Talisman

September 25, 2018

目录

1 代数

1.1 FFT-gwx

```
#include <complex>

int n, m;
int rev[maxn];  //maxn > 2 ^ k

void fft(Complex *a, int f)
{
    for(int i = 0; i < m; i++)
        if(i < r[i]) swap(a[i], a[r[i]]);
    for(int l = 2; l <= m; l <<= 1)
    {
        int h = l >> 1;
        Complex ur = (Complex){cos(pi / h), f * sin(pi / h)};
        for(int i = 0; i < m; i += l)
        {
            Complex w = (Complex){1, 0};
            for(int k = 0; k < h; k++, w = w * ur)
            {
                Complex x = a[i + k], y = a[i + k + h] * w;
                a[i + k] = x + y; a[i + k + h] = x - y;
            }
        }
    }
    if(f == -1)
        for(int i = 0; i < m; i++)
            a[i] = a[i] / m;
}

void multi(Complex *a, Complex *b)
{
    fft(a, 1); fft(b, 1);
    for(int i = 0; i < m; i++)
        a[i] *= b[i];
    fft(a, -1);
}

void init()
{
    for(m = 1; m <= 2 * n; m <<= 1);
    for(int i = 0, j = 0; i < m; i++)
    {
        rev[i] = j;
        for(int x = m >> 1; (j ^= x) < x; x >>= 1);
    }
}
```

1.2 FFT-wrz

```
int len;
struct comp
{
    double r, i;
    comp operator + (const comp &that) {return (comp){r+that.r, i+that.i};}
    comp operator - (const comp &that) {return (comp){r-that.r, i-that.i};}
    comp operator * (const comp &that) {return (comp){r*that.r-i*that.i, r*that.i+i*that.r};}
}w[N<<1], a[N<<1], b[N<<1], c[N<<1]; // 数组记得至少开两倍
void init()
{
    double pi = acos(-1.0);
    for(int i = 0; i < len; i++) w[i] = (comp){cos(2*pi*i/len), sin(2*pi*i/len)};
}
void FFT(comp *a, comp *w)
{
    for(int i = 0, j = 0; i < len; i++)
    {
        if(i<j) swap(a[i], a[j]);
        for(int l = len>>1; (j^=l)<l; l >>= 1);
    }
    for(int i = 2; i <= len; i <<= 1)
    {
        int m = i >> 1;
        for(int j = 0; j < len; j += i)
        {
            for(int k = 0; k < m; k++)
            {
                comp tmp = w[len/i*k] * a[j+k+m];
                a[j+k+m] = a[j+k] - tmp;
                a[j+k] = a[j+k] + tmp;
            }
        }
    }
}
void mul(comp *a, comp *b, comp *c, int l) // 多项式乘法, c = a * b , c的长度为l
{
    for(len = 1; len <= l; len <<= 1);
    init(); FFT(a, w); FFT(b, w);
    for(int i = 0; i < len; i++) c[i] = a[i] * b[i];
    reverse(c+1, c+len); FFT(c, w);
    for(int i = 0; i < len; i++) c[i].r /= len; // 转化为int等时应加0.5, 如int(c[i].r+0.5)
}
```

1.3 高精度-wrz

```
#include<cmath>
#include<cstdio>
#include<cstring>
```

```
#include<algorithm>
#define BASE 10000
#define L 20005
using namespace std;
int p;
char s[10*L];
struct bigint
{
    int num[L], len;
    bigint(int x = 0)
    {
        memset(num,0,sizeof(num));
        len = 1;
        num[0] = x;
    }
    bigint operator + (bigint b)
    {
        bigint c;
        c.len = max(b.len, len);
        for(int i = 0; i < c.len; i++)
        {
            c.num[i] += num[i] + b.num[i];
            c.num[i+1] = c.num[i] / BASE;
            c.num[i] %= BASE;
        }
        if(c.num[c.len])c.len++;
        return c;
    }
    bigint operator - (bigint b)
    {
        bigint c;
        c.len = max(len, b.len);
        for(int i = 0; i < c.len; i++)
        {
            c.num[i] += num[i] - b.num[i];
            if(c.num[i] < 0)
            {
                c.num[i] += BASE;
                c.num[i+1]--;
            }
        }
        while(!c.num[c.len-1] && c.len > 1)c.len--;
        return c;
    }
    void operator -= (int b)
    {
        num[0] -= b;
        for(int i = 0; i < len; i++)
        {
            num[i+1] += num[i] / BASE;
            num[i] %= BASE;
            if(num[i] < 0)num[i] += BASE, num[i+1]--;
        }
        while(!num[len-1] && len > 1) len--;
    }
    bigint operator * (bigint b)
    {
        bigint c;
        c.len = len + b.len;
        for(int i = 0; i < len; i++)
        {
            for(int j = 0; j < b.len; j++)
            {
                c.num[i+j] += num[i] * b.num[j];
                c.num[i+j+1] += c.num[i+j] / BASE;
                c.num[i+j] %= BASE;
            }
        }
        if(!c.num[c.len-1] && c.len > 1)c.len--;
        return c;
    }
    bigint operator * (int b)
    {
        bigint c;
        for(int i = 0; i < len; i++)
            c.num[i] = num[i] * b; // long long
        for(int i = 0; i < len; i++)
        {
            c.num[i+1] += c.num[i] / BASE;
            c.num[i] %= BASE;
        }
        c.len = len;
        while(c.num[c.len])c.len++;
        return c;
    }
    bool substract(bigint b, int pos)
    {
        if(len < b.len - pos)return false;
        else if(len == b.len-pos)
            for(int i = len-1; i>=0; i--)
                if(num[i] < b.num[i+pos])return false;
                else if(num[i] > b.num[i+pos])break;
        for(int i = 0; i < len; i++)
        {
            num[i] -= b.num[i+pos];
            if(num[i] < 0)
            {
                num[i] += BASE;
                num[i+1] --;
            }
        }
        while(!num[len-1] && len > 1)len--;
```

```
        return true;
    }

    // remember to change [BASE] to 10 !!!
    // [this] is the remainder
    bigint operator / (bigint b)
    {
        bigint c;
        if(len < b.len)return c;
        int k = len - b.len;
        c.len = k + 1;
        for(int i = len-1; i>=0; i--)
        {
            if(i>=k)b.num[i] = b.num[i-k];
            else b.num[i] = 0;
        }
        b.len = len;
        for(int i = 0; i <= k; i++)
            while(this->substract(b,i)) c.num[k-i]++;
        for(int i = 0; i < c.len; i++)
        {
            c.num[i+1] += c.num[i] / BASE;
            c.num[i] %= BASE;
        }
        while(!c.num[c.len-1] && c.len > 0) c.len--;
        return c;
    }

    // [this] is not the remainder
    bigint operator / (int b)
    {
        bigint c;
        int tmp = 0;
        for(int i = len-1; i>=0; i--)
        {
            tmp = tmp * BASE + num[i];
            c.num[i] = tmp / b;
            tmp %= b;
        }
        for(c.len = len; !c.num[c.len-1] && c.len > 1; c.len--);
        return c;
    }
    bool scan()
    {
        int n = -1;
        char ch = getchar();
        while(ch < '0' || ch > '9') if(ch == EOF)return false; else ch = getchar();
        while(ch >= '0' && ch <= '9') s[++n] = ch - '0', ch = getchar();
        len = 0;
        for(int i = n; i >= 0; i-=4)
        {
            num[len] += s[i];
            if(i>=1)num[len] += s[i-1] * 10;
            if(i>=2)num[len] += s[i-2] * 100;
            if(i>=3)num[len] += s[i-3] * 1000;
            ++len;
        }
        return true;
    }
    void clr()
    {
        memset(num,0,sizeof(num));
    }
    void print()
    {
        printf("%d",num[len-1]);
        for(int i = len-2; i>=0; i--)
            printf("%04d",num[i]);
        printf("\n");
    }
};
```

1.4 线性基-gwx

```
ll solve()
{
    ll res = 0;
    memset(b, 0, sizeof(b));
    for(int i = 1; i <= tot; i++)
        for(int j = 60; j >= 0; j--)
            if((a[i] >> j) & 1)
            {
                if(!b[j])
                {
                    b[j] = a[i];
                    break;
                }
                a[i] ^= b[j];
            }
    for(int i = 60; i >= 0; i--)
        res = max(res, res ^ b[i]);
    return res;
}
```

1.5 单纯形

```
// max{c * x | Ax <= b, x >= 0}的解, 无解返回空的vector, 否则就是解. 答案在an中
template <int MAXN = 100, int MAXM = 100>
struct simplex {
    int n, m; double a[MAXM][MAXN], b[MAXM], c[MAXN];
```

```
    bool infeasible, unbounded;
    double v, an[MAXN + MAXM]; int q[MAXN + MAXM];
    void pivot (int l, int e) {
        std::swap (q[e], q[l + n]);
        double t = a[l][e]; a[l][e] = 1; b[l] /= t;
        for (int i = 0; i < n; ++i) a[l][i] /= t;
        for (int i = 0; i < m; ++i) if (i != l && std::abs (a[i][e]) > EPS) {
            t = a[i][e]; a[i][e] = 0; b[i] -= t * b[l];
            for (int j = 0; j < n; ++j) a[i][j] -= t * a[l][j]; }
        if (std::abs (c[e]) > EPS) {
            t = c[e]; c[e] = 0; v += t * b[l];
            for (int j = 0; j < n; ++j) c[j] -= t * a[l][j]; } }
    bool pre () {
        for (int l, e; ; ) {
            l = e = -1;
            for (int i = 0; i < m; ++i) if (b[i] < -EPS && (!~l || rand () & 1)) l = i;
            if (!~l) return false;
            for (int i = 0; i < n; ++i) if (a[l][i] < -EPS && (!~e || rand () & 1)) e = i;
            if (!~e) return infeasible = true;
            pivot (l, e); } }
    double solve () {
        double p; std::fill (q, q + n + m, -1);
        for (int i = 0; i < n; ++i) q[i] = i;
        v = 0; infeasible = unbounded = false;
        if (pre ()) return 0;
        for (int l, e; ; pivot (l, e)) {
            l = e = -1; for (int i = 0; i < n; ++i) if (c[i] > EPS) { e = i; break; }
            if (!~e) break; p = INF;
            for (int i = 0; i < m; ++i) if (a[i][e] > EPS && p > b[i] / a[i][e])
                p = b[i] / a[i][e], l = i;
            if (!~l) return unbounded = true, 0; }
        for (int i = n; i < n + m; ++i) if (~q[i]) an[q[i]] = b[i - n];
        return v; } };
```

1.6 NTT-gwx

```
const int G;
int n, m, inm;
int rev[maxn], a[maxn], b[maxn];    //maxn > 2 ^ k

ll power(ll b, int k)
{
    ll res = 1;
    for(; k; k >>= 1, b = b * b % mod)
        if(k & 1)
            res = res * b % mod;
    return res;
}

void ntt(ll*a, int f)
{
    for(int i = 0; i < m; i++)
        if(rev[i] < i)
            swap(a[i], a[rev[i]]);
    for(int l = 2, h = 1; l <= m; h = l, l <<= 1)
    {
        int ur;
        if(f == 1)
            ur = power(G, (mod - 1) / l);
        else
            ur = power(G, mod - 1 - (mod - 1) / l);
        for(int i = 0; i < m; i += l)
        {
            ll w = 1;
            for(int k = i; k < i + h; k++, w = w * ur % mod)
            {
                int x = a[k], y = a[k + h] * w % mod;
                a[k] = (x + y) % mod;
                a[k + h] = (x - y + mod) % mod;
            }
        }
    }
    if(f == -1)
        for(int i = 0; i < m; i++)
            a[i] = a[i] * inm % mod;
}

void multi()
{
    ntt(a, 1); ntt(b, 1);
    for(int i = 0; i < m; i++)
        a[i] = a[i] * b[i] % mod;
    ntt(a, -1);
}

void init()
{
    for(m = 1; m <= 2 * n; m <<= 1) ;
    for(int i = 1; i < m; i++)
    {
        rev[i] = rev[i - 1];
        for(int j = m >> 1; (rev[i] ^= j) < j; j >>= 1) ;
    }
}
```

2 计算几何

2.1 二维计算几何-wrz

```
// c++11
```

```

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

const double inf = 1e9;
const double eps = 1e-9;
const double pi = acos(-1.0);

/*精度误差下的各种运算*/
bool le(double x, double y){return x < y - eps;} // x严格小于y
bool leq(double x, double y){return x < y + eps;} // x小于等于y
bool equ(double x, double y){return fabs(x - y) < eps;} // x等于y
double mysqrt(double x) {return x < eps ? 0 : sqrt(x);} // 开根号
double sqr(double x) {return x * x;} // 平方

struct point // 点或向量
{
    double x, y;
    double operator * (point that){return x*that.x + y*that.y;}
    double operator ^ (point that){return x*that.y - y*that.x;}
    point operator * (double t){return (point){x*t, y*t};}
    point operator / (double t){return (point){x/t, y/t};}
    point operator + (point that) {return (point){x + that.x, y + that.y};}
    point operator - (point that) {return (point){x - that.x, y - that.y};}
    double len(){return mysqrt(x*x+y*y);} // 到原点距离/向量长度
    point reset_len(double t) // 改变向量长度为t，t为正则方向不变，t为负则方向相反
    {
        double p = len();
        return (point){x*t/p, y*t/p};
    }
    point rot90() {return (point){-y, x};} // 逆时针旋转90度
    point rotate(double angle) // 使向量逆时针旋转angle弧度
    {
        double c = cos(angle), s = sin(angle);
        return (point){c * x  - s * y, s * x + c * y};
    }
};

struct line // 参数方程表示，p为线上一点，v为方向向量
{
    point p, v; // p为线上一点，v为方向向量

    double angle; // 半平面交用，用atan2计算，此时v的左侧为表示的半平面。注意有的函数声明一个新的line时没有初始化这个值！
    bool operator < (const line &that) const {return angle < that.angle;} // 半平面交用，按与x轴夹角排序
};

struct circle
{
    point c; double r;
};


double distance(point a, point b) // a，b两点距离
{
    return mysqrt(sqr(a.x - b.x) + sqr(a.y - b.y));
}

circle make_circle(point a, point b) // 以a，b两点为直径作圆
{
    double d = distance(a, b);
    return (circle){(a+b)/2, d/2};
}

double point_to_line(point a, line b) // 点a到直线b距离
{
    return fabs((b.v ^ (a - b.p)) / b.v.len());
}

point project_to_line(point a, line b) // 点a到直线b的垂足/投影
{
    return b.v.reset_len((a - b.p) * b.v / b.v.len()) + b.p;
}

vector<point> circle_inter(circle a, circle b) // 圆a和圆b的交点，需保证两圆不重合，圆的半径必须大于0
{
    double d = distance(a.c, b.c);
    vector<point> ret;
    if(le(a.r + b.r, d) || le(a.r + d, b.r) || le(b.r + d, a.r)) return vector<point>(); // 相离或内含
    point r = (b.c - a.c).reset_len(1);
    double x = ((sqr(a.r) - sqr(b.r)) / d + d) / 2;
    double h = mysqrt(sqr(a.r) - sqr(x));
    if(equ(h, 0)) return vector<point>({a.c + r * x}); // 内切或外切
    else return vector<point>({a.c + r*x + r.rot90()*h, a.c + r*x - r.rot90()*h}); // 相交两点
}

vector<point> line_circle_inter(line a, circle b) // 直线a和圆b的交点
{
    double d = point_to_line(b.c, a);
    if(le(b.r, d)) return vector<point>(); // 不交
    double x = mysqrt(sqr(b.r) - sqr(d));
    point p = project_to_line(b.c, a);
    if(equ(x, 0)) return vector<point> ({p}); // 相切
    else return vector<point> ({p + a.v.reset_len(x), p - a.v.reset_len(x)}); // 相交两点
}

point line_inter(line a, line b) // 直线a和直线b的交点，需保证两直线不平行
{
    double s1 = a.v ^ (b.p - a.p);
```

```
    double s2 = a.v ^ (b.p + b.v - a.p);
    return (b.p * s2 - (b.p + b.v) * s1) / (s2 - s1);
}

vector<point> tangent(point p, circle a) // 过点p的圆a的切线的切点，圆的半径必须大于0
{
    circle c = make_circle(p, a.c);
    return circle_inter(a, c);
}

vector<line> intangent(circle a, circle b) // 圆a和圆b的内公切线
{
    point p = (b.c * a.r + a.c * b.r) / (a.r + b.r);
    vector<point> va = tangent(p, a), vb = tangent(p, b);
    vector<line> ret;
    if(va.size() == 2 && vb.size() == 2)
    {
        ret.push_back((line){va[0], vb[0] - va[0]});
        ret.push_back((line){va[1], vb[1] - va[1]});
    }
    else if(va.size() == 1 && vb.size() == 1)
    {
        ret.push_back((line){p, (a.c - b.c).rot90()});
    }
    return ret;
}

// 判断半平面交是否有解，若有解需保证半平面交必须有界，可以通过外加四个大半平面解决
// lcnt为半平面数量，l为需要做的所有半平面的数组，p为存交点的临时数组，h为时刻更新的合法的半平面数组，下标均从1开
    始
bool HP(int lcnt, line *l, line *h, point *p)
{
    sort(l+1, l+1+lcnt);
    int head = 1, tail = 1;
    h[1] = l[1];
    for(int i = 2; i <= lcnt; i++)
    {
        line cur = l[i];
        for(; head < tail && le(cur.v ^ (p[tail-1]-cur.p), 0); tail--); // 先删队尾再删队头，顺序不能换
        for(; head < tail && le(cur.v ^ (p[head]-cur.p), 0); head++);
        h[++tail] = cur;
        if(equ(h[tail].v ^ h[tail-1].v, 0)) // 平行
        {
            if(le(h[tail].v * h[tail-1].v, 0)) return false; // 方向相反的平行直线，显然已经不可能围出有界半平面
                了
            tail--;
            if(le(h[tail+1].v ^ (h[tail].p - h[tail+1].p), 0)) h[tail] = h[tail+1];
        }
        if(head < tail) p[tail-1] = line_inter(h[tail-1], h[tail]);
    }
    for(; head < tail && le(h[head].v ^ (p[tail-1]-h[head].p), 0); tail--);
    return tail - head > 1;
}

double calc(double X){return 0;} // 计算给定X坐标上的覆盖的长度，配合辛普森积分使用
// 自适应辛普森积分，参数分别为(左端点x坐标，中点x坐标，右端点x坐标，左端点答案，中点答案，右端点答案)
// 改变计算深度应调整eps
double simpson(double l, double mid, double r, double fl, double fm, double fr)
{
    double lmid = (l+mid)/2, rmid = (r+mid)/2, flm = calc(lmid), frm = calc(rmid);
    double ans = (r-l) * (fl + 4*fm + fr), ansl = (mid-l) * (fl + 4*flm + fm), ansr = (r-mid) * (fm + 4*frm + fr)
        ;
    if(fabs(ansl + ansr - ans) < eps) return ans / 6;
    else return simpson(l,lmid,mid,fl,flm,fm) + simpson(mid,rmid,r,fm,frm,fr);
}


int main(){}
```

