> qs;
  int n = ps.size();
  for (int i = 0; i < n; ++i) {
    Point p1 = ps[i], p2 = ps[(i + 1) % n];
```

```
    int d1 = sign(det(l.a, l.b, p1)), d2 =
↪   sign(det(l.a, l.b, p2));
    if (d1 >= 0) qs.push_back(p1);
    if (d1 * d2 < 0) qs.push_back(isLL(Line(p1,
↪   p2), l));
  }
  return qs;
}
vector<Point> convexHull(vector<Point> ps) { // 求
↪   点集 ps 组成的凸包
  int n = ps.size(); if (n <= 1) return ps;
  sort(ps.begin(), ps.end());
  vector<Point> qs;
  for (int i = 0; i < n; qs.push_back(ps[i++]))
    while (qs.size() > 1 &&
↪   sign(det(qs[qs.size()-2],qs.back(),ps[i])) <=
↪   0) qs.pop_back();
  for (int i = n - 2, t = qs.size(); i >= 0;
↪   qs.push_back(ps[i--]))
    while ((int)qs.size() > t &&
↪   sign(det(qs[(int)qs.size()-2],qs.back(),ps[i]))
↪   <= 0) qs.pop_back();
  qs.pop_back(); return qs;
}
```

10.2 半平面交

```
using ld = long double;
const ld eps = 1E-14;

int sign(ld x) {
  return x < -eps ? -1 : x > eps ? 1 : 0;
}

struct Point {
  ld x, y;
  Point(ld x = 0, ld y = 0): x(x), y(y) {}
  Point operator + (const Point &p) const {
    return Point(x + p.x, y + p.y);
  }
  Point operator - (const Point &p) const {
    return Point(x - p.x, y - p.y);
  }
  Point operator * (const ld &k) const {
    return Point(x * k, y * k);
  }
  Point operator / (const ld &k) const {
    return Point(x / k, y / k);
  }
  int quad() const {
    return sign(y) == 1 || (sign(y) == 0 &&
↪   sign(x) >= 0);
  }
  Point turn90() const {
    return Point(-y, x);
  }
  ld mod() const {
    return sqrt(x * x + y * y);
  }
  Point norm() const {
    ld m = mod();
    if (sign(m) == 0) return Point(0, 0);
    return Point(x / m, y / m);
  }
  void input() {
    x = read();
    y = read();
  }
};

ostream& operator << (ostream &os, const Point &p)
↪   {
  return os << "(" << p.x << ", " << p.y << ")";
}

bool operator == (const Point &a, const Point &b)
↪   {
  return sign(a.x - b.x) == 0 && sign(a.y - b.y)
↪   == 0;
}

bool operator < (const Point &a, const Point &b) {
  return sign(a.x - b.x) ? a.x < b.x : sign(a.y -
↪   b.y) ? a.y < b.y : false;
}

bool operator != (const Point &a, const Point &b)
↪   {
  return !(a == b);
}

ld dot(const Point &a, const Point &b) {
  return a.x * b.x + a.y * b.y;
}

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

struct Line {
  Point a, b;
  Line(Point a = Point(), Point b = Point()) :
↪   a(a), b(b) {}
  bool include(const Point &p) const {
    return sign(det(b - a, p - a)) > 0;
  }
  Line push() const {
    Point delta = (b - a).turn90().norm() * eps;
    return Line(a - delta, b - delta);
  }
};

bool on_seg(const Line &l, const Point &p) {
  return sign(det(p - l.a, l.b - l.a)) == 0 &&
    sign(dot(p - l.a, p - l.b)) <= 0;
}

bool parallel(const Line &l1, const Line &l2) {
  return sign(det(l1.b - l1.a, l2.b - l2.a)) == 0;
}

Point intersect(const Line &l1, const Line &l2) {
```

```cpp
  double s1 = det(l2.b - l2.a, l1.a - l2.a);
  double s2 = -det(l2.b - l2.a, l1.b - l2.a);
  return (l1.a * s2 + l1.b * s1) / (s1 + s2);
}

bool same_dir(const Line &l0, const Line &l1) {
  return parallel(l0, l1) && sign(dot(l0.b - l0.a, l1.b - l1.a)) == 1;
}

bool sp_comp_point(const Point &a, const Point &b) {
  if (a.quad() != b.quad()) {
    return a.quad() < b.quad();
  } else {
    return sign(det(a, b)) > 0;
  }
}

bool operator < (const Line &l0, const Line &l1) {
  if (same_dir(l0, l1)) {
    return l1.include(l0.a);
  } else {
    return sp_comp_point(l0.b - l0.a, l1.b - l1.a);
  }
}

bool check(const Line &u, const Line &v, const Line &w) {
  return w.include(intersect(u, v));
}

vector<Point> intersection(vector<Line> l) {
  sort(begin(l), end(l));
  deque<Line> q;
  for (int i = 0; i < (int) l.size(); ++i) {
    if (i && same_dir(l[i], l[i - 1])) continue;
    while (q.size() > 1 && !check(q[q.size() - 2], q.back(), l[i])) q.pop_back();
    while (q.size() > 1 && !check(q[1], q[0], l[i])) q.pop_front();
    q.emplace_back(l[i]);
  }
  while (q.size() > 2 && !check(q[q.size() - 2], q.back(), q[0])) q.pop_back();
  while (q.size() > 2 && !check(q[1], q[0], q.back())) q.pop_front();
  vector<Point> ret;
  for (int i = 0; i < (int) q.size(); ++i) {
    ret.emplace_back(intersect(q[i], q[(i + 1) % q.size()]));
  }
  return ret;
}

ld calc_area(const vector<Point> &vc) {
  ld ret = 0;
  for (int i = 0; i < (int) vc.size(); ++i) {
    ret += det(vc[i], vc[(i + 1) % vc.size()]);
  }
  return ret * .5;
}
```

10.3 凸包操作

```cpp
// Dreadnought
/*
   给定凸包, $\log n$ 内完成各种询问, 具体操作有 :
   1. 判定一个点是否在凸包内
   2. 询问凸包外的点到凸包的两个切点
   3. 询问一个向量关于凸包的切点
   4. 询问一条直线和凸包的交点
   inf 为坐标范围, 需要定义点类大于号
   改成实数只需修改 sign 函数, 以及把 long long 改为 double 即可
   构造函数时传入凸包要求无重点, 面积非空, 以及 pair(x,y) 的最小点放在第一个
   */
const int inf = 1000000000;
struct convex
{
  int n;
  vector<point> a, upper, lower;
  convex(vector<point> _a) : a(_a) {
    n = a.size();
    int ptr = 0;
    for(int i = 1; i < n; ++ i) if (a[ptr] < a[i]) ptr = i;
    for(int i = 0; i <= ptr; ++ i) lower.push_back(a[i]);
    for(int i = ptr; i < n; ++ i) upper.push_back(a[i]);
    upper.push_back(a[0]);
  }
  int sign(long long x) { return x < 0 ? -1 : x > 0; }
  pair<long long, int> get_tangent(vector<point> &convex, point vec) {
    int l = 0, r = (int)convex.size() - 2;
    for( ; l + 1 < r; ) {
      int mid = (l + r) / 2;
      if (sign((convex[mid + 1] - convex[mid]).det(vec)) > 0) r = mid;
      else l = mid;
    }
    return max(make_pair(vec.det(convex[r]), r), make_pair(vec.det(convex[0]), 0));
  }
  void update_tangent(const point &p, int id, int &i0, int &i1) {
    if ((a[i0] - p).det(a[id] - p) > 0) i0 = id;
    if ((a[i1] - p).det(a[id] - p) < 0) i1 = id;
  }
  void binary_search(int l, int r, point p, int &i0, int &i1) {
    if (l == r) return;
    update_tangent(p, l % n, i0, i1);
    int sl = sign((a[l % n] - p).det(a[(l + 1) % n] - p));
    for( ; l + 1 < r; ) {
      int mid = (l + r) / 2;
```

```
      int smid = sign((a[mid % n] - p).det(a[(mid + 1) % n] - p));
      if (smid == sl) l = mid;
      else r = mid;
    }
    update_tangent(p, r % n, i0, i1);
  }
  int binary_search(point u, point v, int l, int r) {
    int sl = sign((v - u).det(a[l % n] - u));
    for( ; l + 1 < r; ) {
      int mid = (l + r) / 2;
      int smid = sign((v - u).det(a[mid % n] - u));
      if (smid == sl) l = mid;
      else r = mid;
    }
    return l % n;
  }
  // 判定点是否在凸包内，在边界返回 true
  bool contain(point p) {
    if (p.x < lower[0].x || p.x > lower.back().x) return false;
    int id = lower_bound(lower.begin(), lower.end(), point(p.x, -inf)) - lower.begin();
    if (lower[id].x == p.x) {
      if (lower[id].y > p.y) return false;
    } else if ((lower[id - 1] - p).det(lower[id] - p) < 0) return false;
    id = lower_bound(upper.begin(), upper.end(), point(p.x, inf), greater<point>()) - upper.begin();
    if (upper[id].x == p.x) {
      if (upper[id].y < p.y) return false;
    } else if ((upper[id - 1] - p).det(upper[id] - p) < 0) return false;
    return true;
  }
  // 求点 p 关于凸包的两个切点，如果在凸包外则有序返回编号，共线的多个切点返回任意一个，否则返回 false
  bool get_tangent(point p, int &i0, int &i1) {
    if (contain(p)) return false;
    i0 = i1 = 0;
    int id = lower_bound(lower.begin(), lower.end(), p) - lower.begin();
    binary_search(0, id, p, i0, i1);
    binary_search(id, (int)lower.size(), p, i0, i1);
    id = lower_bound(upper.begin(), upper.end(), p, greater<point>()) - upper.begin();
    binary_search((int)lower.size() - 1, (int)lower.size() - 1 + id, p, i0, i1);
    binary_search((int)lower.size() - 1 + id, (int)lower.size() - 1 + (int)upper.size(), p, i0, i1);
    return true;
  }
  // 求凸包上和向量 vec 叉积最大的点，返回编号，共线的多个切点返回任意一个
  int get_tangent(point vec) {
    pair<long long, int> ret = get_tangent(upper, vec);
    ret.second = (ret.second + (int)lower.size() - 1) % n;
    ret = max(ret, get_tangent(lower, vec));
    return ret.second;
  }
  // 求凸包和直线 u,v 的交点，如果无严格相交返回 false. 如果有则是和 (i,next(i)) 的交点，两个点无序，交在点上不确定返回前后两条线段其中之一
  bool get_intersection(point u, point v, int &i0, int &i1) {
    int p0 = get_tangent(u - v), p1 = get_tangent(v - u);
    if (sign((v - u).det(a[p0] - u)) * sign((v - u).det(a[p1] - u)) < 0) {
      if (p0 > p1) swap(p0, p1);
      i0 = binary_search(u, v, p0, p1);
      i1 = binary_search(u, v, p1, p0 + n);
      return true;
    } else {
      return false;
    }
  }
};
```

10.4 动态维护凸壳

```
// CodeChef TSUM2
// 动态维护凸壳，求 $x$ 为横坐标时的最大取值
// 一个直线 y = kx + b 可用平面上的点 (k, b) 表示
// 两个点的斜率，即两条直线交点横坐标的相反数，因此可以用两点的斜率衡量某一条直线可否删除
// 具体地，设 l1.k < l2.k < l3.k, l2 可以删除当且仅当 l1 与 l2 的交点 > l2 与 l3 的交点，即 (l2 - l1) % (l3 - l2) > 0
struct Point {
    i64 x, y;
    Point(i64 x = 0, i64 y = 0) :
        x(x), y(y) {}
    Point operator - (const Point &p) const {
        return Point(x - p.x, y - p.y);
    }
    i64 operator % (const Point &p) const {
        return x * p.y - y * p.x;
    }
    bool operator < (const Point &p) const {
        if (x != p.x) return x < p.x;
        return y < p.y;
    }
};
bool comp(const Point &p, const Point &q) { // p's slope greater than q's
    return p % q < 0;
}
struct Node {
    Point p;
```

```cpp
    mutable Point slope;
    bool type;
    Node() : type(false) {}
    Node(Point p) : p(p), type(false) {
    bool operator < (const Node &n) const {
        assert(!type);
        if (n.type) {
            return comp(slope, n.slope);
        } else {
            return p < n.p;
        }
    }
};

struct Hull {
    using iter = set<Node>::iterator;
    set<Node> s;
    Hull() {}
    bool has_lft(iter it) {
        return it != s.begin();
    }
    bool has_rht(iter it) {
        return ++it != s.end();
    }
    void update_border(iter it) {
        {
            if (has_lft(it)) {
                iter jt = it; --jt;
                it->slope = it->p - jt->p;
            } else {
                it->slope = Point(1, (i64) 1e14);
            }
        }
        if (has_rht(it)) {
            iter jt = it; ++jt;
            jt->slope = jt->p - it->p;
        }
    }
    void add(const Point &p) {
        iter it = s.emplace(Node(p)).first, jt, kt;
        if (has_lft(it) && has_rht(it)) {
            jt = it; --jt;
            kt = it; ++kt;
            if (!comp(it->p - jt->p, kt->p - it->p)) {
                s.erase(it);
                return;
            }
        }
        while (has_lft(it)) {
            jt = it; --jt;
            if (has_lft(jt)) {
                kt = jt; --kt;
                if (!comp(jt->p - kt->p, it->p - jt->p)) {
                    s.erase(jt);
                } else {
                    break;
                }
            } else {
                    break;
            }
        }
        while (has_rht(it)) {
            jt = it; ++jt;
            if (has_rht(jt)) {
                kt = jt; ++kt;
                if (!comp(jt->p - it->p, kt->p - jt->p)) {
                    s.erase(jt);
                } else {
                    break;
                }
            } else {
                break;
            }
        }
        update_border(it);
    }

    i64 query(i64 k) {
        assert(!s.empty());
        Node n;
        n.slope = Point(1, -k);
        n.type = true;
        auto it = s.lower_bound(n);
        if (it != s.begin()) --it;
        return k * it->p.x + it->p.y;
    }
};
```

10.5 三角形的心

```cpp
// Dreadnought
Point inCenter(const Point &A, const Point &B, const Point &C) { // 内心
  double a = (B - C).len(), b = (C - A).len(), c = (A - B).len(),
         s = fabs(det(B - A, C - A)),
         r = s / p;
  return (A * a + B * b + C * c) / (a + b + c);
}
Point circumCenter(const Point &a, const Point &b, const Point &c) { // 外心
  Point bb = b - a, cc = c - a;
  double db = bb.len2(), dc = cc.len2(), d = 2 * det(bb, cc);
  return a - Point(bb.y * dc - cc.y * db, cc.x * db - bb.x * dc) / d;
}
Point othroCenter(const Point &a, const Point &b, const Point &c) { // 垂心
  Point ba = b - a, ca = c - a, bc = b - c;
  double Y = ba.y * ca.y * bc.y,
         A = ca.x * ba.y - ba.x * ca.y,
         x0 = (Y + ca.x * ba.y * b.x - ba.x * ca.y * c.x) / A,
         y0 = -ba.x * (x0 - c.x) / ba.y + ca.y;
  return Point(x0, y0);
}
```

10.6 圆与多边形面积交

```
// Dreadnought
double areaCT(Point pa, Point pb, double r) {
  if (pa.len() < pb.len()) swap(pa, pb);
  if (sign(pb.len()) == 0) return 0;
  double a = pb.len(), b = pa.len(), c = (pb -
↪  pa).len();
  double sinB = fabs(det(pb, pb - pa) / a / c),
         cosB = dot(pb, pb - pa) / a / c,
         sinC = fabs(det(pa, pb) / a/ b),
         cosC = dot(pa, pb) / a / b;
  double B = atan2(sinB, cosB), C = atan2(sinC,
↪  cosC);
  if (a > r) {
    S = C / 2 * r * r;
    h = a * b * sinC / c;
    if (h < r && B < PI / 2) {
      S -= (acos(h / r) * r * r - h * sqrt(r * r -
↪  h * h));
    }
  } else if (b > r) {
    double theta = PI - B - asin(sinB / r * a);
    S = a * r * sin(theta) / 2 + (C - theta) / 2 *
↪  r * r;
  } else {
    S = sinC * a * b / 2;
  }
  return S;
}
```

10.7 圆的面积模板 ($n^2 \log n$)

