Standard Code Library

Tempest

October, 2014

Contents

Chapter 1

数论算法

1.1 $O(m^2 \log n)$ 求线性递推数列第 n 项

已知 $a_0, a_1, ..., a_{m-1}$
$a_n = c_0 * a_{n-m} + ... + c_{m-1} * a_{n-1}$
求 $a_n = v_0 * a_0 + v_1 * a_1 + ... + v_{m-1} * a_{m-1}$

```
1  void linear_recurrence(long long n, int m, int a[], int c[], int p) {
2      long long v[M] = {1 % p}, u[M << 1], msk = !!n;
3      for(long long i(n); i > 1; i >>= 1) {
4          msk <<= 1;
5      }
6      for(long long x(0); msk; msk >>= 1, x <<= 1) {
7          fill_n(u, m << 1, 0);
8          int b(!!(n & msk));
9          x |= b;
10         if(x < m) {
11             u[x] = 1 % p;
12         }else {
13             for(int i(0); i < m; i++) {
14                 for(int j(0), t(i + b); j < m; j++, t++) {
15                     u[t] = (u[t] + v[i] * v[j]) % p;
16                 }
17             }
18             for(int i((m << 1) - 1); i >= m; i--) {
19                 for(int j(0), t(i - m); j < m; j++, t++) {
20                     u[t] = (u[t] + c[j] * u[i]) % p;
21                 }
22             }
23         }
24         copy(u, u + m, v);
25     }
26     //a[n] = v[0] * a[0] + v[1] * a[1] + ... + v[m - 1] * a[m - 1].
27     for(int i(m); i < 2 * m; i++) {
```

```
28          a[i] = 0;
29          for(int j(0); j < m; j++) {
30              a[i] = (a[i] + (long long)c[j] * a[i + j - m]) % p;
31          }
32      }
33      for(int j(0); j < m; j++) {
34          b[j] = 0;
35          for(int i(0); i < m; i++) {
36              b[j] = (b[j] + v[i] * a[i + j]) % p;
37          }
38      }
39      for(int j(0); j < m; j++) {
40          a[j] = b[j];
41      }
42  }
```

1.2 NTT

```
 1  const int modulo(786433);
 2  const int G(10);//原根
 3  int pw[999999];
 4  void FFT(int P[], int n, int oper) {
 5      for(int i(1), j(0); i < n - 1; i++) {
 6          for(int s(n); j ^= s >>= 1, ~j & s;);
 7          if (i < j)
 8              swap(P[i], P[j]);
 9      }
10      int unit_p0;
11      for(int d(0); (1 << d) < n; d++) {
12          int m(1 << d), m2(m * 2);
13          unit_p0 = oper == 1?pw[(modulo - 1) / m2]:pw[modulo - 1 - (modulo - 1) / m2];
14          for(int i = 0; i < n; i += m2) {
15              int unit(1);
16              for(int j(0); j < m; j++) {
17                  int &P1 = P[i + j + m], &P2 = P[i + j];
18                  int t = (long long)unit * P1 % modulo;
19                  P1 = (P2 - t + modulo) % modulo;
20                  P2 = (P2 + t) % modulo;
21                  unit = (long long)unit * unit_p0 % modulo;
22              }
23          }
24      }
25  }
26
27  int nn;
28  int A[N], B[N], C[N];
29  //A * B = C;
30  //len = nn
```

```
31 void multiply() {
32     FFT(A, nn, 1);
33     FFT(B, nn, 1);
34     for(int i(0); i < nn; i++) {
35         C[i] = (long long)A[i] * B[i] % modulo;
36     }
37     FFT(C,  nn, -1);
38 }
39
40 int main() {
41     pw[0] = 1;
42     for(int i(1); i < modulo; i++) {
43         pw[i] = (long long)pw[i - 1] * G % modulo;
44     }
45 }
```

1.3 中国剩余定理

包括扩展欧几里得，求逆元，和保证除数互质条件下的 CRT

```
 1 LL x, y;
 2 void exGcd(LL a, LL b)
 3 {
 4     if (b == 0) {
 5         x = 1;
 6         y = 0;
 7         return;
 8     }
 9     exGcd(b, a % b);
10     LL k = y;
11     y = x - a / b * y;
12     x = k;
13 }
14
15 LL inversion(LL a, LL b)
16 {
17     exGcd(a, b);
18     return (x % b + b) % b;
19 }
20
21 LL CRT(vector<LL> m, vector<LL> a)
22 {
23     int N = m.size();
24     LL M = 1, ret = 0;
25     for(int i = 0; i < N; ++ i)
26         M *= m[i];
27
28     for(int i = 0; i < N; ++ i) {
29         ret = (ret + (M / m[i]) * a[i] % M * inversion(M / m[i], m[i])) % M;
```

```
30      }
31      return ret;
32  }
```

1.4 中国剩余定理 (可不互质)

```
 1  namespace number_theory_basic {
 2      inline void euclid(const long long &a, const long long &b, long long &x, long
            long &y) {
 3          if (b == 0) {
 4              x = 1;
 5              y = 0;
 6          } else {
 7              euclid(b, a % b, x, y);
 8              x -= a / b * y;
 9              swap(x, y);
10          }
11      }
12  }
13  namespace chinese_remainder_theorem {
14      inline bool crt(int n, long long r[], long long m[], long long &remainder, long
            long &modular) {
15          remainder = modular = 1;
16          for (int i = 1; i <= n; ++i) {
17              long long x, y;
18              euclid(modular, m[i], x, y);
19              long long divisor = gcd(modular, m[i]);
20              if ((r[i] - remainder) % divisor) {
21                  return false;
22              }
23              x *= (r[i] - remainder) / divisor;
24              remainder += modular * x;
25              modular *= m[i] / divisor;
26              ((remainder %= modular) += modular) %= modular;
27          }
28          return true;
29      }
30  }
```

1.5 Miller Rabin

miller_rabin_32 是针对 32 位以下整数的; miller_rabin_64 是针对 64 位以下整数的. 直接调用 prime() 函数, 当返回值是 true 时表示是素数, 否则不是质数.

```
1  namespace miller_rabin_32 {
2      int const n = 3;
3      int const base[] = {2, 7, 61};
```

```
 4
 5      inline long long power(int x, int k, int p) {
 6          long long ans = 1, num = x % p;
 7          for (int i = k; i > 0; i >>= 1) {
 8              if (i & 1) {
 9                  (ans *= num) %= p;
10              }
11              (num *= num) %= p;
12          }
13          return ans;
14      }
15
16      inline bool check(int p, int base) {
17          int n = p - 1;
18          while (!(n & 1)) {
19              n >>= 1;
20          }
21          long long m = power(base, n, p);
22          while (n != p - 1 && m != 1 && m != p - 1) {
23              (m *= m) %= p;
24              n <<= 1;
25          }
26          return m == p - 1 || (n & 1) == 1);
27      }
28
29      inline bool prime(int p) {
30          for (int i = 0; i < n; ++i) {
31              if (base[i] == p) {
32                  return true;
33              }
34          }
35          if (p == 1 || !(p & 1)) {
36              return false;
37          }
38          for (int i = 0; i < n; ++i) {
39              if (!check(p, base[i])) {
40                  return false;
41              }
42          }
43          return true;
44      }
45  }
46
47  namespace miller_rabin_64 {
48      int const n = 9;
49      int const base[] = {2, 3, 5, 7, 11, 13, 17, 19, 23};
50
51      inline long long multiply(const long long &x, const long long &y, const long long
            &p) {
```

```
52          long long ans = 0, num = x % p;
53          for (long long i = y; i > 0; i >>= 1) {
54              if (i & 1) {
55                  (ans += num) %= p;
56              }
57              (num += num) %= p;
58          }
59          return ans;
60      }
61
62      inline long long power(const long long &x, const long long &k, const long long &p
            ) {
63          long long ans = 1, num = x % p;
64          for (long long i = k; i > 0; i >>= 1) {
65              if (i & 1) {
66                  ans = multiply(ans, num, p);
67              }
68              num = multiply(num, num, p);
69          }
70          return ans;
71      }
72
73      inline bool check(const long long &p, const long long &base) {
74          long long n = p - 1;
75          while (!(n & 1)) {
76              n >>= 1;
77          }
78          long long m = power(base, n, p);
79          while (n != p - 1 && m != 1 && m != p - 1) {
80              m = multiply(m, m, p);
81              n <<= 1;
82          }
83          return m == p - 1 || (n & 1) == 1;
84      }
85
86      inline bool prime(const long long &p) {
87          for (int i = 0; i < n; ++i) {
88              if (base[i] == p) {
89                  return true;
90              }
91          }
92          if (p == 1 || !(p & 1)) {
93              return false;
94          }
95          for (int i = 0; i < n; ++i) {
96              if (!check(p, base[i])) {
97                  return false;
98              }
99          }
```

```
100             return true;
101         }
102     }
```

1.6 Pollard Rho

```
1   模板需要配合miller\_rabin一起使用.
2   调用factor()函数, 会返回vector<long long>, 表示分解结果. (例如分解12, 会返回2, 2和3)
3   namespace pollard_rho {
4       //可以改成LL*LL%LL的形式
5       inline long long multiply(const long long &x, const long long &y, const long long
            &p) {
6           long long ans = 0, num = x % p;
7           for (long long i = y; i > 0; i >>= 1) {
8               if (i & 1) {
9                   (ans += num) %= p;
10              }
11              (num += num) %= p;
12          }
13          return ans;
14      }
15
16      inline long long gcd(long long x, long long y) {
17          while (y > 0) {
18              x %= y;
19              swap(x, y);
20          }
21          return x;
22      }
23
24      inline long long pollard_rho(const long long &n, const long long &c) {
25          long long x = rand() % (n - 1) + 1, y = x;
26          int head = 1, tail = 2;
27          while (true) {
28              x = multiply(x, x, n);
29              if ((x += c) >= n) {
30                  x -= n;
31              }
32              if (x == y) {
33                  return n;
34              }
35              long long d = gcd(abs(x - y), n);
36              if (d > 1 && d < n) {
37                  return d;
38              }
39              if ((++head) == tail) {
40                  y = x;
```

```
41                 tail <<= 1;
42             }
43         }
44     }
45
46     inline vector<long long> mergy(const vector<long long> &a, const vector<long long
       > &b) {
47         vector<long long> vec;
48         for (int i = 0; i < (int)a.size(); ++i) {
49             vec.push_back(a[i]);
50         }
51         for (int i = 0; i < (int)b.size(); ++i) {
52             vec.push_back(b[i]);
53         }
54         return vec;
55     }
56
57     inline vector<long long> factor(const long long &n) {
58         if (n <= 1) {
59             return vector<long long>();
60         }
61         if (miller_rabin::prime(n)) {
62             return vector<long long>(1, n);
63         }
64         long long p = n;
65         while (p >= n) {
66             p = pollard_rho(n, rand() % (n - 1) + 1);
67         }
68         return mergy(factor(n / p), factor(p));
69     }
70 }
```

1.7 离散对数

```
1  #include <iostream>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <algorithm>
5  #include <cmath>
6  #include <map>
7  #include <cstring>
8
9  using namespace std;
10
11 typedef long long int64;
12
13 struct hash_table {
14     static const int MAXN = 100003;
```

```
15      int first[MAXN], key[MAXN], value[MAXN], next[MAXN], tot;
16      hash_table() : tot(0) {
17          memset(first, 255, sizeof first);
18      }
19      void clear() {
20          memset(first, 255, sizeof first);
21          tot = 0;
22      }
23      int &operator[] (const int &o) {
24          int pos = o % MAXN;
25          for (int i = first[pos]; i != -1; i = next[i])
26              if (key[i] == o)
27                  return value[i];
28          next[tot] = first[pos];
29          first[pos] = tot;
30          key[tot] = o;
31          return value[tot++];
32      }
33      bool has_key(const int &o) {
34          int pos = o % MAXN;
35          for (int i = first[pos]; i != -1; i = next[i])
36              if (key[i] == o)
37                  return true;
38          return false;
39      }
40  };
41
42  int discrete_log(int base, int n, int mod) {
43      int block = int(sqrt(mod)) + 1;
44      int val = 1;
45      hash_table dict;
46      for (int i = 0; i < block; ++i) {
47          if (dict.has_key(val) == 0)
48              dict[val] = i;
49          val = (int64)val * base % mod;
50      }
51      int inv = inverse(val, mod);
52      val = 1;
53      for (int i = 0; i < block; ++i) {
54          if (dict.has_key((int64)val * n % mod))
55              return dict[(int64)val * n % mod] + i * block;
56          val = (int64)val * inv % mod;
57      }
58      return -1;
59  }
60
61  int main() {
62      int base, n, p;
63      while (scanf("%d␣%d␣%d", &p, &base, &n) == 3) {
```

```
64          int ans = discrete_log(base, n, p);
65          if (ans == -1)
66              puts("no solution");
67          else
68              printf("%d\n", ans);
69      }
70  }
```

1.8 原根

```
 1  int primitive_root(int p) {
 2      int n = p - 1;
 3      while (true) {
 4          int root = rand() % (p - 1) + 1, m = n;
 5          bool found = true;
 6          for (int i = 0; i < (int)prim.size(); ++i) {
 7              int cur = prim[i];
 8              if (m / cur < cur)
 9                  break;
10              if (m % cur == 0) {
11                  if (pow_mod(root, n / cur, p) == 1) {
12                      found = false;
13                      break;
14                  }
15                  while (m % cur == 0)
16                      m /= cur;
17              }
18          }
19          if (m > 1)
20              if (pow_mod(root, n / m, p) == 1)
21                  found = false;
22          if (found)
23              return root;
24      }
25  }
26
27  vector<int> discrete_root(int expo, int n, int mod) {
28      if (n == 0)
29          return vector<int>(1, 0);
30      int g = primitive_root(mod);
31      int e = discrete_log(g, n, mod);
32      int64 u, v;
33      int d = extend_euclid(expo, mod - 1, u, v);
34      if (e % d != 0)
35          return vector<int>();
36      int64 delta = (mod - 1) / d;
37      u = u * e / d % delta;
38      if (u < 0)
```

```
39          u += delta;
40      vector<int> ret;
41      while (u < mod - 1) {
42          ret.push_back(pow_mod(g, u, mod));
43          u += delta;
44      }
45      return ret;
46  }
```

1.9 离散二次方根

```
 1  inline bool quad_resi(int x, int p) {
 2      return pow_mod(x, (p - 1) / 2, p) == 1;
 3  }
 4
 5  struct quad_poly {
 6      int zero, one, val, mod;
 7
 8      quad_poly(int zero, int one, int val, int mod) : zero(zero), one(one), val
 9              (val), mod(mod) {}
10
11      quad_poly multiply(quad_poly o) {
12          int z0 = (zero * o.zero + one * o.one % mod * val % mod) % mod;
13          int z1 = (zero * o.one + one * o.zero) % mod;
14          return quad_poly(z0, z1, val, mod);
15      }
16
17      quad_poly pow(int x) {
18          if (x == 1)
19              return *this;
20          quad_poly ret = this->pow(x / 2);
21          ret = ret.multiply(ret);
22          if (x & 1)
23              ret = ret.multiply(*this);
24          return ret;
25      }
26  };
27
28  inline int calc(int a, int p) {
29      a %= p;
30      if (a < 2)
31          return a;
32      if (!quad_resi(a, p))
33          return p;           // no solution
34      if (p % 4 == 3)
35          return pow_mod(a, (p + 1) / 4, p);
36      int b = 0;
37      while (quad_resi((my_sqr(b) - a + p) % p, p))
```

```
38          b = rand() % p;
39      quad_poly ret = quad_poly(b, 1, (my_sqr(b) - a + p) % p, p);
40      ret = ret.pow((p + 1) / 2);
41      return ret.zero;
42  }
```

1.10 牛顿迭代求平方根

```
1   //use newton-method to solve f(x) = 0
2   //init x0
3   //xi -> x(i + 1) = xi - f(xi) / f'(xi)
4   //O(N^2logN)
5   int64 square_root(int64 x) {
6       if (x <= 0)
7           return 0;
8       int64 last_root = -1, root = 1 << (bit_length(x) / 2);
9       while (true) {
10          int64 next_root = (root + x / root) >> 1;
11          if (next_root == last_root)
12              return min(next_root, root);
13          last_root = root;
14          root = next_root;
15      }
16  }
```

1.11 Pell 方程求根

$x^2 - n * y^2 = 1$

```
1   pair<int64, int64> solve_pell64(int64 n) {
2       const static int MAXC = 111;
3       int64 p[MAXC], q[MAXC], a[MAXC], g[MAXC], h[MAXC];
4       p[1] = 1; p[0] = 0;
5       q[1] = 0; q[0] = 1;
6       a[2] = square_root(n);
7       g[1] = 0; h[1] = 1;
8       for (int i = 2; ; ++i) {
9           g[i] = -g[i - 1] + a[i] * h[i - 1];
10          h[i] = (n - g[i] * g[i]) / h[i - 1];
11          a[i + 1] = (g[i] + a[2]) / h[i];
12          p[i] = a[i] * p[i - 1] + p[i - 2];
13          q[i] = a[i] * q[i - 1] + q[i - 2];
14          if (p[i] * p[i] - n * q[i] * q[i] == 1)
15              return make_pair(p[i], q[i]);
16      }
17  }
```

1.12 直线下整点个数

求 $\sum_{i=0}^{n-1}\lfloor\frac{a+bi}{m}\rfloor$.

```
typedef long long LL;

LL count(LL n, LL a, LL b, LL m) {
    if (b == 0) {
        return n * (a / m);
    }
    if (a >= m) {
        return n * (a / m) + count(n, a % m, b, m);
    }
    if (b >= m) {
        return (n − 1) * n / 2 * (b / m) + count(n, a, b % m, m);
    }
    return count((a + b * n) / m, (a + b * n) % m, m, b);
}
```

Chapter 2

数値算法

2.1 FFT

```
 1  void FFT(Complex P[], int n, int oper) {
 2      for (int i(1), j(0); i < n - 1; i++) {
 3          for (int s(n); j ^= s >>= 1, ~j & s;);
 4          if (i < j)
 5              swap(P[i], P[j]);
 6      }
 7      Complex unit_p0;
 8      for (int d(0); (1 << d) < n; d++) {
 9          int m(1 << d), m2(m * 2);
10          double p0(pi / m * oper);
11          unit_p0.imag(sin(p0));
12          unit_p0.real(cos(p0));
13          for (int i(0); i < n; i += m2) {
14              Complex unit = 1;
15              for (int j = 0; j < m; j++) {
16                  Complex &P1 = P[i + j + m], &P2 = P[i + j];
17                  Complex t = unit * P1;
18                  P1 = P2 - t;
19                  P2 = P2 + t;
20                  unit = unit * unit_p0;
21              }
22          }
23      }
24  }
25  void multiply() {
26      FFT(a, n, 1);
27      FFT(b, n, 1);
28      for(int i(0); i < n; i++) {
29          c[i] = a[i] * b[i];
30      }
31      FFT(c, n, -1);
```

```
32      for(int i(0); i < n; i++) {
33          ans[i] += (int)(c[i].real() / n + 0.5);
34      }
35  }
```

2.2 解一元三次方程 + 求三阶二次型的标准型

```
 1  double sqr(const double & x) {
 2      return x * x;
 3  }
 4  double eps(1e-8);
 5  int main() {
 6      double A, B, C, D, E, F;
 7      for(;6 == scanf("%lf%lf%lf%lf%lf%lf", &A, &B, &C, &D, &E, &F);) {
 8          D /= 2; E /= 2; F /= 2;
 9          complex<double> a(1), b(-A - B - C), c(A * B + B * C + C * A - sqr(D) - sqr(E
                ) - sqr(F)), d(-A * B * C - 2 * D * E * F + A * sqr(D) + B * sqr(E) + C *
                sqr(F));
10          complex<double> delta(pow(pow(b * c / 6. / a / a - b * b * b / 27. / a / a /
                a - d / 2. / a, 2) + pow(c / 3. / a - b * b / 9. / a / a, 3), 0.5));
11          complex<double> p(pow(b * c / 6. / a / a - b * b * b / 27. / a / a / a - d /
                2. / a + delta, 1. / 3));
12          complex<double> q(pow(b * c / 6. / a / a - b * b * b / 27. / a / a / a - d /
                2. / a - delta, 1. / 3));
13          complex<double> omega1(-0.5, 0.5 * sqrt(3.)), omega2(-0.5, -0.5 * sqrt(3.));
14          complex<double> x1(-b / 3. / a + p + q), x2(-b / 3. / a + omega1 * p + omega2
                 * q), x3(-b / 3. / a + omega2 * p + omega1 * q);
15          printf("%.10f\n", min(min(sqrt(1 / x1.real()), sqrt(1 / x2.real())), sqrt(1 /
                 x3.real())));
16      }
17  }
```

2.3 高斯消元

```
 1  vector<double> operator* (const vector<double> &a, double b) {
 2      vector<double> ret;
 3      for (int i = 0; i < (int)a.size(); ++i)
 4          ret.push_back(a[i] * b);
 5      return ret;
 6  }
 7
 8  vector<double> operator+ (const vector<double> &a, const vector<double> &b) {
 9      vector<double> ret;
10      for (int i = 0; i < (int)a.size(); ++i)
11          ret.push_back(a[i] + b[i]);
12      return ret;
```

```
13 }
14
15 vector<double> operator- (const vector<double> &a, const vector<double> &b) {
16     vector<double> ret;
17     for (int i = 0; i < (int)a.size(); ++i)
18         ret.push_back(a[i] - b[i]);
19     return ret;
20 }
21
22 struct solution {
23     int size, dimension;
24     vector<vector<double> > null_space;
25     vector<double> special;
26     solution(int size = 0, int dimension = 0) : size(size), dimension(dimension)
27             {
28         special = vector<double>(size, 0);
29         null_space = vector<vector<double> >(size, vector<double>(dimension,
30                 0));
31     }
32 };
33
34 solution gauss_elimination(vector<vector<double> > a, vector<double> b) {
35     int n = (int)a.size(), m = (int)a[0].size();
36     static const int MAX_SIZE = 211;
37     int index[MAX_SIZE], row = 0;
38     bool pivot[MAX_SIZE];
39     fill(index, index + n, -1);
40     fill(pivot, pivot + m, false);
41
42     for (int col = 0; row < n && col < m; ++col) {
43         int best = row;
44         for (int i = row + 1; i < n; ++i)
45             if (fabs(a[i][col]) > fabs(a[best][col]))
46                 best = i;
47         swap(a[best], a[row]);
48         swap(b[best], b[row]);
49         if (fabs(a[row][col]) < EPS)
50             continue;
51         pivot[col] = true;
52         index[row] = col;
53         double coef = a[row][col];
54         a[row] = a[row] * (1. / coef);
55         b[row] = b[row] * (1. / coef);
56         for (int i = 0; i < n; ++i)
57             if (i != row && fabs(a[i][col]) > EPS) {
58                 double coef = a[i][col];
59                 a[i] = a[i] - a[row] * coef;
60                 b[i] = b[i] - b[row] * coef;
61             }
```

```
62          ++row;
63      }
64
65      for (int i = row; i < n; ++i)
66          if (fabs(b[i]) > EPS)
67              return solution(0, 0);                  //no solution
68
69      solution ret(m, m - row);
70      for (int i = 0; i < row; ++i)
71          ret.special[index[i]] = b[i];
72
73      int cnt = 0;
74      for (int i = 0; i < m; ++i)
75          if (!pivot[i]) {
76              for (int j = 0; j < row; ++j)
77                  ret.null_space[index[j]][cnt] = a[j][i];
78              ret.null_space[i][cnt++] = -1;
79          }
80      return ret;
81  }
```

2.4 最小二乘法

```
1  // calculate argmin ||AX - B||
2  solution least_squares(vector<vector<double> > a, vector<double> b) {
3      int n = (int)a.size(), m = (int)a[0].size();
4      vector<vector<double> > p(m, vector<double>(m, 0));
5      vector<double> q(m, 0);
6      for (int i = 0; i < m; ++i)
7          for (int j = 0; j < m; ++j)
8              for (int k = 0; k < n; ++k)
9                  p[i][j] += a[k][i] * a[k][j];
10     for (int i = 0; i < m; ++i)
11         for (int j = 0; j < n; ++j)
12             q[i] += a[j][i] * b[j];
13     return gauss_elimination(p, q);
14 }
```

2.5 多项式求根

```
1  const double eps=1e-12;
2  double a[10][10];
3  typedef vector<double> vd;
4  int sgn(double x) { return x < -eps ? -1 : x > eps; }
5  double mypow(double x,int num){
6      double ans=1.0;
```

```
7       for(int i=1;i<=num;++i)ans*=x;
8       return ans;
9   }
10  double f(int n,double x){
11      double ans=0;
12      for(int i=n;i>=0;--i)ans+=a[n][i]*mypow(x,i);
13      return ans;
14  }
15  double getRoot(int n,double l,double r){
16      if(sgn(f(n,l))==0)return l;
17      if(sgn(f(n,r))==0)return r;
18      double temp;
19      if(sgn(f(n,l))>0)temp=-1;else temp=1;
20      double m;
21      for(int i=1;i<=10000;++i){
22          m=(l+r)/2;
23          double mid=f(n,m);
24          if(sgn(mid)==0){
25              return m;
26          }
27          if(mid*temp<0)l=m;else r=m;
28      }
29      return (l+r)/2;
30  }
31  vd did(int n){
32      vd ret;
33      if(n==1){
34          ret.push_back(-1e10);
35          ret.push_back(-a[n][0]/a[n][1]);
36          ret.push_back(1e10);
37          return ret;
38      }
39      vd mid=did(n-1);
40      ret.push_back(-1e10);
41      for(int i=0;i+1<mid.size();++i){
42          int t1=sgn(f(n,mid[i])),t2=sgn(f(n,mid[i+1]));
43          if(t1*t2>0)continue;
44          ret.push_back(getRoot(n,mid[i],mid[i+1]));
45      }
46      ret.push_back(1e10);
47      return ret;
48  }
49  int main(){
50      int n; scanf("%d",&n);
51      for(int i=n;i>=0;--i){
52          scanf("%lf",&a[n][i]);
53      }
54      for(int i=n-1;i>=0;--i)
55          for(int j=0;j<=i;++j)a[i][j]=a[i+1][j+1]*(j+1);
```

```
56      vd ans=did(n);
57      sort(ans.begin(),ans.end());
58      for(int i=1;i+1<ans.size();++i)printf("%.10f\n",ans[i]);
59      return 0;
60  }
```

2.6 自适应辛普森

```
 1  namespace adaptive_simpson {
 2      template<typename function>
 3      inline double area(function f, const double &left, const double &right) {
 4          double mid = (left + right) / 2;
 5          return (right - left) * (f(left) + 4 * f(mid) + f(right)) / 6;
 6      }
 7
 8      template<typename function>
 9      inline double simpson(function f, const double &left, const double &right, const
            double &eps, const double &area_sum) {
10          double mid = (left + right) / 2;
11          double area_left = area(f, left, mid);
12          double area_right = area(f, mid, right);
13          double area_total = area_left + area_right;
14          if (fabs(area_total - area_sum) <= 15 * eps) {
15              return area_total + (area_total - area_sum) / 15;
16          }
17          return simpson(f, left, right, eps / 2, area_left) + simpson(f, mid, right,
                eps / 2, area_right);
18      }
19
20      template<typename function>
21      inline double simpson(function f, const double &left, const double &right, const
            double &eps) {
22          return simpson(f, left, right, eps, area(f, left, right));
23      }
24  }
```

