# Arondight

For Manual/Intelligence

August 6, 2018

# Arondight's Standard Code Library*

*Shanghai Jiao Tong University*

Dated: November 16, 2017

*https://github.com/footoredo/Arondight

# Contents

# Chapter 1

# 计算几何

```
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
    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;
```

## 1.1 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();
    }
```

```cpp
    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;
    }
};
```

## 1.2 三角形的心

```
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);
}
```

## 1.3 半平面交

```
struct Point {
    int quad() const { return sign(y) == 1 || (sign(y) == 0 && sign(x) >= 0);}
};
struct Line {
    bool include(const Point &p) const { return sign(det(b - a, p - a)) > 0; }
    Line push() const{ // 将半平面向外推 eps
        const double eps = 1e-6;
        Point delta = (b - a).turn90().norm() * eps;
        return Line(a - delta, b - delta);
    }
};
bool sameDir(const Line &l0, const Line &l1) { return parallel(l0, l1) && sign(dot(l0.b - l0.a,
  ↪l1.b - l1.a)) == 1; }
bool operator < (const Point &a, const Point &b) {
    if (a.quad() != b.quad()) {
        return a.quad() < b.quad();
    } else {
        return sign(det(a, b)) > 0;
    }
```

```
}
bool operator < (const Line &l0, const Line &l1) {
    if (sameDir(l0, l1)) {
        return l1.include(l0.a);
    } else {
        return (l0.b - l0.a) < (l1.b - l1.a);
    }
}
bool check(const Line &u, const Line &v, const Line &w) { return w.include(intersect(u, v)); }
vector<Point> intersection(vector<Line> &l) {
    sort(l.begin(), l.end());
    deque<Line> q;
    for (int i = 0; i < (int)l.size(); ++i) {
        if (i && sameDir(l[i], l[i - 1])) {
            continue;
        }
        while (q.size() > 1 && !check(q[q.size() - 2], q[q.size() - 1], l[i])) q.pop_back();
        while (q.size() > 1 && !check(q[1], q[0], l[i])) q.pop_front();
        q.push_back(l[i]);
    }
    while (q.size() > 2 && !check(q[q.size() - 2], q[q.size() - 1], q[0])) q.pop_back();
    while (q.size() > 2 && !check(q[1], q[0], q[q.size() - 1])) q.pop_front();
    vector<Point> ret;
    for (int i = 0; i < (int)q.size(); ++i) ret.push_back(intersect(q[i], q[(i + 1) %
    ↪ q.size()]));
    return ret;
}
```

## 1.4 圆交面积及重心

```
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];
        }
}
```

## 1.5 极角排序

```
bool cmp(const Point &a,const Point &b)
{
    int tmp1=(a.y>0 || (a.y==0 && a.x>0))?1:2;
    int tmp2=(b.y>0 || (b.y==0 && b.x>0))?1:2;
    if (tmp1!=tmp2)    return tmp1<tmp2;
    return det(a,b)>0;
}
```

## 1.6 极角求交

```
bool in(Point x,Point y,Point z)
{
    if (sign(det(x,y))>0)    swap(x,y);
    return sign(det(z,y))<=0 && sign(det(x,z))<=0;
}
bool jiao(Point &x,Point &y,Point l,Point r)
{
    if (!in(x,y,l) && !in(x,y,r))    swap(x,l),swap(y,r);
    if (!in(x,y,l) && !in(x,y,r))    return false;
    if (!in(x,y,l))    swap(l,r);
    if (in(x,y,l) && in(x,y,r))
    {
        x=l;
        y=r;
        return true;
    }
    bool xx=in(l,r,x),yy=in(l,r,y);
    if (xx && yy)
```

```
    {
        return true;
    }
    if (!xx && !yy)
    {
        x=l;
        y=r;
        return true;
    }
    if (xx && !yy)
    {
        y=l;
        return true;
    }
    x=l;
    return true;
}
```

## 1.7 最小圆覆盖

```
Point get_circle_center(const Point &a,const Point &b,const Point &c)
{
    Point center;
    double a1=b.x-a.x;
    double b1=b.y-a.y;
    double c1=(a1*a1+b1*b1)/2.0;
    double a2=c.x-a.x;
    double b2=c.y-a.y;
    double c2=(a2*a2+b2*b2)/2.0;
    double d=a1*b2-a2*b1;
    center.x=a.x+(c1*b2-c2*b1)/d;
    center.y=a.y+(a1*c2-a2*c1)/d;
    return center;
}
inline bool inCircle(const Point &p,const Circle &c){
    return sign((p-c.o).len()-c.r)<=0;
}
Circle Min_cover(vector<Point> p)
{//注意考虑没有点的情况
    random_shuffle(p.begin(),p.end());
    int n=p.size();
    Circle ans;
    ans.o=p[0];
    ans.r=0;
    for (int i=1;i<n;i++)
    if (!inCircle(p[i],ans))
    {
        ans.o=p[i];
```

```
        ans.r=0;
        for (int j=0;j<i;j++)
        if (!inCircle(p[j],ans))
        {
            ans.o=(p[j]+p[i])/2.0;
            ans.r=((p[j]-p[i]).len())/2.0;
            for (int k=0;k<j;k++)
            if (!inCircle(p[k],ans))
            {
                ans.o=get_circle_center(p[i],p[j],p[k]);
                ans.r=(p[i]-ans.o).len();
            }
        }
    }
    return ans;
}
```

## 1.8 三维向量绕轴旋转

```
// 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]);
}
```

## 1.9 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);
}
```

# Chapter 2

# 数论

## 2.1 $O(m^2 \log n)$ 求线性递推数列第 n 项

Given $a_0, a_1, \ldots, a_{m-1}$
$a_n = c_0 \times a_{n-m} + \cdots + c_{m-1} \times a_{n-1}$
Solve for $a_n = v_0 \times a_0 + v_1 \times a_1 + \cdots + v_{m-1} \times a_{m-1}$

```
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];
    }
}
```

## 2.2 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;
}
```

## 2.3 中国剩余定理

```
// 返回 (ans, M)，其中 ans 是模 M 意义下的解
std::pair<long long, long long> CRT(const std::vector<long long>& m, const std::vector<long
  ↪ long>& a) {
    long long M = 1, ans = 0;
    int n = m.size();
    for (int i = 0; i < n; i++) M *= m[i];
    for (int i = 0; i < n; i++) {
        ans = (ans + (M / m[i]) * a[i] % M * inv(M / m[i], m[i])) % M;  // 可能需要大整数相乘取模
    }
    return std::make_pair(ans, M);
}
```

```cpp
// 模数不互质的情况
bool solve(int n, std::pair<long long, long long> input[],
                  std::pair<long long, long long> &output) {
    output = std::make_pair(1, 1);
    for (int i = 0; i < n; ++i) {
        long long number, useless;
        // euclid(a, b, x, y)
        euclid(output.second, input[i].second, number, useless);
        long long divisor = std::__gcd(output.second, input[i].second);
        if ((input[i].first - output.first) % divisor) return false;
        number *= (input[i].first - output.first) / divisor;
        fix(number, input[i].second);    // fix 成正的
        output.first += output.second * number;
        output.second *= input[i].second / divisor;
        fix(output.first, output.second);
    }
    return true;
}
```

## 2.4 魔法 CRT

```cpp
// MOD is the given module
// Do not depend on LL * LL % LL
inline int CRT(int *a) {
    static int x[N];
    for (int i = 0; i < N; i ++) {
        x[i] = a[i];
        for (int j = 0; j < i; j ++) {
            int t = (x[i] - x[j] + mod[i]) % mod[i];
            if (t < 0) t += mod[i];
            x[i] = 1LL * t * Inv[j][i] % mod[i];
        }
    }
    int sum = 1, ret = x[0] % MOD;
    for (int i = 1; i < N; i ++) {
        sum = 1LL * sum * mod[i - 1] % MOD;
        ret += 1LL * x[i] * sum % MOD;
        if (ret >= MOD) ret -= MOD;
    }
    return ret;
}
for (int i = 0; i < N; i ++)
    for (int j = i + 1; j < N; j ++) {
        Inv[i][j] = fpw(mod[i], mod[j] - 2, mod[j]);
    }
```

## 2.5 素性测试

```
int strong_pseudo_primetest(long long n,int base) {
    long long n2=n-1,res;
    int s=0;
    while(n2%2==0) n2>>=1,s++;
    res=powmod(base,n2,n);
    if((res==1)||(res==n-1)) return 1;
    s--;
    while(s>=0) {
        res=mulmod(res,res,n);
        if(res==n-1) return 1;
        s--;
    }
    return 0; // n is not a strong pseudo prime
}
int isprime(long long 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(strong_pseudo_primetest(n,testNum[i])==0) return 0;
    }
    return 1;
}
```

## 2.6 EX-BSGS

```
/*
 * a^x = b (mod p)
 * p may not be a prime
 */
unordered_map<int,int> mp;

ll exbsgs(ll a,ll b,ll p)
{
    if (b == 1) return 0;
    ll t, d = 1, k = 0;
    while ((t = __gcd(a, p)) != 1) {
        if (b % t) return -1;
        ++k, b /= t, p /= t, d = d * (a / t) % p;
        if (b == d) return k;
    }
    mp.clear();
    ll m = std::ceil(std::sqrt(p));
    ll a_m = powmod(a, m, p);
```

```
    ll mul = b;
    for (ll j = 1; j <= m; ++j) {
        mul = mul * a % p;
        mp[mul] = j;
    }
    for (ll i = 1; i <= m; ++i) {
        d = d * a_m % p;
        if (mp.count(d)) return i * m - mp[d] + k;
    }
    return -1;
}
```

## 2.7 质因数分解

```
int ansn; LL ans[1000];
LL func(LL x,LL n){ return(mod_mul(x,x,n)+1)%n; }
LL Pollard(LL n){
    LL i,x,y,p;
    if(Rabin_Miller(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;
    }
}
void factor(LL n){
    LL x;
    x=Pollard(n);
    if(x==n){ ans[ansn++]=x; return; }
    factor(x), factor(n/x);
}
```

## 2.8 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);
}
```

## 2.9 线性同余不等式

```
// Find the minimal non-negtive solutions for l ≤ d·x mod m ≤ r
// 0 ≤ d,l,r < m; l ≤ r, O(log n)
ll cal(ll m, ll d, ll l, ll r)
{
    if (l == 0) return 0;
    if (d == 0) return MXL; // 无解
    if (d * 2 > m) return cal(m, m - d, m - r, m - l);
    if ((l - 1) / d < r / d) return (l - 1) / d + 1;
    ll k = cal(d, (-m % d + d) % d, l % d, r % d);
    return k == MXL ? MXL : (k * m + l - 1) / d + 1; // 无解 2
}
```