2.2 basis

```
int sign(DB x) {
    return (x > eps) - (x < -eps);
}
DB msqrt(DB x) {
    return sign(x) > 0 ? sqrt(x) : 0;
}

struct Point {
    DB x, y;
    Point rotate(DB ang) const {  // 逆时针旋转 ang 弧度
        return Point(cos(ang) * x - sin(ang) * y,
                cos(ang) * y + sin(ang) * x);
    }
    Point turn90() const {  // 逆时针旋转 90 度
        return Point(-y, x);
    }
    Point unit() const {
        return *this / len();
    }
};
DB dot(const Point& a, const Point& b) {
    return a.x * b.x + a.y * b.y;
}
DB det(const Point& a, const Point& b) {
    return a.x * b.y - a.y * b.x;
}
#define cross(p1,p2,p3) ((p2.x-p1.x)*(p3.y-p1.y)-(p3.x-p1.x)*(p2.y-p1.y))
#define crossOp(p1,p2,p3) sign(cross(p1,p2,p3))
bool isLL(const Line& l1, const Line& l2, Point& p) {  // 直线与直线交点
```

```cpp
    DB s1 = det(l2.b - l2.a, l1.a - l2.a),
       s2 = -det(l2.b - l2.a, l1.b - l2.a);
    if (!sign(s1 + s2)) return false;
    p = (l1.a * s2 + l1.b * s1) / (s1 + s2);
    return true;
}
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) {
    return l.a + (l.b - l.a) * (dot(p - l.a, l.b - l.a) / (l.b - l.a).len2());
}
DB disToLine(const Line& l, const Point& p) {  // 点到*直线*距离
    return fabs(det(p - l.a, l.b - l.a) / (l.b - l.a).len());
}
DB 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) : std::min((p - l
        .a).len(), (p - l.b).len());
}
// 圆与直线交点
bool isCL(Circle a, Line l, Point& p1, Point& p2) {
    DB 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;
    Point p = l.a - ((l.b - l.a) * (x / y)), delta = (l.b - l.a) * (msqrt(d) / y);
    p1 = p + delta; p2 = p - delta;
    return true;
}
//圆与圆的交面积
DB areaCC(const Circle& c1, const Circle& c2) {
    DB d = (c1.o - c2.o).len();
    if (sign(d - (c1.r + c2.r)) >= 0) return 0;
    if (sign(d - std::abs(c1.r - c2.r)) <= 0) {
        DB r = std::min(c1.r, c2.r);
        return r * r * PI;
    }
    DB 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);
}
// 圆与圆交点
bool isCC(Circle a, Circle b, P& p1, P& p2) {
    DB s1 = (a.o - b.o).len();
    if (sign(s1 - a.r - b.r) > 0 || sign(s1 - std::abs(a.r - b.r)) < 0) return false;
    DB s2 = (a.r * a.r - b.r * b.r) / s1;
    DB aa = (s1 + s2) * 0.5, bb = (s1 - s2) * 0.5;
    P o = (b.o - a.o) * (aa / (aa + bb)) + a.o;
    P delta = (b.o - a.o).unit().turn90() * msqrt(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
std::vector<Line> intanCC(const Circle &c1, const Circle &c2) {
    std::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;  // Here I guess.
        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;
}
// 用半平面 (q1,q2) 的逆时针方向去切凸多边形
std::vector<Point> convexCut(const std::vector<Point>&ps, Point q1, Point q2) {
    std::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 = crossOp(q1,q2,p1), d2 = crossOp(q1,q2,p2);
        if (d1 >= 0) qs.push_back(p1);
        if (d1 * d2 < 0) qs.push_back(isSS(p1, p2, q1, q2));
    }
    return qs;
}
// 求凸包
std::vector<Point> convexHull(std::vector<Point> ps) {
    int n = ps.size(); if (n <= 1) return ps;
    std::sort(ps.begin(), ps.end());
    std::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[qs.size() - 2], qs.back(), ps[i])) <= 0)
            qs.pop_back();
    return qs;
```

2.3 circle-arc-struct

```
struct circle {
    point o;
    double r;
    circle(point o, double r) : o(o), r(r) {}
};
struct arcs { // 点l顺时针到点r
    point o, l, r;
    arcs() {}
    arcs(point o, point l, point r) : o(o), l(l), r(r) {}
};
bool isCL(circle a, line l, point &p1, point &p2) { // 圆与直线的交点
    double x = dot(l.a - a.o, l.b - l.a);
    double y = (l.b - l.a).len2();
    double 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;
}
double ang(const point &d1, const point &d2) { // 向量d1顺时针转到向量d2的角度
    if (sign(det(d1, d2)) == 0)
        return (sign(dot(d1, d2)) > 0) ? 0 : pi;
    if (sign(det(d1, d2)) < 0)
        return acos(dot(d1, d2) / d1.len() / d2.len());
    else return 2 * pi - acos(dot(d1, d2) / d1.len() / d2.len());
}
bool onArcs(const point &p, const arcs &a) { // 点在圆弧上
    return sign(ang(a.l - a.o, a.r - a.o) - ang(a.l - a.o, p - a.o)) > 0;
}

/*struct circle {
    point o;
    double r;
    circle(point o, double r) : o(o), r(r) {}
};

struct arcs {
    //l -> r clockwise
    point o, l, r;
    arcs() {}
    arcs(point o, point l, point r) : o(o), l(l), r(r) {}
};

//Circle intersect with Line
bool isCL(circle a, line l, point &p1, point &p2) {
    double x = dot(l.a - a.o, l.b - l.a);
    double y = (l.b - l.a).len2();
    double 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;
}

//use acos !precision

//angle (d1, d2) clockwise
double ang(const point &d1, const point &d2) {
    if (sign(det(d1, d2)) == 0)
        return (sign(dot(d1, d2)) > 0) ? 0 : pi;
    if (sign(det(d1, d2)) < 0)
        return acos(dot(d1, d2) / d1.len() / d2.len());
    else return 2 * pi - acos(dot(d1, d2) / d1.len() / d2.len());
}
//
bool onArcs(const point &p, const arcs &a) {
    return sign(ang(a.l - a.o, a.r - a.o) - ang(a.l - a.o, p - a.o)) > 0;
}*/
```

2.4 circles-intersections

```
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; }
Circle c[N];
double area[N];  // area[k] -> area of intersections >= k.
Point centroid[N];
bool keep[N];
void add(int cnt, DB a, Point c) {
    area[cnt] += a;
    centroid[cnt] = centroid[cnt] + c * a;
}
void solve(int C) {
    for (int i = 1; i <= C; ++ i) {
        area[i] = 0;
        centroid[i] = Point(0, 0);
    }
    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() == 0u) {
            add(cnt, PI * c[i].r * c[i].r, c[i].o);
        } 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;
                add(cnt, det(evt[j].p, evt[j + 1].p) / 2, (evt[j].p + evt[j + 1].p) / 3);
                double ang = evt[j + 1].ang - evt[j].ang;
                if (ang < 0) {
                    ang += PI * 2;
                }
                if (sign(ang) == 0) continue;
                add(cnt, ang * c[i].r * c[i].r / 2, c[i].o +
                    Point(sin(ang1) - sin(ang0), -cos(ang1) + cos(ang0)) * (2 / (3 * ang) * c[i].r));
                add(cnt, -sin(ang) * c[i].r * c[i].r / 2, (c[i].o + evt[j].p + evt[j + 1].p) / 3);
            }
        }
    }
    for (int i = 1; i <= C; ++ i)
        if (sign(area[i])) {
            centroid[i] = centroid[i] / area[i];
        }
}
```

2.5 cirque-area-merge

```
//n^2*logn
struct point {
    point rotate(const double &ang) {
        return point(cos(ang) * x - sin(ang) * y, cos(ang) * y + sin(ang) * x);
    }
    double ang() {
        return atan2(y, x);
    }
};
struct Circle {
    point o;
    double r;
    int tp; // 正圆为1 反向圆为-1
    Circle (point o = point(0, 0), double r = 0, int tp = 0) : o(o), r(r), tp(tp) {}
};
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 / 2;
    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, b.tp));
    evt.push_back(Event(q0, ang0, -b.tp));
    cnt += (ang1 > ang0) * b.tp;
}
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() { // area[1]..area[C]
    memset(area, 0, sizeof(double) * (C + 1));
    for (int i = 0; i < C; ++i) {
        int cnt = (c[i].tp > 0);
        vector<Event> evt;
        for (int j = 0; j < i; ++j) if (issame(c[i], c[j])) cnt += c[j].tp;
        for (int j = 0; j < C; ++j)
            if (j != i && !issame(c[i], c[j]) && overlap(c[j], c[i])) cnt += c[j].tp;
        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] += c[i].tp * 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] += c[i].tp * 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] += c[i].tp * (ang * c[i].r * c[i].r / 2 - sin(ang) * c[i].r * c[i].r / 2);
            }
        }
    }
}
```

2.6 凸包

```
// 凸包中的点按逆时针方向
struct Convex {
    int n;
    std::vector<Point> a, upper, lower;
    void make_shell(const std::vector<Point>& p,
            std::vector<Point>& shell) {  // p needs to be sorted.
        clear(shell); int n = p.size();
        for (int i = 0, j = 0; i < n; i++, j++) {
            for (; j >= 2 && sign(det(shell[j-1] - shell[j-2],
                            p[i] - shell[j-2])) <= 0; --j) shell.pop_back();
            shell.push_back(p[i]);
        }
    }
    void make_convex() {
        std::sort(a.begin(), a.end());
        make_shell(a, lower);
        std::reverse(a.begin(), a.end());
        make_shell(a, upper);
        a = lower; a.pop_back();
        a.insert(a.end(), upper.begin(), upper.end());
        if ((int)a.size() >= 2) a.pop_back();
        n = a.size();
    }
    void init(const std::vector<Point>& _a) {
        clear(a); a = _a; n = a.size();
        make_convex();
    }
    void read(int _n) {  // Won't make convex.
        clear(a); n = _n; a.resize(n);
        for (int i = 0; i < n; i++)
            a[i].read();
    }
    std::pair<DB, int> get_tangent(
            const std::vector<Point>& convex, const Point& vec) {
        int l = 0, r = (int)convex.size() - 2;
        assert(r >= 0);
        for (; l + 1 < r; ) {
            int mid = (l + r) / 2;
            if (sign(det(convex[mid + 1] - convex[mid], vec)) > 0)
                r = mid;
            else l = mid;
        }
        return std::max(std::make_pair(det(vec, convex[r]), r),
```

```
                std::make_pair(det(vec, convex[0]), 0));
    }
    int binary_search(Point u, Point v, int l, int r) {
        int s1 = sign(det(v - u, a[l % n] - u));
        for (; l + 1 < r; ) {
            int mid = (l + r) / 2;
            int smid = sign(det(v - u, a[mid % n] - u));
            if (smid == s1) l = mid;
            else r = mid;
        }
        return l % n;
    }
    // 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) {
        std::pair<DB, int> ret = get_tangent(upper, vec);
        ret.second = (ret.second + (int)lower.size() - 1) % n;
        ret = std::max(ret, get_tangent(lower, vec));
        return ret.second;
    }
    // 求凸包和直线 u, v 的交点，如果不相交返回 false，如果有则是和 (i, next(i)) 的交点，交在点上不确定返回前后两
        条边其中之一
    bool get_intersection(Point u, Point v, int &i0, int &i1) {
        int p0 = get_tangent(u - v), p1 = get_tangent(v - u);
        if (sign(det(v - u, a[p0] - u)) * sign(det(v - u, a[p1] - u)) <= 0) {
            if (p0 > p1) std::swap(p0, p1);
            i0 = binary_search(u, v, p0, p1);
            i1 = binary_search(u, v, p1, p0 + n);
            return true;
        }
        else return false;
    }
};
```

2.7 point-in-polygon

```
bool pit_in_polygon(pit q){ // µ
    int cnt = 0;
    for(int i = 1; i <= n; ++i){
        pit p1 = p[i];
        pit p2 = p[suc[i]];
        if(pit_on_seg(q, p1, p2)) return true;
        int k = dcmp(det(p2 - p1, q - p1));
        int d1 = dcmp(p1.y - q.y);
        int d2 = dcmp(p2.y - q.y);
        if(k > 0 && d1 <= 0 && d2 > 0) ++cnt;
        if(k < 0 && d2 <= 0 && d1 > 0) --cnt;
    }
    if(cnt != 0) return true;
    else return false;
}
bool seg_in_polygon(pit a, pit b){ //
    vec v = b - a;
    for(int t = 1; t <= 1000; ++t){
        pit c = a + v * (1.00 * (rand() % 10000) / 10000);
        if(pit_in_polygon(c)) continue;
        else return false;
    }
    return true;
}
```

2.8 point-struct

```
const double eps = 1e-8;
const double PI = acos(-1.0);

int sign(double x) {
    return (x < -eps) ? -1 : (x > eps);
}
double sqr(double x) {
    return x * x;
}

struct point {
    double x, y;
    point(double x = 0, double y = 0) : x(x), y(y) {}
    point operator + (const point &rhs) const {
        return point(x + rhs.x, y + rhs.y);
    }
    point operator - (const point &rhs) const {
        return point(x - rhs.x, y - rhs.y);
    }
    point operator * (const double &k) const {
        return point(x * k, y * k);
    }
    point operator / (const double &k) const {
        return point(x / k, y / k);
    }
    double len2() {
        return x * x + y * y;
    }
    double len() {
        return sqrt(len2());
    }
    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);
    }
    double ang() {
```

```
        return atan2(y, x);
    }
    point operator < (const point &rhs) {
        return (x < rhs.x || x == rhs.x && y < rhs.y);
    }
};
double dot(const point &a, const point &b) {
    return a.x * b.x + a.y * b.y;
}
double 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 b) : a(a), b(b) {}
};
bool onSeg(const point &p, const line &l) { // 点在线段上 包含端点
    return sign(det(p - l.a, l.b - l.a)) == 0 && sign(dot(p - l.a, p - l.b)) <= 0;
}
double disToLine(const point &p, const line &l) { // 点到直线距离
    return fabs(det(p - l.a, l.b - l.a) / (l.b - l.a).len());
}
double disToSeg(const point &p, const line &l) { // 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 projection(const point &p, const line &l) { // 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());
}
point symmetry(const point &a, const point &b) { // 点a关于点b的对称点
    return b + b - a;
}
point reflection(const point &p, const line &l) { // 点关于直线的对称点
    return symmetry(p, projection(p, l));
}
bool isLL(const line &l1, const line &l2, point &p) { // 直线求交
    long long s1 = det(l2.b - l2.a, l1.a - l2.a);
    long long s2 = -det(l2.b - l2.a, l1.b - l2.a);
    if(!sign(s1 + s2)) return false;
    p = (l1.a * s2 + l1.b * s1) / (s1 + s2);
    return true;
}
bool p_in_tri(const point &p, const point &a, const point &b, const point &c) { //点在三角形内(包含边界)
    return sign(abs(det(a - p, b - p)) + abs(det(b - p, c - p))
        + abs(det(c - p, a - p)) - abs(det(b - a, c - a))) == 0;
}
bool Check(const point &p, const point &d, const point &a, const point &b) {
    return sign(det(d, a - p)) * sign(det(b - p, d)) >= 0;
}
bool isll(const point &p, const point &q, const point &a, const point &b) { // 跨立实验
    return Check(p, q - p, a, b) && Check(a, b - a, p, q);
}
```

2.9 三角形内心，外心，垂心

```
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);
}
```

2.10 三维计算几何

```
// 三维绕轴旋转，大拇指指向 axis 向量方向，四指弯曲方向转 w 弧度
Point rotate(const Point& s, const Point& axis, DB w) {
    DB x = axis.x, y = axis.y, z = axis.z;
    DB s1 = x * x + y * y + z * z, ss1 = msqrt(s1),
       cosw = cos(w), sinw = sin(w);
    DB a[4][4];
    memset(a, 0, sizeof a);
    a[3][3] = 1;
    a[0][0] = ((y * y + z * z) * cosw + x * x) / s1;
    a[0][1] = x * y * (1 - cosw) / s1 + z * sinw / ss1;
    a[0][2] = x * z * (1 - cosw) / s1 - y * sinw / ss1;
    a[1][0] = x * y * (1 - cosw) / s1 - z * sinw / ss1;
    a[1][1] = ((x * x + z * z) * cosw + y * y) / s1;
    a[1][2] = y * z * (1 - cosw) / s1 + x * sinw / ss1;
    a[2][0] = x * z * (1 - cosw) / s1 + y * sinw / ss1;
    a[2][1] = y * z * (1 - cosw) / s1 - x * sinw / ss1;
    a[2][2] = ((x * x + y * y) * cos(w) + z * z) / s1;
    DB ans[4] = {0, 0, 0, 0}, c[4] = {s.x, s.y, s.z, 1};
    for (int i = 0; i < 4; ++ i)
        for (int j = 0; j < 4; ++ j)
            ans[i] += a[j][i] * c[j];
```

```
    return Point(ans[0], ans[1], ans[2]);
}
```

2.11 三维凸包

```
__inline P cross(const P& a, const P& b) {
    return P(
            a.y * b.z - a.z * b.y,
            a.z * b.x - a.x * b.z,
            a.x * b.y - a.y * b.x
        );
}

__inline DB mix(const P& a, const P& b, const P& c) {
    return dot(cross(a, b), c);
}

__inline DB volume(const P& a, const P& b, const P& c, const P& d) {
    return mix(b - a, c - a, d - a);
}

struct Face {
    int a, b, c;
    __inline Face() {}
    __inline Face(int _a, int _b, int _c):
        a(_a), b(_b), c(_c) {}
    __inline DB area() const {
        return 0.5 * cross(p[b] - p[a], p[c] - p[a]).len();
    }
    __inline P normal() const {
        return cross(p[b] - p[a], p[c] - p[a]).unit();
    }
    __inline DB dis(const P& p0) const {
        return dot(normal(), p0 - p[a]);
    }
};

std::vector<Face> face, tmp;  // Should be O(n).
int mark[N][N], Time, n;

__inline void add(int v) {
    ++ Time;
    clear(tmp);
    for (int i = 0; i < (int)face.size(); ++ i) {
        int a = face[i].a, b = face[i].b, c = face[i].c;
        if (sign(volume(p[v], p[a], p[b], p[c])) > 0) {
            mark[a][b] = mark[b][a] = mark[a][c] =
                mark[c][a] = mark[b][c] = mark[c][b] = Time;
        }
        else {
            tmp.push_back(face[i]);
        }
    }
    clear(face); face = tmp;
    for (int i = 0; i < (int)tmp.size(); ++ i) {
        int a = face[i].a, b = face[i].b, c = face[i].c;
        if (mark[a][b] == Time) face.emplace_back(v, b, a);
        if (mark[b][c] == Time) face.emplace_back(v, c, b);
        if (mark[c][a] == Time) face.emplace_back(v, a, c);
        assert(face.size() < 500u);
    }
}

void reorder() {
    for (int i = 2; i < n; ++ i) {
        P tmp = cross(p[i] - p[0], p[i] - p[1]);
        if (sign(tmp.len())) {
            std::swap(p[i], p[2]);
            for (int j = 3; j < n; ++ j)
                if (sign(volume(p[0], p[1], p[2], p[j]))) {
                    std::swap(p[j], p[3]);
                    return;
                }
        }
    }
}

void build_convex() {
    reorder();
    clear(face);
    face.emplace_back(0, 1, 2);
    face.emplace_back(0, 2, 1);
    for (int i = 3; i < n; ++ i)
        add(i);
}
```

3 数据结构

3.1 KD 树

```
long long norm(const long long &x) {
    //    For manhattan distance
    return std::abs(x);
    //    For euclid distance
    return x * x;
}

struct Point {
    int x, y, id;

