Standard Code Library

Shanghai Jiao Tong University

June 28, 2017

Contents

Chapter 1

计算几何

1.1 二维几何基础

```cpp
1  inline int sign(double x) { return x < -EPS ? -1 : x > EPS; }
2  inline double sqr(double x) { return x * x; }
3
4  struct point {
5    double x, y;
6    point(double x = 0, double y = 0) : x(x), y(y) {}
7    inline double length() const { return sqrt(x * x + y * y); }
8    inline double norm() const { return length(); }
9    inline double norm2() const { return x * x + y * y; }
10   inline point unit() const {
11     double len = length();
12     return point(x / len, y / len);
13   }
14   inline point negate() const { return point(-x, -y); }
15   inline point rot90() const {  // counter - clockwise
16     return point(-y, x);
17   }
18   inline point _rot90() const {  // clockwise
19     return point(y, -x);
20   }
21   inline point rotate(double theta) const {  // counter - clockwise
22     double c = cos(theta), s = sin(theta);
23     return point(x * c - y * s, x * s + y * c);
24   }
25   int get() { return scanf("%lf %lf", &x, &y); }
26   void out() { printf("(%.5f, %.5f)\n", x, y); }
27 };
28
29 inline bool operator==(const point &a, const point &b) {
30   return fabs(a.x - b.x) < EPS && fabs(a.y - b.y) < EPS;
31 }
32 inline bool operator!=(const point &a, const point &b) {
33   return fabs(a.x - b.x) > EPS || fabs(a.y - b.y) > EPS;
34 }
35 inline bool operator<(const point &a, const point &b) {
36   if (fabs(a.x - b.x) > EPS) return a.x < b.x;
37   return a.y + EPS < b.y;
38 }
39 inline point operator+(const point &a, const point &b) {
40   return point(a.x + b.x, a.y + b.y);
41 }
42 inline point operator-(const point &a, const point &b) {
43   return point(a.x - b.x, a.y - b.y);
44 }
45 inline point operator*(const point &a, const double &b) {
46   return point(a.x * b, a.y * b);
47 }
48 inline point operator/(const point &a, const double &b) {
49   return point(a.x / b, a.y / b);
50 }
51 inline double det(const point &a, const point &b) {
```

```
52       return a.x * b.y - b.x * a.y;
53     }
54     inline double dot(const point &a, const point &b) {
55       return a.x * b.x + a.y * b.y;
56     }
57     inline double dis(const point &a, const point &b) {
58       return sqrt(sqr(a.x - b.x) + sqr(a.y - b.y));
59     }
60     struct line {
61       point s, t;
62       line(point s = point(), point t = point()) : s(s), t(t) {}
63       inline double length() const { return dis(s, t); }
64     };
65     //线段交点
66     //注意如果两条线段是共线的且有交点，那么 intersect_judgement 确实会返回 true,
67     //但是 line_intersect 会求错，所以这种情况需要特判.
68     inline bool point_on_line(const point &a, const line &b) {
69       return sign(det(a - b.s, b.t - b.s)) == 0 && dot(b.s - a, b.t - a) < EPS;
70     }
71     inline bool two_side(const point &a, const point &b, const line &c) {
72       return sign(det(a - c.s, c.t - c.s)) * sign(det(b - c.s, c.t - c.s)) < 0;
73     }
74     inline bool intersect_judgement(const line &a, const line &b) {
75       if (point_on_line(b.s, a) || point_on_line(b.t, a)) return true;
76       if (point_on_line(a.s, b) || point_on_line(a.t, b)) return true;
77       return two_side(a.s, a.t, b) && two_side(b.s, b.t, a);
78     }
79     inline point line_intersect(const line &a, const line &b) {
80       double s1 = det(a.t - a.s, b.s - a.s);
81       double s2 = det(a.t - a.s, b.t - a.s);
82       return (b.s * s2 - b.t * s1) / (s2 - s1);
83     }
84     //点到直线的距离
85     double point_to_line(const point &p, const line &l) {
86       return fabs(det(l.t - l.s, p - l.s)) / dis(l.s, l.t);
87     }
88     inline double min_point_to_line(const point &a, const line &b) {
89       if (dot(b.s - a, b.t - a) < EPS)
90         return fabs(det(b.s - a, b.t - a) / b.length());
91       return min(dis(a, b.s), dis(a, b.t));
92     }
93     //点在多边形内
94     bool in_polygon(const point &p, const vector<point> &poly) {
95       int n = (int)poly.size();
96       int counter = 0;
97       for (int i = 0; i < n; ++i) {
98         point a = poly[i], b = poly[(i + 1) % n];
99         if (point_on_line(p, line(a, b))) return false;  // bounded excluded
100        int x = sign(det(p - a, b - a));
101        int y = sign(a.y - p.y);
102        int z = sign(b.y - p.y);
103        if (x > 0 && y <= 0 && z > 0) counter++;
104        if (x < 0 && z <= 0 && y > 0) counter--;
105      }
106      return counter != 0;
107    }
108    //点到直线的投影
109    point project_to_line(const point &p, const line &l) {
110      return l.s + (l.t - l.s) * (dot(p - l.s, l.t - l.s) / (l.t - l.s).norm2());
111    }
112    //圆类
113    struct circle {
114      point center;
115      double radius;
116      circle(point center = point(), double radius = 0)
117          : center(center), radius(radius) {}
118    };
119    inline bool operator==(const circle &a, const circle &b) {
120      return a.center == b.center && fabs(a.radius - b.radius) < EPS;
121    }
```

```
122  inline bool operator!=(const circle &a, const circle &b) {
123    return a.center != b.center || fabs(a.radius - b.radius) > EPS;
124  }
125  inline bool in_circle(const point &p, const circle &c) {
126    return dis(p, c.center) < c.radius + EPS;
127  }
128  //圆的生成函数
129  circle make_circle(const point &a, const point &b) {
130    return circle((a + b) / 2, dis(a, b) / 2);
131  }
132  circle make_circle(const point &a, const point &b, const point &c) {
133    point center = circumcenter(a, b, c);
134    return circle(center, dis(center, a));
135  }
136  //点到圆的切线
137  pair<line, line> tangent(const point &p, const circle &c) {
138    circle a = make_circle(p, c.center);
139    return make_pair(circle_intersect(a, c), circle_intersect(c, a));
140  }
141  //直线与圆的交点
142  //返回 AB 方向的第一个交点
143  point line_circle_intersect(const line &l, const circle &c) {
144    double x = sqrt(sqr(c.radius) - sqr(point_to_line(c.center, l)));
145    return project_to_line(c.center, l) + (l.s - l.t).unit() * x;
146  }
147  //圆与圆的交点
148  point circle_intersect(const circle &a, const circle &b) {  // get another point
149    using circle_intersect(b, a) point r = (b.center - a.center).unit();
150    double d = dis(a.center, b.center);
151    double x = .5 * ((sqr(a.radius) - sqr(b.radius)) / d + d);
152    double h = sqrt(sqr(a.radius) - sqr(x));
153    return a.center + r * x + r.rot90() * h;
154  }
```

1.2 快速凸包

```
1   //水平序凸包
2   inline bool turn_left(const point &a, const point &b, const point &c) {
3     return det(b - a, c - a) > EPS;
4   }
5   inline bool turn_right(const point &a, const point &b, const point &c) {
6     return det(b - a, c - a) < -EPS;
7   }
8   inline vector<point> convex_hull(vector<point> a) {
9     int n = (int)a.size(), cnt = 0;
10    sort(a.begin(), a.end());
11    vector<point> ret;
12    for (int i = 0; i < n; ++i) {
13      while (cnt > 1 && turn_left(ret[cnt - 2], a[i], ret[cnt - 1])) {
14        --cnt;
15        ret.pop_back();
16      }
17      ret.push_back(a[i]);
18      ++cnt;
19    }
20    int fixed = cnt;
21    for (int i = n - 1; i >= 0; --i) {
22      while (cnt > fixed && turn_left(ret[cnt - 2], a[i], ret[cnt - 1])) {
23        --cnt;
24        ret.pop_back();
25      }
26      ret.push_back(a[i]);
27      ++cnt;
28    }
29    // this algorithm will preserve the points which are collineation
30    // the lowest point will occur twice , i.e. ret.front () == ret.back ()
31    return ret;
32  }
```

1.3   半平面交

```
1   //半平面交
2   inline bool two_side(const point &a, const point &b, const line &c) {
3     return sign(det(a - c.s, c.t - c.s)) * sign(det(b - c.s, c.t - c.s)) < 0;
4   }
5   vector<point> cut(const vector<point> &c, line p) {
6     vector<point> ret;
7     if (c.empty()) return ret;
8     for (int i = 0; i < (int)c.size(); ++i) {
9       int j = (i + 1) % (int)c.size();
10      if (!turn_right(p.s, p.t, c[i])) ret.push_back(c[i]);
11      if (two_side(c[i], c[j], p))
12        ret.push_back(line_intersubsection(p, line(c[i], c[j])));
13    }
14    return ret;
15  }
16  static const double BOUND = 1e5;
17  /*
18  convex .clear ();
19  convex . push_back ( point (-BOUND , -BOUND ));
20  convex . push_back ( point (BOUND , -BOUND ));
21  convex . push_back ( point (BOUND , -BOUND ));
22  convex . push_back ( point (BOUND , -BOUND ));
23  convex = cut(convex , line(point , point));
24  Judgement : convex . empty ();
25  */
26  //高效半平面交
27  // plane[] 按照法向量 (逆时针 90 度) 极角排序, 去除平行半平面
28  inline bool turn_left(const line &l, const point &p) {
29    return turn_left(l.s, l.t, p);
30  }
31  vector<line> half_plane_intersect(const vector<line> &h) {
32    int fore = 0, rear = -1;
33    vector<line> ret;
34    for (int i = 0; i < (int)h.size(); ++i) {
35      while (fore < rear &&
36             !turn_left(h[i], line_intersect(ret[rear - 1], ret[rear])))
37        --rear;
38      while (fore < rear &&
39             !turn_left(h[i], line_intersect(ret[fore], ret[fore + 1])))
40        ++fore;
41      ++rear;
42      ret.push_back(h[i]);
43    }
44    while (rear - fore > 1 &&
45           !turn_left(ret[fore], line_intersect(ret[rear - 1], ret[rear])))
46      --rear;
47    while (rear - fore > 1 &&
48           !turn_left(ret[rear], line_intersect(ret[fore], ret[fore + 1])))
49      ++fore;
50    if (rear - fore < 2) return vector<line>();
51    return ret;
52  }
```

1.4   三角形的心

```
1   //三角形的内心
2   point incenter(const point &a, const point &b, const point &c) {
3     double p = (a - b).length() + (b - c).length() + (c - a).length();
4     return (a * (b - c).length() + b * (c - a).length() + c * (a - b).length()) /
5            p;
6   }
7   //三角形的外心
8   point circumcenter(const point &a, const point &b, const point &c) {
9     point p = b - a, q = c - a, s(dot(p, p) / 2, dot(q, q) / 2);
10    double d = det(p, q);
11    return a + point(det(s, point(p.y, q.y)), det(point(p.x, q.x), s)) / d;
12  }
```

```
13  //三角形的垂心
14  point orthocenter(const point &a, const point &b, const point &c) {
15    return a + b + c - circumcenter(a, b, c) * 2.0;
16  }
```

1.5 圆与多边形面积交

```
1   // 求扇形面积
2   double getSectorArea(const Point &a, const Point &b, const double &r) {
3     double c = (2.0 * r * r - sqrdist(a, b)) / (2.0 * r * r);
4     double alpha = acos(c);
5     return r * r * alpha / 2.0;
6   }
7   // 求二次方程 ax^2 + bx + c = 0 的解
8   std::pair<double, double> getSolution(const double &a, const double &b, const double &c) {
9     double delta = b * b - 4.0 * a * c;
10    if (dcmp(delta) < 0) return std::make_pair(0, 0);
11    else return std::make_pair((-b - sqrt(delta)) / (2.0 * a), (-b + sqrt(delta)) / (2.0 * a));
12  }
13  // 直线与圆的交点
14  std::pair<Point, Point> getIntersection(const Point &a, const Point &b, const double &r) {
15    Point d = b - a;
16    double A = dot(d, d);
17    double B = 2.0 * dot(d, a);
18    double C = dot(a, a) - r * r;
19    std::pair<double, double> s = getSolution(A, B, C);
20    return std::make_pair(a + d * s.first, a + d * s.second);
21  }
22  // 原点到线段 AB 的距离
23  double getPointDist(const Point &a, const Point &b) {
24    Point d = b - a;
25    int sA = dcmp(dot(a, d)), sB = dcmp(dot(b, d));
26    if (sA * sB <= 0) return det(a, b) / dist(a, b);
27    else return std::min(dist(a), dist(b));
28  }
29  // a 和 b 和原点组成的三角形与半径为 r 的圆的交的面积
30  double getArea(const Point &a, const Point &b, const double &r) {
31    double dA = dot(a, a), dB = dot(b, b), dC = getPointDist(a, b), ans = 0.0;
32    if (dcmp(dA - r * r) <= 0 && dcmp(dB - r * r) <= 0) return det(a, b) / 2.0;
33    Point tA = a / dist(a) * r;
34    Point tB = b / dist(b) * r;
35    if (dcmp(dC - r) > 0) return getSectorArea(tA, tB, r);
36    std::pair<Point, Point> ret = getIntersection(a, b, r);
37    if (dcmp(dA - r * r) > 0 && dcmp(dB - r * r) > 0) {
38      ans += getSectorArea(tA, ret.first, r);
39      ans += det(ret.first, ret.second) / 2.0;
40      ans += getSectorArea(ret.second, tB, r);
41      return ans;
42    }
43    if (dcmp(dA - r * r) > 0) return det(ret.first, b) / 2.0 + getSectorArea(tA, ret.first, r);
44    else return det(a, ret.second) / 2.0 + getSectorArea(ret.second, tB, r);
45  }
46  // 求圆与多边形的交的主过程
47  double getArea(int n, Point *p, const Point &c, const double r)  {
48    double ret = 0.0;
49    for (int i = 0; i < n; i++) {
50      int sgn = dcmp(det(p[i] - c, p[(i + 1) % n] - c));
51      if (sgn > 0) ret += getArea(p[i] - c, p[(i + 1) % n] - c, r);
52      else ret -= getArea(p[(i + 1) % n] - c, p[i] - c, r);
53    }
54    return fabs(ret);
55  }
```

1.6 圆并求面积

注意事项： 复杂度 $\mathcal{O}(n^2 \log n)$

```
1   struct arc {
2     double theta;
```

```
 3      int delta;
 4      point p;
 5      arc(){};
 6      arc(const double &theta, const point &p, int d)
 7          : theta(theta), p(p), delta(d) {}
 8    };
 9
10    vector<arc> vec;
11    vector<double> ans;
12    vector<point> center;
13    int cnt = 0;
14
15    inline bool operator<(const arc &a, const arc &b) {
16      return a.theta + EPS < b.theta;
17    }
18
19    inline void psh(const double t1, const point p1, const double t2,
20                    const point p2) {
21      if (t2 + EPS < t1) cnt++;
22      vec.push_back(arc(t1, p1, 1));
23      vec.push_back(arc(t2, p2, -1));
24    }
25
26    inline double cub(const double &x) { return x * x * x; }
27    inline void combine(int d, const double &area, const point &o) {
28      if (sign(area) == 0) return;
29      center[d] = (center[d] * ans[d] + o * area) * (1 / (ans[d] + area));
30      ans[d] += area;
31    }
32
33    void area(vector<circle> &cir) {
34      int n = cir.size();
35      vector<bool> f;
36      f.resize(n);
37      vec.clear();
38      cnt = 0;
39      for (int i = 0; i < n; i++) {
40        f[i] = true;
41        for (int j = 0; j < n; j++)
42          if (i != j) {
43            if ((cir[i] == cir[j] && i < j) ||
44                (cir[i] != cir[j] && cir[i].radius < cir[j].radius + EPS &&
45                 (cir[i].center - cir[j].center).length() <
46                     fabs(cir[i].radius - cir[j].radius) + EPS)) {
47              f[i] = false;
48              break;
49            }
50          }
51      }
52      int n1 = 0;
53      for (int i = 0; i < n; i++)
54        if (f[i]) cir[n1++] = cir[i];
55      n = n1;
56      ans.clear();
57      center.clear();
58      ans.resize(n + 1);
59      center.resize(n + 1);
60      point dvd;
61      for (int i = 0; i < n; i++) {
62        dvd = cir[i].center - point(cir[i].radius, 0);
63        vec.clear();
64        vec.push_back(arc(-PI, dvd, 1));
65        cnt = 0;
66        for (int j = 0; j < n; j++)
67          if (j != i) {
68            double d = (cir[j].center - cir[i].center).norm2();
69            if (d < sqr(cir[j].radius - cir[i].radius) + EPS) {
70              if (cir[i].radius + i * EPS < cir[j].radius + j * EPS)
71                psh(-PI, dvd, PI, dvd);
72            } else if (d + EPS < sqr(cir[j].radius + cir[i].radius)) {
```

```
73              double lambda =
74                  0.5 * (1 + (sqr(cir[i].radius) - sqr(cir[j].radius)) / d);
75            point cp(cir[i].center + (cir[j].center - cir[i].center) * lambda);
76            point nor((cir[j].center - cir[i].center)._rot90().unit() *
77                  (sqrt(sqr(cir[i].radius) - (cp - cir[i].center).norm2())));
78            point frm(cp + nor);
79            point to(cp - nor);
80            psh(atan2((frm - cir[i].center).y, (frm - cir[i].center).x), frm,
81                atan2((to - cir[i].center).y, (to - cir[i].center).x), to);
82          }
83        }
84      sort(vec.begin() + 1, vec.end());
85      vec.push_back(arc(PI, dvd, -1));
86      for (int j = 0; j + 1 < vec.size(); j++) {
87        cnt += vec[j].delta;
88        double theta(vec[j + 1].theta - vec[j].theta);
89        double area(sqr(cir[i].radius) * theta * 0.5);
90        combine(cnt, area, cir[i].center +
91                            point(sin(vec[j + 1].theta) - sin(vec[j].theta),
92                                cos(vec[j].theta) - cos(vec[j + 1].theta)) *
93                            (1. / area / 3 * cub(cir[i].radius)));
94        combine(cnt, -sqr(cir[i].radius) * sin(theta) * 0.5,
95                (cir[i].center + vec[j].p + vec[j + 1].p) / 3.);
96        combine(cnt, det(vec[j].p, vec[j + 1].p) * 0.5,
97                (vec[j].p + vec[j + 1].p) / 3.);
98      }
99    }
100 }
```

1.7 最小覆盖圆

```
1  circle minimum_circle(vector<point> p) {
2    circle ret;
3    random_shuffle(p.begin(), p.end());
4    for (int i = 0; i < (int)p.size(); ++i)
5      if (!in_circle(p[i], ret)) {
6        ret = circle(p[i], 0);
7        for (int j = 0; j < i; ++j)
8          if (!in_circle(p[j], ret)) {
9            ret = make_circle(p[j], p[i]);
10            for (int k = 0; k < j; ++k)
11              if (!in_circle(p[k], ret)) ret = make_circle(p[i], p[j], p[k]);
12          }
13      }
14    return ret;
15 }
```

1.8 最小覆盖球

```
1  double eps(1e-8);
2  int sign(const double & x) {
3    return (x > eps) - (x + eps < 0);
4  }
5  bool equal(const double & x, const double & y) {
6    return x + eps > y and y + eps > x;
7  }
8  struct Point {
9    double x, y, z;
10   Point() {
11   }
12   Point(const double & x, const double & y, const double & z) : x(x), y(y), z(z){
13   }
14   void scan() {
15     scanf("%lf%lf%lf", &x, &y, &z);
16   }
17   double sqrlen() const {
18     return x * x + y * y + z * z;
19   }
```

```
20      double len() const {
21        return sqrt(sqrlen());
22      }
23      void print() const {
24        printf("(%lf %lf %lf)\n", x, y, z);
25      }
26  } a[33];
27  Point operator + (const Point & a, const Point & b) {
28      return Point(a.x + b.x, a.y + b.y, a.z + b.z);
29  }
30  Point operator - (const Point & a, const Point & b) {
31      return Point(a.x - b.x, a.y - b.y, a.z - b.z);
32  }
33  Point operator * (const double & x, const Point & a) {
34      return Point(x * a.x, x * a.y, x * a.z);
35  }
36  double operator % (const Point & a, const Point & b) {
37      return a.x * b.x + a.y * b.y + a.z * b.z;
38  }
39  Point operator * (const Point & a, const Point & b) {
40      return Point(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);
41  }
42  struct Circle {
43      double r;
44      Point o;
45      Circle() {
46          o.x = o.y = o.z = r = 0;
47      }
48      Circle(const Point & o, const double & r) : o(o), r(r) {
49      }
50      void scan() {
51          o.scan();
52          scanf("%lf", &r);
53      }
54      void print() const {
55          o.print();
56          printf("%lf\n", r);
57      }
58  };
59  struct Plane {
60      Point nor;
61      double m;
62      Plane(const Point & nor, const Point & a) : nor(nor){
63          m = nor % a;
64      }
65  };
66  Point intersect(const Plane & a, const Plane & b, const Plane & c) {
67      Point c1(a.nor.x, b.nor.x, c.nor.x), c2(a.nor.y, b.nor.y, c.nor.y), c3(a.nor.z, b.nor.z,
    ↪  c.nor.z), c4(a.m, b.m, c.m);
68      return 1 / ((c1 * c2) % c3) * Point((c4 * c2) % c3, (c1 * c4) % c3, (c1 * c2) % c4);
69  }
70  bool in(const Point & a, const Circle & b) {
71      return sign((a - b.o).len() - b.r) <= 0;
72  }
73  bool operator < (const Point & a, const Point & b) {
74      if(!equal(a.x, b.x)) {
75          return a.x < b.x;
76      }
77      if(!equal(a.y, b.y)) {
78          return a.y < b.y;
79      }
80      if(!equal(a.z, b.z)) {
81          return a.z < b.z;
82      }
83      return false;
84  }
85  bool operator == (const Point & a, const Point & b) {
86      return equal(a.x, b.x) and equal(a.y, b.y) and equal(a.z, b.z);
87  }
88  vector<Point> vec;
```

```
89  Circle calc() {
90    if(vec.empty()) {
91      return Circle(Point(0, 0, 0), 0);
92    }else if(1 == (int)vec.size()) {
93      return Circle(vec[0], 0);
94    }else if(2 == (int)vec.size()) {
95      return Circle(0.5 * (vec[0] + vec[1]), 0.5 * (vec[0] - vec[1]).len());
96    }else if(3 == (int)vec.size()) {
97      double r((vec[0] - vec[1]).len() * (vec[1] - vec[2]).len() * (vec[2] - vec[0]).len() / 2 /
    ↪  fabs(((vec[0] - vec[2]) * (vec[1] - vec[2])).len())));
98      return Circle(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])),
99                Plane(vec[2] - vec[1], 0.5 * (vec[2] + vec[1])),
100          Plane((vec[1] - vec[0]) * (vec[2] - vec[0]), vec[0])), r);
101    }else {
102      Point o(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])),
103          Plane(vec[2] - vec[0], 0.5 * (vec[2] + vec[0])),
104          Plane(vec[3] - vec[0], 0.5 * (vec[3] + vec[0]))));
105      return Circle(o, (o - vec[0]).len());
106    }
107  }
108  Circle miniBall(int n) {
109    Circle res(calc());
110    for(int i(0); i < n; i++) {
111      if(!in(a[i], res)) {
112        vec.push_back(a[i]);
113        res = miniBall(i);
114        vec.pop_back();
115        if(i) {
116          Point tmp(a[i]);
117          memmove(a + 1, a, sizeof(Point) * i);
118          a[0] = tmp;
119        }
120      }
121    }
122    return res;
123  }
124  int main() {
125    int n;
126    for(;;) {
127      scanf("%d", &n);
128      if(!n) {
129        break;
130      }
131      for(int i(0); i < n; i++) {
132        a[i].scan();
133      }
134      sort(a, a + n);
135      n = unique(a, a + n) - a;
136      vec.clear();
137      printf("%.10f\n", miniBall(n).r);
138    }
139  }
```