```
// Dreadnought
struct Event {
  Point p;
  double ang;
  int delta;
  Event (Point p = Point(0, 0), double ang = 0,
↪  double delta = 0) : p(p), ang(ang),
↪  delta(delta) {}
};
bool operator < (const Event &a, const Event &b) {
  return a.ang < b.ang;
}
void addEvent(const Circle &a, const Circle &b,
↪  vector<Event> &evt, int &cnt) {
  double d2 = (a.o - b.o).len2(),
         dRatio = ((a.r - b.r) * (a.r + b.r) / d2
↪  + 1) / 2,
         pRatio = sqrt(-(d2 - sqr(a.r - b.r)) *
↪  (d2 - sqr(a.r + b.r)) / (d2 * d2 * 4));
  Point d = b.o - a.o, p = d.rotate(PI / 2),
        q0 = a.o + d * dRatio + p * pRatio,
        q1 = a.o + d * dRatio - p * pRatio;
  double ang0 = (q0 - a.o).ang(),
         ang1 = (q1 - a.o).ang();
  evt.push_back(Event(q1, ang1, 1));
  evt.push_back(Event(q0, ang0, -1));
  cnt += ang1 > ang0;
}
bool issame(const Circle &a, const Circle &b) {
↪  return sign((a.o - b.o).len()) == 0 &&
↪  sign(a.r - b.r) == 0; }
bool overlap(const Circle &a, const Circle &b) {
↪  return sign(a.r - b.r - (a.o - b.o).len()) >=
↪  0; }
bool intersect(const Circle &a, const Circle &b) {
↪  return sign((a.o - b.o).len() - a.r - b.r) <
↪  0; }
int C;
Circle c[N];
double area[N];
void solve() {
  memset(area, 0, sizeof(double) * (C + 1));
  for (int i = 0; i < C; ++i) {
    int cnt = 1;
    vector<Event> evt;
    for (int j = 0; j < i; ++j) if (issame(c[i],
↪  c[j])) ++cnt;
    for (int j = 0; j < C; ++j) {
      if (j != i && !issame(c[i], c[j]) &&
↪  overlap(c[j], c[i])) {
        ++cnt;
      }
    }
    for (int j = 0; j < C; ++j) {
      if (j != i && !overlap(c[j], c[i]) &&
↪  !overlap(c[i], c[j]) && intersect(c[i], c[j]))
↪  {
        addEvent(c[i], c[j], evt, cnt);
      }
    }
    if (evt.size() == 0) {
      area[cnt] += PI * c[i].r * c[i].r;
    } else {
      sort(evt.begin(), evt.end());
      evt.push_back(evt.front());
      for (int j = 0; j + 1 < (int)evt.size();
↪  ++j) {
        cnt += evt[j].delta;
        area[cnt] += det(evt[j].p, evt[j + 1].p) /
↪  2;
        double ang = evt[j + 1].ang - evt[j].ang;
        if (ang < 0) {
          ang += PI * 2;
        }
        area[cnt] += ang * c[i].r * c[i].r / 2 -
↪  sin(ang) * c[i].r * c[i].r / 2;
      }
    }
  }
}
```

10.8 Delaunay 三角剖分

```
// Dreadnought
/*
   Delaunay Triangulation 随机增量算法 :
   节点数至少为点数的 6 倍,
↪  空间消耗较大注意计算内存使用
```

```
  建图的过程在 build 中,
↪  注意初始化内存池和初始三角形的坐标范围
↪  (Triangulation::LOTS)
  Triangulation::find 返回包含某点的三角形
  Triangulation::add_point 将某点加入三角剖分
  某个 Triangle 在三角剖分中当且仅当它的
↪  has_children 为 0
  如果要找到三角形 u 的邻域, 则枚举它的所有
↪  u.edge[i].tri, 该条边的两个点为 u.p[(i+1)%3],
↪  u.p[(i+2)%3]
  */
const int N = 100000 + 5, MAX_TRIS = N * 6;
const double EPSILON = 1e-6, PI = acos(-1.0);
struct Point {
  double x,y; Point():x(0),y(0){} Point(double x,
↪  double y):x(x),y(y){}
  bool operator ==(Point const& that)const
↪  {return x==that.x&&y==that.y;}
};
inline double sqr(double x) { return x*x; }
double dist_sqr(Point const& a, Point const&
↪  b){return sqr(a.x-b.x)+sqr(a.y-b.y);}
bool in_circumcircle(Point const& p1, Point
↪  const& p2, Point const& p3, Point const& p4) {
  double u11 = p1.x - p4.x, u21 = p2.x - p4.x,
↪  u31 = p3.x - p4.x;
  double u12 = p1.y - p4.y, u22 = p2.y - p4.y,
↪  u32 = p3.y - p4.y;
  double u13 = sqr(p1.x) - sqr(p4.x) + sqr(p1.y)
↪  - sqr(p4.y);
  double u23 = sqr(p2.x) - sqr(p4.x) + sqr(p2.y)
↪  - sqr(p4.y);
  double u33 = sqr(p3.x) - sqr(p4.x) + sqr(p3.y)
↪  - sqr(p4.y);
  double det = -u13*u22*u31 + u12*u23*u31 +
↪  u13*u21*u32 - u11*u23*u32 - u12*u21*u33 +
↪  u11*u22*u33;
  return det > EPSILON;
}
double side(Point const& a, Point const& b, Point
↪  const& p) { return (b.x-a.x)*(p.y-a.y) -
↪  (b.y-a.y)*(p.x-a.x);}
typedef int SideRef; struct Triangle; typedef
↪  Triangle* TriangleRef;
struct Edge {
  TriangleRef tri; SideRef side; Edge() : tri(0),
↪  side(0) {}
  Edge(TriangleRef tri, SideRef side) : tri(tri),
↪  side(side) {}
};
struct Triangle {
  Point p[3]; Edge edge[3]; TriangleRef
↪  children[3]; Triangle() {}
  Triangle(Point const& p0, Point const& p1,
↪  Point const& p2) {
    p[0]=p0;p[1]=p1;p[2]=p2;children[0]=children[⌋
↪  1]=children[2]=0;
  }
  bool has_children() const { return children[0]
↪  != 0; }
  int num_children() const {
    return children[0] == 0 ? 0
      : children[1] == 0 ? 1
      : children[2] == 0 ? 2 : 3;
  }
  bool contains(Point const& q) const {
    double a=side(p[0],p[1],q),
↪  b=side(p[1],p[2],q), c=side(p[2],p[0],q);
    return a >= -EPSILON && b >= -EPSILON && c >=
↪  -EPSILON;
  }
} triange_pool[MAX_TRIS], *tot_triangles;
void set_edge(Edge a, Edge b) {
  if (a.tri) a.tri->edge[a.side] = b;
  if (b.tri) b.tri->edge[b.side] = a;
}
class Triangulation {
  public:
    Triangulation() {
      const double LOTS = 1e6;
      the_root = new(tot_triangles++)
↪  Triangle(Point(-LOTS,-LOTS),Point(+LOTS,-LOTS⌋
↪  ),Point(0,+LOTS));
    }
    TriangleRef find(Point p) const { return
↪  find(the_root,p); }
    void add_point(Point const& p) {
↪  add_point(find(the_root,p),p); }
  private:
    TriangleRef the_root;
    static TriangleRef find(TriangleRef root,
↪  Point const& p) {
      for( ; ; ) {
        if (!root->has_children()) return root;
        else for (int i = 0; i < 3 &&
↪  root->children[i] ; ++i)
          if (root->children[i]->contains(p))
          {root = root->children[i]; break;}
      }
    }
    void add_point(TriangleRef root, Point const&
↪  p) {
      TriangleRef tab,tbc,tca;
      tab = new(tot_triangles++)
↪  Triangle(root->p[0], root->p[1], p);
      tbc = new(tot_triangles++)
↪  Triangle(root->p[1], root->p[2], p);
      tca = new(tot_triangles++)
↪  Triangle(root->p[2], root->p[0], p);
      set_edge(Edge(tab,0),Edge(tbc,1));set_edge(⌋
↪  Edge(tbc,0),Edge(tca,1));
      set_edge(Edge(tca,0),Edge(tab,1));set_edge(⌋
↪  Edge(tab,2),root->edge[2]);
      set_edge(Edge(tbc,2),root->edge[0]);set_edg⌋
↪  e(Edge(tca,2),root->edge[1]);
      root->children[0]=tab;root->children[1]=tbc⌋
↪  ;root->children[2]=tca;
      flip(tab,2); flip(tbc,2); flip(tca,2);
    }
    void flip(TriangleRef tri, SideRef pi) {
```

```cpp
      TriangleRef trj = tri->edge[pi].tri; int pj = tri->edge[pi].side;
      if(!trj||!in_circumcircle(tri->p[0],tri->p[1],tri->p[2],trj->p[pj])) return;
      TriangleRef trk = new(tot_triangles++) Triangle(tri->p[(pi+1)%3], trj->p[pj], tri->p[pi]);
      TriangleRef trl = new(tot_triangles++) Triangle(trj->p[(pj+1)%3], tri->p[pi], trj->p[pj]);
      set_edge(Edge(trk,0), Edge(trl,0));
      set_edge(Edge(trk,1), tri->edge[(pi+2)%3]); set_edge(Edge(trk,2), trj->edge[(pj+1)%3]);
      set_edge(Edge(trl,1), trj->edge[(pj+2)%3]); set_edge(Edge(trl,2), tri->edge[(pi+1)%3]);
      tri->children[0]=trk;tri->children[1]=trl;tri->children[2]=0;
      trj->children[0]=trk;trj->children[1]=trl;trj->children[2]=0;
      flip(trk,1); flip(trk,2); flip(trl,1); flip(trl,2);
    }
};
int n; Point ps[N];
void build(){
  tot_triangles = triange_pool; cin >> n;
  for(int i = 0; i < n; ++ i) scanf("%lf%lf",&ps[i].x,&ps[i].y);
  random_shuffle(ps, ps + n); Triangulation tri;
  for(int i = 0; i < n; ++ i) tri.add_point(ps[i]);
}
```

10.9 三维几何基本操作

```cpp
// Dreadnought
struct Point3D {
  double x, y, z;
};
Point3D det(const Point3D &a, const Point3D &b) {
  return Point3D(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);
}
// 平面法向量 : 平面上两个向量叉积
// 点共平面 : 平面上一点与之的向量点积法向量为 0
// 点在线段 ( 直线 ) 上 : 共线且两边点积非正
// 点在三角形内 ( 不包含边界，需再判断是与某条边共线 )
bool pointInTri(const Point3D &a, const Point3D &b, const Point3D &c, const Point3D &p) {
  return sign(det(a - b, a - c).len() - det(p - a, p - b).len() - det(p - b, p - c).len() - det(p - c, p - a).len()) == 0;
}
// 共平面的两点是否在这平面上一条直线的同侧
bool sameSide(const Point3D &a, const Point3D &b, const Point3D &p0, const Point3D &p1) {
  return sign(dot(det(a - b, p0 - b), det(a - b, p1 - b))) > 0;
}
// 两点在平面同侧 : 点积法向量符号相同
// 两直线平行 / 垂直 : 同二维
// 平面平行 / 垂直 : 判断法向量
// 线面垂直 : 法向量和直线平行
// 判断空间线段是否相交 : 四点共面两线段不平行相互在异侧
// 线段和三角形是否相交 : 线段在三角形平面不同侧 三角形任意两点在线段和第三点组成的平面的不同侧
// 求空间直线交点
Point3D intersection(const Point3D &a0, const Point3D &b0, const Point3D &a1, const Point3D &b1) {
  double t = ((a0.x - a1.x) * (a1.y - b1.y) - (a0.y - a1.y) * (a1.x - b1.x)) / ((a0.x - b0.x) * (a1.y - b1.y) - (a0.y - b0.y) * (a1.x - b1.x));
  return a0 + (b0 - a0) * t;
}
// 求平面和直线的交点
Point3D intersection(const Point3D &a, const Point3D &b, const Point3D &c, const Point3D &l0, const Point3D &l1) {
  Point3D p = pVec(a, b, c); // 平面法向量
  double t = (p.x * (a.x - l0.x) + p.y * (a.y - l0.y) + p.z * (a.z - l0.z)) / (p.x * (l1.x - l0.x) + p.y * (l1.y - l0.y) + p.z * (l1.z - l0.z));
  return l0 + (l1 - l0) * t;
}
// 求平面交线 : 取不平行的一条直线的一个交点，以及法向量叉积得到直线方向
// 点到直线距离 : 叉积得到三角形的面积除以底边
// 点到平面距离 : 点积法向量
// 直线间距离 : 平行时随便取一点求距离，否则叉积方向向量得到方向点积计算长度
// 直线夹角 : 点积 平面夹角 : 法向量点积
// 三维向量旋转操作 (绕向量 s 旋转 ang 角度)，对于右手系 s 指向观察者时逆时针
// 矩阵版
void rotate(const Point3D &s, double ang) {
  double l = s.len(), x = s.x / l, y = s.y / l, z = s.z / l, sinA = sin(ang), cosA = cos(ang);
  double p[4][4]= {CosA + (1 - CosA) * x * x, (1 - CosA) * x * y - SinA * z, (1 - CosA) * x * z + SinA * y, 0,
    (1 - CosA) * y * x + SinA * z, CosA + (1 - CosA) * y * y, (1 - CosA) * y * z - SinA * x, 0,
    (1 - CosA) * z * x - SinA * y, (1 - CosA) * z * y + SinA * x, CosA + (1 - CosA) * z * z, 0,
    0, 0, 0, 1 };
}
// 计算版 : 把需要旋转的向量按照 s 分解，做二维旋转，再回到三维
```

10.10 三维凸包

```cpp
// Dreadnought
#define SIZE(X) (int(X.size()))
#define PI 3.14159265358979323846264338327950288
```

```cpp
struct Point {
  Point cross(const Point &p) const {
    return Point(y * p.z - z * p.y, z * p.x - x * p.z, x * p.y - y * p.x);
  }
} info[1005];
int mark[1005][1005],n, cnt;;
double mix(const Point &a, const Point &b, const Point &c) {
  return a.dot(b.cross(c));
}
double area(int a, int b, int c) {
  return ((info[b] - info[a]).cross(info[c] - info[a])).length();
}
double volume(int a, int b, int c, int d) {
  return mix(info[b] - info[a], info[c] - info[a], info[d] - info[a]);
}
struct Face {
  int a, b, c; Face() {}
  Face(int a, int b, int c): a(a), b(b), c(c) {}
  int &operator [](int k) {
    if (k == 0) return a; if (k == 1) return b; return c;
  }
};
vector <Face> face;
inline void insert(int a, int b, int c) {
  face.push_back(Face(a, b, c));
}
void add(int v) {
  vector <Face> tmp; int a, b, c; cnt++;
  for (int i = 0; i < SIZE(face); i++) {
    a = face[i][0]; b = face[i][1]; c = face[i][2];
    if (Sign(volume(v, a, b, c)) < 0)
      mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b] = mark[c][a] = mark[a][c] = cnt;
    else tmp.push_back(face[i]);
  } face = tmp;
  for (int i = 0; i < SIZE(tmp); i++) {
    a = face[i][0]; b = face[i][1]; c = face[i][2];
    if (mark[a][b] == cnt) insert(b, a, v);
    if (mark[b][c] == cnt) insert(c, b, v);
    if (mark[c][a] == cnt) insert(a, c, v);
  }
}
int Find() {
  for (int i = 2; i < n; i++) {
    Point ndir = (info[0] - info[i]).cross(info[1] - info[i]);
    if (ndir == Point()) continue; swap(info[i], info[2]);
    for (int j = i + 1; j < n; j++) if (Sign(volume(0, 1, 2, j)) != 0) {
      swap(info[j], info[3]); insert(0, 1, 2); insert(0, 2, 1); return 1;
    }
  }
  return 0;
}
int main() {
  for (; scanf("%d", &n) == 1; ) {
    for (int i = 0; i < n; i++) info[i].Input();
    sort(info, info + n); n = unique(info, info + n) - info;
    face.clear(); random_shuffle(info, info + n);
    if (Find()) {
      memset(mark, 0, sizeof(mark)); cnt = 0;
      for (int i = 3; i < n; i++) add(i); vector<Point> Ndir;
      for (int i = 0; i < SIZE(face); ++i) {
        Point p = (info[face[i][0]] - info[face[i][1]]).cross(info[face[i][2]] - info[face[i][1]]);
        p = p / p.length(); Ndir.push_back(p);
      }
      sort(Ndir.begin(), Ndir.end());
      int ans = unique(Ndir.begin(), Ndir.end()) - Ndir.begin();
      printf("%d\n", ans);
    } else printf("1\n");
  }
}
// 求重心
double calcDist(const Point &p, int a, int b, int c) {
  return fabs(mix(info[a] - p, info[b] - p, info[c] - p) / area(a, b, c));
}
//compute the minimal distance of center of any faces
double findDist() { //compute center of mass
  double totalWeight = 0;
  Point center(.0, .0, .0);
  Point first = info[face[0][0]];
  for (int i = 0; i < SIZE(face); ++i) {
    Point p = (info[face[i][0]]+info[face[i][1]]+info[face[i][2]]+first)*.25;
    double weight = mix(info[face[i][0]] - first, info[face[i][1]] - first, info[face[i][2]] - first);
    totalWeight += weight; center = center + p * weight;
  }
  center = center / totalWeight;
  double res = 1e100; //compute distance
  for (int i = 0; i < SIZE(face); ++i)
    res = min(res, calcDist(center, face[i][0], face[i][1], face[i][2]));
  return res;
}
```

10.11 求四点外接球

```cpp
// Dreadnought
// 注意，无法处理小于四点的退化情况
int nouter; Tpoint outer[4], res; double radius;
void ball() {
  Tpoint q[3]; double m[3][3], sol[3], L[3], det;
```

```
  int i,j; res.x = res.y = res.z = radius = 0;
  for (i=0; i<3; ++i) q[i]=outer[i+1]-outer[0],
↪  sol[i]=dot(q[i], q[i]);
  for (i=0;i<3;++i) for(j=0;j<3;++j)
↪  m[i][j]=dot(q[i],q[j])*2;
  det= m[0][0]*m[1][1]*m[2][2]
    + m[0][1]*m[1][2]*m[2][0]
    + m[0][2]*m[2][1]*m[1][0]
    - m[0][2]*m[1][1]*m[2][0]
    - m[0][1]*m[1][0]*m[2][2]
    - m[0][0]*m[1][2]*m[2][1];
  if ( fabs(det)<eps ) return;
  for (j=0; j<3; ++j) {
    for (i=0; i<3; ++i) m[i][j]=sol[i];
    L[j]=( m[0][0]*m[1][1]*m[2][2]
        + m[0][1]*m[1][2]*m[2][0]
        + m[0][2]*m[2][1]*m[1][0]
        - m[0][2]*m[1][1]*m[2][0]
        - m[0][1]*m[1][0]*m[2][2]
        - m[0][0]*m[1][2]*m[2][1]
        ) / det;
    for (i=0; i<3; ++i) m[i][j]=dot(q[i], q[j])*2;
  } res=outer[0];
  for (i=0; i<3; ++i ) res = res + q[i] * L[i];
  radius=dist2(res, outer[0]);
}
```

11 其他

11.1 Dancing Links