```

```
    const int& operator [] (int index) const {
        if (index == 0) {
            return x;
        } else {
            return y;
        }
    }

    friend long long dist(const Point &a, const Point &b) {
        long long result = 0;
        for (int i = 0; i < 2; ++i) {
            result += norm(a[i] - b[i]);
        }
        return result;
    }
} point[N];

struct Rectangle {
    int min[2], max[2];

    Rectangle() {
        min[0] = min[1] = INT_MAX;  // sometimes int is not enough
        max[0] = max[1] = INT_MIN;
    }

    void add(const Point &p) {
        for (int i = 0; i < 2; ++i) {
            min[i] = std::min(min[i], p[i]);
            max[i] = std::max(max[i], p[i]);
        }
    }

    long long dist(const Point &p) {
        long long result = 0;
        for (int i = 0; i < 2; ++i) {
            //    For minimum distance
            result += norm(std::min(std::max(p[i], min[i]), max[i]) - p[i]);
            //    For maximum distance
            result += std::max(norm(max[i] - p[i]), norm(min[i] - p[i]));
        }
        return result;
    }
};

struct Node {
    Point seperator;
    Rectangle rectangle;
    int child[2];

    void reset(const Point &p) {
        seperator = p;
        rectangle = Rectangle();
        rectangle.add(p);
        child[0] = child[1] = 0;
    }
} tree[N << 1];

int size, pivot;

bool compare(const Point &a, const Point &b) {
    if (a[pivot] != b[pivot]) {
        return a[pivot] < b[pivot];
    }
    return a.id < b.id;
}

// 左閉右開: build(1, n + 1)
int build(int l, int r, int type = 1) {
    pivot = type;
    if (l >= r) {
        return 0;
    }
    int x = ++size;
    int mid = l + r >> 1;
    std::nth_element(point + l, point + mid, point + r, compare);
    tree[x].reset(point[mid]);
    for (int i = l; i < r; ++i) {
        tree[x].rectangle.add(point[i]);
    }
    tree[x].child[0] = build(l, mid, type ^ 1);
    tree[x].child[1] = build(mid + 1, r, type ^ 1);
    return x;
}

int insert(int x, const Point &p, int type = 1) {
    pivot = type;
    if (x == 0) {
        tree[++size].reset(p);
        return size;
    }
    tree[x].rectangle.add(p);
    if (compare(p, tree[x].seperator)) {
        tree[x].child[0] = insert(tree[x].child[0], p, type ^ 1);
    } else {
        tree[x].child[1] = insert(tree[x].child[1], p, type ^ 1);
    }
    return x;
}

// For minimum distance
// For maximum:下面递归query时0, 1 换顺序;< and >;min and max
void query(int x, const Point &p, std::pair<long long, int> &answer, int type = 1) {
```

```cpp
    pivot = type;
    if (x == 0 || tree[x].rectangle.dist(p) > answer.first) {
        return;
    }
    answer = std::min(answer,
             std::make_pair(dist(tree[x].seperator, p), tree[x].seperator.id));
    if (compare(p, tree[x].seperator)) {
        query(tree[x].child[0], p, answer, type ^ 1);
        query(tree[x].child[1], p, answer, type ^ 1);
    } else {
        query(tree[x].child[1], p, answer, type ^ 1);
        query(tree[x].child[0], p, answer, type ^ 1);
    }
}

std::priority_queue<std::pair<long long, int> > answer;

void query(int x, const Point &p, int k, int type = 1) {
    pivot = type;
    if (x == 0 || (int)answer.size() == k && tree[x].rectangle.dist(p) > answer.top().first) {
        return;
    }
    answer.push(std::make_pair(dist(tree[x].seperator, p), tree[x].seperator.id));
    if ((int)answer.size() > k) {
        answer.pop();
    }
    if (compare(p, tree[x].seperator)) {
        query(tree[x].child[0], p, k, type ^ 1);
        query(tree[x].child[1], p, k, type ^ 1);
    } else {
        query(tree[x].child[1], p, k, type ^ 1);
        query(tree[x].child[0], p, k, type ^ 1);
    }
}
```

3.2 KD 树-gwx

```cpp
struct Point
{
    double x, y;
    int id;
    Point operator - (const Point &a) const {
        return (Point){x - a.x, y - a.y, id};
    }
} b[maxn], c[maxn];
struct node
{
    Point p;
    int ch[2];
} a[maxn];
struct rev
{
    int id;
    double dis;
    bool operator < (const rev &a) const{
        int tmp = sign(dis - a.dis);
        if(tmp)
            return tmp < 0;
        return id < a.id;
    }
};
typedef pr priority_queue <rev>;
pr p0;

int build(int l, int r, int f)
{
    if(l > r)
        return 0;
    int x = (l + r) >> 1;
    if(f == 0)
        nth_element(a + l, a + x, a + r + 1, cmp0); //按x排序
    else
        nth_element(a + l, a + x, a + r + 1, Cmp1); //按y排序
    a[x].ch[0] = build(l, x - 1, f ^ 1);
    a[x].ch[1] = build(x + 1, r, f ^ 1);
    return x;
}

void update(pr &a, rev x)
{
    if(a.size() < K)
        a.push(x);
    else if(x < a.top())
    {
        a.pop();
        a.push(x);
    }
}

pr merge(pr a, pr b)
{
    int s1 = a.size(), s2 = b.size();
    if(s1 < s2)
    {
        while(!a.empty())
        {
            update(b, a.top());
            a.pop();
        }
        return b;
    }
```

```
    else
    {
        while(!b.empty())
        {
            update(a, b.top());
            b.pop();
        }
        return a;
    }
}

pr query(int u, Point x, int f)
{
    if(!u)
        return p0;//empty priority_queue
    int d = (dis(a[a[u].ch[0]].p, x) > dis(a[a[u].ch[1]].p, x));
    double dx;
    pr res = query(a[u].ch[d], x, f ^ 1);
    update(res, (rev){a[u].p.id, dis(a[u].p, x)});
    if(f == 0)
        dx = abs(x.x - a[u].p.x);
    else
        dx = abs(x.y - a[u].p.y);
    if(dx > res.top().dis)
        return res;
    res = merge(res, query(a[u].ch[d ^ 1], x, f ^ 1));
    return res;
}

pr solve(Point p)   //离p最近的K个点
{
    int root = build(1, tot, 0);
    return query(root, p, 0);
}
```

3.3 lct-gwx

```
int pa[maxn], st[maxn];
struct node
{
    int ch[2], pa;
    ll s, w, sw;    //s: size of subtree; w: value; sw: sum of value
    ll m, p;    //tags of addition and multiplication
    bool f; //tag of flip
} a[maxn];

void flip(int u)
{
    if(!u) return;
    swap(a[u].ch[0], a[u].ch[1]);
    a[u].f ^= 1;
}

void add(int u, int c)
{
    if(!u) return;
    (a[u].p += c) %= mod;
    (a[u].w += c) %= mod;
    (a[u].sw += a[u].s * c % mod) % mod;
}

void mult(int u, int c)
{
    if(!u) return;
    if(a[u].m == -1) a[u].m = c;
    else (a[u].m *= c) %= mod;
    (a[u].p *= c) %= mod;
    (a[u].w *= c) %= mod;
    (a[u].sw *= c) %= mod;
}

void pushdown(int u)
{
    if(!u) return;
    int l = a[u].ch[0], r = a[u].ch[1];
    if(a[u].m != -1)
    {
        mult(l, a[u].m); mult(r, a[u].m);
        a[u].m = -1;
    }
    if(a[u].p)
    {
        add(l, a[u].p); add(r, a[u].p);
        a[u].p = 0;
    }
    if(a[u].f)
    {
        flip(l); flip(r);
        a[u].f ^= 1;
    }
}

void maintain(int u)
{
    pushdown(u);
    int l = a[u].ch[0], r = a[u].ch[1];
    a[u].s = a[l].s + a[r].s + 1;
    a[u].sw = (a[l].sw + a[r].sw + a[u].w) % mod;
}

void rotate(int u)
```

```
{
    int x = a[u].pa, y = a[x].pa, d = (a[x].ch[1] == u);
    if(!y) pa[u] = pa[x], pa[x] = 0;
    else a[y].ch[a[y].ch[1] == x] = u;
    a[x].ch[d] = a[u].ch[d ^ 1], a[a[u].ch[d ^ 1]].pa = x;
    a[u].ch[d ^ 1] = x; a[x].pa = u; a[u].pa = y;
    maintain(x); maintain(u);
}

void splay(int u)
{
    int t = u;
    while(a[t].pa) st[++top] = t, t = a[t].pa;
    pushdown(t);
    while(top) pushdown(st[top]), top--;
    while(a[u].pa)
    {
        int x = a[u].pa, y = a[x].pa;
        if(!y)  {rotate(u); return;}
        if(a[x].ch[1] == u ^ a[y].ch[1] == x) rotate(u);
        else rotate(x);
        rotate(u);
    }
}

void access(int u)
{
    splay(u);
    if(a[u].ch[1])
        a[a[u].ch[1]].pa = 0, pa[a[u].ch[1]] = u, a[u].ch[1] = 0, maintain(u);
    while(pa[u])
    {
        int v = pa[u];
        splay(v);
        if(a[v].ch[1])
            a[a[v].ch[1]].pa = 0, pa[a[v].ch[1]] = v, a[v].ch[1] = 0;
        a[v].ch[1] = u; a[u].pa = v; pa[u] = 0;
        maintain(v); splay(u);
    }
}

void sroot(int u)
{
    access(u); flip(u);
}

void get(int u, int v)
{
    sroot(u); access(v);
}

void cut(int u, int v)
{
    get(u, v); a[v].ch[0] = a[u].pa = 0;
    maintain(v);
}

void join(int u, int v)
{
    access(u); sroot(v);
    a[u].ch[1] = v; a[v].pa = u;
    maintain(u);
}
```

3.4 LCT-wrz

```
struct node
{
    node *ch[2], *fa;
    uint v, sum, k, b; int rev, siz;
}mem[N], *tot, *null, *pos[N];
void init()
{
    null = tot = mem;
    null->ch[0] = null->ch[1] = null->fa = null;
    null->v = null->sum = null->b = null->rev = null->siz = 0; null->k = 1;
    for(int i = 1; i <= n; i++) pos[i] = ++tot, *pos[i] = *null, pos[i]->v = pos[i]->sum = 1;
}
int type(node *x){return x->fa->ch[1]==x?1:0;}
int isroot(node *x){return x->fa->ch[type(x)] != x;}
void mswap(node *&x, node *&y){node *t = x; x = y; y = t;}
void pushup(node *x)
{
    x->sum = (x->v + x->ch[0]->sum + x->ch[1]->sum) % MOD;
    x->siz = (x->ch[0]->siz + x->ch[1]->siz + 1) % MOD;
}
void pushdown(node *x)
{
    if(x->rev)
    {
        x->rev = 0, x->ch[0]->rev ^= 1, x->ch[1]->rev ^= 1;
        mswap(x->ch[0]->ch[0], x->ch[0]->ch[1]);
        mswap(x->ch[1]->ch[0], x->ch[1]->ch[1]);
    }
    for(int i = 0; i <= 1; i++)
    {
        x->ch[i]->v = (x->k * x->ch[i]->v % MOD + x->b) % MOD;
        x->ch[i]->sum = (x->ch[i]->sum * x->k % MOD + x->b * x->ch[i]->siz % MOD) % MOD;
        (x->ch[i]->k *= x->k) %= MOD;
        (x->ch[i]->b *= x->k) %= MOD, (x->ch[i]->b += x->b) %= MOD;
    }
```

```
    x->k = 1;  x->b = 0;
}
void update(node *x){if(!isroot(x))update(x->fa); pushdown(x);}
void rotate(node *x)
{
    node *f = x->fa; int d = type(x);
    x->fa = f->fa, !isroot(f) ? x->fa->ch[type(f)] = x : 0;
    (f->ch[d] = x->ch[d^1]) != null ? f->ch[d]->fa = f : 0;
    f->fa = x, x->ch[d^1] = f; pushup(f);
}
void splay(node *x)
{
    update(x);
    for(; !isroot(x); )
    {
        if(isroot(x->fa)) rotate(x);
        else if(type(x) == type(x->fa)) rotate(x->fa), rotate(x);
        else rotate(x),rotate(x);
    }
    pushup(x);
}
void access(node *x)
{
    node *tmp = null;
    for(; x != null; )
    {
        splay(x);
        x->ch[1] = tmp;
        pushup(x);
        tmp = x;
        x = x->fa;
    }
}
void makeroot(node *x)
{
    access(x);
    splay(x);
    x->rev ^= 1;
    swap(x->ch[0], x->ch[1]);
}
void link(node *x, node *y)
{
    makeroot(x);
    x->fa = y;
}
void cut(node *x, node *y)
{
    makeroot(x); access(y);
    splay(y); y->ch[0] = x->fa = null;
    pushup(y);
}
```

3.5 左偏树-wrz

```
struct heap
{
    heap *ch[2];
    int dis, siz, v;
}mem[N*2], *h[N], *null, *tot;
heap* newheap()
{
    heap *p = ++tot;
    *p = *null;
    return p;
}
void init()
{
    null = tot = mem;
    null->ch[0] = null->ch[1] = null;
    null->v = null->dis = null->siz = 0;
    for(int i = 1; i <= n; i++) h[i] = null;
}
heap *merge(heap *x, heap *y) // big
{
    if(x == null) return y;
    if(y == null) return x;
    if(x->v < y->v) swap(x, y);
    x->ch[1] = merge(x->ch[1], y);
    if(x->ch[0]->dis < x->ch[1]->dis) swap(x->ch[0], x->ch[1]);
    x->dis = x->ch[1]->dis + 1;
    x->siz = x->ch[0]->siz + x->ch[1]->siz + 1;
    return x;
}
heap *pop(heap *x){return merge(x->ch[0], x->ch[1]);}
int main()
{
    init();
    heap *a = newheap(); a->siz = 1; a->v = 233;
    heap *b = newheap(); b->siz = 1; b->v = 233;
    heap *c = merge(a, b);
}
```

3.6 线段树-gwx

```
void revise()
{
    //pay attention to the order of tags
}
void pushdown()
{
```

```
    calculate current node
    revise sons
}
void maintain()
{
    pushdown(sons)
    update current node
}
void modify()
{
    if this is the segment
    {
        revise tags
        return
    }
    modify_subtree
    maintain()
}
int query()
{
    pushdown()
    query_subtree
}
```

3.7 splay-gwx

```
struct node
{
    int pa, ch[2], s, f;
}a[maxn];

void flip(int u)
{
    a[u].f ^= 1;
    swap(a[u].ch[0], a[u].ch[1]);
}

void pushdown(int k)
{
    if(!k)  return;
    int l = a[k].ch[0], r = a[k].ch[1];
    if(a[k].flip)
    {
    flip(l);
    flip(r);
    a[k].flip ^= 1;
    }
}

int pre(int u)
{
    u = a[u].ch[0];
    while(a[u].ch[1])
    u = a[u].ch[1];
    return u;
}

int post(int u)
{
    u = a[u].ch[1];
    while(a[u].ch[0])
    u = a[u].ch[0];
    return u;
}

void maintain(int u)
{
    int l = a[u].ch[0], r = a[u].ch[1];
    a[u].s = a[l].s + a[r].s + 1;
}

void rotate(int u)
{
    int x = a[u].pa, y = a[x].pa, d = (a[x].ch[1] == u);
    if(!y)
    root = u;
    else
    a[y].ch[a[y].ch[1] == x] = u;
    a[x].ch[d] = a[u].ch[d ^ 1];
    a[a[u].ch[d ^ 1]].pa = x;
    a[u].ch[d ^ 1] = x;
    a[x].pa = u;
    a[u].pa = y;
    maintain(x);
    maintain(u);
}

void splay(int u, int pa)  //u的父亲为pa
{
    int t;
    for(t = u; a[t].pa != pa; t = a[t].pa)
    st[++top] = t;
    pushdown(t);
    for(; top; top--)
        pushdown(st[top]);   //pushdown the tags

    while(a[u].pa != pa)
    {
    int x = a[u].pa, y = a[x].pa;
    if(y == pa)
    {
```

```
        rotate(u);
        return;
    }
    if((a[x].ch[0] == u) ^ (a[y].ch[0] == x))
        rotate(u);
    else
        rotate(x);
    rotate(u);
    }
}

void splay2(int u, int &g)  //将u旋转到g
{
    while(u != g)
    {
    int x = a[u].pa, y = a[x].pa;
    if(x == g)
    {
        rotate(u);
        return;
    }
    if((a[x].ch[0] == u) ^ (a[y].ch[0] == x))
        rotate(u);
    else
        rotate(x);
    rotate(u);
    }
}

int find_kth(int u, int k)
{
    pushdown(u);
    int size = a[a[u].ch[0]].s;
    if(k <= size)
    return find_kth(a[u].ch[0], k);
    if(k > size + 1)
    return find_kth(a[u].ch[1], k - size - 1);
    return u;
}

int get(int l, int r)
{
    int L = find_kth(root, l), R = find_kth(root, r + 2);  //L = pre(l), R = post(r)
    splay(L, 0);  //splay2(L, root)
    splay(R, L);  //splay2(R, a[root].ch[1])
    return a[R].ch[0];
}

int new_node() //recycle
{
    int res = q.front();
    q.pop();
    a[res].init();
    return res;
}

int build(int l, int r, int pa)
{
    if(l > r)
    return 0;
    int mid = (l + r) >> 1, u = new_node();
    a[u].pa = pa;
    a[u].s = 1;
    a[u].ch[0] = build(l, mid - 1, u);
    a[u].ch[1] = build(mid + 1, r, u);
    maintain(u);
    return u;
}

void recycle(int u)
{
    q.push(u);
    if(a[u].ch[0])
    recycle(a[u].ch[0]);
    if(a[u].ch[1])
    recycle(a[u].ch[1]);
}

void del(int l, int r)
{
    int r = get(l, r);
    recycle(a[r].ch[0]);
    a[a[r].pa].ch[0] = 0;
    maintain(a[r].pa);
    maintain(root);
}
```