Chapter 3

计算几何

3.1 圆与多边形交

```
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <cmath>
#include <vector>
using namespace std;

const double eps = 5e-7;
const int N = 2222;
const double pi = acos(-1.0);

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

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

struct Point {
    double x, y;
    Point (double x = 0, double y = 0) : x(x), y(y) {}
    friend inline Point operator +(const Point &a, const Point &b) {
        return Point(a.x + b.x, a.y + b.y);
    }
    friend inline Point operator -(const Point &a, const Point &b) {
        return Point(a.x - b.x, a.y - b.y);
    }
    friend inline Point operator *(const Point &a, double k) {
        return Point(a.x * k, a.y * k);
    }
```

```
32      friend inline Point operator /(const Point &a, double k) {
33          return Point(a.x / k, a.y / k);
34      }
35      double dist() const {
36          return hypot(x, y);
37          return sqrt(x * x + y * y);
38      }
39      double dist2() const {
40          return x * x + y * y;
41      }
42      double ang() const {
43          return atan2(y, x);
44      }
45  };
46
47  vector<Point> convex;
48
49  int n;
50  double radius;
51  Point points[N][2];
52  Point target;
53
54  double det(Point a, Point b, Point c) {
55      return (b.x - a.x) * (c.y - a.y) - (c.x - a.x) * (b.y - a.y);
56  }
57
58  double dot(Point a, Point b, Point c) {
59      return (b.x - a.x) * (c.x - a.x) + (b.y - a.y) * (c.y - a.y);
60  }
61
62  double det(Point a, Point b) {
63      return a.x * b.y - b.x * a.y;
64  }
65
66  double dot(Point a, Point b) {
67      return a.x * b.x + a.y * b.y;
68  }
69
70  inline bool point_on_line(const Point &a, const Point &b, const Point &c) {
71      return sign(det(Point(0, 0), a - b, c - b)) == 0 && dot(Point(0, 0), b - a, c - a
            ) < eps;
72  }
73
74  double point_to_line(const Point &a, const Point &b, const Point &c) {
75      return fabs(det(Point(0, 0), c - b, a - b)) / (b - c).dist();
76  }
77
78  Point project_to_line(const Point &p, const Point &a, const Point &b) {
79      return a + (b - a) * dot(Point(0, 0), p - a, b - a) / sqr((b - a).dist());
```

```
 80 }
 81
 82 Point intersect(Point a, Point b, Point c, Point d) {
 83     double s1 = det(a, b, c);
 84     double s2 = det(a, b, d);
 85     return (c * s2 - d * s1) / (s2 - s1);
 86 }
 87
 88 inline Point line_to_circle(const Point &a, const Point &b) {
 89     double x = sqrt(sqr(radius) - sqr(point_to_line(Point(0, 0), a, b)));
 90     return project_to_line(Point(0, 0), a, b) - (b - a) / (b - a).dist() * x;
 91 }
 92
 93 inline double area_tri(Point a, Point b) {
 94     return det(Point(0, 0), a, b) / 2;
 95 }
 96
 97 inline double area_cir(Point a, Point b, double radius) {
 98     if (sign(det(Point(0, 0), a, b)) == 0)
 99         return 0;
100     a = a / a.dist() * radius;
101     b = b / b.dist() * radius;
102     double d = atan2(det(Point(0, 0), a, b), dot(Point(0, 0), a, b));
103     //printf("%f\n", sqr(radius) * d / 2);
104     return sqr(radius) * d / 2;
105 }
106
107 int intersect(const Point &a, const Point &b, Point &u, Point &v, double radius) {
108     if (point_to_line(Point(0, 0), a, b) + eps > radius)
109         return 0;
110     u = line_to_circle(a, b);
111     v = line_to_circle(b, a);
112     return point_on_line(u, a, b) + point_on_line(v, a, b);
113 }
114
115 vector<Point> calc(vector<Point> vec, Point a, Point b) {
116     vector<Point> result;
117     for(int i = 0; i < (int)vec.size(); i++) {
118         Point c = vec[i], d = vec[(i + 1) % (int)vec.size()];
119         if (det(a, b, c) > -eps) {
120             result.push_back(c);
121         }
122         if (sign(det(a, b, c)) * sign(det(a, b, d)) == -1) {
123             result.push_back(intersect(a, b, c, d));
124         }
125     }
126     return result;
127 }
128 double areaCT(double R, Point pa, Point pb)
```

```
129 {
130     if (pa.dist() < pb.dist()) swap(pa, pb);
131     if (pb.dist() < eps) return 0;
132     Point pc = pb - pa;
133     double a = pb.dist(), b = pa.dist(), c = pc.dist();
134     double cosB = dot(pb, pc) / a / c, B = acos(cosB);
135     double cosC = dot(pa, pb) / a / b, C = acos(cosC);
136     double S, h, theta;
137     if (a > R) {
138         S = C * 0.5 * R * R;
139         h = a * b * sin(C) / c;
140         if (h < R && B < pi * 0.5)
141             S -= acos(h / R) * R * R - h * sqrt(max(0.0, R * R - h * h));
142     } else if (b > R) {
143         theta = pi - B - asin(sin(B) / R * a);
144         S = 0.5 * a * R * sin(theta) + (C - theta) * 0.5 * R * R;
145     } else {
146         S = 0.5 * sin(C) * a * b;
147     }
148     return S;
149 }
150
151 void solve() {
152     scanf("%lf%d", &radius, &n);
153     convex.clear();
154     convex.push_back(Point(-radius, -radius));
155     convex.push_back(Point(radius, -radius));
156     convex.push_back(Point(radius, radius));
157     convex.push_back(Point(-radius, radius));
158     for(int i = 1; i <= n; i++) {
159         scanf("%lf%lf%lf%lf", &points[i][0].x, &points[i][0].y, &points[i][1].x, &
               points[i][1].y);
160     }
161     scanf("%lf␣%lf", &target.x, &target.y);
162     for(int i = 1; i <= n; i++) {
163         if (det(points[i][0], points[i][1], target) < -eps) {
164             swap(points[i][0], points[i][1]);
165         }
166         convex = calc(convex, points[i][0], points[i][1]);
167     }
168     double ans = 0;
169     for(int i = 0; i < (int)convex.size(); i++) {
170         ans += areaCT(radius, convex[i], convex[(i + 1) % (int)convex.size()]) * sign
               (det(convex[i], convex[(i + 1) % (int)convex.size()]));
171     }
172     printf("%.5f", max(0., fabs(ans) / (pi * radius * radius) * 100));
173     puts("%");
174 }
175
```

```
176 int main() {
177     int test;
178     scanf("%d", &test);
179     while(test--) {
180         static int testCount = 0;
181         printf("Case␣%d:␣", ++testCount);
182         solve();
183     }
184 }
```

3.2 动态凸包

```
  1 #define x first
  2 #define y second
  3 typedef map<int, int> mii;
  4 typedef map<int, int>::iterator mit;
  5 struct point { // something omitted
  6     point(const mit &p): x(p->first), y(p->second) {}
  7 };
  8 inline bool checkInside(mii &a, const point &p) { // `border inclusive`
  9     int x = p.x, y = p.y;
 10     mit p1 = a.lower_bound(x);
 11     if (p1 == a.end()) return false;
 12     if (p1->x == x) return y <= p1->y;
 13     if (p1 == a.begin()) return false;
 14     mit p2(p1--);
 15     return sign(det(p - point(p1), point(p2) - p)) >= 0;
 16 }
 17 inline void addPoint(mii &a, const point &p) { // `no collinear points`
 18     int x = p.x, y = p.y;
 19     mit pnt = a.insert(make_pair(x, y)).first, p1, p2;
 20     for (pnt->y = y; ; a.erase(p2)) {
 21         p1 = pnt;
 22         if (++p1 == a.end())
 23             break;
 24         p2 = p1;
 25         if (++p1 == a.end())
 26             break;
 27         if (det(point(p2) - p, point(p1) - p) < 0)
 28             break;
 29     }
 30     for ( ; ; a.erase(p2)) {
 31         if ((p1 = pnt) == a.begin())
 32             break;
 33         if (--p1 == a.begin())
 34             break;
 35         p2 = p1--;
 36         if (det(point(p2) - p, point(p1) - p) > 0)
```

```
37              break;
38      }
39  }
```

`upperHull $\leftarrow (x, y)$` `lowerHull $\leftarrow (x, -y)$`

3.3 farmland

```
1
2   const int N = 11111, M = 111111 * 4;
3
4   struct eglist {
5       int other[M], succ[M], last[M], sum;
6       void clear() {
7           memset(last, -1, sizeof(last));
8           sum = 0;
9       }
10      void addEdge(int a, int b) {
11          other[sum] = b, succ[sum] = last[a], last[a] = sum++;
12          other[sum] = a, succ[sum] = last[b], last[b] = sum++;
13      }
14  }e;
15
16  int n, m;
17  struct point {
18      int x, y;
19      point(int x, int y) : x(x), y(y) {}
20      point() {}
21      friend point operator -(point a, point b) {
22          return point(a.x - b.x, a.y - b.y);
23      }
24      double arg() {
25          return atan2(y, x);
26      }
27  }points[N];
28
29  vector<pair<int, double> > vecs;
30  vector<int> ee[M];
31  vector<pair<double, pair<int, int> > > edges;
32  double length[M];
33  int tot, father[M], next[M], visit[M];
34
35  int find(int x) {
36      return father[x] == x ? x : father[x] = find(father[x]);
37  }
38
39  long long det(point a, point b) {
40      return 1LL * a.x * b.y - 1LL * b.x * a.y;
41  }
```

```
42
43 double dist(point a, point b) {
44     return sqrt(1.0 * (a.x - b.x) * (a.x - b.x) + 1.0 * (a.y - b.y) * (a.y - b.y));
45 }
46
47 int main() {
48     scanf("%d␣%d", &n, &m);
49     e.clear();
50     for(int i = 1; i <= n; i++) {
51         scanf("%d␣%d", &points[i].x, &points[i].y);
52     }
53     for(int i = 1; i <= m; i++) {
54         int a, b;
55         scanf("%d␣%d", &a, &b);
56         e.addEdge(a, b);
57     }
58     for(int x = 1; x <= n; x++) {
59         vector<pair<double, int> > pairs;
60         for(int i = e.last[x]; ~i; i = e.succ[i]) {
61             int y = e.other[i];
62             pairs.push_back(make_pair((points[y] - points[x]).arg(), i));
63         }
64         sort(pairs.begin(), pairs.end());
65         for(int i = 0; i < (int)pairs.size(); i++) {
66             next[pairs[(i + 1) % (int)pairs.size()].second ^ 1] = pairs[i].second;
67         }
68     }
69     memset(visit, 0, sizeof(visit));
70     tot = 0;
71     for(int start = 0; start < e.sum; start++) {
72         if (visit[start])
73             continue;
74         long long total = 0;
75         int now = start;
76         vecs.clear();
77         while(!visit[now]) {
78             visit[now] = 1;
79             total += det(points[e.other[now ^ 1]], points[e.other[now]]);
80             vecs.push_back(make_pair(now / 2, dist(points[e.other[now ^ 1]], points[e
                .other[now]])));
81             now = next[now];
82         }
83         if (now == start && total > 0) {
84             ++tot;
85             for(int i = 0; i < (int)vecs.size(); i++) {
86                 ee[vecs[i].first].push_back(tot);
87             }
88         }
89     }
```

```
90
91      for(int i = 0; i < e.sum / 2; i++) {
92          int a = 0, b = 0;
93          if (ee[i].size() == 0)
94              continue;
95          else if (ee[i].size() == 1) {
96              a = ee[i][0];
97          } else if (ee[i].size() == 2) {
98              a = ee[i][0], b = ee[i][1];
99          }
100         edges.push_back(make_pair(dist(points[e.other[i * 2]], points[e.other[i * 2 +
                1]]), make_pair(a, b)));
101     }
102     sort(edges.begin(), edges.end());
103     for(int i = 0; i <= tot; i++)
104         father[i] = i;
105     double ans = 0;
106     for(int i = 0; i < (int)edges.size(); i++) {
107         int a = edges[i].second.first, b = edges[i].second.second;
108         double v = edges[i].first;
109         if (find(a) != find(b)) {
110             ans += v;
111             father[father[a]] = father[b];
112         }
113     }
114     printf("%.5f\n", ans);
115 }
```

3.4 farmland 完全体

```
1
2  const int MAXN = 200;
3  const int MAXV = MAXN * MAXN;
4  const int MAXE = MAXV * 6;
5  const double eps = 1e-8;
6
7  int sign(double x) {
8      return x < -eps ? -1 : x > eps;
9  }
10
11 struct Point {
12     double x, y;
13
14     Point(int x, int y) : x(x), y(y) {}
15     Point() {}
16
17     Point &operator +=(const Point &o) {
18         x += o.x;
```

```
19          y += o.y;
20          return *this;
21      }
22
23      Point &operator -=(const Point &o) {
24          x -= o.x;
25          y -= o.y;
26          return *this;
27      }
28
29      Point &operator *=(double k) {
30          x *= k;
31          y *= k;
32          return *this;
33      }
34
35      Point &operator /=(double k) {
36          x /= k;
37          y /= k;
38          return *this;
39      }
40
41      double norm2() const {
42          return x * x + y * y;
43      }
44
45      double norm() const {
46          return sqrt(norm2());
47      }
48
49      double arg() const {
50          return atan2(y, x);
51      }
52
53      bool on(const Point &, const Point &) const;
54      bool in(const vector<Point> &) const;
55  };
56
57  bool operator <(const Point &a, const Point &b) {
58      return sign(a.x - b.x) < 0 || sign(a.x - b.x) == 0 && sign(a.y - b.y) < 0;
59  }
60
61  bool operator ==(const Point &a, const Point &b) {
62      return sign(a.x - b.x) == 0 && sign(a.y - b.y) == 0;
63  }
64
65  Point operator +(Point a, const Point &b) {
66      return a += b;
67  }
```

```
68
69  Point operator -(Point a, const Point &b) {
70      return a -= b;
71  }
72
73  Point operator /(Point a, double k) {
74      return a /= k;
75  }
76
77  Point operator *(Point a, double k) {
78      return a *= k;
79  }
80
81  Point operator *(double k, Point a) {
82      return a *= k;
83  }
84
85  double det(const Point &a, const Point &b) {
86      return a.x * b.y - b.x * a.y;
87  }
88
89  double dot(const Point &a, const Point &b) {
90      return a.x * b.x + a.y * b.y;
91  }
92
93  bool parallel(const Point &a, const Point &b, const Point &c, const Point &d) {
94      return sign(det(b - a, d - c)) == 0;
95  }
96
97  Point intersect(const Point &a, const Point &b, const Point &c, const Point &d) {
98      double s1 = det(b - a, c - a);
99      double s2 = det(b - a, d - a);
100     return (c * s2 - d * s1) / (s2 - s1);
101 }
102
103 bool Point::on(const Point &a, const Point &b) const {
104     const Point &p = *this;
105     return sign(det(p - a, p - b)) == 0 && sign(dot(p - a, p - b)) <= 0;
106 }
107
108 bool Point::in(const vector<Point> &polygon) const {
109     const Point &p = *this;
110     int n = polygon.size();
111     int count = 0;
112     for (int i = 0; i < n; ++ i) {
113         const Point &a = polygon[i];
114         const Point &b = polygon[(i + 1) % n];
115         if (p.on(a, b)){
116             return false;
```

```
117         }
118         int t0 = sign(det(a - p, b - p));
119         int t1 = sign(a.y - p.y);
120         int t2 = sign(b.y - p.y);
121         count += t0 > 0 && t1 <= 0 && t2 > 0;
122         count -= t0 < 0 && t2 <= 0 && t1 > 0;
123     }
124     return count != 0;
125 }
126
127 struct eglist {
128     int other[MAXE], succ[MAXE], last[MAXE], sum;
129     set<pair<int, int> > Edges;
130     void clear() {
131         memset(last, -1, sizeof(last));
132         sum = 0;
133         Edges.clear();
134     }
135     void addEdge(int a, int b) {
136         if (Edges.count(make_pair(a, b)))
137             return;
138         Edges.insert(make_pair(a, b));
139         other[sum] = b, succ[sum] = last[a], last[a] = sum;
140         sum++;
141     }
142     void _addEdge(int a, int b) {
143         addEdge(a, b);
144         addEdge(b, a);
145     }
146 }e, topo;
147
148 vector<Point> Points;
149
150 Point segments[MAXE][2];
151 double W, H;
152 int n, next[MAXE];
153 vector<double> areas, allAreas;
154 vector<vector<Point> > regions;
155
156 void addSegment(Point a, Point b) {
157     segments[n][0] = a;
158     segments[n][1] = b;
159     n++;
160 }
161
162 int getPointID(const Point &p) {
163     return lower_bound(Points.begin(), Points.end(), p) - Points.begin();
164 }
165
```

```
166 const int VERTEX = 0;
167 const int EDGE = 1;
168 const int REGION = 2;
169
170 int getID(int type, int id) {
171     if (type == VERTEX) {
172         return id;
173     }
174     if (type == EDGE) {
175         return id + Points.size();
176     }
177     if (type == REGION) {
178         return id + Points.size() + e.sum / 2;
179     }
180     assert(false);
181 }
182
183 double getArea(int id) {
184     id -= Points.size() + e.sum / 2;
185     return id < 0 ? 0 : areas[id];
186 }
187
188 int locate(const Point &p) {
189     for (int i = 0; i < e.sum; i += 2) {
190         if (p.on(Points[e.other[i]], Points[e.other[i ^ 1]])) {
191             return getID(EDGE, i >> 1);
192         }
193     }
194     int best = -1;
195     for (int i = 0; i < regions.size(); ++i) {
196         if (p.in(regions[i]) && (best == -1 || allAreas[best] > allAreas[i])) {
197             best = i;
198         }
199     }
200     return getID(REGION, best);
201 }
202
203 vector<string> colorNames;
204 map<string, int> colorIDs;
205
206 int getColorID(const char *color) {
207     if (!colorIDs.count(color)) {
208         colorNames.push_back(color);
209         int newID = colorIDs.size();
210         colorIDs[color] = newID;
211     }
212     return colorIDs[color];
213 }
214
```

```
215 int color[MAXV * 10];
216
217 void paint(const Point &p, const char * c) {
218     int start = locate(p);
219     int old = color[start];
220     int cid = getColorID(c);
221     if (old == cid)
222         return;
223     queue<int> q;
224     q.push(start);
225     color[start] = cid;
226     while(!q.empty()) {
227         int x = q.front();
228         q.pop();
229         for (int i = topo.last[x]; ~i; i = topo.succ[i]) {
230             int y = topo.other[i];
231             if (color[y] == old) {
232                 color[y] = cid;
233                 q.push(y);
234             }
235         }
236     }
237 }
238
239 int main() {
240     freopen("input.txt", "r", stdin);
241     //freopen("output.txt", "w", stdout);
242     scanf("%lf␣%lf␣%d", &W, &H, &n);
243     for (int i = 0; i < n; i++) {
244         scanf("%lf␣%lf␣%lf␣%lf", &segments[i][0].x, &segments[i][0].y, &segments[i
                ][1].x, &segments[i][1].y);
245     }
246     addSegment(Point(0, 0), Point(W, 0));
247     addSegment(Point(W, 0), Point(W, H));
248     addSegment(Point(W, H), Point(0, H));
249     addSegment(Point(0, H), Point(0, 0));
250
251     for (int i = 0; i < n; i++) {
252         Points.push_back(segments[i][0]);
253         Points.push_back(segments[i][1]);
254         for (int j = 0; j < i; j++) {
255             if (!parallel(segments[i][0], segments[i][1], segments[j][0], segments[j
                    ][1])) {
256                 Point p = intersect(segments[i][0], segments[i][1], segments[j][0],
                        segments[j][1]);
257                 if (p.on(segments[i][0], segments[i][1]) && p.on(segments[j][0],
                        segments[j][1])) {
258                     Points.push_back(p);
259                 }
```

```
260             }
261         }
262     }
263     sort(Points.begin(), Points.end());
264     Points.erase(unique(Points.begin(), Points.end()), Points.end());
265 
266     e.clear();
267     for (int i = 0; i < n; i++) {
268         vector<pair<double, int> > pairs;
269         for (int j = 0; j < Points.size(); j++) {
270             if (Points[j].on(segments[i][0], segments[i][1]))
271                 pairs.push_back(make_pair((Points[j] - segments[i][0]).norm(), j));
272         }
273         sort(pairs.begin(), pairs.end());
274         for (int i = 1; i < pairs.size(); i++) {
275             e.addEdge(pairs[i - 1].second, pairs[i].second);
276             e.addEdge(pairs[i].second, pairs[i - 1].second);
277         }
278     }
279 
280     for (int u = 0; u < Points.size(); u++) {
281         vector<pair<double, int> > pairs;
282         for (int iter = e.last[u]; ~iter; iter = e.succ[iter]) {
283             pairs.push_back(make_pair((Points[e.other[iter]] - Points[u]).arg(), iter
                ));
284         }
285         sort(pairs.begin(), pairs.end());
286         for(int i = 0; i < pairs.size(); i++) {
287             next[pairs[(i + 1) % pairs.size()].second ^ 1] = pairs[i].second;
288         }
289     }
290 
291     vector<pair<Point, double> > waits;
292     static bool visit[MAXV];
293     memset(visit, 0, sizeof(visit));
294     for (int start = 0; start < e.sum; ++start) {
295         if (!visit[start]) {
296             int v = start;
297             double totalArea = 0;
298             vector <Point> region;
299             for(; !visit[v]; v = next[v]) {
300                 visit[v] = true;
301                 totalArea += det(Points[e.other[v ^ 1]], Points[e.other[v]]);
302                 region.push_back(Points[e.other[v]]);
303             }
304 
305             if (sign(totalArea) > 0) {
306                 regions.push_back(region);
307                 areas.push_back(totalArea);
```

```
308                 allAreas.push_back(totalArea);
309             } else {
310                 waits.push_back(make_pair(region.front(), -totalArea));
311             }
312         }
313     }
314
315     //build
316     topo.clear();
317     for (int i = 0; i < e.sum; i++) {
318         topo._addEdge(getID(EDGE, i >> 1), getID(VERTEX, e.other[i]));
319     }
320     for (int i = 0; i < regions.size(); i++) {
321         topo._addEdge(getID(REGION, i), getID(VERTEX, getPointID(regions[i].front()))
            );
322     }
323     for (int iter = 0; iter < waits.size(); iter++) {
324         const Point &p = waits[iter].first;
325         int best = -1;
326         for (int i = 0; i < regions.size(); i++) {
327             if (p.in(regions[i]) && (best == -1 || allAreas[best] > allAreas[i])) {
328                 best = i;
329             }
330         }
331         if (best != -1) {
332             areas[best] -= waits[iter].second;
333             topo._addEdge(getID(REGION, best), getID(VERTEX, getPointID(p)));
334         }
335     }
336
337
338     getColorID("white");
339     getColorID("blake");
340     getColorID("__COLOR__");
341
342     for (int i = 0; i < regions.size(); i++) {
343         color[getID(REGION, i)] = getColorID("white");
344     }
345     for (int i = 0; i < Points.size(); i++) {
346         color[getID(VERTEX, i)] = getColorID("black");
347     }
348     for(int i = 0; i < e.sum / 2; i++) {
349         color[getID(EDGE, i)] = getColorID("black");
350     }
351     paint(Point(0, 0), "__COLOR__");
352     int m;
353     scanf("%d", &m);
354     while (m --) {
355         Point p;
```

```
356         char buffer[16];
357         scanf("%lf␣%lf␣%s", &p.x, &p.y, buffer);
358         paint(p, buffer);
359     }
360
361     map<string, double> answer;
362     for (int i = 0; i < Points.size() + (e.sum >> 1) + regions.size(); ++i) {
363         const string &name = colorNames[color[i]];
364         if (name != "__COLOR__") {
365             answer[name] += getArea(i);
366         }
367     }
368     for (map<string, double> :: iterator iter = answer.begin(); iter != answer.end();
          ++ iter) {
369         printf("%s␣%.8lf\n", iter->first.c_str(), 0.5 * iter->second);
370     }
371 }
```

3.5 半平面交

```
 1 struct Border {
 2     point p1, p2; double alpha;
 3     Border() : p1(), p2(), alpha(0.0) {}
 4     Border(const point &a, const point &b): p1(a), p2(b), alpha( atan2(p2.y - p1.y,
          p2.x - p1.x) ) {}
 5     bool operator == (const Border &b) const {
 6         return sign(alpha - b.alpha) == 0;
 7     }
 8     bool operator < (const Border &b) const {
 9         int c = sign(alpha - b.alpha); if (c != 0) return c > 0;
10         return sign(det(b.p2 - b.p1, p1 - b.p1)) >= 0;
11     }
12 };
13 point isBorder(const Border &a, const Border &b) { // a and b should not be parallel
14     point is;
15     lineIntersect(a.p1, a.p2, b.p1, b.p2, is);
16     return is;
17 }
18 bool checkBorder(const Border &a, const Border &b, const Border &me) {
19     point is;
20     lineIntersect(a.p1, a.p2, b.p1, b.p2, is);
21     return sign(det(me.p2 - me.p1, is - me.p1)) > 0;
22 }
23 double HPI(int N, Border border[]) {
24     static Border que[MAXN * 2 + 1]; static point ps[MAXN];
25     int head = 0, tail = 0, cnt = 0; // [head, tail)
26     sort(border, border + N);
27     N = unique(border, border + N) - border;
```

```
28      for (int i = 0; i < N; ++i) {
29          Border &cur = border[i];
30          while (head + 1 < tail && !checkBorder(que[tail - 2], que[tail - 1], cur))
31              --tail;
32          while (head + 1 < tail && !checkBorder(que[head], que[head + 1], cur))
33              ++head;
34          que[tail++] = cur;
35      }
36      while (head + 1 < tail && !checkBorder(que[tail - 2], que[tail - 1], que[head]))
37          --tail;
38      while (head + 1 < tail && !checkBorder(que[head], que[head + 1], que[tail - 1]))
39          ++head;
40      if (tail - head <= 2)
41          return 0.0;
42      //Foru(i, a, b) : a <= i < b
43      Foru(i, head, tail)
44          ps[cnt++] = isBorder(que[i], que[(i + 1 == tail) ? (head) : (i + 1)]);
45      double area = 0;
46      Foru(i, 0, cnt)
47          area += det(ps[i], ps[(i + 1) % cnt]);
48      return fabs(area * 0.5); // or (-area * 0.5)
49  }
```

3.6 三维绕轴旋转

```
 1  const double pi = acos(-1.0);
 2  int n, m; char ch1; bool flag;
 3  double a[4][4], s1, s2, x, y, z, w, b[4][4], c[4][4];
 4  double sqr(double x)
 5  {
 6      return x*x;
 7  }
 8  int main()
 9  {
10      scanf("%d\n", &n);
11      memset(b, 0, sizeof(b));
12      b[0][0] = b[1][1] = b[2][2] = b[3][3] = 1;//initial matrix
13      for(int i = 1; i <= n; i++)
14      {
15          scanf("%c", &ch1);
16          if(ch1 == 'T')
17          {
18              scanf("%lf␣%lf␣%lf\n", &x, &y, &z);//plus each coordinate by a number (x,
                    y, z)
19              memset(a, 0, sizeof(a));
20              a[0][0] = 1; a[3][0] = x;
21              a[1][1] = 1; a[3][1] = y;
22              a[2][2] = 1; a[3][2] = z;
```

```
23              a[3][3] = 1;
24          }else if(ch1 == 'S')
25          {
26              scanf("%lf␣%lf␣%lf\n", &x, &y, &z);//multiply each coordinate by a number
                    (x, y, z)
27              memset(a, 0, sizeof(a));
28              a[0][0] = x;
29              a[1][1] = y;
30              a[2][2] = z;
31              a[3][3] = 1;
32          }else
33          {
34              scanf("%lf␣%lf␣%lf␣%lf\n", &x, &y, &z, &w);
35              //大拇指指向x轴正方向时，4指弯曲由y轴正方向指向z轴正方向
36              //大拇指沿着原点到点(x, y, z)的向量，4指弯曲方向旋转w度
37              w = w*pi/180;
38              memset(a, 0, sizeof(a));
39              s1 = x*x+y*y+z*z;
40              a[3][3] = 1;
41              a[0][0] = ((y*y+z*z)*cos(w)+x*x)/s1;            a[0][1] = x*y*(1-cos(w))/
                    s1+z*sin(w)/sqrt(s1);  a[0][2] = x*z*(1-cos(w))/s1-y*sin(w)/sqrt(s1);
42              a[1][0] = x*y*(1-cos(w))/s1-z*sin(w)/sqrt(s1);  a[1][1] = ((x*x+z*z)*cos(
                    w)+y*y)/s1;            a[1][2] = y*z*(1-cos(w))/s1+x*sin(w)/sqrt(s1);
43              a[2][0] = x*z*(1-cos(w))/s1+y*sin(w)/sqrt(s1);  a[2][1] = y*z*(1-cos(w))/
                    s1-x*sin(w)/sqrt(s1);  a[2][2] = ((x*x+y*y)*cos(w)+z*z)/s1;
44          }
45          memset(c, 0, sizeof(c));
46          for(int i = 0; i < 4; i++)
47              for(int j = 0; j < 4; j++)
48                  for(int k = 0; k < 4; k++)
49                      c[i][j] += b[i][k]*a[k][j];
50          memcpy(b, c, sizeof(c));
51      }
52      scanf("%d", &m);
53      for(int i = 1; i <= m; i++)
54      {
55          scanf("%lf%lf%lf", &x, &y, &z);//initial vector
56          printf("%lf␣%lf␣%lf\n", x*b[0][0]+y*b[1][0]+z*b[2][0]+b[3][0], x*b[0][1]+y*b
                [1][1]+z*b[2][1]+b[3][1], x*b[0][2]+y*b[1][2]+z*b[2][2]+b[3][2]);
57      }
58      return 0;
59  }
```

3.7 点到凸包切线

???