## 2.10 原根相关

1. 模 $m$ 有原根的充要条件：$m = 2, 4, p^a, 2p^a$，其中 $p$ 是奇素数；
2. 求任意数 $p$ 原根的方法：对 $\phi(p)$ 因式分解，即 $\phi(p) = p_1^{r_1} p_2^{r_2} \cdots p_k^{r_k}$，若恒成立：

$$g^{\frac{p-1}{g}} \neq 1 \pmod p$$

   那么 $g$ 就是 $p$ 的原根。
3. 若模 $m$ 有原根，那么它一共有 $\Phi(\Phi(m))$ 个原根。

# Chapter 3

# 代数

## 3.1 快速傅里叶变换

```
int prepare(int n) {
    int len = 1;
    for (; len <= 2 * n; len <<= 1);
    for (int i = 0; i < len; i++) {
        e[0][i] = Complex(cos(2 * pi * i / len), sin(2 * pi * i / len));
        e[1][i] = Complex(cos(2 * pi * i / len), -sin(2 * pi * i / len));
    }
    return len;
}
void DFT(Complex *a, int n, int f) {
    for (int i = 0, j = 0; i < n; i++) {
        if (i > j) std::swap(a[i], a[j]);
        for (int t = n >> 1; (j ^= t) < t; t >>= 1);
    }
    for (int i = 2; i <= n; i <<= 1)
        for (int j = 0; j < n; j += i)
            for (int k = 0; k < (i >> 1); k++) {
                Complex A = a[j + k];
                Complex B = e[f][n / i * k] * a[j + k + (i >> 1)];
                a[j + k] = A + B;
                a[j + k + (i >> 1)] = A - B;
            }
    if (f == 1) {
        for (int i = 0; i < n; i++)
            a[i].a /= n;
    }
}
```

## 3.2 任意数快速傅里叶变换

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

typedef long double db;
typedef long long ll;
const int mask=(1<<15)-1;
const int maxn=100010;
const int maxl=262144;
const db pi=acos(-1.0);
const int mo=998244353;

struct Complex
{
    db r,i;
    Complex(db x=0.0,db y=0.0)
    {
        r=x,i=y;
    }
    Complex operator+(const Complex &o)    const
    {
        return Complex(r+o.r,i+o.i);
    }
    Complex operator-(const Complex &o)    const
    {
        return Complex(r-o.r,i-o.i);
    }
    Complex operator*(const Complex &o)    const
    {
        return Complex(r*o.r-i*o.i,r*o.i+i*o.r);
    }
    Complex conj()    const
    {
        return Complex(r,-i);
    }
}    A[maxl],B[maxl],C[maxl],D[maxl];

int a[maxn<<1],b[maxn<<1],c[maxn<<1];

int q;
void FFT(Complex *a,int n,int isdft)
{
    for (register int i=0,j=0;i<n;i++)
    {
        if (i>j)    swap(a[i],a[j]);
        for (int t=n>>1;(j^=t)<t;t>>=1);
    }
    for (int i=2;i<=n;i<<=1)
```

```
    {
        Complex wn(cos(isdft*2*pi/i),sin(isdft*2*pi/i));
        for (int j=0;j<n;j+=i)
        {
            Complex w(1,0),u,v;
            for (int k=j;k<j+i/2;k++)
            {
                u=a[k],v=a[k+i/2]*w;
                a[k]=u+v;
                a[k+i/2]=u-v;
                w=w*wn;
            }
        }
    }
    if (isdft==-1)
    for (int i=0;i<n;i++)    a[i].r/=n;
}
inline void cheng(int *x,int *y,int *z,int len)
{
//    len++;
    int l=1;
    q=0;
    while (l<len*2)    l*=2,q++;
    for (int i=0;i<len;i++)
        A[i]=Complex(x[i]>>15,x[i]&mask);
    for (int i=len;i<l;i++)
        A[i]=Complex(0,0);
    for (int i=0;i<len;i++)
        B[i]=Complex(y[i]>>15,y[i]&mask);
    for (int i=len;i<l;i++)
        B[i]=Complex(0,0);
    FFT(A,l,1);
    FFT(B,l,1);
    for (int i=0;i<l;i++)
    {
        int j=(l-i)%l;
        Complex _a=(A[i]-A[j].conj())*Complex(0,-0.5);
        Complex _b=(A[i]+A[j].conj())*Complex(0.5,0);
        Complex _c=(B[i]-B[j].conj())*Complex(0,-0.5);
        Complex _d=(B[i]+B[j].conj())*Complex(0.5,0);
        C[j]=_a*_d+_a*_c*Complex(0,1);
        D[j]=_b*_d+_b*_c*Complex(0,1);
    }
    FFT(C,l,1);
    FFT(D,l,1);
    for (int i=0;i<l;i++)
    {
        ll _a=((ll)(C[i].i/l+0.5))%mo;
        ll _b=((ll)(C[i].r/l+0.5))%mo;
```

```
        ll _c=((ll)(D[i].i/l+0.5))%mo;
        ll _d=((ll)(D[i].r/l+0.5))%mo;
        z[i]=((_d<<30)+((_b+_c)<<15)+_a)%mo;
    }
}
int main()
{
    int n,m;
    scanf("%d%d",&n,&m);
    n++,m++;
    for (int i=0;i<n;i++)
        scanf("%d",&a[i]);
    for (int i=0;i<m;i++)
        scanf("%d",&b[i]);
    cheng(a,b,c,max(n,m));
    for (int i=0;i<n+m-1;i++)
        printf("%d\n",c[i]);
    return 0;
}
```

## 3.3 fwt

```
/*
 xor(A) = (xor(A0+A1),xor(A0-A1))
_xor(A) = (_xor((A0+A1)/2),_xor((A0-A1)/2))

 and(A) = (and(A0+A1),and(A1))
_and(A) = (_and(A0-A1),_and(A1))

 or(A) = (or(A0),or(A0+A1))
_or(A) = (_or(A0),_or(A1-A0))
*/

void FWT(int *a,int n)
{
    for (int h=2;h<=n;h<<=1)
        for (int j=0;j<n;j+=h)
            for (int k=j;k<j+h/2;k++)
            {
                int u=a[k],v=a[k+h/2];
                //xor : a[k]=(u+v)%mo; a[k+h/2]=(u-v+mo)%mo;
                //and : a[k]=(u+v)%mo; a[k+h/2]=v;
                //or : a[k]=u; a[k+h/2]=(u+v)%mo;
            }
}
void IFWT(int *a,int n)
{
    for (int h=2;h<=n;h<<=1)
```

```
        for (int j=0;j<n;j+=h)
            for (int k=j;k<j+h/2;k++)
            {
                int u=a[k],v=a[k+h/2];
                //xor : a[k]=((u+v)*1LL*inv_2)%mo; a[k+h/2]=((u-v)*1LL*inv_2 %mo + mo)%mo;
                //and : a[k]=(u-v+mo)%mo; a[k+h/2]=v;
                //or : a[k]=u; a[k+h/2]=(u-v+mo)%mo;
            }
}
void cheng(int *a,int *b,int *c,int len)
{
    int l=1;
    while (l<len)    l<<=1;
    len=l;
    FWT(a,len);
    FWT(b,len);
    for (int i=0;i<len;i++)
        c[i]=(a[i]*1LL*b[i])%mo;
    IFWT(c,len);
}
```

## 3.4 快速数论变换

```
// meminit(A, l, r) 是将数组 A 的 [l, r) 清 0。
// memcopy(target, source, l, r) 是将 source 的 [l, r) 复制到 target 的 [l, r)
#define meminit(A, l, r) memset(A + (l), 0, sizeof(*A) * ((r) - (l)))
#define memcopy(B, A, l, r) memcpy(B, A + (l), sizeof(*A) * ((r) - (l)))
void DFT(int *a, int n, int f) { // 封闭形式，常数小（10^7 跑 2.23 秒）
    for (register int i = 0, j = 0; i < n; i++) {
        if (i > j) std::swap(a[i], a[j]);
        for (register int t = n >> 1; (j ^= t) < t; t >>= 1);
    }
    for (register int i = 2; i <= n; i <<= 1) {
        static int exp[MAXN];
        exp[0] = 1; exp[1] = fpm(PRT, (MOD - 1) / i);
        if (f == 1) exp[1] = fpm(exp[1], MOD - 2);
        for (register int k = 2; k < (i >> 1); k++) {
            exp[k] = 1ll * exp[k - 1] * exp[1] % MOD;
        }
        for (register int j = 0; j < n; j += i) {
            for (register int k = 0; k < (i >> 1); k++) {
                register int &pA = a[j + k], &pB = a[j + k + (i >> 1)];
                register int A = pA, B = 1ll * pB * exp[k] % MOD;
                pA = (A + B) % MOD;
                pB = (A - B + MOD) % MOD;
            }
        }
    }
```

```
    if (f == 1) {
        register int rev = fpm(n, MOD - 2, MOD);
        for (register int i = 0; i < n; i++) {
            a[i] = 1ll * a[i] * rev % MOD;
        }
    }
}
// 在不写高精度的情况下合并 FFT 所得结果对 MOD 取模后的答案
// 值得注意的是，这个东西不能最后再合并，而是应该每做一次多项式乘法就 CRT 一次
int CRT(int *a) {
    static int x[3];
    for (int i = 0; i < 3; i++) {
        x[i] = a[i];
        for (int j = 0; j < i; j++) {
            int t = (x[i] - x[j] + FFT[i] -> MOD) % FFT[i] -> MOD;
            if (t < 0) t += FFT[i] -> MOD;
            x[i] = 1LL * t * inv[j][i] % FFT[i] -> MOD;
        }
    }
    int sum = 1, ret = x[0] % MOD;
    for (int i = 1; i < 3; i ++) {
        sum = 1LL * sum * FFT[i - 1] -> MOD % MOD;
        ret += 1LL * x[i] * sum % MOD;
        if(ret >= MOD) ret -= MOD;
    }
    return ret;
}
for (int i = 0; i < 3; i++) // inv 数组的预处理过程，inverse(x, p) 表示求 x 在 p 下逆元
    for (int j = 0; j < 3; j++)
        inv[i][j] = inverse(FFT[i] -> MOD, FFT[j] -> MOD);
```

## 3.5 自适应辛普森积分