3.8 splay-wrz

```
struct node
{
    node *ch[2], *fa;
    ll key; int siz, tag;
}mem[N*20], *tot, *null, *root;
void init()
{
    root = null = tot = mem;
    null->ch[0] = null->ch[1] = null->fa = null;
    null->key = null->siz = null->tag = 0;
}
int type(node *x){return x->fa->ch[1]==x;}
```

```
node *newnode(ll key)
{
    node *p = ++tot; *p = *null;
    p->key = key; p->siz = 1;
    return p;
}
void pushup(node *x)
{
    x->siz = x->ch[0]->siz + x->ch[1]->siz + 1;
}
void rotate(node *x)
{
    node *f =  x->fa; int d = type(x);
    (x->fa = f->fa) != null ? x->fa->ch[type(f)] = x : 0;
    (f->ch[d] = x->ch[!d]) != null ? f->ch[d]->fa = f : 0;
    x->ch[!d] = f, f->fa = x, pushup(f);
}
void pushdown(node *x)
{
    if(x->tag)
    {
        int &tag = x->tag;
        if(x->ch[0] != null) x->ch[0]->key += tag, x->ch[0]->tag += tag;
        if(x->ch[1] != null) x->ch[1]->key += tag, x->ch[1]->tag += tag;
        tag = 0;
    }
}
void update(node *x)
{
    if(x==null) return;
    update(x->fa);
    pushdown(x);
}
void splay(node *x, node *top)
{
    update(x);
    for(;x->fa!=top;)
    {
        if(x->fa->fa == top) rotate(x);
        else if(type(x) == type(x->fa)) rotate(x->fa), rotate(x);
        else rotate(x), rotate(x);
    }
    if(top == null) root = x;
    pushup(x);
}
void insert(node *x, node *f, node *p, int d)
{
    if(x == null)
    {
        p->fa = f, f->ch[d] = p;
        return;
    }
    pushdown(x);
    if(p->key < x->key) insert(x->ch[0], x, p ,0);
    else insert(x->ch[1], x, p, 1);
    pushup(x);
}

void insert(node *p)
{
    if(root == null) root = p, p->fa = p->ch[0] = p->ch[1] = null;
    else insert(root, null, p, 0), splay(p, null);
}
node *findl(node *x){return x->ch[0]==null?x:findl(x->ch[0]);}
node *findr(node *x){return x->ch[1]==null?x:findr(x->ch[1]);}
void insertlr()
{
    insert(newnode(-INF));
    insert(newnode(INF));
}
void delet(node *p)
{
    splay(p, null);
    node *lp = findr(p->ch[0]), *rp = findl(p->ch[1]);
    if(lp == null && rp != null) root = p->ch[1], root->fa = null;
    else if(lp != null && rp == null) root = p->ch[0], root->fa = null;
    else if(lp == null && rp == null)root = null;
    else
    {
        splay(rp, null); splay(lp,rp);
        lp->ch[1] = null; splay(lp,null);
    }
}
node* findk(node *p, int k)
{
    for(; ; )
    {
        pushdown(p);
        if(p->ch[0]->siz >= k) p = p->ch[0];
        else if(p->ch[0]->siz + 1 == k) {splay(p, null); return p;}
        else k -= p->ch[0]->siz + 1, p = p->ch[1];
    }
}
node* findv(node *p, int v)
{
    node* ret = null;
    for(; p!=null; )
    {
        pushdown(p);
        if(p->key >= v) ret = p, p = p->ch[0];
        else p = p->ch[1];
```

```
    }
    splay(ret, null);
    return ret;
}
void addv(node *p, int v)
{
    if(p == null) return;
    p->key += v;
    p->tag += v;
}
```

3.9 treap-gwx

```
struct node
{
    int pri, val, c, s;    //pri: random value; c: times of showing; s: size of subtree
    int ch[2];
    int cmp(int x) const {
        if(x == val) return -1;
        return x < val ? 0 : 1;
    }
} a[maxn];

void maintain(int u)  {
    a[u].s = a[u].c + a[a[u].ch[0]].s + a[a[u].ch[1]].s;
}

void rotate(int &u, int d)
{
    int tmp = a[u].ch[d ^ 1];
    a[u].ch[d ^ 1] = a[tmp].ch[d];
    a[tmp].ch[d] = u;
    maintain(u); maintain(tmp);
    u = tmp;
}

void insert(int &u, int val)
{
    if(!u)
    {
        u = ++cnt;
        a[cnt] = (node){rand(), val, 1, 1};
        return;
    }
    a[u].s++;
    int d = a[u].cmp(val);
    if(d == -1)  {a[u].c++; return;}
    insert(a[u].ch[d], val);
    if(a[a[u].ch[d]].pri > a[u].pri) rotate(u, d ^ 1);
}

int find(int u, int val, int comp, int &res)
{
    int d = a[u].cmp(val);
    if(!u) return -1;
    if(d == -1) return u;
    if(d == comp)
    {
        if(d) res = max(res, a[u].val);
        else res = min(res, a[u].val);
    }
    return find(a[u].ch[d], val, comp, res);
}

void remove(int &u)
{
    if(!a[u].ch[0]) u = a[u].ch[1];
    else if(!a[u].ch[1]) u = a[u].ch[0];
    else
    {
        int d = a[a[u].ch[0]].pri < a[a[u].ch[1]].pri ? 0 : 1;
        rotate(u, d); remove(a[u].ch[d]);
    }
}

void del(int &u, int val)
{
    if(find(root, val, -2, val) == -1) return;
    a[u].s--;
    int d = a[u].cmp(val);
    if(d == -1)
    {
        a[u].c--;
        if(!a[u].c) remove(u);
    }
    else del(a[u].ch[d], val);
}

int find_rank(int u, int val)
{
    int d = a[u].cmp(val);
    if(d == -1) return 1 + a[a[u].ch[0]].s;
    if(d == 0) return find_rank(a[u].ch[0], val);
    return a[u].s - a[a[u].ch[1]].s + find_rank(a[u].ch[1], val);
}

int find_kth(int u, int k)
{
    if(k <= a[a[u].ch[0]].s) return find_kth(a[u].ch[0], k);
    if(k > a[a[u].ch[0]].s + a[u].c) return find_kth(a[u].ch[1], k - a[a[u].ch[0]].s - a[u].c);
    return a[u].val;
```

```
}

int pre(int val)
{
    int ans = -inf;
    int pos = find(root, val, 1, ans);
    if(pos != -1 && a[pos].ch[0])
    {
        pos = a[pos].ch[0];
        while(a[pos].ch[1]) pos = a[pos].ch[1];
        ans = max(ans, a[pos].val);
    }
    return ans;
}

int post(int val)
{
    int ans = inf;
    int pos = find(root, val, 0, ans);
    if(pos != -1 && a[pos].ch[1])
    {
        pos = a[pos].ch[1];
        while(a[pos].ch[0]) pos = a[pos].ch[0];
        ans = min(ans, a[pos].val);
    }
    return ans;
}
//srand()
```

3.10 zkw 线段树

```
//zkw-segment-tree
int n, M, q;
int d[N << 1];
inline void build(int n) {
    for(M = 1; M < n; M <<= 1);
    for(int i = M + 1; i <= M + n; i++) t[i] = in();
    //sum
    for(int i = M - 1; i; --i) d[i] = d[i << 1] + d[i << 1 | 1];
    //max
    for(int i = M - 1; i; --i) d[i] = max(d[i << 1], d[i << 1 | 1]);
    //min
    for(int i = M - 1; i; --i) d[i] = min(d[i << 1], d[i << 1 | 1]);
}

//单点修改
void change(int x, int v) {
    t[x = M + x] += v;
    while(x) d[x >>= 1] = d[x << 1] + d[x << 1 | 1];
}

//区间查询
int Sum(int s,int t,int Ans=0){
    for (s = s + M - 1, t = t + M + 1; s ^ t ^ 1; s >>= 1, t >>= 1) {
        if(~ s & 1) Ans += d[s ^ 1];
        if(t & 1) Ans += d[t ^ 1];
    }
    return Ans;
}
    void Sum(int s,int t,int L=0,int R=0){
        for(s=s+M-1,t=t+M+1;s^t^1;s>>=1,t>>=1){
            L+=d[s],R+=d[t];
            if(~s&1) L=min(L,d[s^1]);
            if(t&1) R=min(R,d[t^1]);
        }
        int res=min(L,R);while(s) res+=d[s>>=1];
    }
//单点查询
//差分, 当前点的值为该点和其父节点的差值
void build(int n) {
    for(M = 1; M <= n + 1; M <<= 1);
    for(int i = M + 1; i <= M + n; i++) d[i] = in();
    for(int i = M - 1; i; --i) {
        d[i] = min(d[i << 1], d[i << 1 | 1]),
        d[i << 1] -= d[i],
        d[i << 1 | 1] -= d[i];
    }
}

void Sum(int x, int res = 0) {
    while(x) res += d[x], x >>= 1;
    return res;
}
//区间最小(差分)
void Sum(int s, int t, int L = 0, int R = 0) {
    for(s = s + M - 1, t = t + M + 1; s ^ t ^ 1; s >>= 1, t >>= 1) {
        L += d[s], R += d[t];
        if(~ s & 1) L = min(L, d[s ^ 1]);
        if(t & 1) R = min(R, d[t ^ 1]);
    }
    int res = min(L, R);
    while(s) res += d[s >>= 1];
}
//区间加法, 维护最小值(差分)
void Add(int s, int t, int v, int A = 0) {
    for(s = s + M - 1, t = t + M + 1; s ^ t ^ 1;s >>= 1, t >>= 1) {
        if(~ s & 1) d[s ^ 1] += v;
        if(t & 1) d[t ^ 1] += v;
        A = min(d[s], d[s ^ 1]);
        d[s] -= A, d[s ^ 1] -= A, d[s >> 1] += A;
```

```
        A = min(d[t], d[t^1]);
        d[t] -= A, d[t ^ 1] -=A, d[t >> 1] += A;
    }
    while(s) {
        A = min(d[s], d[s ^ 1]),
        d[s] -= A,
        d[s ^ 1] -= A,
        d[s >>= 1] += A;
    }
}
```

4 图论

4.1 2-SAT

```
2-SAT的tarjan做法适用于一类如果A->B, 则一定有B’->A’ 的对称的图。
一个强联通分量里的所有点, 要么一起选要么一起不选, 那就缩起来。
一个重要的结论是如果一个强联通分量里同时有A和A’ , 则此图无解, 否则一定有解。
无解的情况显然正确。
有解的情况考虑构造。每次随便从点集里抓一个点A出来, 选中A的所有可达点, 删去所有可达A’ 的点。显然这是可以做到的。
那这样会不会把图弄成无解?考虑如果一个B->A, 那么选了A及其可达点, 那B选不选是不影响的。对于不可达A的显然也不影响,
    因此可以这样构造。
一个特例是存在A->A’ 的边, 那这样选A就挂了, 因此逆拓扑序 来构造才是更一般的做法。
因此构造方案只需对于任意一对点A,A', 取dfs序大的即可。
```

4.2 边双联通-gwx

```
//G[i]: 第i个边双联通分量中有哪些点
void tarjan(int u, int pa)
{
    d[u] = l[u] = ++timer;
    for(int i = tail[u]; i; i = e[i].next)
    {
        if(!d[e[i].v])
        {
            st[++top] = i;
            tarjan(e[i].v, u);
            l[u] = min(l[u], l[e[i].v]);
            if(l[e[i].v] >= d[u])
            {
                bcc++;
                while(true)
                {
                    int now = st[top--];
                    if(vst[e[now].u] != bcc)
                    {
                        vst[e[now].u] = bcc;
                        G[bcc].push_back(e[now].u);
                    }
                    if(vst[e[now].v] != bcc)
                    {
                        vst[e[now].v] = bcc;
                        G[bcc].push_back(e[now].v);
                    }
                    if(now == i) break;
                }
            }
        }
        else if(e[i].v != pa)
            l[u] = min(l[u], d[e[i].v]);
    }
}
```

4.3 带花树

```
vector<int> link[maxn];
int n,match[maxn],Queue[maxn],head,tail;
int pred[maxn],base[maxn],start,finish,newbase;
bool InQueue[maxn],InBlossom[maxn];
void push(int u){ Queue[tail++]=u;InQueue[u]=true; }
int pop(){ return Queue[head++]; }
int FindCommonAncestor(int u,int v){
    bool InPath[maxn];
    for(int i=0;i<n;i++) InPath[i]=0;
    while(true){ u=base[u];InPath[u]=true;if(u==start) break;u=pred[match[u]]; }
    while(true){ v=base[v];if(InPath[v]) break;v=pred[match[v]]; }
    return v;
}
void ResetTrace(int u){
    int v;
    while(base[u]!=newbase){
        v=match[u];
        InBlossom[base[u]]=InBlossom[base[v]]=true;
        u=pred[v];
        if(base[u]!=newbase) pred[u]=v;
    }
}
void BlossomContract(int u,int v){
    newbase=FindCommonAncestor(u,v);
    for (int i=0;i<n;i++)
    InBlossom[i]=0;
    ResetTrace(u);ResetTrace(v);
    if(base[u]!=newbase) pred[u]=v;
    if(base[v]!=newbase) pred[v]=u;
    for(int i=0;i<n;++i)
    if(InBlossom[base[i]]){
        base[i]=newbase;
        if(!InQueue[i]) push(i);
```

```
    }
}
bool FindAugmentingPath(int u){
    bool found=false;
    for(int i=0;i<n;++i) pred[i]=-1,base[i]=i;
    for (int i=0;i<n;i++) InQueue[i]=0;
    start=u;finish=-1; head=tail=0; push(start);
    while(head<tail){
        int u=pop();
        for(int i=link[u].size()-1;i>=0;i--){
            int v=link[u][i];
            if(base[u]!=base[v]&&match[u]!=v)
                if(v==start||(match[v]>=0&&pred[match[v]]>=0))
                    BlossomContract(u,v);
                else if(pred[v]==-1){
                    pred[v]=u;
                    if(match[v]>=0) push(match[v]);
                    else{ finish=v; return true; }
                }
        }
    }
    return found;
}
void AugmentPath(){
    int u=finish,v,w;
    while(u>=0){ v=pred[u];w=match[v];match[v]=u;match[u]=v;u=w; }
}
void FindMaxMatching(){
    for(int i=0;i<n;++i) match[i]=-1;
    for(int i=0;i<n;++i) if(match[i]==-1) if(FindAugmentingPath(i)) AugmentPath();
}
```

4.4 dijkstra-wrz

```
const int INF = 1 << 29; // 有时候要开longlong
int dis[N], inq[N];
struct item{int x, dis;};
bool operator < (item a, item b){return a.dis > b.dis;}
void dijk(int S) // S为源点, dis数组即到各点最短路
{
    for(int i = 1; i <= n; i++) dis[i] = INF;
    priority_queue<item> q;
    q.push((item){S, dis[S] = 0});
    for(; !q.empty(); )
    {
        int x = q.top().x; q.pop();
        if(inq[x]) continue; inq[x] = 1;
        for(int i = last[x]; i; i = e[i].next)
        {
            int y = e[i].to;
            if(dis[x] + e[i].val < dis[y])
                q.push((item){y, dis[y] = dis[x] + e[i].val});
        }
    }
}
```

4.5 支配树-gwx

```
/*
    用ins()加边
    build前设置n为点数, s为源点
    树中的i号点对应原图的id[i]号点
*/
struct Dominator_Tree {
    int n, s, cnt;
    int dfn[N], id[N], pa[N], semi[N], idom[N], p[N], mn[N];
    vector<int>e[N], dom[N], be[N];
    void ins(int x, int y) {e[x].push_back(y);}
    void dfs(int x) {
        dfn[x] = ++cnt; id[cnt] = x;
        for (int i : e[x]) {
            if (!dfn[i])dfs(i), pa[dfn[i]] = dfn[x];
            be[dfn[i]].push_back(dfn[x]);
        }
    }
    int get(int x) {
        if (p[x] != p[p[x]]) {
            if (semi[mn[x]] > semi[get(p[x])])mn[x] = get(p[x]);
            p[x] = p[p[x]];
        }
        return mn[x];
    }
    void LT() {
        for (int i = cnt; i > 1; i--) {
            for (int j : be[i])semi[i] = min(semi[i], semi[get(j)]);
            dom[semi[i]].push_back(i);
            int x = p[i] = pa[i];
            for (int j : dom[x])idom[j] = (semi[get(j)] < x ? get(j) : x);
            dom[x].clear();
        }
        for (int i = 2; i <= cnt; i++) {
            if (idom[i] != semi[i])idom[i] = idom[idom[i]];
            dom[id[idom[i]]].push_back(id[i]);
        }
    }
    void build() {
        for(int i = 1; i <= n; ++i)
            dfn[i] = 0, dom[i].clear(), be[i].clear(), p[i] = mn[i] = semi[i] = i;
        cnt = 0; dfs(s); LT();
    }
```

```
};
```

4.6 支配树-wrz

```
int dfn[N],redfn[N],fa[N],sdom[N],idom[N],fo[N],vo[N],dtimer;
vector<int> pre[N],bkt[N];
int dom_find(int x)
{
    if(fo[x]==x) return x;
    int r = dom_find(fo[x]);
    if(sdom[vo[fo[x]]]<sdom[vo[x]]) vo[x] = vo[fo[x]];
    return fo[x] = r;
}
int dom_eval(int x){dom_find(x); return vo[x];}
void dom_dfs(int x)
{
    redfn[dfn[x]=++dtimer] = x, sdom[x] = dfn[x];
    for(int i=last[x];i;i=e[i].next) if(!dfn[e[i].to])
        dom_dfs(e[i].to), fa[e[i].to] = x;
}
void dom_build(int S)
{
    int i,x;
    dom_dfs(S);
    for(i = dtimer; i >=2; i--)
    {
        x = redfn[i];
        for(int i = 0, ii = pre[x].size(); i < ii; i++)
        {
            int k = pre[x][i];
            if(dfn[k]) sdom[x] = min(sdom[x],sdom[dom_eval(k)]);
        }
        bkt[redfn[sdom[x]]].push_back(x);
        int fp = fa[x]; fo[x] = fa[x];
        for(int i = 0, ii = bkt[fp].size(); i < ii; i++)
        {
            int v = bkt[fp][i];
            int u = dom_eval(v);
            idom[v] = sdom[u]==sdom[v]?fp:u;
        }
        bkt[fp].clear();
    }
    for(int i = 2;i <= dtimer; i++) x = redfn[i], idom[x] = idom[x]==redfn[sdom[x]]?idom[x]:idom[idom[x]];
    for(int i = 2;i <= dtimer; i++) x = redfn[i], sdom[x] = redfn[sdom[x]];
}
void dom_init()
{
    dtimer = 0;
    for(int i = 1; i <= n; i++)
    {
        dfn[i] = 0;
        fo[i] = vo[i] = i;
        pre[i].clear(), bkt[i].clear();
    }
    for(int x = 1; x <= n; x++) for(int i = last[x]; i; i = e[i].next) pre[e[i].to].push_back(x);
}
/*
    步骤:
    1.建好原图
    2.dom_init() // 必须保证原图上所有的边已经连好
    3.dom_build(S) // S为支配树的根结点标号
    4.得到idom数组 // idom[x]表示x在支配树上的父结点，别的数组用处不大
*/
```

4.7 欧拉回路-wrz

```
#include<cstdio>
#define N 100005
#define M 200005
using namespace std;
int last[N], ecnt = 1, cnt, ans[M], in_deg[N], out_deg[N];
bool vis[M];
struct edge{int next,to;}e[M<<1];
void addedge(int a, int b)
{
    e[++ecnt] = (edge){last[a], b};
    last[a] = ecnt;
}
void dfs(int x)
{
    for(int &i = last[x]; i; i = e[i].next)
    {
        int y = e[i].to, j = i;
        if(!vis[j>>1])
        {
            vis[j>>1] = 1;
            dfs(y);
            ans[++cnt] = j;
        }
    }
}
int main()
{
    int t, n, m, a, b;
    scanf("%d%d%d",&t,&n,&m);
    for(int i = 1; i <= m; i++)
    {
        scanf("%d%d",&a,&b);
        addedge(a,b);
        if(t == 1)addedge(b,a), in_deg[a]++, in_deg[b]++;
```

```cpp
        else ecnt++, in_deg[b]++, out_deg[a]++;
    }

    if(t == 1) // 无向
    {
        for(int i = 1; i <= n; i++)
            if((in_deg[i]+out_deg[i]) & 1)
                return !printf("NO\n");
    }
    else // 有向
    {
        for(int i = 1; i <= n; i++)
            if(in_deg[i] != out_deg[i])
                return !printf("NO\n");
    }
    dfs(a);
    if(cnt != m)
    {
        puts("NO");
    }
    else
    {
        puts("YES");
        for(int i = cnt; i; i--)
        {
            printf("%d␣",ans[i]&1?-(ans[i]>>1):(ans[i]>>1));
        }
    }
}
```