```
// Dreadnought
// 精确覆盖
const int MAXD = 1120;
const int MAXN = 1000200;

int n, m, t, size;
int U[MAXN], D[MAXN], L[MAXN], R[MAXN], C[MAXN],
↪  Row[MAXN];
int H[MAXD], S[MAXD];

void init(int n, int m) {
  for(int i = 0; i <= m; ++i) {
    S[i] = 0, D[i] = U[i] = i;
    L[i+1] = i, R[i] = i + 1;
  }
  R[m] = 0, size = m;
  for(int i = 1; i <= n; ++i)
    H[i] = -1;
}
void link(int r, int c) {
  ++S[C[++size] = c];
  Row[size] = r;
  D[size] = D[c], U[D[c]] = size;
  U[size] = c, D[c] = size;
  if(H[r] < 0) H[r] = L[size] = R[size] = size;
  else {
    R[size] = R[H[r]], L[R[size]] = size;
    L[size] = H[r];
    R[H[r]] = size;
  }
}
void remove(int c) {
  R[L[c]] = R[c], L[R[c]] = L[c];
  for(int i = D[c]; i != c; i = D[i])
    for(int j = R[i]; j != i; j = R[j])
      U[D[j]] = U[j], D[U[j]] = D[j], -- S[C[j]];
}
void resume(int c) {
  R[L[c]] = L[R[c]] = c;
  for(int i = U[c]; i != c; i = U[i])
    for(int j = L[i]; j != i; j = L[j])
      U[D[j]] = D[U[j]] = j, ++S[C[j]];
}
int ans[MAXD], cnt;
bool dance(int k) {
  int i, j, tmp, c;
  if( !R[0] ) return 1;
  for(tmp = MAXD, i = R[0]; i; i = R[i])
    if(S[i] < tmp) tmp = S[c = i];
  remove(c);
  for(i = D[c]; i != c; i = D[i]) {
    ans[cnt++] = Row[i]; //用栈记录解
    for(j = R[i]; j != i; j = R[j]) remove(C[j]);
    if(dance(k + 1)) return 1;
    --cnt;
    for(j = L[i]; j != i; j = L[j]) resume(C[j]);
  }
  resume(c);
  return 0;
}

// 可重复覆盖
const int mxm = 15 * 15 + 10;
const int MAXD = 15 * 15 + 10;
const int MAXDode = MAXD * mxm;
const int INF = 0x3f3f3f3f;
//能不加的行尽量不加，减少搜索时间
int size;
int U[MAXDode], D[MAXDode], R[MAXDode],
↪  L[MAXDode], Row[MAXDode],  Col[MAXDode];
int H[MAXD], S[mxm];
int ansd;
void init(int n, int m) {
  int i;
  for(i = 0; i <= m; ++i) {
    S[i] = 0, U[i] = D[i] = i;
    L[i] = i - 1, R[i] = i + 1;
  }
  R[m] = 0, L[0] = m, size = m;
  for(i = 1; i <= n; ++i) H[i] = -1;
}
void link(int r, int c) {
  ++S[Col[++size] = c];
  Row[size] = r; D[size] = D[c]; U[D[c]] = size;
↪  U[size] = c; D[c] = size;
  if(H[r] < 0) H[r] = L[size] = R[size] = size;
  else {
    R[size] = R[H[r]];
    L[R[H[r]]] = size;
    L[size] = H[r];
```

```
    R[H[r]] = size;
  }
}
void remove(int c) {
  for(int i = D[c]; i != c; i = D[i])
    L[R[i]] = L[i], R[L[i]] = R[i];
}
void resume(int c) {
  for(int i = U[c]; i != c; i = U[i])
    L[R[i]] = R[L[i]] = i;
}
bool vv[mxm];
int f() {
  int ret = 0, c, i, j;
  for(c = R[0]; c != 0; c = R[c]) vv[c] = 1;
  for(c = R[0]; c != 0; c = R[c])
    if(vv[c]) {
      ++ret, vv[c] = 0;
      for(i = D[c]; i != c; i = D[i])
        for(j = R[i]; j != i; j = R[j])
          vv[Col[j]] = 0;
    }
  return ret;
}
void dance(int d) {
  if(d + f() >= ansd) return;
  if(R[0] == 0) {
    if(d < ansd) ansd = d;
    return;
  }
  int c = R[0], i, j;
  for(i = R[0]; i; i = R[i])
    if(S[i] < S[c]) c = i;
  for(i = D[c]; i != c; i = D[i]) {
    remove(i);
    for(j = R[i]; j != i; j = R[j]) remove(j);
    dance(d + 1);
    for(j = L[i]; j != i; j = L[j]) resume(j);
    resume(i);
  }
}
```

11.2 日期

```
// Dreadnought
int zeller(int y,int m,int d) {
  if (m<=2) y--,m+=12; int c=y/100; y%=100;
  int
↪  w=((c>>2)-(c<<1)+y+(y>>2)+(13*(m+1)/5)+d-1)%7;
  if (w<0) w+=7; return(w);
}
int getId(int y, int m, int d) {
  if (m < 3) {y --; m += 12};
  return 365 * y + y / 4 - y / 100 + y / 400 +
↪  (153 * m + 2) / 5 + d;
}
```

11.3 环状最长公共子序列

```
// Dreadnought
int n, a[N << 1], b[N << 1];
bool has(int i, int j) { return a[(i - 1) % n] ==
↪  b[(j - 1) % n];}
const int DELTA[3][2] = {{0, -1}, {-1, -1}, {-1,
↪  0}};
int from[N][N];
int solve() {
  memset(from, 0, sizeof(from));
  int ret = 0;
  for (int i = 1; i <= 2 * n; ++ i) {
    from[i][0] = 2;
    int left = 0, up = 0;
    for (int j = 1; j <= n; ++ j) {
      int upleft = up + 1 + !!from[i - 1][j];
      if (!has(i, j)) upleft = INT_MIN;
      int max = std::max(left, std::max(upleft,
↪  up));
      if (left == max) {
        from[i][j] = 0;
      } else if (upleft == max) {
        from[i][j] = 1;
      } else {
        from[i][j] = 2;
      }
      left = max;
    }
    if (i >= n) {
      int count = 0;
      for (int x = i, y = n; y;) {
        int t = from[x][y];
        count += t == 1;
        x += DELTA[t][0];
        y += DELTA[t][1];
      }
      ret = std::max(ret, count);
      int x = i - n + 1, y = 0;
      from[x][0] = 0;
      while (y <= n && from[x][y] == 0) y++;
      for (; x <= i; ++ x) {
        from[x][y] = 0;
        if (x == i) break;
        for (; y <= n; ++ y) {
          if (from[x + 1][y] == 2) break;
          if (y + 1 <= n && from[x + 1][y + 1] ==
↪  1) {
            y ++; break;
          }
        }
      }
    }
  }
  return ret;
}
```

11.4 经纬度球面距离

```
// Dreadnought
double sphereDis(double lon1, double lat1, double
↪  lon2, double lat2, double R) {
  return R * acos(cos(lat1) * cos(lat2) * cos(lon1
↪  - lon2) + sin(lat1) * sin(lat2));
}
```

11.5 长方体表面两点最短距离

```cpp
// Dreadnought
int r;
void turn(int i, int j, int x, int y, int z,int x0, int y0, int L, int W, int H) {
  if (z==0) { int R = x*x+y*y; if (R<r) r=R;
  } else {
    if(i>=0 && i< 2) turn(i+1, j, x0+L+z, y, x0+L-x, x0+L, y0, H, W, L);
    if(j>=0 && j< 2) turn(i, j+1, x, y0+W+z, y0+W-y, x0, y0+W, L, H, W);
    if(i<=0 && i>-2) turn(i-1, j, x0-z, y, x-x0, x0-H, y0, H, W, L);
    if(j<=0 && j>-2) turn(i, j-1, x, y0-z, y-y0, x0, y0-H, L, H, W);
  }
}
int main(){
  int L, H, W, x1, y1, z1, x2, y2, z2;
  cin >> L >> W >> H >> x1 >> y1 >> z1 >> x2 >> y2 >> z2;
  if (z1!=0 && z1!=H) if (y1==0 || y1==W)
    swap(y1,z1), std::swap(y2,z2), std::swap(W,H);
  else swap(x1,z1), std::swap(x2,z2), std::swap(L,H);
  if (z1==H) z1=0, z2=H-z2;
  r=0x3fffffff;
  turn(0,0,x2-x1,y2-y1,z2,-x1,-y1,L,W,H);
  cout<<r<<endl;
}
```

11.6 模拟退火

```cpp
void simulateAnneal() {
    const double INIT_TEMP = 2e5;
    const double DELTA = 0.997;
    const double EPS = 1e-14;
    double curx = ansx, cury = ansy;
    for (double temp = INIT_TEMP; temp > EPS; temp *= DELTA) {
        double xx = curx + ((rand() << 1) - RAND_MAX) * temp;
        double yy = cury + ((rand() << 1) - RAND_MAX) * temp;
        double cure = calcEnergy(xx, yy);
        double diff = cure - anse;
        if (diff < 0) {
            ansx = curx = xx;
            ansy = cury = yy;
            anse = cure;
        } else if (exp(-diff / temp) * RAND_MAX > rand()) {
            curx = xx;
            cury = yy;
        }
    }
}
```

11.7 Simpson 积分

```cpp
// Quasar
double area(const double &left, const double &right) {
    double mid = (left + right) / 2;
    return (right - left) * (calc(left) + 4 * calc(mid) + calc(right)) / 6;
}

double simpson(const double &left, const double &right,
               const double &eps, const double &area_sum) {
    double mid = (left + right) / 2;
    double area_left = area(left, mid);
    double area_right = area(mid, right);
    double area_total = area_left + area_right;
    if (std::abs(area_total - area_sum) < 15 * eps) {
        return area_total + (area_total - area_sum) / 15;
    }
    return simpson(left, mid, eps / 2, area_left)
         + simpson(mid, right, eps / 2, area_right);
}

double simpson(const double &left, const double &right, const double &eps) {
    return simpson(left, right, eps, area(left, right));
}
```

11.8 线性规划

```cpp
// Dreadnought
// 求 $\max\{cx\,|\,Ax \leq b, x \geq 0\}$ 的解
typedef vector<double> VD;
VD simplex(vector<VD> A, VD b, VD c) {
  int n = A.size(), m = A[0].size() + 1, r = n, s = m - 1;
  vector<VD> D(n + 2, VD(m + 1, 0)); vector<int> ix(n + m);
  for (int i = 0; i < n + m; ++ i) ix[i] = i;
  for (int i = 0; i < n; ++ i) {
    for (int j = 0; j < m - 1; ++ j) D[i][j] = -A[i][j];
    D[i][m - 1] = 1; D[i][m] = b[i];
    if (D[r][m] > D[i][m]) r = i;
  }
  for (int j = 0; j < m - 1; ++ j) D[n][j] = c[j];
  D[n + 1][m - 1] = -1;
  for (double d; ; ) {
    if (r < n) {
      int t = ix[s]; ix[s] = ix[r + m]; ix[r + m] = t;
      D[r][s] = 1.0 / D[r][s]; vector<int> speedUp;
      for (int j = 0; j <= m; ++ j) if (j != s) {
        D[r][j] *= -D[r][s];
```

