Standrad Code Library

The WaySoFar @ Beihang University

May 13, 2016

E	 录			5.6 5.7	后缀树	
1	Data Structure 1.1 Treap 1.2 序列维护 Treap 1.3 Splay . 1.4 树状数组 1.5 Link-Cut-Tree 1.6 树边分治 1.7 K-d Tree 1.8 2D-Segment-Tree	1 1 2 3 3 4 5 6	6	Oth 6.1 6.2 6.3 6.4 6.5 6.6 6.7 6.8	er emacs Dancing Links unorderedmap 插头 DP 压位 LCS bitset 区间询问 转转转 Time-travel	24
2	2.10 Gomory-Hu-tree	7 7 7 8 8 8 9 9 10 10				
3	3.1 高斯消元 3.2 行列式 3.3 FFT 3.4 Euler 筛 3.5 自适应 simpson 积分 3.6 离散对数 BSGS 3.7 素性测试启发式分解 3.8 线性递推数列	11 11 11 12 12 12 12 13 13				
4	4.1 基础 2D 几何 4.2 直线与圆 4.3 点在多边形内判定 4.4 2D 凸包相关 4.5 半平面交 4.6 圆面积相关 4.7 平面划分	13 13 14 15 15 16 16 17 18				
5	5.1 Ext-KMP 5.2 manacher 5.3 最小表示法 5.4 后缀数组	19 19 19 19 19 20				

Data Structure

1.1 Treap

```
template <class T> struct Treap {
         int nodecnt , prior[M];
3
         int cnt[M] , size[M] , c[M][2];
         T key[M], GCD[M];
         void clear() {
 5
 6
            nodecnt = 1;
             prior[0] = -1 << 30;
             c[0][0] = c[0][1] = 0;
 8
q
             key[0] = GCD[0] = cnt[0] = size[0] = 0;
10
11
         Treap () {
             clear();
12
13
        inline void pushup(int p) {
14
15
             size[p] = size[c[p][0]] + size[c[p][1]] + cnt[p];
             GCD[p] = \_gcd(\_gcd(GCD[c[p][0]], GCD[c[p][1]])
16
             , key[p]);
17
         inline void rotate (int& x , int t) {
18
19
             int y = c[x][t];
             c[x][t] = c[y][!t] , c[y][!t] = x;
20
21
             pushup(x) , pushup(y) , x = y;
22
         inline void newnode(int& p , T w) {
23
24
             p = nodecnt ++;
25
             key[p] = GCD[p] = w, cnt[p] = size[p] = 1;
             prior[p] = rand() << 15 | rand(), c[p][0] = c[p]
26
             ][1] = 0;
27
         void insert(int& p , T w) {
28
             if (!p) {
                 newnode(p , w);
30
31
                 return:
32
             if (key[p] == w)
33
34
                 ++ cnt[p];
35
             else {
                 int t = \text{key}[p] < w;
36
37
                 insert(c[p][t], w);
                 if (prior[c[p][t]] > prior[p])
38
39
                     rotate(p , t);
40
             pushup(p);
41
42
         void erase(int& p , T w) {
43
             if (!p) return;
44
             if (key[p] == w) {
45
                 if (cnt[p] == 1) {
46
                     if (!c[p][0] && !c[p][1])
47
                         p = 0;
49
                     else {
50
                          rotate(p , prior[c[p][0]] < prior[c[p</pre>
                          ][1]]);
51
                          erase(p , w);
52
                 } else
53
54
                       - cnt[p];
55
                 erase(c[p][key[p] < w], w);
56
57
             pushup(p);
58
         T getKth(int p , int K) {
59
             if (K <= size[c[p][0]])</pre>
60
                 return getKth(c[p][0] , K);
61
62
             K = size[c[p][0]] + cnt[p];
             if (K <= 0) return key[p];</pre>
             return getKth(c[p][1] , K);
64
65
         T lower_bound(int p , T w) {
66
             if (!p) return 1 << 30;
67
             if (\text{key}[p] >= w)
68
                 return min(lower_bound(c[p][0] , w) , key[p]);
69
70
71
                 return lower_bound(c[p][1] , w);
72
73
         T range(int p, int l, int r) {
             if (!p || 1 > r) return 0;
74
             if (1 <= key[p] && key[p] <= r) {</pre>
75
```

```
int ans = key[p];
              if (1 == -1 << 30) {
                                                                      77
                  ans = \underline{gcd}(ans , GCD[c[p][0]]);
                                                                      78
                  ans =
                          _gcd(ans , range(c[p][1] , l , r));
                                                                      79
              } else if (r == 1 << 30) {
                                                                      80
                  ans = \underline{gcd}(ans , GCD[c[p][1]]);
                                                                      81
                  ans = \underline{gcd}(ans , range(c[p][0] , 1 , r));
                                                                      82
              } else {
                                                                      83
                  ans = \underline{gcd}(ans , range(c[p][0] , 1 , 1 <<
                                                                      84
                   30));
                          \_gcd(ans , range(c[p][1] , -1 << 30
                                                                      85
                  ans =
                    , r));
                                                                      86
              return ans;
                                                                      87
                                                                      88
         if (r < key[p])
                                                                      89
              return range(c[p][0] , 1 , r);
                                                                      90
         else
                                                                      91
              return range(c[p][1] , 1 , r);
                                                                      92
                                                                      93
    void merge(int& p , int& q) {
                                                                      94
         if (!p) return;
                                                                      95
         merge(c[p][0] , q);
                                                                      96
         merge(c[p][1] , q);
                                                                      97
         insert(q , key[p]);
erase(p , key[p]);
                                                                      98
                                                                      99
                                                                      100
};
```