4.8 Hopcoft-Karp

```cpp
// O(sqrt(n)m)
template <int MAXN = 100000, int MAXM = 100000>
struct hopcoft_karp {
    int mx[MAXN], my[MAXM], lv[MAXN];
    bool dfs (edge_list <MAXN, MAXM> &e, int x) {
        for (int i = e.begin[x]; ~i; i = e.next[i]) {
            int y = e.dest[i], w = my[y];
            if (!~w || (lv[x] + 1 == lv[w] && dfs (e, w))) {
                mx[x] = y; my[y] = x; return true; } }
        lv[x] = -1; return false; }
    int solve (edge_list <MAXN, MAXM> &e, int n, int m) {
        std::fill (mx, mx + n, -1); std::fill (my, my + m, -1);
        for (int ans = 0; ; ) {
            std::vector <int> q;
            for (int i = 0; i < n; ++i)
                if (mx[i] == -1) {
                    lv[i] = 0; q.push_back (i);
                } else lv[i] = -1;
            for (int head = 0; head < (int) q.size(); ++head) {
                int x = q[head];
                for (int i = e.begin[x]; ~i; i = e.next[i]) {
                    int y = e.dest[i], w = my[y];
                    if (~w && lv[w] < 0) { lv[w] = lv[x] + 1; q.push_back (w); } } }
            int d = 0; for (int i = 0; i < n; ++i) if (!~mx[i] && dfs (e, i)) ++d;
            if (d == 0) return ans; else ans += d; } } };
```

4.9 匈牙利算法-wrz

```cpp
bool match(int x)
{
    for(int i = last[x]; i; i = e[i].next)
    {
        int y = e[i].to;
        if(vis[y])continue;
        vis[y] = 1;
        if(!mat[y] || match(mat[y]))
        {
            mat[y] = x;
            return 1;
        }
    }
    return 0;
}
void check()
{
    for(int i = 1; i <= n; i++)
    {
        memset(vis,0,sizeof(vis));
        if(match(i))ans++;
    }
}
```

4.10 KM-gwx

```cpp
//最大匹配
bool find(int k)
{
    if(k == 0)
        return 0;
    px[k] = 1;
    for(int i = 1; i <= m; i++)
        if(!py[i])
        {
            int t = x[k] + y[i] - a[k][i];
            if(!t)
            {
                py[i] = 1;
```

```cpp
                if(!match[i] || find(match[i]) == 1)
                {
                    match[i] = k;
                    return 1;
                }
            }
            else
                slack[i] = min(slack[i], t);
        }
    return 0;
}

void revise()
{
    int d = inf;
    for(int i = 1; i <= m; i++)
        if(!py[i])
            d = min(d, slack[i]);
    for(int i = 1; i <= n; i++)
        if(px[i])
            x[i] -= d;
    for(int i = 1; i <= m; i++)
        if(py[i])
            y[i] += d;
        else
            slack[i] -= d;
}

void solve()
{
    for(int i = 1; i <= n; i++)
    {
        for(int j = 1; j <= m; j++)
            slack[j] = inf;
        while(true)
        {
            memset(px, 0, sizeof(px));
            memset(py, 0, sizeof(py));
            if(find(i) == 1)
                break;
            revise();
        }
    }
    for(int i = 1; i <= m; i++)
        if(match[i])
            f[match[i]] = i;
}
```

4.11 KM-truly-n3

```cpp
struct KM {
    // Truly O(n^3)
    // 邻接矩阵, 不能连的边设为 -INF, 求最小权匹配时边权取负, 但不能连的还是 -INF, 使用时先对 1 -> n 调用 hungary
        () , 再 get_ans() 求值
    int w[N][N];
    int lx[N], ly[N], match[N], way[N], slack[N];
    bool used[N];
    void init() {
        for (int i = 1; i <= n; i++) {
            match[i] = 0;
            lx[i] = 0;
            ly[i] = 0;
            way[i] = 0;
        }
    }
    void hungary(int x) {
        match[0] = x;
        int j0 = 0;
        for (int j = 0; j <= n; j++) {
            slack[j] = INF;
            used[j] = false;
        }

        do {
            used[j0] = true;
            int i0 = match[j0], delta = INF, j1 = 0;
            for (int j = 1; j <= n; j++) {
                if (used[j] == false) {
                    int cur = -w[i0][j] - lx[i0] - ly[j];
                    if (cur < slack[j]) {
                        slack[j] = cur;
                        way[j] = j0;
                    }
                    if (slack[j] < delta) {
                        delta = slack[j];
                        j1 = j;
                    }
                }
            }
            for (int j = 0; j <= n; j++) {
                if (used[j]) {
                    lx[match[j]] += delta;
                    ly[j] -= delta;
                }
                else slack[j] -= delta;
            }
            j0 = j1;
        } while (match[j0] != 0);

        do {
            int j1 = way[j0];
```

```
            match[j0] = match[j1];
            j0 = j1;
        } while (j0);
    }

    int get_ans() {
        int sum = 0;
        for(int i = 1; i <= n; i++) {
            if (w[match[i]][i] == -INF) ; // 无解
            if (match[i] > 0) sum += w[match[i]][i];
        }
        return sum;
    }
} km;
```

4.12 k 短路 a 星-gwx

```
const int maxn = 1005;
int n, m;
int S, T, K;
int dist[maxn], cnt[maxn];
bool vst[maxn];
vector<pair<int, int>> G[maxn], H[maxn];    //正图&反图
struct node
{
    ll d;
    int id;
    node(){}
    node(ll d, int id): d(d), id(id) {}
    bool operator< (const node &other) const{
        return d + dist[id] > other.d + dist[other.id];
    }
};

priority_queue <pair<ll, int>> q;
priority_queue <node> Q;

void init()
{
    for(int i = 1; i <= n; ++i)
        G[i].clear(), H[i].clear(), cnt[i] = 0;
}

void dijkstra(int S)
{
    memset(dist, 127, sizeof(dist));
    memset(vst, 0, sizeof(vst));
    while(!q.empty()) q.pop();
    dist[S] = 0;
    q.push(make_pair(0, S));
    for(int i = 1; i <= n; ++i)
    {
        if(q.empty()) break;
        while(vst[q.top().second]) q.pop();
        int u = q.top().second; q.pop();
        vst[u] = 1;
        for(auto i: H[u])
        {
            if(dist[i.first] > dist[u] + i.second)
            {
                dist[i.first] = dist[u] + i.second;
                q.push(make_pair(-dist[i.first], i.first));
            }
        }
    }
}

int solve()
{
    while(!Q.empty()) Q.pop();
    Q.push(node(0, S));
    while(!Q.empty())
    {
        auto u = Q.top(); Q.pop();
        if(++cnt[u.id] > K) continue;
        if(u.d + dist[u.id] > ti) continue;
        if(u.id == T && cnt[T] == K)
            return u.d;
        for(auto i: G[u.id])
            Q.push(node(u.d + i.second, i.first));
    }
    return -1;
}
```

4.13 K 短路可并堆

```
//Kth Shortest Path via Persistable Mergeable Heap
//可持久化可并堆求k短路 O(SSSP+(m+k)\log n)
//By ysf
//通过题目: USACO Mar08 牛跑步 (板子题)

//注意这是个多项式算法, 在k比较大时很有优势, 但k比较小时最好还是用A*
//DAG和有环的情况都可以, 有重边或自环也无所谓, 但不能有零环
//以下代码以Dijkstra+可持久化左偏树为例

const int maxn=1005,maxe=10005,maxm=maxe*30;//点数, 边数, 左偏树结点数

//需要用到的结构体定义
struct A{//用来求最短路
```

```
    int x,d;
    A(int x,int d):x(x),d(d){}
    bool operator<(const A &a)const{return d>a.d;}
};

struct node{//左偏树结点
    int w,i,d;//i: 最后一条边的编号 d: 左偏树附加信息
    node *lc,*rc;
    node(){}
    node(int w,int i):w(w),i(i),d(0){}
    void refresh(){d=rc->d+1;}
}null[maxm],*ptr=null,*root[maxn];

struct B{//维护答案用
    int x,w;//x是结点编号, w表示之前已经产生的权值
    node *rt;//这个答案对应的堆顶, 注意可能不等于任何一个结点的堆
    B(int x,node *rt,int w):x(x),w(w),rt(rt){}
    bool operator<(const B &a)const{return w+rt->w>a.w+a.rt->w;}
};

//全局变量和数组定义
vector<int>G[maxn],W[maxn],id[maxn];//最开始要存反向图, 然后把G清空作为儿子列表
bool vis[maxn],used[maxe];//used表示边是否在最短路树上
int u[maxe],v[maxe],w[maxe];//存下每条边, 注意是有向边
int d[maxn],p[maxn];//p表示最短路树上每个点的父边
int n,m,k,s,t;//s,t分别表示起点和终点

//以下是主函数中较关键的部分
for(int i=0;i<=n;i++)root[i]=null;//一定要加上! ! !
//(读入&建反向图)
Dijkstra();
//(清空G,W,id)
for(int i=1;i<=n;i++)
    if(p[i]){
        used[p[i]]=true;//在最短路树上
        G[v[p[i]]].push_back(i);
    }
for(int i=1;i<=m;i++){
    w[i]-=d[u[i]]-d[v[i]];//现在的w[i]表示这条边能使路径长度增加多少
    if(!used[i])
        root[u[i]]=merge(root[u[i]],newnode(w[i],i));
}
dfs(t);
priority_queue<B>heap;
heap.push(B(s,root[s],0));//初始状态是找贡献最小的边加进去
printf("%d\n",d[s]);//第1短路需要特判
while(--k){//其余k-1短路径用二叉堆维护
    if(heap.empty())printf("-1\n");
    else{
        int x=heap.top().x,w=heap.top().w;
        node *rt=heap.top().rt;
        heap.pop();
        printf("%d\n",d[s]+w+rt->w);
        if(rt->lc!=null||rt->rc!=null)
            heap.push(B(x,merge(rt->lc,rt->rc),w));//pop掉当前边, 换成另一条贡献大一点的边
        if(root[v[rt->i]]!=null)
            heap.push(B(v[rt->i],root[v[rt->i]],w+rt->w));//保留当前边, 往后面再接上另一条边
    }
}
//主函数到此结束

//Dijkstra预处理最短路 O(m\log n)
void Dijkstra(){
    memset(d,63,sizeof(d));
    d[t]=0;
    priority_queue<A>heap;
    heap.push(A(t,0));
    while(!heap.empty()){
        int x=heap.top().x;
        heap.pop();
        if(vis[x])continue;
        vis[x]=true;
        for(int i=0;i<(int)G[x].size();i++)
            if(!vis[G[x][i]]&&d[G[x][i]]>d[x]+W[x][i]){
                d[G[x][i]]=d[x]+W[x][i];
                p[G[x][i]]=id[x][i];
                heap.push(A(G[x][i],d[G[x][i]]));
            }
    }
}

//dfs求出每个点的堆 总计O(m\log n)
//需要调用merge, 同时递归调用自身
void dfs(int x){
    root[x]=merge(root[x],root[v[p[x]]]);
    for(int i=0;i<(int)G[x].size();i++)
        dfs(G[x][i]);
}

//包装过的new node() O(1)
node *newnode(int w,int i){
    *++ptr=node(w,i);
    ptr->lc=ptr->rc=null;
    return ptr;
}

//带可持久化的左偏树合并 总计O(\log n)
//递归调用自身
```

```
node *merge(node *x,node *y){
    if(x==null)return y;
    if(y==null)return x;
    if(x->w>y->w)swap(x,y);
    node *z=newnode(x->w,x->i);
    z->lc=x->lc;
    z->rc=merge(x->rc,y);
    if(z->lc->d>z->rc->d)swap(z->lc,z->rc);
    z->refresh();
    return z;
}
```

4.14 最大团

```
/*
Int g[][]为图的邻接矩阵。
    MC(V)表示点集V的最大团
    令Si={vi, vi+1, ..., vn}, mc[i]表示MC(Si)
    倒着算mc[i], 那么显然MC(V)=mc[1]
    此外有mc[i]=mc[i+1] or mc[i]=mc[i+1]+1
*/
void init(){
    int i, j;
    for (i=1; i<=n; ++i) for (j=1; j<=n; ++j) scanf("%d", &g[i][j]);
}
void dfs(int size){
    int i, j, k;
    if (len[size]==0) {
        if (size>ans) {
            ans=size; found=true;
        }
        return;
    }
    for (k=0; k<len[size] && !found; ++k) {
        if (size+len[size]-k<=ans) break;
        i=list[size][k];
        if (size+mc[i]<=ans) break;
        for (j=k+1, len[size+1]=0; j<len[size]; ++j)
        if (g[i][list[size][j]]) list[size+1][len[size+1]++]=list[size][j];
        dfs(size+1);
    }
}
void work(){
    int i, j;
    mc[n]=ans=1;
    for (i=n-1; i; --i) {
        found=false;
        len[1]=0;
        for (j=i+1; j<=n; ++j) if (g[i][j]) list[1][len[1]++]=j;
        dfs(1);
        mc[i]=ans;
    }
}
```

4.15 SAP 网络流

```
#include<bits/stdc++.h>
typedef long long ll;
using std::min;

void read(int &digit)
{
    digit=0;
    char c;
    for (c=getchar();(c<'0' || c>'9') && c!='-';c=getchar());
    bool type=false;
    if (c=='-')
        type=true,c=getchar();
    for (;c>='0' && c<='9';digit=digit*10+c-'0',c=getchar());
    if (type==true)
        digit=-digit;
}

#define maxn 1010
const int INF=1<<30;
int n,m;
int S,T;
struct Edge
{
    int v,flow,next;
}   e[510010];
int g[maxn],tot=1;//tot初值必须赋为1
void addedge(int x,int y,int flow)
{
    e[++tot].v=y;e[tot].flow=flow;e[tot].next=g[x];g[x]=tot;
    e[++tot].v=x;e[tot].flow=0;e[tot].next=g[y];g[y]=tot;
}
int w[maxn],hash[maxn],d[maxn];
int que[maxn],pre1[maxn],pre2[maxn],p[maxn];
bool vis[maxn];
int maxflow()
{
    for (int i=1;i<=n;i++)  hash[i]=0,d[i]=0,vis[i]=false;
    for (int i=1;i<=n;i++)  p[i]=g[i];
    //hash[0]=n;
    int l,r;
    l=r=1;
    que[1]=T;hash[0]=1;vis[T]=true;
    while (l<=r)
    {
```

```
        int u=que[l++];
        for (int i=g[u];i;i=e[i].next)
        if ((i&1) && !vis[e[i].v])
        {
            que[++r]=e[i].v;
            vis[e[i].v]=true;
            d[e[i].v]=d[u]+1;
            hash[d[e[i].v]]++;
        }
    }
    for (int i=1;i<=n;i++)
    if (!vis[i])    d[i]=n,hash[n]++;
    int flow=INF;
    int ans=0;
    int u=S;
    while (d[S]<n)
    {
        w[u]=flow;
        bool bo=true;
        for (int i=p[u];i;i=e[i].next)
        if (e[i].flow && d[e[i].v]==d[u]-1)
        {
            flow=min(flow,e[i].flow);
            p[u]=i;
            pre1[e[i].v]=u;
            pre2[e[i].v]=i;
            u=e[i].v;
            bo=false;
            if (u==T)
            {
                ans+=flow;
                while (u!=S)
                {
                    e[pre2[u]].flow-=flow;
                    e[pre2[u]^1].flow+=flow;
                    u=pre1[u];
                }
                flow=INF;
            }
            break;
        }
        if (!bo)    continue;
        int minx=n,pos=0;
        for (int i=g[u];i;i=e[i].next)
        if (e[i].flow && d[e[i].v]<minx)    minx=d[e[i].v],pos=i;
        p[u]=pos;
        hash[d[u]]--;
        if (hash[d[u]]==0)  break;
        d[u]=minx+1;
        hash[d[u]]++;
        if (u!=S)   u=pre1[u],flow=w[u];
    }
    return ans;
}
int main()
{
    int n1,n2;
    read(n1),read(n2),read(m);
    n=n1+n2+2;
    S=n1+n2+1,T=n1+n2+2;
    tot=1;
    for (int i=1;i<=n1;i++) addedge(S,i,1);
    for (int i=1;i<=n2;i++) addedge(i+n1,T,1);
    while (m--)
    {
        int x,y;
        read(x),read(y);
        addedge(x,y+n1,1);
    }
    int mjy=maxflow();
    printf("%d\n",mjy);
    for (int i=1;i<=n1;i++)
    {
        bool bo=true;
        for (int j=g[i];j;j=e[j].next)
            if (!(j&1) && e[j].flow==0) {bo=false;printf("%d␣",e[j].v-n1);break;}
        if (bo) printf("0␣");
    }
    printf("\n");
    return 0;
}

//求割的方案：从S开始，沿着非满流边bfs，能遍历到的地方为集合SS，其余为集合TT，横跨两个集合的边为割边
```

4.16 最短路-gwx

```
void spfa()
{
    queue <int> q;
    memset(d, 127, sizeof(d));
    d[s] = 0;
    vst[s] = 1;
    q.push(s);
    while(!q.empty())
    {
    int u = q.front();
    q.pop();
    vst[u] = 0;
    for(int i = tail[u]; i; i = a[i].next)
        if(d[a[i].v] > d[u] + a[i].c)
        {
```

```
        d[a[i].v] = d[u] + a[i].c;
        if(!vst[a[i].v])
        {
            vst[a[i].v] = 1;
            q.push(a[i].v);
        }
        }
    }
}

void dijkstra()
{
    memset(d, 127, sizeof(d));
    d[s] = 0;
    for(int i = 1; i <= n; i++)
    {
    int dis = inf, u;
    for(int j = 1; j <= n; j++)
        if(!vst[j] && d[j] < dis)
        {
        dis = d[j];
        pos = j;
        }
    vst[pos] = 1;
    for(int i = tail[u]; i; i = a[i].next)
        if(!vst[a[i].v])
        d[a[i].v] = min(d[a[i].v], d[u] + a[i].c);
    }
}

void floyd()
{
    for(int i = 1; i <= n; i++)
    for(int j = 1; j <= n; j++)
        if(i != j)
        for(int k = 1; k <= n; k++)
            if(i != k && j != k)
            d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
}
```

4.17 SPFA 判负环-wrz

```
int inq[N], inqt[N], dis[N];
bool SPFA()
{
    queue<int> q;
    for(int i = 1; i <= n; i++) dis[i] = 0, q.push(i), inq[i] = 1; // 全部入队
    for(; !q.empty(); )
    {
        int x = q.front(); q.pop(); inq[x] = 0;
        for(int i = last[x]; i; i = e[i].next)
        {
            int y = e[i].to;
            if(dis[x] + e[i].val < dis[y])
            {
                dis[y] = dis[x] + e[i].val;
                if(!inq[y])
                {
                    if(++inqt[y] > n) return false; // 入队n次即有负环
                    inq[y] = 1;
                    q.push(y);
                }
            }
        }
    }
    return true;
}
/*
    步骤:
    1.建好原图
    2.SPFA() // 若返回为true表示无负环, false表示有负环

    多次调用时记得清空inqt等数组
    有负环时理论复杂度是O(n^2)的
*/
```