```
namespace adaptive_simpson {
    template<typename function>
    inline double area(function f, const double &left, const double &right) {
        double mid = (left + right) / 2;
        return (right - left) * (f(left) + 4 * f(mid) + f(right)) / 6;
    }

    template<typename function>
    inline double simpson(function f, const double &left, const double &right, const double
      &eps, const double &area_sum) {
        double mid = (left + right) / 2;
        double area_left = area(f, left, mid);
        double area_right = area(f, mid, right);
        double area_total = area_left + area_right;
        if (fabs(area_total - area_sum) <= 15 * eps) {
```

```
            return area_total + (area_total - area_sum) / 15;
        }
        return simpson(f, left, right, eps / 2, area_left) + simpson(f, mid, right, eps / 2,
    ↪ area_right);
    }

    template<typename function>
    inline double simpson(function f, const double &left, const double &right, const double
    ↪ &eps) {
        return simpson(f, left, right, eps, area(f, left, right));
    }
}
```

## 3.6 单纯形

```
const double eps = 1e-8;
// max{c * x | Ax <= b, x >= 0} 的解，无解返回空的 vector，否则就是解.
vector<double> simplex(vector<vector<double> > &A, vector<double> b, vector<double> c) {
    int n = A.size(), m = A[0].size() + 1, r = n, s = m - 1;
    vector<vector<double> > D(n + 2, vector<double>(m + 1));
    vector<int> ix(n + m);
    for(int i = 0; i < n + m; i++)  {
        ix[i] = i;
    }
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < m - 1; j++) {
            D[i][j] = -A[i][j];
        }
        D[i][m - 1] = 1;
        D[i][m] = b[i];
        if (D[r][m] > D[i][m]) {
            r = i;
        }
    }

    for(int j = 0; j < m - 1; j++) {
        D[n][j] = c[j];
    }
    D[n + 1][m - 1] = -1;
    for(double d; ;) {
        if (r < n) {
            swap(ix[s], ix[r + m]);
            D[r][s] = 1. / D[r][s];
            for(int j = 0; j <= m; j++) {
                if (j != s) {
                    D[r][j] *= -D[r][s];
                }
            }
```

```
            for(int i = 0; i <= n + 1; i++) {
                if (i != r) {
                    for(int j = 0; j <= m; j++) {
                        if (j != s) {
                            D[i][j] += D[r][j] * D[i][s];
                        }
                    }
                    D[i][s] *= D[r][s];
                }
            }
        }
        r = -1, s = -1;
        for(int j = 0; j < m; j++) {
            if (s < 0 || ix[s] > ix[j]) {
                if (D[n + 1][j] > eps || D[n + 1][j] > -eps && D[n][j] > eps) {
                    s = j;
                }
            }
        }
        if (s < 0) {
            break;
        }
        for(int i = 0; i < n; i++) {
            if (D[i][s] < -eps) {
                if (r < 0 || (d = D[r][m] / D[r][s] - D[i][m] / D[i][s]) < -eps
                    || d < eps && ix[r + m] > ix[i + m]) {

                    r = i;
                }
            }
        }

        if (r < 0) {
            return vector<double> ();
        }
    }
    if (D[n + 1][m] < -eps) {
        return vector<double> ();
    }

    vector<double> x(m - 1);
    for(int i = m; i < n + m; i++) {
        if (ix[i] < m - 1) {
            x[ix[i]] = D[i - m][m];
        }
    }
    return x;
}
```

# Chapter 4

# 字符串

## 4.1 后缀数组

```
const int MAXN = MAXL * 2 + 1;
int a[MAXN], x[MAXN], y[MAXN], c[MAXN], sa[MAXN], rank[MAXN], height[MAXN];
void calc_sa(int n) {
    int m = alphabet, k = 1;
    memset(c, 0, sizeof(*c) * (m + 1));
    for (int i = 1; i <= n; ++i) c[x[i] = a[i]]++;
    for (int i = 1; i <= m; ++i) c[i] += c[i - 1];
    for (int i = n; i; --i) sa[c[x[i]]--] = i;
    for (; k <= n; k <<= 1) {
        int tot = k;
        for (int i = n - k + 1; i <= n; ++i) y[i - n + k] = i;
        for (int i = 1; i <= n; ++i)
            if (sa[i] > k) y[++tot] = sa[i] - k;
        memset(c, 0, sizeof(*c) * (m + 1));
        for (int i = 1; i <= n; ++i) c[x[i]]++;
        for (int i = 1; i <= m; ++i) c[i] += c[i - 1];
        for (int i = n; i; --i) sa[c[x[y[i]]]--] = y[i];
        for (int i = 1; i <= n; ++i) y[i] = x[i];
        tot = 1; x[sa[1]] = 1;
        for (int i = 2; i <= n; ++i) {
            if (max(sa[i], sa[i - 1]) + k > n || y[sa[i]] != y[sa[i - 1]] || y[sa[i] + k] !=
              y[sa[i - 1] + k]) ++tot;
            x[sa[i]] = tot;
        }
        if (tot == n) break; else m = tot;
    }
}
void calc_height(int n) {
    for (int i = 1; i <= n; ++i) rank[sa[i]] = i;
    for (int i = 1; i <= n; ++i) {
        height[rank[i]] = max(0, height[rank[i - 1]] - 1);
        if (rank[i] == 1) continue;
        int j = sa[rank[i] - 1];
```

```
        while (max(i, j) + height[rank[i]] <= n && a[i + height[rank[i]]] == a[j + height[rank[i]]]) ++height[rank[i]];
    }
}
```

## 4.2 后缀自动机

```
static const int MAXL = MAXN * 2;  // MAXN is original length
static const int alphabet = 26;  // sometimes need changing
int l, last, cnt, trans[MAXL][alphabet], par[MAXL], sum[MAXL], seq[MAXL], mxl[MAXL], size[MAXL]; // mxl is maxlength, size is the size of right
char str[MAXL];
inline void init() {
    l = strlen(str + 1); cnt = last = 1;
    for (int i = 0; i <= l * 2; ++i) memset(trans[i], 0, sizeof(trans[i]));
    memset(par, 0, sizeof(*par) * (l * 2 + 1));
    memset(mxl, 0, sizeof(*mxl) * (l * 2 + 1));
    memset(size, 0, sizeof(*size) * (l * 2 + 1));
}
inline void extend(int pos, int c) {
    int p = last, np = last = ++cnt;
    mxl[np] = mxl[p] + 1; size[np] = 1;
    for (; p && !trans[p][c]; p = par[p]) trans[p][c] = np;
    if (!p) par[np] = 1;
    else {
        int q = trans[p][c];
        if (mxl[p] + 1 == mxl[q]) par[np] = q;
        else {
            int nq = ++cnt;
            mxl[nq] = mxl[p] + 1;
            memcpy(trans[nq], trans[q], sizeof(trans[nq]));
            par[nq] = par[q];
            par[np] = par[q] = nq;
            for (; trans[p][c] == q; p = par[p]) trans[p][c] = nq;
        }
    }
}
inline void buildsam() {
    for (int i = 1; i <= l; ++i) extend(i, str[i] - 'a');
    memset(sum, 0, sizeof(*sum) * (l * 2 + 1));
    for (int i = 1; i <= cnt; ++i) sum[mxl[i]]++;
    for (int i = 1; i <= l; ++i) sum[i] += sum[i - 1];
    for (int i = cnt; i; --i) seq[sum[mxl[i]]--] = i;
    for (int i = cnt; i; --i) size[par[seq[i]]] += size[seq[i]];
}
```

## 4.3 EX 后缀自动机

```
inline void add_node(int x, int &last) {
    int lastnode = last;
    if (c[lastnode][x]) {
        int nownode = c[lastnode][x];
        if (l[nownode] == l[lastnode] + 1) last = nownode;
        else {
            int auxnode = ++cnt; l[auxnode] = l[lastnode] + 1;
            for (int i = 0; i < alphabet; ++i) c[auxnode][i] = c[nownode][i];
            par[auxnode] = par[nownode]; par[nownode] = auxnode;
            for (; lastnode && c[lastnode][x] == nownode; lastnode = par[lastnode]) {
                c[lastnode][x] = auxnode;
            }
            last = auxnode;
        }
    } else {
        int newnode = ++cnt; l[newnode] = l[lastnode] + 1;
        for (; lastnode && !c[lastnode][x]; lastnode = par[lastnode]) c[lastnode][x] = newnode;
        if (!lastnode) par[newnode] = 1;
        else {
            int nownode = c[lastnode][x];
            if (l[lastnode] + 1 == l[nownode]) par[newnode] = nownode;
            else {
                int auxnode = ++cnt; l[auxnode] = l[lastnode] + 1;
                for (int i = 0; i < alphabet; ++i) c[auxnode][i] = c[nownode][i];
                par[auxnode] = par[nownode]; par[nownode] = par[newnode] = auxnode;
                for (; lastnode && c[lastnode][x] == nownode; lastnode = par[lastnode]) {
                    c[lastnode][x] = auxnode;
                }
            }
        }
        last = newnode;
    }
}
```

## 4.4 后缀树

1. 边上的字符区间是左闭右开区间；
2. 如果要建立关于多个串的后缀树，请用不同的分隔符，并且对于每个叶子结点，去掉和它父亲的连边上出现的第一个分隔符之后的所有字符；

## 4.5 回文自动机

```
int nT, nStr, last, c[MAXT][26], fail[MAXT], r[MAXN], l[MAXN], s[MAXN];
int allocate(int len) {
    l[nT] = len;
```

```
    r[nT] = 0;
    fail[nT] = 0;
    memset(c[nT], 0, sizeof(c[nT]));
    return nT++;
}
void init() {
    nT = nStr = 0;
    int newE = allocate(0);
    int newO = allocate(-1);
    last = newE;
    fail[newE] = newO;
    fail[newO] = newE;
    s[0] = -1;
}
void add(int x) {
    s[++nStr] = x;
    int now = last;
    while (s[nStr - l[now] - 1] != s[nStr]) now = fail[now];
    if (!c[now][x]) {
        int newnode = allocate(l[now] + 2), &newfail = fail[newnode];
        newfail = fail[now];
        while (s[nStr - l[newfail] - 1] != s[nStr]) newfail = fail[newfail];
        newfail = c[newfail][x];
        c[now][x] = newnode;
    }
    last = c[now][x];
    r[last]++;
}
void count() {
    for (int i = nT - 1; i >= 0; i--) {
        r[fail[i]] += r[i];
    }
}
```

## 4.6 回文自动机-mxh