1.9   三维几何基础

```
1  int dcmp(const double &x) {
2    return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1);
3  }
4
5  struct TPoint{
6    double x, y, z;
7    TPoint() {}
8    TPoint(double x, double y, double z) : x(x), y(y), z(z) {}
9    TPoint operator +(const TPoint &p)const {
10     return TPoint(x + p.x, y + p.y, z + p.z);
11   }
12   TPoint operator -(const TPoint &p)const {
13     return TPoint(x - p.x, y - p.y, z - p.z);
14   }
```

```
15      TPoint operator *(const double &p)const {
16        return TPoint(x * p, y * p, z * p);
17      }
18      TPoint operator /(const double &p)const {
19        return TPoint(x / p, y / p, z / p);
20      }
21      bool operator <(const TPoint &p)const {
22        int dX = dcmp(x - p.x), dY = dcmp(y - p.y), dZ = dcmp(z - p.z);
23        return dX < 0 || (dX == 0 && (dY < 0 || (dY == 0 && dZ < 0)));
24      }
25      bool read() {
26        return scanf("%lf%lf%lf", &x, &y, &z) == 3;
27      }
28    };
29
30    double sqrdist(const TPoint &a) {
31      double ret = 0;
32      ret += a.x * a.x;
33      ret += a.y * a.y;
34      ret += a.z * a.z;
35      return ret;
36    }
37    double sqrdist(const TPoint &a, const TPoint &b) {
38      double ret = 0;
39      ret += (a.x - b.x) * (a.x - b.x);
40      ret += (a.y - b.y) * (a.y - b.y);
41      ret += (a.z - b.z) * (a.z - b.z);
42      return ret;
43    }
44    double dist(const TPoint &a) {
45      return sqrt(sqrdist(a));
46    }
47    double dist(const TPoint &a, const TPoint &b) {
48      return sqrt(sqrdist(a, b));
49    }
50    TPoint det(const TPoint &a, const TPoint &b) {
51      TPoint ret;
52      ret.x = a.y * b.z - b.y * a.z;
53      ret.y = a.z * b.x - b.z * a.x;
54      ret.z = a.x * b.y - b.x * a.y;
55      return ret;
56    }
57    double dot(const TPoint &a, const TPoint &b) {
58      double ret = 0;
59      ret += a.x * b.x;
60      ret += a.y * b.y;
61      ret += a.z * b.z;
62      return ret;
63    }
64    double detdot(const TPoint &a, const TPoint &b, const TPoint &c, const TPoint &d) {
65      return dot(det(b - a, c - a), d - a);
66    }
```

1.10　三维凸包

```
1     struct Triangle{
2       TPoint a, b, c;
3       Triangle() {}
4       Triangle(TPoint a, TPoint b, TPoint c) : a(a), b(b), c(c) {}
5       double getArea() {
6         TPoint ret = det(b - a, c - a);
7         return dist(ret) / 2.0;
8       }
9     };
10    namespace Convex_Hull {
11      struct Face{
12        int a, b, c;
13        bool isOnConvex;
14        Face() {}
```

```cpp
15        Face(int a, int b, int c) : a(a), b(b), c(c) {}
16      };
17
18      int nFace, left, right, whe[MAXN][MAXN];
19      Face queue[MAXF], tmp[MAXF];
20
21      bool isVisible(const std::vector<TPoint> &p, const Face &f, const TPoint &a) {
22        return dcmp(detdot(p[f.a], p[f.b], p[f.c], a)) > 0;
23      }
24
25      bool init(std::vector<TPoint> &p) {
26        bool check = false;
27        for (int i = 1; i < (int)p.size(); i++) {
28          if (dcmp(sqrdist(p[0], p[i]))) {
29            std::swap(p[1], p[i]);
30            check = true;
31            break;
32          }
33        }
34        if (!check) return false;
35        check = false;
36        for (int i = 2; i < (int)p.size(); i++) {
37          if (dcmp(sqrdist(det(p[i] - p[0], p[1] - p[0])))) {
38            std::swap(p[2], p[i]);
39            check = true;
40            break;
41          }
42        }
43        if (!check) return false;
44        check = false;
45        for (int i = 3; i < (int)p.size(); i++) {
46          if (dcmp(detdot(p[0], p[1], p[2], p[i]))) {
47            std::swap(p[3], p[i]);
48            check = true;
49            break;
50          }
51        }
52        if (!check) return false;
53        for (int i = 0; i < (int)p.size(); i++)
54          for (int j = 0; j < (int)p.size(); j++) {
55            whe[i][j] = -1;
56          }
57        return true;
58      }
59
60      void pushface(const int &a, const int &b, const int &c) {
61        nFace++;
62        tmp[nFace] = Face(a, b, c);
63        tmp[nFace].isOnConvex = true;
64        whe[a][b] = nFace;
65        whe[b][c] = nFace;
66        whe[c][a] = nFace;
67      }
68
69      bool deal(const std::vector<TPoint> &p, const std::pair<int, int> &now, const TPoint &base) {
70        int id = whe[now.second][now.first];
71        if (!tmp[id].isOnConvex) return true;
72        if (isVisible(p, tmp[id], base)) {
73          queue[++right] = tmp[id];
74          tmp[id].isOnConvex = false;
75          return true;
76        }
77        return false;
78      }
79
80      std::vector<Triangle> getConvex(std::vector<TPoint> &p) {
81        static std::vector<Triangle> ret;
82        ret.clear();
83        if (!init(p)) return ret;
84        if (!isVisible(p, Face(0, 1, 2), p[3])) pushface(0, 1, 2); else pushface(0, 2, 1);
```

```
85      if (!isVisible(p, Face(0, 1, 3), p[2])) pushface(0, 1, 3); else pushface(0, 3, 1);
86      if (!isVisible(p, Face(0, 2, 3), p[1])) pushface(0, 2, 3); else pushface(0, 3, 2);
87      if (!isVisible(p, Face(1, 2, 3), p[0])) pushface(1, 2, 3); else pushface(1, 3, 2);
88      for (int a = 4; a < (int)p.size(); a++) {
89        TPoint base = p[a];
90        for (int i = 1; i <= nFace; i++) {
91          if (tmp[i].isOnConvex && isVisible(p, tmp[i], base)) {
92            left = 0, right = 0;
93            queue[++right] = tmp[i];
94            tmp[i].isOnConvex = false;
95            while (left < right) {
96              Face now = queue[++left];
97              if (!deal(p, std::make_pair(now.a, now.b), base)) pushface(now.a, now.b, a);
98              if (!deal(p, std::make_pair(now.b, now.c), base)) pushface(now.b, now.c, a);
99              if (!deal(p, std::make_pair(now.c, now.a), base)) pushface(now.c, now.a, a);
100           }
101           break;
102         }
103       }
104     }
105     for (int i = 1; i <= nFace; i++) {
106       Face now = tmp[i];
107       if (now.isOnConvex) {
108         ret.push_back(Triangle(p[now.a], p[now.b], p[now.c]));
109       }
110     }
111     return ret;
112   }
113 };
114
115 int n;
116 std::vector<TPoint> p;
117 std::vector<Triangle> answer;
118
119 int main() {
120   scanf("%d", &n);
121   for (int i = 1; i <= n; i++) {
122     TPoint a;
123     a.read();
124     p.push_back(a);
125   }
126   answer = Convex_Hull::getConvex(p);
127   double areaCounter = 0.0;
128   for (int i = 0; i < (int)answer.size(); i++) {
129     areaCounter += answer[i].getArea();
130   }
131   printf("%.3f\n", areaCounter);
132   return 0;
133 }
```

1.11   三维绕轴旋转

注意事项： 逆时针绕轴 AB 旋转 θ 角。

```
1  Matrix getTrans(const double &a, const double &b, const double &c) {
2      Matrix ret;
3      ret.a[0][0] = 1; ret.a[0][1] = 0; ret.a[0][2] = 0; ret.a[0][3] = 0;
4      ret.a[1][0] = 0; ret.a[1][1] = 1; ret.a[1][2] = 0; ret.a[1][3] = 0;
5      ret.a[2][0] = 0; ret.a[2][1] = 0; ret.a[2][2] = 1; ret.a[2][3] = 0;
6      ret.a[3][0] = a; ret.a[3][1] = b; ret.a[3][2] = c; ret.a[3][3] = 1;
7      return ret;
8  }
9  Matrix getRotate(const double &a, const double &b, const double &c, const double &theta) {
10     Matrix ret;
11     ret.a[0][0] = a * a * (1 - cos(theta)) + cos(theta);
12     ret.a[0][1] = a * b * (1 - cos(theta)) + c * sin(theta);
13     ret.a[0][2] = a * c * (1 - cos(theta)) - b * sin(theta);
14     ret.a[0][3] = 0;
15
16     ret.a[1][0] = b * a * (1 - cos(theta)) - c * sin(theta);
```

```
17      ret.a[1][1] = b * b * (1 - cos(theta)) + cos(theta);
18      ret.a[1][2] = b * c * (1 - cos(theta)) + a * sin(theta);
19      ret.a[1][3] = 0;
20
21      ret.a[2][0] = c * a * (1 - cos(theta)) + b * sin(theta);
22      ret.a[2][1] = c * b * (1 - cos(theta)) - a * sin(theta);
23      ret.a[2][2] = c * c * (1 - cos(theta)) + cos(theta);
24      ret.a[2][3] = 0;
25
26      ret.a[3][0] = 0;
27      ret.a[3][1] = 0;
28      ret.a[3][2] = 0;
29      ret.a[3][3] = 1;
30      return ret;
31  }
32  Matrix getRotate(const double &ax, const double &ay, const double &az, const double &bx, const
    ↪   double &by, const double &bz, const double &theta) {
33      double l = dist(Point(0, 0, 0), Point(bx, by, bz));
34      Matrix ret = getTrans(-ax, -ay, -az);
35      ret = ret * getRotate(bx / l, by / l, bz / l, theta);
36      ret = ret * getTrans(ax, ay, az);
37      return ret;
38  }
```

1.12 Delaunay 三角剖分

```
1   /*
2   Delaunay Triangulation 随机增量算法 :
3   节点数至少为点数的 6 倍，空间消耗较大注意计算内存使用
4   建图的过程在 build 中，注意初始化内存池和初始三角形的坐标范围 (Triangulation::LOTS)
5   Triangulation::find 返回包含某点的三角形
6   Triangulation::add_point 将某点加入三角剖分
7   某个 Triangle 在三角剖分中当且仅当它的 has_children 为 0
8   如果要找到三角形 u 的邻域，则枚举它的所有 u.edge[i].tri，该条边的两个点为 u.p[(i+1)%3], u.p[(i+2)%3]
9   */
10  const int N = 100000 + 5, MAX_TRIS = N * 6;
11  const double EPSILON = 1e-6, PI = acos(-1.0);
12  struct Point {
13    double x,y; Point():x(0),y(0){} Point(double x, double y):x(x),y(y){}
14    bool operator ==(Point const& that)const {return x==that.x&&y==that.y;}
15  };
16  inline double sqr(double x) { return x*x; }
17  double dist_sqr(Point const& a, Point const& b){return sqr(a.x-b.x)+sqr(a.y-b.y);}
18  bool in_circumcircle(Point const& p1, Point const& p2, Point const& p3, Point const& p4) {
19    double u11 = p1.x - p4.x, u21 = p2.x - p4.x, u31 = p3.x - p4.x;
20    double u12 = p1.y - p4.y, u22 = p2.y - p4.y, u32 = p3.y - p4.y;
21    double u13 = sqr(p1.x) - sqr(p4.x) + sqr(p1.y) - sqr(p4.y);
22    double u23 = sqr(p2.x) - sqr(p4.x) + sqr(p2.y) - sqr(p4.y);
23    double u33 = sqr(p3.x) - sqr(p4.x) + sqr(p3.y) - sqr(p4.y);
24    double det = -u13*u22*u31 + u12*u23*u31 + u13*u21*u32 - u11*u23*u32 - u12*u21*u33 +
    ↪   u11*u22*u33;
25    return det > EPSILON;
26  }
27  double side(Point const& a, Point const& b, Point const& p) { return (b.x-a.x)*(p.y-a.y) -
    ↪   (b.y-a.y)*(p.x-a.x);}
28  typedef int SideRef; struct Triangle; typedef Triangle* TriangleRef;
29  struct Edge {
30    TriangleRef tri; SideRef side; Edge() : tri(0), side(0) {}
31    Edge(TriangleRef tri, SideRef side) : tri(tri), side(side) {}
32  };
33  struct Triangle {
34    Point p[3]; Edge edge[3]; TriangleRef children[3]; Triangle() {}
35    Triangle(Point const& p0, Point const& p1, Point const& p2) {
36      p[0]=p0;p[1]=p1;p[2]=p2;children[0]=children[1]=children[2]=0;
37    }
38    bool has_children() const { return children[0] != 0; }
39    int num_children() const {
40      return children[0] == 0 ? 0
41        : children[1] == 0 ? 1
```

```
42          : children[2] == 0 ? 2 : 3;
43    }
44    bool contains(Point const& q) const {
45      double a=side(p[0],p[1],q), b=side(p[1],p[2],q), c=side(p[2],p[0],q);
46      return a >= -EPSILON && b >= -EPSILON && c >= -EPSILON;
47    }
48  } triange_pool[MAX_TRIS], *tot_triangles;
49  void set_edge(Edge a, Edge b) {
50    if (a.tri) a.tri->edge[a.side] = b;
51    if (b.tri) b.tri->edge[b.side] = a;
52  }
53  class Triangulation {
54    public:
55      Triangulation() {
56        const double LOTS = 1e6;
57        the_root = new(tot_triangles++)
  ↪  Triangle(Point(-LOTS,-LOTS),Point(+LOTS,-LOTS),Point(0,+LOTS));
58      }
59      TriangleRef find(Point p) const { return find(the_root,p); }
60      void add_point(Point const& p) { add_point(find(the_root,p),p); }
61    private:
62      TriangleRef the_root;
63      static TriangleRef find(TriangleRef root, Point const& p) {
64        for( ; ; ) {
65          if (!root->has_children()) return root;
66          else for (int i = 0; i < 3 && root->children[i] ; ++i)
67              if (root->children[i]->contains(p))
68                {root = root->children[i]; break;}
69        }
70      }
71      void add_point(TriangleRef root, Point const& p) {
72        TriangleRef tab,tbc,tca;
73        tab = new(tot_triangles++) Triangle(root->p[0], root->p[1], p);
74        tbc = new(tot_triangles++) Triangle(root->p[1], root->p[2], p);
75        tca = new(tot_triangles++) Triangle(root->p[2], root->p[0], p);
76        set_edge(Edge(tab,0),Edge(tbc,1));set_edge(Edge(tbc,0),Edge(tca,1));
77        set_edge(Edge(tca,0),Edge(tab,1));set_edge(Edge(tab,2),root->edge[2]);
78        set_edge(Edge(tbc,2),root->edge[0]);set_edge(Edge(tca,2),root->edge[1]);
79        root->children[0]=tab;root->children[1]=tbc;root->children[2]=tca;
80        flip(tab,2); flip(tbc,2); flip(tca,2);
81      }
82      void flip(TriangleRef tri, SideRef pi) {
83        TriangleRef trj = tri->edge[pi].tri; int pj = tri->edge[pi].side;
84        if(!trj||!in_circumcircle(tri->p[0],tri->p[1],tri->p[2],trj->p[pj])) return;
85        TriangleRef trk = new(tot_triangles++) Triangle(tri->p[(pi+1)%3], trj->p[pj], tri->p[pi]);
86        TriangleRef trl = new(tot_triangles++) Triangle(trj->p[(pj+1)%3], tri->p[pi], trj->p[pj]);
87        set_edge(Edge(trk,0), Edge(trl,0));
88        set_edge(Edge(trk,1), tri->edge[(pi+2)%3]); set_edge(Edge(trk,2), trj->edge[(pj+1)%3]);
89        set_edge(Edge(trl,1), trj->edge[(pj+2)%3]); set_edge(Edge(trl,2), tri->edge[(pi+1)%3]);
90        tri->children[0]=trk;tri->children[1]=trl;tri->children[2]=0;
91        trj->children[0]=trk;trj->children[1]=trl;trj->children[2]=0;
92        flip(trk,1); flip(trk,2); flip(trl,1); flip(trl,2);
93      }
94  };
95  int n; Point ps[N];
96  void build(){
97    tot_triangles = triange_pool; cin >> n;
98    for(int i = 0; i < n; ++ i) scanf("%lf%lf",&ps[i].x,&ps[i].y);
99    random_shuffle(ps, ps + n); Triangulation tri;
100   for(int i = 0; i < n; ++ i) tri.add_point(ps[i]);
101 }
```

Chapter 2

图论

2.1 点双连通分量

```
1   // 坚固的点双连通分量
2   int n, m, x, y, ans1, ans2, tot1, tot2, flag, size, ind2, dfn[N], low[N], block[M], vis[N];
3   vector<int> a[N];
4   pair<int, int> stack[M];
5   void tarjan(int x, int p) {
6     dfn[x] = low[x] = ++ind2;
7     for (int i = 0; i < a[x].size(); ++i)
8       if (dfn[x] > dfn[a[x][i]] && a[x][i] != p){
9         stack[++size] = make_pair(x, a[x][i]);
10        if (i == a[x].size() - 1 || a[x][i] != a[x][i + 1])
11          if (!dfn[a[x][i]]){
12            tarjan(a[x][i], x);
13            low[x] = min(low[x], low[a[x][i]]);
14            if (low[a[x][i]] >= dfn[x]){
15              tot1 = tot2 = 0;
16              ++flag;
17              for (; ; ){
18                if (block[stack[size].first] != flag) {
19                  ++tot1;
20                  block[stack[size].first] = flag;
21                }
22                if (block[stack[size].second] != flag) {
23                  ++tot1;
24                  block[stack[size].second] = flag;
25                }
26                if (stack[size].first == x && stack[size].second == a[x][i])
27                  break;
28                ++tot2;
29                --size;
30              }
31              for (; stack[size].first == x && stack[size].second == a[x][i]; --size)
32                ++tot2;
33              if (tot2 < tot1)
34                ans1 += tot2;
35              if (tot2 > tot1)
36                ans2 += tot2;
37            }
38          }
39          else
40            low[x] = min(low[x], dfn[a[x][i]]);
41      }
42  }
43  int main(){
44    for (; ; ){
45      scanf("%d%d", &n, &m);
46      if (n == 0 && m == 0) return 0;
47      for (int i = 1; i <= n; ++i) {
48        a[i].clear();
49        dfn[i] = 0;
50      }
51      for (int i = 1; i <= m; ++i){
```

```
52        scanf("%d%d",&x, &y);
53        ++x, ++y;
54        a[x].push_back(y);
55        a[y].push_back(x);
56      }
57      for (int i = 1; i <= n; ++i)
58        sort(a[i].begin(), a[i].end());
59      ans1 = ans2 = ind2 = 0;
60      for (int i = 1; i <= n; ++i)
61        if (!dfn[i]) {
62          size = 0;
63          tarjan(i, 0);
64        }
65      printf("%d %d\n", ans1, ans2);
66    }
67    return 0;
68  }
69  // 朴素的点双连通分量
70  void tarjan(int x){
71    dfn[x] = low[x] = ++ind2;
72    v[x] = 1;
73    for (int i = nt[x]; pt[i]; i = nt[i])
74      if (!dfn[pt[i]]){
75        tarjan(pt[i]);
76        low[x] = min(low[x], low[pt[i]]);
77        if (dfn[x] <= low[pt[i]])
78          ++v[x];
79      }
80      else
81        low[x] = min(low[x], dfn[pt[i]]);
82  }
83  int main(){
84    for (; ; ){
85      scanf("%d%d", &n, &m);
86      if (n == 0 && m == 0)
87        return 0;
88      for (int i = 1; i <= ind; ++i)
89        nt[i] = pt[i] = 0;
90      ind = n;
91      for (int i = 1; i <= ind; ++i)
92        last[i] = i;
93      for (int i = 1; i <= m; ++i){
94        scanf("%d%d", &x, &y);
95        ++x, ++y;
96        edge(x, y), edge(y, x);
97      }
98      memset(dfn, 0, sizeof(dfn));
99      memset(v, 0, sizeof(v));
100     ans = num = ind2 = 0;
101     for (int i = 1; i <= n; ++i)
102       if (!dfn[i]){
103         root = i;
104         size = 0;
105         ++num;
106         tarjan(i);
107         --v[root];
108       }
109     for (int i = 1; i <= n; ++i)
110       if (v[i] + num - 1 > ans)
111         ans = v[i] + num - 1;
112     printf("%d\n",ans);
113   }
114   return 0;
115 }
```