序列维护 Treap 1.2

return:

```
struct Treap {
    int nodecnt;
                                                                       2
    int L[N] , R[N] , cnt[N];
                                                                       3
    int key[N];
    int Min[N] , add[N] , rev[N];
    bool hey(int A , int B) -
                                                                       6
         return rand() % (cnt[A] + cnt[B]) < cnt[A];</pre>
    int newnode(int val) {
                                                                       9
         ++ nodecnt , L[nodecnt] = R[nodecnt] = 0;
                                                                       10
         cnt[nodecnt] = 1 , Min[nodecnt] = key[nodecnt] =
                                                                       11
         val:
         rev[nodecnt] = add[nodecnt] = 0;
                                                                       12
         return nodecnt;
                                                                       13
                                                                       14
     void pushup(int x) {
                                                                       15
         cnt[x] = 1 , Min[x] = key[x];
                                                                       16
         if (L[x]) cnt[x] += cnt[L[x]] , Min[x] = min(Min[x])
             Min[L[x]])
         if (R[x]) cnt[x] += cnt[R[x]] , Min[x] = min(Min[x
                                                                       18
         ] , Min[R[x]]);
                                                                       19
    void pushdown(int x) {
                                                                       20
         if (rev[x]) {
                                                                       21
              if (L[x]) rev[L[x]] ^= 1 , swap(L[L[x]] , R[L[
                                                                       22
              if (R[x]) rev[R[x]] \stackrel{}{\sim} 1, swap(L[R[x]]), R[R[
                                                                       23
              x]]);
              rev[x] = 0;
                                                                       24
                                                                       25
         if (add[x]) {
                                                                       26
              if (L[x]) add[L[x]] += add[x], Min[L[x]] +=
                                                                       27
              add[x] , key[L[x]] += add[x];
              if(R[x]) add[R[x]] += add[x], Min[R[x]] +=
                                                                       28
              add[x] , key[R[x]] += add[x];;
add[x] = 0;
                                                                       29
         }
                                                                       30
                                                                       31
    void merge(int& p , int x , int y) {
                                                                       32
         if (!x || !y)
              p = x \mid y
                                                                       34
         else if ( hey(x , y) ) // key[x] < key[y]
                                                                       35
              pushdown(x) , merge(R[x] , R[x] , y) , pushup(
                                                                       36
              p = x):
         else
                                                                       37
              pushdown(y) , merge(L[y] , x , L[y]) , pushup(
                                                                       38
                                                                       39
    \textbf{void} \ \texttt{split}(\textbf{int} \ \texttt{p} \ , \ \textbf{int} \& \ \texttt{x} \ , \ \textbf{int} \& \ \texttt{y} \ , \ \textbf{int} \ \texttt{size}) \ \{
                                                                       40
         if (!size) {
                                                                       41
              x = 0 , y = p;
                                                                       42
```

43

```
44
                 pushdown(p);
             if (cnt[L[p]] >= size)
45
46
                 y = p , split(L[p] , x , L[y] , size) , pushup
                 (y);
47
             else
                 x = p , split(R[p] , R[x] , y , <math>size - cnt[L[p]]
48
                 ]]-1) , pushup(x);
49
    };
50
```