```
struct PAM
{
    int trans[maxn][maxc];
    int fail[maxn];
    int cnt[maxn];//出现次数
    int num[maxn];
    int len[maxn];
    int s[maxn];
    int last,n,tot;
    int newnode(int l)
    {
        for (int i=0;i<maxc;i++)    trans[tot][i]=0;
```

```
        cnt[tot]=num[tot]=len[tot]=0;
        len[tot]=l;
        return tot++;
    }
    void clear()
    {
        last=n=tot=0;
        newnode(0);newnode(-1);
        s[0]=-1;fail[0]=1;
    }
    int get_fail(int x)
    {
        while (s[n-len[x]-1]!=s[n])    x=fail[x];
        return x;
    }
    void add(int c)
    {
        s[++n]=c;
        int cur=get_fail(last);
        if (!trans[cur][c])
        {
            int now=newnode(len[cur]+2);
            fail[now]=trans[get_fail(fail[cur])][c];
            trans[cur][c]=now;
            num[now]=num[fail[now]]+1;
        }
        last=trans[cur][c];
        cnt[last]++;
    }
    void count()
    {
        for (int i=tot-1;i>=0;i--)    cnt[fail[i]]+=cnt[i];
    }
}    pam;
```

# Chapter 5

# 数据结构

## 5.1 KD-Tree

```
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) {
    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);
    }
}
```

## 5.2 Treap

```
struct Node{
    int mn, key, size, tag;
    bool rev;
    Node* ch[2];
    Node(int mn, int key, int size): mn(mn), key(key), size(size), rev(0), tag(0){}
    void downtag();
    Node* update(){
        mn = min(ch[0] -> mn, min(key, ch[1] -> mn));
        size = ch[0] -> size + 1 + ch[1] -> size;
        return this;
    }
};
typedef pair<Node*, Node*> Pair;
Node *null, *root;
void Node::downtag(){
    if(rev){
        for(int i = 0; i < 2; i++)
            if(ch[i] != null){
                ch[i] -> rev ^= 1;
                swap(ch[i] -> ch[0], ch[i] -> ch[1]);
            }
        rev = 0;
    }
    if(tag){
        for(int i = 0; i < 2; i++)
            if(ch[i] != null){
                ch[i] -> key += tag;
                ch[i] -> mn += tag;
                ch[i] -> tag += tag;
            }
```

```
        tag = 0;
    }
}
int r(){
    static int s = 3023192386;
    return (s += (s << 3) + 1) & (~0u >> 1);
}
bool random(int x, int y){
    return r() % (x + y) < x;
}
Node* merge(Node *p, Node *q){
    if(p == null) return q;
    if(q == null) return p;
    p -> downtag();
    q -> downtag();
    if(random(p -> size, q -> size)){
        p -> ch[1] = merge(p -> ch[1], q);
        return p -> update();
    }else{
        q -> ch[0] = merge(p, q -> ch[0]);
        return q -> update();
    }
}
Pair split(Node *x, int n){
    if(x == null) return make_pair(null, null);
    x -> downtag();
    if(n <= x -> ch[0] -> size){
        Pair ret = split(x -> ch[0], n);
        x -> ch[0] = ret.second;
        return make_pair(ret.first, x -> update());
    }
    Pair ret = split(x -> ch[1], n - x -> ch[0] -> size - 1);
    x -> ch[1] = ret.first;
    return make_pair(x -> update(), ret.second);
}
pair<Node*, Pair> get_segment(int l, int r){
    Pair ret = split(root, l - 1);
    return make_pair(ret.first, split(ret.second, r - l + 1));
}
int main(){
    null = new Node(INF, INF, 0);
    null -> ch[0] = null -> ch[1] = null;
    root = null;
}
```

## 5.3 Link/cut Tree

```
inline void reverse(int x) {
    tr[x].rev ^= 1; swap(tr[x].c[0], tr[x].c[1]);
}

inline void rotate(int x, int k) {
    int y = tr[x].fa, z = tr[y].fa;
    tr[x].fa = z; tr[z].c[tr[z].c[1] == y] = x;
    tr[tr[x].c[k ^ 1]].fa = y; tr[y].c[k] = tr[x].c[k ^ 1];
    tr[x].c[k ^ 1] = y; tr[y].fa = x;
}

inline void splay(int x, int w) {
    int z = x; pushdown(x);
    while (tr[x].fa != w) {
        int y = tr[x].fa; z = tr[y].fa;
        if (z == w) {
            pushdown(z = y); pushdown(x);
            rotate(x, tr[y].c[1] == x);
            update(y); update(x);
        } else {
            pushdown(z); pushdown(y); pushdown(x);
            int t1 = tr[y].c[1] == x, t2 = tr[z].c[1] == y;
            if (t1 == t2) rotate(y, t2), rotate(x, t1);
            else rotate(x, t1), rotate(x, t2);
            update(z); update(y); update(x);
        }
    }
    update(x);
    if (x != z) par[x] = par[z], par[z] = 0;
}

inline void access(int x) {
    for (int y = 0; x; y = x, x = par[x]) {
        splay(x, 0);
        if (tr[x].c[1]) par[tr[x].c[1]] = x, tr[tr[x].c[1]].fa = 0;
        tr[x].c[1] = y; par[y] = 0; tr[y].fa = x; update(x);
    }
}

inline void makeroot(int x) {
    access(x); splay(x, 0); reverse(x);
}

inline void link(int x, int y) {
    makeroot(x); par[x] = y;
}

```

```
inline void cut(int x, int y) {
    access(x); splay(y, 0);
    if (par[y] != x) swap(x, y), access(x), splay(y, 0);
    par[y] = 0;
}

inline void split(int x, int y) {  // x will be the root of the tree
    makeroot(y); access(x); splay(x, 0);
}
```

## 5.4 树状数组查询第 k 小元素

```
int find(int k){
    int cnt=0,ans=0;
    for(int i=22;i>=0;i--){
        ans+=(1<<i);
        if(ans>n || cnt+d[ans]>=k)ans-=(1<<i);
        else cnt+=d[ans];
    }
    return ans+1;
}
```

# Chapter 6

# 图论

## 6.1 基础

```
struct Graph {  // Remember to call .init()!
    int e, nxt[M], v[M], adj[N], n;
    bool base;
    __inline void init(bool _base, int _n = 0) {
        assert(n < N);
        n = _n; base = _base;
        e = 0; memset(adj + base, -1, sizeof(*adj) * n);
    }
    __inline int new_node() {
        adj[n + base] = -1;
        assert(n + base + 1 < N);
        return n++ + base;
    }
    __inline void ins(int u0, int v0) {  // directional
        assert(u0 < n + base && v0 < n + base);
        v[e] = v0; nxt[e] = adj[u0]; adj[u0] = e++;
        assert(e < M);
    }
    __inline void bi_ins(int u0, int v0) {  // bi-directional
        ins(u0, v0); ins(v0, u0);
    }
};
```

## 6.2 KM

```
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];
```

```cpp
    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;
```

## 6.3 HK

```
int matchx[N], matchy[N], level[N];
vector<int> edge[N];
bool dfs(int x) {
    for (int i = 0; i < (int)edge[x].size(); ++i) {
        int y = edge[x][i];
        int w = matchy[y];
        if (w == -1 || level[x] + 1 == level[w] && dfs(w)) {
            matchx[x] = y;
            matchy[y] = x;
            return true;
        }
    }
    level[x] = -1;
    return false;
}
int solve() {
    memset(matchx, -1, sizeof(*matchx) * n);
    memset(matchy, -1, sizeof(*matchy) * m);
    for (int ans = 0; ; ) {
        std::vector<int> q;
        for (int i = 0; i < n; ++i) {
            if (matchx[i] == -1) {
                level[i] = 0;
                q.push_back(i);
            } else {
                level[i] = -1;
            }
        }
        for (int head = 0; head < (int)q.size(); ++head) {
            int x = q[head];
            for (int i = 0; i < (int)edge[x].size(); ++i) {
                int y = edge[x][i];
                int w = matchy[y];
                if (w != -1 && level[w] < 0) {
                    level[w] = level[x] + 1;
```

```
                    q.push_back(w);
                }
            }
        }
        int delta = 0;
        for (int i = 0; i < n; ++i) {
            if (matchx[i] == -1 && dfs(i)) {
                delta++;
            }
        }
        if (delta == 0) {
            return ans;
        } else {
            ans += delta;
        }
    }
}
```

## 6.4 点双连通分量

`bcc.forest` is a set of connected tree whose vertices are chequered with cut-vertex and BCC.

```
const bool BCC_VERTEX = 0, BCC_EDGE = 1;
struct BCC {  // N = N0 + M0. Remember to call init(&raw_graph).
    Graph *g, forest; // g is raw graph ptr.
    int dfn[N], DFN, low[N];
    int stack[N], top;
    int expand_to[N];       // Where edge i is expanded to in expaned graph.
    // Vertex i expaned to i.
    int compress_to[N];  // Where vertex i is compressed to.
    bool vertex_type[N], cut[N], compress_cut[N], branch[M];
    //std::vector<int> BCC_component[N];  // Cut vertex belongs to none.
    __inline void init(Graph *raw_graph) {
        g = raw_graph;
    }
    void DFS(int u, int pe) {
        dfn[u] = low[u] = ++DFN; cut[u] = false;
        if (!~g->adj[u]) {
            cut[u] = 1;
            compress_to[u] = forest.new_node();
            compress_cut[compress_to[u]] = 1;
        }
        for (int e = g->adj[u]; ~e; e = g->nxt[e]) {
            int v = g->v[e];
            if ((e ^ pe) > 1 && dfn[v] > 0 && dfn[v] < dfn[u]) {
                stack[top++] = e;
                low[u] = std::min(low[u], dfn[v]);
            }
            else if (!dfn[v]) {
```