2.2   Hungary 求最大匹配

```
1  int n, m, stamp;
2  int match[N], visit[N];
```

```
3
4   bool dfs(int x) {
5       for (int i = 0; i < (int)edge[x].size(); ++i) {
6           int y = edge[x][i];
7           if (visit[y] != stamp) {
8               visit[y] = stamp;
9               if (match[y] == -1 || dfs(match[y])) {
10                  match[y] = x;
11                  return true;
12              }
13          }
14      }
15      return false;
16  }
17
18  int solve() {
19      std::fill(match, match + m, -1);
20      int answer = 0;
21      for (int i = 0; i < n; ++i) {
22          stamp++;
23          answer += dfs(i);
24      }
25      return answer;
26  }
```

2.3   Hopcoft-Karp 求最大匹配

```
1   int matchx[N], matchy[N], level[N];
2
3   bool dfs(int x) {
4       for (int i = 0; i < (int)edge[x].size(); ++i) {
5           int y = edge[x][i];
6           int w = matchy[y];
7           if (w == -1 || level[x] + 1 == level[w] && dfs(w)) {
8               matchx[x] = y;
9               matchy[y] = x;
10              return true;
11          }
12      }
13      level[x] = -1;
14      return false;
15  }
16
17  int solve() {
18      std::fill(matchx, matchx + n, -1);
19      std::fill(matchy, matchy + m, -1);
20      for (int answer = 0; ; ) {
21          std::vector<int> queue;
22          for (int i = 0; i < n; ++i) {
23              if (matchx[i] == -1) {
24                  level[i] = 0;
25                  queue.push_back(i);
26              } else {
27                  level[i] = -1;
28              }
29          }
30          for (int head = 0; head < (int)queue.size(); ++head) {
31              int x = queue[head];
32              for (int i = 0; i < (int)edge[x].size(); ++i) {
33                  int y = edge[x][i];
34                  int w = matchy[y];
35                  if (w != -1 && level[w] < 0) {
36                      level[w] = level[x] + 1;
37                      queue.push_back(w);
38                  }
39              }
40          }
41          int delta = 0;
42          for (int i = 0; i < n; ++i) {
```

```
43              if (matchx[i] == -1 && dfs(i)) {
44                  delta++;
45              }
46          }
47          if (delta == 0) {
48              return answer;
49          } else {
50              answer += delta;
51          }
52      }
53  }
```

2.4   KM 带权匹配

注意事项： 最小权完美匹配，复杂度为 $\mathcal{O}(|V|^3)$。

```
1   int DFS(int x){
2       visx[x] = 1;
3       for (int y = 1;y <= ny;y ++){
4           if (visy[y]) continue;
5           int t = lx[x] + ly[y] - w[x][y];
6           if (t == 0) {
7               visy[y] = 1;
8               if (link[y] == -1||DFS(link[y])){
9                   link[y] = x;
10                  return 1;
11              }
12          }
13          else slack[y] = min(slack[y],t);
14      }
15      return 0;
16  }
17  int KM(){
18      int i,j;
19      memset(link,-1,sizeof(link));
20      memset(ly,0,sizeof(ly));
21      for (i = 1; i <= nx; i++)
22          for (j = 1, lx[i] = -inf; j <= ny; j++)
23            lx[i] = max(lx[i],w[i][j]);
24      for (int x = 1; x <= nx; x++){
25          for (i = 1; i <= ny; i++) slack[i] = inf;
26          while (true) {
27              memset(visx, 0, sizeof(visx));
28              memset(visy, 0, sizeof(visy));
29              if (DFS(x)) break;
30              int d = inf;
31              for (i = 1; i <= ny;i++)
32                  if (!visy[i] && d > slack[i]) d = slack[i];
33              for (i = 1; i <= nx; i++)
34                  if (visx[i]) lx[i] -= d;
35              for (i = 1; i <= ny; i++)
36                  if (visy[i]) ly[i] += d;
37                  else slack[i] -= d;
38          }
39      }
40      int res = 0;
41      for (i = 1;i <= ny;i ++)
42          if (link[i] > -1) res += w[link[i]][i];
43      return res;
44  }
```

2.5   稀疏图最大流

注意事项： 适用于比较稀疏的一般图。

```
1   int Maxflow_Isap(int s,int t,int n) {
2     std::fill(pre + 1, pre + n + 1, 0);
3     std::fill(d + 1, d + n + 1, 0);
4     std::fill(gap + 1, gap + n + 1, 0);
```

```
5      for (int i = 1; i <= n; i++) cur[i] = h[i];
6      gap[0] = n;
7      int u = pre[s] = s, v, maxflow = 0;
8      while (d[s] < n) {
9        v = n + 1;
10       for (int i = cur[u]; i; i = e[i].next)
11       if (e[i].flow && d[u] == d[e[i].node] + 1) {
12         v = e[i].node; cur[u]=i; break;
13       }
14       if (v <= n) {
15         pre[v] = u; u = v;
16         if (v == t) {
17           int dflow = INF, p = t; u = s;
18           while (p != s) {
19             p = pre[p];
20             dflow = std::min(dflow, e[cur[p]].flow);
21           }
22           maxflow += dflow; p = t;
23           while (p != s) {
24             p = pre[p];
25             e[cur[p]].flow -= dflow;
26             e[e[cur[p]].opp].flow += dflow;
27           }
28         }
29       }
30       else{
31         int mindist = n + 1;
32         for (int i = h[u]; i; i = e[i].next)
33           if (e[i].flow && mindist > d[e[i].node]) {
34             mindist = d[e[i].node]; cur[u] = i;
35           }
36         if (!--gap[d[u]]) return maxflow;
37         gap[d[u] = mindist + 1]++; u = pre[u];
38       }
39     }
40     return maxflow;
41   }
```

2.6 稠密图最大流

注意事项： 适用于二分图以及一些比较稠密的、增广路径比较短的图。

```
1  int bfs(){
2    for (int i = 1;i <= t;i ++) d[i] = -1;
3    int l,r;
4    q[l = r = 0] = s, d[s] = 0;
5    for (;l <= r;l ++)
6      for (int k = h[q[l]]; k > -1; k = nxt[k])
7        if (d[p[k]] == -1 && c[k] > 0) d[p[k]] = d[q[l]] + 1, q[++ r] = p[k];
8    return d[t] > -1 ? 1 : 0;
9  }
10 int dfs(int u,int ext){
11   if (u == t) return ext;
12   int k = w[u],ret = 0;
13   for (; k > -1; k = nxt[k], w[u] = k){        //w 数组为当前弧
14     if (ext == 0) break;
15     if (d[p[k]] == d[u] + 1 && c[k] > 0){
16       int flow = dfs(p[k], min(c[k], ext));
17       if (flow > 0){
18         c[k] -= flow, c[k ^ 1] += flow;
19         ret += flow, ext -= flow;      //ret 累计增广量，ext 记录还可增广的量
20       }
21     }
22   }
23   if (k == -1) d[u] = -1;
24   return ret;
25 }
26 void dinic() {
27   while (bfs()) {
28     for (int i = 1; i <= t;i ++) w[i] = h[i];
```

```
29          dfs(s, inf);
30      }
31  }
```

2.7   稀疏图费用流

```
1   struct EdgeList {
2       int size;
3       int last[N];
4       int succ[M], other[M], flow[M], cost[M];
5       void clear(int n) {
6           size = 0;
7           std::fill(last, last + n, -1);
8       }
9       void add(int x, int y, int c, int w) {
10          succ[size] = last[x];
11          last[x] = size;
12          other[size] = y;
13          flow[size] = c;
14          cost[size++] = w;
15      }
16  } e;
17
18  int n, source, target;
19  int prev[N];
20
21  void add(int x, int y, int c, int w) {
22      e.add(x, y, c, w);
23      e.add(y, x, 0, -w);
24  }
25
26  bool augment() {
27      static int dist[N], occur[N];
28      std::vector<int> queue;
29      std::fill(dist, dist + n, INT_MAX);
30      std::fill(occur, occur + n, 0);
31      dist[source] = 0;
32      occur[source] = true;
33      queue.push_back(source);
34      for (int head = 0; head < (int)queue.size(); ++head) {
35          int x = queue[head];
36          for (int i = e.last[x]; ~i; i = e.succ[i]) {
37              int y = e.other[i];
38              if (e.flow[i] && dist[y] > dist[x] + e.cost[i]) {
39                  dist[y] = dist[x] + e.cost[i];
40                  prev[y] = i;
41                  if (!occur[y]) {
42                      occur[y] = true;
43                      queue.push_back(y);
44                  }
45              }
46          }
47          occur[x] = false;
48      }
49      return dist[target] < INT_MAX;
50  }
51
52  std::pair<int, int> solve() {
53      std::pair<int, int> answer = std::make_pair(0, 0);
54      while (augment()) {
55          int number = INT_MAX;
56          for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
57              number = std::min(number, e.flow[prev[i]]);
58          }
59          answer.first += number;
60          for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
61              e.flow[prev[i]] -= number;
62              e.flow[prev[i] ^ 1] += number;
63              answer.second += number * e.cost[prev[i]];
```

```
64          }
65      }
66      return answer;
67  }
```

2.8 稠密图费用流

```
1   int aug(int no,int res) {
2       if(no == t) return cost += pi1 * res,res;
3       v[no] = true;
4       int flow = 0;
5       for(int i = h[no]; ~ i ;i = nxt[i])
6       if(cap[i] && !expense[i] && !v[p[i]]) {
7           int d = aug(p[i],min(res,cap[i]));
8           cap[i] -= d,cap[i ^ 1] += d,flow += d,res -= d;
9           if( !res ) return flow;
10      }
11      return flow;
12  }
13  bool modlabel() {
14      int d = maxint;
15      for(int i = 1;i <= t;++ i)
16      if(v[i]) {
17          for(int j = h[i]; ~ j ;j = nxt[j])
18              if(cap[j] && !v[p[j]] && expense[j] < d) d = expense[j];
19      }
20      if(d == maxint)return false;
21      for(int i = 1;i <= t;++ i)
22      if(v[i]) {
23          for(int j = h[i];~ j;j = nxt[j])
24              expense[j] -= d,expense[j ^ 1] += d;
25      }
26      pi1 += d;
27      return true;
28  }
29  void minimum_cost_flow_zkw() {
30    cost = 0;
31    do{
32      do{
33        memset(v,false,sizeof v);
34      }while (aug(s,maxint));
35    }while (modlabel());
36  }
```

2.9 2-SAT 问题

```
1   int stamp, comps, top;
2   int dfn[N], low[N], comp[N], stack[N];
3
4   void add(int x, int a, int y, int b) {
5       edge[x << 1 | a].push_back(y << 1 | b);
6   }
7
8   void tarjan(int x) {
9       dfn[x] = low[x] = ++stamp;
10      stack[top++] = x;
11      for (int i = 0; i < (int)edge[x].size(); ++i) {
12          int y = edge[x][i];
13          if (!dfn[y]) {
14              tarjan(y);
15              low[x] = std::min(low[x], low[y]);
16          } else if (!comp[y]) {
17              low[x] = std::min(low[x], dfn[y]);
18          }
19      }
20      if (low[x] == dfn[x]) {
21          comps++;
22          do {
```

```
23              int y = stack[--top];
24              comp[y] = comps;
25          } while (stack[top] != x);
26      }
27  }
28
29  bool solve() {
30      int counter = n + n + 1;
31      stamp = top = comps = 0;
32      std::fill(dfn, dfn + counter, 0);
33      std::fill(comp, comp + counter, 0);
34      for (int i = 0; i < counter; ++i) {
35          if (!dfn[i]) {
36              tarjan(i);
37          }
38      }
39      for (int i = 0; i < n; ++i) {
40          if (comp[i << 1] == comp[i << 1 | 1]) {
41              return false;
42          }
43          answer[i] = (comp[i << 1 | 1] < comp[i << 1]);
44      }
45      return true;
46  }
```

2.10   有根树的同构

```
1   const unsigned long long MAGIC = 4423;
2
3   unsigned long long magic[N];
4   std::pair<unsigned long long, int> hash[N];
5
6   void solve(int root) {
7       magic[0] = 1;
8       for (int i = 1; i <= n; ++i) {
9           magic[i] = magic[i - 1] * MAGIC;
10      }
11      std::vector<int> queue;
12      queue.push_back(root);
13      for (int head = 0; head < (int)queue.size(); ++head) {
14          int x = queue[head];
15          for (int i = 0; i < (int)son[x].size(); ++i) {
16              int y = son[x][i];
17              queue.push_back(y);
18          }
19      }
20      for (int index = n - 1; index >= 0; --index) {
21          int x = queue[index];
22          hash[x] = std::make_pair(0, 0);
23
24          std::vector<std::pair<unsigned long long, int> > value;
25          for (int i = 0; i < (int)son[x].size(); ++i) {
26              int y = son[x][i];
27              value.push_back(hash[y]);
28          }
29          std::sort(value.begin(), value.end());
30
31          hash[x].first = hash[x].first * magic[1] + 37;
32          hash[x].second++;
33          for (int i = 0; i < (int)value.size(); ++i) {
34              hash[x].first = hash[x].first * magic[value[i].second] + value[i].first;
35              hash[x].second += value[i].second;
36          }
37          hash[x].first = hash[x].first * magic[1] + 41;
38          hash[x].second++;
39      }
40  }
```

2.11   Dominator Tree

```
1   #define foreach(A, x, it) for (int it = A.h[x]; it; it = A.e[it].next)
2
3   const int MAXN = 200001;
4   const int MAXM = 400001;
5
6   template<class T, int MAXN, int MAXM>
7   struct AdjList{
8     struct Edge{
9       T data;
10      int next;
11    }e[MAXM];
12    int h[MAXN], t;
13    void add(int x, const T &data) {
14      t++;
15      e[t].data = data;
16      e[t].next = h[x];
17      h[x] = t;
18    }
19    void drop(int x) {
20      h[x] = e[h[x]].next;
21    }
22    T & operator [](const int &index) {
23      return e[index].data;
24    }
25    void clear(int n) {
26      std::fill(h + 1, h + n + 1, t = 0);
27    }
28  };
29  // fa 是并查集的父亲, f 是树的父亲, sdom 是半必经点, idom 是必经点, smin 是带权并查集的权值
30  // pred 是前驱链表, succ 是后继链表
31  int dfn[MAXN], sdom[MAXN], idom[MAXN], id[MAXN], f[MAXN], fa[MAXN], smin[MAXN];
32  AdjList<int, MAXN, MAXM> pred, succ;
33  long long answer[MAXN];
34
35  void predfs(int x) {
36    id[dfn[x] = ++stamp] = x;
37    foreach(succ, x, i) {
38      int y = succ[i];
39      if (!dfn[y]) {
40        f[y] = x;
41        predfs(y);
42      }
43    }
44  }
45
46  int getfa(int x) {
47    if (fa[x] == x) return x;
48    int ret = getfa(fa[x]);
49    if (dfn[sdom[smin[fa[x]]]] < dfn[sdom[smin[x]]]) {
50      smin[x] = smin[fa[x]];
51    }
52    return fa[x] = ret;
53  }
54
55  void tarjan(int s) {
56    static AdjList<int, MAXN, MAXN> tmp;
57    stamp = tmp.t = 0;
58    predfs(s);
59    for (int i = 1; i <= stamp; i++) {
60      fa[id[i]] = smin[id[i]] = id[i];
61      tmp.h[id[i]] = idom[id[i]] = 0;
62    }
63    for (int o = stamp; o >= 1; o--) {
64      int x = id[o];
65      if (o != 1) {
66        sdom[x] = f[x];
67        foreach(pred, x, i) {
68          int p = pred[i];
```

```
69         if (!dfn[p]) continue;
70         if (dfn[p] > dfn[x]) {
71           getfa(p);
72           p = sdom[smin[p]];
73         }
74         if (dfn[sdom[x]] > dfn[p]) {
75           sdom[x] = p;
76         }
77       }
78       tmp.add(sdom[x], x);
79     }
80     while (tmp.h[x] != 0) {
81       int y = tmp[tmp.h[x]];
82       tmp.drop(x);
83       getfa(y);
84       if (x != sdom[smin[y]]) {
85         idom[y] = smin[y];
86       } else {
87         idom[y] = x;
88       }
89     }
90     foreach(succ, x, i) {
91       if (f[succ[i]] == x) {
92         fa[succ[i]] = x;
93       }
94     }
95   }
96   idom[s] = s;
97   for (int i = 2; i <= stamp; i++) {
98     int x = id[i];
99     if (idom[x] != sdom[x]) {
100      idom[x] = idom[idom[x]];
101    }
102  }
103 }
```

2.12   哈密尔顿回路（ORE 性质的图）

```
1  int left[N], right[N], next[N], last[N];
2
3  void cover(int x) {
4      left[right[x]] = left[x];
5      right[left[x]] = right[x];
6  }
7
8  int adjacent(int x) {
9      for (int i = right[0]; i <= n; i = right[i]) {
10         if (graph[x][i]) {
11             return i;
12         }
13     }
14     return 0;
15 }
16
17 std::vector<int> solve() {
18     for (int i = 1; i <= n; ++i) {
19         left[i] = i - 1;
20         right[i] = i + 1;
21     }
22     int head, tail;
23     for (int i = 2; i <= n; ++i) {
24         if (graph[1][i]) {
25             head = 1;
26             tail = i;
27             cover(head);
28             cover(tail);
29             next[head] = tail;
30             break;
31         }
```

```
32          }
33          while (true) {
34              int x;
35              while (x = adjacent(head)) {
36                  next[x] = head;
37                  head = x;
38                  cover(head);
39              }
40              while (x = adjacent(tail)) {
41                  next[tail] = x;
42                  tail = x;
43                  cover(tail);
44              }
45              if (!graph[head][tail]) {
46                  for (int i = head, j; i != tail; i = next[i]) {
47                      if (graph[head][next[i]] && graph[tail][i]) {
48                          for (j = head; j != i; j = next[j]) {
49                              last[next[j]] = j;
50                          }
51                          j = next[head];
52                          next[head] = next[i];
53                          next[tail] = i;
54                          tail = j;
55                          for (j = i; j != head; j = last[j]) {
56                              next[j] = last[j];
57                          }
58                          break;
59                      }
60                  }
61              }
62              next[tail] = head;
63              if (right[0] > n) {
64                  break;
65              }
66              for (int i = head; i != tail; i = next[i]) {
67                  if (adjacent(i)) {
68                      head = next[i];
69                      tail = i;
70                      next[tail] = 0;
71                      break;
72                  }
73              }
74          }
75          std::vector<int> answer;
76          for (int i = head; ; i = next[i]) {
77              if (i == 1) {
78                  answer.push_back(i);
79                  for (int j = next[i]; j != i; j = next[j]) {
80                      answer.push_back(j);
81                  }
82                  answer.push_back(i);
83                  break;
84              }
85              if (i == tail) {
86                  break;
87              }
88          }
89          return answer;
90      }
```

2.13 无向图最小割

```
1   int node[N], dist[N];
2   bool visit[N];
3
4   int solve(int n) {
5       int answer = INT_MAX;
6       for (int i = 0; i < n; ++i) {
7           node[i] = i;
```

```
 8          }
 9          while (n > 1) {
10              int max = 1;
11              for (int i = 0; i < n; ++i) {
12                  dist[node[i]] = graph[node[0]][node[i]];
13                  if (dist[node[i]] > dist[node[max]]) {
14                      max = i;
15                  }
16              }
17              int prev = 0;
18              memset(visit, 0, sizeof(visit));
19              visit[node[0]] = true;
20              for (int i = 1; i < n; ++i) {
21                  if (i == n - 1) {
22                      answer = std::min(answer, dist[node[max]]);
23                      for (int k = 0; k < n; ++k) {
24                          graph[node[k]][node[prev]] =
25                              (graph[node[prev]][node[k]] += graph[node[k]][node[max]]);
26                      }
27                      node[max] = node[--n];
28                  }
29                  visit[node[max]] = true;
30                  prev = max;
31                  max = -1;
32                  for (int j = 1; j < n; ++j) {
33                      if (!visit[node[j]]) {
34                          dist[node[j]] += graph[node[prev]][node[j]];
35                          if (max == -1 || dist[node[max]] < dist[node[j]]) {
36                              max = j;
37                          }
38                      }
39                  }
40              }
41          }
42      return answer;
43  }
```

2.14   弦图判定

```
 1  int n, m, first[1001], l, next[2000001], where[2000001],f[1001], a[1001], c[1001], L[1001],
    ↪  R[1001],
 2  v[1001], idx[1001], pos[1001];
 3  bool b[1001][1001];
 4
 5  inline void makelist(int x, int y){
 6      where[++l] = y;
 7      next[l] = first[x];
 8      first[x] = l;
 9  }
10
11  bool cmp(const int &x, const int &y){
12      return(idx[x] < idx[y]);
13  }
14
15  int main(){
16      for (;;)
17      {
18          n = read(); m = read();
19          if (!n && !m) return 0;
20          memset(first, 0, sizeof(first)); l = 0;
21          memset(b, false, sizeof(b));
22          for (int i = 1; i <= m; i++)
23          {
24              int x = read(), y = read();
25              if (x != y && !b[x][y])
26              {
27                  b[x][y] = true; b[y][x] = true;
28                  makelist(x, y); makelist(y, x);
29              }
```

```
30              }
31          memset(f, 0, sizeof(f));
32          memset(L, 0, sizeof(L));
33          memset(R, 255, sizeof(R));
34          L[0] = 1; R[0] = n;
35          for (int i = 1; i <= n; i++) c[i] = i, pos[i] = i;
36          memset(idx, 0, sizeof(idx));
37          memset(v, 0, sizeof(v));
38          for (int i = n; i; --i)
39          {
40              int now = c[i];
41              R[f[now]]--;
42              if (R[f[now]] < L[f[now]]) R[f[now]] = -1;
43              idx[now] = i; v[i] = now;
44              for (int x = first[now]; x; x = next[x])
45                  if (!idx[where[x]])
46                  {
47                      swap(c[pos[where[x]]], c[R[f[where[x]]]]);
48                      pos[c[pos[where[x]]]] = pos[where[x]];
49                      pos[where[x]] = R[f[where[x]]];
50                      L[f[where[x]] + 1] = R[f[where[x]]]--;
51                      if (R[f[where[x]]] < L[f[where[x]]]) R[f[where[x]]] = -1;
52                      if (R[f[where[x]] + 1] == -1)
53                          R[f[where[x]] + 1] = L[f[where[x]] + 1];
54                      ++f[where[x]];
55                  }
56          }
57          bool ok = true;
58          //v 是完美消除序列.
59          for (int i = 1; i <= n && ok; i++)
60          {
61              int cnt = 0;
62              for (int x = first[v[i]]; x; x = next[x])
63                  if (idx[where[x]] > i) c[++cnt] = where[x];
64              sort(c + 1, c + cnt + 1, cmp);
65              bool can = true;
66              for (int j = 2; j <= cnt; j++)
67                  if (!b[c[1]][c[j]])
68                  {
69                      ok = false;
70                      break;
71                  }
72          }
73          if (ok) printf("Perfect\n");
74          else printf("Imperfect\n");
75          printf("\n");
76      }
77  }
```