3.8 直线凸包交点

```
int n;
double eps(1e-8);
int sign(const double & x) {
    return (x > eps) - (x + eps < 0);
}
struct Point {
    double x, y;
    void scan() {
        scanf("%lf%lf", &x, &y);
    }
    void print() {
        printf("%lf %lf\n", x, y);
    }
    Point() {
    }
    Point(const double & x, const double & y) : x(x), y(y) {
    }
};
Point operator + (const Point & a, const Point & b) {
    return Point(a.x + b.x, a.y + b.y);
}
Point operator - (const Point & a, const Point & b) {
    return Point(a.x - b.x, a.y - b.y);
}
Point operator * (const double & a, const Point & b) {
    return Point(a * b.x, a * b.y);
}
double operator * (const Point & a, const Point & b) {
    return a.x * b.y - a.y * b.x;
}
bool isUpper(const Point & a) {
    return sign(a.x) < 0 or sign(a.x) == 0 and sign(a.y > 0);
}
Point crs(const Point & as, const Point & at, const Point & bs, const Point & bt) {
    if(sign((at - as) * (bt - bs)) == 0) {
        return bs;
    }
    double lambda((bs - as) * (bt - bs) / ((at - as) * (bt - bs)));
    return as + lambda * (at - as);
}
struct reca {
    Point a[50000];
    double s[50000];
    Point & operator [] (int x) {
        assert(x % n < 50000);
        return a[x % n];
    }
```

```
48      void init() {
49          s[0] = a[0] * a[1];
50          for(int i(1); i < n; i++) {
51              s[i] = s[i - 1] + a[i] * (i == n - 1?a[0]:a[i + 1]);
52          }
53      }
54
55      double getS(int le, int ri) {
56          if(le > ri)
57              return 0;
58          le %= n;
59          ri %= n;
60          if(le <= ri) {
61              return s[ri] - (le?s[le - 1]:0);
62          }else {
63              return getS(le, n - 1) + getS(0, ri);
64          }
65      }
66  } a;
67
68  int lowerBound(int le, int ri, const Point & dir) {
69      while(le < ri) {
70          int mid((le + ri) / 2);
71          if(sign((a[mid + 1] - a[mid]) * dir) >= 0) {
72              le = mid + 1;
73          }else {
74              ri = mid;
75          }
76      }
77      return le;
78  }
79  int boundLower(int le, int ri, const Point & s, const Point & t) {
80      while(le < ri) {
81          int mid((le + ri + 1) / 2);
82          if(sign((a[mid] - s) * (t - s)) >= 0) {
83              le = mid;
84          }else {
85              ri = mid - 1;
86          }
87      }
88      return le;
89  }
90  bool check(const Point & a, const Point & b, const Point & c, const Point & d) {
91      return sign((a - c) * (d - c)) * sign((b - c) * (d - c)) <= 0;
92  }
93  bool f[55555];
94  int main() {
95      scanf("%d", &n);
96      for(int i(0); i < n; i++) {
```

```
97          //printf("%d\n", n);
98          a[i].scan();
99          //return 0;
100     }
101     //return 0;
102     for(int i(0); i < n; i++) {
103         int d(sign((a[i + 1] - a[i]) * (a[i + 2] - a[i + 1])));
104         if(d) {
105             if(d < 0) {
106                 reverse(a.a, a.a + n);
107             }
108             break;
109         }
110     }
111     for(int i(0); i < n; i++) {
112         if(!sign(a[i].x - a[i + 1].x) and !sign(a[i].y - a[i + 1].y)) {
113             f[i] = false;
114         }else {
115             f[i] = true;
116         }
117     }
118     int n1(0);
119     for(int i(0); i < n; i++) {
120         if(f[i]) {
121             a[n1++] = a[i];
122         }
123     }
124     n = n1;
125     //现在a必须是严格逆时针凸包
126     a.init();
127     int i1, j1;
128     for(int i(0); i < n; i++) {
129         if(isUpper(a[i + 1] - a[i])) {
130             for(int j(i + 1); j != i; ++j %= n) {
131                 if(!isUpper(a[j + 1] - a[j])) {
132                     i1 = i; j1 = j;
133                     break;
134                 }
135             }
136             break;
137         }
138     }
139     if(i1 > j1) {
140         j1 += n;
141     }
142     int m;
143     scanf("%d", &m);
144     for(int i(0); i < m; i++) {
145         Point s, t;
```

```
146         s.scan(); t.scan();
147         if(!isUpper(t - s)) {
148             swap(t, s);
149         }
150         int i3(lowerBound(i1, j1, t - s));
151         int j3(lowerBound(j1, i1 + n, s - t));
152         int i4(boundLower(i3, j3, s, t));
153         int j4(boundLower(j3, i3 + n, t, s));
154         if(check(a[i4], a[i4 + 1], s, t)) {
155             Point p1(crs(s, t, a[i4], a[i4 + 1]));
156             Point p2(crs(s, t, a[j4], a[j4 + 1]));
157             if(sign(p1.x - p2.x) or sign(p1.y - p2.y)) {
158                 assert(i4 % n != j4 % n);
159                 double area1(p1 * a[i4 + 1] + a.getS(i4 + 1, j4 - 1) + a[j4] * p2 +
                        p2 * p1);
160                 double area2(p2 * a[j4 + 1] + a.getS(j4 + 1, i4 + n - 1) + a[i4] * p1
                         + p1 * p2);
161                 printf("%.6f\n", min(fabs(area1), fabs(area2)) / 2);
162             }else {
163                 printf("0.000000\n");
164             }
165         }else {
166             printf("0.000000\n");
167         }
168     }
169 }
```

3.9 exhausted_robot 凸多边形卡壳 + 凸多边形交

```
 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;
10     Point () {
11     }
12     Point(const double & x, const double & y) : x(x), y(y) {
13     }
14     void scan() {
15         scanf("%lf%lf", &x, &y);
16     }
17     double sqrlen() const {
18         return x * x + y * y;
19     }
```

```
20      double len() const  {
21          return sqrt(sqrlen());
22      }
23      Point zoom(const double & l) const {
24          double lambda(l / len());
25          return Point(lambda * x, lambda * y);
26      }
27      Point rev() const {
28          return Point(-y, x);
29      }
30      void print() const {
31          printf("(%f␣%f)\n", x, y);
32      }
33  };
34
35  vector<Point> blocks[22], denied[22], robot;
36
37  vector<pair<double, int> > vec;
38
39  bool f[111];
40
41  Point operator - (const Point & a, const Point & b) {
42      return Point(a.x - b.x, a.y - b.y);
43  }
44  Point operator + (const Point & a, const Point & b) {
45      return Point(a.x + b.x, a.y + b.y);
46  }
47  Point operator * (const double & a, const Point & b) {
48      return Point(a * b.x, a * b.y);
49  }
50  double operator * (const Point & a, const Point & b) {
51      return a.x * b.y - a.y * b.x;
52  }
53  double operator % (const Point & a, const Point & b) {
54      return a.x * b.x + a.y * b.y;
55  }
56
57  bool operator < (const Point & a, const Point & b) {
58      if(!equal(a.x, b.x))
59          return a.x < b.x;
60      else if(!equal(a.y, b.y));
61          return a.y < b.y;
62      return false;
63  }
64  bool operator == (const Point & a, const Point & b) {
65      return equal(a.x, b.x) and equal(a.y, b.y);
66  }
67
68  void scan(vector<Point> & vec) {
```

```
69      vec.clear();
70      int x;
71      scanf("%d", &x);
72      for(int i(0); i < x; i++) {
73          Point tmp;
74          tmp.scan();
75          vec.push_back(tmp);
76      }
77  }
78
79  Point intersect(const Point & as, const Point & ad, const Point & bs, const Point &
        bd) {
80      double lambda((bs - as) * bd / (ad * bd));
81      return as + lambda * ad;
82  }
83
84  void cut(vector<Point> & vec, const Point & s, const Point & d) {
85      vector<Point> vec1;
86      for(int i(0); i < (int)vec.size(); i++) {
87          if(sign((vec[i] - s) * d) <= 0) {
88              vec1.push_back(vec[i]);
89          }
90          if(sign((vec[i] - s) * d) * sign((vec[(i + 1) % (int)vec.size()] - s) * d) <
                0) {
91              vec1.push_back(intersect(s, d, vec[i], vec[(i + 1) % (int)vec.size()] -
                    vec[i]));
92          }
93      }
94      vec = vec1;
95  }
96
97  int mi;
98
99  Point getMax(const Point & norm) {
100     Point res(robot[0]);
101     mi = 0;
102     for(int i(0); i < (int)robot.size(); i++) {
103         if(sign(robot[i] % norm - res % norm) > 0) {
104             res = robot[i];
105             mi = i;
106         }
107     }
108     return res;
109 }
110
111 bool vecCmp(const pair<double, int> & a, const pair<double, int> & b) {
112     if(!equal(a.first, b.first))
113         return a.first < b.first;
114     else
```

```
115         return a.second > b.second;
116 }
117
118 bool vecEql(const pair<double, int> & a, const pair<double, int> & b) {
119     return equal(a.first, b.first) and a.second == b.second;
120 }
121
122 void print(const vector<Point> & vec) {
123     printf("print:\n");
124     for(int i(0); i < (int)vec.size(); i++) {
125         vec[i].print();
126     }
127     printf("endprint\n");
128 }
129
130 void getConvex(vector<Point> & vec) {
131     sort(vec.begin(), vec.end());
132     vector<Point> vec1;
133     for(int i(0); i < (int)vec.size(); i++) {
134         while(vec1.size() >= 2 and sign((vec1.back() - vec1[(int)vec1.size() - 2]) *
                (vec[i] - vec1.back())) <= 0)
135             vec1.pop_back();
136         vec1.push_back(vec[i]);
137     }
138     vector<Point> vec2;
139     for(int i((int)vec.size() - 1); i >= 0; i--) {
140         while(vec2.size() >= 2 and sign((vec2.back() - vec2[(int)vec2.size() - 2]) *
                (vec[i] - vec2.back())) <= 0)
141             vec2.pop_back();
142         vec2.push_back(vec[i]);
143     }
144     vec.clear();
145     for(int i(0); i + 1 < (int)vec1.size(); i++)
146         vec.push_back(vec1[i]);
147     for(int i(0); i + 1 < (int)vec2.size(); i++)
148         vec.push_back(vec2[i]);
149 }
150
151 int main() {
152     int tst;
153     scanf("%d", &tst);
154     for(int qq(1); qq <= tst; qq++) {
155         int n;
156         scanf("%d", &n);
157         for(int i(0); i < n; i++)
158             scan(blocks[i]);
159         scan(robot);
160         double x1, y1, x2, y2;
161         scanf("%lf%lf%lf%lf", &x1, &y1, &x2, &y2);
```

```
162         x1 += robot[0].x - getMax(Point(-1, 0)).x;
163         y1 += robot[0].y - getMax(Point(0, -1)).y;
164         x2 -= getMax(Point(1, 0)).x - robot[0].x;
165         y2 -= getMax(Point(0, 1)).y - robot[0].y;
166         double ans((x2 - x1) * (y2 - y1));
167         for(int i(0); i < n; i++) {
168             int siz(blocks[i].size());
169             denied[i].clear();
170             int p1, p2;
171             p1 = 0;
172             getMax((blocks[i][1] - blocks[i][0]).rev());
173             p2 = mi;
174             denied[i].push_back(blocks[i][0] + robot[0] - robot[mi]);
175             for(int j1(1), j2(mi); j1 != p1 or j2 != p2; ) {
176                 denied[i].push_back(blocks[i][j1] + robot[0] - robot[j2]);
177                 Point dir((blocks[i][(j1 + 1) % (int)blocks[i].size()] - blocks[i][j1
                        ]).rev());
178                 getMax(dir);
179                 if(equal(robot[j2] % dir, robot[mi] % dir))
180                     ++j1 %= (int)blocks[i].size();
181                 else
182                     ++j2 %= (int)robot.size();
183             }
184         }
185         for(int i(0); i < n; i++) {
186             cut(denied[i], Point(x1, y1), Point(x2 - x1, 0));
187             cut(denied[i], Point(x2, y1), Point(0, y2 - y1));
188             cut(denied[i], Point(x2, y2), Point(x1 - x2, 0));
189             cut(denied[i], Point(x1, y2), Point(0, y1 - y2));
190             for(int j(0); j < (int)denied[i].size(); j++) {
191                 f[j] = !(denied[i][j] == denied[i][(j + 1) % (int)denied[i].size()]);
192             }
193             getConvex(denied[i]);
194             denied[i].push_back(denied[i].front());
195         }
196         for(int i(0); i < n; i++) {
197             for(int j(0); j + 1 < (int)denied[i].size(); j++) {
198                 vec.clear();
199                 vec.push_back(make_pair(0., 0));
200                 vec.push_back(make_pair(1., 0));
201                 Point norm(denied[i][j + 1] - denied[i][j]);
202                 Point a(denied[i][j]), b(denied[i][j + 1]);
203                 norm = norm.zoom(1 / norm.len());
204                 for(int k(0); k < n; k++) if(k != i) {
205                     int sz(vec.size());
206                     for(int l(0); l + 1 < (int)denied[k].size(); l++) {
207                         Point c(denied[k][l]), d(denied[k][l + 1]);
208                         int s1(sign((c - a) * norm));
209                         int s2(sign((d - a) * norm));
```

```
210                         if(!s1 and !s2 and k < i and sign((d - c) % norm) > 0) {
211                             vec.push_back(make_pair((c - a) % norm, 1));
212                             vec.push_back(make_pair((d - a) % norm, -1));
213                         } else if(s1 <= 0 and s2 > 0 or s1 > 0 and s2 <= 0) {
214                             double a1((d - c) * (a - c));
215                             double a2((d - c) * (b - c));
216                             vec.push_back(make_pair(a1 / (a1 - a2), (s1 < 0 or s2 >
                                    0)?1:-1));
217                         }
218                     }
219                 }
220                 sort(vec.begin(), vec.end(), vecCmp);
221                 int cnt(0);
222                 double tot(0);
223                 for(int k(0); k + 1 < (int)vec.size(); k++) {
224                     cnt += vec[k].second;
225                     if(cnt == 0 and sign(vec[k].first) >= 0 and sign(vec[k + 1].first
                            - 1) <= 0) {
226                         tot += vec[k + 1].first - vec[k].first;
227                     }
228                 }
229                 ans -= tot * (denied[i][j] * denied[i][j + 1]) / 2;
230             }
231         }
232         printf("Case #%d: %.3f\n", qq, ans);
233     }
234 }
```

3.10 判断圆存在交集 O(nlogk)

传入 n 个圆，圆心存在 cir 中，半径存在 radius 中，nlogk 判断是否存在交集

```
1  int n;
2  double sx, sy, d;
3  vector<Point> cir;
4  vector<double> radius;
5
6  int isIntersectCircleToCircle(Point c1, double r1, Point c2, double r2)
7  {
8      double dis = c1.distTo(c2);
9      return sign(dis - (r1 + r2)) <= 0;
10 }
11
12 void getRange(double x, Point &c, double r, double &retl, double &retr)
13 {
14     double tmp = sqrt(max(r * r - (c.x - x) * (c.x - x), 0.0));
15     retl = c.y - tmp; retr = c.y + tmp;
16 }
17
```

```
18 int checkInLine(double x)
19 {
20     double minR = INF, maxL = -INF;
21     double tmpl, tmpr;
22     for(int i = 0; i < n; ++ i) {
23         if (sign(cir[i].x + radius[i] - x) < 0 || sign(cir[i].x - radius[i] - x) > 0)
24             return false;
25         getRange(x, cir[i], radius[i], tmpl, tmpr);
26         maxL = max(tmpl, maxL);
27         minR = min(tmpr, minR);
28         if (maxL > minR) return false;
29     }
30     return true;
31 }
32
33 int shouldGoLeft(double x)
34 {
35     if (checkInLine(x)) return 2;
36     int onL = 0, onR = 0;
37     for(int i = 0; i < n; ++ i) {
38         if (sign(cir[i].x + radius[i] - x) < 0) onL = 1;
39         if (sign(cir[i].x - radius[i] - x) > 0) onR = 1;
40     }
41     if (onL && onR) return -1;
42     if (onL) return 1;
43     if (onR) return 0;
44
45     double minR = INF, maxL = -INF, tmpl, tmpr;
46     int idMinR, idMaxL;
47
48     for(int i = 0; i < n; ++ i) {
49         getRange(x, cir[i], radius[i], tmpl, tmpr);
50         if (tmpr < minR) {
51             minR = tmpr;
52             idMinR = i;
53         }
54         if (tmpl > maxL) {
55             maxL = tmpl;
56             idMaxL = i;
57         }
58     }
59     if (! isIntersectCircleToCircle(cir[idMinR], radius[idMinR], cir[idMaxL], radius[
       idMaxL]))
60         return -1;
61     Point p1, p2;
62     intersectionCircleToCircle(cir[idMinR], radius[idMinR], cir[idMaxL], radius[
       idMaxL], p1, p2);
63     return (p1.x < x);
64 }
```

```
65
66 int hasIntersectionCircles()
67 {
68     double l = -INF, r = INF, mid;
69     for(int i = 0; i < 100; ++ i) {
70         mid = (l + r) * 0.5;
71         int tmp = shouldGoLeft(mid);
72         if (tmp < 0) return 0;
73         if (tmp == 2) return 1;
74         if (tmp) r = mid;
75         else l = mid;
76     }
77     mid = (l + r) * 0.5;
78     return checkInLine(mid);
79 }
```

3.11 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 }
```

3.12 最小覆盖圆

```
 1 #include<cmath>
 2 #include<cstdio>
 3 #include<algorithm>
 4 using namespace std;
 5 const double eps=1e-6;
 6 struct couple
 7 {
 8     double x, y;
 9     couple(){}
10     couple(const double &xx, const double &yy)
11     {
12         x = xx; y = yy;
13     }
14 } a[100001];
15 int n;
16 bool operator < (const couple & a, const couple & b)
17 {
18     return a.x < b.x - eps or (abs(a.x - b.x) < eps and a.y < b.y - eps);
19 }
20 bool operator == (const couple & a, const couple & b)
21 {
22     return !(a < b) and !(b < a);
23 }
24 inline couple operator - (const couple &a, const couple &b)
25 {
26     return couple(a.x-b.x, a.y-b.y);
27 }
28 inline couple operator + (const couple &a, const couple &b)
29 {
30     return couple(a.x+b.x, a.y+b.y);
```

```
31 }
32 inline couple operator * (const couple &a, const double &b)
33 {
34     return couple(a.x*b, a.y*b);
35 }
36 inline couple operator / (const couple &a, const double &b)
37 {
38     return a*(1/b);
39 }
40 inline double operator * (const couple &a, const couple &b)
41 {
42     return a.x*b.y-a.y*b.x;
43 }
44 inline double len(const couple &a)
45 {
46     return a.x*a.x+a.y*a.y;
47 }
48 inline double di2(const couple &a, const couple &b)
49 {
50     return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
51 }
52 inline double dis(const couple &a, const couple &b)
53 {
54     return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
55 }
56 struct circle
57 {
58     double r; couple c;
59 } cir;
60 inline bool inside(const couple & x)
61 {
62     return di2(x, cir.c) < cir.r*cir.r+eps;
63 }
64 inline void p2c(int x, int y)
65 {
66     cir.c.x = (a[x].x+a[y].x)/2;
67     cir.c.y = (a[x].y+a[y].y)/2;
68     cir.r = dis(cir.c, a[x]);
69 }
70 inline void p3c(int i, int j, int k)
71 {
72     couple x = a[i], y = a[j], z = a[k];
73     cir.r = sqrt(di2(x,y)*di2(y,z)*di2(z,x))/fabs(x*y+y*z+z*x)/2;
74     couple t1((x-y).x, (y-z).x), t2((x-y).y, (y-z).y), t3((len(x)-len(y))/2, (len(y)-
        len(z))/2);
75     cir.c = couple(t3*t2, t1*t3)/(t1*t2);
76 }
77 inline circle mi()
78 {
```

```
79      sort(a + 1, a + 1 + n);
80      n = unique(a + 1, a + 1 + n) - a - 1;
81      if(n == 1)
82      {
83          cir.c = a[1];
84          cir.r = 0;
85          return cir;
86      }
87      random_shuffle(a + 1, a + 1 + n);
88      p2c(1, 2);
89      for(int i = 3; i <= n; i++)
90          if(!inside(a[i]))
91          {
92              p2c(1, i);
93              for(int j = 2; j < i; j++)
94                  if(!inside(a[j]))
95                  {
96                      p2c(i, j);
97                      for(int k = 1; k < j; k++)
98                          if(!inside(a[k]))
99                              p3c(i,j, k);
100                 }
101         }
102     return cir;
103 }
```

3.13 圆交 $O(n^2 \log n)$ 计算面积和重心

```
1  double pi = acos(-1.0), eps = 1e-12;
2  double sqr(const double & x) {
3      return x * x;
4  }
5  double ans[2001];
6  int sign(const double & x) {
7      return x < -eps?-1:x > eps;
8  }
9  struct Point {
10     double x, y;
11     Point(){}
12     Point(const double & x, const double & y) : x(x), y(y) {}
13     void scan() {scanf("%lf%lf", &x, &y);}
14     double sqrlen() {return sqr(x) + sqr(y);}
15     double len() {return sqrt(sqrlen());}
16     Point rev() {return Point(y, -x);}
17     void print() {printf("%f␣%f\n", x, y);}
18     Point zoom(const double & d) {double lambda = d / len(); return Point(lambda * x,
           lambda * y);}
19 } dvd, a[2001];
```

```
20 Point centre[2001];
21 double atan2(const Point & x) {
22     return atan2(x.y, x.x);
23 }
24 Point operator − (const Point & a, const Point & b) {
25     return Point(a.x − b.x, a.y − b.y);
26 }
27 Point operator + (const Point & a, const Point & b) {
28     return Point(a.x + b.x, a.y + b.y);
29 }
30 double operator * (const Point & a, const Point & b) {
31     return a.x * b.y − a.y * b.x;
32 }
33 Point operator * (const double & a, const Point & b) {
34     return Point(a * b.x, a * b.y);
35 }
36 double operator % (const Point & a, const Point & b) {
37     return a.x * b.x + a.y * b.y;
38 }
39 struct circle {
40     double r; Point o;
41     circle() {}
42     void scan() {
43         o.scan();
44         scanf("%lf", &r);
45     }
46 } cir[2001];
47 struct arc {
48     double theta;
49     int delta;
50     Point p;
51     arc() {};
52     arc(const double & theta, const Point & p, int d) : theta(theta), p(p), delta(d)
        {}
53 } vec[4444];
54 int nV;
55 inline bool operator < (const arc & a, const arc & b) {
56     return a.theta + eps < b.theta;
57 }
58 int cnt;
59 inline void psh(const double t1, const Point p1, const double t2, const Point p2) {
60     if(t2 + eps < t1)
61         cnt++;
62     vec[nV++] = arc(t1, p1, 1);
63     vec[nV++] = arc(t2, p2, −1);
64 }
65 inline double cub(const double & x) {
66     return x * x * x;
67 }
```

```
68  inline void combine(int d, const double & area, const Point & o) {
69      if(sign(area) == 0) return;
70      centre[d] = 1 / (ans[d] + area) * (ans[d] * centre[d] + area * o);
71      ans[d] += area;
72  }
73  bool equal(const double & x, const double & y) {
74      return x + eps>  y and y + eps > x;
75  }
76  bool equal(const Point & a, const Point & b) {
77      return equal(a.x, b.x) and equal(a.y, b.y);
78  }
79  bool equal(const circle & a, const circle & b) {
80      return equal(a.o, b.o) and equal(a.r, b.r);
81  }
82  bool f[2001];
83  int main() {
84      //freopen("hdu4895.in", "r", stdin);
85      int n, m, index;
86      while(EOF != scanf("%d%d%d", &m, &n, &index)) {
87          index--;
88          for(int i(0); i < m; i++) {
89              a[i].scan();
90          }
91          for(int i(0); i < n; i++) {
92              cir[i].scan();//n个圆
93          }
94          for(int i(0); i < n; i++) {//这一段在去重圆 能加速 删掉不会错
95              f[i] = true;
96              for(int j(0); j < n; j++) if(i != j) {
97                  if(equal(cir[i], cir[j]) and i < j or !equal(cir[i], cir[j]) and cir[
                        i].r < cir[j].r + eps and (cir[i].o - cir[j].o).sqrlen() < sqr(cir
                        [i].r - cir[j].r) + eps) {
98                      f[i] = false;
99                      break;
100                 }
101             }
102         }
103         int n1(0);
104         for(int i(0); i < n; i++)
105             if(f[i])
106                 cir[n1++] = cir[i];
107         n = n1;//去重圆结束
108         fill(ans, ans + n + 1, 0);//ans[i]表示被圆覆盖至少i次的面积
109         fill(centre, centre + n + 1, Point(0, 0));//centre[i]表示上面ans[i]部分的重心
110         for(int i(0); i < m; i++)
111             combine(0, a[i] * a[(i + 1) % m] * 0.5, 1. / 3 * (a[i] + a[(i + 1) % m]))
                    ;
112         for(int i(0); i < n; i++) {
113             dvd = cir[i].o - Point(cir[i].r, 0);
```

```
114             nV = 0;
115             vec[nV++] = arc(-pi, dvd, 1);
116             cnt = 0;
117             for(int j(0); j < n; j++) if(j != i) {
118                 double d = (cir[j].o - cir[i].o).sqrlen();
119                 if(d < sqr(cir[j].r - cir[i].r) + eps) {
120                     if(cir[i].r + i * eps < cir[j].r + j * eps)
121                         psh(-pi, dvd, pi, dvd);
122                 }else if(d + eps < sqr(cir[j].r + cir[i].r)) {
123                     double lambda = 0.5 * (1 + (sqr(cir[i].r) - sqr(cir[j].r)) / d);
124                     Point cp(cir[i].o + lambda * (cir[j].o - cir[i].o));
125                     Point nor((cir[j].o - cir[i].o).rev().zoom(sqrt(sqr(cir[i].r) - (
                            cp - cir[i].o).sqrlen())));
126                     Point frm(cp + nor);
127                     Point to(cp - nor);
128                     psh(atan2(frm - cir[i].o), frm, atan2(to - cir[i].o), to);
129                 }
130             }
131             sort(vec + 1, vec + nV);
132             vec[nV++] = arc(pi, dvd, -1);
133             for(int j = 0; j + 1 < nV; j++) {
134                 cnt += vec[j].delta;
135                 //if(cnt == 1) {//如果只算ans[1]和centre[1], 可以加这个if加速.
136                     double theta(vec[j + 1].theta - vec[j].theta);
137                     double area(sqr(cir[i].r) * theta * 0.5);
138                     combine(cnt, area, cir[i].o + 1. / area / 3 * cub(cir[i].r) *
                            Point(sin(vec[j + 1].theta) - sin(vec[j].theta), cos(vec[j].
                            theta) - cos(vec[j + 1].theta)));
139                     combine(cnt, -sqr(cir[i].r) * sin(theta) * 0.5, 1. / 3 * (cir[i].
                            o + vec[j].p + vec[j + 1].p));
140                     combine(cnt, vec[j].p * vec[j + 1].p * 0.5, 1. / 3 * (vec[j].p +
                            vec[j + 1].p));
141                 //}
142             }
143         }//板子部分结束 下面是题目
144         combine(0, -ans[1], centre[1]);
145         for(int i = 0; i < m; i++) {
146             if(i != index)
147                 (a[index] - Point((a[i] - a[index]) * (centre[0] - a[index]), (a[i] -
                        a[index]) % (centre[0] - a[index])).zoom((a[i] - a[index]).len())
                        ).print();
148             else
149                 a[i].print();
150         }
151
152     }
153     fclose(stdin);
154     return 0;
155 }
```

3.14 三维跨立实验 + 点到线段的垂足在线段上 + 分数类

```
long long gcd(long long a, long long b) {
    return b?gcd(b, a % b):a;
}
struct frac {
    long long x, y;
    frac() {}
    frac(const long long & xx, const long long & yy) : x(xx), y(yy) {
        long long d(gcd(x, y));
        x /= d; y /= d;
        if(y < 0)
            y = -y, x = -x;
    }
    void print() const {
        printf("(%lld/%lld)\n", x, y);
    }
};
frac operator + (const frac & a, const frac & b) {
    //long long y = a.y / gcd(a.y, b.y) * b.y;
    //return frac(y / a.y * a.x + y / b.y * b.x, y);//这里可以减小中间结果，以避免爆long long.
    return frac(a.x * b.y + b.x * a.y, a.y * b.y);
}
frac operator - (const frac & a, const frac & b) {
    //long long y = a.y / gcd(a.y, b.y) * b.y;
    //return frac(y / a.y * a.x - y / b.y * b.x, y);
    return frac(a.x * b.y - b.x * a.y, a.y * b.y);
}
frac operator * (const frac & a, const frac & b) {
    //long long v(gcd(a.x, b.y)), w(gcd(a.y, b.x));
    //return frac((a.x / v) * (b.x / w), (a.y / w) * (b.y / v));
    return frac(a.x * b.x, a.y * b.y);
}
frac operator / (const frac & a, const frac & b) {
    //long long v(gcd(a.x, b.x)), w(gcd(a.y, b.y));
    //return frac((a.x / v) * (b.y / w), (a.y / w) * (b.x / v));
    return frac(a.x * b.y, a.y * b.x);
}
bool operator < (const frac & a, const frac & b) {
    return a.x * b.y < b.x * a.y;
}
bool operator == (const frac & a, const frac & b) {
    return a.x * b.y == b.x * a.y;
}
bool operator <= (const frac & a, const frac & b) {
    return a.x * b.y <= b.x * a.y;
}
```

```
46
47 frac sqr(const frac & a) {
48     return a * a;
49 }
50 struct Point {
51     frac x, y, z;
52     Point () {}
53     void scan() {cin >> x.x >> y.x >> z.x; x.y = y.y = z.y = 1;}
54     Point(const frac & x, const frac & y, const frac & z) :x(x), y(y), z(z) {}
55     frac sqrlen() {return x * x + y * y + z * z;}
56     void print() const {printf("{");x.print(); y.print(); z.print();printf("}\n");}
57 } a, b, c, d;
58 Point operator - (const Point & a, const Point & b) {
59     return Point(a.x - b.x, a.y - b.y, a.z - b.z);
60 }
61 Point operator + (const Point & a, const Point & b) {
62     return Point(a.x + b.x, a.y + b.y, a.z + b.z);
63 }
64 Point operator * (const frac & a, const Point & b) {
65     return Point(a * b.x, a * b.y, a * b.z);
66 }
67 frac operator % (const Point & a, const Point & b) {
68     return a.x * b.x + a.y * b.y + a.z * b.z;
69 }
70 Point operator * (const Point & a, const Point & b) {
71     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)
        ;
72 }
73 bool _ (const Point & a) {
74     return a.x == frac(0, 1) and a.y == frac(0, 1) and a.z == frac(0, 1);
75 }
76 void check(frac & ans, const Point & a, const Point & s, const Point & t) {
77     if(sign((a - s) % (t - s)) * sign((a - t) % (t - s)) <= 0) {//
        点到线段的垂足在线段上(端点含)
78         ans = min(ans, ((a - s) * (t - s)).sqrlen() / (t - s).sqrlen());//
            点到直线距离
79     }
80 }
81 int sign(const frac & a) {
82     return a.x < 0?-1:a.x > 0;
83 }
84 int main() {
85     int tst;
86     scanf("%d", &tst);
87     for(int qq = 1; qq <= tst; qq++) {
88         a.scan(); b.scan();
89         c.scan(); d.scan();//线段(a->b), (c->d)
90         frac ans = (a - c).sqrlen();
91         ans = min(ans, (a - d).sqrlen());
```