```
                stack[top++] = e; branch[e] = 1;
                DFS(v, e);
                low[u] = std::min(low[v], low[u]);
                if (low[v] >= dfn[u]) {
                    if (!cut[u]) {
                        cut[u] = 1;
                        compress_to[u] = forest.new_node();
                        compress_cut[compress_to[u]] = 1;
                    }
                    int cc = forest.new_node();
                    forest.bi_ins(compress_to[u], cc);
                    compress_cut[cc] = 0;
                    //BCC_component[cc].clear();
                    do {
                        int cur_e = stack[--top];
                        compress_to[expand_to[cur_e]] = cc;
                        compress_to[expand_to[cur_e^1]] = cc;
                        if (branch[cur_e]) {
                            int v = g->v[cur_e];
                            if (cut[v])
                                forest.bi_ins(cc, compress_to[v]);
                            else {
                                //BCC_component[cc].push_back(v);
                                compress_to[v] = cc;
                            }
                        }
                    } while (stack[top] != e);
                }
            }
        }
    }
    void solve() {
        forest.init(g->base);
        int n = g->n;
        for (int i = 0; i < g->e; i++) {
            expand_to[i] = g->new_node();
        }
        memset(branch, 0, sizeof(*branch) * g->e);
        memset(dfn + g->base, 0, sizeof(*dfn) * n); DFN = 0;
        for (int i = 0; i < n; i++)
            if (!dfn[i + g->base]) {
                top = 0;
                DFS(i + g->base, -1);
            }
    }
} bcc;

bcc.init(&raw_graph);
bcc.solve();
```

```
// Do something with bcc.forest ...
```

## 6.5 边双连通分量

```
struct BCC {
    Graph *g, forest;
    int dfn[N], low[N], stack[N], tot[N], belong[N], vis[N], top, dfs_clock;
    // tot[] is the size of each BCC, belong[] is the BCC that each node belongs to
    pair<int, int > ori[M]; // bridge in raw_graph(raw node)
    bool is_bridge[M];
    __inline void init(Graph *raw_graph) {
        g = raw_graph;
        memset(is_bridge, false, sizeof(*is_bridge) * g -> e);
        memset(vis + g -> base, 0, sizeof(*vis) * g -> n);
    }
    void tarjan(int u, int from) {
        dfn[u] = low[u] = ++dfs_clock; vis[u] = 1; stack[++top] = u;
        for (int p = g -> adj[u]; ~p; p = g -> nxt[p]) {
            if ((p ^ 1) == from) continue;
            int v = g -> v[p];
            if (vis[v]) {
                if (vis[v] == 1) low[u] = min(low[u], dfn[v]);
            } else {
                tarjan(v, p);
                low[u] = min(low[u], low[v]);
                if (low[v] > dfn[u]) is_bridge[p / 2] = true;
            }
        }
        if (dfn[u] != low[u]) return;
        tot[forest.new_node()] = 0;
        do {
            belong[stack[top]] = forest.n;
            vis[stack[top]] = 2;
            tot[forest.n]++;
            --top;
        } while (stack[top + 1] != u);
    }
    void solve() {
        forest.init(g -> base);
        int n = g -> n;
        for (int i = 0; i < n; ++i)
            if (!vis[i + g -> base]) {
                top = dfs_clock = 0;
                tarjan(i + g -> base, -1);
            }
        for (int i = 0; i < g -> e / 2; ++i)
            if (is_bridge[i]) {
                int e = forest.e;
```

```
                forest.bi_ins(belong[g -> v[i * 2]], belong[g -> v[i * 2 + 1]], g -> w[i * 2]);
                ori[e] = make_pair(g -> v[i * 2 + 1], g -> v[i * 2]);
                ori[e + 1] = make_pair(g -> v[i * 2], g -> v[i * 2 + 1]);
            }
    }
} bcc;
```

## 6.6 最小树形图

```
const int MAXN,INF;// INF >= sum( W_ij )
int from[MAXN + 10][MAXN * 2 + 10],n,m,edge[MAXN + 10][MAXN * 2 + 10];
int sel[MAXN * 2 + 10],fa[MAXN * 2 + 10],vis[MAXN * 2 + 10];
int getfa(int x){if(x == fa[x]) return x; return fa[x] = getfa(fa[x]);}
void liuzhu(){ // 1-base: root is 1, answer = (sel[i], i) for i in [2..n]
    fa[1] = 1;
    for(int i = 2; i <= n; ++i){
        sel[i] = 1; fa[i] = i;
        for(int j = 1; j <= n; ++j) if(fa[j] != i)
            if(from[j][i] = i, edge[sel[i]][i] > edge[j][i]) sel[i] = j;
    }
    int limit = n;
    while(1){
        int prelimit = limit; memset(vis, 0, sizeof(vis)); vis[1] = 1;
        for(int i = 2; i <= prelimit; ++i) if(fa[i] == i && !vis[i]){
            int j = i; while(!vis[j]) vis[j] = i, j = getfa(sel[j]);
            if(j == 1 || vis[j] != i) continue; vector<int> C; int k = j;
            do C.push_back(k), k = getfa(sel[k]); while(k != j);
            ++limit;
            for(int i = 1; i <= n; ++i){
                edge[i][limit] = INF, from[i][limit] = limit;
            }
            fa[limit] = vis[limit] = limit;
            for(int i = 0; i < int(C.size()); ++i){
                int x = C[i], fa[x] = limit;
                for(int j = 1; j <= n; ++j)
                    if(edge[j][x] != INF && edge[j][limit] > edge[j][x] - edge[sel[x]][x]){
                        edge[j][limit] = edge[j][x] - edge[sel[x]][x];
                        from[j][limit] = x;
                    }
            }
            for(int j=1;j<=n;++j) if(getfa(j)==limit) edge[j][limit] = INF;
            sel[limit] = 1;
            for(int j = 1; j <= n; ++j)
                if(edge[sel[limit]][limit] > edge[j][limit]) sel[limit] = j;
        }
        if(prelimit == limit) break;
    }
    for(int i = limit; i > 1; --i) sel[from[sel[i]][i]] = sel[i];
```

```
}
```

## 6.7 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();
}
```

## 6.8 带权带花树

```
//maximum weight blossom,  change g[u][v].w to INF - g[u][v].w when minimum weight blossom is
  ↪ needed
//type of ans is long long
//replace all int to long long if weight of edge is long long

struct WeightGraph {
    static const int INF = INT_MAX;
    static const int MAXN = 400;
    struct edge{
        int u, v, w;
        edge() {}
        edge(int u, int v, int w): u(u), v(v), w(w) {}
    };
    int n, n_x;
    edge g[MAXN * 2 + 1][MAXN * 2 + 1];
    int lab[MAXN * 2 + 1];
    int match[MAXN * 2 + 1], slack[MAXN * 2 + 1], st[MAXN * 2 + 1], pa[MAXN * 2 + 1];
    int flower_from[MAXN * 2 + 1][MAXN+1], S[MAXN * 2 + 1], vis[MAXN * 2 + 1];
    vector<int> flower[MAXN * 2 + 1];
    queue<int> q;
    inline int e_delta(const edge &e){ // does not work inside blossoms
        return lab[e.u] + lab[e.v] - g[e.u][e.v].w * 2;
    }
    inline void update_slack(int u, int x){
        if(!slack[x] || e_delta(g[u][x]) < e_delta(g[slack[x]][x]))
```

```
            slack[x] = u;
    }
    inline void set_slack(int x){
        slack[x] = 0;
        for(int u = 1;u <= n; ++u)
            if(g[u][x].w > 0 && st[u] != x && S[st[u]] == 0)
                update_slack(u, x);
    }
    void q_push(int x){
        if(x <= n)q.push(x);
        else for(size_t i = 0;i < flower[x].size(); i++)
            q_push(flower[x][i]);
    }
    inline void set_st(int x, int b){
        st[x]=b;
        if(x > n) for(size_t i = 0;i < flower[x].size(); ++i)
                    set_st(flower[x][i], b);
    }
    inline int get_pr(int b, int xr){
        int pr = find(flower[b].begin(), flower[b].end(), xr) - flower[b].begin();
        if(pr % 2 == 1){
            reverse(flower[b].begin() + 1, flower[b].end());
            return (int)flower[b].size() - pr;
        } else return pr;
    }
    inline void set_match(int u, int v){
        match[u]=g[u][v].v;
        if(u > n){
            edge e=g[u][v];
            int xr = flower_from[u][e.u], pr=get_pr(u, xr);
            for(int i = 0;i < pr; ++i)
                set_match(flower[u][i], flower[u][i ^ 1]);
            set_match(xr, v);
            rotate(flower[u].begin(), flower[u].begin()+pr, flower[u].end());
        }
    }
    inline void augment(int u, int v){
        for(; ; ){
            int xnv=st[match[u]];
            set_match(u, v);
            if(!xnv)return;
            set_match(xnv, st[pa[xnv]]);
            u=st[pa[xnv]], v=xnv;
        }
    }
    inline int get_lca(int u, int v){
        static int t=0;
        for(++t; u || v; swap(u, v)){
            if(u == 0)continue;
```

```
            if(vis[u] == t)return u;
            vis[u] = t;
            u = st[match[u]];
            if(u) u = st[pa[u]];
        }
        return 0;
    }
    inline void add_blossom(int u, int lca, int v){
        int b = n + 1;
        while(b <= n_x && st[b]) ++b;
        if(b > n_x) ++n_x;
        lab[b] = 0, S[b] = 0;
        match[b] = match[lca];
        flower[b].clear();
        flower[b].push_back(lca);
        for(int x = u, y; x != lca; x = st[pa[y]]) {
            flower[b].push_back(x),
            flower[b].push_back(y = st[match[x]]),
            q_push(y);
        }
        reverse(flower[b].begin() + 1, flower[b].end());
        for(int x = v, y; x != lca; x = st[pa[y]]) {
            flower[b].push_back(x),
            flower[b].push_back(y = st[match[x]]),
            q_push(y);
        }
        set_st(b, b);
        for(int x = 1; x <= n_x; ++x) g[b][x].w = g[x][b].w = 0;
        for(int x = 1; x <= n; ++x) flower_from[b][x] = 0;
        for(size_t i = 0 ; i < flower[b].size(); ++i){
            int xs = flower[b][i];
            for(int x = 1; x <= n_x; ++x)
                if(g[b][x].w == 0 || e_delta(g[xs][x]) < e_delta(g[b][x]))
                    g[b][x] = g[xs][x], g[x][b] = g[x][xs];
            for(int x = 1;x <= n; ++x)
                if(flower_from[xs][x]) flower_from[b][x] = xs;
        }
        set_slack(b);
    }
    inline void expand_blossom(int b){ // S[b]  ==  1
        for(size_t i = 0; i < flower[b].size(); ++i)
            set_st(flower[b][i], flower[b][i]);
        int xr = flower_from[b][g[b][pa[b]].u], pr = get_pr(b, xr);
        for(int i = 0; i < pr; i += 2){
            int xs = flower[b][i], xns = flower[b][i + 1];
            pa[xs] = g[xns][xs].u;
            S[xs] = 1, S[xns] = 0;
            slack[xs] = 0, set_slack(xns);
            q_push(xns);
```

```
        }
        S[xr] = 1, pa[xr] = pa[b];
        for(size_t i = pr + 1;i < flower[b].size(); ++i){
            int xs = flower[b][i];
            S[xs] = -1, set_slack(xs);
        }
        st[b] = 0;
    }
    inline bool on_found_edge(const edge &e){
        int u = st[e.u], v = st[e.v];
        if(S[v] == -1){
            pa[v] = e.u, S[v] = 1;
            int nu = st[match[v]];
            slack[v] = slack[nu] = 0;
            S[nu] = 0, q_push(nu);
        }else if(S[v] == 0){
            int lca = get_lca(u, v);
            if(!lca) return augment(u, v), augment(v, u), true;
            else add_blossom(u, lca, v);
        }
        return false;
    }
    inline bool matching(){
        memset(S + 1, -1, sizeof(int) * n_x);
        memset(slack + 1, 0, sizeof(int) * n_x);
        q = queue<int>();
        for(int x = 1;x <= n_x; ++x)
            if(st[x] == x && !match[x]) pa[x]=0, S[x]=0, q_push(x);
        if(q.empty())return false;
        for(;;){
            while(q.size()){
                int u = q.front();q.pop();
                if(S[st[u]] == 1)continue;
                for(int v = 1;v <= n; ++v)
                    if(g[u][v].w > 0 && st[u] != st[v]){
                        if(e_delta(g[u][v]) == 0){
                            if(on_found_edge(g[u][v]))return true;
                        }else update_slack(u, st[v]);
                    }
            }
            int d = INF;
            for(int b = n + 1; b <= n_x;++b)
                if(st[b] == b && S[b] == 1)d = min(d, lab[b]/2);
            for(int x = 1; x <= n_x; ++x)
                if(st[x] == x && slack[x]){
                    if(S[x] == -1)d = min(d, e_delta(g[slack[x]][x]));
                    else if(S[x] == 0)d = min(d, e_delta(g[slack[x]][x])/2);
                }
            for(int u = 1; u <= n; ++u){
```

```
                if(S[st[u]] == 0){
                    if(lab[u] <= d)return 0;
                    lab[u] -= d;
                }else if(S[st[u]] == 1)lab[u] += d;
            }
            for(int b = n+1; b <= n_x; ++b)
                if(st[b] == b){
                    if(S[st[b]] == 0) lab[b] += d * 2;
                    else if(S[st[b]] == 1) lab[b] -= d * 2;
                }
            q=queue<int>();
            for(int x = 1; x <= n_x; ++x)
                if(st[x] == x && slack[x] && st[slack[x]] != x && e_delta(g[slack[x]][x]) == 0)
                    if(on_found_edge(g[slack[x]][x]))return true;
            for(int b = n + 1; b <= n_x; ++b)
                if(st[b] == b && S[b] == 1 && lab[b] == 0)expand_blossom(b);
        }
        return false;
    }
    inline pair<long long, int> solve(){
        memset(match + 1, 0, sizeof(int) * n);
        n_x = n;
        int n_matches = 0;
        long long tot_weight = 0;
        for(int u = 0; u <= n; ++u) st[u] = u, flower[u].clear();
        int w_max = 0;
        for(int u = 1; u <= n; ++u)
            for(int v = 1; v <= n; ++v){
                flower_from[u][v] = (u == v ? u : 0);
                w_max = max(w_max, g[u][v].w);
            }
        for(int u = 1; u <= n; ++u) lab[u] = w_max;
        while(matching()) ++n_matches;
        for(int u = 1; u <= n; ++u)
            if(match[u] && match[u] < u)
                tot_weight += g[u][match[u]].w;
        return make_pair(tot_weight, n_matches);
    }
    inline void init(){
        for(int u = 1; u <= n; ++u)
            for(int v = 1; v <= n; ++v)
                g[u][v]=edge(u, v, 0);
    }
};
```