2.15 弦图求最大团

```
1  int n, m, first[100001], next[2000001], where[2000001], l, L[100001], R[100001], c[100001],
   ↪  f[100001],
2  pos[100001], idx[100001], v[100001], ans;
3
4  inline void makelist(int x, int y){
5      where[++l] = y;
6      next[l] = first[x];
7      first[x] = l;
8  }
9
10 int read(){
11     char ch;
12     for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
13     int cnt = 0;
14     for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
15     return(cnt);
16 }
17
```

```
18   int main(){
19       //freopen("1006.in", "r", stdin);
20       //freopen("1006.out", "w", stdout);
21       memset(first, 0, sizeof(first)); l = 0;
22       n = read(); m = read();
23       for (int i = 1; i <= m; i++)
24       {
25           int x, y;
26           x = read(); y = read();
27           makelist(x, y); makelist(y, x);
28       }
29       memset(L, 0, sizeof(L));
30       memset(R, 255, sizeof(R));
31       memset(f, 0, sizeof(f));
32       memset(idx, 0, sizeof(idx));
33       for (int i = 1; i <= n; i++) c[i] = i, pos[i] = i;
34       L[0] = 1; R[0] = n; ans = 0;
35       for (int i = n; i; --i)
36       {
37           int now = c[i], cnt = 1;
38           idx[now] = i; v[i] = now;
39           if (--R[f[now]] < L[f[now]]) R[f[now]] = -1;
40           for (int x = first[now]; x; x = next[x])
41               if (!idx[where[x]])
42               {
43                   swap(c[pos[where[x]]], c[R[f[where[x]]]]);
44                   pos[c[pos[where[x]]]] = pos[where[x]];
45                   pos[where[x]] = R[f[where[x]]];
46                   L[f[where[x]] + 1] = R[f[where[x]]]--;
47                   if (R[f[where[x]]] < L[f[where[x]]]) R[f[where[x]]] = -1;
48                   if (R[f[where[x]] + 1] == -1) R[f[where[x]] + 1] = L[f[where[x]] + 1];
49                   ++f[where[x]];
50               }
51               else ++cnt;
52           ans = max(ans, cnt);
53       }
54       printf("%d\n", ans);
55   }
```

2.16   最大团搜索

```
1    // mc[i] 代表只用 i-n 号点的答案
2    // g 代表连通性
3    void dfs(int size) {
4      int i, j, k;
5      if (len[size] == 0) {
6        if (size > ans) {
7          ans = size;
8          found = true;
9        }
10       return;
11     }
12     for (k = 0; k < len[size] && !found; k ++) {
13       if (size + len[size] - k <= ans) break;
14       i = list[size][k];
15       if (size + mc[i] <= ans) break;
16       for (j = k + 1, len[size + 1] = 0; j < len[size]; j ++)
17         if (g[i][list[size][j]]) list[size + 1][len[size + 1] ++] = list[size][j];
18       dfs(size + 1);
19       if (found) {
20         prin[size + 1] = i;
21       }
22     }
23   }
24   void work() {
25     int i, j;
26     mc[n] = ans = 1;
27     ansi = 1;
28     for (i = n - 1; i; i --) {
```

```
29        found = false;
30        len[1] = 0;
31        for (j = i + 1; j <= n; j ++) if (g[i][j]) list[1][len[1]++] = j;
32        dfs(1);
33        mc[i] = ans;
34        if (found) prin[1] = i;
35    }
36 }
37 void print() {
38    printf("%d\n", ans);
39    for (int i = 1; i < ans; i ++) printf("%d ", prin[i]);
40    printf("%d\n", prin[ans]);
41 }
```

2.17 极大团计数

```
1  bool g[N][N];
2  int ne[N], ce[N], list[N][N], ans;
3  void dfs(int size) {
4    if (ans > 1000) return;
5    int i, j, k, t, cnt, best = 0;
6    bool bb;
7    if (ne[size] == ce[size]) {
8      if (ce[size] == 0) ++ans;
9      return;
10   }
11   for (t = 0, i = 1; i <= ne[size]; ++i) {
12     for (cnt = 0, j = ne[size] + 1; j <= ce[size]; ++j)
13       if (!g[list[size][i]][list[size][j]]) ++cnt;
14     if (t == 0 || cnt < best) t = i, best = cnt;
15   }
16   if (t && best <= 0) return;
17   for (k = ne[size] + 1; k <= ce[size]; ++k) {
18     if (t > 0) {
19       for (i = k; i <= ce[size]; ++i)
20         if (!g[list[size][t]][list[size][i]]) break;
21       swap(list[size][k], list[size][i]);
22     }
23     i = list[size][k];
24     ne[size + 1] = ce[size + 1] = 0;
25     for (j = 1; j < k; ++j)
26       if (g[i][list[size][j]])
27         list[size + 1][++ne[size + 1]] = list[size][j];
28     for (ce[size + 1] = ne[size + 1], j = k + 1; j <= ce[size]; ++j)
29       if (g[i][list[size][j]]) list[size + 1][++ce[size + 1]] = list[size][j];
30     dfs(size + 1);
31     ++ne[size];
32     --best;
33     for (j = k + 1, cnt = 0; j <= ce[size]; ++j)
34       if (!g[i][list[size][j]]) ++cnt;
35     if (t == 0 || cnt < best) t = k, best = cnt;
36     if (t && best <= 0) break;
37   }
38 }
39 int main(){
40   int n, m;
41   while (scanf("%d%d", &n, &m) == 2) {
42     for (int i = 1; i <= n; ++i)
43       for (int j = 1; j <= n; ++j)
44         g[i][j] = false;
45     while (m--) {
46       int x, y;
47       scanf("%d%d", &x, &y);
48       g[x][y] = g[y][x] = true;
49     }
50     ne[0] = 0;
51     ce[0] = 0;
52     for (int i = 1; i <= n; ++i)
53       list[0][++ce[0]] = i;
```

```
54        ans = 0;
55        dfs(0);
56        if (ans > 1000) puts("Too many maximal sets of friends.");
57        else printf("%d\n", ans);
58      }
59      return 0;
60    }
```

2.18   最小树形图

```
1   int n, m, used[N], pass[N], eg[N], more, queue[N];
2   double g[N][N];
3
4   void combine(int id, double &sum) {
5     int tot = 0, from, i, j, k;
6     for (; id != 0 && !pass[id]; id = eg[id]) {
7       queue[tot++] = id;
8       pass[id] = 1;
9     }
10
11    for (from = 0; from < tot && queue[from] != id; from++);
12    if  (from == tot) return;
13    more = 1;
14    for (i = from; i < tot; i++) {
15      sum += g[eg[queue[i]]][queue[i]];
16      if (i != from) {
17        used[queue[i]] = 1;
18        for (j = 1; j <= n; j++) if (!used[j]) {
19          if (g[queue[i]][j] < g[id][j]) g[id][j] = g[queue[i]][j];
20        }
21      }
22    }
23
24    for (i = 1; i <= n; i++) if (!used[i] && i != id) {
25      for (j = from; j < tot; j++) {
26        k = queue[j];
27        if (g[i][id] > g[i][k] - g[eg[k]][k]) g[i][id] = g[i][k] - g[eg[k]][k];
28      }
29    }
30  }
31
32  double mdst(int root) {
33    int i, j, k;
34    double sum = 0;
35    memset(used, 0, sizeof(used));
36    for (more = 1; more; ) {
37      more = 0;
38      memset(eg, 0, sizeof(eg));
39      for (i = 1; i <= n; i++) if (!used[i] && i != root) {
40        for (j = 1, k = 0; j <= n; j++) if (!used[j] && i != j)
41          if (k == 0 || g[j][i] < g[k][i]) k = j;
42        eg[i] = k;
43      }
44
45      memset(pass, 0, sizeof(pass));
46      for (i = 1; i <= n; i++) if (!used[i] && !pass[i] && i != root) combine(i, sum);
47    }
48
49    for (i = 1; i <= n; i++) if (!used[i] && i != root) sum += g[eg[i]][i];
50    return sum;
51  }
```

2.19   带花树

```
1   int match[N], belong[N], next[N], mark[N], visit[N];
2   std::vector<int> queue;
3
4   int find(int x) {
```

```
 5          if (belong[x] != x) {
 6              belong[x] = find(belong[x]);
 7          }
 8          return belong[x];
 9      }
10
11      void merge(int x, int y) {
12          x = find(x);
13          y = find(y);
14          if (x != y) {
15              belong[x] = y;
16          }
17      }
18
19      int lca(int x, int y) {
20          static int stamp = 0;
21          stamp++;
22          while (true) {
23              if (x != -1) {
24                  x = find(x);
25                  if (visit[x] == stamp) {
26                      return x;
27                  }
28                  visit[x] = stamp;
29                  if (match[x] != -1) {
30                      x = next[match[x]];
31                  } else {
32                      x = -1;
33                  }
34              }
35              std::swap(x, y);
36          }
37      }
38
39      void group(int a, int p) {
40          while (a != p) {
41              int b = match[a], c = next[b];
42              if (find(c) != p) {
43                  next[c] = b;
44              }
45              if (mark[b] == 2) {
46                  mark[b] = 1;
47                  queue.push_back(b);
48              }
49              if (mark[c] == 2) {
50                  mark[c] = 1;
51                  queue.push_back(c);
52              }
53              merge(a, b);
54              merge(b, c);
55              a = c;
56          }
57      }
58
59      void augment(int source) {
60          queue.clear();
61          for (int i = 0; i < n; ++i) {
62              next[i] = visit[i] = -1;
63              belong[i] = i;
64              mark[i] = 0;
65          }
66          mark[source] = 1;
67          queue.push_back(source);
68          for (int head = 0; head < (int)queue.size() && match[source] == -1; ++head) {
69              int x = queue[head];
70              for (int i = 0; i < (int)edge[x].size(); ++i) {
71                  int y = edge[x][i];
72                  if (match[x] == y || find(x) == find(y) || mark[y] == 2) {
73                      continue;
74                  }
```

```
75                if (mark[y] == 1) {
76                    int r = lca(x, y);
77                    if (find(x) != r) {
78                        next[x] = y;
79                    }
80                    if (find(y) != r) {
81                        next[y] = x;
82                    }
83                    group(x, r);
84                    group(y, r);
85                } else if (match[y] == -1) {
86                    next[y] = x;
87                    for (int u = y; u != -1; ) {
88                        int v = next[u];
89                        int mv = match[v];
90                        match[v] = u;
91                        match[u] = v;
92                        u = mv;
93                    }
94                    break;
95                } else {
96                    next[y] = x;
97                    mark[y] = 2;
98                    mark[match[y]] = 1;
99                    queue.push_back(match[y]);
100               }
101           }
102       }
103   }
104
105   int solve() {
106       std::fill(match, match + n, -1);
107       for (int i = 0; i < n; ++i) {
108           if (match[i] == -1) {
109               augment(i);
110           }
111       }
112       int answer = 0;
113       for (int i = 0; i < n; ++i) {
114           answer += (match[i] != -1);
115       }
116       return answer;
117   }
```

2.20　度限制生成树

```
1    int n, m, S, K, ans , cnt , Best[N], fa[N], FE[N];
2    int f[N], p[M], t[M], c[M], o, Cost[N];
3    bool u[M], d[M];
4    pair<int, int> MinCost[N];
5    struct Edge {
6      int a, b, c;
7      bool operator < (const Edge & E) const { return c < E.c; }
8    }E[M];
9    vector<int> SE;
10   inline int F(int x) {
11     return fa[x] == x ? x : fa[x] = F(fa[x]);
12   }
13   inline void AddEdge(int a, int b, int C) {
14     p[++o] = b; c[o] = C;
15     t[o] = f[a]; f[a] = o;
16   }
17   void dfs(int i, int father) {
18     fa[i] = father;
19     if (father == S) Best[i] = -1;
20     else {
21       Best[i] = i;
22       if (~Best[father] && Cost[Best[father]] > Cost[i]) Best[i] = Best[father];
23     }
```

```
24      for (int j = f[i]; j; j = t[j])
25      if (!d[j] && p[j] != father) {
26        Cost[p[j]] = c[j];
27        FE[p[j]] = j;
28        dfs(p[j], i);
29      }
30    }
31    inline bool Kruskal() {
32      cnt = n - 1,ans = 0; o = 1;
33      for (int i = 1; i <= n; i++) fa[i] = i, f[i] = 0;
34      sort(E + 1, E + m + 1);
35      for (int i = 1; i <= m; i++) {
36        if (E[i].b == S) swap(E[i].a, E[i].b);
37        if (E[i].a != S && F(E[i].a) != F(E[i].b)) {
38          fa[F(E[i].a)] = F(E[i].b);
39          ans += E[i].c;
40          cnt --;
41          u[i] = true;
42          AddEdge(E[i].a, E[i].b, E[i].c);
43          AddEdge(E[i].b, E[i].a, E[i].c);
44        }
45      }
46      for (int i = 1; i <= n; i++) MinCost[i] = make_pair(INF, INF);
47      for (int i = 1; i <= m; i++)
48      if (E[i].a == S) {
49        SE.push_back(i);
50        MinCost[F(E[i].b)] = min(MinCost[F(E[i].b)], make_pair(E[i].c, i));
51      }
52      int dif = 0;
53      for (int i = 1; i <= n; i++)
54      if (i != S && fa[i] == i) {
55        if (MinCost[i].second == INF) return false;
56        if (++ dif > K) return false;
57        dfs(E[MinCost[i].second].b, S);
58        u[MinCost[i].second] = true;
59        ans += MinCost[i].first;
60      }
61      return true;
62    }
63    bool Solve() {
64      memset(d,false,sizeof d);
65      memset(u,false,sizeof u);
66      if (!Kruskal()) return false;
67      for (int i = cnt + 1; i <= K && i <= n; i++) {
68        int MinD = INF, MinID = -1;
69        for (int j = (int) SE.size() - 1; j >= 0; j--)
70        if (u[SE[j]])
71          SE.erase(SE.begin() + j);
72        for (int j = 0; j < (int) SE.size(); j++) {
73          int tmp = E[SE[j]].c - Cost[Best[E[SE[j]].b]];
74          if (tmp < MinD) {
75            MinD = tmp;
76            MinID= SE[j];
77          }
78        }
79        if (MinID == -1) return true;
80        if (MinD >= 0) break;
81        ans += MinD;
82        u[MinID] = true;
83        d[FE[Best[E[MinID].b]]] = d[FE[Best[E[MinID].b]] ^ 1] = true;
84        dfs(E[MinID].b, S);
85      }
86      return true;
87    }
88    int main(){
89      Solve();
90      return 0;
91    }
```

Chapter 3

数论及代数

3.1 快速求逆

```cpp
long long inverse(const long long &x, const long long &mod) {
    if (x == 1) {
        return 1;
    } else {
        return (mod - mod / x) * inverse(mod % x, mod) % mod;
    }
}
```

3.2 魔幻多项式

快速傅里叶变换

注意事项：请实现复数类 Complex，并注意快速傅里叶变换精度较差，建议使用快速数论变换。

```cpp
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;
  }
}
```

光速数论变换

注意事项：MOD 应该为一个特殊的质数 $2^n + 1$ 且 n 应该要足够大，PRT 为这个质数的原根。

```cpp
// 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)))
```

```
5   void DFT(int *a, int n, int f) { // 封闭形式，常数小 (10^7 跑 2.23 秒)
6     for (register int i = 0, j = 0; i < n; i++) {
7       if (i > j) std::swap(a[i], a[j]);
8       for (register int t = n >> 1; (j ^= t) < t; t >>= 1);
9     }
10    for (register int i = 2; i <= n; i <<= 1) {
11      static int exp[MAXN];
12      exp[0] = 1; exp[1] = fpm(PRT, (MOD - 1) / i);
13      if (f == 1) exp[1] = fpm(exp[1], MOD - 2);
14      for (register int k = 2; k < (i >> 1); k++) {
15        exp[k] = 1ll * exp[k - 1] * exp[1] % MOD;
16      }
17      for (register int j = 0; j < n; j += i) {
18        for (register int k = 0; k < (i >> 1); k++) {
19          register int &pA = a[j + k], &pB = a[j + k + (i >> 1)];
20          register int A = pA, B = 1ll * pB * exp[k] % MOD;
21          pA = (A + B) % MOD;
22          pB = (A - B + MOD) % MOD;
23        }
24      }
25    }
26    if (f == 1) {
27      register int rev = fpm(n, MOD - 2, MOD);
28      for (register int i = 0; i < n; i++) {
29        a[i] = 1ll * a[i] * rev % MOD;
30      }
31    }
32  }
33  // 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
34  // 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
35  int CRT(int *a) {
36    static int x[3];
37    for (int i = 0; i < 3; i++) {
38      x[i] = a[i];
39      for (int j = 0; j < i; j++) {
40        int t = (x[i] - x[j] + FFT[i] -> MOD) % FFT[i] -> MOD;
41        if (t < 0) t += FFT[i] -> MOD;
42        x[i] = 1LL * t * inv[j][i] % FFT[i] -> MOD;
43      }
44    }
45    int sum = 1, ret = x[0] % MOD;
46    for (int i = 1; i < 3; i ++) {
47      sum = 1LL * sum * FFT[i - 1] -> MOD % MOD;
48      ret += 1LL * x[i] * sum % MOD;
49      if(ret >= MOD) ret -= MOD;
50    }
51    return ret;
52  }
53  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
54    for (int j = 0; j < 3; j++)
55      inv[i][j] = inverse(FFT[i] -> MOD, FFT[j] -> MOD);
```

牛顿迭代法

问题描述： 给出多项式 $G(x)$，求解多项式 $F(x)$ 满足：

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

答案只需要精确到 $F(x) \bmod x^n$ 即可。

实现原理： 考虑倍增，假设有：

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

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

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

那么就有：

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

注意事项： $G(F(x))$ 的常数项系数必然为 0，这个可以作为求解的初始条件；

多项式求逆

原理： 令 $G(x) = x * A - 1$（其中 A 是一个多项式系数），根据牛顿迭代法有：

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

注意事项：

1. $F(x)$ 的常数项系数必然不为 0，否则没有逆元；

2. 复杂度是 $O(n \log n)$ 但是常数比较大（10^5 大概需要 0.3 秒左右）；

3. 传入的两个数组必须不同，但传入的次数界没有必要是 2 的次幂；

```
1   void getInv(int *a, int *b, int n) {
2     static int tmp[MAXN];
3     b[0] = fpm(a[0], MOD - 2, MOD);
4     for (int c = 2, M = 1; c < (n << 1); c <<= 1) {
5       for (; M <= 3 * (c - 1); M <<= 1);
6       meminit(b, c, M);
7       meminit(tmp, c, M);
8       memcopy(tmp, a, 0, c);
9       DFT(tmp, M, 0);
10      DFT(b, M, 0);
11      for (int i = 0; i < M; i++) {
12        b[i] = 1ll * b[i] * (2ll - 1ll * tmp[i] * b[i] % MOD + MOD) % MOD;
13      }
14      DFT(b, M, 1);
15      meminit(b, c, M);
16    }
17  }
```

多项式取指数和对数

作用： 给出一个多项式 $A(x)$，求一个多项式 $F(x)$ 满足 $e^A(x) - F(x) \equiv 0 \pmod{x^n}$。

原理： 令 $G(x) = \ln x - A$（其中 A 是一个多项式系数），根据牛顿迭代法有：

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

求 $\ln F_t(x)$ 可以用先求导再积分的办法，即：

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

多项式的求导和积分可以在 $O(n)$ 的时间内完成，因此总复杂度为 $O(n \log n)$。

应用： 加速多项式快速幂。

注意事项：

1. 进行 log 的多项式必须保证常数项系数为 1，否则必须要先求出 $\log a[0]$ 是多少；

2. 传入的两个数组必须不同，但传入的次数界没有必要是 2 的次幂；

3. 常数比较大，10^5 的数据求指数和对数分别需要 0.37s 和 0.85s 左右的时间，注意这里 `memset` 几乎不占用时。

```
1   void getDiff(int *a, int *b, int n) { // 多项式取微分
2     for (int i = 0; i + 1 < n; i++) {
3       b[i] = 1ll * (i + 1) * a[i + 1] % MOD;
4     }
5     b[n - 1] = 0;
6   }
7   void getInt(int *a, int *b, int n) { // 多项式取积分，积分常数为 0
8     static int inv[MAXN];
9     inv[1] = 1;
10    for (int i = 2; i < n; i++) {
```

```
11      inv[i] = 1ll * (MOD - MOD / i) * inv[MOD % i] % MOD;
12    }
13    b[0] = 0;
14    for (int i = 1; i < n; i++) {
15      b[i] = 1ll * a[i - 1] * inv[i] % MOD;
16    }
17  }
18  void getLn(int *a, int *b, int n) {
19    static int inv[MAXN], d[MAXN];
20    int M = 1;
21    for (; M <= 2 * (n - 1); M <<= 1);
22    getInv(a, inv, n);
23    getDiff(a, d, n);
24    meminit(d, n, M);
25    meminit(inv, n, M);
26    DFT(d, M, 0); DFT(inv, M, 0);
27    for (int i = 0; i < M; i++) {
28      d[i] = 1ll * d[i] * inv[i] % MOD;
29    }
30    DFT(d, M, 1);
31    getInt(d, b, n);
32  }
33  void getExp(int *a, int *b, int n) {
34    static int ln[MAXN], tmp[MAXN];
35    b[0] = 1;
36    for (int c = 2, M = 1; c < (n << 1); c <<= 1) {
37      for (; M <= 2 * (c - 1); M <<= 1);
38      int bound = std::min(c, n);
39      memcopy(tmp, a, 0, bound);
40      meminit(tmp, bound, M);
41      meminit(b, c, M);
42      getLn(b, ln, c);
43      meminit(ln, c, M);
44      DFT(b, M, 0);
45      DFT(tmp, M, 0);
46      DFT(ln, M, 0);
47      for (int i = 0; i < M; i++) {
48        b[i] = 1ll * b[i] * (1ll - ln[i] + tmp[i] + MOD) % MOD;
49      }
50      DFT(b, M, 1);
51      meminit(b, c, M);
52    }
53  }
```