```
92          ans = min(ans, (b - c).sqrlen());
93          ans = min(ans, (b - d).sqrlen());
94          Point nor;
95          if(!_(nor = (b - a) * (d - c)))//线段平行
96              if(sign((c - a) * (d - a) % nor) * sign((c - b) * (d - b) % nor) <= 0 and
                    sign((a - c) * (b - c) % nor) * sign((a - d) * (b - d) % nor) <= 0)//
                    三维跨立实验
97                  ans = min(ans, sqr(nor % (c - a)) / nor.sqrlen());
98          check(ans, a, c, d);
99          check(ans, b, c, d);
100         check(ans, c, a, b);
101         check(ans, d, a, b);
102         cout << ans.x << '␣' << ans.y << endl;
103     }
104     return 0;
105 }
```

3.15 平面图形的转动惯量计算

```
1  int n, m;
2  double eps = 1e-8;
3  int sign(const double & x) {
4      return x < -eps?-1:x > eps;
5  }
6  struct Point {
7      double x, y;
8      void scan() {
9          scanf("%lf%lf", &x, &y);
10     }
11     void print() {
12         printf("(%f␣%f)\n", x, y);
13     }
14     Point(const double & x, const double & y) : x(x), y(y) {}
15     Point() {}
16     double len() {return sqrt(x * x + y * y);}
17     Point rev() {return Point(-y, x);}
18 } a[222], b[222];
19 Point operator + (const Point & a, const Point & b) {
20     return Point(a.x + b.x, a.y + b.y);
21 }
22 Point operator - (const Point & a, const Point & b) {
23     return Point(a.x - b.x, a.y - b.y);
24 }
25 Point operator * (const double & a, const Point & b) {
26     return Point(a * b.x, a * b.y);
27 }
28 double operator % (const Point & a, const Point & b) {
```

```
29      return a.x * b.x + a.y * b.y;
30  }
31  double operator * (const Point & a, const Point & b) {
32      return a.x * b.y - a.y * b.x;
33  }
34  double sqr(const double & x) {
35      return x * x;
36  }
37  double cub(const double & x) {
38      return x * x * x;
39  }
40  double calc(const double & Y, const double & c0, const double & c1, const double & c2
        , const double & c3) {
41      return Y * c0 + 0.5 * Y * Y * c1 + Y * Y * Y * c2 / 3 + Y * Y * Y * Y * c3 / 4;
42  }
43  int main() {
44      scanf("%d%d", &n, &m);
45      for(int i = 1; i <= n; i++) {
46          a[i].scan();
47      }
48      a[0] = a[n];
49      double area(0);
50      for(int i = 1; i <= n; i++) {
51          area += (a[i - 1] * a[i]);
52      }
53      for(int i = 1; i <= m; i++) {
54          b[i].scan();
55      }
56      double ans(0);
57      for(int i = 1; i <= m; i++) {
58          vector<Point> vec(a + 1, a + 1 + n);
59          for(int j = 1; j <= m; j++) if(j != i) {
60              vector<Point> vec1;
61              Point mid(0.5 * (b[i] + b[j])), dir((b[j] - b[i]).rev());
62              for(int k = 0; k < (int)vec.size(); k++) {
63                  if(sign((vec[k] - mid) * dir) <= 0)
64                      vec1.push_back(vec[k]);
65                  Point dir1(vec[(k + 1) % (int)vec.size()] - vec[k]);
66                  if(sign((vec[k] - mid) * dir) * sign((vec[(k + 1) % (int)vec.size()]
                        - mid) * dir) < 0) {
67                      double lambda((mid - vec[k]) * dir / (dir1 * dir));
68                      vec1.push_back(vec[k] + lambda * dir1);
69                  }
70              }
71              vec = vec1;
72          }
73          for(int j = 0; j < (int)vec.size(); j++)
74              vec[j] = vec[j] - b[i];
75          for(int j = 0; j < (int)vec.size(); j++){
```

```
76              double X1(vec[j].len()), X(vec[(j + 1) % (int)vec.size()] % vec[j] / vec[
                    j].len()), Y(vec[j] * vec[(j + 1) % (int)vec.size()] / vec[j].len());
77              //若是vec[j].len()为0 或者Y为0 则转动惯量为0
78              //旋转中心在原点 三角形((0, 0), vec[j], vec[j + 1])的转动惯量, 其中若vec[
                    j] * vec[j + 1] < 0求出来的是转动惯量的相反数.
79              ans += calc(Y, cub(X1) / 3, sqr(X1) * (X - X1) / Y, X1 * sqr((X - X1) / Y
                    ), (cub((X - X1) / Y) - cub(X / Y)) / 3);
80              ans += calc(Y, 0, 0, X1, -X1 / Y);
81          }
82
83      }
84
85      printf("%.10f\n", ans / area * 2);
86      fclose(stdin);
87      return 0;
88  }
```

3.16 凸多边形内的最大圆 $O(n \log n)$

```
 1  double eps(1e-8);
 2  int sign(const double & x) {
 3      return x < -eps?-1:x > eps;
 4  }
 5  struct Point {
 6      double x, y;
 7      Point() {
 8      }
 9      Point(const double & x, const double & y) : x(x), y(y) {
10      }
11      double sqrlen() const {
12          return x * x + y * y;
13      }
14      double len() const {
15          return sqrt(sqrlen());
16      }
17      void scan() {
18          scanf("%lf%lf", &x, &y);
19      }
20      void print() const {
21          printf("(%f␣%f)\n", x, y);
22      }
23  };
24  Point operator + (const Point & a, const Point & b) {
25      return Point(a.x + b.x, a.y + b.y);
26  }
27  Point operator - (const Point & a, const Point & b) {
28      return Point(a.x - b.x, a.y - b.y);
```

```
29 }
30 Point operator * (const double & a, const Point & b) {
31     return Point(a * b.x, a * b.y);
32 }
33 double operator * (const Point & a, const Point & b) {
34     return a.x * b.y - a.y * b.x;
35 }
36 struct Line {
37     Point s, d;
38     Line() {
39     }
40     Line(const Point & s, const Point & d) : s(s), d(d) {
41     }
42 };
43 Point crs(const Line & a, const Line & b) {
44     double lambda((b.s - a.s) * b.d / (a.d * b.d));
45     return a.s + lambda * a.d;
46 }
47 struct reca {
48     Point a, b;
49     int prv, nxt;
50     Point d() const {
51         return b - a;
52     }
53     double calc();
54 } a[11111];
55 reca (&c)[11111](a);
56 double reca::calc() {
57     if(sign(d() * c[prv].d()) and sign(d() * c[nxt].d())) {
58         double len1(c[prv].d().len()), len2(d().len()), len3(c[nxt].d().len());
59         Point cp(crs(Line(a, 1 / (len1 + len2) * (len2 * c[prv].a + len1 * b) - a),
               Line(b, 1 / (len2 + len3) * (len3 * a + len2 * c[nxt].b) - b)));
60         return fabs((cp - a) * d() / d().len());
61     }else
62         return 1e100;
63
64 }
65 double val[11111];
66 bool f[11111];
67 int main() {
68     int n;
69     scanf("%d", &n);
70     for(int i(0); i < n; i++) {
71         a[i].a.scan();
72     }
73     for(int i(0); i < n; i++) {
74         a[i].b = a[(i + 1) % n].a;
75         a[i].prv = (i + n - 1) % n;
76         a[i].nxt = (i + 1) % n;
```

```
77      }
78      priority_queue<pair<double, int>, vector<pair<double, int> >, greater<pair<double
            , int> > > hp;
79      for(int i(0); i < n; i++) {
80          hp.push(make_pair(val[i] = a[i].calc(), i));
81      }
82      for(int i(1); i <= n - 3; i++) {
83          int prv(a[hp.top().second].prv), nxt(a[hp.top().second].nxt);
84          a[prv].nxt = nxt;
85          a[nxt].prv = prv;
86          if(sign(a[prv].d() * a[nxt].d()))
87              a[prv].b = a[nxt].a = crs(Line(a[prv].a, a[prv].d()), Line(a[nxt].a, a[
                    nxt].d()));
88          f[hp.top().second] = true;
89          hp.pop();
90          hp.push(make_pair(val[prv] = a[prv].calc(), prv));
91          hp.push(make_pair(val[nxt] = a[nxt].calc(), nxt));
92          while(f[hp.top().second] or val[hp.top().second] != hp.top().first)
93              hp.pop();
94      }
95      int y(hp.top().second);
96      printf("%f\n", min(min(val[a[y].prv], val[a[y].nxt]), val[y]));
97  }
```

3.17 三维凸包

```
 1  const double eps = 1e-8;
 2  int mark[1005][1005];
 3  Point info[1005];
 4  int n, cnt;
 5  double mix(const Point &a, const Point &b, const Point &c) {
 6      return a.dot(b.cross(c));}
 7  double area(int a, int b, int c) {
 8      return ((info[b] - info[a]).cross(info[c] - info[a])).length();}
 9  double volume(int a, int b, int c, int d) {
10      return mix(info[b] - info[a], info[c] - info[a], info[d] - info[a]);}
11  struct Face {
12      int a, b, c;
13      Face() {}
14      Face(int a, int b, int c): a(a), b(b), c(c) {}
15      int &operator [](int k) { return k==0?a:k==1?b:c; }
16  };
17  vector <Face> face;
18  inline void insert(int a, int b, int c) { face.push_back(Face(a, b, c));}
19  void add(int v) {
20      vector <Face> tmp;
21      int a, b, c;
22      cnt++;
```

```
23      for (int i = 0; i < SIZE(face); i++) {
24          a = face[i][0];  b = face[i][1];  c = face[i][2];
25          if (Sign(volume(v, a, b, c)) < 0)
26              mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b] = mark[c][a] =
27                      mark[a][c] = cnt;
28          else tmp.push_back(face[i]);
29      }
30      face = tmp;
31      for (int i = 0; i < SIZE(tmp); i++) {
32          a = face[i][0]; b = face[i][1]; c = face[i][2];
33          if (mark[a][b] == cnt) insert(b, a, v);
34          if (mark[b][c] == cnt) insert(c, b, v);
35          if (mark[c][a] == cnt) insert(a, c, v);
36      }
37  }
38  int Find() {
39      for (int i = 2; i < n; i++) {
40          Point ndir = (info[0] - info[i]).cross(info[1] - info[i]);
41          if (ndir == Point()) continue;
42          swap(info[i], info[2]);
43          for (int j = i + 1; j < n; j++)
44              if (Sign(volume(0, 1, 2, j)) != 0) {
45                  swap(info[j], info[3]);
46                  insert(0, 1, 2);  insert(0, 2, 1);
47                  return 1;
48              }
49      }
50      return 0;
51  }
52  int main() {
53      for (; scanf("%d", &n) == 1; ) {
54          for (int i = 0; i < n; i++)
55              info[i].Input();
56          sort(info, info + n);
57          n = unique(info, info + n) - info;
58          face.clear();
59          random_shuffle(info, info + n);
60          if (Find()) {
61              memset(mark, 0, sizeof(mark));
62              cnt = 0;
63              for (int i = 3; i < n; i++) add(i);
64              vector<Point> Ndir;
65              for (int i = 0; i < SIZE(face); ++i) {
66                  Point p = (info[face[i][0]] - info[face[i][1]]).cross
67                          (info[face[i][2]] - info[face[i][1]]);
68                  p = p / p.length();
69                  Ndir.push_back(p);
70              }
71              sort(Ndir.begin(), Ndir.end());
```

```
72              int ans = unique(Ndir.begin(), Ndir.end()) - Ndir.begin();
73              printf("%d\n", ans);
74          } else {
75              printf("1\n");
76          }
77      }
78  }
```

3.18 点在多边形内

```
 1  bool in_polygon(const point &p, const vector<point> &poly) {
 2      int n = (int)poly.size();
 3      int counter = 0;
 4      for (int i = 0; i < n; ++i) {
 5          point a = poly[i], b = poly[(i + 1) % n];
 6          if (point_on_line(p, line(a, b)))
 7              return false; // bounded excluded
 8          int x = sign(det(p - a, b - a));
 9          int y = sign(a.y - p.y);
10          int z = sign(b.y - p.y);
11          if (x > 0 && y <= 0 && z > 0)
12              counter++;
13          if (x < 0 && z <= 0 && y > 0)
14              counter--;
15      }
16      return counter != 0;
17  }
```

3.19 三角形的内心

```
1  point incenter(const point &a, const point &b, const point &c) {
2      double p = (a - b).length() + (b - c).length() + (c - a).length();
3      return (a * (b - c).length() + b * (c - a).length() + c * (a - b).length()) / p;
4  }
```

3.20 三角形的外心

```
1  point circumcenter(const point &a, const point &b, const point &c) {
2      point p = b - a, q = c - a, s(dot(p, p) / 2, dot(q, q) / 2);
3      double d = det(p, q);
4      return a + point(det(s, point(p.y, q.y)), det(point(p.x, q.x), s)) / d;
5  }
```

3.21 三角形的垂心

```
point orthocenter(const point &a, const point &b, const point &c) {
    return a + b + c - circumcenter(a, b, c) * 2.0;
}
```

3.22 V 图

```
const int AIX = 5;
const int MAXM = AIX * MAXN;

struct point {
    double x, y;
    int index;
    struct Edge *in;
    point(double _x = 0, double _y = 0) : x(_x), y(_y) {}
};
inline bool operator< (const point &a, const point &b) {
    return a.x < b.x || (sgn(a.x - b.x) == 0 && a.y < b.y);
}
inline double cross(const point &a, const point &b, const point &c) { return det
        (b - a, c - a); }
struct Edge {
    point *Org, *Dest;
    Edge *Onext, *Oprev, *Dnext, *Dprev;
};
inline point* Other(const Edge *e, const point *p) { return e->Org == p ?
         e->Dest : e->Org; }
inline Edge* Next(const Edge *e, const point *p) { return e->Org == p ? e->Onext
         : e->Dnext; }
inline Edge* Prev(const Edge *e, const point *p) { return e->Org == p ? e->Oprev
         : e->Dprev; }
struct gEdge {
    int u, v;
    double w;
    gEdge() {}
    gEdge(int _u, int _v, double _w) : u(_u), v(_v), w(_w) {}
};
inline bool operator< (const gEdge &a, const gEdge &b) { return a.w < b.w; }
point p[MAXN], *Q[MAXN];
Edge mem[AIX * MAXN], *elist[AIX * MAXN];
static int nfree;
//Alloc memory
inline void Alloc_Memory(const int &n) {
    nfree = AIX * n;
    Edge *e = mem;
    for (int i = 0; i < nfree; ++i)
```

```
40          elist[i] = e++;
41  }
42  //Add an edge to a ring of edges
43  inline void Splice(Edge *a, Edge *b, point *v) {
44      Edge *next;
45      if (a->Org == v)
46          next = a->Onext, a->Onext = b;
47      else
48          next = a->Dnext, a->Dnext = b;
49      if (next->Org == v)
50          next->Oprev = b;
51      else
52          next->Dprev = b;
53      if (b->Org == v)
54          b->Onext = next, b->Oprev = a;
55      else
56          b->Dnext = next, b->Dprev = a;
57  }
58  //Initialise a new edge
59  inline Edge *MakeEdge(point *u, point *v) {
60      Edge *e = elist[--nfree];
61      e->Onext = e->Oprev = e->Dnext = e->Dprev = e;
62      e->Org = u, e->Dest = v;
63      if (!u->in)
64          u->in = e;
65      if (!v->in)
66          v->in = e;
67      return e;
68  }
69  //Creates a new edge and adds it to two rings of edges.
70  inline Edge *Join(Edge *a, point *u, Edge *b, point *v, int side) {
71      Edge *e = MakeEdge(u, v);
72      if (side == 1) {
73          if (a->Org == u)
74              Splice(a->Oprev, e, u);
75          else
76              Splice(a->Dprev, e, u);
77          Splice(b, e, v);
78      }
79      else {
80          Splice(a, e, u);
81          if (b->Org == v)
82              Splice(b->Oprev, e, v);
83          else
84              Splice(b->Dprev, e, v);
85      }
86      return e;
87  }
88  //Remove an edge
```

```
89  inline void Remove(Edge *e) {
90      point *u = e->Org, *v = e->Dest;
91      if (u->in == e)
92          u->in = e->Onext;
93      if (v->in == e)
94          v->in = e->Dnext;
95      if (e->Onext->Org == u)
96          e->Onext->Oprev = e->Oprev;
97      else
98          e->Onext->Dprev = e->Oprev;
99      if (e->Oprev->Org == u)
100         e->Oprev->Onext = e->Onext;
101     else
102         e->Oprev->Dnext = e->Onext;
103     if (e->Dnext->Org == v)
104         e->Dnext->Oprev = e->Dprev;
105     else
106         e->Dnext->Dprev = e->Dprev;
107     if (e->Dprev->Org == v)
108         e->Dprev->Onext = e->Dnext;
109     else
110         e->Dprev->Dnext = e->Dnext;
111     elist[nfree++] = e;
112 }
113 //Determines the lower tangent of two triangulations
114 inline void Low_tangent(Edge *e_l, point *o_l, Edge *e_r, point *o_r, Edge
115          **l_low, point **OL, Edge **r_low, point **OR) {
116     point *d_l = Other(e_l, o_l), *d_r = Other(e_r, o_r);
117     while (true) {
118         if (cross(*o_l, *o_r, *d_l) < -EPS) {
119             e_l = Prev(e_l, d_l);
120             o_l = d_l;
121             d_l = Other(e_l, o_l);
122         }
123         else if (cross(*o_l, *o_r, *d_r) < -EPS) {
124             e_r = Next(e_r, d_r);
125             o_r = d_r;
126             d_r = Other(e_r, o_r);
127         }
128         else
129             break;
130     }
131     *OL = o_l, *OR = o_r;
132     *l_low = e_l, *r_low = e_r;
133 }
134 inline void Merge(Edge *lr, point *s, Edge *rl, point *u, Edge **tangent) {
135     double cot_L, cot_R, N1, cot_N, P1, cot_P;
136     point l1, l2, r1, r2, uu, vv;
137     point *O, *D, *OR, *OL;
```

```
138     Edge *B, *L, *R;
139     Low_tangent(lr, s, rl, u, &L, &OL, &R, &OR);
140     *tangent = B = Join(L, OL, R, OR, 0);
141     O = OL, D = OR;
142     do {
143         Edge *El = Next(B, O), *Er = Prev(B, D), *next, *prev;
144         point *l = Other(El, O), *r = Other(Er, D);
145         l1 = *O - *l, l2 = *D - *l, r1 = *O - *r, r2 = *D - *r;
146         double cl = det(l1, l2), cr = det(r1, r2);
147         bool BL = cl > EPS, BR = cr > EPS;
148         if (!BL && !BR)
149             break;
150         if (BL) {
151             double dl = dot(l1, l2);
152             cot_L = dl / cl;
153             do {
154                 next = Next(El, O);
155                 uu = *O - *Other(next, O);
156                 vv = *D - *Other(next, O);
157                 N1 = det(uu, vv);
158                 if (!(N1 > EPS))
159                     break;
160                 cot_N = dot(uu, vv) / N1;
161                 if (cot_N > cot_L)
162                     break;
163                 Remove(El);
164                 El = next;
165                 cot_L = cot_N;
166             }
167             while (true);
168         }
169         if (BR) {
170             double dr = dot(r1, r2);
171             cot_R = dr / cr;
172             do {
173                 prev = Prev(Er, D);
174                 uu = *O - *Other(prev, D);
175                 vv = *D - *Other(prev, D);
176                 P1 = det(uu, vv);
177                 if (!(P1 > EPS))
178                     break;
179                 cot_P = dot(uu, vv) / P1;
180                 if (cot_P > cot_R)
181                     break;
182                 Remove(Er);
183                 Er = prev;
184                 cot_R = cot_P;
185             }
186             while (true);
```

```
187         }
188         l = Other(El, O); r = Other(Er, D);
189         if (!BL || (BL && BR && cot_R < cot_L)) {
190             B = Join(B, O, Er, r, 0);
191             D = r;
192         }
193         else {
194             B = Join(El, l, B, D, 0);
195             O = l;
196         }
197     }
198     while (true);
199 }
200 inline void Divide(int s, int t, Edge **L, Edge **R) {
201     Edge *a, *b, *c, *ll, *lr, *rl, *rr, *tangent;
202     int n = t - s + 1;
203     if (n == 2)
204         *L = *R = MakeEdge(Q[s], Q[t]);
205     else if (n == 3) {
206         a = MakeEdge(Q[s], Q[s + 1]);
207         b = MakeEdge(Q[s + 1], Q[t]);
208         Splice(a, b, Q[s + 1]);
209         double v = cross(*Q[s], *Q[s + 1], *Q[t]);
210         if (v > EPS) {
211             c = Join(a, Q[s], b, Q[t], 0);
212             *L = a, *R = b;
213         }
214         else if (v < -EPS) {
215             c = Join(a, Q[s], b, Q[t], 1);
216             *L = c, *R = c;
217         }
218         else
219             *L = a, *R = b;
220     }
221     else if(n > 3) {
222         int split = (s + t) / 2;
223         Divide(s, split, &ll, &lr);
224         Divide(split + 1, t, &rl, &rr);
225         Merge(lr, Q[split], rl, Q[split + 1], &tangent);
226         if (tangent->Org == Q[s])
227             ll = tangent;
228         if (tangent->Dest == Q[t])
229             rr = tangent;
230         *L = ll; *R = rr;
231     }
232 }
233 int task, n, m, k, root[MAXN];
234 gEdge E[MAXM], MST[MAXN];
235 inline int Make_Graph() {
```

```
236     Edge *start, *e;
237     int M = 0;
238     point *u, *v;
239     for(int i = 0; i < n; ++i) {
240         u = p + i;
241         start = e = u->in;
242         do {
243             v = Other(e, u);
244             if (u < v)
245                 E[M++] = gEdge(u - p + 1, v - p + 1, dis(*u, *v));
246             e = Next(e, u);
247         }
248         while(e != start);
249     }
250     return M;
251 }
252 int find_root(const int &x) { return root[x] ? root[x] = find_root(root[x]) : x;
253           }
254 inline bool merge(const int &x, const int &y) {
255     int p = find_root(x), q = find_root(y);
256     if (p != q) {
257         root[p] = q;
258         return true;
259     }
260     else
261         return false;
262 }
263 inline void kruskal(gEdge *E, int m, int n, gEdge* MST) {
264     for (int i = 1; i <= n; ++i)
265         root[i] = 0;
266     sort(E, E + m);
267     int tot = 0;
268     for (int i = 0; i < m; ++i)
269         if (merge(E[i].u, E[i].v))
270             MST[tot++] = E[i];
271 }
272 inline void MinimumEuclideanSpaningTree(point* p, int n, gEdge* MST) {
273     Alloc_Memory(n);
274     sort(p, p + n);
275     for (int i = 0; i < n; ++i)
276         Q[i] = p + i;
277     Edge *L, *R;
278     Divide(0, n - 1, &L, &R);
279     m = Make_Graph();
280     kruskal(E, m, n, MST);
281 }
282 int main() {
283     for (scanf("%d", &task); task--; ) {
284         scanf("%d", &k);
```

```
285         for (n = 0; scanf("%lf", &p[n].x) == 1 && p[n].x != -1; ++n) {
286             scanf("%lf", &p[n].y);
287             p[n].in = NULL;
288             p[n].index = n;
289         }
290         if (n == 1) {
291             printf("0\n");
292             continue;
293         }
294         MinimumEuclideanSpaningTree(p, n, MST);
295         printf("%d\n", int(ceil(k > n ? 0 : MST[n - k - 1].w) + EPS));
296     }
297 }
```

Chapter 4

数据结构

4.1 KD 树

```
1
2  曼哈顿距离版，欧几里得只需要把sqr改成x*x即可。
3  tested on bzoj 2648, 2626
4
5  namespace k_dimensional_tree {
6      int const N = ;
7
8      struct point {
9          int x, y, id;
10     };
11
12     inline long long sqr(const long long &x) {
13         return abs(x);
14     }
15
16     inline long long dist(const point &a, const point &b) {
17         return sqr(a.x - b.x) + sqr(a.y - b.y);
18     }
19
20     struct rectangle {
21         int lx, rx, ly, ry;
22         inline void set(const point &p) {
23             lx = rx = p.x;
24             ly = ry = p.y;
25         }
26         inline void mergy(const point &p) {
27             lx = min(lx, p.x);
28             rx = max(rx, p.x);
29             ly = min(ly, p.y);
30             ry = max(ry, p.y);
31         }
```

```
32          inline void mergy(const rectangle &r) {
33              lx = min(lx, r.lx);
34              rx = max(rx, r.rx);
35              ly = min(ly, r.ly);
36              ry = max(ry, r.ry);
37          }
38          /*  minimum distance    */
39          inline long long dist(const point &p) {
40              if (p.x <= lx && p.y <= ly) {
41                  return sqr(p.x - lx) + sqr(p.y - ly);
42              }
43              if (p.x <= rx && p.y <= ly) {
44                  return sqr(p.y - ly);
45              }
46              if (p.x >= rx && p.y <= ly) {
47                  return sqr(p.x - rx) + sqr(p.y - ly);
48              }
49              if (p.x >= rx && p.y <= ry) {
50                  return sqr(p.x - rx);
51              }
52              if (p.x >= rx && p.y >= ry) {
53                  return sqr(p.x - rx) + sqr(p.y - ry);
54              }
55              if (p.x >= lx && p.y >= ry) {
56                  return sqr(p.y - ry);
57              }
58              if (p.x <= lx && p.y >= ry) {
59                  return sqr(p.x - lx) + sqr(p.y - ry);
60              }
61              if (p.x <= lx && p.y >= ly) {
62                  return sqr(p.x - lx);
63              }
64              return 0;
65          }
66          /*  maximum distance    */
67          inline long long dist(const point &p) {
68              long long ret = 0;
69              ret += max(sqr(rx - p.x), sqr(lx - p.x));
70              ret += max(sqr(ry - p.y), sqr(ly - p.y));
71              return ret;
72          }
73      };
74
75      struct node {
76          int child[2];
77          point p;
78          rectangle r;
79          inline void set(const point &_p) {
80              p = _p;
```

```
81              r.set(p);
82              child[0] = child[1] = 0;
83          }
84      };
85
86      int size;
87      point a[N];
88      node tree[N];
89
90      inline bool xcompare(const point &a, const point &b) {
91          return a.x < b.x || a.x == b.x && a.y < b.y;
92      }
93
94      inline bool ycompare(const point &a, const point &b) {
95          return a.y < b.y || a.y == b.y && a.x < b.x;
96      }
97
98      inline int build(int left, int right, bool dim = 0) {
99          int x = ++size, mid = left + right >> 1;
100         nth_element(a + left, a + mid, a + right, dim ? xcompare : ycompare);
101         tree[x].set(a[mid]);
102         if (left < mid) {
103             tree[x].child[0] = build(left, mid, dim ^ 1);
104             tree[x].r.mergy(tree[tree[x].child[0]].r);
105         }
106         if (mid + 1 < right) {
107             tree[x].child[1] = build(mid + 1, right, dim ^ 1);
108             tree[x].r.mergy(tree[tree[x].child[1]].r);
109         }
110         return x;
111     }
112
113     inline int insert(int x, const point &p, bool dim = 0) {
114         if (x == 0) {
115             tree[++size].set(p);
116             return size;
117         }
118         tree[x].r.mergy(p);
119         if (dim && xcompare(p, tree[x].p) || !dim && ycompare(p, tree[x].p)) {
120             tree[x].child[0] = insert(tree[x].child[0], p, dim ^ 1);
121         } else {
122             tree[x].child[1] = insert(tree[x].child[1], p, dim ^ 1);
123         }
124         return x;
125     }
126
127     /*  query minimum   */
128     inline void query(int x, const point &p, long long &ret, bool dim = 0) {
129         if (tree[x].r.dist(p) >= ret) {
```