```
        if(D[r][j]) speedUp.push_back(j);
      }
      for (int i = 0; i <= n + 1; ++ i) if (i !=
↪   r) {
        for(int j = 0; j < speedUp.size(); ++ j)
          D[i][speedUp[j]] += D[r][speedUp[j]] *
↪   D[i][s];
        D[i][s] *= D[r][s];
      }} r = -1; s = -1;
    for (int j = 0; j < m; ++ j) if (s < 0 ||
↪   ix[s] > ix[j])
      if (D[n + 1][j] > EPS || (D[n + 1][j] > -EPS
↪   && D[n][j] > EPS)) s = j;
    if (s < 0) break;
    for (int i = 0; i < n; ++ i) if (D[i][s] <
↪   -EPS)
      if (r < 0 || (d = D[r][m] / D[r][s] -
↪   D[i][m] / D[i][s]) < -EPS
          || (d < EPS && ix[r + m] > ix[i + m])) r
↪   = i;
    if (r < 0) return VD(); // 无边界
  }
  if (D[n + 1][m] < -EPS) return VD(); // 无解
  VD x(m - 1);
  for (int i = m; i < n + m; ++ i) if (ix[i] < m -
↪   1) x[ix[i]] = D[i - m][m];
  return x; // 最优值在 D[n][m]
}
```

12 积分表

$\int \frac{1}{1+x^2}dx = \tan^{-1}x$ $\int \frac{1}{a^2+x^2}dx = \frac{1}{a}\tan^{-1}\frac{x}{a}$

$\int \frac{x}{a^2+x^2}dx = \frac{1}{2}\ln|a^2+x^2|$ $\int \frac{x^2}{a^2+x^2}dx = x - a\tan^{-1}\frac{x}{a}$

$\int \sqrt{x^2 \pm a^2}dx = \frac{1}{2}x\sqrt{x^2 \pm a^2} \pm \frac{1}{2}a^2 \ln\left|x+\sqrt{x^2 \pm a^2}\right|$

$\int \sqrt{a^2 - x^2}dx = \frac{1}{2}x\sqrt{a^2-x^2} + \frac{1}{2}a^2\tan^{-1}\frac{x}{\sqrt{a^2-x^2}}$

$\int \frac{x^2}{\sqrt{x^2 \pm a^2}}dx = \frac{1}{2}x\sqrt{x^2 \pm a^2} \mp \frac{1}{2}a^2 \ln\left|x+\sqrt{x^2 \pm a^2}\right|$

$\int \frac{1}{\sqrt{x^2 \pm a^2}}dx = \ln\left|x+\sqrt{x^2 \pm a^2}\right|$

$\int \frac{1}{\sqrt{a^2-x^2}}dx = \sin^{-1}\frac{x}{a}$

$\int \frac{x}{\sqrt{x^2 \pm a^2}}dx = \sqrt{x^2 \pm a^2}$

$\int \frac{x}{\sqrt{a^2-x^2}}dx = -\sqrt{a^2-x^2}$

$\int \sqrt{ax^2+bx+c}dx = \frac{b+2ax}{4a}\sqrt{ax^2+bx+c}+$
$\frac{4ac-b^2}{8a^{3/2}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx^+c)}\right|$

$\int x^n e^{ax}\,\mathrm{d}x = \frac{x^n e^{ax}}{a} - \frac{n}{a}\int x^{n-1}e^{ax}\,\mathrm{d}x$

$\int \sin^2 ax dx = \frac{x}{2} - \frac{1}{4a}\sin 2ax$ $\int \sin^3 ax dx = -\frac{3\cos ax}{4a} + \frac{\cos 3ax}{12a}$

$\int \cos^2 ax dx = \frac{x}{2} + \frac{\sin 2ax}{4a}$ $\int \cos^3 ax dx = \frac{3\sin ax}{4a} + \frac{\sin 3ax}{12a}$

$\int \tan ax dx = -\frac{1}{a}\ln\cos ax$ $\int \tan^2 ax dx = -x + \frac{1}{a}\tan ax$

$\int x\cos ax dx = \frac{1}{a^2}\cos ax + \frac{x}{a}\sin ax$ $\int x^2\cos ax dx = \frac{2x\cos ax}{a^2} + \frac{a^2x^2-2}{a^3}\sin ax$

$\int x\sin ax dx = -\frac{x\cos ax}{a} + \frac{\sin ax}{a^2}$ $\int x^2\sin ax dx = \frac{2-a^2x^2}{a^3}\cos ax + \frac{2x\sin ax}{a^2}$

13 Dreadnought

13.1 弦图

设 $next(v)$ 表示 $N(v)$ 中最前的点. 令 $w*$ 表示所有满足 $A \in B$ 的 w 中最后的一个点, 判断 $v \cup N(v)$ 是否为极大团, 只需判断是否存在一个 $w \in w*$, 满足 $Next(w) = v$ 且 $|N(v)| + 1 \le |N(w)|$ 即可.

13.2 重心

半径为 r , 圆心角为 θ 的扇形重心与圆心的距离为 $\frac{4r\sin(\theta/2)}{3\theta}$
半径为 r , 圆心角为 θ 的圆弧重心与圆心的距离为 $\frac{4r\sin^3(\theta/2)}{3(\theta-\sin(\theta))}$

13.3 三角公式

$\sin(a\pm b) = \sin a\cos b \pm \cos a\sin b$ $\cos(a\pm b) = \cos a\cos b \mp \sin a\sin b$

$\tan(a\pm b) = \frac{\tan(a)\pm\tan(b)}{1\mp\tan(a)\tan(b)}$ $\tan(a)\pm\tan(b) = \frac{\sin(a\pm b)}{\cos(a)\cos(b)}$

$\sin(a)+\sin(b) = 2\sin(\frac{a+b}{2})\cos(\frac{a-b}{2})$ $\sin(a)-\sin(b) = 2\cos(\frac{a+b}{2})\sin(\frac{a-b}{2})$

$\cos(a)+\cos(b) = 2\cos(\frac{a+b}{2})\cos(\frac{a-b}{2})$ $\cos(a)-\cos(b) = -2\sin(\frac{a+b}{2})\sin(\frac{a-b}{2})$

$\sin(na) = n\cos^{n-1}a\sin a - \binom{n}{3}\cos^{n-3}a\sin^3 a + \binom{n}{5}\cos^{n-5}a\sin^5 a - \dots$

$\cos(na) = \cos^n a - \binom{n}{2}\cos^{n-2}a\sin^2 a + \binom{n}{4}\cos^{n-4}a\sin^4 a - \dots$

14 Blazar

14.1 博弈

14.1.1 巴什博弈

1. 只有一堆 n 个物品，两个人轮流从这堆物品中取物，规定每次至少取一个，最多取 m 个。最后取光者得胜。
2. 显然，如果 $n = m + 1$，那么由于一次最多只能取 m 个，所以，无论先取者拿走多少个，后取者都能够一次拿走剩余的物品，后者取胜。因此我们发现了如何取胜的法则：如果 $n = \ m + 1 \, r + s$，（r 为任意自然数，$s \le m$), 那么先取者要拿走 s 个物品，如果后取者拿走 $k(k \le m)$ 个，那么先取者再拿走 $m + 1 - k$ 个，结果剩下 $(m+1)(r-1)$ 个，以后保持这样的取法，那么先取者肯定获胜。总之，要保持给对手留下 $(m+1)$ 的倍数，就能最后获胜。

14.1.2 威佐夫博弈

1. 有两堆各若干个物品，两个人轮流从某一堆或同时从两堆中取同样多的物品，规定每次至少取一个，多者不限，最后取光者得胜。
2. 判断一个局势 (a, b) 为奇异局势（必败态）的方法：

$$a_k = [k(1+\sqrt{5})/2] \; b_k = a_k + k$$

14.1.3 阶梯博奕

1. 博弈在一列阶梯上进行，每个阶梯上放着自然数个点，两个人进行阶梯博弈，每一步则是将一个阶梯上的若干个点（至少一个）移到前面去，最后没有点可以移动的人输。
2. 解决方法：把所有奇数阶梯看成 N 堆石子，做 NIM。(把石子从奇数堆移动到偶数堆可以理解为拿走石子，就相当于几个奇数堆的石子在做 Nim)

14.1.4 链的删边游戏

1. 游戏规则：对于一条链，其中一个端点是根，两人轮流删边，脱离根的部分也算被删去，最后没边可删的人输。

2. 做法：$sg[i] = n - dist(i) - 1$（其中 n 表示总点数，$dist(i)$ 表示离根的距离）

14.1.5 树的删边游戏

1. 游戏规则：对于一棵有根树，两人轮流删边，脱离根的部分也算被删去，没边可删的人输。

2. 做法：叶子结点的 $sg = 0$，其他节点的 sg 等于儿子结点的 $sg + 1$ 的异或和。

14.1.6 局部连通图的删边游戏

1. 游戏规则：在一个局部连通图上，两人轮流删边，脱离根的部分也算被删去，没边可删的人输。局部连通图的构图规则是，在一棵基础树上加边得到，所有形成的环保证不共用边，且只与基础树有一个公共点。

2. 做法：去掉所有的偶环，将所有的奇环变为长度为 1 的链，然后做树的删边游戏。

14.2 求和公式

1. $\sum_{k=1}^{n}(2k-1)^2 = \frac{n(4n^2-1)}{3}$

2. $\sum_{k=1}^{n} k^3 = [\frac{n(n+1)}{2}]^2$

3. $\sum_{k=1}^{n}(2k-1)^3 = n^2(2n^2-1)$

4. $\sum_{k=1}^{n} k^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$

5. $\sum_{k=1}^{n} k^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12}$

6. $\sum_{k=1}^{n} k(k+1) = \frac{n(n+1)(n+2)}{3}$

7. $\sum_{k=1}^{n} k(k+1)(k+2) = \frac{n(n+1)(n+2)(n+3)}{4}$

8. $\sum_{k=1}^{n} k(k+1)(k+2)(k+3) = \frac{n(n+1)(n+2)(n+3)(n+4)}{5}$

14.3 斐波那契数列

1. $fib_0 = 0, fib_1 = 1, fib_n = fib_{n-1} + fib_{n-2}$

2. $fib_{n+2} \cdot fib_n - fib_{n+1}^2 = (-1)^{n+1}$

3. $fib_{-n} = (-1)^{n-1} fib_n$

4. $fib_{n+k} = fib_k \cdot fib_{n+1} + fib_{k-1} \cdot fib_n$

5. $gcd(fib_m, fib_n) = fib_{gcd(m,n)}$

6. $fib_m | fib_n^2 \Leftrightarrow nfib_n | m$

14.4 错排公式

1. $D_n = (n-1)(D_{n-2} - D_{n-1})$

2. $D_n = n! \cdot (1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \ldots + \frac{(-1)^n}{n!})$

14.5 欧拉公式

平面图的顶点个数、边数和面的个数有如下关系：

$$V - E + F = C + 1$$

其中，V 是顶点的数目，E 是边的数目，F 是面的数目，C 是组成图形的连通部分的数目。当图是单连通图的时候，公式简化为：

$$V - E + F = 2$$

14.6 皮克定理

给定顶点坐标均是整点（或正方形格点）的简单多边形，其面积 A 和内部格点数目 i、边上格点数目 b 的关系：

$$A = i + \frac{b}{2} - 1$$

14.7 牛顿恒等式

设

$$\prod_{i=1}^{n}(x - x_i) = a_n + a_{n-1}x + \cdots + a_1 x^{n-1} + a_0 x^n$$

$$p_k = \sum_{i=1}^{n} x_i^k$$

则

$$a_0 p_k + a_1 p_{k-1} + \cdots + a_{k-1} p_1 + k a_k = 0$$

特别地，对于

$$|\boldsymbol{A} - \lambda \boldsymbol{E}| = (-1)^n (a_n + a_{n-1}\lambda + \cdots + a_1 \lambda^{n-1} + a_0 \lambda^n)$$

有

$$p_k = \mathrm{Tr}(\boldsymbol{A}^k)$$

14.8 平面几何公式

14.8.1 三角形

1. 半周长

$$p = \frac{a+b+c}{2}$$

2. 面积

$$S = \frac{a \cdot H_a}{2} = \frac{ab \cdot sinC}{2} = \sqrt{p(p-a)(p-b)(p-c)}$$

3. 中线

$$M_a = \frac{\sqrt{2(b^2+c^2) - a^2}}{2} = \frac{\sqrt{b^2 + c^2 + 2bc \cdot cosA}}{2}$$

4. 角平分线

$$T_a = \frac{\sqrt{bc \cdot [(b+c)^2 - a^2]}}{b+c} = \frac{2bc}{b+c} cos\frac{A}{2}$$

5. 高线

$$H_a = bsinC = csinB = \sqrt{b^2 - (\frac{a^2+b^2-c^2}{2a})^2}$$

6. 内切圆半径

$$r = \frac{S}{p} = \frac{arcsin\frac{B}{2} \cdot sin\frac{C}{2}}{sin\frac{B+C}{2}} = 4R \cdot sin\frac{A}{2}sin\frac{B}{2}sin\frac{C}{2}$$
$$= \sqrt{\frac{(p-a)(p-b)(p-c)}{p}} = p \cdot tan\frac{A}{2}tan\frac{B}{2}tan\frac{C}{2}$$

7. 外接圆半径

$$R = \frac{abc}{4S} = \frac{a}{2sinA} = \frac{b}{2sinB} = \frac{c}{2sinC}$$

14.8.2 四边形

D_1, D_2 为对角线，M 对角线中点连线，A 为对角线夹角，p 为半周长

1. $a^2 + b^2 + c^2 + d^2 = D_1^2 + D_2^2 + 4M^2$
2. $S = \frac{1}{2}D_1D_2sinA$
3. 对于圆内接四边形

$$ac + bd = D_1D_2$$

4. 对于圆内接四边形

$$S = \sqrt{(p-a)(p-b)(p-c)(p-d)}$$

14.8.3 正 n 边形

R 为外接圆半径，r 为内切圆半径

1. 中心角

$$A = \frac{2\pi}{n}$$

2. 内角

$$C = \frac{n-2}{n}\pi$$

3. 边长

$$a = 2\sqrt{R^2 - r^2} = 2R \cdot sin\frac{A}{2} = 2r \cdot tan\frac{A}{2}$$

4. 面积

$$S = \frac{nar}{2} = nr^2 \cdot tan\frac{A}{2} = \frac{nR^2}{2} \cdot sinA = \frac{na^2}{4 \cdot tan\frac{A}{2}}$$

14.8.4 圆

1. 弧长

$$l = rA$$

2. 弦长

$$a = 2\sqrt{2hr - h^2} = 2r \cdot sin\frac{A}{2}$$

3. 弓形高

$$h = r - \sqrt{r^2 - \frac{a^2}{4}} = r(1 - cos\frac{A}{2}) = \frac{1}{2} \cdot arctan\frac{A}{4}$$

4. 扇形面积

$$S_1 = \frac{rl}{2} = \frac{r^2A}{2}$$

5. 弓形面积

$$S_2 = \frac{rl - a(r-h)}{2} = \frac{r^2}{2}(A - sinA)$$

14.8.5 棱柱

1. 体积

$$V = Ah$$

A 为底面积，h 为高

2. 侧面积

$$S = lp$$

l 为棱长，p 为直截面周长

3. 全面积

$$T = S + 2A$$

14.8.6 棱锥

1. 体积

$$V = Ah$$

A 为底面积，h 为高

2. 正棱锥侧面积

$$S = lp$$

l 为棱长，p 为直截面周长

3. 正棱锥全面积

$$T = S + 2A$$

14.8.7 棱台

1. 体积

$$V = (A_1 + A_2 + \sqrt{A_1A_2}) \cdot \frac{h}{3}$$

A_1, A_2 为上下底面积，h 为高

2. 正棱台侧面积

$$S = \frac{p_1 + p_2}{2}l$$

p_1, p_2 为上下底面周长，l 为斜高

3. 正棱台全面积

$$T = S + A_1 + A_2$$

14.8.8 圆柱

1. 侧面积

$$S = 2\pi rh$$

2. 全面积

$$T = 2\pi r(h + r)$$

3. 体积

$$V = \pi r^2h$$

14.8.9 圆锥

1. 母线

$$l = \sqrt{h^2 + r^2}$$

2. 侧面积

$$S = \pi rl$$

3. 全面积

$$T = \pi r(l + r)$$

4. 体积

$$V = \frac{\pi}{3}r^2h$$

14.8.10 圆台

1. 母线
$$l = \sqrt{h^2 + (r_1 - r_2)^2}$$
2. 侧面积
$$S = \pi(r_1 + r_2)l$$
3. 全面积
$$T = \pi r_1(l + r_1) + \pi r_2(l + r_2)$$
4. 体积
$$V = \frac{\pi}{3}(r_1^2 + r_2^2 + r_1 r_2)h$$

14.8.11 球

1. 全面积
$$T = 4\pi r^2$$
2. 体积
$$V = \frac{4}{3}\pi r^3$$

14.8.12 球台

1. 侧面积
$$S = 2\pi rh$$
2. 全面积
$$T = \pi(2rh + r_1^2 + r_2^2)$$
3. 体积
$$V = \frac{\pi h[3(r_1^2 + r_2^2) + h^2]}{6}$$

14.8.13 球扇形

1. 全面积
$$T = \pi r(2h + r_0)$$
h 为球冠高，r_0 为球冠底面半径
2. 体积
$$V = \frac{2}{3}\pi r^2 h$$

14.9 立体几何公式

14.9.1 球面三角公式

设 a, b, c 是边长，A, B, C 是所对的二面角，有余弦定理

$$cosa = cosb \cdot cosc + sinb \cdot sinc \cdot cosA$$

正弦定理
$$\frac{sinA}{sina} = \frac{sinB}{sinb} = \frac{sinC}{sinc}$$
三角形面积是 $A + B + C - \pi$

14.9.2 四面体体积公式

U, V, W, u, v, w 是四面体的 6 条棱，U, V, W 构成三角形，$(U, u), (V, v), (W, w)$ 互为对棱，则

$$V = \frac{\sqrt{(s - 2a)(s - 2b)(s - 2c)(s - 2d)}}{192uvw}$$

其中
$$\begin{cases} a & = & \sqrt{xYZ}, \\ b & = & \sqrt{yZX}, \\ c & = & \sqrt{zXY}, \\ d & = & \sqrt{xyz}, \\ s & = & a + b + c + d, \\ X & = & (w - U + v)(U + v + w), \\ x & = & (U - v + w)(v - w + U), \\ Y & = & (u - V + w)(V + w + u), \\ y & = & (V - w + u)(w - u + V), \\ Z & = & (v - W + u)(W + u + v), \\ z & = & (W - u + v)(u - v + W) \end{cases}$$

15 常见问题

15.1 编译指令

1. -O2 -g -std=c++11
2. -Wall -Wextra -Wconversion
3. -fsantitze=(address/undefined)：检查有符号整数溢出 (无符号整数溢出不算 ub)、数组越界

15.2 问题汇总

1. 数组或者变量类型开错，例如将 double 开成 int，或忘开 long long；
2. 函数忘记返回返回值；
3. 初始化数组没有初始化完全；
4. 对空间限制判断不足导致 MLE；
5. 多测清空，清空所有全局变量、数组（不是只有数组）；
6. n, m，i, j 打错、打反；
7. 边界 ± 1 的情况；
8. 访问负下标，$+n$ 是否足够；
9. 输入量达到 5×10^6 可以考虑快读，并采用缓冲区；
10. PE 显示成了 WA，或其他错误。

16 cheat.pdf

Theoretical Computer Science Cheat Sheet

Definitions

$f(n) = O(g(n))$	iff $\exists$ positive c, n_0 such that $0 \le f(n) \le cg(n)\ \forall n \ge n_0$.
$f(n) = \Omega(g(n))$	iff $\exists$ positive c, n_0 such that $f(n) \ge cg(n) \ge 0\ \forall n \ge n_0$.
$f(n) = \Theta(g(n))$	iff $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.
$f(n) = o(g(n))$	iff $\lim_{n\to\infty} f(n)/g(n) = 0$.
$\lim\limits_{n\to\infty} a_n = a$	iff $\forall \epsilon > 0$, $\exists n_0$ such that $\lvert a_n - a\rvert < \epsilon$, $\forall n \ge n_0$.
$\sup S$	least $b \in \mathbb{R}$ such that $b \ge s$, $\forall s \in S$.
$\inf S$	greatest $b \in \mathbb{R}$ such that $b \le s$, $\forall s \in S$.
$\liminf\limits_{n\to\infty} a_n$	$\lim\limits_{n\to\infty} \inf\{a_i \mid i \ge n, i \in \mathbb{N}\}$.
$\limsup\limits_{n\to\infty} a_n$	$\lim\limits_{n\to\infty} \sup\{a_i \mid i \ge n, i \in \mathbb{N}\}$.
$\binom{n}{k}$	Combinations: Size k subsets of a size n set.
$\left[{n \atop k}\right]$	Stirling numbers (1st kind): Arrangements of an n element set into k cycles.
$\left\{{n \atop k}\right\}$	Stirling numbers (2nd kind): Partitions of an n element set into k non-empty sets.
$\left\langle{n \atop k}\right\rangle$	1st order Eulerian numbers: Permutations $\pi_1\pi_2\ldots\pi_n$ on $\{1, 2, \ldots, n\}$ with k ascents.
$\left\langle\!\left\langle{n \atop k}\right\rangle\!\right\rangle$	2nd order Eulerian numbers.