## 6.9 Dominator Tree

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

```
            }
            if (num[v] < num[sem[u]]) {
                sem[u] = v;
            }
        }
        buf[sem[u]].push_back(u);
        mins[u] = u;
        merge(u, p);
        while (buf[p].size()) {
            int v = buf[p].back();
            buf[p].pop_back();
            find(v);
            if (sem[v] == sem[mins[v]]) {
                dom[v] = sem[v];
            } else {
                buf2[v] = mins[v];
            }
        }
    }
    dom[ord[0]] = ord[0];
    for (int i = 0; i < (int)ord.size(); ++i) {
        int u = ord[i];
        if (~buf2[u]) {
            dom[u] = dom[buf2[u]];
        }
    }
}
```

## 6.10 树 hash

```
const unsigned long long MAGIC = 4423;

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

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

```
    }
    for (int index = n - 1; index >= 0; --index) {
        int x = queue[index];
        hash[x] = std::make_pair(0, 0);

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

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

## 6.11 无向图最小割

```
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);
}
```

## 6.12 ISAP 最大流

```
const unsigned long long MAGIC = 4423;

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

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

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

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

```
            hash[x].first = hash[x].first * magic[value[i].second] + value[i].first;
            hash[x].second += value[i].second;
        }
        hash[x].first = hash[x].first * magic[1] + 41;
        hash[x].second++;
    }
}
```

## 6.13 重口味费用流

```
int S, T, totFlow, totCost;

int dis[N], slack[N], visit[N];

int modlable () {
    int delta = INF;
    for (int i = 1; i <= T; i++) {
        if (!visit[i] && slack[i] < delta) delta = slack[i];
        slack[i] = INF;
    }
    if (delta == INF) return 1;
    for (int i = 1; i <= T; i++)
        if (visit[i]) dis[i] += delta;
    return 0;
}

int dfs (int x, int flow) {
    if (x == T) {
        totFlow += flow;
        totCost += flow * (dis[S] - dis[T]);
        return flow;
    }
    visit[x] = 1;
    int left = flow;
    for (int i = e.last[x]; ~i; i = e.succ[i])
        if (e.cap[i] > 0 && !visit[e.other[i]]) {
            int y = e.other[i];
            if (dis[y] + e.cost[i] == dis[x]) {
                int delta = dfs (y, min (left, e.cap[i]));
                e.cap[i] -= delta;
                e.cap[i ^ 1] += delta;
                left -= delta;
                if (!left) { visit[x] = 0; return flow; }
            } else {
                slack[y] = min (slack[y], dis[y] + e.cost[i] - dis[x]);
            }
        }
    return flow - left;
```

```
}

pair <int, int> minCost () {
    totFlow = 0; totCost = 0;
    fill (dis + 1, dis + T + 1, 0);
    do {
        do {
            fill (visit + 1, visit + T + 1, 0);
        } while (dfs (S, INF));
    } while (!modlable ());
    return make_pair (totFlow, totCost);
}
```

## 6.14 2-SAT

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

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

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

bool solve() {
    int counter = n + n + 1;
    stamp = top = comps = 0;
    std::fill(dfn, dfn + counter, 0);
    std::fill(comp, comp + counter, 0);
```

```
    for (int i = 0; i < counter; ++i) {
        if (!dfn[i]) {
            tarjan(i);
        }
    }
    for (int i = 0; i < n; ++i) {
        if (comp[i << 1] == comp[i << 1 | 1]) {
            return false;
        }
        answer[i] = (comp[i << 1 | 1] < comp[i << 1]);
    }
    return true;
}
```

## 6.15 欧拉遍历

```
//从一个奇度点 dfs, sqn 即为回路/路径
//first 存点, second 存边的编号, 正反边编号一致
//清空 cur、used 数组
void getCycle(int u)
{
    for(int &i=cur[u]; i < (int)adj[u].size(); ++ i){
        int id = adj[u][i].second;
        if (used[id]) continue;
        used[id] = true;
        getCycle(adj[u][i].first);
    }
    sqn.push_back(u);
}
```

## 6.16 最大团搜索

```
/*
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;
    }
}
void print(){
    printf("%d\n", ans);
}
```

## 6.17 线性规划

```
void sieve(){
    f[1]=mu[1]=phi[1]=1;
    for(int i=2;i<maxn;i++){
        if(!minp[i]){
            minp[i]=i;
            minpw[i]=i;
            mu[i]=-1;
            phi[i]=i-1;
            f[i]=i-1;
            p[++p[0]]=i;//Case 1 prime
        }
        for(int j=1;j<=p[0]&&(LL)i*p[j]<maxn;j++){
            minp[i*p[j]]=p[j];
            if(i%p[j]==0){
                //Case 2 not coprime
                minpw[i*p[j]]=minpw[i]*p[j];
                phi[i*p[j]]=phi[i]*p[j];
                mu[i*p[j]]=0;
```

```
                if(i==minpw[i]){
                    f[i*p[j]]=i*p[j]-i;//Special Case for f(p^k)
                }else{
                    f[i*p[j]]=f[i/minpw[i]]*f[minpw[i]*p[j]];
                }
                break;
            }else{
                //Case 3 coprime
                minpw[i*p[j]]=p[j];
                f[i*p[j]]=f[i]*f[p[j]];
                phi[i*p[j]]=phi[i]*(p[j]-1);
                mu[i*p[j]]=-mu[i];
            }
        }
    }
}
```