```
130             return;
131         }
132         ret = min(ret, dist(tree[x].p, p));
133         if (dim && xcompare(p, tree[x].p) || !dim && ycompare(p, tree[x].p)) {
134             if (tree[x].child[0]) {
135                 query(tree[x].child[0], p, ret, dim ^ 1);
136             }
137             if (tree[x].child[1]) {
138                 query(tree[x].child[1], p, ret, dim ^ 1);
139             }
140         } else {
141             if (tree[x].child[1]) {
142                 query(tree[x].child[1], p, ret, dim ^ 1);
143             }
144             if (tree[x].child[0]) {
145                 query(tree[x].child[0], p, ret, dim ^ 1);
146             }
147         }
148     }
149
150     /*  query maximum   */
151     inline void query(int x, const point &p, long long &ret, bool dim = 0) {
152         if (tree[x].r.dist(p) <= ret) {
153             return;
154         }
155         ret = max(ret, dist(tree[x].p, p));
156         if (dim && xcompare(p, tree[x].p) || !dim && ycompare(p, tree[x].p)) {
157             if (tree[x].child[1]) {
158                 query(tree[x].child[1], p, ret, dim ^ 1);
159             }
160             if (tree[x].child[0]) {
161                 query(tree[x].child[0], p, ret, dim ^ 1);
162             }
163         } else {
164             if (tree[x].child[0]) {
165                 query(tree[x].child[0], p, ret, dim ^ 1);
166             }
167             if (tree[x].child[1]) {
168                 query(tree[x].child[1], p, ret, dim ^ 1);
169             }
170         }
171     }
172
173     /*  query kth-minimum   */
174     inline void query(int x, const point &p, int k, pair<long long, int> ret[], bool
        dim = 0) {
175         if (tree[x].r.dist(p) > ret[k].first) {
176             return;
177         }
```

```
178         pair<long long, int> val = make_pair(dist(tree[x].p, p), tree[x].p.id);
179         for (int i = 1; i <= k; ++i) {
180             if (val < ret[i]) {
181                 for (int j = k + 1; j > i; --j) {
182                     ret[j] = ret[j - 1];
183                 }
184                 ret[i] = val;
185                 break;
186             }
187         }
188         if (dim && xcompare(p, tree[x].p) || !dim && ycompare(p, tree[x].p)) {
189             if (tree[x].child[0]) {
190                 query(tree[x].child[0], p, k, ret, dim ^ 1);
191             }
192             if (tree[x].child[1]) {
193                 query(tree[x].child[1], p, k, ret, dim ^ 1);
194             }
195         } else {
196             if (tree[x].child[1]) {
197                 query(tree[x].child[1], p, k, ret, dim ^ 1);
198             }
199             if (tree[x].child[0]) {
200                 query(tree[x].child[0], p, k, ret, dim ^ 1);
201             }
202         }
203     }
204
205     /*  query kth-maximum   */
206     inline void query(int x, const point &p, int k, pair<long long, int> ret[], bool
        dim = 0) {
207         if (tree[x].r.dist(p) < ret[k].first) {
208             return;
209         }
210         pair<long long, int> val = make_pair(dist(tree[x].p, p), -tree[x].p.id);
211         for (int i = 1; i <= k; ++i) {
212             if (val > ret[i]) {
213                 for (int j = k + 1; j > i; --j) {
214                     ret[j] = ret[j - 1];
215                 }
216                 ret[i] = val;
217                 break;
218             }
219         }
220         if (dim && xcompare(p, tree[x].p) || !dim && ycompare(p, tree[x].p)) {
221             if (tree[x].child[1]) {
222                 query(tree[x].child[1], p, k, ret, dim ^ 1);
223             }
224             if (tree[x].child[0]) {
225                 query(tree[x].child[0], p, k, ret, dim ^ 1);
```

```
226             }
227         } else {
228             if (tree[x].child[0]) {
229                 query(tree[x].child[0], p, k, ret, dim ^ 1);
230             }
231             if (tree[x].child[1]) {
232                 query(tree[x].child[1], p, k, ret, dim ^ 1);
233             }
234         }
235     }
236
237     inline void clear() {
238         size = 0;
239     }
240 }
```

4.2 树链剖分

```
 1 namespace heavy_light_decomposition {
 2     int const N = ;
 3
 4     int n;
 5     vector<int> adj[N];
 6     int father[N], height[N], size[N], son[N], top[N], idx[N], num[N];
 7
 8     inline void prepare() {
 9         vector<int> queue;
10         queue.push_back(1);
11         father[1] = height[1] = 0;
12         for (int head = 0; head < (int)queue.size(); ++head) {
13             int x = queue[head];
14             for (int i = 0; i < (int)adj[x].size(); ++i) {
15                 int y = adj[x][i];
16                 if (y != father[x]) {
17                     queue.push_back(y);
18                     height[y] = height[x] + 1;
19                     father[y] = x;
20                 }
21             }
22         }
23         for (int i = n - 1; i >= 0; --i) {
24             int x = queue[i];
25             size[x] = 1;
26             son[x] = -1;
27             for (int j = 0; j < (int)adj[x].size(); ++j) {
28                 int y = adj[x][j];
29                 if (y != father[x]) {
30                     size[x] += size[y];
```

```
31                      if (son[x] == -1 || size[son[x]] < size[y]) {
32                          son[x] = y;
33                      }
34                  }
35              }
36          }
37          int tot = 0;
38          fill(top + 1, top + n + 1, 0);
39          for (int i = 0; i < n; ++i) {
40              int x = queue[i];
41              if (top[x] == 0) {
42                  for (int y = x; y != -1; y = son[y]) {
43                      top[y] = x;
44                      idx[y] = ++tot;
45                      num[tot] = //data[y];
46                  }
47              }
48          }
49          build(1, 1, n);
50      }
51
52      inline void handle(int x, int y) {
53          for (; true; ) {
54              if (top[x] == top[y]) {
55                  if (x == y) {
56                      handle(1, 1, n, idx[x], idx[x]);
57                  } else {
58                      if (height[x] < height[y]) {
59                          handle(1, 1, n, idx[x], idx[y]);
60                      } else {
61                          handle(1, 1, n, idx[y], idx[x]);
62                      }
63                  }
64                  break;
65              }
66              if (height[top[x]] > height[top[y]]) {
67                  handle(1, 1, n, idx[top[x]], idx[x]);
68                  x = father[top[x]];
69              } else {
70                  handle(1, 1, n, idx[top[y]], idx[y]);
71                  y = father[top[y]];
72              }
73          }
74      }
75  }
```

4.3 可持久化左偏树

```
Node * persiMerge(Node * a, Node * b) {
    if(!a) return b;
    if(!b) return a;
    Node * res;
    if(a->v < b->v) {
        res = new Node(*a);
        res->s[1] = persiMerge(b, res->s[1]);
    }else {
        res = new Node(*b);
        res->s[1] = persiMerge(a, res->s[1]);
    }
    if(!res->s[0] or res->s[1] and res->s[0]->l < res->s[1]->l)
        swap(res->s[0], res->s[1]);
    res->l = res->s[1]?res->s[1]->l + 1:0;
    return res;
}
```

4.4 treap

```
namespace treap {
    struct node {
        node *left, *right;
        int key;
        int size, count, aux;
        inline node(int _aux) {
            left = right = 0;
            key = size = count = 0;
            aux = _aux;
        }
        inline void update() {
            this->size = this->left->size + this->count + this->right->size;
        }
    };

    node *null;

    inline void print(node *&x) {
        if (x == null) {
            return;
        }
        print(x->left);
        printf("%d␣", x->key);
        print(x->right);
    }

    inline node* create(int key) {
        node *x = new node(rand() % INT_MAX);
        x->key = key;
```

```
30          x->count = x->size = 1;
31          x->left = x->right = null;
32          return x;
33      }
34
35      inline void left_rotate(node *&x) {
36          node *y = x->right;
37          x->right = y->left;
38          y->left = x;
39          x->update();
40          y->update();
41          x = y;
42      }
43
44      inline void right_rotate(node *&x) {
45          node *y = x->left;
46          x->left = y->right;
47          y->right = x;
48          x->update();
49          y->update();
50          x = y;
51      }
52
53      inline void insert(node *&x, int key) {
54          if (x == null) {
55              x = create(key);
56              return;
57          }
58          if (x->key == key) {
59              x->count++;
60          } else if (x->key > key) {
61              insert(x->left, key);
62              if (x->left->aux < x->aux) {
63                  right_rotate(x);
64              }
65          } else {
66              insert(x->right, key);
67              if (x->right->aux < x->aux) {
68                  left_rotate(x);
69              }
70          }
71          x->update();
72      }
73
74      inline void erase(node *&x, int key) {
75          if (x == null) {
76              return;
77          }
78          if (x->key == key) {
```

```
79              if (x->count > 1) {
80                  x->count--;
81              } else if (x->left == null && x->right == null) {
82                  delete(x);
83                  x = null;
84                  return;
85              } else if (x->left->aux < x->right->aux) {
86                  right_rotate(x);
87                  erase(x->right, key);
88              } else {
89                  left_rotate(x);
90                  erase(x->left, key);
91              }
92          } else if (x->key > key) {
93              erase(x->left, key);
94          } else {
95              erase(x->right, key);
96          }
97          x->update();
98      }
99
100     inline void prepare() {
101         null = new node(INT_MAX);
102     }
103 }
```

4.5 functional_treap

```
1  namespace functional_treap {
2      struct node {
3          int size;
4          node *left, *right;
5          inline node(node *_left, node *_right) {
6              left = _left;
7              right = _right;
8          }
9          inline node* update() {
10             size = left->size + 1 + right->size;
11             return this;
12         }
13         inline pair<node*, node*> split(int);
14     };
15
16     node* null;
17
18     inline bool random(int x, int y) {
19         return rand() % (x + y) < x;
20     }
```

```
21
22      inline node* mergy(node* x, node* y) {
23          if (x == null) {
24              return y;
25          }
26          if (y == null) {
27              return x;
28          }
29          if (random(x->size, y->size)) {
30              x->right = mergy(x->right, y);
31              return x->update();
32          }
33          y->left = mergy(x, y->left);
34          return y->update();
35      }
36
37      inline pair<node*, node*> node::split(int n) {
38          if (this == null) {
39              return make_pair(null, null);
40          }
41          if (n <= left->size) {
42              pair<node*, node*> ret = left->split(n);
43              left = null;
44              return make_pair(ret.first, mergy(ret.second, this->update()));
45          }
46          pair<node*, node*> ret = right->split(n - left->size);
47          right = null;
48          return make_pair(mergy(this->update(), ret.first), ret.second);
49      }
50
51      inline void prepare() {
52          null = new node(null, null);
53          null->left = null->right = null;
54      }
55  }
```

4.6 LCT

```
 1  namespace link_cut_tree {
 2      struct node {
 3          node *child[2], *father;
 4          bool head, rev;
 5          int val, sum, size;
 6          inline node() {
 7              head = rev = val = sum = size = 0;
 8          }
 9          inline void set(node *temp, int dir) {
10              child[dir] = temp;
```

```
11              temp->father = this;
12          }
13          inline int which() {
14              return father->child[1] == this;
15          }
16          inline void update() {
17              sum = val + child[0]->sum + child[1]->sum;
18              size = 1 + child[0]->size + child[1]->size;
19          }
20          inline void release() {
21              if (rev) {
22                  child[0]->reverse();
23                  child[1]->reverse();
24                  rev = 0;
25              }
26          }
27          inline void reverse() {
28              if (size == 0) {
29                  return;
30              }
31              rev ^= 1;
32              swap(child[0], child[1]);
33          }
34      };
35
36      node *null, *tree[N];
37
38      inline node* create(int val) {
39          node *temp = new node();
40          temp->val = temp->sum = val;
41          temp->size = 1;
42          temp->child[0] = temp->child[1] = temp->father = null;
43          temp->head = true;
44          return temp;
45      }
46
47      inline void rotate(node *root) {
48          node *father = root->father;
49          father->release();
50          root->release();
51          int dir = root->which();
52          father->set(root->child[!dir], dir);
53          if (father->head) {
54              father->head = false;
55              root->head = true;
56              root->father = father->father;
57          } else {
58              father->father->set(root, father->which());
59          }
```

```
60          root->set(father, !dir);
61          father->update();
62      }
63
64      inline void splay(node *root) {
65          for (root->release(); !root->head; ) {
66              if (root->father->head) {
67                  rotate(root);
68              } else {
69                  root->which() == root->father->which() ? (rotate(root->father),
                        rotate(root)) : (rotate(root), rotate(root));
70              }
71          }
72          root->update();
73      }
74
75      inline void access(node *root) {
76          for (node *temp = null; root != null; temp = root, root = root->father) {
77              splay(root);
78              root->child[1]->head = true;
79              root->child[1] = temp;
80              root->child[1]->head = false;
81              root->update();
82          }
83      }
84
85      inline void link(int son, int father) {
86          access(tree[son]);
87          splay(tree[son]);
88          tree[son]->father = tree[father];
89          tree[son]->reverse();
90      }
91
92      inline void cut(int x, int y) {
93          access(tree[y]);
94          splay(tree[x]);
95          if (tree[x]->father == tree[y]) {
96              tree[x]->father = null;
97          } else {
98              access(tree[x]);
99              splay(tree[y]);
100             if (tree[y]->father == tree[x]) {
101                 tree[y]->father = null;
102             }
103         }
104     }
105
106     inline void handle(int x, int y) {
107         access(tree[x]);
```

```
108         node *root = tree[y];
109         for (node *temp = null; root != null; temp = root, root = root->father) {
110             splay(root);
111             if (root->father == null) {
112 
113             }
114             root->child[1]->head = true;
115             root->child[1] = temp;
116             root->child[1]->head = false;
117             root->update();
118         }
119     }
120 
121     inline void init(int n, int val[]) {
122         for (int i = 1; i <= n; ++i) {
123             tree[i] = create(val[i]);
124         }
125     }
126 
127     inline void prepare() {
128         null = new node();
129         null->child[0] = null->child[1] = null->father = null;
130     }
131 }
```

4.7 Splay

```
 1 namespace splay {
 2     struct node {
 3         node *child[2], *father;
 4         int val, sum, size;
 5         inline node() {
 6             val = sum = size = 0;
 7         }
 8         inline int which() {
 9             return father->child[1] == this;
10         }
11         inline void set(node *temp, int dir) {
12             child[dir] = temp;
13             temp->father = this;
14         }
15         inline void update() {
16             sum = val + child[0]->sum + child[1]->sum;
17             size = 1 + child[0]->size + child[1]->size;
18         }
19         inline void release() {
20 
21         }
```

```
22      };
23
24      node *null, *head;
25
26      inline void print(node *root) {
27          if (root == null) {
28              return;
29          }
30          print(root->child[0]);
31          printf("%d␣", root->val);
32          print(root->child[1]);
33      }
34
35      inline node* create(int val = 0) {
36          node *temp = new node();
37          temp->val = val;
38          temp->child[0] = temp->child[1] = temp->father = null;
39          return temp;
40      }
41
42      inline void rotate(node *root) {
43          node *father = root->father;
44          int dir = root->which();
45          father->release();
46          root->release;
47          father->set(root->child[!dir], dir);
48          father->father->set(root, father->which());
49          root->set(father, !dir);
50          if (father == head) {
51              head = root;
52          }
53          father->update();
54      }
55
56      inline void splay(node *root, node *target) {
57          for (root->release(); root->father != target; ) {
58              if (root->father->father == target) {
59                  rotate(root);
60              } else {
61                  root->which() == root->father->which() ? (rotate(root->father),
                        rotate(root)) : (rotate(root), rotate(root));
62              }
63          }
64          root->update();
65      }
66
67      inline int rank(node *root) {
68          splay(root, null);
69          return root->child[0]->size + 1;
```

```
70      }
71
72      inline node* find(int rank) {
73          node *now = head;
74          for (; now->child[0]->size + 1 != rank; ) {
75              now->release();
76              if (now->child[0]->size + 1 > rank) {
77                  now = now->child[0];
78              } else {
79                  rank -= now->child[0]->size + 1;
80                  now = now->child[1];
81              }
82          }
83          return now;
84      }
85
86      inline void splay(int left, int right) {
87          splay(find(right), null);
88          splay(find(left), head);
89      }
90
91      inline node* insert(int pos, int val) {
92          splay(pos, pos + 1);
93          node *now = head->child[0];
94          node *cur = create(val);
95          now->set(cur, 1);
96          splay(cur, null);
97          return head;
98      }
99
100     inline void insert(int pos, int n, int val[]) {
101         splay(pos, pos + 1);
102         node *now = head->child[0];
103         for (int i = 1; i <= n; ++i) {
104             node *cur = create(val[i]);
105             now->set(cur, 1);
106             now = cur;
107         }
108         splay(now, null);
109     }
110
111     inline void erase(node *root) {
112         int pos = rank(root);
113         splay(pos - 1, pos + 1);
114         head->child[0]->child[1] = null;
115         head->child[0]->update();
116         head->update();
117     }
118
```

```
119     inline int query(int left, int right) {
120         splay(left - 1, right + 1);
121         return head->child[0]->child[1]->sum;
122     }
123 
124     inline void prepare() {
125         null = new node();
126         head = create();
127         node *tail = create();
128         head->set(tail, 1);
129         splay(tail, null);
130     }
131 }
```

Chapter 5

图论

5.1 Gabow 算法求点双连通分量 (非递归)

边 (u, v) 属于 min(color[u], color[v]) 这个点双连通分量.

```
 1  int color[222222], siz[222222], cnt[222222];
 2  long long ans[222222];
 3  vector<int> edges[222222];
 4  vector<pair<int, int> > st0, st2;
 5  vector<int> st1;
 6  void psh(int v) {
 7      st0.push_back(make_pair(v, 0));
 8      color[v] = st1.size();
 9      st1.push_back(v);
10  }
11  int main() {
12      freopen("travel.in", "r", stdin);
13      freopen("travel.out", "w", stdout);
14      int n, m;
15      scanf("%d%d", &n, &m);
16      for(int i(1); i <= m; i++) {
17          int x, y;
18          scanf("%d%d", &x, &y);
19          edges[x].push_back(y);
20          edges[y].push_back(x);
21      }
22      int c(n);
23      fill(color + 1, color + 1 + n, 0);
24      fill(ans + 1, ans + 1 + n, 0);
25      fill(cnt + 1, cnt + 1 + n, 0);
26      fill(siz + 1, siz + 1 + n, 0);
27      for(int i(1); i <= n; i++) if(!color[i]) {
28          psh(i);
29          while(!st0.empty()) {
30
```

```
31              int v(st0.back().first), p(st0.back().second++);
32              if(p != (int)edges[v].size()) {
33                  int y(edges[v][p]);
34                  if(!color[y]) {
35                      psh(y);
36                      st2.push_back(make_pair(color[v], color[y]));
37                  }else
38                      while(!st2.empty() and st2.back().first > color[y])
39                          st2.pop_back();
40              }else {
41                  st0.pop_back();
42                  siz[v]++;
43                  if(color[v] == 1)
44                      color[v] = c;
45                  else {
46                      int fa(st0.back().first);
47                      if(st2.back().second == color[v]) {
48                          st2.pop_back();
49                          color[v] = ++c;
50                          while(st1.back() != v) {
51                              color[st1.back()] = c;
52                              st1.pop_back();
53                          }
54                          st1.pop_back();
55                          ans[fa] += (long long)cnt[fa] * siz[v];
56                          cnt[fa] += siz[v];
57                      }
58                      siz[fa] += siz[v];
59                  }
60                  ans[v] += (long long)(n - cnt[v]) * cnt[v] + n - cnt[v] - 1;
61              }
62          }
63      }
64      for(int i(1); i <= n; i++) {
65          cout << ans[i] << endl;//ans[i]:删去点i后, 无法连通的{a, b}数, 其中a, b
                为图中不同节点且无序.
66      }
67      fclose(stdin);
68      fclose(stdout);
69      return 0;
70  }
```

5.2 Hopcroft Karp 求二分图最大匹配 $O(EV^{0.5})$

```
1  // hint :: 全部都是0base
2  // 用的时候, 建好边, 左边n个点, 右边m个点, 直接调用maxMatch即可
3
```

```
4  const int N = 3333;
5
6  vector<int> e[N];
7  int pairx[N], pairy[N], level[N];
8  int n, m;
9
10 bool dfs(int x) {
11     for(int i = 0; i < (int)e[x].size(); i++) {
12         int y = e[x][i];
13         int w = pairy[y];
14         if (w == -1 || level[x] + 1 == level[w] && dfs(w)) {
15             pairx[x] = y;
16             pairy[y] = x;
17             return true;
18         }
19     }
20     level[x] = -1;
21     return false;
22 }
23
24 int maxMatch() {
25     fill(pairx, pairx + n, -1);
26     fill(pairy, pairy + m, -1);
27
28     for(int answer = 0; ; ) {
29         vector<int> queue;
30         for(int i = 0; i < n; i++) {
31             if (pairx[i] == -1) {
32                 level[i] = 0;
33                 queue.push_back(i);
34             } else {
35                 level[i] = -1;
36             }
37         }
38
39         for(int head = 0; head < (int)queue.size(); head++) {
40             int x = queue[head];
41             for(int i = 0; i < (int)e[x].size(); i++) {
42                 int y = e[x][i];
43                 int w = pairy[y];
44                 if (w != -1 && level[w] < 0) {
45                     level[w] = level[x] + 1;
46                     queue.push_back(w);
47                 }
48             }
49         }
50
51         int delta = 0;
52         for(int i = 0; i < n; i++) {
```

```
53              if (pairx[i] == -1 && dfs(i)) {
54                  delta++;
55              }
56          }
57          if (delta == 0) {
58              return answer;
59          } else {
60              answer += delta;
61          }
62      }
63  }
64
65  int solve() {
66      int timing;
67      scanf("%d", &timing);
68
69      static int x[N], y[N], s[N];
70      scanf("%d", &n);
71      for(int i = 0; i < n; i++) {
72          scanf("%d␣%d␣%d", &x[i], &y[i], &s[i]);
73          e[i].clear();
74      }
75
76      scanf("%d", &m);
77      for(int i = 0; i < m; i++) {
78          int xx, yy;
79          scanf("%d␣%d", &xx, &yy);
80          for(int j = 0; j < n; j++) {
81              if (timing * timing * s[j] * s[j] >= (xx - x[j]) * (xx - x[j]) + (yy - y[
                    j]) * (yy - y[j])) {
82                  e[j].push_back(i);
83              }
84          }
85      }
86
87      return maxMatch();
88  }
89
90  int main() {
91      freopen("input.txt", "r", stdin);
92      int test;
93      scanf("%d", &test);
94      while(test--) {
95          static int testCount = 0;
96          printf("Scenario␣#%d:\n", ++testCount);
97          printf("%d\n", solve());
98          puts("");
99      }
100     return 0;
```

```
101 }
```

5.3 最小树形图

```
  1 const int maxn=1100;
  2
  3 int n,m , g[maxn][maxn] , used[maxn] , pass[maxn] , eg[maxn] , more , queue[maxn];
  4
  5 void combine (int id , int &sum ) {
  6     int tot = 0 , from , i , j , k ;
  7     for ( ; id!=0 && !pass[ id ] ; id=eg[id] ) {
  8         queue[tot++]=id ; pass[id]=1;
  9     }
 10     for ( from=0; from<tot && queue[from]!=id ; from++);
 11     if  ( from==tot ) return ;
 12     more = 1 ;
 13     for ( i=from ; i<tot ; i++) {
 14         sum+=g[eg[queue[i]]][queue[i]] ;
 15         if ( i!=from ) {
 16             used[queue[i]]=1;
 17             for ( j = 1 ; j <= n ; j++) if ( !used[j] )
 18                 if ( g[queue[i]][j]<g[id][j] ) g[id][j]=g[queue[i]][j] ;
 19         }
 20     }
 21     for ( i=1; i<=n ; i++) if ( !used[i] && i!=id ) {
 22         for ( j=from ; j<tot ; j++){
 23             k=queue[j];
 24             if ( g[i][id]>g[i][k]-g[eg[k]][k] ) g[i][id]=g[i][k]-g[eg[k]][k];
 25         }
 26     }
 27 }
 28
 29 int mdst( int root ) { // return the total length of MDST
 30     int i , j , k , sum = 0 ;
 31     memset ( used , 0 , sizeof ( used ) ) ;
 32     for ( more =1; more ; ) {
 33         more = 0 ;
 34         memset (eg,0,sizeof(eg)) ;
 35         for ( i=1 ; i <= n ; i ++) if ( !used[i] && i!=root ) {
 36             for ( j=1 , k=0 ; j <= n ; j ++) if ( !used[j] && i!=j )
 37                 if ( k==0 || g[j][i] < g[k][i] ) k=j ;
 38             eg[i] = k ;
 39         }
 40         memset(pass,0,sizeof(pass));
 41         for ( i=1; i<=n ; i++) if ( !used[i] && !pass[i] && i!= root ) combine ( i ,
                sum ) ;
 42     }
 43     for ( i =1; i<=n ; i ++) if ( !used[i] && i!= root ) sum+=g[eg[i]][i];
```

```
44      return sum ;
45  }
46
47
48  int main(){
49     freopen("input.txt","r",stdin);
50     freopen("output.txt","w",stdout);
51     int i,j,k,test,cases;
52     cases=0;
53     scanf("%d",&test);
54     while (test){
55          test--;
56          //if (n==0) break;
57          scanf("%d%d",&n,&m);
58  //      memset(g,60,sizeof(g));
59          foru(i,1,n)
60            foru(j,1,n) g[i][j]=1000001;
61          foru(i,1,m) {
62              scanf("%d%d",&j,&k);
63              j++;k++;
64              scanf("%d",&g[j][k]);
65          }
66          cases++;
67          printf("Case #%d: ",cases);
68          k=mdst(1);
69          if (k>1000000) printf("Possums!\n");  //===no
70          else printf("%d\n",k);
71      }
72
73     return 0;
74  }
```

5.4 KM

```
1   #include <cstdio>
2   #include <cstdlib>
3   #include <algorithm>
4   #include <vector>
5   #include <cstring>
6   #include <string>
7   #include <iostream>
8
9   #define foreach(e, x) for(__typeof(x.begin()) e = x.begin(); e != x.end(); ++e)
10
11  using namespace std;
12
13  const int N = 333;
14  const int INF = (1 << 30);
```

```
15
16 int mat[N][N], lx[N], ly[N], vx[N], vy[N], slack[N];
17 int n, match[N];
18
19 bool find(int x) {
20     vx[x] = 1;
21     for(int i = 1; i <= n; i++) {
22         if (vy[i]) {
23             continue;
24         }
25         int temp = lx[x] + ly[i] - mat[x][i];
26         if (temp == 0) {
27             vy[i] = 1;
28             if (match[i] == -1 || find(match[i])) {
29                 match[i] = x;
30                 return true;
31             }
32         } else {
33             slack[i] = min(slack[i], temp);
34         }
35     }
36     return false;
37 }
38
39 int KM() {
40     for(int i = 1; i <= n; i++) {
41         lx[i] = -INF;
42         ly[i] = 0;
43         match[i] = -1;
44         for(int j = 1; j <= n; j++) {
45             lx[i] = max(lx[i], mat[i][j]);
46         }
47     }
48     for(int i = 1; i <= n; i++) {
49         for(int j = 1; j <= n; j++) {
50             slack[j] = INF;
51         }
52         for(; ;) {
53             memset(vx, 0, sizeof(vx));
54             memset(vy, 0, sizeof(vy));
55             for(int j = 1; j <= n; j++) {
56                 slack[j] = INF;
57             }
58             if (find(i)) {
59                 break;
60             }
61             int delta = INF;
62             for(int j = 1; j <= n; j++) {
63                 if (!vy[j]) {
```

```
64                      delta = min(delta, slack[j]);
65                  }
66              }
67              for(int j = 1; j <= n; j++) {
68                  if (vx[j]) {
69                      lx[j] -= delta;
70                  }
71                  if (vy[j]) {
72                      ly[j] += delta;
73                  } else {
74                      slack[j] -= delta;
75                  }
76              }
77          }
78      }
79      int answer = 0;
80      for(int i = 1; i <= n; i++) {
81          answer += mat[match[i]][i];
82      }
83      return answer;
84  }
85
86  int main() {
87      while(scanf("%d", &n) != EOF) {
88          for(int i = 1; i <= n; i++) {
89              for(int j = 1; j <= n; j++) {
90                  scanf("%d", &mat[i][j]);
91              }
92          }
93          printf("%d\n", KM());
94      }
95      return 0;
96  }
```

5.5 扩展 KM

```
1   #include <cstdio>
2   #include <cstdlib>
3   #include <algorithm>
4   #include <iostream>
5   #include <cstring>
6   using namespace std;
7
8   const int N = 205;
9   const int inf = 100000000;
10
11  int a[N], b[N], c[N][N], vx[N], vy[N], w[N][N], dx[N], dy[N];
12  int ans, m, n, slack[N], lk[N], next[N];
```

```
13
14  bool hungary(int x) {
15      vx[x] = 1;
16      for(int i = 1; i <= n; i++) {
17          if (vy[i])
18              continue;
19          int delta = dx[x] + dy[i] - w[x][i];
20          if (delta == 0) {
21              vy[i] = 1;
22              if (b[i]) {
23                  lk[x] = i;
24                  next[x] = 0;
25                  return true;
26              }
27              for(int j = 1; j <= m; j++) {
28                  if (vx[j])
29                      continue;
30                  if (c[j][i] && hungary(j)) {
31                      lk[x] = i;
32                      next[x] = j;
33                      return true;
34                  }
35              }
36          } else {
37              slack[i] = min(slack[i], delta);
38          }
39      }
40      return false;
41  }
42
43  void travel(int x) {
44      int flow = a[x];
45      for(int i = x; i; i = next[i]) {
46          if (next[i])
47              flow = min(flow, c[next[i]][lk[i]]);
48          else
49              flow = min(flow, b[lk[i]]);
50      }
51      a[x] -= flow;
52      for(int i = x; i; i = next[i]) {
53          if (next[i])
54              c[next[i]][lk[i]] -= flow;
55          else
56              b[lk[i]] -= flow;
57          c[i][lk[i]] += flow;
58      }
59  }
60
61  int Main() {
```