C_n	Catalan Numbers: Binary trees with $n+1$ vertices.

Series

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}, \quad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}.$$

In general:

$$\sum_{i=1}^{n} i^m = \frac{1}{m+1}\left[(n+1)^{m+1} - 1 - \sum_{i=1}^{n}\left((i+1)^{m+1} - i^{m+1} - (m+1)i^m\right)\right]$$

$$\sum_{i=1}^{n-1} i^m = \frac{1}{m+1}\sum_{k=0}^{m}\binom{m+1}{k} B_k n^{m+1-k}.$$

Geometric series:

$$\sum_{i=0}^{n} c^i = \frac{c^{n+1}-1}{c-1}, \quad c \ne 1, \quad \sum_{i=0}^{\infty} c^i = \frac{1}{1-c}, \quad \sum_{i=1}^{\infty} c^i = \frac{c}{1-c}, \quad |c| < 1,$$

$$\sum_{i=0}^{n} ic^i = \frac{nc^{n+2} - (n+1)c^{n+1} + c}{(c-1)^2}, \quad c \ne 1, \quad \sum_{i=0}^{\infty} ic^i = \frac{c}{(1-c)^2}, \quad |c| < 1.$$

Harmonic series:

$$H_n = \sum_{i=1}^{n} \frac{1}{i}, \quad \sum_{i=1}^{n} iH_i = \frac{n(n+1)}{2}H_n - \frac{n(n-1)}{4}.$$

$$\sum_{i=1}^{n} H_i = (n+1)H_n - n, \quad \sum_{i=1}^{n}\binom{i}{m}H_i = \binom{n+1}{m+1}\left(H_{n+1} - \frac{1}{m+1}\right).$$

1. $\displaystyle\binom{n}{k} = \frac{n!}{(n-k)!k!}$, **2.** $\displaystyle\sum_{k=0}^{n}\binom{n}{k} = 2^n$, **3.** $\displaystyle\binom{n}{k} = \binom{n}{n-k}$,

4. $\displaystyle\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$, **5.** $\displaystyle\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$,

6. $\displaystyle\binom{n}{m}\binom{m}{k} = \binom{n}{k}\binom{n-k}{m-k}$, **7.** $\displaystyle\sum_{k=0}^{n}\binom{r+k}{k} = \binom{r+n+1}{n}$,

8. $\displaystyle\sum_{k=0}^{n}\binom{k}{m} = \binom{n+1}{m+1}$, **9.** $\displaystyle\sum_{k=0}^{n}\binom{r}{k}\binom{s}{n-k} = \binom{r+s}{n}$,

10. $\displaystyle\binom{n}{k} = (-1)^k\binom{k-n-1}{k}$, **11.** $\displaystyle\left\{{n \atop 1}\right\} = \left\{{n \atop n}\right\} = 1$,

12. $\displaystyle\left\{{n \atop 2}\right\} = 2^{n-1} - 1$, **13.** $\displaystyle\left\{{n \atop k}\right\} = k\left\{{n-1 \atop k}\right\} + \left\{{n-1 \atop k-1}\right\}$,

14. $\displaystyle\left[{n \atop 1}\right] = (n-1)!$, **15.** $\displaystyle\left[{n \atop 2}\right] = (n-1)!H_{n-1}$, **16.** $\displaystyle\left[{n \atop n}\right] = 1$, **17.** $\displaystyle\left[{n \atop k}\right] \ge \left\{{n \atop k}\right\}$,

18. $\displaystyle\left[{n \atop k}\right] = (n-1)\left[{n-1 \atop k}\right] + \left[{n-1 \atop k-1}\right]$, **19.** $\displaystyle\left\{{n \atop n-1}\right\} = \left[{n \atop n-1}\right] = \binom{n}{2}$, **20.** $\displaystyle\sum_{k=0}^{n}\left[{n \atop k}\right] = n!$, **21.** $\displaystyle C_n = \frac{1}{n+1}\binom{2n}{n}$,

22. $\displaystyle\left\langle{n \atop 0}\right\rangle = \left\langle{n \atop n-1}\right\rangle = 1$, **23.** $\displaystyle\left\langle{n \atop k}\right\rangle = \left\langle{n \atop n-1-k}\right\rangle$, **24.** $\displaystyle\left\langle{n \atop k}\right\rangle = (k+1)\left\langle{n-1 \atop k}\right\rangle + (n-k)\left\langle{n-1 \atop k-1}\right\rangle$,

25. $\displaystyle\left\langle{0 \atop k}\right\rangle = \begin{cases} 1 & \text{if } k = 0, \\ 0 & \text{otherwise} \end{cases}$ **26.** $\displaystyle\left\langle{n \atop 1}\right\rangle = 2^n - n - 1$, **27.** $\displaystyle\left\langle{n \atop 2}\right\rangle = 3^n - (n+1)2^n + \binom{n+1}{2}$,

28. $\displaystyle x^n = \sum_{k=0}^{n}\left\langle{n \atop k}\right\rangle\binom{x+k}{n}$, **29.** $\displaystyle\left\langle{n \atop m}\right\rangle = \sum_{k=0}^{m}\binom{n+1}{k}(m+1-k)^n(-1)^k$, **30.** $\displaystyle m!\left\{{n \atop m}\right\} = \sum_{k=0}^{n}\left\langle{n \atop k}\right\rangle\binom{k}{n-m}$,

31. $\displaystyle\left\langle{n \atop m}\right\rangle = \sum_{k=0}^{n}\left\{{n \atop k}\right\}\binom{n-k}{m}(-1)^{n-k-m}k!$, **32.** $\displaystyle\left\langle\!\left\langle{n \atop 0}\right\rangle\!\right\rangle = 1$, **33.** $\displaystyle\left\langle\!\left\langle{n \atop n}\right\rangle\!\right\rangle = 0 \quad \text{for } n \ne 0$,

34. $\displaystyle\left\langle\!\left\langle{n \atop k}\right\rangle\!\right\rangle = (k+1)\left\langle\!\left\langle{n-1 \atop k}\right\rangle\!\right\rangle + (2n-1-k)\left\langle\!\left\langle{n-1 \atop k-1}\right\rangle\!\right\rangle$, **35.** $\displaystyle\sum_{k=0}^{n}\left\langle\!\left\langle{n \atop k}\right\rangle\!\right\rangle = \frac{(2n)^{\underline{n}}}{2^n}$,

36. $\displaystyle\left\{{x \atop x-n}\right\} = \sum_{k=0}^{n}\left\langle\!\left\langle{n \atop k}\right\rangle\!\right\rangle\binom{x+n-1-k}{2n}$, **37.** $\displaystyle\left\{{n+1 \atop m+1}\right\} = \sum_{k}\binom{n}{k}\left\{{k \atop m}\right\} = \sum_{k=0}^{n}\left\{{k \atop m}\right\}(m+1)^{n-k}$,

Theoretical Computer Science Cheat Sheet

Identities Cont.

38. $\left[{n+1 \atop m+1}\right] = \sum_k \left[{n \atop k}\right]\binom{k}{m} = \sum_{k=0}^{n} \left[{k \atop m}\right] n^{\underline{n-k}} = n! \sum_{k=0}^{n} \frac{1}{k!}\left[{k \atop m}\right],$

39. $\left[{x \atop x-n}\right] = \sum_{k=0}^{n} \left\langle\!\!\left\langle {n \atop k} \right\rangle\!\!\right\rangle \binom{x+k}{2n},$

40. $\left\{{n \atop m}\right\} = \sum_k \binom{n}{k}\left\{{k+1 \atop m+1}\right\}(-1)^{n-k},$

41. $\left[{n \atop m}\right] = \sum_k \left[{n+1 \atop k+1}\right]\binom{k}{m}(-1)^{m-k},$

42. $\left\{{m+n+1 \atop m}\right\} = \sum_{k=0}^{m} k\left\{{n+k \atop k}\right\},$

43. $\left[{m+n+1 \atop m}\right] = \sum_{k=0}^{m} k(n+k)\left[{n+k \atop k}\right],$

44. $\binom{n}{m} = \sum_k \left\{{n+1 \atop k+1}\right\}\left[{k \atop m}\right](-1)^{m-k},$

45. $(n-m)!\binom{n}{m} = \sum_k \left[{n+1 \atop k+1}\right]\left\{{k \atop m}\right\}(-1)^{m-k}, \quad \text{for } n \ge m,$

46. $\left\{{n \atop n-m}\right\} = \sum_k \binom{m-n}{m+k}\binom{m+n}{n+k}\left[{m+k \atop k}\right],$

47. $\left[{n \atop n-m}\right] = \sum_k \binom{m-n}{m+k}\binom{m+n}{n+k}\left\{{m+k \atop k}\right\},$

48. $\left\{{n \atop \ell+m}\right\}\binom{\ell+m}{\ell} = \sum_k \left\{{k \atop \ell}\right\}\left\{{n-k \atop m}\right\}\binom{n}{k},$

49. $\left[{n \atop \ell+m}\right]\binom{\ell+m}{\ell} = \sum_k \left[{k \atop \ell}\right]\left[{n-k \atop m}\right]\binom{n}{k}.$

Trees

Every tree with n vertices has $n-1$ edges.

Kraft inequality: If the depths of the leaves of a binary tree are $d_1, \ldots, d_n$:
$$\sum_{i=1}^{n} 2^{-d_i} \le 1,$$
and equality holds only if every internal node has 2 sons.

Recurrences

Master method:
$$T(n) = aT(n/b) + f(n), \quad a \ge 1, b > 1$$

If $\exists \epsilon > 0$ such that $f(n) = O(n^{\log_b a - \epsilon})$ then
$$T(n) = \Theta(n^{\log_b a}).$$

If $f(n) = \Theta(n^{\log_b a})$ then
$$T(n) = \Theta(n^{\log_b a} \log_2 n).$$

If $\exists \epsilon > 0$ such that $f(n) = \Omega(n^{\log_b a + \epsilon})$, and $\exists c < 1$ such that $af(n/b) \le cf(n)$ for large n, then
$$T(n) = \Theta(f(n)).$$

Substitution (example): Consider the following recurrence
$$T_{i+1} = 2^{2^i} \cdot T_i^2, \quad T_1 = 2.$$

Note that T_i is always a power of two. Let $t_i = \log_2 T_i$. Then we have
$$t_{i+1} = 2^i + 2t_i, \quad t_1 = 1.$$

Let $u_i = t_i/2^i$. Dividing both sides of the previous equation by 2^{i+1} we get
$$\frac{t_{i+1}}{2^{i+1}} = \frac{2^i}{2^{i+1}} + \frac{t_i}{2^i}.$$

Substituting we find
$$u_{i+1} = \tfrac{1}{2} + u_i, \qquad u_1 = \tfrac{1}{2},$$

which is simply $u_i = i/2$. So we find that T_i has the closed form $T_i = 2^{i2^{i-1}}$.

Summing factors (example): Consider the following recurrence
$$T(n) = 3T(n/2) + n, \quad T(1) = 1.$$

Rewrite so that all terms involving T are on the left side
$$T(n) - 3T(n/2) = n.$$

Now expand the recurrence, and choose a factor which makes the left side "telescope"
$$\begin{aligned}
1\big(T(n) - 3T(n/2) &= n\big)\\
3\big(T(n/2) - 3T(n/4) &= n/2\big)\\
\vdots \quad\quad \vdots \quad &\quad \vdots\\
3^{\log_2 n - 1}\big(T(2) - 3T(1) &= 2\big)
\end{aligned}$$

Let $m = \log_2 n$. Summing the left side we get $T(n) - 3^m T(1) = T(n) - 3^m = T(n) - n^k$ where $k = \log_2 3 \approx 1.58496$. Summing the right side we get
$$\sum_{i=0}^{m-1} \frac{n}{2^i} 3^i = n \sum_{i=0}^{m-1} \left(\tfrac{3}{2}\right)^i.$$

Let $c = \frac{3}{2}$. Then we have
$$\begin{aligned}
n \sum_{i=0}^{m-1} c^i &= n\left(\frac{c^m - 1}{c - 1}\right)\\
&= 2n(c^{\log_2 n} - 1)\\
&= 2n(c^{(k-1)\log_c n} - 1)\\
&= 2n^k - 2n,
\end{aligned}$$

and so $T(n) = 3n^k - 2n$. Full history recurrences can often be changed to limited history ones (example): Consider
$$T_i = 1 + \sum_{j=0}^{i-1} T_j, \quad T_0 = 1.$$

Note that
$$T_{i+1} = 1 + \sum_{j=0}^{i} T_j.$$

Subtracting we find
$$\begin{aligned}
T_{i+1} - T_i &= 1 + \sum_{j=0}^{i} T_j - 1 - \sum_{j=0}^{i-1} T_j\\
&= T_i.
\end{aligned}$$

And so $T_{i+1} = 2T_i = 2^{i+1}$.

Generating functions:

1. Multiply both sides of the equation by x^i.
2. Sum both sides over all i for which the equation is valid.
3. Choose a generating function $G(x)$. Usually $G(x) = \sum_{i=0}^{\infty} x^i g_i$.
3. Rewrite the equation in terms of the generating function $G(x)$.
4. Solve for $G(x)$.
5. The coefficient of x^i in $G(x)$ is g_i.

Example:
$$g_{i+1} = 2g_i + 1, \quad g_0 = 0.$$

Multiply and sum:
$$\sum_{i\ge 0} g_{i+1} x^i = \sum_{i \ge 0} 2g_i x^i + \sum_{i \ge 0} x^i.$$

We choose $G(x) = \sum_{i \ge 0} x^i g_i$. Rewrite in terms of $G(x)$:
$$\frac{G(x) - g_0}{x} = 2G(x) + \sum_{i \ge 0} x^i.$$

Simplify:
$$\frac{G(x)}{x} = 2G(x) + \frac{1}{1-x}.$$

Solve for $G(x)$:
$$G(x) = \frac{x}{(1-x)(1-2x)}.$$

Expand this using partial fractions:
$$\begin{aligned}
G(x) &= x\left(\frac{2}{1-2x} - \frac{1}{1-x}\right)\\
&= x\left(2\sum_{i\ge 0} 2^i x^i - \sum_{i \ge 0} x^i\right)\\
&= \sum_{i \ge 0} (2^{i+1} - 1)x^{i+1}.
\end{aligned}$$

So $g_i = 2^i - 1$.

Theoretical Computer Science Cheat Sheet

$\pi \approx 3.14159$, $\quad e \approx 2.71828$, $\quad \gamma \approx 0.57721$, $\quad \phi = \frac{1+\sqrt{5}}{2} \approx 1.61803$, $\quad \hat{\phi} = \frac{1-\sqrt{5}}{2} \approx -.61803$

i	2^i	p_i
1	2	2
2	4	3
3	8	5
4	16	7
5	32	11
6	64	13
7	128	17
8	256	19
9	512	23
10	1,024	29
11	2,048	31
12	4,096	37
13	8,192	41
14	16,384	43
15	32,768	47
16	65,536	53
17	131,072	59
18	262,144	61
19	524,288	67
20	1,048,576	71
21	2,097,152	73
22	4,194,304	79
23	8,388,608	83
24	16,777,216	89
25	33,554,432	97
26	67,108,864	101
27	134,217,728	103
28	268,435,456	107
29	536,870,912	109
30	1,073,741,824	113
31	2,147,483,648	127
32	4,294,967,296	131

Pascal's Triangle

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1

General

Bernoulli Numbers ($B_i = 0$, odd $i \neq 1$):

$$B_0 = 1,\ B_1 = -\tfrac{1}{2},\ B_2 = \tfrac{1}{6},\ B_4 = -\tfrac{1}{30},$$
$$B_6 = \tfrac{1}{42},\ B_8 = -\tfrac{1}{30},\ B_{10} = \tfrac{5}{66}.$$

Change of base, quadratic formula:

$$\log_b x = \frac{\log_a x}{\log_a b}, \qquad \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Euler's number e:

$$e = 1 + \tfrac{1}{2} + \tfrac{1}{6} + \tfrac{1}{24} + \tfrac{1}{120} + \cdots$$
$$\lim_{n\to\infty}\left(1 + \frac{x}{n}\right)^n = e^x.$$
$$\left(1 + \tfrac{1}{n}\right)^n < e < \left(1 + \tfrac{1}{n}\right)^{n+1}.$$
$$\left(1 + \tfrac{1}{n}\right)^n = e - \frac{e}{2n} + \frac{11e}{24n^2} - O\left(\frac{1}{n^3}\right).$$

Harmonic numbers:

$$1,\ \tfrac{3}{2},\ \tfrac{11}{6},\ \tfrac{25}{12},\ \tfrac{137}{60},\ \tfrac{49}{20},\ \tfrac{363}{140},\ \tfrac{761}{280},\ \tfrac{7129}{2520},\ \ldots$$

$$\ln n < H_n < \ln n + 1,$$
$$H_n = \ln n + \gamma + O\left(\frac{1}{n}\right).$$

Factorial, Stirling's approximation:

$$1,\ 2,\ 6,\ 24,\ 120,\ 720,\ 5040,\ 40320,\ 362880,\ \ldots$$

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right).$$