4.18 spfa 费用流-gwx

```
void mcf()
{
    while(true)
    {
        queue <int> q;
        memset(d, 127, sizeof(d));
        d[s] = 0;
        vst[s] = 1;
        q.push(s);
        while(!q.empty())
        {
            int u = q.front();
            q.pop();
            vst[u] = 0;
            for(int i = tail[u]; i; i = a[i].next)
                if(a[i].c > 0 && d[a[i].v] > d[u] + a[i].s)
                {
                    d[a[i].v] = d[u] + a[i].s;
                    pre[a[i].v] = u;
                    pe[a[i].v] = i;
                    if(!vst[a[i].v])
```

```
                    {
                        vst[a[i].v] = 1;
                        q.push(a[i].v);
                    }
                }
        }
        if(d[t] > inf)
            break;
        int f = inf;
        for(int i = t; i != 1; i = pre[i])
            f = min(f, a[pe[i]].c);
        flow += f;
        for(int i = t; i != 1; i = pre[i])
        {
            cost += f * a[pe[i]].s;
            a[pe[i]].c -= f;
            a[pe[i] ^ 1].c += f;
        }
    }
}

bool bfs()
{
    queue <int> q;
    memset(d, -1, sizeof(d));
    d[s] = 0;
    q.push(s);
    while(!q.empty())
    {
        int u = q.front();
        q.pop();
        for(int i = tail[u]; i; i = a[i].next)
            if(d[a[i].v] == -1 && a[i].c > 0)
            {
                d[a[i].v] = d[u] + 1;
                q.push(a[i].v);
            }
    }
    return d[t] != -1;
}

int dfs(int u, int flow)
{
    int used = 0;
    if(u == t)
        return flow;
    for(int &i = cur[u]; i; i = a[i].next)
        if(d[a[i].v] == d[u] + 1 && a[i].c > 0)
        {
            int w = dfs(a[i].v, min(a[i].c, flow - used));
            a[i].c -= w;
            a[i ^ 1].c += w;
            used += w;
            if(used == flow)
                return flow;
        }
    if(!used)
        d[u] = -1;
    return used;
}

int dinic()
{
    int res = 0;
    while(bfs())
    {
        memcpy(cur, tail, sizeof(tail));
        res += dfs(s, inf);
    }
    return res;
}
```

4.19 斯坦纳树

```
//Nµ    , M±    , P¹  µ
const int inf = 0x3f3f3f3f;
int n, m, p, status, idx[P], f[1 << P][N];
priority_queue<pair<int, int> > q; //int top, h[N];
void dijkstra(int dis[]) {}
void Steiner_Tree() {
    for (int i = 1; i < status; i++) {
        while (!q.empty()) q.pop(); //top = 0;
        memset(vis, 0, sizeof(vis));
        for (int j = 1; j <= n; j++) {
            for (int k = i & (i - 1); k; (--k) &= i)
                f[i][j] = min(f[i][j], f[k][j] + f[i ^ k][j]);
            if (f[i][j] != inf)
                q.push(make_pair(-f[i][j], j)); //h[++top] = j, vis[j] = 1;
        }
        dijkstra(f[i]); //SPFA(f[i]);
    }
}
int main() {
    scanf("%d%d%d", &n, &m, &p);
    status = 1 << p;
    tot = 0; memset(lst, 0, sizeof(lst));
    /*        ³
      ÿ¿    ³                  ¸ F µ  I´  ¼
       ¿ª ¸       µ  0   I
      Add(0, i, val[i]); Add(i, 0, val[i]);*/
    for (int i = 1; i <= p; i++) scanf("%d", &idx[i]);
```

```
    memset(f, 0x3f, sizeof(f));
    for (int i = 1; i <= n; i++) f[0][i] = 0;
    for (int i = 1; i <= p; i++) f[1 << (i - 1)][idx[i]] = 0;
    Steiner_Tree();
    int ans = inf;
    for (int i = 1; i <= n; i++) ans = min(ans, f[status - 1][i]);
}
```

4.20 stoer-wagner 无向图最小割树

```
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);
}
```

4.21 tarjan-gwx

```
//cut[i]: i是否为割点
//bridge[i]: e[i]是否为桥
void dfs(int u, int pa)
{
    d[u] = l[u] = ++timer;
    int child = 0;
    for(int i = tail[u]; i; i = e[i].next)
        if(!d[e[i].v])
        {
            child++;
            dfs(e[i].v, u);
            l[u] = min(l[u], l[e[i].v]);
            if(l[e[i].v] >= d[u])
            {
                cut[u] = 1;
                if(l[e[i].v] > d[u])
                    bridge[i] = 1;
            }
        }
        else if(vst[e[i].v]) l[u] = min(l[u], d[e[i].v]);
    if(!pa && child < 2) cut[u] = 0;
    if(l[u] == d[u])
    {
        int v; scc++;
        while(true)
        {
            v = st.top(); st.pop();
            id[v] = scc; vst[v] = 0; size[scc]++;
            if(u == v) break;
        }
    }
}
```

4.22 tarjan-wrz

```
void tarjan(int x) // 找割点
{
    low[x] = dfn[x] = ++timer;
    int siz = 0;
    for(int i = last[x]; i; i = e[i].next)
    {
        int y = e[i].to;
        if(!dfn[y])
        {
            tarjan(y); siz++;
            cmin(low[x], low[y]);
            if(x != 1 && low[y] >= dfn[x]) cut[x] = 1;
        }
        else cmin(low[x], dfn[y]);
    }
    if(x == 1 && siz > 1) cut[1] = 1;
```

```
}

void tarjan(int x) // 有向图 缩
{
    dfn[x] = low[x] = ++timer; sta[++stacnt] = x; insta[x] = 1;
    for(int i = last[x]; i; i = e[i].next)
    {
        int y = e[i].to;
        if(!dfn[y]) tarjan(y), low[x] = min(low[x], low[y]); // 根据不同需求适当修改
        else if(insta[y])low[x] = min(low[x], dfn[y]);
    }
    if(low[x] == dfn[x])
    {
        bel[x] = ++bcnt; insta[x] = 0;
        for(; sta[stacnt] != x; stacnt--)
            bel[sta[stacnt]] = bcnt, insta[sta[stacnt]] = 0;
        stacnt--;
    }
}
```

4.23 朱刘算法-gwx

```
//时间复杂度: O(nm)
int N, m;
int pre[maxn], in[maxn], f[maxn], id[maxn];
struct node {int u, v, w;} a[maxm * 2]; //边表

int find(int x)
{
    return f[x] == x ? x : f[x] = find(f[x]);
}

int mst()
{
    long long res = 0;
    int root = 1;
    int n = N;
    while(true)
    {
        for(int i = 1; i <= n; i++) in[i] = INT_MAX, pre[i] = 0;
        for(int i = 1; i <= m; i++)
            if(a[i].u != a[i].v  && in[a[i].v] > a[i].w)
                in[a[i].v] = a[i].w, pre[a[i].v] = a[i].u;
        for(int i = 1; i <= n; i++)
            if(in[i] == INT_MAX && i != root) return 0;
        int cnt = 0;
        for(int i = 1; i <= n; i++) f[i] = i, id[i] = 0;
        for(int i = 1; i <= n; i++)
        {
            if(i == root) continue;
            res += in[i];
            if(find(i) != find(pre[i])) f[f[i]] = f[pre[i]];
            else
            {
                cnt++;
                for(int j = i; j && !id[j]; j = pre[j])
                    id[j] = cnt;
            }
        }
        if(!cnt) break;
        for(int i = 1; i <= n; i++)
            if(!id[i]) id[i] = ++cnt;
        for(int i = 1; i <= m; i++)
        {
            if(id[a[i].u] != id[a[i].v]) a[i].w -= in[a[i].v];
            a[i].u = id[a[i].u];
            a[i].v = id[a[i].v];
        }
        n = cnt;
        root = id[root];
    }
    return res;
}
```

4.24 zkw 费用流

```
//稠密图、二分图中较快, 稀疏图中不如SPFA
int flow, cost, price;

int dfs(int u, int f)
{
    if(u == t)
    {
        flow += f;
        cost += price * f;
        return f;
    }
    vst[u] = 1;
    int used = 0;
    for(int i = tail[u]; i; i = e[i].next)
        if(!vst[e[i].v] && e[i].c > 0 && e[i].w == 0)
        {
            int w = dfs(e[i].v, min(e[i].c, f - used));
            e[i].c -= w; e[i ^ 1].c += w; used += w;
            if(used == f) return f;
        }
    return used;
}
bool modlabel()
{
```

```
    int d = inf;
    for(int u = s; u <= t; u++)
        if(vst[u])
            for(int i = tail[u]; i; i = e[i].next)
                if(e[i].c > 0 && !vst[e[i].v]) d = min(d, e[i].w);
    if(d == inf) return 0;
    for(int u = s; u <= t; u++)
        if(vst[u])
            for(int i = tail[u]; i; i = e[i].next)
                e[i].w -= d, e[i ^ 1].w += d;
    price += d;
    return 1;
}
void zkw()
{
    do
        do memset(vst, 0, sizeof(vst));
        while(dfs(s, inf) > 0);
    while(modlabel());
}
```

5 数论

5.1 杜教筛

```
#define N 1000005 // (10^9)^(2/3)
#define M 3333331 // hash siz
int prime[N], notprime[N], pcnt, mu[N], pre[N];
int hash[M], nocnt; struct node{int id, f, next;}no[1000000];
int F(int n) // calculate mu[1]+mu[2]+...+mu[n]
{
    if(n<N) return pre[n];
    int h = n%M; for(int i = hash[h]; i; i = no[i].next) if(no[i].id == n) return no[i].f;
    int ret = 1;
    for(int i = 2, j; i <= n; i = j + 1)
    {
        j = n/(n/i);
        ret -= F(n/i) * (j-i+1);
    }
    no[++nocnt] = (node){n, ret, hash[h]};
    hash[h] = nocnt;
    return ret;
}
void init()
{
    mu[1] = 1;
    for(int i = 2; i < N; i++)
    {
        if(!notprime[i]) prime[++pcnt] = i, mu[i] = -1;
        for(int j = 1; j <= pcnt && prime[j] * i < N; j++)
        {
            notprime[prime[j] * i] = 1;
            if(i % prime[j]) mu[prime[j] * i] = -mu[i];
            else {mu[prime[j] * i] = 0; break;}
        }
    }
    for(int i = 1; i < N; i++) pre[i] = pre[i-1] + mu[i];
}

/*
    用之前必须先init()
    如果n很大，求和记得开long long
    如果有取模，求和记得改取模
*/
```

5.2 求逆元

```
void ex_gcd(long long a, long long b, long long &x, long long &y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return;
    }
    long long xx, yy;
    ex_gcd(b, a % b, xx, yy);
    y = xx - a / b * yy;
    x = yy;
}

long long inv(long long x, long long MODN) {
    long long inv_x, y;
    ex_gcd(x, MODN, inv_x, y);
    return (inv_x % MODN + MODN) % MODN;
}
```

5.3 莫比乌斯-gwx

```
void init()
{
    mu[1] = 1;
    for(int i = 2; i <= lim; i++)
    {
        if(!p[i])
        {
            prime[++cnt] = i;
            mu[i] = -1;
        }
        for(int j = 1; j <= cnt && i * prime[j] <= lim; j++)
        {
```

```
            p[i * prime[j]] = 1;
            if(i % prime[j] == 0)
            {
                mu[i * prime[j]] = 0;
                break;
            }
            else
                mu[i * prime[j]] = -mu[i];
        }
    }
}
//gwx
    for(int i = 2; i <= N; i++)
    {
        if(!phi[i])
        {
            pri[++tot]=i;
            phi[i] = i-1;
        }
        for(int j = 1; j <= tot && i * pri[j] <= N; j++)
        if(i % pri[j] == 0)
        {
            phi[i * pri[j]] = phi[i] * pri[j];
            break;
        }
        else phi[i * pri[j]] = phi[i] * (pri[j] - 1);
    }

//莫比乌斯反演+分块
void init()
{
    mu[1] = 1;
    for(int i = 2; i <= up; i++)
    {
        if(!p[i]) prime[++cnt] = i, mu[i] = -1;
        for(int j = 1; j <= cnt && i * prime[j] <= up; j++)
        {
            p[i * prime[j]] = 1;
            if(i % prime[j] == 0)  {mu[i * prime[j]] = 0; break;}
            else mu[i * prime[j]] = -mu[i];
        }
    }
}

void solve()
{
    for(int i = 1, j = 1; i <= n; i = j + 1)
    {
        j = min(m / (m / i), n / (n / i));
        ans += (ll)(mu[j] - mu[i - 1]) * d[n / i] * d[m / i];
    }
}
```

5.4 直线下整点

```
// $\sum_{i=0}^{n-1} \lfloor \frac{a+bi}{m}\rfloor$, $n,m,a,b>0$
LL solve(LL n,LL a,LL b,LL m){
    if(b==0) return n*(a/m);
    if(a>=m) return n*(a/m)+solve(n,a%m,b,m);
    if(b>=m) return (n-1)*n/2*(b/m)+solve(n,a,b%m,m);
    return solve((a+b*n)/m,(a+b*n)%m,m,b);
}
```

5.5 拉格朗日插值

```
#define MOD 1000000007
int inv[N], invf[N], f[N];
int fpow(int a, int b)
{
    int r = 1;
    for(; b; b >>= 1)
    {
        if(b & 1) r = 1ll*r*a%MOD;
        a = 1ll*a*a%MOD;
    }
    return r;
}
int la(int x, int k) // k次, 求f(x)
{
    int lim = k+2, ff = 1;
    for(int i = 1; i <= lim; i++)
        ff = 1ll * ff * (x-i) % MOD;
    for(int i = 1; i <= lim; i++)
        f[i] = (f[i-1] + fpow(i, k)) % MOD; // 预处理 f(1),f(2),...,f(lim), 注意修改
    if(x <= lim) return f[x];
    int ret = 0;
    for(int i = 1; i <= lim; i++)
    {
        (ret += 1ll * f[i]
                * ff % MOD * (x-i < N ? inv[x-i] : fpow(x-i, MOD-2)) % MOD // 复杂度
                * invf[i-1] % MOD * invf[lim-i] % MOD * ((lim-i) % 2 ? MOD-1 : 1) % MOD
        ) %= MOD;
    }
    return ret;
}
void init()
{
    inv[1] = 1;
    for(int i = 2; i < N; i++) inv[i] = 1ll * (MOD - MOD / i) * inv[MOD % i] % MOD;
    invf[0] = 1;
```

```
    for(int i = 1; i < N; i++) invf[i] = 1ll * invf[i-1] * inv[i] % MOD;
}
/*
    用之前必须先init()
    如果所有的逆元都能预处理就是O(n)的，否则是O(nlogn)的
*/
```

5.6 线性回归

```
// O(m^2logn)
// Given a[0], a[1], ..., a[m - 1]
// a[n] = c[0] * a[n - m] + ... + c[m - 1] * a[n - 1]
// Solve for a[n] = v[0] * a[0] + v[1] * a[1] + ... + v[m - 1] * a[m - 1]

void linear_recurrence(long long n, int m, int a[], int c[], int p) {
    long long v[M] = {1 % p}, u[M << 1], msk = !!n;
    for(long long i(n); i > 1; i >>= 1) {
        msk <<= 1;
    }
    for(long long x(0); msk; msk >>= 1, x <<= 1) {
        fill_n(u, m << 1, 0);
        int b(!!(n & msk));
        x |= b;
        if(x < m) {
            u[x] = 1 % p;
        }else {
            for(int i(0); i < m; i++) {
                for(int j(0), t(i + b); j < m; j++, t++) {
                    u[t] = (u[t] + v[i] * v[j]) % p;
                }
            }
            for(int i((m << 1) - 1); i >= m; i--) {
                for(int j(0), t(i - m); j < m; j++, t++) {
                    u[t] = (u[t] + c[j] * u[i]) % p;
                }
            }
        }
        copy(u, u + m, v);
    }
    //a[n] = v[0] * a[0] + v[1] * a[1] + ... + v[m - 1] * a[m - 1].
    for(int i(m); i < 2 * m; i++) {
        a[i] = 0;
        for(int j(0); j < m; j++) {
            a[i] = (a[i] + (long long)c[j] * a[i + j - m]) % p;
        }
    }
    for(int j(0); j < m; j++) {
        b[j] = 0;
        for(int i(0); i < m; i++) {
            b[j] = (b[j] + v[i] * a[i + j]) % p;
        }
    }
    for(int j(0); j < m; j++) {
        a[j] = b[j];
    }
}
```

5.7 素数测试-gwx

```
ll multi(ll x, ll y, ll M) {
    ll res = 0;
    for(; y; y >>= 1, x = (x + x) % M)
        if(y & 1) res = (res + x) % M;
    return res;
}
ll power(ll x, ll y, ll p)
{
    ll res = 1;
    for(; y; y >>= 1, x = multi(x, x, p))
        if(y & 1) res = multi(res, x, p);
    return res;
}
int primetest(ll n, int base)
{
    ll n2 = n - 1, res;
    int s = 0;
    while(!(n2 & 1)) n2 >>= 1, s++;
    res = power(base, n2, n);
    if(res == 1 || res == n - 1) return 1;
    s--;
    while(s >= 0)
    {
        res = multi(res, res, n);
        if(res == n - 1) return 1;
        s--;
    }
    return 0;   // n is not a strong pseudo prime
}
int isprime(ll n)
{
    static ll testNum[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
    static ll lim[] = {4, 0, 1373653ll, 25326001ll, 25000000000ll, 2152302898747ll, 3474749660383ll,
        341550071728321ll, 0, 0, 0, 0};
    if(n < 2 || n == 3215031751ll) return 0;
    for(int i = 0; i < 12; i++)
    {
        if(n < lim[i]) return 1;
        if(!primetest(n, testNum[i])) return 0;
    }
    return 1;
```

```
}
ll pollard(ll n)
{
    ll i, x, y, p;
    if(isprime(n)) return n;
    if(!(n & 1)) return 2;
    for(i = 1; i < 20; i++)
    {
        x = i, y = func(x, n), p = gcd(y - x, n);
        while(p == 1)
        {
            x = func(x, n);
            y = func(func(y, n), n);
            p = gcd((y - x + n) % n, n) % n;
        }
        if(p == 0 || p == n) continue;
        return p;
    }
}
```

5.8 原根-gwx

```
bool check_force(int g, int p)
{
    int cnt = 0, prod = g;
    for(int i = 1; i <= p - 1; ++i, prod = prod * g % p)
        if(prod == 1)
            if(++cnt > 1) return 0;
    return 1;
}

//d[]: prime divisor of (p - 1)
bool check_fast(int g, int p)
{
    for(int i = 1; i <= m; ++i)
        if(power(g, (p - 1) / d[i], p) == 1)
            return 0;
    return 1;
}

int primitive_root(int p)
{
    for(int i = 2; i < p; ++i)
        if(check(i, p)) return i;
}
```

5.9 勾股数

```
a=m^2-n^2
b=2mn
c=m^2+n^2
其中m和n中有一个是偶数，则(a，b，c)是素勾股数
```

6 字符串

6.1 AC 自动机-gwx

```
void add(int now)
{
    int k = 0;
    for(int i = 1; i <= n; i++)
    {
        int c = s[i] - 'A';
        if(!ch[k][c])
            ch[k][c] = ++cnt;
        k = ch[k][c];
    }
    ed[k] = 1;      //或vector全部记录
    id[now] = k;
}

void build()
{
    q.push(0);
    while(!q.empty())
    {
        int u = q.front(), v;
        q.pop();
        for(int i = 0; i < m; i++)
            if(v = ch[u][i])
            {
                int k = pa[u];
                while(k && !ch[k][i])
                    k = pa[k];
                if(u)
                    pa[v] = ch[k][i];
                q.push(v);
            }
            else
                ch[u][i] = ch[pa[u]][i];
    }
}
```