```
62      scanf("%d␣%d", &m, &n);
63      for(int i = 1; i <= m; i++)
64          scanf("%d", &a[i]);
65      for(int i = 1; i <= n; i++)
66          scanf("%d", &b[i]);
67      for(int i = 1; i <= m; i++)
68          for(int j = 1; j <= n; j++) {
69              scanf("%d", &w[i][j]);
70              w[i][j] *= -1;
71              c[i][j] = 0;
72      }
73      memset(dy, 0, sizeof(dy));
74      for(int i = 1; i <= m; i++) {
75          dx[i] = -inf;
76          for(int j = 1; j <= n; j++)
77              dx[i] = max(dx[i], w[i][j]);
78      }
79      for(int i = 1; i <= m; i++) {
80          while(1) {
81              for(int j = 1; j <= n; j++)
82                  slack[j] = inf;
83              while (a[i]) {
84                  fill(vx + 1, vx + m + 1, 0);
85                  fill(vy + 1, vy + n + 1, 0);
86                  if (hungary(i))
87                      travel(i);
88                  else
89                      break;
90              }
91              if (!a[i])
92                  break;
93              int delta = inf;
94              for(int j = 1; j <= n; j++)
95                  if (!vy[j])
96                      delta = min(delta, slack[j]);
97              for(int j = 1; j <= m; j++)
98                  if (vx[j])
99                      dx[j] -= delta;
100             for(int j = 1; j <= n; j++)
101                 if (vy[j])
102                     dy[j] += delta;
103         }
104     }
105     long long ans = 0;
106     for(int i = 1; i <= m; i++)
107         for(int j = 1; j <= n; j++) {
108             ans += (long long)c[i][j] * w[i][j];
109         }
110     cout << -ans << endl;
```

```
111     return 0;
112 }
113
114 int main() {
115     int testCount;
116     scanf("%d", &testCount);
117     while(testCount--) {
118         Main();
119     }
120     return 0;
121 }
```

5.6 度限制生成树

```
 1 const int N = 55, M = 1010, INF = 1e8;
 2 int n, m, S, K, ans, cnt, Best[N], fa[N], FE[N];
 3 int f[N], p[M], t[M], c[M], o, Cost[N];
 4 bool u[M], d[M];
 5 pair<int, int> MinCost[N];
 6 struct Edge {
 7     int a, b, c;
 8     bool operator < (const Edge & E) const { return c < E.c; }
 9 }E[M];
10 vector<int> SE;
11 inline int F(int x) { return fa[x] == x ? x : fa[x] = F(fa[x]); }
12 inline void AddEdge(int a, int b, int C) {
13     p[++o] = b; c[o] = C;
14     t[o] = f[a]; f[a] = o;
15 }
16 void dfs(int i, int father) {
17     fa[i] = father;
18     if (father == S) Best[i] = -1;
19     else {
20         Best[i] = i;
21         if (Cost[Best[father]] > Cost[i]) Best[i] = Best[father];
22     }
23     for (int j = f[i]; j; j = t[j])
24     if (!d[j] && p[j] != father) {
25         Cost[p[j]] = c[j];
26         FE[p[j]] = j;
27         dfs(p[j], i);
28     }
29 }
30 inline void Kruskal() {
31     cnt = n - 1; ans = 0; o = 1;
32     for (int i = 1; i <= n; i++) fa[i] = i, f[i] = 0;
33     sort(E + 1, E + m + 1);
34     for (int i = 1; i <= m; i++) {
```

```
35          if (E[i].b == S) swap(E[i].a, E[i].b);
36          if (E[i].a != S && F(E[i].a) != F(E[i].b)) {
37              fa[F(E[i].a)] = F(E[i].b);
38              ans += E[i].c;
39              cnt--;
40              u[i] = true;
41              AddEdge(E[i].a, E[i].b, E[i].c);
42              AddEdge(E[i].b, E[i].a, E[i].c);
43          }
44      }
45      for (int i = 1; i <= n; i++) MinCost[i] = make_pair(INF, INF);
46      for (int i = 1; i <= m; i++)
47      if (E[i].a == S) {
48          SE.push_back(i);
49          MinCost[F(E[i].b)] = min(MinCost[F(E[i].b)], make_pair(E[i].c, i));
50      }
51      for (int i = 1; i <= n; i++)
52      if (i != S && fa[i] == i) {
53          dfs(E[MinCost[i].second].b, S);
54          u[MinCost[i].second] = true;
55          ans += MinCost[i].first;
56      }
57  }
58  bool Solve() {
59      Kruskal();
60      for (int i = cnt + 1; i <= K && i <= n; i++) {
61          int MinD = INF, MinID = -1;
62          for (int j = (int) SE.size() - 1; j >= 0; j--)
63          if (u[SE[j]])
64              SE.erase(SE.begin() + j);
65          for (int j = 0; j < (int) SE.size(); j++) {
66              int tmp = E[SE[j]].c - Cost[Best[E[SE[j]].b]];
67              if (tmp < MinD) {
68                  MinD = tmp;
69                  MinID= SE[j];
70              }
71          }
72          if (MinID == -1) return false;
73          if (MinD >= 0) break;
74          ans += MinD;
75          u[MinID] = true;
76          d[FE[Best[E[MinID].b]]] = d[FE[Best[E[MinID].b]] ^ 1] = true;
77          dfs(E[MinID].b, S);
78      }
79      return true;
80  }
```

5.7 一般图匹配

```
const int N = 300;
int n, Next[N], f[N], mark[N], visited[N], Link[N], Q[N], head, tail;
vector <int> E[N];
int getf(int x) { return f[x] == x ? x : f[x] = getf(f[x]); }
void merge(int x, int y) { x = getf(x); y = getf(y); if (x != y) f[x] = y; }
int LCA(int x, int y) {
    static int flag = 0;
    flag++;
    for (; ; swap(x, y)) if (x != -1) {
        x = getf(x);
        if (visited[x] == flag) return x;
        visited[x] = flag;
        if (Link[x] != -1) x = Next[Link[x]];
        else x = -1;
    }
}
void go(int a, int p) {
    while (a != p) {
        int b = Link[a], c = Next[b];
        if (getf(c) != p) Next[c] = b;
        if (mark[b] == 2) mark[Q[tail++] = b] = 1;
        if (mark[c] == 2) mark[Q[tail++] = c] = 1;
        merge(a, b); merge(b, c); a = c;
    }
}
void find(int s) {
    for (int i = 0; i < n; i++) {
        Next[i] = -1; f[i] = i;
        mark[i] = 0; visited[i] = -1;
    }
    head = tail = 0; Q[tail++] = s; mark[s] = 1;
    for (; head < tail && Link[s] == -1; ) {
        for (int i = 0, x = Q[head++]; i < (int)E[x].size(); i++) {
            if (Link[x] != E[x][i] && getf(x) != getf(E[x][i]) && mark[E[x][i]] != 2)
                {
                int y = E[x][i];
                if (mark[y] == 1) {
                    int p = LCA(x, y);
                    if (getf(x) != p) Next[x] = y;
                    if (getf(y) != p) Next[y] = x;
                    go(x, p);
                    go(y, p);
                }
                else if (Link[y] == -1) {
                    Next[y] = x;
                    for (int j = y; j != -1; ) {
                        int k = Next[j];
```

```
47                          int tmp = Link[k];
48                          Link[j] = k;
49                          Link[k] = j;
50                          j = tmp;
51                      }
52                      break;
53                  }
54                  else {
55                      Next[y] = x;
56                      mark[Q[tail++] = Link[y]] = 1;
57                      mark[y] = 2;
58                  }
59              }
60          }
61      }
62  }
63  int main() {
64      for (int i = 0; i < n; i++) Link[i] = -1;
65      for (int i = 0; i < n; i++) if (Link[i] == -1) {
66          find(i);
67      }
68      int ans = 0;
69      for (int i = 0; i < n; i++) ans += Link[i] != -1;
70      return ans;
71  }
```

5.8 无向图最小割

```
1   const int V = 100;
2   #define typec int
3   const typec inf = 0x3f3f3f; // max of res
4   const typec maxw = 1000; // maximum edge weight
5   typec g[V][V], w[V]; //g[i][j] = g[j][i]
6   int a[V], v[V], na[V];
7   typec mincut(int n) {
8       int i, j, pv, zj;
9       typec best = maxw * n * n;
10      for (i = 0; i < n; i++) v[i] = i; // vertex: 0 ~ n-1
11      while (n > 1) {
12          for (a[v[0]] = 1, i = 1; i < n; i++) {
13              a[v[i]] = 0; na[i - 1] = i;
14              w[i] = g[v[0]][v[i]];
15          }
16          for (pv = v[0], i = 1; i < n; i++) {
17              for (zj = -1, j = 1; j < n; j++)
18                  if (!a[v[j]] && (zj < 0 || w[j] > w[zj]))
19                      zj = j;
20              a[v[zj]] = 1;
```

```
21              if (i == n - 1) {
22                  if (best > w[zj]) best = w[zj];
23                  for (i = 0; i < n; i++)
24                      g[v[i]][pv] = g[pv][v[i]] +=
25                          g[v[zj]][v[i]];
26                  v[zj] = v[--n];
27                  break;
28              }
29              pv = v[zj];
30              for (j = 1; j < n; j++)
31                  if(!a[v[j]])
32                      w[j] += g[v[zj]][v[j]];
33          }
34      }
35      return best;
36  }
```

5.9 Hamilton 回路

```
 1  bool graph[N][N];
 2  int n, l[N], r[N], next[N], last[N], s, t;
 3  char buf[10010];
 4  void cover(int x) { l[r[x]] = l[x]; r[l[x]] = r[x]; }
 5  int adjacent(int x) {
 6      for (int i = r[0]; i <= n; i = r[i]) if (graph[x][i]) return i;
 7      return 0;
 8  }
 9  int main() {
10      scanf("%d\n", &n);
11      for (int i = 1; i <= n; ++i) {
12          gets(buf);
13          string str = buf;
14          istringstream sin(str);
15          int x;
16          while (sin >> x) {
17              graph[i][x] = true;
18          }
19          l[i] = i - 1;
20          r[i] = i + 1;
21      }
22      for (int i = 2; i <= n; ++i)
23          if (graph[1][i]) {
24              s = 1;
25              t = i;
26              cover(s);
27              cover(t);
28              next[s] = t;
29              break;
```

```
30          }
31      while (true) {
32          int x;
33          while (x = adjacent(s)) {
34              next[x] = s;
35              s = x;
36              cover(s);
37          }
38          while (x = adjacent(t)) {
39              next[t] = x;
40              t = x;
41              cover(t);
42          }
43          if (!graph[s][t]) {
44              for (int i = s, j; i != t; i = next[i])
45                  if (graph[s][next[i]] && graph[t][i]) {
46                      for (j = s; j != i; j = next[j])
47                          last[next[j]] = j;
48                      j = next[s];
49                      next[s] = next[i];
50                      next[t] = i;
51                      t = j;
52                      for (j = i; j != s; j = last[j])
53                          next[j] = last[j];
54                      break;
55                  }
56          }
57          next[t] = s;
58          if (r[0] > n)
59              break;
60          for (int i = s; i != t; i = next[i])
61              if (adjacent(i)) {
62                  s = next[i];
63                  t = i;
64                  next[t] = 0;
65                  break;
66              }
67      }
68      for (int i = s; ; i = next[i]) {
69          if (i == 1) {
70              printf("%d", i);
71              for (int j = next[i]; j != i; j = next[j])
72                  printf("␣%d", j);
73              printf("␣%d\n", i);
74              break;
75          }
76          if (i == t)
77              break;
78      }
```

```
79 }
```

5.10 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 int read(){
 6     char ch;
 7     for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
 8     int cnt = 0;
 9     for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
10     return(cnt);
11 }
12
13 inline void makelist(int x, int y){
14     where[++l] = y;
15     next[l] = first[x];
16     first[x] = l;
17 }
18
19 bool cmp(const int &x, const int &y){
20     return(idx[x] < idx[y]);
21 }
22
23 int main(){
24    //freopen("1015.in", "r", stdin);
25    // freopen("1015.out", "w", stdout);
26     for (;;)
27     {
28         n = read(); m = read();
29         if (!n && !m) return 0;
30         memset(first, 0, sizeof(first)); l = 0;
31         memset(b, false, sizeof(b));
32         for (int i = 1; i <= m; i++)
33         {
34             int x = read(), y = read();
35             if (x != y && !b[x][y])
36             {
37                b[x][y] = true; b[y][x] = true;
38                makelist(x, y); makelist(y, x);
39             }
40         }
41         memset(f, 0, sizeof(f));
42         memset(L, 0, sizeof(L));
43         memset(R, 255, sizeof(R));
```

```
44          L[0] = 1; R[0] = n;
45          for (int i = 1; i <= n; i++) c[i] = i, pos[i] = i;
46          memset(idx, 0, sizeof(idx));
47          memset(v, 0, sizeof(v));
48          for (int i = n; i; --i)
49          {
50              int now = c[i];
51              R[f[now]]--;
52              if (R[f[now]] < L[f[now]]) R[f[now]] = -1;
53              idx[now] = i; v[i] = now;
54              for (int x = first[now]; x; x = next[x])
55                  if (!idx[where[x]])
56                  {
57                     swap(c[pos[where[x]]], c[R[f[where[x]]]]);
58                     pos[c[pos[where[x]]]] = pos[where[x]];
59                     pos[where[x]] = R[f[where[x]]];
60                     L[f[where[x]] + 1] = R[f[where[x]]]--;
61                     if (R[f[where[x]]] < L[f[where[x]]]) R[f[where[x]]] = -1;
62                     if (R[f[where[x]] + 1] == -1)
63                         R[f[where[x]] + 1] = L[f[where[x]] + 1];
64                     ++f[where[x]];
65                  }
66          }
67          bool ok = true;
68          //v是完美消除序列.
69          for (int i = 1; i <= n && ok; i++)
70          {
71              int cnt = 0;
72              for (int x = first[v[i]]; x; x = next[x])
73                  if (idx[where[x]] > i) c[++cnt] = where[x];
74              sort(c + 1, c + cnt + 1, cmp);
75              bool can = true;
76              for (int j = 2; j <= cnt; j++)
77                  if (!b[c[1]][c[j]])
78                  {
79                      ok = false;
80                      break;
81                  }
82          }
83          if (ok) printf("Perfect\n");
84          else printf("Imperfect\n");
85          printf("\n");
86      }
87  }
```

5.11 弦图求团数

```
int n, m, first[100001], next[2000001], where[2000001], l, L[100001], R[100001], c
    [100001], f[100001],
pos[100001], idx[100001], v[100001], ans;

inline void makelist(int x, int y){
    where[++l] = y;
    next[l] = first[x];
    first[x] = l;
}

int read(){
    char ch;
    for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
    int cnt = 0;
    for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
    return(cnt);
}

int main(){
    freopen("1006.in", "r", stdin);
    freopen("1006.out", "w", stdout);
    memset(first, 0, sizeof(first)); l = 0;
    n = read(); m = read();
    for (int i = 1; i <= m; i++)
    {
        int x, y;
        x = read(); y = read();
        makelist(x, y); makelist(y, x);
    }
    memset(L, 0, sizeof(L));
    memset(R, 255, sizeof(R));
    memset(f, 0, sizeof(f));
    memset(idx, 0, sizeof(idx));
    for (int i = 1; i <= n; i++) c[i] = i, pos[i] = i;
    L[0] = 1; R[0] = n; ans = 0;
    for (int i = n; i; --i)
    {
        int now = c[i], cnt = 1;
        idx[now] = i; v[i] = now;
        if (--R[f[now]] < L[f[now]]) R[f[now]] = -1;
        for (int x = first[now]; x; x = next[x])
            if (!idx[where[x]])
            {
                swap(c[pos[where[x]]], c[R[f[where[x]]]]);
                pos[c[pos[where[x]]]] = pos[where[x]];
                pos[where[x]] = R[f[where[x]]];
                L[f[where[x]] + 1] = R[f[where[x]]]--;
                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  }
```

5.12 有根树的同构

```
1   //http://acm.sdut.edu.cn/judgeonline/showproblem?problem_id=1861 ÓÐ¸ùÊ÷µÄÍ¬¹¹
2   const int mm=1051697,p=4773737;
3   int m,n,first[101],where[10001],next[10001],l,hash[10001],size[10001],pos[10001];
4   long long f[10001],rt[10001];
5   bool in[10001];
6
7   inline void makelist(int x,int y){
8       where[++l]=y;
9       next[l]=first[x];
10      first[x]=l;
11  }
12
13
14  inline void hashwork(int now){
15      int a[1001],v[1001],tot=0;
16      size[now]=1;
17      for (int x=first[now];x;x=next[x])
18      {
19          hashwork(where[x]);
20          a[++tot]=f[where[x]];
21          v[tot]=size[where[x]];
22          size[now]+=size[where[x]];
23      }
24      a[++tot]=size[now];
25      v[tot]=1;
26      int len=0;
27      for (int i=1;i<=tot;i++)
28         for (int j=i+1;j<=tot;j++)
29            if (a[j]<a[i])
30            {
31               int u=a[i];a[i]=a[j];a[j]=u;
32               u=v[i];v[i]=v[j];v[j]=u;
33            }
34      f[now]=1;
35      for (int i=1;i<=tot;i++)
36         {
```

```
37              f[now]=((f[now]*a[i])%p*rt[len])%p;
38              len+=v[i];
39          }
40  }
41
42  int main(){
43      //freopen("1.txt","r",stdin);
44      //freopen("2.txt","w",stdout);
45      scanf("%d%d",&n,&m);
46      rt[0]=1;
47      for (int i=1;i<=100;i++)
48          rt[i]=(rt[i-1]*mm)%p;
49      for (int i=1;i<=n;i++)
50      {
51          memset(first,0,sizeof(first));
52          memset(in,false,sizeof(in));
53          l=0;
54          for (int j=1;j<m;j++)
55          {
56              int x,y;
57              scanf("%d%d",&x,&y);
58              makelist(x,y);
59              in[y]=true;
60          }
61          int root=0;
62          for (int j=1;j<=m;j++)
63          if (!in[j])
64          {
65              root=j;
66              break;
67          }
68          memset(size,0,sizeof(size));
69          memset(f,0,sizeof(f));
70          hashwork(root);
71          hash[i]=f[root];
72      }
73      for (int i=1;i<=n;i++) pos[i]=i;
74      memset(in,false,sizeof(in));
75      for (int i=1;i<=n;i++)
76       if (!in[i])
77       {
78                  printf("%d",i);
79                  for (int j=i+1;j<=n;j++)
80                  if (hash[j]==hash[i])
81                  {
82                      in[j]=true;
83                      printf("=%d",j);
84                  }
85                  printf("\n");
```

```
86      }
87  }
```

5.13 zkw 费用流

```
 1  #include <cstdio>
 2  #include <cstdlib>
 3  #include <algorithm>
 4  #include <cstring>
 5  #include <cmath>
 6  using namespace std;
 7
 8  const int N = 105 << 2, M = 205 * 205 * 2;
 9  const int inf = 1000000000;
10
11  struct eglist {
12      int other[M], succ[M], last[N], cap[M], cost[M], sum;
13      void clear() {
14          memset(last, -1, sizeof(last));
15          sum = 0;
16      }
17      void _addEdge(int a, int b, int c, int d) {
18          other[sum] = b, succ[sum] = last[a], last[a] = sum, cost[sum] = d, cap[sum++]
                = c;
19      }
20      void addEdge(int a, int b, int c, int d) {
21          _addEdge(a, b, c, d);
22          _addEdge(b, a, 0, -d);
23      }
24  }e;
25
26  int n, m, S, T, tot, totFlow, totCost;
27  int dis[N], slack[N], visit[N], cur[N];
28
29  int modlable() {
30      int delta = inf;
31      for(int i = 1; i <= T; i++) {
32          if (!visit[i] && slack[i] < delta)
33              delta = slack[i];
34          slack[i] = inf;
35          cur[i] = e.last[i];
36      }
37      if (delta == inf)
38          return 1;
39      for(int i = 1; i <= T; i++)
40          if (visit[i])
41              dis[i] += delta;
42      return 0;
```

```
43  }
44
45  int dfs(int x, int flow) {
46      if (x == T) {
47          totFlow += flow;
48          totCost += flow * (dis[S] - dis[T]);
49          return flow;
50      }
51      visit[x] = 1;
52      int left = flow;
53      for(int &i = cur[x]; ~i; i = e.succ[i])
54          if (e.cap[i] > 0 && !visit[e.other[i]]) {
55              int y = e.other[i];
56              if (dis[y] + e.cost[i] == dis[x]) {
57                  int delta = dfs(y, min(left, e.cap[i]));
58                  e.cap[i] -= delta;
59                  e.cap[i ^ 1] += delta;
60                  left -= delta;
61                  if (!left)
62                      return flow;
63              } else {
64                  slack[y] = min(slack[y], dis[y] + e.cost[i] - dis[x]);
65              }
66          }
67      return flow - left;
68  }
69
70  pair<int, int> minCost() {
71      totFlow = 0, totCost = 0;
72      fill(dis + 1, dis + T + 1, 0);
73      for(int i = 1; i <= T; i++)
74          cur[i] = e.last[i];
75      do {
76          do {
77              fill(visit + 1, visit + T + 1, 0);
78          } while(dfs(S, inf));
79      } while(!modlable());
80      return make_pair(totFlow, totCost);
81  }
82
83  void run() {
84      scanf("%d␣%d", &m, &n);
85      e.clear();
86      S = m + n + 1, T = m + n + 2;
87      tot = 0;
88      for(int i = 1; i <= m; i++) {
89          int times;
90          scanf("%d", &times);
91          e.addEdge(S, i, times, 0);
```

```
92      }
93      for(int i = 1; i <= n; i++) {
94          int times;
95          scanf("%d", &times);
96          e.addEdge(i + m, T, times, 0);
97      }
98      for(int i = 1; i <= m; i++)
99          for(int j = 1; j <= n; j++) {
100             int cost;
101             scanf("%d", &cost);
102             e.addEdge(i, j + m, inf, cost);
103         }
104     pair<int, int> tmp = minCost();
105     printf("%d\n", tmp.second);
106 }
107
108 int main() {
109     int Test;
110     scanf("%d", &Test);
111     for(; Test--; run());
112     return 0;
113 }
```

Chapter 6

字符串

6.1 扩展 KMP

传入字符串 s 和长度 N, next[i]=LCP(s, s[i..N-1])

```
1  void z(char *s, int *next, int N)
2  {
3      int j = 0, k = 1;
4      while (j + 1 < N && s[j] == s[j + 1]) ++ j;
5      next[0] = N - 1; next[1] = j;
6      for(int i = 2; i < N; ++ i) {
7          int far = k + next[k] - 1, L = next[i - k];
8          if (L < far - i + 1) next[i] = L;
9          else {
10             j = max(0, far - i + 1);
11             while (i + j < N && s[j] == s[i + j]) ++ j;
12             next[i] = j; k = i;
13         }
14     }
15 }
```

6.2 后缀数组

字符串后面会自动加上一个最小字符 \0.

```
1  const int N = 4 * int(1e5) + 10;
2
3  int n, m;
4  int sa[N], ta[N], tb[N], *rank = ta, *tmp = tb;
5  int height[N], myLog[N], f[N][20];
6  int str[N];
7
8  bool cmp(int i, int j, int l) {
9      return tmp[i] == tmp[j] && tmp[i + l] == tmp[j + l];
10 }
```

```
11
12  void radixSort() {
13      static int w[N];
14      fill(w, w + m, 0);
15      for (int i = 0; i < n; i++) {
16          w[rank[i]]++;
17      }
18      for (int i = 1; i < m; i++) {
19          w[i] += w[i - 1];
20      }
21      for (int i = n - 1; i >= 0; i--) {
22          sa[--w[rank[tmp[i]]]] = tmp[i];
23      }
24  }
25
26  void suffixArray() {
27      for (int i = 0; i < n; i++) {
28          rank[i] = str[i];
29          tmp[i] = i;
30      }
31      radixSort();
32      for (int j = 1, i, p; j < n; j <<= 1, m = p) {
33          for (i = n - j, p = 0; i < n; i++) {
34              tmp[p++] = i;
35          }
36          for (i = 0; i < n; i++) {
37              if (sa[i] >= j) {
38                  tmp[p++] = sa[i] - j;
39              }
40          }
41          radixSort();
42          for (swap(tmp, rank), rank[sa[0]] = 0, i = p = 1; i < n; i++) {
43              rank[sa[i]] = cmp(sa[i - 1], sa[i], j) ? p - 1 : p++;
44          }
45      }
46      for (int i = 0, j, k = 0; i < n; ++i, k = max(k - 1, 0)) {
47          if (rank[i]) {
48              j = sa[rank[i] - 1];
49              for (; str[i + k] == str[j + k]; k++);
50              height[rank[i]] = k;
51          }
52      }
53      for (int i = 2; i <= n; i++) {
54          myLog[i] = myLog[i >> 1] + 1;
55      }
56      for (int i = 1; i < n; i++) {
57          f[i][0] = height[i];
58      }
59      for (int j = 1; 1 << j <= n; j++) {
```

```
60          for (int i = 1; i + (1 << j) <= n; i++) {
61              f[i][j] = min(f[i][j - 1], f[i + (1 << j - 1)][j - 1]);
62          }
63      }
64  }
65
66  int lcp(int l, int r) {
67      if (l > r) {
68          return 0;
69      }
70      int len = myLog[r - l + 1];
71      return min(f[l][len], f[r - (1 << len) + 1][len]);
72  }
73
74  int nBase, mBase;
75  int cnt[N];
76  char buf[N];
77
78  int pos(int x) {
79      return x / (mBase << 1 | 1 );
80  }
81
82  int main() {
83      n = 0;
84      m = 256;
85      scanf("%d%d", &nBase, &mBase);
86      for (int i = 0; i < nBase; i++) {
87          scanf("%s", buf);
88          for (int j = 0; j < mBase; j++) {
89              str[n++] = buf[j];
90          }
91          for (int j = 0; j < mBase; j++) {
92              str[n++] = buf[j];
93          }
94          str[n++] = i < nBase - 1 ? m++ : 0;
95      }
96      suffixArray();
97      int result = 0, total = 0;
98      for (int i = 0, j = 0; i < n; i++) {
99          for (; j < n && total < nBase; j++) {
100             int p = pos(sa[j]);
101             total += cnt[p]++ == 0;
102         }
103         if (total == nBase) {
104             result = max(result, lcp(i + 1, j - 1));
105         }
106         int p = pos(sa[i]);
107         total -= --cnt[p] == 0;
108     }
```

```
109     result = min(result, mBase);
110     printf("%d\n", result);
111     vector <int> ans(n);
112     total = 0;
113     memset(cnt, 0, sizeof(cnt));
114     for (int i = 0, j = 0; i < n; i++) {
115         for (; j < n && total < nBase; j++) {
116             int p = pos(sa[j]);
117             total += cnt[p]++ == 0;
118         }
119         if (total == nBase && lcp(i + 1, j - 1) >= result) {
120             for (int k = i; k < j; k++) {
121                 int p = pos(sa[k]);
122                 ans[p] = sa[k] % (mBase << 1 | 1);
123             }
124             break;
125         }
126         int p = pos(sa[i]);
127         total -= --cnt[p] == 0;
128     }
129     for (int i = 0; i < nBase; i++) {
130         printf("%d\n", ans[i] % mBase + 1);
131     }
132 }
```

6.3 DC3

```
1  //`DC3 待排序的字符串放在r 数组中，从r[0]到r[n-1]，长度为n，且最大值小于m.`
2  //`约定除r[n-1]外所有的r[i]都大于0, r[n-1]=0。`
3  //`函数结束后，结果放在sa 数组中，从sa[0]到sa[n-1]。`
4  //`r必须开长度乘3`
5  #define maxn 10000
6  #define F(x) ((x)/3+((x)%3==1?0:tb))
7  #define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
8
9  int wa[maxn],wb[maxn],wv[maxn],wss[maxn];
10 int s[maxn*3],sa[maxn*3];
11 int c0(int *r,int a,int b)
12 {
13     return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];
14 }
15 int c12(int k,int *r,int a,int b)
16 {
17     if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
18     else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
19 }
20 void sort(int *r,int *a,int *b,int n,int m)
```

```
{
    int i;
    for(i=0;i<n;i++) wv[i]=r[a[i]];
    for(i=0;i<m;i++) wss[i]=0;
    for(i=0;i<n;i++) wss[wv[i]]++;
    for(i=1;i<m;i++) wss[i]+=wss[i-1];
    for(i=n-1;i>=0;i--) b[--wss[wv[i]]]=a[i];
}
void dc3(int *r,int *sa,int n,int m)
{
    int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
    r[n]=r[n+1]=0;
    for(i=0;i<n;i++)
        if(i%3!=0) wa[tbc++]=i;
    sort(r+2,wa,wb,tbc,m);
    sort(r+1,wb,wa,tbc,m);
    sort(r,wa,wb,tbc,m);
    for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
        rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
    if (p<tbc) dc3(rn,san,tbc,p);
    else for (i=0;i<tbc;i++) san[rn[i]]=i;
    for (i=0;i<tbc;i++)
        if(san[i]<tb) wb[ta++]=san[i]*3;
    if(n%3==1) wb[ta++]=n-1;
    sort(r,wb,wa,ta,m);
    for(i=0;i<tbc;i++)
        wv[wb[i]=G(san[i])]=i;
    for(i=0,j=0,p=0;i<ta && j<tbc;p++)
        sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
    for(;i<ta;p++) sa[p]=wa[i++];
    for(;j<tbc;p++) sa[p]=wb[j++];
}

int main(){
    int n,m=0;
    scanf("%d",&n);
    for (int i=0;i<n;i++) scanf("%d",&s[i]),s[i]++,m=max(s[i]+1,m);
    printf("%d\n",m);
    s[n++]=0;
    dc3(s,sa,n,m);
    for (int i=0;i<n;i++) printf("%d␣",sa[i]);printf("\n");
}
```