Ackermann's function and inverse:

$$a(i,j) = \begin{cases} 2^j & i = 1 \\ a(i-1, 2) & j = 1 \\ a(i-1, a(i, j-1)) & i, j \geq 2 \end{cases}$$
$$\alpha(i) = \min\{j \mid a(j,j) \geq i\}.$$

Binomial distribution:

$$\Pr[X = k] = \binom{n}{k} p^k q^{n-k}, \qquad q = 1 - p,$$
$$\mathrm{E}[X] = \sum_{k=1}^{n} k \binom{n}{k} p^k q^{n-k} = np.$$

Poisson distribution:

$$\Pr[X = k] = \frac{e^{-\lambda}\lambda^k}{k!}, \quad \mathrm{E}[X] = \lambda.$$

Normal (Gaussian) distribution:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}, \quad \mathrm{E}[X] = \mu.$$

The "coupon collector": We are given a random coupon each day, and there are n different types of coupons. The distribution of coupons is uniform. The expected number of days to pass before we to collect all n types is

$$nH_n.$$

Probability

Continuous distributions: If

$$\Pr[a < X < b] = \int_a^b p(x)\, dx,$$

then p is the probability density function of X. If

$$\Pr[X < a] = P(a),$$

then P is the distribution function of X. If P and p both exist then

$$P(a) = \int_{-\infty}^{a} p(x)\, dx.$$

Expectation: If X is discrete

$$\mathrm{E}[g(X)] = \sum_x g(x) \Pr[X = x].$$

If X continuous then

$$\mathrm{E}[g(X)] = \int_{-\infty}^{\infty} g(x) p(x)\, dx = \int_{-\infty}^{\infty} g(x)\, dP(x).$$

Variance, standard deviation:

$$\mathrm{VAR}[X] = \mathrm{E}[X^2] - \mathrm{E}[X]^2,$$
$$\sigma = \sqrt{\mathrm{VAR}[X]}.$$

For events A and B:

$$\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$$
$$\Pr[A \wedge B] = \Pr[A] \cdot \Pr[B],$$

iff A and B are independent.

$$\Pr[A|B] = \frac{\Pr[A \wedge B]}{\Pr[B]}$$

For random variables X and Y:

$$\mathrm{E}[X \cdot Y] = \mathrm{E}[X] \cdot \mathrm{E}[Y],$$

if X and Y are independent.

$$\mathrm{E}[X + Y] = \mathrm{E}[X] + \mathrm{E}[Y],$$
$$\mathrm{E}[cX] = c\,\mathrm{E}[X].$$

Bayes' theorem:

$$\Pr[A_i|B] = \frac{\Pr[B|A_i]\Pr[A_i]}{\sum_{j=1}^{n}\Pr[A_j]\Pr[B|A_j]}.$$

Inclusion-exclusion:

$$\Pr\left[\bigvee_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \Pr[X_i] +$$
$$\sum_{k=2}^{n} (-1)^{k+1} \sum_{i_i < \cdots < i_k} \Pr\left[\bigwedge_{j=1}^{k} X_{i_j}\right].$$

Moment inequalities:

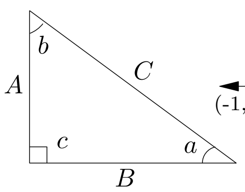
$$\Pr\left[|X| \geq \lambda\,\mathrm{E}[X]\right] \leq \frac{1}{\lambda},$$
$$\Pr\left[\left|X - \mathrm{E}[X]\right| \geq \lambda \cdot \sigma\right] \leq \frac{1}{\lambda^2}.$$

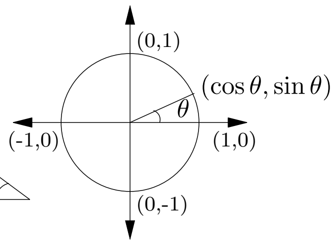
Geometric distribution:

$$\Pr[X = k] = pq^{k-1}, \qquad q = 1 - p,$$
$$\mathrm{E}[X] = \sum_{k=1}^{\infty} kpq^{k-1} = \frac{1}{p}.$$

Theoretical Computer Science Cheat Sheet

Trigonometry





Pythagorean theorem:

$$C^2 = A^2 + B^2.$$

Definitions:

$$\sin a = A/C, \quad \cos a = B/C,$$
$$\csc a = C/A, \quad \sec a = C/B,$$
$$\tan a = \frac{\sin a}{\cos a} = \frac{A}{B}, \quad \cot a = \frac{\cos a}{\sin a} = \frac{B}{A}.$$

Area, radius of inscribed circle:

$$\tfrac{1}{2}AB, \quad \frac{AB}{A+B+C}.$$

Identities:

$$\sin x = \frac{1}{\csc x}, \qquad \cos x = \frac{1}{\sec x},$$
$$\tan x = \frac{1}{\cot x}, \qquad \sin^2 x + \cos^2 x = 1,$$
$$1 + \tan^2 x = \sec^2 x, \qquad 1 + \cot^2 x = \csc^2 x,$$
$$\sin x = \cos\left(\tfrac{\pi}{2} - x\right), \qquad \sin x = \sin(\pi - x),$$
$$\cos x = -\cos(\pi - x), \qquad \tan x = \cot\left(\tfrac{\pi}{2} - x\right),$$
$$\cot x = -\cot(\pi - x), \qquad \csc x = \cot\tfrac{x}{2} - \cot x,$$
$$\sin(x \pm y) = \sin x \cos y \pm \cos x \sin y,$$
$$\cos(x \pm y) = \cos x \cos y \mp \sin x \sin y,$$
$$\tan(x \pm y) = \frac{\tan x \pm \tan y}{1 \mp \tan x \tan y},$$
$$\cot(x \pm y) = \frac{\cot x \cot y \mp 1}{\cot x \pm \cot y},$$
$$\sin 2x = 2\sin x \cos x, \qquad \sin 2x = \frac{2\tan x}{1 + \tan^2 x},$$
$$\cos 2x = \cos^2 x - \sin^2 x, \qquad \cos 2x = 2\cos^2 x - 1,$$
$$\cos 2x = 1 - 2\sin^2 x, \qquad \cos 2x = \frac{1 - \tan^2 x}{1 + \tan^2 x},$$
$$\tan 2x = \frac{2\tan x}{1 - \tan^2 x}, \qquad \cot 2x = \frac{\cot^2 x - 1}{2\cot x},$$
$$\sin(x+y)\sin(x-y) = \sin^2 x - \sin^2 y,$$
$$\cos(x+y)\cos(x-y) = \cos^2 x - \sin^2 y.$$

Euler's equation:

$$e^{ix} = \cos x + i\sin x, \qquad e^{i\pi} = -1.$$

Matrices

Multiplication:

$$C = A \cdot B, \quad c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}.$$

Determinants: $\det A \neq 0$ iff A is non-singular.

$$\det A \cdot B = \det A \cdot \det B,$$
$$\det A = \sum_{\pi} \prod_{i=1}^{n} \operatorname{sign}(\pi) a_{i,\pi(i)}.$$

2×2 and 3×3 determinant:

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc,$$
$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = g\begin{vmatrix} b & c \\ e & f \end{vmatrix} - h\begin{vmatrix} a & c \\ d & f \end{vmatrix} + i\begin{vmatrix} a & b \\ d & e \end{vmatrix}$$
$$= \begin{array}{c} aei + bfg + cdh \\ -ceg - fha - ibd. \end{array}$$

Permanents:

$$\operatorname{perm} A = \sum_{\pi} \prod_{i=1}^{n} a_{i,\pi(i)}.$$

Hyperbolic Functions

Definitions:

$$\sinh x = \frac{e^x - e^{-x}}{2}, \qquad \cosh x = \frac{e^x + e^{-x}}{2},$$
$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \qquad \operatorname{csch} x = \frac{1}{\sinh x},$$
$$\operatorname{sech} x = \frac{1}{\cosh x}, \qquad \coth x = \frac{1}{\tanh x}.$$

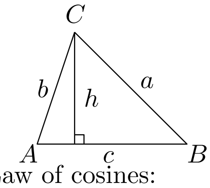
Identities:

$$\cosh^2 x - \sinh^2 x = 1, \qquad \tanh^2 x + \operatorname{sech}^2 x = 1,$$
$$\coth^2 x - \operatorname{csch}^2 x = 1, \qquad \sinh(-x) = -\sinh x,$$
$$\cosh(-x) = \cosh x, \qquad \tanh(-x) = -\tanh x,$$
$$\sinh(x+y) = \sinh x \cosh y + \cosh x \sinh y,$$
$$\cosh(x+y) = \cosh x \cosh y + \sinh x \sinh y,$$
$$\sinh 2x = 2\sinh x \cosh x,$$
$$\cosh 2x = \cosh^2 x + \sinh^2 x,$$
$$\cosh x + \sinh x = e^x, \qquad \cosh x - \sinh x = e^{-x},$$
$$(\cosh x + \sinh x)^n = \cosh nx + \sinh nx, \quad n \in \mathbb{Z},$$
$$2\sinh^2\tfrac{x}{2} = \cosh x - 1, \qquad 2\cosh^2\tfrac{x}{2} = \cosh x + 1.$$

θ	$\sin\theta$	$\cos\theta$	$\tan\theta$
0	0	1	0
$\frac{\pi}{6}$	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{3}}{3}$
$\frac{\pi}{4}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$	1
$\frac{\pi}{3}$	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$	$\sqrt{3}$
$\frac{\pi}{2}$	1	0	∞

…in mathematics you don't understand things, you just get used to them.
– J. von Neumann

More Trig.



Law of cosines:

$$c^2 = a^2 + b^2 - 2ab\cos C.$$

Area:

$$A = \tfrac{1}{2}hc,$$
$$= \tfrac{1}{2}ab\sin C,$$
$$= \frac{c^2 \sin A \sin B}{2\sin C}.$$

Heron's formula:

$$A = \sqrt{s \cdot s_a \cdot s_b \cdot s_c},$$
$$s = \tfrac{1}{2}(a+b+c),$$
$$s_a = s - a,$$
$$s_b = s - b,$$
$$s_c = s - c.$$

More identities:

$$\sin\tfrac{x}{2} = \sqrt{\frac{1 - \cos x}{2}},$$
$$\cos\tfrac{x}{2} = \sqrt{\frac{1 + \cos x}{2}},$$
$$\tan\tfrac{x}{2} = \sqrt{\frac{1 - \cos x}{1 + \cos x}},$$
$$= \frac{1 - \cos x}{\sin x},$$
$$= \frac{\sin x}{1 + \cos x},$$
$$\cot\tfrac{x}{2} = \sqrt{\frac{1 + \cos x}{1 - \cos x}},$$
$$= \frac{1 + \cos x}{\sin x},$$
$$= \frac{\sin x}{1 - \cos x},$$
$$\sin x = \frac{e^{ix} - e^{-ix}}{2i},$$
$$\cos x = \frac{e^{ix} + e^{-ix}}{2},$$
$$\tan x = -i\frac{e^{ix} - e^{-ix}}{e^{ix} + e^{-ix}},$$
$$= -i\frac{e^{2ix} - 1}{e^{2ix} + 1},$$
$$\sin x = \frac{\sinh ix}{i},$$
$$\cos x = \cosh ix,$$
$$\tan x = \frac{\tanh ix}{i}.$$

sseiden@acm.org
http://www.csc.lsu.edu/~seiden

Theoretical Computer Science Cheat Sheet

Number Theory

The Chinese remainder theorem: There exists a number C such that:

$$C \equiv r_1 \bmod m_1$$
$$\vdots \quad \vdots \quad \vdots$$
$$C \equiv r_n \bmod m_n$$

if m_i and m_j are relatively prime for $i \neq j$.

Euler's function: $\phi(x)$ is the number of positive integers less than x relatively prime to x. If $\prod_{i=1}^n p_i^{e_i}$ is the prime factorization of x then

$$\phi(x) = \prod_{i=1}^{n} p_i^{e_i-1}(p_i - 1).$$

Euler's theorem: If a and b are relatively prime then

$$1 \equiv a^{\phi(b)} \bmod b.$$

Fermat's theorem:

$$1 \equiv a^{p-1} \bmod p.$$

The Euclidean algorithm: if $a > b$ are integers then

$$\gcd(a, b) = \gcd(a \bmod b, b).$$

If $\prod_{i=1}^n p_i^{e_i}$ is the prime factorization of x then

$$S(x) = \sum_{d|x} d = \prod_{i=1}^{n} \frac{p_i^{e_i+1} - 1}{p_i - 1}.$$

Perfect Numbers: x is an even perfect number iff $x = 2^{n-1}(2^n - 1)$ and $2^n - 1$ is prime.
Wilson's theorem: n is a prime iff

$$(n-1)! \equiv -1 \bmod n.$$

Möbius inversion:

$$\mu(i) = \begin{cases} 1 & \text{if } i = 1. \\ 0 & \text{if } i \text{ is not square-free.} \\ (-1)^r & \text{if } i \text{ is the product of} \\ & r \text{ distinct primes.} \end{cases}$$

If

$$G(a) = \sum_{d|a} F(d),$$

then

$$F(a) = \sum_{d|a} \mu(d) G\Big(\frac{a}{d}\Big).$$

Prime numbers:

$$p_n = n \ln n + n \ln\ln n - n + n\frac{\ln\ln n}{\ln n} + O\Big(\frac{n}{\ln n}\Big),$$

$$\pi(n) = \frac{n}{\ln n} + \frac{n}{(\ln n)^2} + \frac{2!n}{(\ln n)^3} + O\Big(\frac{n}{(\ln n)^4}\Big).$$

Graph Theory

Definitions:

Loop	An edge connecting a vertex to itself.
Directed	Each edge has a direction.
Simple	Graph with no loops or multi-edges.
Walk	A sequence $v_0 e_1 v_1 \ldots e_\ell v_\ell$.
Trail	A walk with distinct edges.
Path	A trail with distinct vertices.
Connected	A graph where there exists a path between any two vertices.
Component	A maximal connected subgraph.
Tree	A connected acyclic graph.
Free tree	A tree with no root.
DAG	Directed acyclic graph.
Eulerian	Graph with a trail visiting each edge exactly once.
Hamiltonian	Graph with a cycle visiting each vertex exactly once.
Cut	A set of edges whose removal increases the number of components.
Cut-set	A minimal cut.
Cut edge	A size 1 cut.
k-Connected	A graph connected with the removal of any $k-1$ vertices.
k-Tough	$\forall S \subseteq V, S \neq \emptyset$ we have $k \cdot c(G - S) \leq \lvert S \rvert$.
k-Regular	A graph where all vertices have degree k.
k-Factor	A k-regular spanning subgraph.
Matching	A set of edges, no two of which are adjacent.
Clique	A set of vertices, all of which are adjacent.
Ind. set	A set of vertices, none of which are adjacent.
Vertex cover	A set of vertices which cover all edges.
Planar graph	A graph which can be embeded in the plane.
Plane graph	An embedding of a planar graph.

$$\sum_{v \in V} \deg(v) = 2m.$$

If G is planar then $n - m + f = 2$, so

$$f \leq 2n - 4, \quad m \leq 3n - 6.$$

Any planar graph has a vertex with degree ≤ 5.

Notation:

$E(G)$	Edge set
$V(G)$	Vertex set
$c(G)$	Number of components
$G[S]$	Induced subgraph
$\deg(v)$	Degree of v
$\Delta(G)$	Maximum degree
$\delta(G)$	Minimum degree
$\chi(G)$	Chromatic number
$\chi_E(G)$	Edge chromatic number
G^c	Complement graph
K_n	Complete graph
K_{n_1,n_2}	Complete bipartite graph
$\mathrm{r}(k,\ell)$	Ramsey number

Geometry

Projective coordinates: triples (x, y, z), not all x, y and z zero.

$$(x, y, z) = (cx, cy, cz) \quad \forall c \neq 0.$$

Cartesian	Projective
(x, y)	$(x, y, 1)$
$y = mx + b$	$(m, -1, b)$
$x = c$	$(1, 0, -c)$

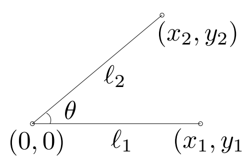
Distance formula, L_p and L_∞ metric:

$$\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2},$$
$$\big[|x_1 - x_0|^p + |y_1 - y_0|^p\big]^{1/p},$$
$$\lim_{p\to\infty} \big[|x_1 - x_0|^p + |y_1 - y_0|^p\big]^{1/p}.$$

Area of triangle (x_0, y_0), (x_1, y_1) and (x_2, y_2):

$$\tfrac{1}{2} \operatorname{abs} \begin{vmatrix} x_1 - x_0 & y_1 - y_0 \\ x_2 - x_0 & y_2 - y_0 \end{vmatrix}.$$

Angle formed by three points:



$$\cos\theta = \frac{(x_1, y_1) \cdot (x_2, y_2)}{\ell_1 \ell_2}.$$

Line through two points (x_0, y_0) and (x_1, y_1):

$$\begin{vmatrix} x & y & 1 \\ x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \end{vmatrix} = 0.$$