多项式除法

作用： 给出两个多项式 $A(x)$ 和 $B(x)$，求两个多项式 $D(x)$ 和 $R(x)$ 满足：

$$A(x) \equiv D(x)B(x) + R(x) \pmod{x^n}$$

注意事项：

1. 常数比较大概为 6 倍 FFT 的时间，即大约 10^5 的数据 0.07s 左右；

2. 传入两个多项式的次数界，没有必要是 2 的次幂，但是要保证除数多项式不为 0。

```
1   void divide(int n, int m, int *a, int *b, int *d, int *r) {
→       // n、m 分别为多项式 A（被除数）和 B（除数）的次数界
2     static int M, tA[MAXN], tB[MAXN], inv[MAXN], tD[MAXN];
3     for (; n > 0 && a[n - 1] == 0; n--);
4     for (; m > 0 && b[m - 1] == 0; m--);
5     for (int i = 0; i < n; i++) tA[i] = a[n - i - 1];
6     for (int i = 0; i < m; i++) tB[i] = b[m - i - 1];
7     for (M = 1; M <= n - m + 1; M <<= 1);
8     meminit(tB, m, M);
9     getInv(tB, inv, M);
10    for (M = 1; M <= 2 * (n - m + 1); M <<= 1);
11    meminit(inv, n - m + 1, M);
12    meminit(tA, n - m + 1, M);
13    DFT(inv, M, 0);
14    DFT(tA, M, 0);
```

```
15    for (int i = 0; i < M; i++) {
16      d[i] = 1ll * inv[i] * tA[i] % MOD;
17    }
18    DFT(d, M, 1);
19    std::reverse(d, d + n - m + 1);
20    for (M = 1; M <= n; M <<= 1);
21    memcopy(tB, b, 0, m); meminit(tB, m, M);
22    memcopy(tD, d, 0, n - m + 1); meminit(tD, n - m + 1, M);
23    DFT(tD, M, 0);
24    DFT(tB, M, 0);
25    for (int i = 0; i < M; i++) {
26      r[i] = 1ll * tD[i] * tB[i] % MOD;
27    }
28    DFT(r, M, 1);
29    meminit(r, n, M);
30    for (int i = 0; i < n; i++) {
31      r[i] = (a[i] - r[i] + MOD) % MOD;
32    }
33  }
```

3.3   线性递推数列求第 n 项

```
1   //已知 a_0, a_1, ..., a_{m-1}
2   //   a_n = c_0 * a_{n-m} + ... + c_{m-1} * a_{n-1}
3   //   求 a_n = v_0 * a_0 + v_1 * a_1 + ... + v_{m-1} * a_{m-1}
4   void linear_recurrence(long long n, int m, int a[], int c[], int p) {
5     long long v[M] = {1 % p}, u[M << 1], msk = !!n;
6     for(long long i(n); i > 1; i >>= 1) {
7       msk <<= 1;
8     }
9     for(long long x(0); msk; msk >>= 1, x <<= 1) {
10      fill_n(u, m << 1, 0);
11      int b(!!(n & msk));
12      x |= b;
13      if(x < m) {
14        u[x] = 1 % p;
15      }else {
16        for(int i(0); i < m; i++) {
17          for(int j(0), t(i + b); j < m; j++, t++) {
18            u[t] = (u[t] + v[i] * v[j]) % p;
19          }
20        }
21        for(int i((m << 1) - 1); i >= m; i--) {
22          for(int j(0), t(i - m); j < m; j++, t++) {
23            u[t] = (u[t] + c[j] * u[i]) % p;
24          }
25        }
26      }
27      copy(u, u + m, v);
28    }
29    //a[n] = v[0] * a[0] + v[1] * a[1] + ... + v[m - 1] * a[m - 1]
30    for(int i(m); i < 2 * m; i++) {
31      a[i] = 0;
32      for(int j(0); j < m; j++) {
33        a[i] = (a[i] + (long long)c[j] * a[i + j - m]) % p;
34      }
35    }
36    for(int j(0); j < m; j++) {
37      b[j] = 0;
38      for(int i(0); i < m; i++) {
39        b[j] = (b[j] + v[i] * a[i + j]) % p;
40      }
41    }
42    for(int j(0); j < m; j++) {
43      a[j] = b[j];
44    }
45  }
```

3.4 线性规划

注意事项： 使用单纯形法求解：

$$max\{c_{1 \times m} \cdot x_{m \times 1} \mid x_{m \times 1} \geq 0_{m \times 1}, a_{n \times m} \cdot x_{m \times 1} \leq b_{n \times 1}\}$$

```cpp
std::vector<double> solve(const std::vector<std::vector<double> > &a,
                          const std::vector<double> &b, const std::vector<double> &c) {
    int n = (int)a.size(), m = (int)a[0].size() + 1;
    std::vector<std::vector<double> > value(n + 2, std::vector<double>(m + 1));
    std::vector<int> index(n + m);
    int r = n, s = m - 1;
    for (int i = 0; i < n + m; ++i) {
        index[i] = i;
    }
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m - 1; ++j) {
            value[i][j] = -a[i][j];
        }
        value[i][m - 1] = 1;
        value[i][m] = b[i];
        if (value[r][m] > value[i][m]) {
            r = i;
        }
    }
    for (int j = 0; j < m - 1; ++j) {
        value[n][j] = c[j];
    }
    value[n + 1][m - 1] = -1;
    for (double number; ; ) {
        if (r < n) {
            std::swap(index[s], index[r + m]);
            value[r][s] = 1 / value[r][s];
            for (int j = 0; j <= m; ++j) {
                if (j != s) {
                    value[r][j] *= -value[r][s];
                }
            }
            for (int i = 0; i <= n + 1; ++i) {
                if (i != r) {
                    for (int j = 0; j <= m; ++j) {
                        if (j != s) {
                            value[i][j] += value[r][j] * value[i][s];
                        }
                    }
                    value[i][s] *= value[r][s];
                }
            }
        }
        r = s = -1;
        for (int j = 0; j < m; ++j) {
            if (s < 0 || index[s] > index[j]) {
                if (value[n + 1][j] > eps || value[n + 1][j] > -eps && value[n][j] > eps) {
                    s = j;
                }
            }
        }
        if (s < 0) {
            break;
        }
        for (int i = 0; i < n; ++i) {
            if (value[i][s] < -eps) {
                if (r < 0
                || (number = value[r][m] / value[r][s] - value[i][m] / value[i][s]) < -eps
                || number < eps && index[r + m] > index[i + m]) {
                    r = i;
                }
            }
        }
        if (r < 0) {
```

```
65                // Solution is unbounded.
66                return std::vector<double>();
67            }
68        }
69        if (value[n + 1][m] < -eps) {
70            // No solution.
71            return std::vector<double>();
72        }
73        std::vector<double> answer(m - 1);
74        for (int i = m; i < n + m; ++i) {
75            if (index[i] < m - 1) {
76                answer[index[i]] = value[i - m][m];
77            }
78        }
79        return answer;
80    }
```

3.5   中国剩余定理

注意事项： p_i 无需两两互质

```
1   bool solve(int n, std::pair<long long, long long> input[],
2                std::pair<long long, long long> &output) {
3     output = std::make_pair(1, 1);
4     for (int i = 0; i < n; ++i) {
5       long long number, useless;
6       // euclid(a, b, x, y)
7       euclid(output.second, input[i].second, number, useless);
8       long long divisor = std::__gcd(output.second, input[i].second);
9       if ((input[i].first - output.first) % divisor) {
10        return false;
11      }
12      number *= (input[i].first - output.first) / divisor;
13      fix(number, input[i].second);
14      output.first += output.second * number;
15      output.second *= input[i].second / divisor;
16      fix(output.first, output.second);
17    }
18    return true;
19  }
```

3.6   直线下整点个数

注意事项： 返回结果为：

$$\sum_{0 \le i < n} \lfloor \frac{a + b \cdot i}{m} \rfloor$$

即直线下整点个数。

```
1   long long solve(const long long &n, const long long &a,
2                const long long &b, const long long &m) {
3     if (b == 0) {
4       return n * (a / m);
5     }
6     if (a >= m) {
7       return n * (a / m) + solve(n, a % m, b, m);
8     }
9     if (b >= m) {
10      return (n - 1) * n / 2 * (b / m) + solve(n, a, b % m, m);
11    }
12    return solve((a + b * n) / m, (a + b * n) % m, m, b);
13  }
```

3.7   闪电素数判定

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

```
3   bool check(const long long &prime, const long long &base) {
4     long long number = prime - 1;
5     for (; ~number & 1; number >>= 1);
6     long long result = power_mod(base, number, prime);
7     for (; number != prime - 1 && result != 1 && result != prime - 1; number <<= 1) {
8       result = multiply_mod(result, result, prime);
9     }
10    return result == prime - 1 || (number & 1) == 1;
11  }
12
13  bool miller_rabin(const long long &number) {
14    if (number < 2) {
15      return false;
16    }
17    if (number < 4) {
18      return true;
19    }
20    if (~number & 1) {
21      return false;
22    }
23    for (int i = 0; i < 12 && BASE[i] < number; ++i) {
24      if (!check(number, BASE[i])) {
25        return false;
26      }
27    }
28    return true;
29  }
```

3.8 闪电质因数分解

```
1   long long pollard_rho(const long long &number, const long long &seed) {
2     long long x = rand() % (number - 1) + 1, y = x;
3     for (int head = 1, tail = 2; ; ) {
4       x = multiply_mod(x, x, number);
5       x = add_mod(x, seed, number);
6       if (x == y) {
7         return number;
8       }
9       long long answer = std::__gcd(abs(x - y), number);
10      if (answer > 1 && answer < number) {
11        return answer;
12      }
13      if (++head == tail) {
14        y = x;
15        tail <<= 1;
16      }
17    }
18  }
19
20  void factorize(const long long &number, std::vector<long long> &divisor) {
21    if (number > 1) {
22      if (miller_rabin(number)) {
23        divisor.push_back(number);
24      } else {
25        long long factor = number;
26        for (; factor >= number;)
27          factor = pollard_rho(number, rand() % (number - 1) + 1));
28        factorize(number / factor, divisor);
29        factorize(factor, divisor);
30      }
31    }
32  }
```

3.9 自适应辛普森

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

```
4    }
5
6    double simpson(const double &left, const double &right,
7                   const double &eps, const double &area_sum) {
8      double mid = (left + right) / 2;
9      double area_left = area(left, mid);
10     double area_right = area(mid, right);
11     double area_total = area_left + area_right;
12     if (std::abs(area_total - area_sum) < 15 * eps) {
13       return area_total + (area_total - area_sum) / 15;
14     }
15     return simpson(left, mid, eps / 2, area_left)
16         + simpson(mid, right, eps / 2, area_right);
17   }
18
19   double simpson(const double &left, const double &right, const double &eps) {
20     return simpson(left, right, eps, area(left, right));
21   }
```

3.10   二次剩余

```
1    void calcH(int &t, int &h, const int p) {
2      int tmp = p - 1; for (t = 0; (tmp & 1) == 0; tmp /= 2) t++; h = tmp;
3    }
4    // solve equation x^2 mod p = a
5    bool solve(int a, int p, int &x, int &y) {
6      srand(19920225);
7      if (p == 2) { x = y = 1; return true; }
8      int p2 = p / 2, tmp = power(a, p2, p);
9      if (tmp == p - 1) return false;
10     if ((p + 1) % 4 == 0) {
11       x = power(a, (p + 1) / 4, p); y = p - x; return true;
12     } else {
13       int t, h, b, pb; calcH(t, h, p);
14       if (t >= 2) {
15         do {b = rand() % (p - 2) + 2;
16         } while (power(b, p / 2, p) != p - 1);
17         pb = power(b, h, p);
18       } int s = power(a, h / 2, p);
19       for (int step = 2; step <= t; step++) {
20         int ss = (((long long)(s * s) % p) * a) % p;
21         for (int i = 0; i < t - step; i++) ss = ((long long)ss * ss) % p;
22         if (ss + 1 == p) s = (s * pb) % p; pb = ((long long)pb * pb) % p;
23       } x = ((long long)s * a) % p; y = p - x;
24     } return true;
25   }
26   // 求方程 a * x^2 + b * x + c == 0 (mod P) 的根:
27   int pDiv2,P,a,b,c,Pb,d;
28   inline int calc(int x,int Time){
29     if (!Time) return 1;  int tmp=calc(x,Time/2);
30     tmp=(long long)tmp*tmp%P;
31     if (Time&1) tmp=(long long)tmp*x%P;  return tmp;
32   }
33   inline int rev(int x){ if (!x) return 0;    return calc(x,P-2);}
34   inline void Compute(){
35     while (1) { b=rand()%(P-2)+2;  if (calc(b,pDiv2)+1==P) return; }
36   }
37   int main(){
38     srand(time(0)^312314);  int T;
39     for (scanf("%d",&T);T;--T) {
40       scanf("%d%d%d%d",&a,&b,&c,&P);
41       if (P == 2) {
42         int cnt = 0;
43         for (int i = 0; i < 2; ++i)
44           if ((a * i * i + b * i + c) % P == 0) ++cnt;
45         printf("%d", cnt);
46         for (int i = 0; i < 2; ++i) {
47           if ((a * i * i + b * i + c)% P == 0)
48             printf(" %d", i);
```

```
49        }
50        puts("");
51    } else {
52        int delta = (long long)b * rev(a) * rev(2) % P;
53        a = (long long)c * rev(a) % P - sqr( (long long)delta ) % P;
54        a %= P; a += P; a %= P;  a = P - a; a %= P;  pDiv2 = P / 2;
55        if (calc(a, pDiv2) + 1 == P) puts("0");
56        else {
57          int t = 0, h = pDiv2;
58          while (!(h % 2)) ++t, h /= 2;
59          int root = calc(a, h/2);
60          if (t > 0) {
61            Compute();
62            Pb = calc(b, h);
63          }
64          for (int i = 1; i <= t; ++i) {
65            d = (long long)root * root * a % P;
66            for (int j = 1; j <= t - i; ++j) {
67              d = (long long)d * d % P;
68            }
69            if (d + 1 == P) {
70              root = (long long)root * Pb % P;
71            }
72            Pb = (long long)Pb * Pb % P;
73          }
74          root = (long long)a * root % P;
75          int root1 = P - root;
76          root -= delta;
77          root %= P;
78          if (root < 0) root += P;
79          root1 -= delta; root1 %= P;
80          if (root1 < 0) root1 += P;
81          if (root > root1) {
82            t = root;
83            root = root1;
84            root1 = t;
85          }
86          if (root == root1) printf("1 %d\n",root);
87          else printf("2 %d %d\n",root,root1);
88        }
89      }
90    }
91    return 0;
92  }
```

3.11   Pell 方程

```
1  ULL A,B,p[maxn],q[maxn],a[maxn],g[maxn],h[maxn];
2  int main() {
3    for (int test=1, n;scanf("%d",&n) && n;++test) {
4      printf("Case %d: ",test);
5      if (fabs(sqrt(n)-floor(sqrt(n)+1e-7))<=1e-7)   {
6        int a=(int)(floor(sqrt(n)+1e-7)); printf("%d %d\n",a,1);
7      } else {
8        // 求 x^2 − ny^2 = 1 的最小正整数根，n 不是完全平方数
9        p[1]=q[0]=h[1]=1;p[0]=q[1]=g[1]=0;
10       a[2]=(int)(floor(sqrt(n)+1e-7));
11       for (int i=2;i;++i) {
12         g[i]=-g[i-1]+a[i]*h[i-1]; h[i]=(n-sqr(g[i]))/h[i-1];
13         a[i+1]=(g[i]+a[2])/h[i]; p[i]=a[i]*p[i-1]+p[i-2];
14         q[i]=a[i]*q[i-1]+q[i-2];
15         if (sqr((ULL)(p[i]))-n*sqr((ULL)(q[i]))==1){
16           A=p[i];B=q[i];break;
17         }
18       }
19       cout << A << ' ' << B <<endl;
20     }
21   }
22 }
```

3.12 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 4

字符串

4.1 广义后缀自动机

注意事项：空间是插入字符串总长度的 2 倍并请注意字符集大小。

```cpp
void add(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 = ++size; l[auxnode] = l[lastnode] + 1;
      for (int i = 0; i < 26; i++) c[auxnode][i] = c[nownode][i];
      f[auxnode] = f[nownode]; f[nownode] = auxnode;
      for (; lastnode && c[lastnode][x] == nownode; lastnode = f[lastnode]) {
        c[lastnode][x] = auxnode;
      }
      last = auxnode;
    }
  }
  else{ // Naive Suffix Automaton
    int newnode = ++size; l[newnode] = l[lastnode] + 1;
    for (; lastnode && !c[lastnode][x]; lastnode = f[lastnode]) c[lastnode][x] = newnode;
    if (!lastnode) f[newnode] = 1;
    else{
      int nownode = c[lastnode][x];
      if (l[lastnode] + 1 == l[nownode]) f[newnode] = nownode;
      else{
        int auxnode = ++size; l[auxnode] = l[lastnode] + 1;
        for (int i = 0; i < 26; i++) c[auxnode][i] = c[nownode][i];
        f[auxnode] = f[nownode]; f[nownode] = f[newnode] = auxnode;
        for (; lastnode && c[lastnode][x] == nownode; lastnode = f[lastnode]) {
          c[lastnode][x] = auxnode;
        }
      }
    }
    last = newnode;
  }
}
```

4.2 后缀数组

注意事项：$\mathcal{O}(n \log n)$ 倍增构造。

```cpp
namespace suffix_array{
  int wa[MAXN], wb[MAXN], ws[MAXN], wv[MAXN];
  bool cmp(int *r, int a, int b, int l) {
    return r[a] == r[b] && r[a + l] == r[b + l];
  }
  void DA(int *r, int *sa, int n, int m) {
    int *x = wa, *y = wb, *t;
    for (int i = 0; i < m; i++) ws[i] = 0;
    for (int i = 0; i < n; i++) ws[x[i] = r[i]]++;
```

```
10      for (int i = 1; i < m; i++) ws[i] += ws[i - 1];
11      for (int i = n - 1; i >= 0; i--) sa[--ws[x[i]]] = i;
12      for (int i, j = 1, p = 1; p < n; j <<= 1, m = p) {
13        for (p = 0, i = n - j; i < n; i++) y[p++] = i;
14        for (i = 0; i < n; i++) if (sa[i] >= j) y[p++] = sa[i] - j;
15        for (i = 0; i < n; i++) wv[i] = x[y[i]];
16        for (i = 0; i < m; i++) ws[i] = 0;
17        for (i = 0; i < n; i++) ws[wv[i]]++;
18        for (i = 1; i < m; i++) ws[i] += ws[i - 1];
19        for (i = n - 1; i >= 0; i--) sa[--ws[wv[i]]] = y[i];
20        for (t = x, x = y, y = t, p = 1, x[sa[0]] = 0, i = 1; i < n; i++)
21          x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p++;
22      }
23    }
24    void getheight(int *r, int *sa, int *rk, int *h, int n) {
25      for (int i = 1; i <= n; i++) rk[sa[i]] = i;
26      for (int i = 0, j, k = 0; i < n; h[rk[i++]] = k)
27        for (k ? k-- : 0, j = sa[rk[i] - 1]; r[i + k] == r[j + k]; k++);
28    }
29  };
```

4.3   回文自动机

注意事项：请注意字符集大小。

```
1  struct Palindromic_Tree{
2    int nTree, nStr, last, c[MAXT][26], fail[MAXT], r[MAXN], l[MAXN], s[MAXN];
3    int allocate(int len) {
4      l[nTree] = len;
5      r[nTree] = 0;
6      fail[nTree] = 0;
7      memset(c[nTree], 0, sizeof(c[nTree]));
8      return nTree++;
9    }
10   void init() {
11     nTree = nStr = 0;
12     int newEven = allocate(0);
13     int newOdd = allocate(-1);
14     last = newEven;
15     fail[newEven] = newOdd;
16     fail[newOdd] = newEven;
17     s[0] = -1;
18   }
19   void add(int x) {
20     s[++nStr] = x;
21     int nownode = last;
22     while (s[nStr - l[nownode] - 1] != s[nStr]) nownode = fail[nownode];
23     if (!c[nownode][x]) {
24       int newnode = allocate(l[nownode] + 2), &newfail = fail[newnode];
25       newfail = fail[nownode];
26       while (s[nStr - l[newfail] - 1] != s[nStr]) newfail = fail[newfail];
27       newfail = c[newfail][x];
28       c[nownode][x] = newnode;
29     }
30     last = c[nownode][x];
31     r[last]++;
32   }
33   void count() {
34     for (int i = nTree - 1; i >= 0; i--) {
35       r[fail[i]] += r[i];
36     }
37   }
38 }
```

4.4   Manacher

注意事项：1-based 算法，请注意下表。

```
1  int manacher(char *text, int length, int *palindrome) {
2    static char buffer[MAXN];
3    for (int i = 1; i <= length; i++) {
4      buffer[2 * i - 1] = text[i];
5      if (i != 0) buffer[2 * i] = '#';
6    }
7
8    palindrome[1] = 1;
9    for (int i = 2, j = 0; i <= 2 * length - 1; ++i) {
10     if (j + palindrome[j] <= i) {
11       palindrome[i] = 0;
12     } else {
13       palindrome[i] = std::min(palindrome[(j << 1) - i], j + palindrome[j] - i);
14     }
15     while (i - palindrome[i] >= 1 && i + palindrome[i] <= 2 * length - 1
16       && buffer[i - palindrome[i]] == buffer[i + palindrome[i]]) {
17       palindrome[i]++;
18     }
19     if (i + palindrome[i] > j + palindrome[j]) {
20       j = i;
21     }
22   }
23
24   int answer = 0;
25   for (int i = 1; i < 2 * length; i++) {
26     if (i & 1) answer = std::max(answer, 2 * (palindrome[i] - 1 >> 1) + 1);
27     else answer = std::max(answer, 2 * (palindrome[i] >> 1));
28   }
29   return answer;
30 }
```