6.4 AC 自动机

```
namespace aho_corasick_automation {
    int const N = ;
    struct node {
```

```
4         node *next[N], *fail;
5         int count;
6         inline node() {
7             memset(next, 0, sizeof(next));
8             fail = 0;
9             count = 0;
10        }
11    };
12
13    node *root;
14
15    inline int idx(char x) {
16        return x - 'a';
17    }
18
19    inline void insert(node *x, char *str) {
20        int len = (int)strlen(str);
21        for (int i = 0; i < len; ++i) {
22            int c = idx(str[i]);
23            if (!x->next[c]) {
24                x->next[c] = new node();
25            }
26            x = x->next[c];
27        }
28        x->count++;
29    }
30
31    inline void build() {
32        vector<node*> queue;
33        queue.push_back(root->fail = root);
34        for (int head = 0; head < (int)queue.size(); ++head) {
35            node* x = queue[head];
36            for (int i = 0; i < N; ++i) {
37                if (x->next[i]) {
38                    x->next[i]->fail = (x == root) ? root : x->fail->next[i];
39                    x->next[i]->count += x->next[i]->fail->count;
40                    queue.push_back(x->next[i]);
41                } else {
42                    x->next[i] = (x == root) ? root : x->fail->next[i];
43                }
44            }
45        }
46    }
47
48    inline void prepare() {
49        root = new node();
50    }
51 }
```

6.5 极长回文子串

```
//CF17 – E
typedef long long int64;
const int N = 4 * int(1e6) + 111;
const int mod = 51123987;
int n;
int input[N];
int start[N], finish[N];
int f[N];
int64 ans;
void prepare() {
    int k = 0;
    for (int i = 0; i < n; ++i) {
        if (k + f[k] < i) {
            int &l = f[i] = 0;
            for (; i – l – 1 >= 0 && i + l + 1 < n && input[i – l – 1] ==
                    input[i + l + 1]; l++);
            k = i;
        } else {
            int &l = f[i] = f[k – (i – k)];
            if (i + l >= k + f[k]) {
                l = min(l, k + f[k] – i);
                for (; i – l – 1 >= 0 && i + l + 1 < n && input[i – l – 1] ==
                        input[i + l + 1]; l++);
                k = i;
            }
        }
        int l = i – f[i], r = i + f[i];
        l += l & 1;
        r –= r & 1;
        if (l <= r) {
            l /= 2;
            r /= 2;
            int mid1 = l + r >> 1;
            int mid2 = mid1 + ((l + r) & 1);
            start[l]++;
            start[mid1 + 1]––;
            finish[mid2]++;
            finish[r + 1]––;
            ans = (ans + (r – l) / 2 + 1) % mod;
        }
    }
}
int main() {
    scanf("%d␣", &n);
    for (int i = 0; i < n; ++i) {
        input[i << 1] = getchar();
        if (i < n – 1)
```

```
48              input[i << 1 | 1] = '*';
49      }
50      n = n * 2 - 1;
51      prepare();
52      ans = ans * (ans - 1) / 2 % mod;
53      n = (n + 1) / 2;
54      int sum = 0;
55      for (int i = 0; i < n; ++i) {
56          if (i) {
57              start[i] = (start[i] + start[i - 1]) % mod;
58              finish[i] = (finish[i] + finish[i - 1]) % mod;
59          }
60          ans = (ans - (int64)start[i] * sum % mod) % mod;
61          sum = (sum + finish[i]) % mod;
62      }
63      cout << (ans + mod) % mod << endl;
64  }
```

6.6 后缀自动机 --多个串的最长公共子串

```
 1
 2  const int N = 255555;
 3  const int C = 36;
 4
 5  struct Node {
 6      Node *next[C], *fail;
 7      int count, len, dp, dp2;
 8      void clear() {
 9          for(int i = 0; i < C; i++)
10              next[i] = NULL;
11          len = count = 0;
12          fail = NULL;
13      }
14  };
15
16  Node *tail, *q[N * 2], pool[N * 2], *head;
17  int used = 0, top = 0;
18  char bufer[N * 2];
19
20  Node *newNode() {
21      pool[used++].clear();
22      return &pool[used - 1];
23  }
24
25  void add(int x) {
26      Node *np = newNode(), *p = tail;
27      tail = np;
28      np->len = p->len + 1;
```

```
29      for(; p && !p->next[x]; p = p->fail)
30          p->next[x] = np;
31      if (!p)
32          np->fail = head;
33      else if (p->len + 1 == p->next[x]->len)
34          np->fail = p->next[x];
35      else {
36          Node *q = p->next[x], *nq = newNode();
37          *nq = *q;
38          nq->len = p->len + 1;
39          q->fail = np->fail = nq;
40          for(; p && p->next[x] == q; p = p->fail)
41              p->next[x] = nq;
42      }
43  }
44
45  int main() {
46      scanf("%s", bufer);
47      int length = strlen(bufer);
48      head = tail = newNode();
49      for(int i = 0; i < length; i++)
50          add(bufer[i] - 'a');
51      for(int i = 0; i < used; i++)
52          pool[i].count = 0, pool[i].dp = pool[i].len;
53      int number = 0;
54      while(scanf("%s", bufer) == 1) {
55          number++;
56          length = strlen(bufer);
57          Node *iter = head;
58          int cur = 0;
59          top = 0;
60          for(int i = 0; i < length; i++) {
61              int x = bufer[i] - 'a';
62              while(iter != head && !iter->next[x])
63                  iter = iter->fail, cur = iter->len;
64              if (iter->next[x]) {
65                  cur++;
66                  iter = iter->next[x];
67              }
68              q[top++] = iter;
69              if (iter->count == number - 1) {
70                  iter->count = number;
71                  iter->dp2 = cur;
72              } else if (iter->count == number) {
73                  iter->dp2 = max(iter->dp2, cur);
74              } else {
75                  top--;
76              }
77          }
```

```
78          for(int i = 0; i < top; i++) {
79              q[i]->dp = min(q[i]->dp, q[i]->dp2);
80          }
81      }
82      int ans = 0;
83      for(int i = 0; i < used; i++)
84          if (pool[i].count == number)
85              ans = max(ans, pool[i].dp);
86      printf("%d\n", ans);
87      return 0;
88  }
```

6.7 后缀自动机 --多次询问串在母串中的出现次数

```
 1
 2  const int N = 255555;
 3  const int C = 36;
 4
 5  struct Node {
 6      Node *next[C], *fail;
 7      int count, len;
 8      void clear() {
 9          for(int i = 0; i < C; i++)
10              next[i] = NULL;
11          len = count = 0;
12          fail = NULL;
13      }
14  };
15
16  Node *tail, *q[N * 2], pool[N * 2], *head;
17  int used = 0;
18  char bufer[N * 2];
19  int buc[N * 2], f[N * 2];
20
21  Node *newNode() {
22      pool[used++].clear();
23      return &pool[used - 1];
24  }
25
26  void add(int x) {
27      Node *np = newNode(), *p = tail;
28      tail = np;
29      np->len = p->len + 1;
30      for(; p && !p->next[x]; p = p->fail)
31          p->next[x] = np;
32      if (!p)
33          np->fail = head;
34      else if (p->len + 1 == p->next[x]->len)
```

```
35          np->fail = p->next[x];
36      else {
37          Node *q = p->next[x], *nq = newNode();
38          *nq = *q;
39          nq->len = p->len + 1;
40          q->fail = np->fail = nq;
41          for(; p && p->next[x] == q; p = p->fail)
42              p->next[x] = nq;
43      }
44  }
45
46  int main() {
47      scanf("%s\n", bufer);
48      int length = strlen(bufer);
49      head = tail = newNode();
50      for(int i = 0; i < length; i++)
51          add(bufer[i] - 'a');
52      for(int i = 0; i < used; ++i)
53          ++buc[pool[i].len];
54      for(int i = 1; i <= length; i++)
55          buc[i] += buc[i - 1];
56      for(int i = used - 1; i >= 0; i--)
57          q[--buc[pool[i].len]] = &pool[i];
58      Node *iter = head;
59      for(int i = 0; i < length; ++i)
60          (iter = iter->next[bufer[i] - 'a'])->count++;
61      for(int i = used - 1; i > 0; --i) {
62          f[q[i]->len] = max(f[q[i]->len], q[i]->count);
63          q[i]->fail->count += q[i]->count;
64      }
65      for(int i = length - 1; i > 0; --i) {
66          f[i] = max(f[i + 1], f[i]);
67      }
68      for(int i = 1; i <= length; i++)
69          printf("%d\n", f[i]);
70      return 0;
71  }
```

6.8 循环串的最小表示

```
1  struct cyc_string
2  {
3      int n, offset;
4      char str[max_length];
5      char & operator [] (int x)
6      {return str[((offset + x) % n)];}
7      cyc_string(){offset = 0;}
8  };
```

```
 9 void minimum_circular_representation(cyc_string & a)
10 {
11     int i = 0, j = 1, dlt = 0, n = a.n;
12     while(i < n and j < n and dlt < n)
13     {
14       if(a[i + dlt] == a[j + dlt]) dlt++;
15       else
16       {
17         if(a[i + dlt] > a[j + dlt]) i += dlt + 1; else j += dlt + 1;
18         dlt = 0;
19       }
20     }
21     a.offset = min(i, j);
22 }
23 int main()
24 {return 0;}
```

Chapter 7

Others

7.1 快速求逆

```
1  int inverse(int x, int modulo) {
2      if(x == 1)
3          return 1;
4      return (long long)(modulo - modulo / x) * inverse(modulo % x, modulo) % modulo;
5  }
```

7.2 求某年某月某日星期几

```
1  int whatday(int d, int m, int y)
2  {
3      int ans;
4      if (m == 1 || m == 2) {
5          m += 12; y --;
6      }
7      if ((y < 1752) || (y == 1752 && m < 9) || (y == 1752 && m == 9 && d < 3))
8          ans = (d + 2 * m + 3 * (m + 1) / 5 + y + y / 4 + 5) % 7;
9      else ans = (d + 2 * m + 3 * (m + 1) / 5 + y + y / 4 - y / 100 + y / 400) % 7;
10     return ans;
11 }
```

7.3 LL*LL%LL

```
1  LL multiplyMod(LL a, LL b, LL P) { // `需要保证 a 和 b 非负`
2      LL t = (a * b - LL((long double)a / P * b + 1e-3) * P) % P;
3      return t < 0 : t + P : t;
4  }
```

7.4 next_nCk

```
void nCk(int n, int k) {
    for (int comb = (1 << k) - 1; comb < (1 << n); ) {
        // ...
        {
            int x = comb & -comb, y = comb + x;
            comb = (((comb & ~y) / x) >> 1) | y;
        }
    }
}
```

7.5 单纯形

test on uva 12567

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

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

```
32                  }
33              }
34              for(int i = 0; i <= n + 1; i++) {
35                  if (i != r) {
36                      for(int j = 0; j <= m; j++) {
37                          if (j != s) {
38                              D[i][j] += D[r][j] * D[i][s];
39                          }
40                      }
41                      D[i][s] *= D[r][s];
42                  }
43              }
44          }
45          r = -1, s = -1;
46          for(int j = 0; j < m; j++) {
47              if (s < 0 || ix[s] > ix[j]) {
48                  if (D[n + 1][j] > eps || D[n + 1][j] > -eps && D[n][j] > eps) {
49                      s = j;
50                  }
51              }
52          }
53          if (s < 0) {
54              break;
55          }
56          for(int i = 0; i < n; i++) {
57              if (D[i][s] < -eps) {
58                  if (r < 0 || (d = D[r][m] / D[r][s] - D[i][m] / D[i][s]) < -eps
59                      || d < eps && ix[r + m] > ix[i + m]) {
60
61                      r = i;
62                  }
63              }
64          }
65
66          if (r < 0) {
67              return vector<double> ();
68          }
69      }
70      if (D[n + 1][m] < -eps) {
71          return vector<double> ();
72      }
73
74      vector<double> x(m - 1);
75      for(int i = m; i < n + m; i++) {
76          if (ix[i] < m - 1) {
77              x[ix[i]] = D[i - m][m];
78          }
79      }
80      return x;
```

```
81 }
```

7.6 曼哈顿最小生成树

```
 1 /*
 2 `只需要考虑每个点的 pi/4*k —— pi/4*(k+1)的区间内的第一个点，这样只有4n条无向边。`
 3 */
 4 const int maxn = 100000+5;
 5 const int Inf = 1000000005;
 6 struct TreeEdge
 7 {
 8     int x,y,z;
 9     void make( int _x,int _y,int _z ) { x=_x; y=_y; z=_z; }
10 } data[maxn*4];
11
12 inline bool operator < ( const TreeEdge& x,const TreeEdge& y ){
13     return x.z<y.z;
14 }
15
16 int x[maxn],y[maxn],px[maxn],py[maxn],id[maxn],tree[maxn],node[maxn],val[maxn],fa[
       maxn];
17 int n;
18 inline bool compare1( const int a,const int b ) { return x[a]<x[b]; }
19 inline bool compare2( const int a,const int b ) { return y[a]<y[b]; }
20 inline bool compare3( const int a,const int b ) { return (y[a]-x[a]<y[b]-x[b] || y[a
       ]-x[a]==y[b]-x[b] && y[a]>y[b]); }
21 inline bool compare4( const int a,const int b ) { return (y[a]-x[a]>y[b]-x[b] || y[a
       ]-x[a]==y[b]-x[b] && x[a]>x[b]); }
22 inline bool compare5( const int a,const int b ) { return (x[a]+y[a]>x[b]+y[b] || x[a
       ]+y[a]==x[b]+y[b] && x[a]<x[b]); }
23 inline bool compare6( const int a,const int b ) { return (x[a]+y[a]<x[b]+y[b] || x[a
       ]+y[a]==x[b]+y[b] && y[a]>y[b]); }
24 void Change_X()
25 {
26     for(int i=0;i<n;++i) val[i]=x[i];
27     for(int i=0;i<n;++i) id[i]=i;
28     sort(id,id+n,compare1);
29     int cntM=1, last=val[id[0]]; px[id[0]]=1;
30     for(int i=1;i<n;++i)
31     {
32         if(val[id[i]]>last) ++cntM,last=val[id[i]];
33         px[id[i]]=cntM;
34     }
35 }
36 void Change_Y()
37 {
38     for(int i=0;i<n;++i) val[i]=y[i];
```

```
39      for(int i=0;i<n;++i) id[i]=i;
40      sort(id,id+n,compare2);
41      int cntM=1, last=val[id[0]]; py[id[0]]=1;
42      for(int i=1;i<n;++i)
43      {
44          if(val[id[i]]>last) ++cntM,last=val[id[i]];
45          py[id[i]]=cntM;
46      }
47  }
48  inline int absValue( int x ) { return (x<0)?-x:x; }
49  inline int Cost( int a,int b ) { return absValue(x[a]-x[b])+absValue(y[a]-y[b]); }
50  int find( int x ) { return (fa[x]==x)?x:(fa[x]=find(fa[x])); }
51  int main()
52  {
53  //  freopen("input.txt", "r", stdin);
54  //  freopen("output.txt", "w", stdout);
55
56      int test=0;
57      while( scanf("%d",&n)!=EOF && n )
58      {
59          for(int i=0;i<n;++i) scanf("%d%d",x+i,y+i);
60          Change_X();
61          Change_Y();
62
63          int cntE = 0;
64          for(int i=0;i<n;++i) id[i]=i;
65          sort(id,id+n,compare3);
66          for(int i=1;i<=n;++i) tree[i]=Inf,node[i]=-1;
67          for(int i=0;i<n;++i)
68          {
69              int Min=Inf, Tnode=-1;
70              for(int k=py[id[i]];k<=n;k+=k&(-k)) if(tree[k]<Min) Min=tree[k],Tnode=
                    node[k];
71              if(Tnode>=0) data[cntE++].make(id[i],Tnode,Cost(id[i],Tnode));
72              int tmp=x[id[i]]+y[id[i]];
73              for(int k=py[id[i]];k;k-=k&(-k)) if(tmp<tree[k]) tree[k]=tmp,node[k]=id[i
                    ];
74          }
75          sort(id,id+n,compare4);
76          for(int i=1;i<=n;++i) tree[i]=Inf,node[i]=-1;
77          for(int i=0;i<n;++i)
78          {
79              int Min=Inf, Tnode=-1;
80              for(int k=px[id[i]];k<=n;k+=k&(-k)) if(tree[k]<Min) Min=tree[k],Tnode=
                    node[k];
81              if(Tnode>=0) data[cntE++].make(id[i],Tnode,Cost(id[i],Tnode));
82              int tmp=x[id[i]]+y[id[i]];
83              for(int k=px[id[i]];k;k-=k&(-k)) if(tmp<tree[k]) tree[k]=tmp,node[k]=id[i
                    ];
```

```
84          }
85          sort(id,id+n,compare5);
86          for(int i=1;i<=n;++i) tree[i]=Inf,node[i]=-1;
87          for(int i=0;i<n;++i)
88          {
89              int Min=Inf, Tnode=-1;
90              for(int k=px[id[i]];k;k-=k&(-k)) if(tree[k]<Min) Min=tree[k],Tnode=node[k
                    ];
91              if(Tnode>=0) data[cntE++].make(id[i],Tnode,Cost(id[i],Tnode));
92              int tmp=-x[id[i]]+y[id[i]];
93              for(int k=px[id[i]];k<=n;k+=k&(-k)) if(tmp<tree[k]) tree[k]=tmp,node[k]=
                    id[i];
94          }
95          sort(id,id+n,compare6);
96          for(int i=1;i<=n;++i) tree[i]=Inf,node[i]=-1;
97          for(int i=0;i<n;++i)
98          {
99              int Min=Inf, Tnode=-1;
100             for(int k=py[id[i]];k<=n;k+=k&(-k)) if(tree[k]<Min) Min=tree[k],Tnode=
                    node[k];
101             if(Tnode>=0) data[cntE++].make(id[i],Tnode,Cost(id[i],Tnode));
102             int tmp=-x[id[i]]+y[id[i]];
103             for(int k=py[id[i]];k;k-=k&(-k)) if(tmp<tree[k]) tree[k]=tmp,node[k]=id[i
                    ];
104         }
105
106         long long Ans = 0;
107         sort(data,data+cntE);
108         for(int i=0;i<n;++i) fa[i]=i;
109         for(int i=0;i<cntE;++i) if(find(data[i].x)!=find(data[i].y))
110         {
111             Ans += data[i].z;
112             fa[fa[data[i].x]]=fa[data[i].y];
113         }
114
115         cout<<"Case␣"<<++test<<":␣"<<"Total␣Weight␣=␣"<<Ans<<endl;
116     }
117     return 0;
118 }
```

7.7 最长公共子序列

7.7.1 最长公共子序列

```
1 const int dx[]={0,-1,0,1};
2 const int dy[]={1,0,-1,0};
3 const string ds="ENWS";
4 char G[52][52];
```

```
5  char A[22222], B[22222], buf[22222];
6  int n, m;
7
8  typedef unsigned long long ll;
9
10 const int M = 62;
11 const int maxn = 20010;
12 const int maxt = 130;
13 const int maxl = maxn / M + 10;
14 const ll Top = ((ll) 1 << (M));
15 const ll Topless = Top - 1;
16 const ll underTop = ((ll) 1 << (M - 1));
17 typedef ll bitarr[maxl];
18 bitarr comp[maxt], row[2], X;
19
20 void get(char *S){
21     int L,x,y,sz=0;
22     scanf("%d%d%d",&L,&x,&y),x--,y--;
23     //scanf(" %s",buf);
24     S[sz++]=G[x][y];
25     for(int i=0;i<L;i++){
26         char ch;
27         scanf("␣%c", &ch);
28         int pos=ds.find(ch);
29         x+=dx[pos],y+=dy[pos];
30         if (x < 0 || y < 0 || x >= n || y >= m) for(;;);
31         S[sz++]=G[x][y];
32     }
33     S[sz]=0;
34 }
35
36 bool calc[maxt];
37
38 void prepare() {
39
40     int u, p;
41     memset(calc, 0, sizeof(calc));
42     for (int i = 0; i < m; i++) {
43         u = B[i];
44         if (calc[u]) continue;  //======仅对所有字符集 ， 每次一次
45         calc[u] = 1;
46         memset(comp[u], 0, sizeof(comp[u]));
47         for (p = 0; p < n; p++) if (u == A[p]) comp[u][p / M] ^= ((ll) 1 << (p % M));
48     }
49 }
50
51 void solve() {
52     prepare();
53     memset(row, 0, sizeof(row));
```

```
54      int prev, curt;
55      int i, u, p, c, cc;
56      int Ln = (n / M) + 1;
57      prev = 0;
58      for (i = 0; i < m; i++) {
59          curt = 1 - prev; u = B[i];
60          for (p = 0; p < Ln; p++) X[p] = row[prev][p] | comp[u][p];
61          c = 0;
62          for (p = 0; p < Ln; p++) {
63              cc = (row[prev][p] & underTop) > 0;
64              row[prev][p] = ((row[prev][p] & (underTop - 1)) << 1) + c;
65              c = cc;
66          }
67          for (p = 0; p < Ln; p++) {
68              if (row[prev][p] != Topless) {
69                  row[prev][p]++;
70                  break;
71              }
72              row[prev][p] = 0;
73          }
74          c = 0;
75          for (p = 0; p < Ln; p++) {
76              if (X[p] >= row[prev][p] + c)
77                  row[prev][p] = X[p] - (row[prev][p] + c), c = 0;
78              else
79                  row[prev][p] = Top + X[p] - (row[prev][p] + c), c = 1;
80          }
81          for (p = 0; p < Ln; p++)
82              row[curt][p] = X[p] & (row[prev][p] ^ X[p]);
83          prev = curt;
84      }
85      int ret = 0;
86      for (i = 0; i < n; i++)
87          if (row[prev][i / M] & ((ll) 1 << (i % M))) ret++;
88  //  printf("%d %d %d\n", n, m, ret);
89  //=========ret 就是最长公共子序列。
90      printf("%d␣%d\n", n - ret, m - ret);
91  }
92
93  int main(){
94      int tests=0,T;
95      scanf("%d",&T);
96      while(T--){
97          scanf("%d%d",&n,&m);
98          for(int i=0;i<n;i++)
99              for (int j = 0; j < m; j++)
100                 scanf("␣%c",&G[i][j]);
101         get(A),get(B);
102
```

```
103         printf("Case␣%d:␣", ++tests);
104 //      printf("A = %s\n, B = %s\n", A, B);
105         n = strlen(A), m = strlen(B);
106         //n = 20000; m = 20000;
107         //for (int i = 0; i < m; i++) A[i] = B[i] = 'A';
108         //A[m] = B[m] = 0;
109         solve();
110     }
111 }
```

7.8 环状最长公共子序列

```
 1 const int N = 2222;
 2
 3 int a[N], b[N];
 4 int n, dp[N][N], from[N][N];
 5
 6 int run() {
 7     scanf("%d", &n);
 8     for(int i = 1; i <= n; i++) {
 9         scanf("%d", &a[i]);
10         a[i + n] = a[i];
11         b[n - i + 1] = a[i];
12     }
13     memset(from, 0, sizeof(from));
14     int ans = 0;
15     for(int i = 1; i <= 2 * n; i++) {
16         from[i][0] = 2;
17         int upleft = 0, up = 0, left = 0;
18         for(int j = 1; j <= n; j++) {
19             upleft = up;
20             if (a[i] == b[j]) {
21                 upleft++;
22             } else {
23                 upleft = INT_MIN;
24             }
25             if (from[i - 1][j])
26                 up++;
27             int mm = max(up, max(left, upleft));
28             if (mm == left) {
29                 from[i][j] = 0;
30             } else if (mm == upleft)
31                 from[i][j] = 1;
32             else
33                 from[i][j] = 2;
34             left = mm;
35         }
36         if (i >= n) {
```

```
37              int count = 0;
38              for(int x = i, y = n; y; ) {
39                  if (from[x][y] == 1) {
40                      x--; y--;
41                      count++;
42                  } else if (from[x][y] == 0)
43                      y--;
44                  else
45                      x--;
46              }
47              ans = max(ans, count);
48              int x = i - n + 1;
49              from[x][0] = 0;
50              int y = 0;
51              for(; y <= n && from[x][y] == 0; y++);
52              for(; x <= i; x++) {
53                  from[x][y] = 0;
54                  if (x == i) {
55                      break;
56                  }
57                  for(; y <= n; ++y) {
58                      if (from[x + 1][y] == 2) {
59                          break;
60                      }
61                      if (y + 1 <= n && from[x + 1][y + 1] == 1) {
62                          y++;
63                          break;
64                      }
65                  }
66              }
67          }
68      }
69      if (n)
70          printf("%d\n", ans);
71      return n;
72  }
73
74  int main() {
75      for(; run(); );
76      return 0;
77  }
```

7.9 长方体表面两点最近距离

```
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) {
4           int R = x*x+y*y;
```

```
5           if (R<r) r=R;
6       }
7       else{
8           if(i>=0 && i< 2)
9               turn(i+1, j, x0+L+z, y, x0+L-x, x0+L, y0, H, W, L);
10          if(j>=0 && j< 2)
11              turn(i, j+1, x, y0+W+z, y0+W-y, x0, y0+W, L, H, W);
12          if(i<=0 && i>-2)
13              turn(i-1, j, x0-z, y, x-x0, x0-H, y0, H, W, L);
14          if(j<=0 && j>-2)
15              turn(i, j-1, x, y0-z, y-y0, x0, y0-H, L, H, W);
16      }
17  }
18  int main(){
19      int L, H, W, x1, y1, z1, x2, y2, z2;
20      cin >> L >> W >> H >> x1 >> y1 >> z1 >> x2 >> y2 >> z2;
21      if (z1!=0 && z1!=H)
22      if (y1==0 || y1==W)
23          swap(y1,z1), std::swap(y2,z2), std::swap(W,H);
24      else
25          swap(x1,z1), std::swap(x2,z2), std::swap(L,H);
26      if (z1==H) z1=0, z2=H-z2;
27      r=0x3fffffff; turn(0,0,x2-x1,y2-y1,z2,-x1,-y1,L,W,H);
28      cout<<r<<endl;
29      return 0;
30  }
```