Area of circle, volume of sphere:

$$A = \pi r^2, \qquad V = \tfrac{4}{3}\pi r^3.$$

If I have seen farther than others, it is because I have stood on the shoulders of giants.
– Issac Newton

Theoretical Computer Science Cheat Sheet

π

Wallis' identity:
$$\pi = 2 \cdot \frac{2 \cdot 2 \cdot 4 \cdot 4 \cdot 6 \cdot 6 \cdots}{1 \cdot 3 \cdot 3 \cdot 5 \cdot 5 \cdot 7 \cdots}$$

Brouncker's continued fraction expansion:
$$\tfrac{\pi}{4} = 1 + \cfrac{1^2}{2 + \cfrac{3^2}{2 + \cfrac{5^2}{2 + \cfrac{7^2}{2 + \cdots}}}}$$

Gregrory's series:
$$\tfrac{\pi}{4} = 1 - \tfrac{1}{3} + \tfrac{1}{5} - \tfrac{1}{7} + \tfrac{1}{9} - \cdots$$

Newton's series:
$$\tfrac{\pi}{6} = \frac{1}{2} + \frac{1}{2 \cdot 3 \cdot 2^3} + \frac{1 \cdot 3}{2 \cdot 4 \cdot 5 \cdot 2^5} + \cdots$$

Sharp's series:
$$\tfrac{\pi}{6} = \frac{1}{\sqrt{3}}\left(1 - \frac{1}{3^1 \cdot 3} + \frac{1}{3^2 \cdot 5} - \frac{1}{3^3 \cdot 7} + \cdots\right)$$

Euler's series:
$$\tfrac{\pi^2}{6} = \tfrac{1}{1^2} + \tfrac{1}{2^2} + \tfrac{1}{3^2} + \tfrac{1}{4^2} + \tfrac{1}{5^2} + \cdots$$
$$\tfrac{\pi^2}{8} = \tfrac{1}{1^2} + \tfrac{1}{3^2} + \tfrac{1}{5^2} + \tfrac{1}{7^2} + \tfrac{1}{9^2} + \cdots$$
$$\tfrac{\pi^2}{12} = \tfrac{1}{1^2} - \tfrac{1}{2^2} + \tfrac{1}{3^2} - \tfrac{1}{4^2} + \tfrac{1}{5^2} - \cdots$$

Partial Fractions

Let $N(x)$ and $D(x)$ be polynomial functions of x. We can break down $N(x)/D(x)$ using partial fraction expansion. First, if the degree of N is greater than or equal to the degree of D, divide N by D, obtaining
$$\frac{N(x)}{D(x)} = Q(x) + \frac{N'(x)}{D(x)},$$
where the degree of N' is less than that of D. Second, factor $D(x)$. Use the following rules: For a non-repeated factor:
$$\frac{N(x)}{(x-a)D(x)} = \frac{A}{x-a} + \frac{N'(x)}{D(x)},$$
where
$$A = \left[\frac{N(x)}{D(x)}\right]_{x=a}.$$
For a repeated factor:
$$\frac{N(x)}{(x-a)^m D(x)} = \sum_{k=0}^{m-1} \frac{A_k}{(x-a)^{m-k}} + \frac{N'(x)}{D(x)},$$
where
$$A_k = \frac{1}{k!}\left[\frac{d^k}{dx^k}\left(\frac{N(x)}{D(x)}\right)\right]_{x=a}.$$

The reasonable man adapts himself to the world; the unreasonable persists in trying to adapt the world to himself. Therefore all progress depends on the unreasonable.
– George Bernard Shaw

Calculus

Derivatives:

1. $\dfrac{d(cu)}{dx} = c\dfrac{du}{dx}$, **2.** $\dfrac{d(u+v)}{dx} = \dfrac{du}{dx} + \dfrac{dv}{dx}$, **3.** $\dfrac{d(uv)}{dx} = u\dfrac{dv}{dx} + v\dfrac{du}{dx}$,

4. $\dfrac{d(u^n)}{dx} = nu^{n-1}\dfrac{du}{dx}$, **5.** $\dfrac{d(u/v)}{dx} = \dfrac{v\left(\frac{du}{dx}\right) - u\left(\frac{dv}{dx}\right)}{v^2}$, **6.** $\dfrac{d(e^{cu})}{dx} = ce^{cu}\dfrac{du}{dx}$,

7. $\dfrac{d(c^u)}{dx} = (\ln c)c^u\dfrac{du}{dx}$, **8.** $\dfrac{d(\ln u)}{dx} = \dfrac{1}{u}\dfrac{du}{dx}$,

9. $\dfrac{d(\sin u)}{dx} = \cos u\dfrac{du}{dx}$, **10.** $\dfrac{d(\cos u)}{dx} = -\sin u\dfrac{du}{dx}$,

11. $\dfrac{d(\tan u)}{dx} = \sec^2 u\dfrac{du}{dx}$, **12.** $\dfrac{d(\cot u)}{dx} = \csc^2 u\dfrac{du}{dx}$,

13. $\dfrac{d(\sec u)}{dx} = \tan u \, \sec u\dfrac{du}{dx}$, **14.** $\dfrac{d(\csc u)}{dx} = -\cot u \, \csc u\dfrac{du}{dx}$,

15. $\dfrac{d(\arcsin u)}{dx} = \dfrac{1}{\sqrt{1-u^2}}\dfrac{du}{dx}$, **16.** $\dfrac{d(\arccos u)}{dx} = \dfrac{-1}{\sqrt{1-u^2}}\dfrac{du}{dx}$,

17. $\dfrac{d(\arctan u)}{dx} = \dfrac{1}{1+u^2}\dfrac{du}{dx}$, **18.** $\dfrac{d(\operatorname{arccot} u)}{dx} = \dfrac{-1}{1+u^2}\dfrac{du}{dx}$,

19. $\dfrac{d(\operatorname{arcsec} u)}{dx} = \dfrac{1}{u\sqrt{1-u^2}}\dfrac{du}{dx}$, **20.** $\dfrac{d(\operatorname{arccsc} u)}{dx} = \dfrac{-1}{u\sqrt{1-u^2}}\dfrac{du}{dx}$,

21. $\dfrac{d(\sinh u)}{dx} = \cosh u\dfrac{du}{dx}$, **22.** $\dfrac{d(\cosh u)}{dx} = \sinh u\dfrac{du}{dx}$,

23. $\dfrac{d(\tanh u)}{dx} = \operatorname{sech}^2 u\dfrac{du}{dx}$, **24.** $\dfrac{d(\coth u)}{dx} = -\operatorname{csch}^2 u\dfrac{du}{dx}$,

25. $\dfrac{d(\operatorname{sech} u)}{dx} = -\operatorname{sech} u \, \tanh u\dfrac{du}{dx}$, **26.** $\dfrac{d(\operatorname{csch} u)}{dx} = -\operatorname{csch} u \, \coth u\dfrac{du}{dx}$,

27. $\dfrac{d(\operatorname{arcsinh} u)}{dx} = \dfrac{1}{\sqrt{1+u^2}}\dfrac{du}{dx}$, **28.** $\dfrac{d(\operatorname{arccosh} u)}{dx} = \dfrac{1}{\sqrt{u^2-1}}\dfrac{du}{dx}$,

29. $\dfrac{d(\operatorname{arctanh} u)}{dx} = \dfrac{1}{1-u^2}\dfrac{du}{dx}$, **30.** $\dfrac{d(\operatorname{arccoth} u)}{dx} = \dfrac{1}{u^2-1}\dfrac{du}{dx}$,

31. $\dfrac{d(\operatorname{arcsech} u)}{dx} = \dfrac{-1}{u\sqrt{1-u^2}}\dfrac{du}{dx}$, **32.** $\dfrac{d(\operatorname{arccsch} u)}{dx} = \dfrac{-1}{|u|\sqrt{1+u^2}}\dfrac{du}{dx}$.

Integrals:

1. $\displaystyle\int cu\,dx = c\int u\,dx$, **2.** $\displaystyle\int (u+v)\,dx = \int u\,dx + \int v\,dx$,

3. $\displaystyle\int x^n\,dx = \frac{1}{n+1}x^{n+1}, \quad n \neq -1$, **4.** $\displaystyle\int \frac{1}{x}dx = \ln x$, **5.** $\displaystyle\int e^x\,dx = e^x$,

6. $\displaystyle\int \frac{dx}{1+x^2} = \arctan x$, **7.** $\displaystyle\int u\frac{dv}{dx}dx = uv - \int v\frac{du}{dx}dx$,

8. $\displaystyle\int \sin x\,dx = -\cos x$, **9.** $\displaystyle\int \cos x\,dx = \sin x$,

10. $\displaystyle\int \tan x\,dx = -\ln|\cos x|$, **11.** $\displaystyle\int \cot x\,dx = \ln|\cos x|$,

12. $\displaystyle\int \sec x\,dx = \ln|\sec x + \tan x|$, **13.** $\displaystyle\int \csc x\,dx = \ln|\csc x + \cot x|$,

14. $\displaystyle\int \arcsin \tfrac{x}{a}dx = \arcsin \tfrac{x}{a} + \sqrt{a^2 - x^2}, \quad a > 0$, Calculus Cont.

15. $\int \arccos \frac{x}{a} dx = \arccos \frac{x}{a} - \sqrt{a^2 - x^2}, \quad a > 0,$

16. $\int \arctan \frac{x}{a} dx = x \arctan \frac{x}{a} - \frac{a}{2} \ln(a^2 + x^2), \quad a > 0,$

17. $\int \sin^2(ax) dx = \frac{1}{2a}\big(ax - \sin(ax)\cos(ax)\big),$

18. $\int \cos^2(ax) dx = \frac{1}{2a}\big(ax + \sin(ax)\cos(ax)\big),$

19. $\int \sec^2 x\, dx = \tan x,$

20. $\int \csc^2 x\, dx = -\cot x,$

21. $\int \sin^n x\, dx = -\frac{\sin^{n-1} x \cos x}{n} + \frac{n-1}{n} \int \sin^{n-2} x\, dx,$

22. $\int \cos^n x\, dx = \frac{\cos^{n-1} x \sin x}{n} + \frac{n-1}{n} \int \cos^{n-2} x\, dx,$

23. $\int \tan^n x\, dx = \frac{\tan^{n-1} x}{n-1} - \int \tan^{n-2} x\, dx, \quad n \neq 1,$

24. $\int \cot^n x\, dx = -\frac{\cot^{n-1} x}{n-1} - \int \cot^{n-2} x\, dx, \quad n \neq 1,$

25. $\int \sec^n x\, dx = \frac{\tan x \sec^{n-1} x}{n-1} + \frac{n-2}{n-1} \int \sec^{n-2} x\, dx, \quad n \neq 1,$

26. $\int \csc^n x\, dx = -\frac{\cot x \csc^{n-1} x}{n-1} + \frac{n-2}{n-1} \int \csc^{n-2} x\, dx, \quad n \neq 1,$

27. $\int \sinh x\, dx = \cosh x,$

28. $\int \cosh x\, dx = \sinh x,$

29. $\int \tanh x\, dx = \ln|\cosh x|,$

30. $\int \coth x\, dx = \ln|\sinh x|,$

31. $\int \operatorname{sech} x\, dx = \arctan \sinh x,$

32. $\int \operatorname{csch} x\, dx = \ln\left|\tanh \frac{x}{2}\right|,$

33. $\int \sinh^2 x\, dx = \frac{1}{4}\sinh(2x) - \frac{1}{2}x,$

34. $\int \cosh^2 x\, dx = \frac{1}{4}\sinh(2x) + \frac{1}{2}x,$

35. $\int \operatorname{sech}^2 x\, dx = \tanh x,$

36. $\int \operatorname{arcsinh} \frac{x}{a} dx = x \operatorname{arcsinh} \frac{x}{a} - \sqrt{x^2 + a^2}, \quad a > 0,$

37. $\int \operatorname{arctanh} \frac{x}{a} dx = x \operatorname{arctanh} \frac{x}{a} + \frac{a}{2} \ln|a^2 - x^2|,$

38. $\int \operatorname{arccosh} \frac{x}{a} dx = \begin{cases} x \operatorname{arccosh} \dfrac{x}{a} - \sqrt{x^2 + a^2}, & \text{if } \operatorname{arccosh} \frac{x}{a} > 0 \text{ and } a > 0, \\ x \operatorname{arccosh} \dfrac{x}{a} + \sqrt{x^2 + a^2}, & \text{if } \operatorname{arccosh} \frac{x}{a} < 0 \text{ and } a > 0, \end{cases}$

39. $\int \frac{dx}{\sqrt{a^2 + x^2}} = \ln\left(x + \sqrt{a^2 + x^2}\right), \quad a > 0,$

40. $\int \frac{dx}{a^2 + x^2} = \frac{1}{a} \arctan \frac{x}{a}, \quad a > 0,$

41. $\int \sqrt{a^2 - x^2}\, dx = \frac{x}{2}\sqrt{a^2 - x^2} + \frac{a^2}{2} \arcsin \frac{x}{a}, \quad a > 0,$

42. $\int (a^2 - x^2)^{3/2} dx = \frac{x}{8}(5a^2 - 2x^2)\sqrt{a^2 - x^2} + \frac{3a^4}{8} \arcsin \frac{x}{a}, \quad a > 0,$

43. $\int \frac{dx}{\sqrt{a^2 - x^2}} = \arcsin \frac{x}{a}, \quad a > 0,$

44. $\int \frac{dx}{a^2 - x^2} = \frac{1}{2a} \ln\left|\frac{a + x}{a - x}\right|,$

45. $\int \frac{dx}{(a^2 - x^2)^{3/2}} = \frac{x}{a^2\sqrt{a^2 - x^2}},$

46. $\int \sqrt{a^2 \pm x^2}\, dx = \frac{x}{2}\sqrt{a^2 \pm x^2} \pm \frac{a^2}{2} \ln\left|x + \sqrt{a^2 \pm x^2}\right|,$

47. $\int \frac{dx}{\sqrt{x^2 - a^2}} = \ln\left|x + \sqrt{x^2 - a^2}\right|, \quad a > 0,$

48. $\int \frac{dx}{ax^2 + bx} = \frac{1}{a} \ln\left|\frac{x}{a + bx}\right|,$

49. $\int x\sqrt{a + bx}\, dx = \frac{2(3bx - 2a)(a + bx)^{3/2}}{15b^2},$

50. $\int \frac{\sqrt{a + bx}}{x} dx = 2\sqrt{a + bx} + a \int \frac{1}{x\sqrt{a + bx}} dx,$

51. $\int \frac{x}{\sqrt{a + bx}} dx = \frac{1}{\sqrt{2}} \ln\left|\frac{\sqrt{a + bx} - \sqrt{a}}{\sqrt{a + bx} + \sqrt{a}}\right|, \quad a > 0,$

52. $\int \frac{\sqrt{a^2 - x^2}}{x} dx = \sqrt{a^2 - x^2} - a \ln\left|\frac{a + \sqrt{a^2 - x^2}}{x}\right|,$

53. $\int x\sqrt{a^2 - x^2}\, dx = -\frac{1}{3}(a^2 - x^2)^{3/2},$

54. $\int x^2\sqrt{a^2 - x^2}\, dx = \frac{x}{8}(2x^2 - a^2)\sqrt{a^2 - x^2} + \frac{a^4}{8} \arcsin \frac{x}{a}, \quad a > 0,$

55. $\int \frac{dx}{\sqrt{a^2 - x^2}} = -\frac{1}{a} \ln\left|\frac{a + \sqrt{a^2 - x^2}}{x}\right|,$

56. $\int \frac{x\, dx}{\sqrt{a^2 - x^2}} = -\sqrt{a^2 - x^2},$

57. $\int \frac{x^2\, dx}{\sqrt{a^2 - x^2}} = -\frac{x}{2}\sqrt{a^2 - x^2} + \frac{a^2}{2} \arcsin \frac{x}{a}, \quad a > 0,$

58. $\int \frac{\sqrt{a^2 + x^2}}{x} dx = \sqrt{a^2 + x^2} - a \ln\left|\frac{a + \sqrt{a^2 + x^2}}{x}\right|,$

59. $\int \frac{\sqrt{x^2 - a^2}}{x} dx = \sqrt{x^2 - a^2} - a \arccos \frac{a}{|x|}, \quad a > 0,$

60. $\int x\sqrt{x^2 \pm a^2}\, dx = \frac{1}{3}(x^2 \pm a^2)^{3/2},$

61. $\int \frac{dx}{x\sqrt{x^2 + a^2}} = \frac{1}{a} \ln\left|\frac{x}{a + \sqrt{a^2 + x^2}}\right|,$

Theoretical Computer Science Cheat Sheet

Calculus Cont.

62. $\displaystyle\int \frac{dx}{x\sqrt{x^2-a^2}} = \tfrac{1}{a}\arccos\tfrac{a}{|x|}, \quad a>0,$

63. $\displaystyle\int \frac{dx}{x^2\sqrt{x^2\pm a^2}} = \mp\frac{\sqrt{x^2\pm a^2}}{a^2x},$

64. $\displaystyle\int \frac{x\,dx}{\sqrt{x^2\pm a^2}} = \sqrt{x^2\pm a^2},$

65. $\displaystyle\int \frac{\sqrt{x^2\pm a^2}}{x^4}\,dx = \mp\frac{(x^2+a^2)^{3/2}}{3a^2x^3},$

66. $$\int \frac{dx}{ax^2+bx+c} = \begin{cases} \dfrac{1}{\sqrt{b^2-4ac}}\ln\left|\dfrac{2ax+b-\sqrt{b^2-4ac}}{2ax+b+\sqrt{b^2-4ac}}\right|, & \text{if } b^2>4ac,\\[2ex] \dfrac{2}{\sqrt{4ac-b^2}}\arctan\dfrac{2ax+b}{\sqrt{4ac-b^2}}, & \text{if } b^2<4ac,\end{cases}$$

67. $$\int \frac{dx}{\sqrt{ax^2+bx+c}} = \begin{cases} \dfrac{1}{\sqrt{a}}\ln\left|2ax+b+2\sqrt{a}\sqrt{ax^2+bx+c}\right|, & \text{if } a>0,\\[2ex] \dfrac{1}{\sqrt{-a}}\arcsin\dfrac{-2ax-b}{\sqrt{b^2-4ac}}, & \text{if } a<0,\end{cases}$$

68. $$\int \sqrt{ax^2+bx+c}\,dx = \frac{2ax+b}{4a}\sqrt{ax^2+bx+c} + \frac{4ax-b^2}{8a}\int\frac{dx}{\sqrt{ax^2+bx+c}},$$

69. $$\int \frac{x\,dx}{\sqrt{ax^2+bx+c}} = \frac{\sqrt{ax^2+bx+c}}{a} - \frac{b}{2a}\int\frac{dx}{\sqrt{ax^2+bx+c}},$$

70. $$\int \frac{dx}{x\sqrt{ax^2+bx+c}} = \begin{cases} \dfrac{-1}{\sqrt{c}}\ln\left|\dfrac{2\sqrt{c}\sqrt{ax^2+bx+c}+bx+2c}{x}\right|, & \text{if } c>0,\\[2ex] \dfrac{1}{\sqrt{-c}}\arcsin\dfrac{bx+2c}{|x|\sqrt{b^2-4ac}}, & \text{if } c<0,\end{cases}$$

71. $\displaystyle\int x^3\sqrt{x^2+a^2}\,dx = (\tfrac13x^2-\tfrac{2}{15}a^2)(x^2+a^2)^{3/2},$

72. $\displaystyle\int x^n\sin(ax)\,dx = -\tfrac1a x^n\cos(ax) + \tfrac{n}{a}\int x^{n-1}\cos(ax)\,dx,$

73. $\displaystyle\int x^n\cos(ax)\,dx = \tfrac1a x^n\sin(ax) - \tfrac{n}{a}\int x^{n-1}\sin(ax)\,dx,$