4.5   循环串的最小表示

注意事项： 0-Based 算法，请注意下标。

```
1  int minrep(const char *s) {
2    int length = strlen(s);
3    int i = 0, j = 1, k = 0;
4    while (i < length && j < length && k < length) {
5      if (s[(i + k) % length] == s[(j + k) % length]) {
6        k++;
7      } else {
8        if (s[(i + k) % length] > s[(j + k) % length]) {
9          i += k + 1;
10       } else {
11         j += k + 1;
12       }
13       if (i == j) j++;
14       k = 0;
15     }
16   }
17   return std::min(i, j);
18 }
```

4.6   后缀树

注意事项：

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；

```
1  const int MAXL = 100001; // The length of the string being inserted into the ST.
2  const int MAXD = 27;     // The size of the alphabet.
3
4  struct SuffixTree{
5    int size, length, pCur, dCur, lCur, lBuf, text[MAXL];
6    std::pair<int, int> suffix[MAXL];
```

```
 7
 8    struct Node{
 9      int left, right, sLink, next[MAXD];
10    }tree[MAXL * 2];
11
12    int getLength(const int &rhs) {
13      return tree[rhs].right ? tree[rhs].right - tree[rhs].left : length + 1 - tree[rhs].left;
14    }
15    void addLink(int &last, int node) {
16      if (last != 0) tree[last].sLink = node;
17      last = node;
18    }
19    int alloc(int left, int right = 0) {
20      size++;
21      memset(&tree[size], 0, sizeof(tree[size]));
22      tree[size].left = left;
23      tree[size].right = right;
24      tree[size].sLink = 1;
25      return size;
26    }
27    bool move(int node) {
28      int length = getLength(node);
29      if (lCur >= length) {
30        lCur -= length;
31        dCur += length;
32        pCur = node;
33        return true;
34      }
35      return false;
36    }
37    void init() {
38      size = length = 0;
39      lCur = dCur = lBuf = 0;
40      pCur = alloc(0);
41    }
42    void extend(int x) {
43      text[++length] = x;
44      lBuf++;
45      for (int last = 0; lBuf > 0; ) {
46        if (lCur == 0) dCur = length;
47        if (!tree[pCur].next[text[dCur]]) {
48          int newleaf = alloc(length);
49          tree[pCur].next[text[dCur]] = newleaf;
50          suffix[length + 1 - lBuf] = std::make_pair(pCur, newleaf);
51          addLink(last, pCur);
52        } else {
53          int nownode = tree[pCur].next[text[dCur]];
54          if (move(nownode)) continue;
55          if (text[tree[nownode].left + lCur] == x) {
56            lCur++;
57            addLink(last, pCur);
58            break;
59          }
60          int newleaf = alloc(length), newnode = alloc(tree[nownode].left, tree[nownode].left +
    ↪  lCur);
61          tree[nownode].left += lCur;
62          tree[pCur].next[text[dCur]] = newnode;
63          tree[newnode].next[x] = newleaf;
64          tree[newnode].next[text[tree[nownode].left]] = nownode;
65          suffix[length + 1 - lBuf] = std::make_pair(newnode, newleaf);
66          addLink(last, newnode);
67        }
68        lBuf--;
69        if (pCur == 1 && lCur > 0) lCur--, dCur++;
70        else pCur = tree[pCur].sLink;
71      }
72    }
73  };
```

Chapter 5

数据结构

5.1　树链剖分

点操作版本

```cpp
void modify(int x, int y, int val) {
  int fx = t[x], fy = t[y];
  while (fx != fy) {
    if (d[fx] > d[fy]) {
      modify(1, 1, n, w[fx], w[x], val);
      x = f[fx]; fx = t[x];
    }
    else{
      modify(1, 1, n, w[fy], w[y], val);
      y = f[fy]; fy = t[y];
    }
  }
  if (d[x] < d[y]) modify(1, 1, n, w[x], w[y], val);
  else modify(1, 1, n, w[y], w[x], val);
}
Node query(int x, int y) {
  int fx = t[x], fy = t[y];
  Node left = Node(), right = Node();
  while (fx != fy) {
    if (d[fx] > d[fy]) {
      left = query(1, 1, n, w[fx], w[x]) + left;
      x = f[fx]; fx = t[x];
    } else {
      right = query(1, 1, n, w[fy], w[y]) + right;
      y = f[fy]; fy = t[y];
    }
  }
  if (d[x] < d[y]) {
    right = query(1, 1, n, w[x], w[y]) + right;
  } else {
    left = query(1, 1, n, w[y], w[x]) + left;
  }
  std::swap(left.lsum, left.rsum);
  return left + right;
}
```

边操作版本

```cpp
void modify(int x, int y) {
  int fx = t[x], fy = t[y];
  while (fx != fy) {
    if (d[fx] > d[fy]) {
      modify(1, 1, n, w[fx], w[x]);
      x = f[fx]; fx = t[x];
    } else {
      modify(1, 1, n, w[fy], w[y]);
      y = f[fy]; fy = t[y];
    }
  }
```

```
12      if (x != y) {
13        if (d[x] < d[y]) modify(1, 1, n, w[z[x]], w[y]);
14        else modify(1, 1, n, w[z[y]], w[x]);
15      }
16    }
17    // TODO 边询问
```

5.2   Link Cut Tree

```
1    struct MsgNode{
2      int leftColor, rightColor, answer;
3      MsgNode() {
4        leftColor = -1;
5        rightColor = -1;
6        answer = 0;
7      }
8      MsgNode(int c) {
9        leftColor = rightColor = c;
10       answer = 1;
11     }
12     MsgNode operator +(const MsgNode &p)const {
13       if (answer == 0) return p;
14       if (p.answer == 0) return *this;
15       MsgNode ret;
16       ret.leftColor = leftColor;
17       ret.rightColor = p.rightColor;
18       ret.answer = answer + p.answer - (rightColor == p.leftColor);
19       return ret;
20     }
21   }d[MAXN], g[MAXN];
22   int n, m, c[MAXN][2], f[MAXN], p[MAXN], s[MAXN], flag[MAXN];
23   bool r[MAXN];
24   void init(int x, int value) {
25     d[x] = g[x] = MsgNode(value);
26     c[x][0] = c[x][1] = 0;
27     f[x] = p[x] = flag[x] = -1;
28     s[x] = 1;
29   }
30   void update(int x) {
31     s[x] = s[c[x][0]] + s[c[x][1]] + 1;
32     g[x] = MsgNode();
33     if (c[x][0 ^ r[x]]) g[x] = g[x] + g[c[x][0 ^ r[x]]];
34     g[x] = g[x] + d[x];
35     if (c[x][1 ^ r[x]]) g[x] = g[x] + g[c[x][1 ^ r[x]]];
36   }
37   void makesame(int x, int c) {
38     flag[x] = c;
39     d[x] = MsgNode(c);
40     g[x] = MsgNode(c);
41   }
42   void pushdown(int x) {
43     if (r[x]) {
44       std::swap(c[x][0], c[x][1]);
45       r[c[x][0]] ^= 1;
46       r[c[x][1]] ^= 1;
47       std::swap(g[c[x][0]].leftColor, g[c[x][0]].rightColor);
48       std::swap(g[c[x][1]].leftColor, g[c[x][1]].rightColor);
49       r[x] = false;
50     }
51     if (flag[x] != -1) {
52       if (c[x][0]) makesame(c[x][0], flag[x]);
53       if (c[x][1]) makesame(c[x][1], flag[x]);
54       flag[x] = -1;
55     }
56   }
57   void rotate(int x, int k) {
58     pushdown(x); pushdown(c[x][k]);
59     int y = c[x][k]; c[x][k] = c[y][k ^ 1]; c[y][k ^ 1] = x;
60     if (f[x] != -1) c[f[x]][c[f[x]][1] == x] = y;
```

```
61     f[y] = f[x]; f[x] = y; f[c[x][k]] = x; std::swap(p[x], p[y]);
62     update(x); update(y);
63   }
64   void splay(int x, int s = -1) {
65     pushdown(x);
66     while (f[x] != s) {
67       if (f[f[x]] != s) rotate(f[f[x]], (c[f[f[x]]][1] == f[x]) ^ r[f[f[x]]]);
68       rotate(f[x], (c[f[x]][1] == x) ^ r[f[x]]);
69     }
70     update(x);
71   }
72   void access(int x) {
73     int y = 0;
74     while (x != -1) {
75       splay(x); pushdown(x);
76       f[c[x][1]] = -1; p[c[x][1]] = x;
77       c[x][1] = y; f[y] = x; p[y] = -1;
78       update(x); x = p[y = x];
79     }
80   }
81   void setroot(int x) {
82     access(x);
83     splay(x);
84     r[x] ^= 1;
85     std::swap(g[x].leftColor, g[x].rightColor);
86   }
87   void link(int x, int y) {
88     setroot(x);
89     p[x] = y;
90   }
91   void cut(int x, int y) {
92       access(x); splay(y, -1);
93       if (p[y] == x) p[y] = -1;
94       else {
95         access(y);
96         splay(x,-1);
97         p[x] = -1;
98       }
99   }
```

5.3　可持久化平衡树

```
1    int ran() {
2      static int ret = 182381727;
3      return (ret += (ret << 1) + 717271723) & (~0u >> 1);
4    }
5    int alloc(int node = 0) {
6      size++;
7      if (node) {
8        c[size][0] = c[node][0];
9        c[size][1] = c[node][1];
10       s[size] = s[node];
11       d[size] = d[node];
12     }
13     else{
14       c[size][0] = 0;
15       c[size][1] = 0;
16       s[size] = 1;
17       d[size] = ' ';
18     }
19     return size;
20   }
21   void update(int x) {
22     s[x] = 1;
23     if (c[x][0]) s[x] += s[c[x][0]];
24     if (c[x][1]) s[x] += s[c[x][1]];
25   }
26   int merge(const std::pair<int, int> &a) {
27     if (!a.first) return a.second;
```

```
28      if (!a.second) return a.first;
29      if (ran() % (s[a.first] + s[a.second]) < s[a.first]) {
30        int newnode = alloc(a.first);
31        c[newnode][1] = merge(std::make_pair(c[newnode][1], a.second));
32        update(newnode);
33        return newnode;
34      }
35      else{
36        int newnode = alloc(a.second);
37        c[newnode][0] = merge(std::make_pair(a.first, c[newnode][0]));
38        update(newnode);
39        return newnode;
40      }
41    }
42    std::pair<int, int> split(int x, int k) {
43      if (!x || !k) return std::make_pair(0, x);
44      int newnode = alloc(x);
45      if (k <= s[c[x][0]]) {
46        std::pair<int, int> ret = split(c[newnode][0], k);
47        c[newnode][0] = ret.second;
48        update(newnode);
49        return std::make_pair(ret.first, newnode);
50      }
51      else{
52        std::pair<int, int> ret = split(c[newnode][1], k - s[c[x][0]] - 1);
53        c[newnode][1] = ret.first;
54        update(newnode);
55        return std::make_pair(newnode, ret.second);
56      }
57    }
58    void travel(int x) {
59      if (c[x][0]) travel(c[x][0]);
60      putchar(d[x]);
61      if (d[x] == 'c') cnt++;
62      if (c[x][1]) travel(c[x][1]);
63    }
64    int build(int l, int r) {
65      int newnode = alloc();
66      d[newnode] = tmp[l + r >> 1];
67      if (l <= (l + r >> 1) - 1) c[newnode][0] = build(l, (l + r >> 1) - 1);
68      if ((l + r >> 1) + 1 <= r) c[newnode][1] = build((l + r >> 1) + 1, r);
69      update(newnode);
70      return newnode;
71    }
72    int main() {
73      scanf("%d", &n);
74      for (int i = 1, last = 0; i <= n; i++) {
75        int op, v, p, l;
76        scanf("%d", &op);
77        if (op == 1) {
78          scanf("%d%s", &p, tmp + 1);
79          p -= cnt;
80          std::pair<int, int> ret = split(rt[last], p);
81          rt[last + 1] = merge(std::make_pair(ret.first, build(1, strlen(tmp + 1))));
82          rt[last + 1] = merge(std::make_pair(rt[last + 1], ret.second));
83          last++;
84        }
85        else if (op == 2) {
86          scanf("%d%d", &p, &l);
87          p -= cnt; l -= cnt;
88          std::pair<int, int> A = split(rt[last], p - 1);
89          std::pair<int, int> B = split(A.second, l);
90          rt[last + 1] = merge(std::make_pair(A.first, B.second));
91          last++;
92        }
93        else if (op == 3) {
94          scanf("%d%d%d", &v, &p, &l);
95          v -= cnt; p -= cnt; l -= cnt;
96          std::pair<int, int> A = split(rt[v], p - 1);
97          std::pair<int, int> B = split(A.second, l);
```

```
 98          travel(B.first);
 99          puts("");
100      }
101   }
102   return 0;
103 }
```

5.4   可持久化左偏树

```
 1 class Node {
 2 public:
 3     Node *left, *right;
 4     int key, dist;
 5     Node(int key) : left(NULL), right(NULL), key(key), dist(0) {}
 6     Node* update() {
 7         if (!left || (right && left->dist < right->dist))
 8             std::swap(left, right);
 9         dist = right ? right->dist + 1 : 0;
10         return this;
11     }
12 };
13 Node* merge(Node *x, Node *y) {
14     if (!x) return y;
15     if (!y) return x;
16     if (x->key < y->key) {
17         x = new Node(*x);
18         x->right = merge(x->right, y);
19         return x->update();
20     } else {
21         y = new Node(*y);
22         y->right = merge(x, y->right);
23         return y->update();
24     }
25 }
```

5.5   Splay Tree

区间操作版本

```
 1 const int INF = 100000000;
 2 const int Maxspace = 500000;
 3 struct SplayNode{
 4   int ls, rs, zs, ms;
 5   SplayNode() {
 6     ms = 0;
 7     ls = rs = zs = -INF;
 8   }
 9   SplayNode(int d) {
10     ms = zs = ls = rs = d;
11   }
12   SplayNode operator +(const SplayNode &p)const {
13     SplayNode ret;
14     ret.ls = max(ls, ms + p.ls);
15     ret.rs = max(rs + p.ms, p.rs);
16     ret.zs = max(rs + p.ls, max(zs, p.zs));
17     ret.ms = ms + p.ms;
18     return ret;
19   }
20 }t[MAXN], d[MAXN];
21 int n, m, rt, top, a[MAXN], f[MAXN], c[MAXN][2], g[MAXN], h[MAXN], z[MAXN];
22 bool r[MAXN], b[MAXN];
23 void makesame(int x, int s) {
24   if (!x) return;
25   b[x] = true;
26   d[x] = SplayNode(g[x] = s);
27   t[x].zs = t[x].ms = g[x] * h[x];
28   t[x].ls = t[x].rs = max(g[x], g[x] * h[x]);
29 }
```

```
30    void makerev(int x) {
31      if (!x) return;
32      r[x] ^= 1;
33      swap(c[x][0], c[x][1]);
34      swap(t[x].ls, t[x].rs);
35    }
36    void pushdown(int x) {
37      if (!x) return;
38      if (r[x]) {
39        makerev(c[x][0]);
40        makerev(c[x][1]);
41        r[x]=0;
42      }
43      if (b[x]) {
44        makesame(c[x][0],g[x]);
45        makesame(c[x][1],g[x]);
46        b[x]=g[x]=0;
47      }
48    }
49    void updata(int x) {
50      if (!x) return;
51      h[x]=h[c[x][0]]+h[c[x][1]]+1;
52      t[x]=t[c[x][0]]+d[x]+t[c[x][1]];
53    }
54    void rotate(int x,int k) {
55      pushdown(x);pushdown(c[x][k]);
56      int y = c[x][k]; c[x][k] = c[y][k^1]; c[y][k^1] = x;
57      if (f[x] != -1) c[f[x]][c[f[x]][1] == x] = y; else rt = y;
58      f[y] = f[x]; f[x] = y; f[c[x][k]] = x;
59      updata(x); updata(y);
60    }
61    void splay(int x, int s) {
62      while (f[x] != s) {
63        if (f[f[x]]!=s) {
64          pushdown(f[f[x]]);
65          rotate(f[f[x]], (c[f[f[x]]][1] == f[x]) ^ r[f[f[x]]]);
66        }
67        pushdown(f[x]);
68        rotate(f[x], (c[f[x]][1]==x) ^ r[f[x]]);
69      }
70    }
71    void build(int &x,int l,int r) {
72      if (l > r) {x = 0; return;}
73      x = z[top--];
74      if (l < r) {
75        build(c[x][0],l,(l+r>>1)-1);
76        build(c[x][1],(l+r>>1)+1,r);
77      }
78      f[c[x][0]] = f[c[x][1]] = x;
79      d[x] = SplayNode(a[l+r>>1]);
80      updata(x);
81    }
82    void init() {
83      d[0] = SplayNode();
84      f[rt=2] = -1;
85      f[1] = 2; c[2][0] = 1;
86      int x;
87      build(x,1,n);
88      c[1][1] = x; f[x] = 1;
89      splay(x, -1);
90    }
91    int find(int z) {
92      int x = rt; pushdown(x);
93      while (z != h[c[x][0]] + 1) {
94        if (z > h[c[x][0]] + 1) {
95          z -= h[c[x][0]] + 1;
96          x = c[x][1];
97        }
98        else x = c[x][0];
99        pushdown(x);
```

```
100        }
101        return x;
102    }
103    void getrange(int &x,int &y) {
104        y = x + y - 1;
105        x = find(x);
106        y = find(y + 2);
107        splay(y, -1);
108        splay(x, y);
109    }
110    void recycle(int x) {
111        if (!x) return;
112        recycle(c[x][0]);
113        recycle(c[x][1]);
114        z[++top]=x;
115        t[x] = d[x] = SplayNode();
116        r[x] = b[x] = g[x] = f[x] = h[x] = 0;
117        c[x][0] = c[x][1]=0;
118    }
119    int main() {
120        scanf("%d%d",&n,&m);
121        for (int i = 1; i <= n; i++) scanf("%d",a+i);
122        for (int i = Maxspace; i>=3; i--) z[++top] = i;
123        init();
124        for (int i = 1; i <= m; i++) {
125            char op[10];
126            int x, y, tmp;
127            scanf("%s", op);
128            if (!strcmp(op, "INSERT")) {
129                scanf("%d%d", &x, &y);
130                n += y;
131                if (!y) continue;
132                for (int i = 1; i <= y; i++) scanf("%d",a+i);
133                build(tmp, 1, y);
134                x = find(x + 1); pushdown(x);
135                if (!c[x][1]) {c[x][1] = tmp; f[tmp] = x;}
136                else{
137                    x = c[x][1]; pushdown(x);
138                    while (c[x][0]) {
139                        x = c[x][0];
140                        pushdown(x);
141                    }
142                    c[x][0] = tmp; f[tmp] = x;
143                }
144                splay(tmp, -1);
145            }
146            else if (!strcmp(op, "DELETE")) {
147                scanf("%d%d", &x, &y); n -= y;
148                if (!y) continue;
149                getrange(x, y);
150                int k = (c[y][0] == x);
151                recycle(c[x][k]);
152                f[c[x][k]] = 0;
153                c[x][k] = 0;
154                splay(x, -1);
155            }
156            else if (!strcmp(op, "REVERSE")) {
157                scanf("%d%d", &x, &y);
158                if (!y) continue;
159                getrange(x, y);
160                int k = (c[y][0]==x);
161                makerev(c[x][k]);
162                splay(c[x][k], -1);
163            }
164            else if (!strcmp(op, "GET-SUM")) {
165                scanf("%d%d", &x, &y);
166                if (!y) {
167                    printf("0\n");
168                    continue;
169                }
```

```
170        getrange(x,y);
171        int k = (c[y][0] == x);
172        printf("%d\n", t[c[x][k]].ms);
173        splay(c[x][k], -1);
174      }
175      else if (!strcmp(op, "MAX-SUM")) {
176        x = 1; y = n;
177        getrange(x, y);
178        int k = (c[y][0] == x);
179        printf("%d\n", t[c[x][k]].zs);
180        splay(c[x][k], -1);
181      }
182      else if (!strcmp(op, "MAKE-SAME")) {
183        scanf("%d%d%d", &x, &y, &tmp);
184        if (!y) continue;
185        getrange(x, y);
186        int k = (c[y][0] == x);
187        makesame(c[x][k], tmp);
188        splay(c[x][k], -1);
189      }
190    }
191    return 0;
192  }
```

序列操作版本

```
1   int pred(int x) {
2     splay(x, -1);
3     for (x = c[x][0]; c[x][1]; x = c[x][1]);
4     return x;
5   }
6   int succ(int x) {
7     splay(x, -1);
8     for (x = c[x][1]; c[x][0]; x = c[x][0]);
9     return x;
10  }
11  void remove(int x) {
12    if (b[x] > 1) {b[x]--; splay(x, -1); return;}
13    splay(x, -1);
14    if (!c[x][0] && !c[x][1]) rt = 0;
15    else if (c[x][0] && !c[x][1]) f[rt = c[x][0]] = -1;
16    else if (!c[x][0] && c[x][1]) f[rt = c[x][1]] = -1;
17    else{
18      int t = pred(x); f[rt = c[x][0]] = -1;
19      c[t][1] = c[x][1]; f[c[x][1]] = t;
20      splay(c[x][1], -1);
21    }
22    c[x][0] = c[x][1] = f[x] = d[x] = s[x] = b[x] = 0;
23  }
24  int find(int z) {
25    int x=rt;
26    while (d[x]!=z)
27      if (c[x][d[x]<z]) x=c[x][d[x]<z];
28      else break;
29    return x;
30  }
31  void insert(int z) {
32    if (!rt) {
33      f[rt = ++size] = -1;
34      d[size] = z; b[size] = 1;
35      splay(size, -1);
36      return;
37    }
38    int x = find(z);
39    if (d[x] == z) {
40      b[x]++;
41      splay(x, -1);
42      return;
43    }
44    c[x][d[x]<z] = ++size; f[size] = x;
```

```
45      d[size] = z; b[size] = s[size] = 1;
46      splay(size, -1);
47  }
48  int select(int z) {
49      int x = rt;
50      while (z < s[c[x][0]] + 1 || z > s[c[x][0]] + b[x])
51          if (z > s[c[x][0]] + b[x]) {
52              z -= s[c[x][0]] + b[x];
53              x = c[x][1];
54          }
55          else x = c[x][0];
56      return x;
57  }
58  int main() {
59      scanf("%d",&n);
60      for (int i = 1; i <= n; i++) {
61          int opt, x;
62          scanf("%d%d", &opt, &x);
63          if (opt == 1) insert(x);
64          else if (opt == 2) remove(find(x)); //删除 x 数 (若有多个相同的数，因只删除一个)
65          else if (opt == 3) { // 查询 x 数的排名 (若有多个相同的数，因输出最小的排名)
66              insert(x);
67              printf("%d\n", s[c[find(x)][0]] + 1);
68              remove(find(x));
69          }
70          else if (opt == 4) printf("%d\n",d[select(x)]);
71          else if (opt == 5) {
72              insert(x);
73              printf("%d\n", d[pred(find(x))]);
74              remove(find(x));
75          }
76          else if (opt == 6) {
77              insert(x);
78              printf("%d\n", d[succ(find(x))]);
79              remove(find(x));
80          }
81      }
82      return 0;
83  }
```