7.10 插头 DP

```
1   #include <cstdio>
2   #include <cstdlib>
3   #include <algorithm>
4   #include <vector>
5   #include <iostream>
6   using namespace std;
7
8   typedef long long int64;
9   typedef pair<int, long long> State;
10  const int MAXN = 8;
11
12  char map[MAXN + 10][MAXN + 10];
13  int n, m, lastx, lasty;
14  int64 ans;
15  vector<State> vec[2];
16
17
18  void mergy(int cur) {
19      sort(vec[cur].begin(), vec[cur].end());
```

```
20      int size = 0;
21      for(int i = 0, j = 0; i < vec[cur].size(); i = j) {
22          vec[cur][size] = vec[cur][i];
23          j = i + 1;
24          while(j < vec[cur].size() && vec[cur][j].first == vec[cur][size].first)
25              vec[cur][size].second += vec[cur][j].second, j++;
26          size++;
27      }
28      vec[cur].resize(size);
29  }
30
31  void next_line(int cur) {
32      int size = 0;
33      for(int i = 0; i < vec[cur].size(); i++) {
34          int sta = vec[cur][i].first;
35          if ((sta >> (m << 1)) == 0) {
36              vec[cur][size] = vec[cur][i];
37              vec[cur][size].first <<= 2;
38              size++;
39          }
40      }
41      vec[cur].resize(size);
42  }
43
44  inline int replace(int sta, int pos, int v) {
45      return (sta & (~(3 << (pos << 1)))) | (v << (pos << 1));
46  }
47
48  inline int replace(int &sta, int pos, int v1, int v2) {
49      int res = replace(sta, pos, v1);
50      res = replace(res, pos + 1, v2);
51      return res;
52  }
53
54  int Trans(int sta, int pos) {
55      int cnt = 1, v = (sta >> (pos << 1) & 3);
56      if (v == 1) {
57          sta = replace(sta, pos, 0, 0);
58          for(int i = pos + 2; ; i++) {
59              if ((sta >> (i << 1) & 3) == 1)
60                  cnt++;
61              else if ((sta >> (i << 1) & 3) == 2)
62                  cnt--;
63              if (cnt == 0)
64                  return replace(sta, i, 1);
65          }
66      } else {
67          sta = replace(sta, pos, 0, 0);
68          for(int i = pos - 1; ; i--) {
```

```
69              if ((sta >> (i << 1) & 3) == 1)
70                  cnt--;
71              else if ((sta >> (i << 1) & 3) == 2)
72                  cnt++;
73              if (cnt == 0)
74                  return replace(sta, i, 2);
75          }
76      }
77  }
78
79  void dp_block(int i, int j, int cur) {
80      for(int s = 0; s < vec[cur].size(); s++) {
81          int sta = vec[cur][s].first;
82          int64 val = vec[cur][s].second;
83          int left = (sta >> (j << 1)) & 3, up = (sta >> ((j + 1) << 1)) & 3;
84          if (left == 0 && up == 0) {
85              vec[cur ^ 1].push_back(State(sta, val));
86          }
87      }
88  }
89
90  void dp_blank(int i, int j, int cur) {
91      for(int s = 0; s < vec[cur].size(); s++) {
92          int sta = vec[cur][s].first;
93          int64 val = vec[cur][s].second;
94          int left = (sta >> (j << 1)) & 3, up = (sta >> ((j + 1) << 1)) & 3, ns = 0;
95          if (left && up) {
96              if (left == 2 && up == 1) {
97                  vec[cur ^ 1].push_back(State(replace(sta, j, 0, 0), val));
98              } else if (left == 1 && up == 2) {
99                  if (replace(sta, j, 0, 0) == 0 && i == lastx && j == lasty)
100                     ans += val;
101             } else if (left == 1 && up == 1) {
102                 vec[cur ^ 1].push_back(State(Trans(sta, j), val));
103             } else if (left == 2 && up == 2) {
104                 vec[cur ^ 1].push_back(State(Trans(sta, j), val));
105             }
106         } else if (left || up) {
107             vec[cur ^ 1].push_back(State(sta, val));
108             vec[cur ^ 1].push_back(State(replace(sta, j, up, left), val));
109         } else {
110             vec[cur ^ 1].push_back(State(replace(sta, j, 1, 2), val));
111         }
112     }
113 }
114
115 void show(int cur) {
116     for(int i = 0; i < vec[cur].size(); i++)
117         printf("%d␣%I64d\n", vec[cur][i].first, vec[cur][i].second);
```

```
118     printf("step\n");
119 }
120
121 int main() {
122     freopen("input.txt", "r", stdin);
123     while(scanf("%d␣%d", &n, &m) == 2) {
124         ans = 0;
125         lastx = lasty = -1;
126         gets(map[0]);
127         for(int i = 0; i < n; i++) {
128             scanf("%s", map[i]);
129             for(int j = 0; j < m; j++) {
130                 if (map[i][j] == '.') {
131                     lastx = i, lasty = j;
132                 }
133             }
134         }
135         if (lastx == -1) {
136             printf("0\n");
137             continue;
138         }
139         int cur = 0;
140         vec[cur].clear();
141         vec[cur].push_back(State(0, 1));
142         for(int i = 0; i < n; i++) {
143             for(int j = 0; j < m; j++) {
144                 vec[cur ^ 1].clear();
145                 if (map[i][j] == '.')
146                     dp_blank(i, j, cur);
147                 else
148                     dp_block(i, j, cur);
149                 cur ^= 1;
150                 mergy(cur);
151                 //show(cur);
152             }
153             next_line(cur);
154         }
155         cout << ans << endl;
156     }
157     return 0;
158 }
```

7.11 最大团搜索

Int g[][] 为图的邻接矩阵。MC(V) 表示点集 V 的最大团令 Si=vi, vi+1, ..., vn, mc[i] 表示 MC(Si) 倒着算 mc[i]，那么显然 MC(V)=mc[1] 此外有 mc[i]=mc[i+1] or mc[i]=mc[i+1]+1

```
1 void init(){
2     int i, j;
```

```
3       for (i=1; i<=n; ++i) for (j=1; j<=n; ++j) scanf("%d", &g[i][j]);
4   }
5   void dfs(int size){
6       int i, j, k;
7       if (len[size]==0) {
8           if (size>ans) {
9               ans=size; found=true;
10          }
11          return;
12      }
13      for (k=0; k<len[size] && !found; ++k) {
14          if (size+len[size]-k<=ans) break;
15          i=list[size][k];
16          if (size+mc[i]<=ans) break;
17          for (j=k+1, len[size+1]=0; j<len[size]; ++j)
18          if (g[i][list[size][j]]) list[size+1][len[size+1]++]=list[size][j];
19          dfs(size+1);
20      }
21  }
22  void work(){
23      int i, j;
24      mc[n]=ans=1;
25      for (i=n-1; i; --i) {
26          found=false;
27          len[1]=0;
28          for (j=i+1; j<=n; ++j) if (g[i][j]) list[1][len[1]++]=j;
29          dfs(1);
30          mc[i]=ans;
31      }
32  }
33  void print(){
34      printf("%d\n", ans);
35  }
```

7.12 Dancing Links

```
1   namespace dancing_links {
2       int const N = , M = , G = ;
3
4       struct node {
5           int col, row, left, right, up, down;
6           inline void clear() {
7               col = row = left = right = up = down = 0;
8           }
9       } grid[G];
10
11      int n, m, tot;
12      int cnt[M], head[N], tail[N];
```

```
13
14      inline void prepare() {
15          tot = m + 1;
16          for (int i = 1; i <= n; ++i) {
17              head[i] = tail[i] = 0;
18          }
19          for (int i = 1; i <= m + 1; ++i) {
20              grid[i].col = i;
21              grid[i].left = i - 1;
22              grid[i].right = i + 1;
23              grid[i].up = i;
24              grid[i].down = i;
25              cnt[i] = 0;
26          }
27          grid[1].left = m + 1;
28          grid[m + 1].right = 1;
29      }
30
31      inline void remove(int x) {
32          grid[grid[x].right].left = grid[x].left;
33          grid[grid[x].left].right = grid[x].right;
34          for (int y = grid[x].down; y != x; y = grid[y].down) {
35              for (int z = grid[y].right; z != y; z = grid[z].right) {
36                  cnt[grid[z].col]--;
37                  grid[grid[z].down].up = grid[z].up;
38                  grid[grid[z].up].down = grid[z].down;
39              }
40          }
41      }
42
43      inline void resume(int x) {
44          for (int y = grid[x].up; y != x; y = grid[y].up) {
45              for (int z = grid[y].left; z != y; z = grid[z].left) {
46                  cnt[grid[z].col]++;
47                  grid[grid[z].up].down = z;
48                  grid[grid[z].down].up = z;
49              }
50          }
51          grid[grid[x].right].left = x;
52          grid[grid[x].left].right = x;
53      }
54
55      inline void add(int x, int y) {
56          tot++;
57          cnt[y]++;
58          if (!head[x]) {
59              head[x] = tot;
60          }
61          if (!tail[x]) {
```

```
62              tail[x] = tot;
63          }
64          grid[tot].row = x; grid[tot].col = y;
65          grid[tot].up = grid[y].up; grid[grid[y].up].down = tot;
66          grid[tot].down = y; grid[y].up = tot;
67          grid[tot].left = tail[x]; grid[tail[x]].right = tot;
68          grid[tot].right = head[x]; grid[head[x]].left = tot;
69          tail[x] = tot;
70      }
71
72      inline bool dfs(int dep) {
73          if (grid[m + 1].right == m + 1) {
74              return true;
75          }
76          int x = grid[m + 1].right;
77          for (int i = x; i != m + 1; i = grid[i].right) {
78              if (cnt[i] < cnt[x]) {
79                  x = i;
80              }
81          }
82          if (!cnt[x]) {
83              return false;
84          }
85          remove(x);
86          for (int i = grid[x].down; i != x; i = grid[i].down) {
87              for (int j = grid[i].right; j != i; j = grid[j].right) {
88                  remove(grid[j].col);
89              }
90              if (dfs(dep + 1)) {
91                  return true;
92              }
93              for (int j = grid[i].left; j != i; j = grid[j].left) {
94                  resume(grid[j].col);
95              }
96          }
97          resume(x);
98          return false;
99      }
100
101     inline void clear() {
102         for (int i = 1; i <= tot; ++i) {
103             grid[i].clear();
104         }
105     }
106 }
```

7.13 极大团计数

Bool g[][] 为图的邻接矩阵，图点的标号由 1 至 n。

```
1  void dfs(int size){
2      int i, j, k, t, cnt, best = 0;
3      bool bb;
4      if (ne[size]==ce[size]){
5          if (ce[size]==0) ++ans;
6          return;
7      }
8      for (t=0, i=1; i<=ne[size]; ++i) {
9          for (cnt=0, j=ne[size]+1; j<=ce[size]; ++j)
10         if (!g[list[size][i]][list[size][j]]) ++cnt;
11         if (t==0 || cnt<best) t=i, best=cnt;
12     }
13     if (t && best<=0) return;
14     for (k=ne[size]+1; k<=ce[size]; ++k) {
15         if (t>0){
16             for (i=k; i<=ce[size]; ++i) if (!g[list[size][t]][list[size][i]]) break;
17             swap(list[size][k], list[size][i]);
18         }
19         i=list[size][k];
20         ne[size+1]=ce[size+1]=0;
21         for (j=1; j<k; ++j)if (g[i][list[size][j]]) list[size+1][++ne[size+1]]=list[
               size][j];
22         for (ce[size+1]=ne[size+1], j=k+1; j<=ce[size]; ++j)
23         if (g[i][list[size][j]]) list[size+1][++ce[size+1]]=list[size][j];
24         dfs(size+1);
25         ++ne[size];
26         --best;
27         for (j=k+1, cnt=0; j<=ce[size]; ++j) if (!g[i][list[size][j]]) ++cnt;
28         if (t==0 || cnt<best) t=k, best=cnt;
29         if (t && best<=0) break;
30     }
31 }
32 void work(){
33     int i;
34     ne[0]=0; ce[0]=0;
35     for (i=1; i<=n; ++i) list[0][++ce[0]]=i;
36     ans=0;
37     dfs(0);
38 }
```

Chapter 8

Hints

8.1 积分表

$$\arcsin x \to \frac{1}{\sqrt{1-x^2}}$$

$$\arccos x \to -\frac{1}{\sqrt{1-x^2}}$$

$$\arctan x \to \frac{1}{1+x^2}$$

$$a^x \to \frac{a^x}{\ln a}$$

$$\sin x \to -\cos x$$

$$\cos x \to \sin x$$

$$\tan x \to -\ln\cos x$$

$$\sec x \to \ln\tan(\frac{x}{2}+\frac{\pi}{4})$$

$$\tan^2 x \to \tan x - x$$

$$\csc x \to \ln\tan\frac{x}{2}$$

$$\sin^2 x \to \frac{x}{2} - \frac{1}{2}\sin x\cos x$$

$$\cos^2 x \to \frac{x}{2} + \frac{1}{2}\sin x\cos x$$

$$\sec^2 x \to \tan x$$

$$\frac{1}{\sqrt{a^2-x^2}} \to \arcsin\frac{x}{a}$$

$$\csc^2 x \to -\cot x$$

$$\frac{1}{a^2-x^2}(|x|<|a|) \to \frac{1}{2a}\ln\frac{a+x}{a-x}$$

$$\frac{1}{x^2-a^2}(|x|>|a|) \to \frac{1}{2a}\ln\frac{x-a}{x+a}$$

$$\sqrt{a^2-x^2} \to \frac{x}{2}\sqrt{a^2-x^2}+\frac{a^2}{2}\,\mathtt{arcsin}\,\frac{x}{a}$$

$$\frac{1}{\sqrt{x^2+a^2}} \to \mathtt{ln}(x+\sqrt{a^2+x^2})$$

$$\sqrt{a^2+x^2} \to \frac{x}{2}\sqrt{a^2+x^2}+\frac{a^2}{2}\,\mathtt{ln}(x+\sqrt{a^2+x^2})$$

$$\frac{1}{\sqrt{x^2-a^2}} \to \mathtt{ln}(x+\sqrt{x^2-a^2})$$

$$\sqrt{x^2-a^2} \to \frac{x}{2}\sqrt{x^2-a^2}-\frac{a^2}{2}\,\mathtt{ln}(x+\sqrt{x^2-a^2})$$

$$\frac{1}{x\sqrt{a^2-x^2}} \to -\frac{1}{a}\,\mathtt{ln}\,\frac{a+\sqrt{a^2-x^2}}{x}$$

$$\frac{1}{x\sqrt{x^2-a^2}} \to \frac{1}{a}\,\mathtt{arccos}\,\frac{a}{x}$$

$$\frac{1}{x\sqrt{a^2+x^2}} \to -\frac{1}{a}\,\mathtt{ln}\,\frac{a+\sqrt{a^2+x^2}}{x}$$

$$\frac{1}{\sqrt{2ax-x^2}} \to \mathtt{arccos}(1-\frac{x}{a})$$

$$\frac{x}{ax+b} \to \frac{x}{a}-\frac{b}{a^2}\,\mathtt{ln}(ax+b)$$

$$\sqrt{2ax-x^2} \to \frac{x-a}{2}\sqrt{2ax-x^2}+\frac{a^2}{2}\,\mathtt{arcsin}(\frac{x}{a}-1)$$

$$\frac{1}{x\sqrt{ax+b}}(b<0) \to \frac{2}{\sqrt{-b}}\,\mathtt{arctan}\,\sqrt{\frac{ax+b}{-b}}$$

$$x\sqrt{ax+b} \to \frac{2(3ax-2b)}{15a^2}(ax+b)^{\frac{3}{2}}$$

$$\frac{1}{x\sqrt{ax+b}}(b>0) \to \frac{1}{\sqrt{b}}\,\mathtt{ln}\,\frac{\sqrt{ax+b}-\sqrt{b}}{\sqrt{ax+b}+\sqrt{b}}$$

$$\frac{x}{\sqrt{ax+b}} \to \frac{2(ax-2b)}{3a^2}\sqrt{ax+b}$$

$$\frac{1}{x^2\sqrt{ax+b}} \to -\frac{\sqrt{ax+b}}{bx}-\frac{a}{2b}\int\frac{\mathtt{d}x}{x\sqrt{ax+b}}$$

$$\frac{\sqrt{ax+b}}{x} \to 2\sqrt{ax+b}+b\int\frac{\mathtt{d}x}{x\sqrt{ax+b}}$$

$$\frac{1}{\sqrt{(ax+b)^n}}(n>2) \to \frac{-2}{a(n-2)}\cdot\frac{1}{\sqrt{(ax+b)^{n-2}}}$$

$$\frac{1}{ax^2+c}(a>0,c>0) \to \frac{1}{\sqrt{ac}}\,\mathtt{arctan}\,(x\sqrt{\frac{a}{c}})$$

$$\frac{x}{ax^2+c} \to \frac{1}{2a}\,\mathtt{ln}(ax^2+c)$$

$$\frac{1}{ax^2+c}(a+,c-)\to\frac{1}{2\sqrt{-ac}}\ln\frac{x\sqrt{a}-\sqrt{-c}}{x\sqrt{a}+\sqrt{-c}}$$

$$\frac{1}{x(ax^2+c)}\to\frac{1}{2c}\ln\frac{x^2}{ax^2+c}$$

$$\frac{1}{ax^2+c}(a-,c+)\to\frac{1}{2\sqrt{-ac}}\ln\frac{\sqrt{c}+x\sqrt{-a}}{\sqrt{c}-x\sqrt{-a}}$$

$$x\sqrt{ax^2+c}\to\frac{1}{3a}\sqrt{(ax^2+c)^3}$$

$$\frac{1}{(ax^2+c)^n}(n>1)\to\frac{x}{2c(n-1)(ax^2+c)^{n-1}}+\frac{2n-3}{2c(n-1)}\int\frac{\mathrm{d}x}{(ax^2+c)^{n-1}}$$

$$\frac{x^n}{ax^2+c}(n\neq1)\to\frac{x^{n-1}}{a(n-1)}-\frac{c}{a}\int\frac{x^{n-2}}{ax^2+c}\mathrm{d}x$$

$$\frac{1}{x^2(ax^2+c)}\to\frac{-1}{cx}-\frac{a}{c}\int\frac{\mathrm{d}x}{ax^2+c}$$

$$\frac{1}{x^2(ax^2+c)^n}(n\geq2)\to\frac{1}{c}\int\frac{\mathrm{d}x}{x^2(ax^2+c)^{n-1}}-\frac{a}{c}\int\frac{\mathrm{d}x}{(ax^2+c)^n}$$

$$\sqrt{ax^2+c}(a>0)\to\frac{x}{2}\sqrt{ax^2+c}+\frac{c}{2\sqrt{a}}\ln(x\sqrt{a}+\sqrt{ax^2+c})$$

$$\sqrt{ax^2+c}(a<0)\to\frac{x}{2}\sqrt{ax^2+c}+\frac{c}{2\sqrt{-a}}\arcsin(x\sqrt{\frac{-a}{c}})$$

$$\frac{1}{\sqrt{ax^2+c}}(a>0)\to\frac{1}{\sqrt{a}}\ln(x\sqrt{a}+\sqrt{ax^2+c})$$

$$\frac{1}{\sqrt{ax^2+c}}(a<0)\to\frac{1}{\sqrt{-a}}\arcsin(x\sqrt{-\frac{a}{c}})$$

$$\sin^2ax\to\frac{x}{2}-\frac{1}{4a}\sin2ax$$

$$\cos^2ax\to\frac{x}{2}+\frac{1}{4a}\sin2ax$$

$$\frac{1}{\sin ax}\to\frac{1}{a}\ln\tan\frac{ax}{2}$$

$$\frac{1}{\cos^2ax}\to\frac{1}{a}\tan ax$$

$$\frac{1}{\cos ax}\to\frac{1}{a}\ln\tan(\frac{\pi}{4}+\frac{ax}{2})$$

$$\ln(ax)\to x\ln(ax)-x$$

$$\sin^3ax\to\frac{-1}{a}\cos ax+\frac{1}{3a}\cos^3ax$$

$$\cos^3ax\to\frac{1}{a}\sin ax-\frac{1}{3a}\sin^3ax$$

$$\frac{1}{\sin^2ax}\to-\frac{1}{a}\cot ax$$

$$x\,\mathtt{ln}(ax) \to \frac{x^2}{2}\,\mathtt{ln}(ax) - \frac{x^2}{4}$$

$$\mathtt{cos}\,ax \to \frac{1}{a}\,\mathtt{sin}\,ax$$

$$x^2 e^{ax} \to \frac{e^{ax}}{a^3}(a^2x^2 - 2ax + 2)$$

$$(\mathtt{ln}(ax))^2 \to x(\mathtt{ln}(ax))^2 - 2x\,\mathtt{ln}(ax) + 2x$$

$$x^2\,\mathtt{ln}(ax) \to \frac{x^3}{3}\,\mathtt{ln}(ax) - \frac{x^3}{9}$$

$$x^n\,\mathtt{ln}(ax) \to \frac{x^{n+1}}{n+1}\,\mathtt{ln}(ax) - \frac{x^{n+1}}{(n+1)^2}$$

$$\mathtt{sin}(\mathtt{ln}\,ax) \to \frac{x}{2}[\mathtt{sin}(\mathtt{ln}\,ax) - \mathtt{cos}(\mathtt{ln}\,ax)]$$

$$\mathtt{cos}(\mathtt{ln}\,ax) \to \frac{x}{2}[\mathtt{sin}(\mathtt{ln}\,ax) + \mathtt{cos}(\mathtt{ln}\,ax)]$$

8.2 数学公式

组合公式

- `fibonacci`
  $f_0 = 0, f_1 = 1$
  $f_{n+2}f_n - f_{n+1}^2 = (-1)^{n+1}$
  $f_{-n} = (-1)^{n-1}f_n$
  $f_{n+k} = f_k f_{n+1} + f_{k-1}f_n$
  $gcd(f_m, f_n) = f_{gcd(m,n)}$
  $f_m | f_n^2 \Leftrightarrow n f_n | m$

- $\sum_{k=1}^{n}(2k-1)^2 = \frac{n(4n^2-1)}{3}$
- $\sum_{k=1}^{n} k^3 = (\frac{n(n+1)}{2})^2$
- $\sum_{k=1}^{n}(2k-1)^3 = n^2(2n^2-1)$
- $\sum_{k=1}^{n} k^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$
- $\sum_{k=1}^{n} k^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12}$
- $\sum_{k=1}^{n} k(k+1) = \frac{n(n+1)(n+2)}{3}$
- $\sum_{k=1}^{n} k(k+1)(k+2) = \frac{n(n+1)(n+2)(n+3)}{4}$
- $\sum_{k=1}^{n} k(k+1)(k+2)(k+3) = \frac{n(n+1)(n+2)(n+3)(n+4)}{5}$
- 错排：$D_n = n!(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \ldots + \frac{(-1)^n}{n!}) = (n-1)(D_{n-2} - D_{n-1})$

8.3 平面几何公式

三角形

1. 半周长 $P=(a+b+c)/2$
2. 面积 $S=aH_a/2=ab\sin(C)/2=\sqrt{P(P-a)(P-b)(P-c)}$
3. 中线 $M_a=\sqrt{2(b^2+c^2)-a^2)}/2=\sqrt{b^2+c^2+2bc\cos(A)}/2$
4. 角平分线 $T_a=\sqrt{bc((b+c)^2-a^2)}/(b+c)=2bc\cos(A/2)/(b+c)$
5. 高线 $H_a=b\sin(C)=c\sin(B)=\sqrt{b^2-((a^2+b^2-c^2)/(2a))^2}$
6. 内切圆半径

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

7. 外接圆半径 $R=abc/(4S)=a/(2\sin(A))=b/(2\sin(B))=c/(2\sin(C))$

四边形

$D1,D2$ 为对角线，M 对角线中点连线，A 为对角线夹角

1. $a^2+b^2+c^2+d^2=D1^2+D2^2+4M^2$
2. $S=D1D2\sin(A)/2$
3. 圆内接四边形 $ac+bd=D1D2$
4. 圆内接四边形，P 为半周长 $S=\sqrt{(P-a)(P-b)(P-c)(P-d)}$

正 n 边形

R 为外接圆半径,r 为内切圆半径

1. 中心角 $A=2\pi/n$
2. 内角 $C=(n-2)\pi/n$
3. 边长 $a=2\sqrt{R^2-r^2}=2R\sin(A/2)=2r\tan(A/2)$
4. 面积 $S=nar/2=nr^2\tan(A/2)=nR^2\sin(A)/2=na^2/(4\tan(A/2))$

圆

1. 弧长 $l=rA$
2. 弦长 $a=2\sqrt{2hr-h^2}=2r\sin(A/2)$
3. 弓形高 $h=r-\sqrt{r^2-a^2/4}=r(1-\cos(A/2))=\arctan(A/4)/2$
4. 扇形面积 $S1=rl/2=r^2A/2$
5. 弓形面积 $S2=(rl-a(r-h))/2=r^2(A-\sin(A))/2$

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 $T = S + 2A$

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 $T = S + 2A$

棱台

1. 体积 $V = (A1 + A2 + \sqrt{A1A2})h/3$，$A1, A2$ 为上下底面积，h 为高
2. 正棱台侧面积 $S = (p1 + p2)l/2$，$p1, p2$ 为上下底面周长，l 为斜高
3. 正棱台全面积 $T = S + A1 + A2$

圆柱

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

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 $V = \pi r^2 h/3$

圆台

1. 母线 $l = \sqrt{h^2 + (r1 - r2)^2}$
2. 侧面积 $S = \pi(r1 + r2)l$
3. 全面积 $T = \pi r1(l + r1) + \pi r2(l + r2)$
4. 体积 $V = \pi(r1^2 + r2^2 + r1r2)h/3$

球

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

球台

1. 侧面积 $S = 2\pi rh$
2. 全面积 $T = \pi(2rh + r1^2 + r2^2)$
3. 体积 $V = \pi h(3(r1^2 + r2^2) + h^2)/6$

球扇形

1. 全面积 $T = \pi r(2h + r0)$，h 为球冠高，$r0$ 为球冠底面半径
2. 体积 $V = 2\pi r^2 h/3$

8.4 网络流 Hints

下界:(u, v) 下界为 c: 超级源到 t 建流量为 c, s 到超级汇建流量为 c, (原来的汇到原来的源建无穷, 如果有), 流一遍超级源出边满了就存在可行流.
下界最大流 (有源汇): 上面的搞完从原来的源到原来的汇流一遍
下界最小流 (有源汇): 上面的搞完从原来的汇到原来的源流一遍

8.5 2-SATHints

每对点都选择强连通时 color 较小的

8.6 二分图相关 Hints

二分图最小覆盖集: 从右边的所有没有匹配过的点出发走增广路, 右边所有没有打上记号的点, 加上左边已经有记号的点.
最小覆盖数 = 最大匹配数.

8.7 java_hints

旧

```
1 import java.io.*;
2 import java.util.*;
3 import java.math.*;
4
5 class InputReader {
```

```
 6      BufferedReader buff;
 7      StringTokenizer tokenizer;
 8
 9      InputReader(InputStream stream) {
10          buff = new BufferedReader(new InputStreamReader(stream));
11          tokenizer = null;
12      }
13      boolean hasNext() {
14          while (tokenizer == null || !tokenizer.hasMoreTokens())
15              try {
16                  tokenizer = new StringTokenizer(buff.readLine());
17              }
18              catch (Exception e) {
19                  return false;
20              }
21          return true;
22      }
23      String next() {
24          if (!hasNext())
25              throw new RuntimeException();
26          return tokenizer.nextToken();
27      }
28      int nextInt() { return Integer.parseInt(next()); }
29      long nextLong() { return Long.parseLong(next()); }
30  }
31
32  class Node implements Comparable<Node> {
33      int key;
34      public int compareTo(Node o) {
35          if (key != o.key)
36              return key < o.key ? -1 : 1;
37          return 0;
38      }
39      public boolean equals(Object o) { return false; }
40      public String toString() { return ""; }
41      public int hashCode() { return key; }
42  }
43
44  class MyComparator implements Comparator<Node> {
45      public int compare(Node a, Node b) {
46          if (a.key != b.key)
47              return a.key < b.key ? -1 : 1;
48          return 0;
49      }
50  }
51
52  public class Main {
53      public static void main(String[] args) {
54          new Main().run();
```

```
55      }
56      void run() {
57          PriorityQueue<Integer> Q = new PriorityQueue<Integer>();
58          Q.offer(1); Q.poll(); Q.peek(); Q.size();
59
60          HashMap<Node, Integer> dict = new HashMap<Node, Integer>();
61          dict.entrySet(); dict.put(new Node(), 0); dict.containsKey(new Node());
62          //Map.Entry e = (Map.Entry)it.next(); e.getValue(); e.getKey();
63
64          HashSet<Node> h = new HashSet<Node>();
65          h.contains(new Node()); h.add(new Node()); h.remove(new Node());
66
67          Random rand = new Random();
68          rand.nextInt(); rand.nextDouble();
69
70          int temp = 0;
71          BigInteger a = BigInteger.ZERO, b = new BigInteger("1"), c =
72                   BigInteger.valueOf(2);
73          a.remainder(b); a.modPow(b, c); a.pow(temp); a.intValue();
74          a.isProbablePrime(temp); // 1 - 1 / 2 ^ certainty
75          a.nextProbablePrime();
76
77          Arrays.asList(array);
78          Arrays.sort(array, fromIndex, toIndex, comparator);
79          Arrays.fill(array, fromIndex, toIndex, value);
80          Arrays.binarySearch(array, key, comparator); // found ? index : -
81                  (insertPoint) - 1
82          Arrays.equals(array, array2);
83          Collection.toArray(arrayType[]);
84
85          Collections.copy(dest, src);
86          Collections.fill(collection, value);
87          Collections.max(collection, comparator);
88          Collections.replaceAll(list, oldValue, newValue);
89          Collections.reverse(list);
90          Collections.reverseOrder();
91          Collections.rotate(list, distance); //  ---------->
92          Collections.shuffle(list); // random_shuffle
93      }
94  }
```

新

```
1  import java.io.*;
2  import java.util.*;
3  import java.math.*;
4
5  public class Main {
6      public static void main(String[] args) {
```

```
7           InputStream inputStream = System.in;
8           OutputStream outputStream = System.out;
9           InputReader in = new InputReader(inputStream);
10          PrintWriter out = new PrintWriter(outputStream);
11          Task solver = new Task();
12          solver.solve(1, in, out);
13          out.close();
14      }
15  }
16
17  class Task {
18      public void solve(int testNumber, InputReader in, PrintWriter out) {
19
20      }
21  }
22
23  class InputReader {
24      public BufferedReader reader;
25      public StringTokenizer tokenizer;
26
27      public InputReader(InputStream stream) {
28          reader = new BufferedReader(new InputStreamReader(stream), 32768);
29          tokenizer = null;
30      }
31
32      public String next() {
33          while (tokenizer == null || !tokenizer.hasMoreTokens()) {
34              try {
35                  tokenizer = new StringTokenizer(reader.readLine());
36              } catch (IOException e) {
37                  throw new RuntimeException(e);
38              }
39          }
40          return tokenizer.nextToken();
41      }
42
43      public int nextInt() {
44          return Integer.parseInt(next());
45      }
46
47      public long nextLong() {
48          return Long.parseLong(next());
49      }
50  }
```

8.8 Usage_of_Rope

```
1  #include <ext/rope>
```

```
2 using __gnu_cxx::crope; using __gnu_cxx::rope;
3 a = b.substr(from, len);        // [from, from + len)
4 a = b.substr(from);             // [from, from]
5 b.c_str();                      // might lead to memory leaks
6 b.delete_c_str();               // delete the c_str that created before
7 a.insert(p, str);               // insert str before position p
8 a.erase(i, n);                  // erase [i, i + n)
```