# Chapter 7

# 其他

## 7.1 Dancing Links

```
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;
}
```

## 7.2 Dancing Links 可重覆盖

```
#include<bits/stdc++.h>
using namespace std;
const int maxnode = 51111, MaxN = 55555, MaxM = 55555;
int n, m;
struct DLX{
    int n,m,SIZE;
    int U[maxnode],D[maxnode],R[maxnode],L[maxnode],Row[maxnode],Col[maxnode];
    int H[MaxN],S[MaxM];
    int ansd, ans[MaxN];
    void init(int _n,int _m) {
        n = _n;
        m = _m;
        for (int i = 0; i <= m; ++i) {
            S[i] = 0;
            U[i] = D[i] = i;
            L[i] = i - 1;
            R[i] = i + 1;
        }
        R[m] = 0; L[0] = m;
```

```
        SIZE = m;
        for (int i = 1; i <= n; ++i) H[i] = -1;
    }
    void Link(int r,int c) {
        ++S[Col[++SIZE]=c];
        Row[SIZE] = r;
        D[SIZE] = D[c];
        U[D[c]] = SIZE;
        U[SIZE] = c;
        D[c] = SIZE;
        if (H[r] < 0) H[r] = L[SIZE] = R[SIZE] = SIZE;
        else {
            R[SIZE] = R[H[r]];
            L[R[H[r]]] = SIZE;
            L[SIZE] = H[r];
            R[H[r]] = SIZE;
        }
    }
    void repeat_remove(int c) {
        for (int i = D[c]; i != c; i = D[i])
            L[R[i]] = L[i], R[L[i]] = R[i];
    }
    void repeat_resume(int c) {
        for (int i = U[c]; i != c; i = U[i])
            L[R[i]] = R[L[i]] = i;
    }
    int f() {
        bool vv[MaxM];
        int ret = 0, c, i, j;
        for (c = R[0]; c != 0; c = R[c]) vv[c] = 1;
        for (c = R[0]; c != 0; c = R[c])
            if (vv[c]) {
                ++ret, vv[c] = 0;
                for (i = D[c]; i != c; i = D[i])
                    for (j = R[i]; j != i; j = R[j]) vv[Col[j]] = 0;
            }

        return ret;
    }
    void repeat_dance(int d) {
        if (d + f() >= ansd) return;
        if (R[0] == 0) {
            if (d < ansd) ansd = d;
            return;
        }
        int c = R[0], i, j;
        for (i = R[0]; i; i = R[i]) if (S[i] < S[c]) c = i;
        for (i = D[c]; i != c; i = D[i]) {
            repeat_remove(i);
```

```
            for (j = R[i]; j != i; j = R[j]) repeat_remove(j);
            repeat_dance(d + 1);
            for (j = L[i]; j != i; j = L[j]) repeat_resume(j);
            repeat_resume(i);
        }
    }
};
DLX g;
void work() {
    g.init(n, m);
    for (int i = 1; i <= m; ++i) {
        int x, y;
        scanf("%d%d", &x, &y);
        g.Link(x, i);
        g.Link(y, i);
    }
    g.ansd = n;
    g.repeat_dance(0);
    cout << g.ansd << endl;
}
int main() {
    while (~scanf("%d%d", &n, &m)) work();
    return 0;
}
```

## 7.3 蔡勒公式

0 for Sunday. Day and month is 1-based.

```
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);
}
```

## 7.4 蔡勒公式 new

0 for Sunday. Day and month is 1-based.

```
int zeller(int y,int m,int d)
{
    if (m<=2)    y--,m+=12;
    int c=y/100;
    y%=100;
    int w;
    if (y<1582 || (y==1582 && (m<10 || (m==10 && d<=4))))
        w=((c/4)-2*c+y+(y/4)+(13*(m+1)/5)+d-1)%7;
    else
```

```
        w=y+(y/4)+(c/4)-2*c+(13*(m+1)/5)+d+2;
    retur w;
}
```

# Chapter 8

# 技巧

## 8.1 真正的释放 STL 容器内存空间

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

## 8.2 无敌的大整数相乘取模

Time complexity $O(1)$.

```
// 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;
}
```

## 8.3 无敌的读入优化

```
// 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;
    }
}
```

## 8.4 梅森旋转算法

High quality pseudorandom number generator, twice as efficient as rand() with `-O2`. C++11 required.

```
#include <random>

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

# Chapter 9

# 提示

## 9.1 反演相关

$$\sigma_k(n) = \sum_{d|n} d^k = \prod_{i=1}^{\omega(n)} \frac{p_i^{(a_i+1)k} - 1}{p_i^k - 1}$$

$$J_k(n) = n^k \prod_{p|n} (1 - \frac{1}{p^k})$$

$J_k(n)$ is the number of $k$-tuples of positive integers all less than or equal to n that form a coprime $(k+1)$-tuple together with $n$.

$$\sum_{\delta|n} J_k(\delta) = n^k$$

$$\sum_{\delta|n} \delta^s J_r(\delta) J_s(\frac{n}{\delta}) = J_{r+s}(n)$$

$$\sum_{\delta|n} \varphi(\delta) d(\frac{n}{\delta}) = \sigma(n), \ \sum_{\delta|n} |\mu(\delta)| = 2^{\omega(n)}$$

$$\sum_{\delta|n} 2^{\omega(\delta)} = d(n^2), \ \sum_{\delta|n} d(\delta^2) = d^2(n)$$

$$\sum_{\delta|n} d(\frac{n}{\delta}) 2^{\omega(\delta)} = d^2(n), \ \sum_{\delta|n} \frac{\mu(\delta)}{\delta} = \frac{\varphi(n)}{n}$$

$$\sum_{\delta|n} \frac{\mu(\delta)}{\varphi(\delta)} = d(n), \ \sum_{\delta|n} \frac{\mu^2(\delta)}{\varphi(\delta)} = \frac{n}{\varphi(n)}$$

$$n|\varphi(a^n - 1)$$

$$\sum_{\substack{1 \le k \le n \\ \gcd(k,n)=1}} f(\gcd(k-1, n)) = \varphi(n) \sum_{d|n} \frac{(\mu * f)(d)}{\varphi(d)}$$

$$\varphi(\operatorname{lcm}(m,n)) \varphi(\gcd(m,n)) = \varphi(m)\varphi(n)$$

$$\sum_{\delta|n} d^3(\delta) = (\sum_{\delta|n} d(\delta))^2$$

$$d(uv) = \sum_{\delta|\gcd(u,v)} \mu(\delta) d(\frac{u}{\delta}) d(\frac{v}{\delta})$$

$$\sigma_k(u)\sigma_k(v) = \sum_{\delta|\gcd(u,v)} \delta^k \sigma_k(\frac{uv}{\delta^2})$$

$$\mu(n) = \sum_{k=1}^{n} [\gcd(k,n)=1] \cos 2\pi \frac{k}{n}$$

$$\varphi(n) = \sum_{k=1}^{n} [\gcd(k,n)=1] = \sum_{k=1}^{n} \gcd(k,n) \cos 2\pi \frac{k}{n}$$

$$\begin{cases} S(n) = \sum_{k=1}^{n} (f*g)(k) \\ \sum_{k=1}^{n} S(\lfloor \frac{n}{k} \rfloor) = \sum_{i=1}^{n} f(i) \sum_{j=1}^{\lfloor \frac{n}{i} \rfloor} (g*1)(j) \end{cases}$$

$$\begin{cases} S(n) = \sum_{k=1}^{n} (f \cdot g)(k), g \text{ completely multiplicative} \\ \sum_{k=1}^{n} S(\lfloor \frac{n}{k} \rfloor) g(k) = \sum_{k=1}^{n} (f*1)(k) g(k) \end{cases}$$

## 9.2 第二类斯特林数

$$\left\{ {n+1 \atop k} \right\} = k \left\{ {n \atop k} \right\} + \left\{ {n \atop k-1} \right\}$$

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

$$m! \left\{ {n \atop m} \right\} = \sum_k \binom{m}{k} k^n (-1)^{m-k}$$

## 9.3 控制 cout 输出实数精度

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

## 9.4 vimrc

```
set nu
set sw=4
set sts=4
set ts=4
```

```
syntax on
set cindent
```

## 9.5 让 make 支持 c ++ 11

In .bashrc or whatever:

```
export CXXFLAGS='-std=c++11 -Wall'
```

## 9.6 tuple 相关

```
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
```

## 9.7 线性规划转对偶

$$\begin{array}{l}\text{maximize } \mathbf{c}^T\mathbf{x} \\ \text{subject to } \mathbf{A}\mathbf{x} \le \mathbf{b}, \mathbf{x} \ge 0\end{array} \Longleftrightarrow \begin{array}{l}\text{minimize } \mathbf{y}^T\mathbf{b} \\ \text{subject to } \mathbf{y}^T\mathbf{A} \ge \mathbf{c}^T, \mathbf{y} \ge 0\end{array}$$

## 9.8 32-bit/64-bit 随机素数

| 32-bit | 64-bit |
|---|---|
| 73550053 | 1249292846855685773 |
| 148898719 | 1701750434419805569 |
| 189560747 | 3605499878424114901 |
| 459874703 | 5648316673387803781 |
| 1202316001 | 6125342570814357977 |
| 1431183547 | 6215155308775851301 |
| 1438011109 | 6294606778040623451 |
| 1538762023 | 6347330550446020547 |
| 1557944263 | 7429632924303725207 |
| 1981315913 | 8524720079480389849 |

## 9.9 NTT 素数及其原根

| Prime | Primitive root |
|---|---|
| 1053818881 | 7 |
| 1051721729 | 6 |
| 1045430273 | 3 |
| 1012924417 | 5 |
| 1007681537 | 3 |

## 9.10 Java Hints

```java
import java.io.*;
import java.lang.*;
```

```
import java.math.*;
import java.util.*;

/*    Regular usage:
        Slower IO :
            Scanner in = new Scanner (System.in);
            Scanner in = new Scanner (new BufferedInputStream (System.in));
            Input :
                in.nextInt () / in.nextBigInteger () / in.nextBigDecimal () / in.nextDouble ()
                in.nextLine () / in.hasNext ()
            Output :
                System.out.print (...);
                System.out.println (...);
                System.out.printf (...);
        Faster IO :
            Shown below.
        BigInteger :
            BigInteger.valueOf (int) : convert to BigInteger.
            abs / negate () / max / min / add / subtract / multiply /
                divide / remainder (BigInteger) : BigInteger algebraic.
            gcd (BigInteger) / modInverse (BigInteger mod) /
                modPow (BigInteger ex, BigInteger mod) / pow (int ex) : Number Theory.
            not () / and / or / xor (BigInteger) / shiftLeft / shiftRight (int) : Bit operation.
            compareTo (BigInteger) : comparation.
            intValue () / longValue () / toString (int radix) : converts to other types.
            isProbablePrime (int certainty) / nextProbablePrime () : checks primitive.
        BigDecimal :
            consists of a BigInteger value and a scale.
            The scale is the number of digits to the right of the decimal point.
            divide (BigDecimal) : exact divide.
            divide (BigDecimal, int scale, RoundingMode roundingMode) :
                divide with roundingMode, which may be:
                    CEILING / DOWN / FLOOR / HALF_DOWN / HALF_EVEN / HALF_UP / UNNECESSARY / UP.
            BigDecimal setScale (int newScale, RoundingMode roundingMode) :
                returns a BigDecimal with newScale.
            doubleValue () / toPlainString () : converts to other types.
        Arrays :
            Arrays.sort (T [] a);
            Arrays.sort (T [] a, int fromIndex, int toIndex);
            Arrays.sort (T [] a, int fromIndex, int toIndex, Comperator <? super T> comperator);
        LinkedList <E> :
            addFirst / addLast (E) / getFirst / getLast / removeFirst / removeLast () :
                deque implementation.
            clear () / add (int, E) / remove (int) : clear, add & remove.
            size () / contains / removeFirstOccurrence / removeLastOccurrence (E) :
                deque methods.
            ListIterator <E> listIterator (int index) : returns an iterator :
                E next / previous () : accesses and iterates.
                hasNext / hasPrevious () : checks availablity.
```