5.6 k-d Tree

```
1   struct Point{
2       int data[MAXK], id;
3   }p[MAXN];
4
5   struct KdNode{
6       int l, r;
7       Point p, dmin, dmax;
8       KdNode() {}
9       KdNode(const Point &rhs) : l(0), r(0), p(rhs), dmin(rhs), dmax(rhs) {}
10      inline void merge(const KdNode &rhs) {
11          for (register int i = 0; i < k; i++) {
12              dmin.data[i] = std::min(dmin.data[i], rhs.dmin.data[i]);
13              dmax.data[i] = std::max(dmax.data[i], rhs.dmax.data[i]);
14          }
15      }
16      inline long long getMinDist(const Point &rhs)const {
17          register long long ret = 0;
18          for (register int i = 0; i < k; i++) {
19              if (dmin.data[i] <= rhs.data[i] && rhs.data[i] <= dmax.data[i]) continue;
20              ret += std::min(1ll * (dmin.data[i] - rhs.data[i]) * (dmin.data[i] - rhs.data[i]),
21                  1ll * (dmax.data[i] - rhs.data[i]) * (dmax.data[i] - rhs.data[i]));
22          }
23          return ret;
24      }
25      long long getMaxDist(const Point &rhs) {
26          long long ret = 0;
27          for (register int i = 0; i < k; i++) {
```

```
28          int tmp = std::max(std::abs(dmin.data[i] - rhs.data[i]),
29              std::abs(dmax.data[i] - rhs.data[i]));
30        ret += 1ll * tmp * tmp;
31      }
32      return ret;
33    }
34  }tree[MAXN * 4];
35
36  struct Result{
37    long long dist;
38    Point d;
39    Result() {}
40    Result(const long long &dist, const Point &d) : dist(dist), d(d) {}
41    bool operator >(const Result &rhs)const {
42      return dist > rhs.dist || (dist == rhs.dist && d.id < rhs.d.id);
43    }
44    bool operator <(const Result &rhs)const {
45      return dist < rhs.dist || (dist == rhs.dist && d.id > rhs.d.id);
46    }
47  };
48
49  inline long long sqrdist(const Point &a, const Point &b) {
50    register long long ret = 0;
51    for (register int i = 0; i < k; i++) {
52      ret += 1ll * (a.data[i] - b.data[i]) * (a.data[i] - b.data[i]);
53    }
54    return ret;
55  }
56
57  inline int alloc() {
58    size++;
59    tree[size].l = tree[size].r = 0;
60    return size;
61  }
62
63  void build(const int &depth, int &rt, const int &l, const int &r) {
64    if (l > r) return;
65    register int middle = l + r >> 1;
66    std::nth_element(p + l, p + middle, p + r + 1,
67      [=](const Point &a, const Point &b){return a.data[depth] < b.data[depth];});
68    tree[rt = alloc()] = KdNode(p[middle]);
69    if (l == r) return;
70    build((depth + 1) % k, tree[rt].l, l, middle - 1);
71    build((depth + 1) % k, tree[rt].r, middle + 1, r);
72    if (tree[rt].l) tree[rt].merge(tree[tree[rt].l]);
73    if (tree[rt].r) tree[rt].merge(tree[tree[rt].r]);
74  }
75
76  std::priority_queue<Result, std::vector<Result>, std::greater<Result> > heap;
77
78  void getMinKth(const int &depth, const int &rt, const int &m, const Point &d) { // 求 K 近点
79    Result tmp = Result(sqrdist(tree[rt].p, d), tree[rt].p);
80    if ((int)heap.size() < m) {
81      heap.push(tmp);
82    } else if (tmp < heap.top()) {
83      heap.pop();
84      heap.push(tmp);
85    }
86    int x = tree[rt].l, y = tree[rt].r;
87    if (x != 0 && y != 0 && sqrdist(d, tree[x].p) > sqrdist(d, tree[y].p)) std::swap(x, y);
88    if (x != 0 && ((int)heap.size() < m || tree[x].getMinDist(d) < heap.top().dist)) {
89      getMinKth((depth + 1) % k, x, m, d);
90    }
91    if (y != 0 && ((int)heap.size() < m || tree[y].getMinDist(d) < heap.top().dist)) {
92      getMinKth((depth + 1) % k, y, m, d);
93    }
94  }
95
96  void getMaxKth(const int &depth, const int &rt, const int &m, const Point &d) { // 求 K 远点
97    Result tmp = Result(sqrdist(tree[rt].p, d), tree[rt].p);
```

```
 98    if ((int)heap.size() < m) {
 99      heap.push(tmp);
100    } else if (tmp > heap.top()) {
101      heap.pop();
102      heap.push(tmp);
103    }
104    int x = tree[rt].l, y = tree[rt].r;
105    if (x != 0 && y != 0 && sqrdist(d, tree[x].p) < sqrdist(d, tree[y].p)) std::swap(x, y);
106    if (x != 0 && ((int)heap.size() < m || tree[x].getMaxDist(d) >= heap.top().dist)) {
  ↪    // 这里的 >= 是因为在距离相等的时候需要按照 id 排序
107      getMaxKth((depth + 1) % k, x, m, d);
108    }
109    if (y != 0 && ((int)heap.size() < m || tree[y].getMaxDist(d) >= heap.top().dist)) {
110      getMaxKth((depth + 1) % k, y, m, d);
111    }
112  }
```

Chapter 6

杂项算法

6.1　Dancing Links

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

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

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

```
52      if(dance(k + 1)) return 1;
53      --cnt;
54      for(j = L[i]; j != i; j = L[j]) resume(C[j]);
55    }
56    resume(c);
57    return 0;
58  }
59
60  // 可重复覆盖
61  const int mxm = 15 * 15 + 10;
62  const int MAXD = 15 * 15 + 10;
63  const int MAXDode = MAXD * mxm;
64  const int INF = 0x3f3f3f3f;
65  //能不加的行尽量不加，减少搜索时间
66  int size;
67  int U[MAXDode], D[MAXDode], R[MAXDode], L[MAXDode], Row[MAXDode],  Col[MAXDode];
68  int H[MAXD], S[mxm];
69  int ansd;
70  void init(int n, int m) {
71    int i;
72    for(i = 0; i <= m; ++i) {
73      S[i] = 0, U[i] = D[i] = i;
74      L[i] = i - 1, R[i] = i + 1;
75    }
76    R[m] = 0, L[0] = m, size = m;
77    for(i = 1; i <= n; ++i) H[i] = -1;
78  }
79  void link(int r, int c) {
80    ++S[Col[++size] = c];
81    Row[size] = r; D[size] = D[c]; U[D[c]] = size; U[size] = c; D[c] = size;
82    if(H[r] < 0) H[r] = L[size] = R[size] = size;
83    else {
84      R[size] = R[H[r]];
85      L[R[H[r]]] = size;
86      L[size] = H[r];
87      R[H[r]] = size;
88    }
89  }
90  void remove(int c) {
91    for(int i = D[c]; i != c; i = D[i])
92      L[R[i]] = L[i], R[L[i]] = R[i];
93  }
94  void resume(int c) {
95    for(int i = U[c]; i != c; i = U[i])
96      L[R[i]] = R[L[i]] = i;
97  }
98  bool vv[mxm];
99  int f() {
100    int ret = 0, c, i, j;
101    for(c = R[0]; c != 0; c = R[c]) vv[c] = 1;
102    for(c = R[0]; c != 0; c = R[c])
103      if(vv[c]) {
104        ++ret, vv[c] = 0;
105        for(i = D[c]; i != c; i = D[i])
106          for(j = R[i]; j != i; j = R[j])
107            vv[Col[j]] = 0;
108      }
109    return ret;
110  }
111  void dance(int d) {
112    if(d + f() >= ansd) return;
113    if(R[0] == 0) {
114      if(d < ansd) ansd = d;
115      return;
116    }
117    int c = R[0], i, j;
118    for(i = R[0]; i; i = R[i])
119      if(S[i] < S[c]) c = i;
120    for(i = D[c]; i != c; i = D[i]) {
121      remove(i);
```

```
122     for(j = R[i]; j != i; j = R[j]) remove(j);
123     dance(d + 1);
124     for(j = L[i]; j != i; j = L[j]) resume(j);
125     resume(i);
126   }
127 }
```

6.2 日期公式

```
1 int zeller(int y, int m, int d) { // y 年 m 月 d 日是星期几
2   if (m <= 2) y--, m += 12; int c = y / 100; y %= 100;
3   int w = ((c >> 2) - (c << 1) + y + (y >> 2) + (13 * (m + 1) / 5) + d - 1) % 7;
4   if (w < 0) w += 7; return w;
5 }
6 int getId(int y, int m, int d) { // y 年 m 月 d 日的日期编号
7   if (m < 3) {y--; m += 12;}
8   return 365 * y + y / 4 - y / 100 + y / 400 + (153 * m + 2) / 5 + d;
9 }
```

6.3 环状最长公共子序列

```
1 int n, a[N << 1], b[N << 1];
2 bool has(int i, int j) { return a[(i - 1) % n] == b[(j - 1) % n];}
3 const int DELTA[3][2] = {{0, -1}, {-1, -1}, {-1, 0}};
4 int from[N][N];
5 int solve() {
6   memset(from, 0, sizeof(from));
7   int ret = 0;
8   for (int i = 1; i <= 2 * n; ++ i) {
9     from[i][0] = 2;
10    int left = 0, up = 0;
11    for (int j = 1; j <= n; ++ j) {
12      int upleft = up + 1 + !!from[i - 1][j];
13      if (!has(i, j)) upleft = INT_MIN;
14      int max = std::max(left, std::max(upleft, up));
15      if (left == max) {
16        from[i][j] = 0;
17      } else if (upleft == max) {
18        from[i][j] = 1;
19      } else {
20        from[i][j] = 2;
21      }
22      left = max;
23    }
24    if (i >= n) {
25      int count = 0;
26      for (int x = i, y = n; y;) {
27        int t = from[x][y];
28        count += t == 1;
29        x += DELTA[t][0];
30        y += DELTA[t][1];
31      }
32      ret = std::max(ret, count);
33      int x = i - n + 1, y = 0;
34      from[x][0] = 0;
35      while (y <= n && from[x][y] == 0) y++;
36      for (; x <= i; ++ x) {
37        from[x][y] = 0;
38        if (x == i) break;
39        for (; y <= n; ++ y) {
40          if (from[x + 1][y] == 2) break;
41          if (y + 1 <= n && from[x + 1][y + 1] == 1) {
42            y ++; break;
43          }
44        }
45      }
46    }
47  }
```

```
48    return ret;
49  }
```

6.4 经纬度球面距离

```
1  double sphereDis(double lon1, double lat1, double lon2, double lat2, double R) {
2    return R * acos(cos(lat1) * cos(lat2) * cos(lon1 - lon2) + sin(lat1) * sin(lat2));
3  }
```

6.5 长方体表面两点最短距离

```
1  int r;
2  void turn(int i, int j, int x, int y, int z,int x0, int y0, int L, int W, int H) {
3    if (z==0) { int R = x*x+y*y; if (R<r) r=R;
4    } else {
5      if(i>=0 && i< 2) turn(i+1, j, x0+L+z, y, x0+L-x, x0+L, y0, H, W, L);
6      if(j>=0 && j< 2) turn(i, j+1, x, y0+W+z, y0+W-y, x0, y0+W, L, H, W);
7      if(i<=0 && i>-2) turn(i-1, j, x0-z, y, x-x0, x0-H, y0, H, W, L);
8      if(j<=0 && j>-2) turn(i, j-1, x, y0-z, y-y0, x0, y0-H, L, H, W);
9    }
10  }
11  int main(){
12    int L, H, W, x1, y1, z1, x2, y2, z2;
13    cin >> L >> W >> H >> x1 >> y1 >> z1 >> x2 >> y2 >> z2;
14    if (z1!=0 && z1!=H) if (y1==0 || y1==W)
15        swap(y1,z1), std::swap(y2,z2), std::swap(W,H);
16    else swap(x1,z1), std::swap(x2,z2), std::swap(L,H);
17    if (z1==H) z1=0, z2=H-z2;
18    r=0x3fffffff;
19    turn(0,0,x2-x1,y2-y1,z2,-x1,-y1,L,W,H);
20    cout<<r<<endl;
21  }
```

Chapter 7

其他

7.1   Java Hints

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

public class Main{
  static class Task{
    void solve(int testId, InputReader cin, PrintWriter cout) {
      // Write down the code you want
    }
  };

  public static void main(String args[]) {
    InputStream inputStream = System.in;
    OutputStream outputStream = System.out;
    InputReader in = new InputReader(inputStream);
    PrintWriter out = new PrintWriter(outputStream);
    TaskA solver = new TaskA();
    solver.solve(1, in, out);
    out.close();
  }

  static class InputReader {
    public BufferedReader reader;
    public StringTokenizer tokenizer;

    public InputReader(InputStream stream) {
      reader = new BufferedReader(new InputStreamReader(stream), 32768);
      tokenizer = null;
    }

    public String next() {
      while (tokenizer == null || !tokenizer.hasMoreTokens()) {
        try {
          tokenizer = new StringTokenizer(reader.readLine());
        } catch (IOException e) {
          throw new RuntimeException(e);
        }
      }
      return tokenizer.nextToken();
    }

    public int nextInt() {
      return Integer.parseInt(next());
    }

  }
};
// Arrays
int a[];
.fill(a[, int fromIndex, int toIndex],val); | .sort(a[, int fromIndex, int toIndex])
// String
```

```
52 String s;
53 .charAt(int i); | compareTo(String) | compareToIgnoreCase () | contains(String) |
54 length () | substring(int l, int len)
55 // BigInteger
56 .abs() | .add() | bitLength () | subtract () | divide () | remainder () | divideAndRemainder () |
   ↪   modPow(b, c) |
57 pow(int) | multiply () | compareTo () |
58 gcd() | intValue () | longValue () | isProbablePrime(int c) (1 - 1/2^c) |
59 nextProbablePrime () | shiftLeft(int) | valueOf ()
60 // BigDecimal
61 .ROUND_CEILING | ROUND_DOWN_FLOOR | ROUND_HALF_DOWN | ROUND_HALF_EVEN | ROUND_HALF_UP | ROUND_UP
62 .divide(BigDecimal b, int scale , int round_mode) | doubleValue () | movePointLeft(int) |
   ↪   pow(int) |
63 setScale(int scale , int round_mode) | stripTrailingZeros ()
64 // StringBuilder
65 StringBuilder sb = new StringBuilder ();
66 sb.append(elem) | out.println(sb)
67 // TODO Java STL 的使用方法以及上面这些方法的检验
```

7.2 vimrc