74. $\displaystyle\int x^ne^{ax}\,dx = \frac{x^ne^{ax}}{a} - \tfrac{n}{a}\int x^{n-1}e^{ax}\,dx,$

75. $\displaystyle\int x^n\ln(ax)\,dx = x^{n+1}\left(\frac{\ln(ax)}{n+1} - \frac{1}{(n+1)^2}\right),$

76. $\displaystyle\int x^n(\ln ax)^m\,dx = \frac{x^{n+1}}{n+1}(\ln ax)^m - \frac{m}{n+1}\int x^n(\ln ax)^{m-1}\,dx.$

$$\begin{aligned}
x^1 &= x^{\underline{1}} &&= x^{\overline{1}}\\
x^2 &= x^{\underline{2}}+x^{\underline{1}} &&= x^{\overline{2}}-x^{\overline{1}}\\
x^3 &= x^{\underline{3}}+3x^{\underline{2}}+x^{\underline{1}} &&= x^{\overline{3}}-3x^{\overline{2}}+x^{\overline{1}}\\
x^4 &= x^{\underline{4}}+6x^{\underline{3}}+7x^{\underline{2}}+x^{\underline{1}} &&= x^{\overline{4}}-6x^{\overline{3}}+7x^{\overline{2}}-x^{\overline{1}}\\
x^5 &= x^{\underline{5}}+15x^{\underline{4}}+25x^{\underline{3}}+10x^{\underline{2}}+x^{\underline{1}} &&= x^{\underline{5}}-15x^{\overline{4}}+25x^{\overline{3}}-10x^{\overline{2}}+x^{\overline{1}}
\end{aligned}$$

$$\begin{aligned}
x^{\overline{1}} &= x^1 & x^{\underline{1}} &= x^1\\
x^{\overline{2}} &= x^2+x^1 & x^{\underline{2}} &= x^2-x^1\\
x^{\overline{3}} &= x^3+3x^2+2x^1 & x^{\underline{3}} &= x^3-3x^2+2x^1\\
x^{\overline{4}} &= x^4+6x^3+11x^2+6x^1 & x^{\underline{4}} &= x^4-6x^3+11x^2-6x^1\\
x^{\overline{5}} &= x^5+10x^4+35x^3+50x^2+24x^1 & x^{\underline{5}} &= x^5-10x^4+35x^3-50x^2+24x^1
\end{aligned}$$

Finite Calculus

Difference, shift operators:

$$\Delta f(x) = f(x+1) - f(x),$$
$$\mathrm{E}\, f(x) = f(x+1).$$

Fundamental Theorem:

$$f(x) = \Delta F(x) \Leftrightarrow \sum f(x)\delta x = F(x) + C.$$
$$\sum_a^b f(x)\delta x = \sum_{i=a}^{b-1} f(i).$$

Differences:

$\Delta(cu) = c\Delta u, \qquad \Delta(u+v) = \Delta u + \Delta v,$

$\Delta(uv) = u\Delta v + \mathrm{E}\,v\Delta u,$

$\Delta(x^{\underline{n}}) = nx^{\underline{n-1}},$

$\Delta(H_x) = x^{\underline{-1}}, \qquad \Delta(2^x) = 2^x,$

$\Delta(c^x) = (c-1)c^x, \qquad \Delta\binom{x}{m} = \binom{x}{m-1}.$

Sums:

$\sum cu\,\delta x = c\sum u\,\delta x,$

$\sum(u+v)\,\delta x = \sum u\,\delta x + \sum v\,\delta x,$

$\sum u\Delta v\,\delta x = uv - \sum \mathrm{E}\,v\Delta u\,\delta x,$

$\sum x^{\underline{n}}\,\delta x = \frac{x^{\underline{n+1}}}{m+1}, \qquad \sum x^{\underline{-1}}\,\delta x = H_x,$

$\sum c^x\,\delta x = \frac{c^x}{c-1}, \qquad \sum\binom{x}{m}\,\delta x = \binom{x}{m+1}.$

Falling Factorial Powers:

$x^{\underline{n}} = x(x-1)\cdots(x-n+1), \quad n>0,$

$x^{\underline{0}} = 1,$

$x^{\underline{n}} = \dfrac{1}{(x+1)\cdots(x+|n|)}, \quad n<0,$

$x^{\underline{n+m}} = x^{\underline{m}}(x-m)^{\underline{n}}.$

Rising Factorial Powers:

$x^{\overline{n}} = x(x+1)\cdots(x+n-1), \quad n>0,$

$x^{\overline{0}} = 1,$

$x^{\overline{n}} = \dfrac{1}{(x-1)\cdots(x-|n|)}, \quad n<0,$

$x^{\overline{n+m}} = x^{\overline{m}}(x+m)^{\overline{n}}.$

Conversion:

$$\begin{aligned}
x^{\underline{n}} &= (-1)^n(-x)^{\overline{n}} = (x-n+1)^{\overline{n}}\\
&= 1/(x+1)^{\overline{-n}},\\
x^{\overline{n}} &= (-1)^n(-x)^{\underline{n}} = (x+n-1)^{\underline{n}}\\
&= 1/(x-1)^{\underline{-n}},
\end{aligned}$$

$$x^n = \sum_{k=1}^n \left\{ n \atop k \right\} x^{\underline{k}} = \sum_{k=1}^n \left\{ n \atop k \right\}(-1)^{n-k}x^{\overline{k}},$$

$$x^{\underline{n}} = \sum_{k=1}^n \left[n \atop k \right](-1)^{n-k}x^k,$$

$$x^{\overline{n}} = \sum_{k=1}^n \left[n \atop k \right] x^k.$$

Theoretical Computer Science Cheat Sheet

Series

Taylor's series:

$$f(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2}f''(a) + \cdots = \sum_{i=0}^{\infty} \frac{(x-a)^i}{i!} f^{(i)}(a).$$

Expansions:

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + x^4 + \cdots = \sum_{i=0}^{\infty} x^i,$$

$$\frac{1}{1-cx} = 1 + cx + c^2x^2 + c^3x^3 + \cdots = \sum_{i=0}^{\infty} c^i x^i,$$

$$\frac{1}{1-x^n} = 1 + x^n + x^{2n} + x^{3n} + \cdots = \sum_{i=0}^{\infty} x^{ni},$$

$$\frac{x}{(1-x)^2} = x + 2x^2 + 3x^3 + 4x^4 + \cdots = \sum_{i=0}^{\infty} ix^i,$$

$$\sum_{k=0}^{n} \left\{ {n \atop k} \right\} \frac{k!z^k}{(1-z)^{k+1}} = x + 2^nx^2 + 3^nx^3 + 4^nx^4 + \cdots = \sum_{i=0}^{\infty} i^n x^i,$$

$$e^x = 1 + x + \tfrac{1}{2}x^2 + \tfrac{1}{6}x^3 + \cdots = \sum_{i=0}^{\infty} \frac{x^i}{i!},$$

$$\ln(1+x) = x - \tfrac{1}{2}x^2 + \tfrac{1}{3}x^3 - \tfrac{1}{4}x^4 - \cdots = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^i}{i},$$

$$\ln \frac{1}{1-x} = x + \tfrac{1}{2}x^2 + \tfrac{1}{3}x^3 + \tfrac{1}{4}x^4 + \cdots = \sum_{i=1}^{\infty} \frac{x^i}{i},$$

$$\sin x = x - \tfrac{1}{3!}x^3 + \tfrac{1}{5!}x^5 - \tfrac{1}{7!}x^7 + \cdots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)!},$$

$$\cos x = 1 - \tfrac{1}{2!}x^2 + \tfrac{1}{4!}x^4 - \tfrac{1}{6!}x^6 + \cdots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!},$$

$$\tan^{-1} x = x - \tfrac{1}{3}x^3 + \tfrac{1}{5}x^5 - \tfrac{1}{7}x^7 + \cdots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)},$$

$$(1+x)^n = 1 + nx + \tfrac{n(n-1)}{2}x^2 + \cdots = \sum_{i=0}^{\infty} \binom{n}{i} x^i,$$

$$\frac{1}{(1-x)^{n+1}} = 1 + (n+1)x + \tbinom{n+2}{2}x^2 + \cdots = \sum_{i=0}^{\infty} \binom{i+n}{i} x^i,$$

$$\frac{x}{e^x - 1} = 1 - \tfrac{1}{2}x + \tfrac{1}{12}x^2 - \tfrac{1}{720}x^4 + \cdots = \sum_{i=0}^{\infty} \frac{B_i x^i}{i!},$$

$$\frac{1}{2x}(1 - \sqrt{1-4x}) = 1 + x + 2x^2 + 5x^3 + \cdots = \sum_{i=0}^{\infty} \frac{1}{i+1}\binom{2i}{i} x^i,$$

$$\frac{1}{\sqrt{1-4x}} = 1 + 2x + 6x^2 + 20x^3 + \cdots = \sum_{i=0}^{\infty} \binom{2i}{i} x^i,$$

$$\frac{1}{\sqrt{1-4x}} \left(\frac{1-\sqrt{1-4x}}{2x} \right)^n = 1 + (2+n)x + \tbinom{4+n}{2}x^2 + \cdots = \sum_{i=0}^{\infty} \binom{2i+n}{i} x^i,$$

$$\frac{1}{1-x} \ln \frac{1}{1-x} = x + \tfrac{3}{2}x^2 + \tfrac{11}{6}x^3 + \tfrac{25}{12}x^4 + \cdots = \sum_{i=1}^{\infty} H_i x^i,$$

$$\frac{1}{2}\left(\ln \frac{1}{1-x} \right)^2 = \tfrac{1}{2}x^2 + \tfrac{3}{4}x^3 + \tfrac{11}{24}x^4 + \cdots = \sum_{i=2}^{\infty} \frac{H_{i-1}x^i}{i},$$

$$\frac{x}{1-x-x^2} = x + x^2 + 2x^3 + 3x^4 + \cdots = \sum_{i=0}^{\infty} F_i x^i,$$

$$\frac{F_n x}{1 - (F_{n-1} + F_{n+1})x - (-1)^n x^2} = F_n x + F_{2n}x^2 + F_{3n}x^3 + \cdots = \sum_{i=0}^{\infty} F_{ni} x^i.$$

Ordinary power series:

$$A(x) = \sum_{i=0}^{\infty} a_i x^i.$$

Exponential power series:

$$A(x) = \sum_{i=0}^{\infty} a_i \frac{x^i}{i!}.$$

Dirichlet power series:

$$A(x) = \sum_{i=1}^{\infty} \frac{a_i}{i^x}.$$

Binomial theorem:

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k.$$

Difference of like powers:

$$x^n - y^n = (x-y) \sum_{k=0}^{n-1} x^{n-1-k} y^k.$$

For ordinary power series:

$$\alpha A(x) + \beta B(x) = \sum_{i=0}^{\infty} (\alpha a_i + \beta b_i) x^i,$$

$$x^k A(x) = \sum_{i=k}^{\infty} a_{i-k} x^i,$$

$$\frac{A(x) - \sum_{i=0}^{k-1} a_i x^i}{x^k} = \sum_{i=0}^{\infty} a_{i+k} x^i,$$

$$A(cx) = \sum_{i=0}^{\infty} c^i a_i x^i,$$

$$A'(x) = \sum_{i=0}^{\infty} (i+1) a_{i+1} x^i,$$

$$xA'(x) = \sum_{i=1}^{\infty} i a_i x^i,$$

$$\int A(x)\,dx = \sum_{i=1}^{\infty} \frac{a_{i-1}}{i} x^i,$$

$$\frac{A(x) + A(-x)}{2} = \sum_{i=0}^{\infty} a_{2i} x^{2i},$$

$$\frac{A(x) - A(-x)}{2} = \sum_{i=0}^{\infty} a_{2i+1} x^{2i+1}.$$

Summation: If $b_i = \sum_{j=0}^{i} a_i$ then

$$B(x) = \frac{1}{1-x} A(x).$$

Convolution:

$$A(x)B(x) = \sum_{i=0}^{\infty} \left(\sum_{j=0}^{i} a_j b_{i-j} \right) x^i.$$

God made the natural numbers;
all the rest is the work of man.
– Leopold Kronecker

Theoretical Computer Science Cheat Sheet

Series

Expansions:

$$\frac{1}{(1-x)^{n+1}}\ln\frac{1}{1-x} = \sum_{i=0}^{\infty}(H_{n+i}-H_n)\binom{n+i}{i}x^i, \qquad \left(\frac{1}{x}\right)^{\overline{-n}} = \sum_{i=0}^{\infty}\left\{ i \atop n \right\}x^i,$$

$$x^{\overline{n}} = \sum_{i=0}^{\infty}\left[n \atop i \right]x^i, \qquad (e^x-1)^n = \sum_{i=0}^{\infty}\left\{ i \atop n \right\}\frac{n!x^i}{i!},$$

$$\left(\ln\frac{1}{1-x}\right)^n = \sum_{i=0}^{\infty}\left[i \atop n \right]\frac{n!x^i}{i!}, \qquad x\cot x = \sum_{i=0}^{\infty}\frac{(-4)^i B_{2i}x^{2i}}{(2i)!},$$

$$\tan x = \sum_{i=1}^{\infty}(-1)^{i-1}\frac{2^{2i}(2^{2i}-1)B_{2i}x^{2i-1}}{(2i)!}, \qquad \zeta(x) = \sum_{i=1}^{\infty}\frac{1}{i^x},$$

$$\frac{1}{\zeta(x)} = \sum_{i=1}^{\infty}\frac{\mu(i)}{i^x}, \qquad \frac{\zeta(x-1)}{\zeta(x)} = \sum_{i=1}^{\infty}\frac{\phi(i)}{i^x},$$

$$\zeta(x) = \prod_p \frac{1}{1-p^{-x}},$$

$$\zeta^2(x) = \sum_{i=1}^{\infty}\frac{d(i)}{x^i} \quad \text{where } d(n) = \textstyle\sum_{d|n} 1,$$

$$\zeta(x)\zeta(x-1) = \sum_{i=1}^{\infty}\frac{S(i)}{x^i} \quad \text{where } S(n) = \textstyle\sum_{d|n} d,$$

$$\zeta(2n) = \frac{2^{2n-1}|B_{2n}|}{(2n)!}\pi^{2n}, \quad n \in \mathbb{N},$$

$$\frac{x}{\sin x} = \sum_{i=0}^{\infty}(-1)^{i-1}\frac{(4^i-2)B_{2i}x^{2i}}{(2i)!},$$

$$\left(\frac{1-\sqrt{1-4x}}{2x}\right)^n = \sum_{i=0}^{\infty}\frac{n(2i+n-1)!}{i!(n+i)!}x^i,$$

$$e^x\sin x = \sum_{i=1}^{\infty}\frac{2^{i/2}\sin\frac{i\pi}{4}}{i!}x^i,$$

$$\sqrt{\frac{1-\sqrt{1-x}}{x}} = \sum_{i=0}^{\infty}\frac{(4i)!}{16^i\sqrt{2}(2i)!(2i+1)!}x^i,$$

$$\left(\frac{\arcsin x}{x}\right)^2 = \sum_{i=0}^{\infty}\frac{4^i i!^2}{(i+1)(2i+1)!}x^{2i}.$$

Escher's Knot

Stieltjes Integration

If G is continuous in the interval $[a,b]$ and F is nondecreasing then

$$\int_a^b G(x)\,dF(x)$$

exists. If $a \le b \le c$ then

$$\int_a^c G(x)\,dF(x) = \int_a^b G(x)\,dF(x) + \int_b^c G(x)\,dF(x).$$

If the integrals involved exist

$$\int_a^b \big(G(x)+H(x)\big)\,dF(x) = \int_a^b G(x)\,dF(x) + \int_a^b H(x)\,dF(x),$$

$$\int_a^b G(x)\,d\big(F(x)+H(x)\big) = \int_a^b G(x)\,dF(x) + \int_a^b G(x)\,dH(x),$$

$$\int_a^b c\cdot G(x)\,dF(x) = \int_a^b G(x)\,d\big(c\cdot F(x)\big) = c\int_a^b G(x)\,dF(x),$$

$$\int_a^b G(x)\,dF(x) = G(b)F(b) - G(a)F(a) - \int_a^b F(x)\,dG(x).$$

If the integrals involved exist, and F possesses a derivative F' at every point in $[a,b]$ then

$$\int_a^b G(x)\,dF(x) = \int_a^b G(x)F'(x)\,dx.$$

Cramer's Rule

If we have equations:

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n &= b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n &= b_2 \\ \vdots \qquad\quad \vdots \qquad\qquad\quad \vdots \quad& \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,n}x_n &= b_n \end{aligned}$$

Let $A = (a_{i,j})$ and B be the column matrix (b_i). Then there is a unique solution iff $\det A \neq 0$. Let A_i be A with column i replaced by B. Then

$$x_i = \frac{\det A_i}{\det A}.$$

00	47	18	76	29	93	85	34	61	52
86	11	57	28	70	39	94	45	02	63
95	80	22	67	38	71	49	56	13	04
59	96	81	33	07	48	72	60	24	15
73	69	90	82	44	17	58	01	35	26
68	74	09	91	83	55	27	12	46	30
37	08	75	19	92	84	66	23	50	41
14	25	36	40	51	62	03	77	88	99
21	32	43	54	65	06	10	89	97	78
42	53	64	05	16	20	31	98	79	87

Fibonacci Numbers

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$$

Definitions:

$$F_i = F_{i-1} + F_{i-2}, \quad F_0 = F_1 = 1,$$

$$F_{-i} = (-1)^{i-1}F_i,$$

$$F_i = \tfrac{1}{\sqrt{5}}\left(\phi^i - \hat{\phi}^i\right),$$

Cassini's identity: for $i > 0$:

$$F_{i+1}F_{i-1} - F_i^2 = (-1)^i.$$

Additive rule:

$$F_{n+k} = F_kF_{n+1} + F_{k-1}F_n,$$

$$F_{2n} = F_nF_{n+1} + F_{n-1}F_n.$$

Calculation by matrices:

$$\begin{pmatrix} F_{n-2} & F_{n-1} \\ F_{n-1} & F_n \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n.$$

The Fibonacci number system: Every integer n has a unique representation

$$n = F_{k_1} + F_{k_2} + \cdots + F_{k_m},$$

where $k_i \ge k_{i+1} + 2$ for all i, $1 \le i < m$ and $k_m \ge 2$.

Improvement makes strait roads, but the crooked roads without Improvement, are roads of Genius.
– William Blake (The Marriage of Heaven and Hell)