6.2 AC 自动机-wrz

```
struct ACAM
{
    ACAM *next[S], *fail;
```

```
    int ban;
}mem[N], *tot, *null, *root, *q[N];
ACAM *newACAM()
{
    ACAM *p = ++tot;
    *p = *null; return p;
}
void init()
{
    null = tot = mem;
    for(int i = 0; i < alpha; i++) null->next[i] = null;
    null->fail = null; null->ban = 0;
    root = newACAM();
}
void inser(char *s)
{
    ACAM *p = root;
    for(int i = 0; s[i]; i++)
    {
        int w = s[i] - 'a';
        if(p->next[w] == null) p->next[w] = newACAM();
        p = p->next[w];
    }
    p->ban = 1;
}
void build()
{
    root->fail = root; int head = 0, tail = 0;
    for(int i = 0; i < alpha; i++)
    {
        if(root->next[i] == null) root->next[i] = root;
        else root->next[i]->fail = root, q[tail++] = root->next[i];
    }
    for(; head < tail; head++)
    {
        ACAM *p = q[head];
        p->ban |= p->fail->ban;
        for(int i = 0; i < alpha; i++)
        {
            if(p->next[i] == null) p->next[i] = p->fail->next[i];
            else p->next[i]->fail = p->fail->next[i], q[tail++] = p->next[i];
        }
    }
}
```

6.3 exKMP-gwx

```
void get_next()
{
    int a = 0, p = 0;
    nxt[0] = m;
    for(int i = 1; i < m; i++)
    {
        if(i >= p || i + nxt[i - a] >= p)
        {
            if(i >= p)  p = i;
            while(p < m && t[p] == t[p - i]) p++;
            nxt[i] = p - i;
            a = i;
        }
        else nxt[i] = nxt[i - a];
    }
}

void exkmp()
{
    int a = 0, p = 0;
    get_next();
    for(int i = 0; i < n; i++)
    {
        if(i >= p || i + nxt[i - a] >= p) // i >= p 的作用：举个典型例子，s 和 t 无一字符相同
        {
            if(i >= p) p = i;
            while(p < n && p - i < m && s[p] == t[p - i]) p++;
            ext[i] = p - i;
            a = i;
        }
        else ext[i] = nxt[i - a];
    }
}
```

6.4 KMP-gwx

```
void kmp(char a[], char b[])  //a串中找b串
{
    int j = 0;
    for(int i = 2; i <= m; i++)
    {
        while(j && b[i] != b[j + 1]) j = p[j];
        if(b[i] == b[j + 1])  j++;
        p[i] = j;
    }
    j = 0;
    for(int i = 1; i <= n; i++)
    {
        while(j && b[j + 1] != a[i]) j = p[j];
        if(b[j + 1] == a[i]) j++;
        if(j == m)
        {
            ans[++cnt] = i;
```

```
            j = p[j];
        }
    }
}
```

6.5 KMP-wrz

```
void KMP_next(char *s, int len, int *next)
{
    next[1] = 0; int p = 0;
    for(int i = 2; i <= len; i++)
    {
        for(; s[p+1] != s[i] && p; p = next[p]);
        if(s[p+1] == s[i]) ++p;
        next[i] = p;
    }
}
```

6.6 最小表示-wrz

```
int min_represent(char *s, int len) // 当s不是字符串时应该将char改成int等, len是s的长度, 下标从0开始到n-1结束
{
    int i = 0, j = 1;
    for(; i < len && j < len; )
    {
        int k = 0;
        for(; s[(i+k)%len] == s[(j+k)%len] && k < len; k++);
        if(k == len) break;
        if(s[(i+k)%len] > s[(j+k)%len])
        {
            i += k+1;
            if(i <= j) i = j + 1;
        }
        else
        {
            j += k+1;
            if(j <= i) j = i + 1;
        }
    }
    return i < j ? i : j;
}
```

6.7 最小表示-gwx

```
//不保证起始位置最靠前(?)
string find(int N, string s) {
    int i, j, k, l;
    for (i = 0, j = 1; j < N; ) {
        for (k = 0; k < N && s[i + k] == s[j + k]; k++);
        if (k >= N) break;
        if (s[i + k] > s[j + k]) j += k + 1;
        else l = i + k, i = j, j = max(l, j) + 1;
    }
    return s.substr(i, N);
}
```

6.8 马拉车-gwx

```
//maxn = 2 * n
void manacher(int n)
{
    int p = 0, r = 0;
    for(int i = 1; i <= n; i++)
    {
        if(i <= r) len[i] = min(len[2 * p - i], r - i + 1);
        else len[i] = 1;
        while(b[i + len[i]] == b[i - len[i]]) len[i]++;
        if(i + len[i] - 1 >= r)
            r = i + len[i] - 1, p = i;
    }
}

int main()
{
    scanf("%d\n%s", &n, a + 1);
    b[++tot] = '@'; b[++tot] = '#';
    for(int i = 1; i < n; i++)
        b[++tot] = a[i], b[++tot] = '#';
    b[++tot] = a[n];
    b[++tot] = '#'; b[++tot] = '$';
    manacher(tot);
}
```

6.9 回文树-wrz

```
char s[N], out[N];
struct PT
{
    PT *fail, *next[A];
    int len;
}mem[N], *tot, *null, *root1, *root0, *last;
PT *newPT()
{
    PT *p = ++tot;
    *p = *null; return p;
}
void init()
{
```

```
    null = tot = mem;
    null->fail = null;
    for(int i = 0; i < A; i++) null->next[i] = null;
    null->len = 0;
    root1 = newPT(); root1->fail = root1; root1->len = -1;
    root0 = newPT(); root0->fail = root1; last = root1;
}
int extend(int c, int i) // 返回这一次是否多了一个回文子串
{
    PT *p = last;
    for(; s[i-p->len-1] != c+'a'; p = p->fail);
    if(p->next[c] != null) {last = p->next[c]; return 0;}
    PT *np = p->next[c] = last = newPT(); np->len = p->len + 2;
    if(p->len == -1) np->fail = root0;
    else
    {
        for(p=p->fail; s[i-p->len-1] != c+'a'; p = p->fail);
        np->fail = p->next[c];
    }
    return 1;
}
void main()
{
    scanf("%s",s+1); init();
    for(int i = 1, ii = strlen(s+1); i <= ii; i++)
        out[i] = extend(s[i]-'a', i)?'1':'0';
    puts(out+1);
}
```

6.10 后缀数组-gwx

```
//sa[i]: 排第i的串的开头位置  rank[i]:开头位置为i的串的排名
//maxn = 2 ^ k

void trans(int*s1, int*s2, int*r1, int*r2)
{
    for(int i = 1; i <= n; i++)
        v[r1[s1[i]]] = i;
    for(int i = n; i >= 1; i--)
        if(s1[i] > k)
            s2[v[r1[s1[i] - k]]--] = s1[i] - k;
    for(int i = n - k + 1; i <= n; i++)
        s2[v[r1[i]]--] = i;
    for(int i = 1; i <= n; i++)
        r2[s2[i]] = r2[s2[i - 1]] + (r1[s2[i]] != r1[s2[i - 1]] || r1[s2[i] + k] != r1[s2[i - 1] + k]);
}

int lcp(int s, int t)
{
    s = rank[p][s], t = rank[p][t];
    if(s > t) swap(s, t);
    s++;
    int k = Log[t - s + 1];
    return min(f[s][k], f[t + 1 - (1 << k)][k]);
}

void work()
{
    for(int k = 0; k <= maxk; k++)
        for(int i = 1 << k; i < (1 << k + 1) && i <= n; i++)
            Log[i] = k;
    int p = 0, q = 1;
    for(int i = 1; i <= n; i++)
        v[a[i]]++;
    for(int i = 1; i <= S; i++)  //S:alphabet_size
        v[i] += v[i - 1];
    for(int i = 1; i <= n; i++)
        sa[p][v[a[i]]--] = i;
    for(int i = 1; i <= n; i++)
        rank[p][sa[p][i]] = rank[p][sa[p][i - 1]] + (a[sa[p][i]] != a[sa[p][i - 1]]);
    k = 1;
    while(k < n)
    {
        trans(sa[p], sa[q], rank[p], rank[q]);
        p ^= 1, q ^= 1;
        k <<= 1;
    }
    for(int i = 1; i <= n; i++)
    {
        h[i] = max(h[i - 1] - 1, 0);
        int j = sa[p][rank[p][i] - 1];
        while(a[i + h[i]] == a[j + h[i]])
            h[i]++;
    }
    for(int i = 2; i <= n; i++)
        f[i][0] = h[i];
    for(int k = 1; k <= maxk; i++)
        for(int i = 2; i + (1 << k) - 1 <= n; i++)
            f[i][k] = min(f[i][k - 1], f[i + (1 << k - 1)][k - 1]);
}
```

6.11 后缀数组-wrz

```
char s[N];
int n, t1[N], t2[N], sa[N], rank[N], sum[N], height[N], lef, rig; // 数组开两倍
void SA_build()
{
    int *x = t1, *y = t2, m = 30;
    for(int i = 1; i <= n; i++) sum[x[i] = s[i] - 'a' + 1]++;
    for(int i = 1; i <= m; i++) sum[i] += sum[i-1];
```

```
    for(int i = n; i >= 1; i--) sa[sum[x[i]]--] = i;
    for(int k = 1; k <= n; k <<= 1)
    {
        int p = 0;
        for(int i = n-k+1; i <= n; i++) y[++p] = i;
        for(int i = 1; i <= n; i++) if(sa[i] - k > 0) y[++p] = sa[i] - k;

        for(int i = 1; i <= m; i++) sum[i] = 0;
        for(int i = 1; i <= n; i++) sum[x[i]]++;
        for(int i = 1; i <= m; i++) sum[i] += sum[i-1];
        for(int i = n; i >= 1; i--) sa[sum[x[y[i]]]--] = y[i];

        swap(x, y);
        for(int i = 1; i <= n; i++)
            x[sa[i]] = x[sa[i-1]] + (y[sa[i]] == y[sa[i-1]] && y[sa[i]+k] == y[sa[i-1]+k] ? 0 : 1);
        m = x[sa[n]];
        if(m == n) break;
    }
    for(int i = 1; i <= n; i++) rank[sa[i]] = i;
    for(int i = 1, k = 0; i <= n; height[rank[i++]] = k?k--:k)
        for(; s[i+k] == s[sa[rank[i]-1]+k]; k++);
}
```

6.12 后缀数组 SAIS

```
// string is 0-based
// sa[] is 1-based
// s[n] < s[i] i = 0...n-1
namespace SA {
    int sa[MAXN], rk[MAXN], ht[MAXN], s[MAXN << 1], t[MAX << 1], p[MAXN], cnt[MAXN], cur[MAXN];
#define pushS(x) sa[cur[s[x]]--] = x
#define pushL(x) sa[cur[s[x]]++] = x
#define inducedSort(v) std::fill_n(sa, n, -1); std::fill_n(cnt, m, 0);\
    for (int i = 0; i < n; i++) cnt[s[i]]++;\
    for (int i = 1; i < m; i++) cnt[i] += cnt[i-1];\
    for (int i = 0; i < m; i++) cur[i] = cnt[i]-1;\
    for (int i = n1-1; ~i; i--) pushS(v[i]);\
    for (int i = 1; i < m; i++) cur[i] = cnt[i-1];\
    for (int i = 0; i < n; i++) if (sa[i] > 0 &&  t[sa[i]-1]) pushL(sa[i]-1);\
    for (int i = 0; i < m; i++) cur[i] = cnt[i]-1;\
    for (int i = n-1;  ~i; i--) if (sa[i] > 0 && !t[sa[i]-1]) pushS(sa[i]-1)
    void sais(int n, int m, int *s, int *t, int *p) {
        int n1 = t[n-1] = 0, ch = rk[0] = -1, *s1 = s+n;
        for (int i = n-2; ~i; i--) t[i] = s[i] == s[i+1] ? t[i+1] : s[i] > s[i+1];
        for (int i = 1; i < n; i++) rk[i] = t[i-1] && !t[i] ? (p[n1] = i, n1++) : -1;
        inducedSort(p);
        for (int i = 0, x, y; i < n; i++) if (~(x = rk[sa[i]])) {
            if (ch < 1 || p[x+1] - p[x] != p[y+1] - p[y]) ch++;
            else for (int j = p[x], k = p[y]; j <= p[x+1]; j++, k++)
                if ((s[j]<<1|t[j]) != (s[k]<<1|t[k])) {ch++; break;}
            s1[y = x] = ch; }
        if (ch+1 < n1) sais(n1, ch+1, s1, t+n, p+n1);
        else for (int i = 0; i < n1; i++) sa[s1[i]] = i;
        for (int i = 0; i < n1; i++) s1[i] = p[sa[i]];
        inducedSort(s1); }
    int mapCharToInt(int n, const T *str) {
        int m = *std::max_element(str, str+n);
        std::fill_n(rk, m+1, 0);
        for (int i = 0; i < n; i++) rk[str[i]] = 1;
        for (int i = 0; i < m; i++) rk[i+1] += rk[i];
        for (int i = 0; i < n; i++) s[i] = rk[str[i]] - 1;
        return rk[m]; }
    void suffixArray(int n, const T *str) {
        int m = mapCharToInt(++n, str);
        sais(n, m, s, t, p);
        for (int i = 0; i < n; i++) rk[sa[i]] = i;
        for (int i = 0, h = ht[0] = 0; i < n-1; i++) {
            int j = sa[rk[i]-1];
            while (i+h < n && j+h < n && s[i+h] == s[j+h]) h++;
            if (ht[rk[i]] = h) h--; } } };
```

6.13 后缀自动机-gwx

```
int root = 1, cnt = 1, last = 1;
int pa[maxn], l[maxn], ch[maxn][maxs];

void add(int c) //c : 0 ~ alpha_size
{
    int np = ++cnt, p = last; last = cnt;
    l[np] = x; r[np] = 1;
    while(p && !ch[p][c])
        ch[p][c] = np, p = pa[p];
    if(!p)
    {
        pa[np] = root;
        return;
    }
    int q = ch[p][c];
    if(l[q] == l[p] + 1)
        pa[np] = q;
    else
    {
        int nq = ++cnt;
        l[nq] = l[p] + 1;
        pa[nq] = pa[q];
        pa[q] = pa[np] = nq;
        memcpy(ch[nq], ch[q], sizeof(ch[q]));
        while(p && ch[p][c] == q)
            ch[p][c] = nq, p = pa[p];
    }
```

```
}

void get_right()
{
    for(int i = 1; i <= n; i++) add(i);
    for(int i = 1; i <= cnt; i++) v[l[i]]++;
    for(int i = 1; i <= n; i++) v[i] += v[i - 1];
    for(int i = cnt; i; i--) t[v[l[i]]--] = i;
    for(int i = cnt; i; i--) if(pa[t[i]]) r[pa[t[i]]] += r[t[i]];
    r[1] = 0;
}
```

6.14 后缀自动机-wrz

```
struct SAM
{
    SAM *next[A], *fail;
    int len, mi, mx;
}mem[N], *tot, *null, *root, *last, *q[N];
SAM *newSAM(int len)
{
    SAM *p = ++tot;
    *p = *null;
    p->len = p->mi = len;
    p->mx = 0;
    return p;
}
void init()
{
    null = tot = mem;
    for(int i = 0; i < A; i++) null->next[i] = null;
    null->fail = null;
    null->len = null->mi = null->mx = 0;
    root = last = newSAM(0);
}
void extend(int v)
{
    SAM *p = last, *np = newSAM(p->len + 1); last = np;
    for(; p->next[v] == null && p != null; p = p->fail) p->next[v] = np;
    if(p==null) np->fail = root;
    else
    {
        SAM *q = p->next[v];
        if(q->len == p->len+1) np->fail = q;
        else
        {
            SAM *nq = newSAM(p->len+1);
            memcpy(nq->next, q->next, sizeof(nq->next));
            nq->fail = q->fail;
            q->fail = np->fail = nq;
            for(; p->next[v] == q && p != null; p = p->fail) p->next[v] = nq;
        }
    }
}
```

6.15 ex 后缀自动机-wrz

```
struct sam
{
    sam *fail, *next[A];
    int len;
}mem[N<<1], *tot, *null, *root;
sam* newsam()
{
    *++tot = *null;
    return tot;
}
void init()
{
    null = tot = mem; null->fail = null; null->len = 0;
    for(int i = 0; i < A; i++) null->next[i] = null;
    root = newsam();
}
sam* extend(sam *p, int v)
{
    if(p->next[v] != null)
    {
        sam *q = p->next[v];
        if(p->len + 1 == q->len) return q;
        else
        {
            sam *nq = newsam(); *nq = *q; nq->len = p->len + 1;
            q->fail = nq;
            for(; p->next[v] == q && p != null; p = p->fail) p->next[v] = nq;
            return nq;
        }
    }
    else
    {
        sam *np = newsam(); np->len = p->len + 1;
        for(; p->next[v] == null && p != null; p = p->fail) p->next[v] = np;
        if(p == null) np->fail = root;
        else
        {
            sam *q = p->next[v];
            if(p->len + 1 == q->len) np->fail = q;
            else
            {
                sam *nq = newsam(); *nq = *q; nq->len = p->len + 1;
                np->fail = q->fail = nq;
                for(; p->next[v] == q && p != null; p = p->fail) p->next[v] = nq;
```

```
            }
        }
        return np;
    }
}
void build_tree()
{
    for(sam *i = tot; i != mem; i--)
        addedge(i->fail - mem, i - mem);
}
```

7 其他

7.1 cout 输出小数

```
std::cout << std::fixed << std::setprecision(5);
```

7.2 枚举子集

```
for (int x = 1; x <= n; x++)
    for (int y  = x & (x - 1); y; (--y) &= x) {
        //y is a subset of x
    }
```

7.3 梅森旋转

```
#include <random>

int main() {
    std::mt19937 g(seed);  // std::mt19937_64
    std::cout << g() << std::endl;
}
```

7.4 乘法取模

```
// 需要保证 x 和 y 非负
long long mult(long long x, long long y, long long MODN) {
    long long t = (x * y - (long long)((long double)x / MODN * y + 1e-3) * MODN) % MODN;
    return t < 0 ? t + MODN : t;
}
```

7.5 释放容器内存

```
template <typename T>
__inline void clear(T& container) {
    container.clear();  // 或者删除了一堆元素
    T(container).swap(container);
}
```

7.6 tuple

```
mytuple = std::make_tuple (10, 2.6, 'a');           // packing values into tuple
std::tie (myint, std::ignore, mychar) = mytuple;    // unpacking tuple into variables
std::get<I>(mytuple) = 20;
std::cout << std::get<I>(mytuple) << std::endl;     // get the Ith(const) element
```

7.7 读入优化

```
// getchar()读入优化 << 关同步cin << 此优化
// 用isdigit()会小幅变慢
// 返回 false 表示读到文件尾
namespace Reader {
    const int L = (1 << 15) + 5;
    char buffer[L], *S, *T;
    __inline bool getchar(char &ch) {
        if (S == T) {
            T = (S = buffer) + fread(buffer, 1, L, stdin);
            if (S == T) {
                ch = EOF;
                return false;
            }
        }
        ch = *S++;
        return true;
    }
    __inline bool getint(int &x) {
        char ch; bool neg = 0;
        for (; getchar(ch) && (ch < '0' || ch > '9'); ) neg ^= ch == '-';
        if (ch == EOF) return false;
        x = ch - '0';
        for (; getchar(ch), ch >= '0' && ch <= '9'; )
            x = x * 10 + ch - '0';
        if (neg) x = -x;
        return true;
    }
}
```