```
1  set ruler
2  set number
3  set smartindent
4  set autoindent
5  set tabstop=4
6  set softtabstop=4
7  set shiftwidth=4
8  set hlsearch
9  set incsearch
10 set autoread
11 set backspace=2
12 set mouse=a
13
14 syntax on
15
16 nmap <C-A> ggVG
17 vmap <C-C> "+y
18
19 filetype plugin indent on
20
21 autocmd FileType cpp set cindent
22 autocmd FileType cpp map <F9> :w <CR> :!g++ % -o %< -g -std=c++11 -Wall -Wextra -Wconversion &&
   ↪   size %< <CR>
23 autocmd FileType cpp map <C-F9> :!g++ % -o %< -std=c++11 -O2 && size %< <CR>
24 autocmd FileType cpp map <F8> :!time ./%< < %<.in <CR>
25 autocmd FileType cpp map <F5> :!time ./%< <CR>
26
27 map <F3> :vnew %<.in <CR>
28 map <F4> :!gedit % <CR>
```

7.3 常用结论

7.3.1 上下界网络流

$B(u,v)$ 表示边 (u,v) 流量的下界，$C(u,v)$ 表示边 (u,v) 流量的上界，$F(u,v)$ 表示边 (u,v) 的流量。设 $G(u,v) = F(u,v) - B(u,v)$，显然有

$$0 \le G(u,v) \le C(u,v) - B(u,v)$$

无源汇的上下界可行流

建立超级源点 S^* 和超级汇点 T^*，对于原图每条边 (u,v) 在新网络中连如下三条边：$S^* \to v$，容量为 $B(u,v)$；$u \to T^*$，容量为 $B(u,v)$；$u \to v$，容量为 $C(u,v) - B(u,v)$。最后求新网络的最大流，判断从超级源点 S^* 出发的边是否都满流即可，边 (u,v) 的最终解中的实际流量为 $G(u,v) + B(u,v)$。

有源汇的上下界可行流

从汇点 T 到源点 S 连一条上界为 ∞, 下界为 0 的边。按照**无源汇的上下界可行流**一样做即可, 流量即为 $T \to S$ 边上的流量。

有源汇的上下界最大流

1. 在**有源汇的上下界可行流**中, 从汇点 T 到源点 S 的边改为连一条上界为 ∞, 下届为 x 的边。x 满足二分性质, 找到最大的 x 使得新网络存在**无源汇的上下界可行流**即为原图的最大流。

2. 从汇点 T 到源点 S 连一条上界为 ∞, 下界为 0 的边, 变成无源汇的网络。按照**无源汇的上下界可行流**的方法, 建立超级源点 S^* 和超级汇点 T^*, 求一遍 $S^* \to T^*$ 的最大流, 再将从汇点 T 到源点 S 的这条边拆掉, 求一次 $S \to T$ 的最大流即可。

有源汇的上下界最小流

1. 在**有源汇的上下界可行流**中, 从汇点 T 到源点 S 的边改为连一条上界为 x, 下界为 0 的边。x 满足二分性质, 找到最小的 x 使得新网络存在**无源汇的上下界可行流**即为原图的最小流。

2. 按照**无源汇的上下界可行流**的方法, 建立超级源点 S^* 与超级汇点 T^*, 求一遍 $S^* \to T^*$ 的最大流, 但是注意这一次不加上汇点 T 到源点 S 的这条边, 即不使之改为无源汇的网络去求解。求完后, 再加上那条汇点 T 到源点 S 上界 ∞ 的边。因为这条边下界为 0, 所以 S^*, T^* 无影响, 再直接求一次 $S^* \to T^*$ 的最大流。若超级源点 S^* 出发的边全部满流, 则 $T \to S$ 边上的流量即为原图的最小流, 否则无解。

上下界费用流

来源: BZOJ 3876 设汇 t, 源 s, 超级源 S, 超级汇 T, 本质是每条边的下界为 1, 上界为 MAX, 跑一遍有源汇的上下界最小费用最小流。(因为上界无穷大, 所以只要满足所有下界的最小费用最小流)

1. 对每个点 x: 从 x 到 t 连一条费用为 0, 流量为 MAX 的边, 表示可以任意停止当前的剧情 (接下来的剧情从更优的路径去走, 画个样例就知道了)

2. 对于每一条边权为 z 的边 x->y:
   - 从 S 到 y 连一条流量为 1, 费用为 z 的边, 代表这条边至少要被走一次。
   - 从 x 到 y 连一条流量为 MAX, 费用为 z 的边, 代表这条边除了至少走的一次之外还可以随便走。
   - 从 x 到 T 连一条流量为 1, 费用为 0 的边。(注意是每一条 x->y 的边都连, 或者你可以记下 x 的出边数 Kx, 连一次流量为 Kx, 费用为 0 的边)。

建完图后从 S 到 T 跑一遍费用流, 即可。(当前跑出来的就是满足上下界的最小费用最小流了)

7.3.2 弦图相关

1. 团数 $\le$ 色数, 弦图团数 = 色数

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

3. 最小染色: 完美消除序列从后往前依次给每个点染色, 给每个点染上可以染的最小的颜色

4. 最大独立集: 完美消除序列从前往后能选就选

5. 弦图最大独立集数 = 最小团覆盖数, 最小团覆盖: 设最大独立集为 $\{p_1, p_2, \ldots, p_t\}$, 则 $\{p_1 \cup N(p_1), \ldots, p_t \cup N(p_t)\}$ 为最小团覆盖

7.3.3 Bernoulli 数

1. 初始化: $B_0(n) = 1$

2. 递推公式:

$$B_m(n) = n^m - \sum_{k=0}^{m-1} \binom{m}{k} \frac{B_k(n)}{m-k+1}$$

3. 应用:

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

7.4 常见错误

1. 数组或者变量类型开错，例如将 double 开成 int；

2. 函数忘记返回返回值；

3. 初始化数组没有初始化完全；

4. 对空间限制判断不足导致 MLE；

7.5 测试列表

1. 检测评测机是否开 O2；

2. 检测 ___int128 以及 ___float128 是否能够使用；

3. 检测是否能够使用 C++11；

4. 检测是否能够使用 Ext Lib；

5. 检测程序运行所能使用的内存大小；

6. 检测程序运行所能使用的栈大小；

7. 检测是否有代码长度限制；

8. 检测是否能够正常返 Runtime Error（assertion、return 1、空指针）；

9. 查清楚厕所方位和打印机方位；

7.6 博弈游戏

7.6.1 巴什博奕

1. 只有一堆 n 个物品，两个人轮流从这堆物品中取物，规定每次至少取一个，最多取 m 个。最后取光者得胜。

2. 显然，如果 $n = m+1$，那么由于一次最多只能取 m 个，所以，无论先取者拿走多少个，后取者都能够一次拿走剩余的物品，后者取胜。因此我们发现了如何取胜的法则：如果 $n = m+1\,r+s$，(r 为任意自然数，$s \leq m$)，那么先取者要拿走 s 个物品，如果后取者拿走 $k(k \leq m)$ 个，那么先取者再拿走 $m+1-k$ 个，结果剩下 $(m+1)(r-1)$ 个，以后保持这样的取法，那么先取者肯定获胜。总之，要保持给对手留下 $(m+1)$ 的倍数，就能最后获胜。

7.6.2 威佐夫博弈

1. 有两堆各若干个物品，两个人轮流从某一堆或同时从两堆中取同样多的物品，规定每次至少取一个，多者不限，最后取光者得胜。

2. 判断一个局势 (a, b) 为奇异局势（必败态）的方法：

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

7.6.3 阶梯博奕

1. 博弈在一列阶梯上进行，每个阶梯上放着自然数个点，两个人进行阶梯博弈，每一步则是将一个阶梯上的若干个点（至少一个）移到前面去，最后没有点可以移动的人输。

2. 解决方法：把所有奇数阶梯看成 N 堆石子，做 NIM。(把石子从奇数堆移动到偶数堆可以理解为拿走石子，就相当于几个奇数堆的石子在做 Nim)

7.6.4 图上删边游戏

链的删边游戏

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

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

树的删边游戏

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

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

局部连通图的删边游戏

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

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

7.7 常用数学公式

7.7.1 求和公式

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

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

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

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

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

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

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

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

7.7.2 斐波那契数列

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

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

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

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

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

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

7.7.3 错排公式

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

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

7.7.4 莫比乌斯函数

$$\mu(n) = \begin{cases} 1 & \text{若} n = 1 \\ (-1)^k & \text{若} n \text{无平方数因子,且} n = p_1 p_2 \ldots p_k \\ 0 & \text{若} n \text{有大于1的平方数因数} \end{cases}$$

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

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

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

7.7.5 Burnside 引理

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

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

7.7.6 五边形数定理

设 $p(n)$ 是 n 的拆分数,有

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

7.7.7 树的计数

1. 有根树计数:$n+1$ 个结点的有根树的个数为

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

其中,

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

2. 无根树计数:当 n 为奇数时,n 个结点的无根树的个数为

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

当 n 为偶数时,n 个结点的无根树的个数为

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

3. n 个结点的完全图的生成树个数为

$$n^{n-2}$$

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

7.7.8   欧拉公式

平面图的顶点个数、边数和面的个数有如下关系：

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

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

$$V - E + F = 2$$

7.7.9   皮克定理

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

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

7.7.10   牛顿恒等式

设

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

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

则

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

特别地，对于

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

有

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

7.8   平面几何公式

7.8.1   三角形

1. 半周长

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

2. 面积

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

3. 中线

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

4. 角平分线

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

5. 高线

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

6. 内切圆半径

$$r = \frac{S}{p} = \frac{arcsin\frac{B}{2} \cdot sin\frac{C}{2}}{sin\frac{B+C}{2}} = 4R \cdot sin\frac{A}{2} sin\frac{B}{2} sin\frac{C}{2}$$

$$= \sqrt{\frac{(p-a)(p-b)(p-c)}{p}} = p \cdot tan\frac{A}{2} tan\frac{B}{2} tan\frac{C}{2}$$

7. 外接圆半径

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

7.8.2 四边形

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

1. $a^2 + b^2 + c^2 + d^2 = D_1^2 + D_2^2 + 4M^2$

2. $S = \frac{1}{2}D_1 D_2 sinA$

3. 对于圆内接四边形

$$ac + bd = D_1 D_2$$

4. 对于圆内接四边形

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

7.8.3 正 n 边形

R 为外接圆半径，r 为内切圆半径

1. 中心角

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

2. 内角

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

3. 边长

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

4. 面积

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

7.8.4 圆

1. 弧长

$$l = rA$$

2. 弦长

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

3. 弓形高

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

4. 扇形面积

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

5. 弓形面积

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

7.8.5　棱柱

1. 体积

$$V = Ah$$

A 为底面积，h 为高

2. 侧面积

$$S = lp$$

l 为棱长，p 为直截面周长

3. 全面积

$$T = S + 2A$$

7.8.6　棱锥

1. 体积

$$V = Ah$$

A 为底面积，h 为高

2. 正棱锥侧面积

$$S = lp$$

l 为棱长，p 为直截面周长

3. 正棱锥全面积

$$T = S + 2A$$

7.8.7　棱台

1. 体积

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

A_1, A_2 为上下底面积，h 为高

2. 正棱台侧面积

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

p_1, p_2 为上下底面周长，l 为斜高

3. 正棱台全面积

$$T = S + A_1 + A_2$$

7.8.8　圆柱

1. 侧面积

$$S = 2\pi rh$$

2. 全面积

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

3. 体积

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

7.8.9　圆锥

1. 母线

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

2. 侧面积

$$S = \pi rl$$

3. 全面积

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

4. 体积

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

7.8.10　圆台

1. 母线

$$l = \sqrt{h^2 + (r_1 - r_2)^2}$$

2. 侧面积

$$S = \pi(r_1 + r_2)l$$

3. 全面积

$$T = \pi r_1(l + r_1) + \pi r_2(l + r_2)$$

4. 体积

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

7.8.11　球

1. 全面积

$$T = 4\pi r^2$$

2. 体积

$$V = \frac{4}{3}\pi r^3$$

7.8.12　球台

1. 侧面积

$$S = 2\pi r h$$

2. 全面积

$$T = \pi(2rh + r_1^2 + r_2^2)$$

3. 体积

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

7.8.13　球扇形

1. 全面积

$$T = \pi r(2h + r_0)$$

h 为球冠高，r_0 为球冠底面半径

2. 体积

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

7.9　立体几何公式

7.9.1　球面三角公式

设 a, b, c 是边长，A, B, C 是所对的二面角，有余弦定理

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

正弦定理

$$\frac{sinA}{sina} = \frac{sinB}{sinb} = \frac{sinC}{sinc}$$

三角形面积是 $A + B + C - \pi$

7.9.2　四面体体积公式

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

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

其中

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

7.10   附录

7.10.1   NTT 素数及原根列表

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

Definitions		Series

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

Series

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

In general:

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

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

Geometric series:

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

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

Harmonic series:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Identities Cont.		Trees

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

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

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

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

46. $\left\{ n \atop n-m \right\} = \sum_k \binom{m-n}{m+k}\binom{m+n}{n+k}\begin{bmatrix} m+k \\ k \end{bmatrix},$ **47.** $\begin{bmatrix} n \\ n-m \end{bmatrix} = \sum_k \binom{m-n}{m+k}\binom{m+n}{n+k}\left\{ m+k \atop k \right\},$

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.** $\begin{bmatrix} n \\ \ell+m \end{bmatrix}\binom{\ell+m}{\ell} = \sum_k \begin{bmatrix} k \\ \ell \end{bmatrix}\begin{bmatrix} n-k \\ m \end{bmatrix}\binom{n}{k}.$

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"

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

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

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
$$T_{i+1} - T_i = 1 + \sum_{j=0}^{i}T_j - 1 - \sum_{j=0}^{i-1}T_j$$
$$= T_i.$$

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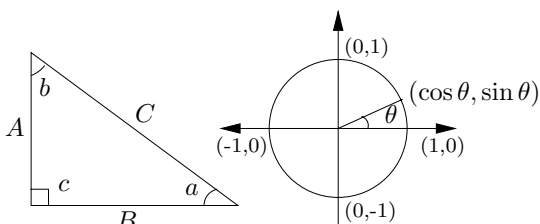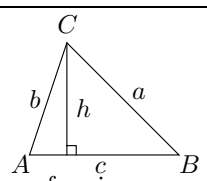
Expand this using partial fractions:
$$G(x) = x\left(\frac{2}{1-2x} - \frac{1}{1-x}\right)$$
$$= x\left(2\sum_{i\geq 0}2^i x^i - \sum_{i\geq 0}x^i\right)$$
$$= \sum_{i\geq 0}(2^{i+1} - 1)x^{i+1}.$$

So $g_i = 2^i - 1.$

Theoretical Computer Science Cheat Sheet

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

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:
$$1, \ \tfrac{3}{2}, \ \tfrac{11}{6}, \ \tfrac{25}{12}, \ \tfrac{137}{60}, \ \tfrac{49}{20}, \ \tfrac{363}{140}, \ \tfrac{761}{280}, \ \tfrac{7129}{2520}, \cdots$$
$$\ln n < H_n < \ln n + 1,$$
$$H_n = \ln n + \gamma + O\left(\frac{1}{n}\right).$$

Factorial, Stirling's approximation:
$$1, 2, 6, 24, 120, 720, 5040, 40320, 362880, \ldots$$
$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right).$$

Ackermann's function and inverse:
$$a(i,j) = \begin{cases} 2^j & i = 1 \\ a(i-1, 2) & j = 1 \\ a(i-1, a(i, j-1)) & i,j \geq 2 \end{cases}$$
$$\alpha(i) = \min\{j \mid a(j,j) \geq i\}.$$

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

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

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

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
$$E[g(X)] = \sum_x g(x) \Pr[X = x].$$
If X continuous then
$$E[g(X)] = \int_{-\infty}^{\infty} g(x) p(x)\, dx = \int_{-\infty}^{\infty} g(x)\, dP(x).$$

Variance, standard deviation:
$$\text{VAR}[X] = E[X^2] - E[X]^2,$$
$$\sigma = \sqrt{\text{VAR}[X]}.$$

For events A and B:
$$\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$$
$$\Pr[A \wedge B] = \Pr[A] \cdot \Pr[B],$$
$$\text{iff } A \text{ and } B \text{ are independent.}$$
$$\Pr[A|B] = \frac{\Pr[A \wedge B]}{\Pr[B]}$$

For random variables X and Y:
$$E[X \cdot Y] = E[X] \cdot E[Y],$$
$$\text{if } X \text{ and } Y \text{ are independent.}$$
$$E[X + Y] = E[X] + E[Y],$$
$$E[cX] = c\, E[X].$$

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

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

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

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

Matrices

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

Determinants: $\det A \neq 0$ iff A is non-singular.
$$\det A \cdot B = \det A \cdot \det B,$$
$$\det A = \sum_{\pi} \prod_{i=1}^{n} \operatorname{sign}(\pi) a_{i,\pi(i)}.$$

2×2 and 3×3 determinant:
$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc,$$
$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = g\begin{vmatrix} b & c \\ e & f \end{vmatrix} - h\begin{vmatrix} a & c \\ d & f \end{vmatrix} + i\begin{vmatrix} a & b \\ d & e \end{vmatrix}$$
$$= \begin{matrix} aei + bfg + cdh \\ - ceg - fha - ibd. \end{matrix}$$

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

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

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

More Trig.

Law of cosines:
$$c^2 = a^2 + b^2 - 2ab\cos C.$$

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

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

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

Theoretical Computer Science Cheat Sheet

Number Theory

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

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

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

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

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

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

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

Fermat's theorem:
$$1 \equiv a^{p-1} \bmod p.$$

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

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

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

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

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

Wilson's theorem: n is a prime iff
$$(n-1)! \equiv -1 \bmod n.$$

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

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

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

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 $k \cdot c(G - S) \leq |S|$.
k-Regular	A graph where all vertices have degree k.
k-Factor	A k-regular spanning subgraph.
Matching	A set of edges, no two of which are adjacent.
Clique	A set of vertices, all of which are adjacent.
Ind. set	A set of vertices, none of which are adjacent.
Vertex cover	A set of vertices which cover all edges.
Planar graph	A graph which can be embeded in the plane.
Plane graph	An embedding of a planar graph.

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

If G is planar then $n - m + f = 2$, so
$$f \leq 2n - 4, \quad m \leq 3n - 6.$$

Any planar graph has a vertex with degree ≤ 5.

Notation:

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

Geometry

Projective coordinates: triples (x, y, z), not all x, y and z zero.
$$(x, y, z) = (cx, cy, cz) \quad \forall c \neq 0.$$

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

Distance formula, L_p and L_∞ metric:
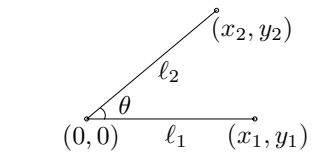$$\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2},$$
$$\left[|x_1 - x_0|^p + |y_1 - y_0|^p \right]^{1/p},$$
$$\lim_{p \to \infty} \left[|x_1 - x_0|^p + |y_1 - y_0|^p \right]^{1/p}.$$

Area of triangle (x_0, y_0), (x_1, y_1) and (x_2, y_2):
$$\tfrac{1}{2} \text{abs} \begin{vmatrix} x_1 - x_0 & y_1 - y_0 \\ x_2 - x_0 & y_2 - y_0 \end{vmatrix}.$$

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

<div align="center">

Theoretical Computer Science Cheat Sheet

</div>

π	Calculus

π

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

26. $\int \csc^n x\, dx = -\dfrac{\cot x \csc^{n-1} x}{n-1} + \dfrac{n-2}{n-1} \int \csc^{n-2} x\, dx, \quad n \neq 1,$ **27.** $\int \sinh x\, dx = \cosh x,$ **28.** $\int \cosh x\, dx = \sinh x,$

29. $\int \tanh x\, dx = \ln|\cosh x|,$ **30.** $\int \coth x\, dx = \ln|\sinh x|,$ **31.** $\int \operatorname{sech} x\, dx = \arctan \sinh x,$ **32.** $\int \operatorname{csch} x\, dx = \ln\left|\tanh \frac{x}{2}\right|,$

33. $\int \sinh^2 x\, dx = \frac{1}{4}\sinh(2x) - \frac{1}{2}x,$ **34.** $\int \cosh^2 x\, dx = \frac{1}{4}\sinh(2x) + \frac{1}{2}x,$ **35.** $\int \operatorname{sech}^2 x\, dx = \tanh x,$

36. $\int \operatorname{arcsinh} \frac{x}{a} dx = x \operatorname{arcsinh} \frac{x}{a} - \sqrt{x^2 + a^2}, \quad a > 0,$ **37.** $\int \operatorname{arctanh} \frac{x}{a} dx = x \operatorname{arctanh} \frac{x}{a} + \frac{a}{2} \ln|a^2 - x^2|,$

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

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

40. $\int \dfrac{dx}{a^2 + x^2} = \frac{1}{a} \arctan \frac{x}{a}, \quad a > 0,$ **41.** $\int \sqrt{a^2 - x^2}\, dx = \frac{x}{2}\sqrt{a^2 - x^2} + \frac{a^2}{2} \arcsin \frac{x}{a}, \quad a > 0,$

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

43. $\int \dfrac{dx}{\sqrt{a^2 - x^2}} = \arcsin \frac{x}{a}, \quad a > 0,$ **44.** $\int \dfrac{dx}{a^2 - x^2} = \frac{1}{2a} \ln\left|\dfrac{a+x}{a-x}\right|,$ **45.** $\int \dfrac{dx}{(a^2 - x^2)^{3/2}} = \dfrac{x}{a^2\sqrt{a^2 - x^2}},$

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

48. $\int \dfrac{dx}{ax^2 + bx} = \frac{1}{a} \ln\left|\dfrac{x}{a + bx}\right|,$ **49.** $\int x\sqrt{a + bx}\, dx = \dfrac{2(3bx - 2a)(a + bx)^{3/2}}{15b^2},$

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

52. $\int \dfrac{\sqrt{a^2 - x^2}}{x} dx = \sqrt{a^2 - x^2} - a \ln\left|\dfrac{a + \sqrt{a^2 - x^2}}{x}\right|,$ **53.** $\int x\sqrt{a^2 - x^2}\, dx = -\frac{1}{3}(a^2 - x^2)^{3/2},$

54. $\int x^2 \sqrt{a^2 - x^2}\, dx = \frac{x}{8}(2x^2 - a^2)\sqrt{a^2 - x^2} + \frac{a^4}{8} \arcsin \frac{x}{a}, \quad a > 0,$ **55.** $\int \dfrac{dx}{\sqrt{a^2 - x^2}} = -\frac{1}{a} \ln\left|\dfrac{a + \sqrt{a^2 - x^2}}{x}\right|,$

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

58. $\int \dfrac{\sqrt{a^2 + x^2}}{x} dx = \sqrt{a^2 + x^2} - a \ln\left|\dfrac{a + \sqrt{a^2 + x^2}}{x}\right|,$ **59.** $\int \dfrac{\sqrt{x^2 - a^2}}{x} dx = \sqrt{x^2 - a^2} - a \arccos \frac{a}{|x|}, \quad a > 0,$

60. $\int x\sqrt{x^2 \pm a^2}\, dx = \frac{1}{3}(x^2 \pm a^2)^{3/2},$ **61.** $\int \dfrac{dx}{x\sqrt{x^2 + a^2}} = \frac{1}{a} \ln\left|\dfrac{x}{a + \sqrt{a^2 + x^2}}\right|,$

Calculus Cont.	Finite Calculus

Calculus Cont.

62. $\int \dfrac{dx}{x\sqrt{x^2-a^2}} = \frac{1}{a}\arccos\frac{a}{|x|}, \quad a > 0,$ **63.** $\int \dfrac{dx}{x^2\sqrt{x^2\pm a^2}} = \mp\dfrac{\sqrt{x^2\pm a^2}}{a^2 x},$

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

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

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

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

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

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

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

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

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

74. $\int x^n e^{ax}\,dx = \dfrac{x^n e^{ax}}{a} - \dfrac{n}{a}\int x^{n-1}e^{ax}\,dx,$

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

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

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

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

Finite Calculus

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

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

Differences:
$$\Delta(cu) = c\Delta u, \qquad \Delta(u+v) = \Delta u + \Delta v,$$
$$\Delta(uv) = u\Delta v + \mathrm{E}\,v\Delta u,$$
$$\Delta(x^{\underline{n}}) = nx^{\underline{n-1}},$$
$$\Delta(H_x) = x^{\underline{-1}}, \qquad\qquad \Delta(2^x) = 2^x,$$
$$\Delta(c^x) = (c-1)c^x, \qquad \Delta\tbinom{x}{m} = \tbinom{x}{m-1}.$$

Sums:
$$\sum cu\,\delta x = c\sum u\,\delta x,$$
$$\sum (u+v)\,\delta x = \sum u\,\delta x + \sum v\,\delta x,$$
$$\sum u\Delta v\,\delta x = uv - \sum \mathrm{E}\,v\Delta u\,\delta x,$$
$$\sum x^{\underline{n}}\,\delta x = \tfrac{x^{\underline{n+1}}}{m+1}, \qquad \sum x^{\underline{-1}}\,\delta x = H_x,$$
$$\sum c^x\,\delta x = \tfrac{c^x}{c-1}, \qquad \sum \tbinom{x}{m}\,\delta x = \tbinom{x}{m+1}.$$

Falling Factorial Powers:
$$x^{\underline{n}} = x(x-1)\cdots(x-n+1), \quad n > 0,$$
$$x^{\underline{0}} = 1,$$
$$x^{\underline{n}} = \dfrac{1}{(x+1)\cdots(x+|n|)}, \quad n < 0,$$
$$x^{\underline{n+m}} = x^{\underline{m}}(x-m)^{\underline{n}}.$$

Rising Factorial Powers:
$$x^{\overline{n}} = x(x+1)\cdots(x+n-1), \quad n > 0,$$
$$x^{\overline{0}} = 1,$$
$$x^{\overline{n}} = \dfrac{1}{(x-1)\cdots(x-|n|)}, \quad n < 0,$$
$$x^{\overline{n+m}} = x^{\overline{m}}(x+m)^{\overline{n}}.$$

Conversion:
$$x^{\underline{n}} = (-1)^n(-x)^{\overline{n}} = (x-n+1)^{\overline{n}}$$
$$= 1/(x+1)^{\overline{-n}},$$
$$x^{\overline{n}} = (-1)^n(-x)^{\underline{n}} = (x+n-1)^{\underline{n}}$$
$$= 1/(x-1)^{\underline{-n}},$$
$$x^n = \sum_{k=1}^n \left\{{n \atop k}\right\} x^{\underline{k}} = \sum_{k=1}^n \left\{{n \atop k}\right\}(-1)^{n-k} x^{\overline{k}},$$
$$x^{\underline{n}} = \sum_{k=1}^n \left[{n \atop k}\right](-1)^{n-k} x^k,$$
$$x^{\overline{n}} = \sum_{k=1}^n \left[{n \atop k}\right] x^k.$$

<table>

</table>

Theoretical Computer Science Cheat Sheet

Series

Taylor's series:

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

Expansions:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Ordinary power series:

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

Exponential power series:

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

Dirichlet power series:

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

Binomial theorem:

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

Difference of like powers:

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

For ordinary power series:

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

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

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

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

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

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

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

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

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

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

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

Convolution:

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

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

<div align="center">
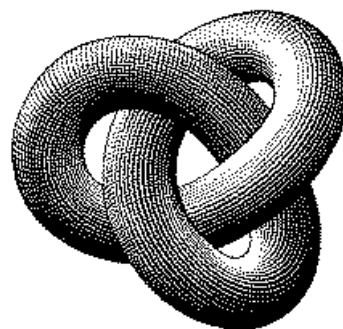
Theoretical Computer Science Cheat Sheet

</div>

Series	Escher's Knot

Series

Expansions:

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

$$x^{\overline{n}} = \sum_{i=0}^{\infty}\begin{bmatrix}n\\i\end{bmatrix}x^i,$$

$$\left(\ln\frac{1}{1-x}\right)^n = \sum_{i=0}^{\infty}\begin{bmatrix}i\\n\end{bmatrix}\frac{n!x^i}{i!},$$

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

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

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

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

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

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

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

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

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

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

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

$$\left(\frac{1}{x}\right)^{\overline{-n}} = \sum_{i=0}^{\infty}\left\{\begin{matrix}i\\n\end{matrix}\right\}x^i,$$

$$(e^x-1)^n = \sum_{i=0}^{\infty}\left\{\begin{matrix}i\\n\end{matrix}\right\}\frac{n!x^i}{i!},$$

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

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

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

Escher's Knot

Stieltjes Integration

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

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

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

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

If the integrals involved exist

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

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

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

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

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

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

Cramer's Rule

If we have equations:

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

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

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

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

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

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

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

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

Fibonacci Numbers

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

Definitions:

$$F_i = F_{i-1} + F_{i-2}, \quad F_0 = F_1 = 1,$$
$$F_{-i} = (-1)^{i-1}F_i,$$
$$F_i = \frac{1}{\sqrt{5}}\left(\phi^i - \hat\phi^i\right),$$

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

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

Additive rule:

$$F_{n+k} = F_k F_{n+1} + F_{k-1}F_n,$$
$$F_{2n} = F_n F_{n+1} + F_{n-1}F_n.$$

Calculation by matrices:

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