```
                nextIndex / previousIndex () : returns the index of a subsequent call.
                add / set (E) / remove () : changes element.
        PriorityQueue <E> (int initcap, Comparator <? super E> comparator) :
            add (E) / clear () / iterator () / peek () / poll () / size () :
                priority queue implementations.
        TreeMap <K, V> (Comparator <? super K> comparator) :
            Map.Entry <K, V> ceilingEntry / floorEntry / higherEntry / lowerEntry (K):
                getKey / getValue () / setValue (V) : entries.
            clear () / put (K, V) / get (K) / remove (K) : basic operation.
            size () : size.
        StringBuilder :
            Mutable string.
            StringBuilder (string) : generates a builder.
            append (int, string, ...) / insert (int offset, ...) : adds objects.
            charAt (int) / setCharAt (int, char) : accesses a char.
            delete (int, int) : removes a substring.
            reverse () : reverses itself.
            length () : returns the length.
            toString () : converts to string.
        String :
            Immutable string.
            String.format (String, ...) : formats a string. i.e. sprintf.
            toLowerCase / toUpperCase () : changes the case of letters.
*/

/* Examples on Comparator :
public class Main {
    public static class Point {
        public int x;
        public int y;
        public Point () {
            x = 0;
            y = 0;
        }
        public Point (int xx, int yy) {
            x = xx;
            y = yy;
        }
    };
    public static class Cmp implements Comparator <Point> {
        public int compare (Point a, Point b) {
            if (a.x < b.x) return -1;
            if (a.x == b.x) {
                if (a.y < b.y) return -1;
                if (a.y == b.y) return 0;
            }
            return 1;
        }
    };
```

```
    public static void main (String [] args) {
        Cmp c = new Cmp ();
        TreeMap <Point, Point> t = new TreeMap <Point, Point> (c);
        return;
    }
};
*/

/*   Another way to implement is to use Comparable.
    However, equalTo and hashCode must be rewritten.
    Otherwise, containers may fail.
    Example :
    public static class Point implements Comparable <Point> {
        public int x;
        public int y;
        public Point () {
            x = 0;
            y = 0;
        }
        public Point (int xx, int yy) {
            x = xx;
            y = yy;
        }
        public int compareTo (Point p) {
            if (x < p.x) return -1;
            if (x == p.x) {
                if (y < p.y) return -1;
                if (y == p.y) return 0;
            }
            return 1;
        }
        public boolean equalTo (Point p) {
            return (x == p.x && y == p.y);
        }
        public int hashCode () {
            return x + y;
        }
    };
*/

//Faster IO :

public class Main {

    static class InputReader {
        public BufferedReader reader;
        public StringTokenizer tokenizer;
        public InputReader (InputStream stream) {
            reader = new BufferedReader (new InputStreamReader (stream), 32768);
```

```java
            tokenizer = null;
        }
        public String next() {
            while (tokenizer == null || !tokenizer.hasMoreTokens()) {
                try {
                    String line = reader.readLine();
                    tokenizer = new StringTokenizer (line);
                } catch (IOException e) {
                    throw new RuntimeException (e);
                }
            }
            return tokenizer.nextToken();
        }
        public BigInteger nextBigInteger() {
            return new BigInteger (next (), 10);    //    customize the radix here.
        }
        public int nextInt() {
            return Integer.parseInt (next());
        }
        public double nextDouble() {
            return Double.parseDouble (next());
        }
    }

    public static void main (String[] args) {
        InputReader in = new InputReader (System.in);

        //    Put your code here.

    }
}

// Arrays
int a[];
.fill(a[, int fromIndex, int toIndex],val) | .sort(a[, int fromIndex, int toIndex])
// String
String s;
.charAt(int i) | compareTo(String) | compareToIgnoreCase () | contains(String) |
length () | substring(int l, int len)
// BigInteger
.abs() | .add() | bitLength () | subtract () | divide () | remainder () | divideAndRemainder () | modPow(b, c) |
pow(int) | multiply () | compareTo () |
gcd() | intValue () | longValue () | isProbablePrime(int c) (1 - 1/2^c) |
nextProbablePrime () | shiftLeft(int) | valueOf ()
// BigDecimal
.ROUND_CEILING | ROUND_DOWN_FLOOR | ROUND_HALF_DOWN | ROUND_HALF_EVEN | ROUND_HALF_UP | ROUND_UP
.divide(BigDecimal b, int scale , int round_mode) | doubleValue () | movePointLeft(int) | pow(int) |
```

```
setScale(int scale(精度) , int round_mode) | stripTrailingZeros ()
// StringBuilder
StringBuilder sb = new StringBuilder ();
sb.append(elem) | out.println(sb)
// Collection
.add(Object o) => boolean | .clear() | .isEmpty() | .contains() | .size() | .toArray() |
  ↪ .remove()
for (Object o: s) {}
// Set
Set s = new HashSet () | TreeSet ();
// List
List s = new LinkedList () | ArrayList ();
.add([int index,] E e) | .get (int index) | .indexOf (Object o) | .set (int index, E e) |
  ↪ .subList (int l, int r) | .remove (int index)
```

# Theoretical Computer Science Cheat Sheet

## Definitions

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

## Series

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

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:

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

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

Harmonic series:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

# Theoretical Computer Science Cheat Sheet

## Identities Cont.

**38.** $\left[{n+1 \atop m+1}\right] = \sum_k \left[{n \atop k}\right]\binom{k}{m} = \sum_{k=0}^{n} \left[{k \atop m}\right] n^{\underline{n-k}} = n! \sum_{k=0}^{n} \frac{1}{k!}\left[{k \atop m}\right],$ **39.** $\left[{x \atop x-n}\right] = \sum_{k=0}^{n} \left\langle\!\!\left\langle {n \atop k} \right\rangle\!\!\right\rangle \binom{x+k}{2n},$

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

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

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

**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"

$$\begin{aligned}
1\big(T(n) - 3T(n/2) = n\big) \\
3\big(T(n/2) - 3T(n/4) = n/2\big) \\
\vdots \quad \vdots \quad \vdots \\
3^{\log_2 n - 1}\big(T(2) - 3T(1) = 2\big)
\end{aligned}$$

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

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

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

## Pascal's Triangle

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

## General

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

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

Change of base, quadratic formula:

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

Euler's number $e$:

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

Harmonic numbers:

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

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

Factorial, Stirling's approximation:

1, 2, 6, 24, 120, 720, 5040, 40320, 362880, ...

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

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:

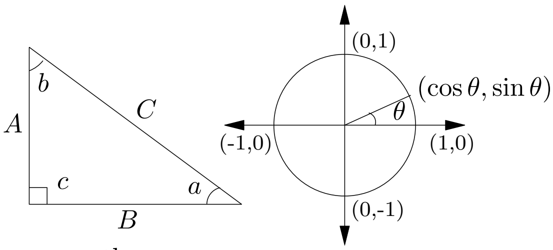
$$\Pr\left[|X| \geq \lambda\,\mathrm{E}[X]\right] \leq \frac{1}{\lambda},$$
$$\Pr\left[\left|X - \mathrm{E}[X]\right| \geq \lambda \cdot \sigma\right] \leq \frac{1}{\lambda^2}.$$

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:

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


v2.02 ©1994 by Steve Seiden
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.

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

$2 \times 2$ and $3 \times 3$ determinant:

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

Permanents:

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

## Hyperbolic Functions

Definitions:

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

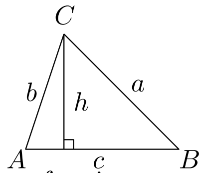
Identities:

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

| $\theta$ | $\sin\theta$ | $\cos\theta$ | $\tan\theta$ |
|---|---|---|---|
| $0$ | $0$ | $1$ | $0$ |
| $\frac{\pi}{6}$ | $\frac{1}{2}$ | $\frac{\sqrt{3}}{2}$ | $\frac{\sqrt{3}}{3}$ |
| $\frac{\pi}{4}$ | $\frac{\sqrt{2}}{2}$ | $\frac{\sqrt{2}}{2}$ | $1$ |
| $\frac{\pi}{3}$ | $\frac{\sqrt{3}}{2}$ | $\frac{1}{2}$ | $\sqrt{3}$ |
| $\frac{\pi}{2}$ | $1$ | $0$ | $\infty$ |

... 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 $\leq 5$.

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 $L_\infty$ metric:

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

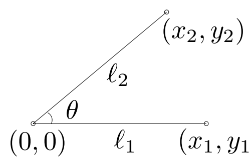
$$\left[|x_1 - x_0|^p + |y_1 - y_0|^p\right]^{1/p},$$

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

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

## $\pi$

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

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

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

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

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

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

## Partial Fractions

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

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

## Calculus

Derivatives:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Integrals:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**33.** $\int \sinh^2 x\,dx = \frac14\sinh(2x) - \frac12 x,$

**34.** $\int \cosh^2 x\,dx = \frac14\sinh(2x) + \frac12 x,$

**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.** $\int \frac{dx}{a^2+x^2} = \frac1a\arctan\frac{x}{a}, \quad a>0,$

**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.** $\int \frac{dx}{ax^2+bx} = \frac1a\ln\left|\frac{x}{a+bx}\right|,$

**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.** $\int \frac{x}{\sqrt{a+bx}}\,dx = \frac{1}{\sqrt2}\ln\left|\frac{\sqrt{a+bx}-\sqrt a}{\sqrt{a+bx}+\sqrt a}\right|, \quad a>0,$

**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.** $\int x\sqrt{a^2-x^2}\,dx = -\frac13(a^2-x^2)^{3/2},$

**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.** $\int \frac{dx}{\sqrt{a^2-x^2}} = -\frac1a\ln\left|\frac{a+\sqrt{a^2-x^2}}{x}\right|,$

**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.** $\int x\sqrt{x^2\pm a^2}\,dx = \frac13(x^2\pm a^2)^{3/2},$

**61.** $\int \frac{dx}{x\sqrt{x^2+a^2}} = \frac1a\ln\left|\frac{x}{a+\sqrt{a^2+x^2}}\right|,$