7.8 蔡勒公式

```
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);
}
```

7.9 dancing-links

```cpp
struct Node {
    Node *l, *r, *u, *d, *col;
    int size, line_no;
    Node() {
        size = 0; line_no = -1;
        l = r = u = d = col = NULL;
    }
} *root;

void cover(Node *c) {
    c->l->r = c->r; c->r->l = c->l;
    for (Node *u = c->d; u != c; u = u->d)
        for (Node *v = u->r; v != u; v = v->r) {
            v->d->u = v->u;
            v->u->d = v->d;
            -- v->col->size;
        }
}

void uncover(Node *c) {
    for (Node *u = c->u; u != c; u = u->u) {
        for (Node *v = u->l; v != u; v = v->l) {
            ++ v->col->size;
            v->u->d = v;
            v->d->u = v;
        }
    }
    c->l->r = c; c->r->l = c;
}

std::vector<int> answer;
bool search(int k) {
    if (root->r == root) return true;
    Node *r = NULL;
    for (Node *u = root->r; u != root; u = u->r)
        if (r == NULL || u->size < r->size)
            r = u;
    if (r == NULL || r->size == 0) return false;
    else {
        cover(r);
        bool succ = false;
        for (Node *u = r->d; u != r && !succ; u = u->d) {
            answer.push_back(u->line_no);
            for (Node *v = u->r; v != u; v = v->r)  // Cover row
                cover(v->col);
            succ |= search(k + 1);
            for (Node *v = u->l; v != u; v = v->l)
                uncover(v->col);
            if (!succ) answer.pop_back();
        }
        uncover(r);
        return succ;
    }
}

bool entry[CR][CC];
Node *who[CR][CC];
int cr, cc;

void construct() {
    root = new Node();
    Node *last = root;
    for (int i = 0; i < cc; ++ i) {
        Node *u = new Node();
        last->r = u; u->l = last;
        Node *v = u; u->line_no = i;
        last = u;
        for (int j = 0; j < cr; ++ j)
            if (entry[j][i]) {
                ++ u->size;
                Node *cur = new Node();
                who[j][i] = cur;
                cur->line_no = j;
                cur->col = u;
                cur->u = v; v->d = cur;
                v = cur;
            }
        v->d = u; u->u = v;
    }
    last->r = root; root->l = last;
    for (int j = 0; j < cr; ++ j) {
        Node *last = NULL;
        for (int i = cc - 1; i >= 0; -- i)
            if (entry[j][i]) {
                last = who[j][i];
                break;
            }
        for (int i = 0; i < cc; ++ i)
            if (entry[j][i]) {
                last->r = who[j][i];
                who[j][i]->l = last;
                last = who[j][i];
            }
    }
}

void destruct() {
    for (Node *u = root->r; u != root; ) {
        for (Node *v = u->d; v != u; ) {
            Node *nxt = v->d;
```

```
            delete(v);
            v = nxt;
        }
        Node *nxt = u->r;
        delete(u); u = nxt;
    }
    delete root;
}
```

8 提示

8.1 费用流

```
有源汇上下界费用流:
    转换为求无源汇上下界最小费用可行循环流, 通过T→S连边, 流量上下界为(原总流量, INF)。

无源汇上下界最小费用可行循环流:
    在原基础上再新增一个超级源点 supS, supT, 构造只有上界的网络。
    对于原图的每一条边 (u, v) , 再新图中添加一条 supS→v 流量为 u, v 流量下界的边, 一条 u→supT 流量为 u, v 流量下界的边, 一条 u→v 流量为 u, v 流量上界-流量下界的边。
    做从 supS→supT 的最小费用流, 限定到达 supT 的流量为满流 (即 supS 所有出边的流量和) 。此即为答案。
    HINT: 原图中所有未提及的边费用都应记为 0 。新图中的重新构造的边的费用等同原图中对应边的费用。
```

8.2 网络流

```
4.7 上下界网络流
B(u, v) 表示边 (u, v) 流量的下界,C(u, v) 表示边 (u, v) 流量的上界,F (u, v) 表示边 (u, v)
的流量。设 G(u, v) = F (u, v) - B(u, v),显然有
0   G(u, v)   C(u, v) - B(u, v)
4.7.1 无源汇的上下界可行流
建立超级源点 S * 和超级汇点 T * ,对于原图每条边 (u, v) 在新网络中连如下三条边: S * → v,
容量为 B(u, v);u → T * ,容量为 B(u, v);u → v,容量为 C(u, v) - B(u, v)。最后求新网络
的最大流,判断从超级源点 S * 出发的边是否都满流即可,边 (u, v) 的最终解中的实际流量为
G(u, v) + B(u, v)。
4.7.2 有源汇的上下界可行流
从汇点 T 到源点 S 连一条上界为 ∞,下界为 0 的边。按照无源汇的上下界可行流一样做
即可,流量即为 T → S 边上的流量。
4.7.3 有源汇的上下界最大流
1. 在有源汇的上下界可行流中,从汇点 T 到源点 S 的边改为连一条上界为 ∞,下届为 x 的
边。x 满足二分性质,找到最大的 x 使得新网络存在无源汇的上下界可行流即为原图的最大
流。
2. 从汇点 T 到源点 S 连一条上界为 ∞,下界为 0 的边,变成无源汇的网络。按照无源汇的
上下界可行流的方法,建立超级源点 S * 和超级汇点 T * ,求一遍 S * → T * 的最大流,再将
从汇点 T 到源点 S 的这条边拆掉,求一次 S → T 的最大流即可。
4.7.4 有源汇的上下界最小流
1. 在有源汇的上下界可行流中,从汇点 T 到源点 S 的边改为连一条上界为 x,下界为 0 的
边。x 满足二分性质,找到最小的 x 使得新网络存在无源汇的上下界可行流即为原图的最小
流。
712. 按照无源汇的上下界可行流的方法,建立超级源点 S * 与超级汇点 T * ,求一遍 S * → T * 的
最大流,但是注意这一次不加上汇点 T 到源点 S 的这条边,即不使之改为无源汇的网络去
求解。求完后,再加上那条汇点 T 到源点 S 上界 ∞ 的边。因为这条边下界为 0,所以
S * ,T * 无影响,再直接求一次 S * → T * 的最大流。若超级源点 S * 出发的边全部满流,则
T → S 边上的流量即为原图的最小流,否则无解。
```

8.3 莫比乌斯

```
F(n) =  {d|n} f(d) => f(n) =   {d|n} μ(d)F(n/d)

μ(d) = 1 (d = 1)
μ(d) = (-1)^k
```

8.4 矩阵树定理

```
C = 度数矩阵-领接矩阵
无向图G的生成树个数 = C的任意n - 1阶主子式(对角线的乘积)
```

8.5 Java

```java
import java.util.*;
import java.math.*;
public class javaNote
{
    static BigInteger q[] = new BigInteger[5000000]; // 定义数组的正确姿势, 记得分配内存

    public static void main(String[] args)
    {

        long currentTime = System.currentTimeMillis(); // 获取时间, 单位是ms

        Scanner sc = new Scanner(System.in); // 定义输入
        int a = sc.nextInt(), b;
        System.out.println("integer␣=␣" + a); // 输出

        BigInteger x = new BigInteger("233"), y = new BigInteger("666");
        BigInteger.valueOf(1); // 将指定的表达式转化成BigInteger类型
        x.add(y); //x+y
        x.subtract(y); //x-y
        x.multiply(y); //x*y
        x.divide(y);

        x.pow(233); // x**233
        x.compareTo(y); // 比较x和y, x < y : -1, x = y : 0, x > y : 1

```

```
        BigDecimal n = new BigDecimal("233"), m = new BigDecimal("666");
        n.divide(m,a,RoundingMode.DOWN); //n/m并精确到小数点后第a位, a=0表示精确到个位, a为负数表示精确到小数点前
            -a+1位, 可能变成科学计数法
        /*
            取整方式
            RoundingMode.CEILING: 取右边最近的整数, 即向正无穷取整
            RoundingMode.FLOOR: 取左边最近的整数, 即向负无穷取整
            RoundingMode.DOWN: 向0取整
            RoundingMode.UP: 远离0取整
            RoundingMode.HALF_UP:上取整的四舍五入, >=0.5会进位, <0.5会舍去, 负数会先取绝对值再四舍五入再变回负数
            RoundingMode.HALF_DOWN:下取整的四舍五入, >0.5会进位, <=0.5会舍去, 负数原理同上
            RoundingMode.HALF_EVEN:分奇偶的四舍五入, >0.5会进位, <0.5会舍去, =0.5会向最近的偶数取整, 如2.5->2,
                (-2.5)->(-2)
        */

        Math.max(a, b);//取大
        Math.min(a, b);//取小
        Math.PI;//pi

        HashSet<BigInteger> hash = new HashSet<BigInteger>(); // hash table
        hash.contains(x); // hash table中是否有a, 有则返回true, 反之返回false
        hash.add(x); // 把x加进hash table
        hash.remove(x); // 从hash table中删去x

    }
}
```

9 附录-数学公式

7.4 常见错误

1. 数组或者变量类型开错，例如将 double 开成 int；
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；

7.5 测试列表

1. 检测评测机是否开 O2；
2. 检测 __int128 以及 __float128 是否能够使用；
3. 检测是否能够使用 C++11；
4. 检测是否能够使用 Ext Lib；
5. 检测程序运行所能使用的内存大小；
6. 检测程序运行所能使用的栈大小；
7. 检测是否有代码长度限制；
8. 检测是否能够正常返 Runtime Error（assertion、return 1、空指针）；
9. 查清楚厕所方位和打印机方位；

7.6 博弈游戏

7.6.1 巴什博奕

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，$s \leq m$)，那么先取者要拿走 s 个物品，如果后取者拿走 $k(k \leq m)$ 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。

7.6.2 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：

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

7.6.3 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）

7.6.4 图上删边游戏

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）

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。

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。

7.7 常用数学公式

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

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

7.7.3 错排公式

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

7.7.4 莫比乌斯函数

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

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

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

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

7.7.5 Burnside 引理

设 G 是一个有限群，作用在集合 X 上。对每个 g 属于 G，令 X^g 表示 X 中在 g 作用下的不动元素，轨道数（记作 $|X/G|$）由如下公式给出：

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

7.7.6 五边形数定理

设 $p(n)$ 是 n 的拆分数，有

$$p(n)=\sum_{k\in\mathbb{Z}\setminus\{0\}}(-1)^{k-1}p\left(n-\frac{k(3k-1)}{2}\right)$$

7.7.7 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

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

   当 n 为偶数时，n 个结点的无根树的个数为

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

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。

7.7.8 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：

$$V - E + F = 2$$

7.7.9 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：

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

7.7.10 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

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

7.8 平面几何公式

7.8.1 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

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

7.8.2 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
$$S = \sqrt{(p-a)(p-b)(p-c)(p-d)}$$

7.8.3 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
$$S = \frac{nar}{2} = nr^2 \cdot tan\frac{A}{2} = \frac{nR^2}{2} \cdot sinA = \frac{na^2}{4 \cdot tan\frac{A}{2}}$$

7.8.4 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
$$S_2 = \frac{rl - a(r-h)}{2} = \frac{r^2}{2}(A - sinA)$$

7.8.5 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
$$T = S + 2A$$

7.8.6 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
$$T = S + 2A$$

7.8.7 棱台

1. 体积
$$V = (A_1 + A_2 + \sqrt{A_1 A_2}) \cdot \frac{h}{3}$$
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
$$T = S + A_1 + A_2$$

7.8.8 圆柱

1. 侧面积
$$S = 2\pi rh$$
2. 全面积
$$T = 2\pi r(h + r)$$
3. 体积
$$V = \pi r^2 h$$

7.8.9 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
$$V = \frac{\pi}{3} r^2 h$$

7.8.10 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
$$V = \frac{\pi}{3}(r_1^2 + r_2^2 + r_1 r_2)h$$

7.8.11 球

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

7.8.12 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
$$V = \frac{\pi h[3(r_1^2 + r_2^2) + h^2]}{6}$$

7.8.13 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
$$V = \frac{2}{3}\pi r^2 h$$

7.9 立体几何公式

7.9.1 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 $A + B + C - \pi$

7.9.2 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

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

7.10 附录

7.10.1 NTT 素数及原根列表

Id	Primes	Primitive Root	Id	Primes	Primitive Root	Id	Primes	Primitive Root
1	7340033	3	38	311427073	7	75	786432001	7
2	13631489	15	39	330301441	22	76	799014913	13
3	23068673	3	40	347078657	3	77	800063489	3
4	26214401	3	41	359661569	3	78	802160641	11
5	28311553	5	42	361758721	29	79	818937857	5
6	69206017	5	43	377487361	7	80	824180737	5
7	70254593	3	44	383778817	5	81	833617921	13
8	81788929	7	45	387973121	6	82	850395137	3
9	101711873	3	46	399507457	5	83	862978049	3
10	104857601	3	47	409993217	3	84	880803841	26
11	111149057	3	48	415236097	5	85	883949569	7
12	113246209	7	49	447741953	3	86	897581057	3
13	120586241	6	50	459276289	11	87	899678209	7
14	132120577	5	51	463470593	3	88	907018241	3
15	136314881	3	52	468713473	5	89	913309697	3
16	138412033	5	53	469762049	3	90	918552577	5
17	141557761	26	54	493879297	10	91	919601153	3
18	147849217	5	55	531628033	5	92	924844033	5
19	155189249	6	56	576716801	6	93	925892609	3
20	158334977	3	57	581959681	11	94	935329793	3
21	163577857	23	58	595591169	3	95	938475521	3
22	167772161	3	59	597688321	11	96	940572673	7
23	169869313	5	60	605028353	3	97	943718401	7
24	185597953	5	61	635437057	11	98	950009857	7
25	186646529	3	62	639631361	6	99	957349889	6
26	199229441	3	63	645922817	3	100	962592769	7
27	204472321	19	64	648019969	17	101	972029953	10
28	211812353	3	65	655360001	3	102	975175681	17
29	221249537	3	66	666894337	5	103	976224257	3
30	230686721	6	67	683671553	3	104	985661441	3
31	246415361	3	68	710934529	17	105	998244353	3
32	249561089	3	69	715128833	3	106	1004535809	3
33	257949697	5	70	718274561	3	107	1007681537	3
34	270532609	22	71	740294657	3	108	1012924417	5
35	274726913	3	72	745537537	5	109	1045430273	3
36	290455553	3	73	754974721	11	110	1051721729	6
37	305135617	5	74	770703361	11	111	1053818881	7

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 $\|a_n - a\| < \epsilon$, $\forall n \ge n_0$.
$\sup S$	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 $\pi_1\pi_2\dots\pi_n$ on $\{1, 2, \dots, n\}$ 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. $\binom{n}{k} = \frac{n!}{(n-k)!k!}$, **2.** $\sum_{k=0}^{n}\binom{n}{k} = 2^n$, **3.** $\binom{n}{k} = \binom{n}{n-k}$,

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 $d_1, \ldots, d_n$:
$$\sum_{i=1}^{n} 2^{-d_i} \leq 1,$$
and equality holds only if every internal node has 2 sons.

Recurrences

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

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 $af(n/b) \leq cf(n)$ 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:
$$\sum_{i \geq 0} g_{i+1} x^i = \sum_{i \geq 0} 2g_i x^i + \sum_{i \geq 0} x^i.$$

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

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:
$$\begin{aligned} G(x) &= x\left(\frac{2}{1-2x} - \frac{1}{1-x}\right) \\ &= x\left(2\sum_{i \geq 0} 2^i x^i - \sum_{i \geq 0} x^i\right) \\ &= \sum_{i \geq 0} (2^{i+1} - 1)x^{i+1}. \end{aligned}$$

So $g_i = 2^i - 1$.

Theoretical Computer Science Cheat Sheet

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

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:

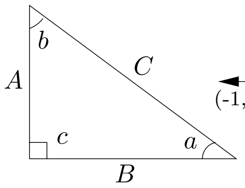
$$\Pr\left[|X| \geq \lambda\,\mathrm{E}[X]\right] \leq \frac{1}{\lambda},$$
$$\Pr\left[\left|X - \mathrm{E}[X]\right| \geq \lambda \cdot \sigma\right] \leq \frac{1}{\lambda^2}.$$

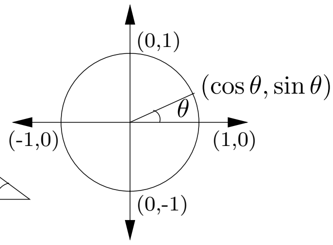
Geometric distribution:

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

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

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

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

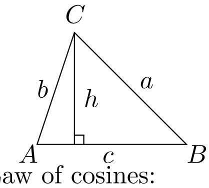
Definitions:
$$\sinh x = \frac{e^x - e^{-x}}{2}, \qquad \cosh x = \frac{e^x + e^{-x}}{2},$$
$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \qquad \operatorname{csch} x = \frac{1}{\sinh x},$$
$$\operatorname{sech} x = \frac{1}{\cosh x}, \qquad \coth x = \frac{1}{\tanh x}.$$

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

Theoretical Computer Science Cheat Sheet

Number Theory

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

$$\begin{aligned} C &\equiv r_1 \bmod m_1 \\ \vdots \;\; & \vdots \;\; \vdots \\ C &\equiv r_n \bmod m_n \end{aligned}$$

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

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

Prime numbers:

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

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

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:

$$\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2},$$

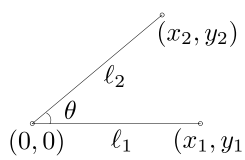
$$\left[|x_1 - x_0|^p + |y_1 - y_0|^p\right]^{1/p},$$

$$\lim_{p \to \infty} \left[|x_1 - x_0|^p + |y_1 - y_0|^p\right]^{1/p}.$$

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

Theoretical Computer Science Cheat Sheet
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. $\displaystyle\int \frac{dx}{x\sqrt{x^2-a^2}} = \frac{1}{a}\arccos\frac{a}{|x|}, \quad a>0,$

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. $\displaystyle\int x^3\sqrt{x^2+a^2}\,dx = (\tfrac{1}{3}x^2-\tfrac{2}{15}a^2)(x^2+a^2)^{3/2},$

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

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

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:

$x^{\underline{n}} = (-1)^n(-x)^{\overline{n}} = (x-n+1)^{\overline{n}}$
$\quad = 1/(x+1)^{\overline{-n}},$

$x^{\overline{n}} = (-1)^n(-x)^{\underline{n}} = (x+n-1)^{\underline{n}}$
$\quad = 1/(x-1)^{\underline{-n}},$

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

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

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

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:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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:

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

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

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

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

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

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

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

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

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

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

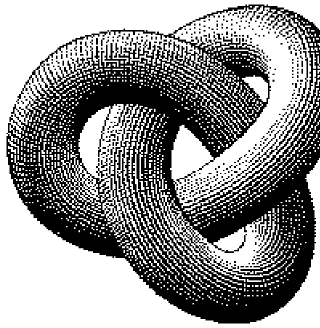
$$\left(\frac{1-\sqrt{1-4x}}{2x}\right)^n = \sum_{i=0}^{\infty}\frac{n(2i+n-1)!}{i!(n+i)!}x^i,$$

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

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

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

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:

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

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

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

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

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

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)