1.3 Splay

```
const int N = 500005;
 2
     struct Node {
          Node *ch[2] , *p;
 3
          int size;
 5
          int val , sum , lm , rm , sm;
          int same;
 6
          bool rev;
          Node () {
9
               size = val = sum = 0;
10
               lm = rm = sm = -1e9;
               same = 1 << 30 , rev = 0;
11
12
13
          bool d() {
              return this == p->ch[1];
14
15
16
          void setc(Node *c , int d) {
              ch[d] = c;
17
18
               c \rightarrow p = this;
19
          void setsame(int x) {
20
               same = val = x;
21
22
               sum = val * size;
               lm = rm = sm = max(val , sum);
23
          void reverse() {
25
26
               rev ^= 1
               swap(ch[0] , ch[1]);
27
28
               swap(lm , rm);
29
30
          void pushdown();
31
          void pushup()
32
               size = ch[0] \rightarrow size + ch[1] \rightarrow size + 1;
               sum = ch[0] \rightarrow sum + val + ch[1] \rightarrow sum;
33
34
               lm = max(ch[0] \rightarrow lm , max(ch[0] \rightarrow sum + val , ch
               [0] \rightarrow sum + val + ch[1] \rightarrow lm));
               rm = max(ch[1] \rightarrow rm , max(ch[1] \rightarrow sum + val , ch
35
               [1] \rightarrow sum + val + ch[0] \rightarrow rm));
              sm = max(val , max(ch[0] \rightarrow sm , ch[1] \rightarrow sm));

sm = max(sm , max(ch[0] \rightarrow rm , ch[1] \rightarrow lm) + val);
36
37
               sm = max(sm , ch[0] -> rm + val + ch[1] -> lm);
39
     }Tnull , *null = &Tnull;
40
     Node mem[N] , *C = mem;
     Node* rub[N];
42
43
     int rubsize;
     void Node::pushdown() {
44
45
          if (rev) {
               if (ch[0] != null)
46
                   ch[0]->reverse();
47
               if (ch[1] != null)
48
                   ch[1]->reverse();
49
               rev ^= 1;
50
51
          if (same != 1 << 30) {
52
               if (ch[0] != null)
5.3
                   ch[0]—>setsame(same);
               if (ch[1] != null)
55
                   ch[1]—>setsame(same);
56
               same = 1 << 30;
58
59
60
     Node* newnode(int v) {
          Node *p = rubsize ? rub[— rubsize] : C ++;
61
          p\rightarrow ch[0] = p\rightarrow ch[1] = p\rightarrow p = null;
62
          p\rightarrow size = 1;
63
          p->val = p->sum = p->lm = p->rm = p->sm = v;
64
65
          p—>same = 1 << 30 , p—>rev = 0;
66
          return p;
67
68
     void rotate(Node *t) {
         Node *p = t \rightarrow p;
```

```
int d = t \rightarrow d();
    p->p->setc(t , p->d());
p->setc(t->ch[!d] , d);
                                                                        71
                                                                       72
     t->setc(p , !d);
                                                                       73
     p->pushup();
                                                                        74
                                                                       75
void update(Node *t) {
                                                                       76
     static Node* Stack[N];
                                                                       77
                                                                        78
     int top = 0;
     while (t != null) {
                                                                       79
         Stack[top ++] = t;
                                                                        80
         t = t \rightarrow p;
                                                                        81
                                                                       82
     for (int i = top - 1; i \ge 0; — i)
                                                                       83
         Stack[i]—>pushdown();
                                                                        84
                                                                       85
void splay(Node *t , Node *f = null) {
    update(t);
                                                                        87
     while (t\rightarrow p != f) {
                                                                        88
         if (t\rightarrow p\rightarrow p == f)
                                                                        89
              rotate(t);
                                                                       90
         else {
                                                                        91
              if (t->d() == t->p->d())
                                                                       92
                  rotate(t->p) , rotate(t);
                                                                        93
                                                                        94
                  rotate(t) , rotate(t);
                                                                       95
         }
                                                                       96
                                                                        97
     t->pushup();
                                                                       98
                                                                        99
Node* select(Node *t , int k) {
                                                                        100
     while (1) {
                                                                       101
         t->pushdown();
                                                                        102
         int c = 1 + t \rightarrow ch[0] \rightarrow size;
                                                                        103
         if (k == c)
                                                                        104
              return t;
                                                                        105
         if (k > c)
                                                                        106
              k = c , t = t \rightarrow ch[1];
                                                                        107
         else
                                                                        108
              t = t \rightarrow ch[0];
                                                                        109
    }
                                                                        110
                                                                       111
void split(Node *p , Node *&x , Node *&y , int K) {
                                                                        112
    if (K == 0) {
                                                                        113
         x = null, y = p;
                                                                       114
     } if (K == p->size) {
                                                                        115
         x = p , y = null;
                                                                        116
     } else {
                                                                       117
         y = select(p, K + 1);
                                                                        118
         splay(y);
                                                                       119
         x = y \rightarrow ch[0];
                                                                       120
         y - ch[0] = x - p = null;
                                                                       121
         y->pushup();
                                                                        122
                                                                        123
                                                                       124
\mathbf{void} merge(Node *&p , Node *x , Node *y) {
                                                                        125
     if (x == null)
                                                                        126
         p = y;
                                                                       127
     else if (y == null)
                                                                        128
        p = x;
                                                                        129
     else {
                                                                       130
         x->pushdown();
                                                                        131
         p = select(x , x \rightarrow size);
                                                                        132
         splay(p);
                                                                       1.3.3
         p->setc(y , 1);
                                                                        134
         p—>pushup();
                                                                        135
    }
                                                                        136
                                                                        137
int n , m , a[N];
                                                                        138
Node* build(int 1 , int r) {
                                                                        139
    if (1 > r)
                                                                        140
         return null;
                                                                        141
     int mid = 1 + r >> 1;
                                                                        142
     Node *t = newnode(a[mid]);
                                                                       143
     t\rightarrow setc(build(1, mid - 1), 0);
                                                                        144
     t\rightarrowsetc(build(mid + 1 , r) , 1);
                                                                        145
     t->pushup();
                                                                        146
     return t;
                                                                        147
                                                                        148
void del(Node *p) {
                                                                       149
     if (p == null)
                                                                        150
         return;
                                                                        151
     rub[rubsize ++] = p;
                                                                       152
```

```
153
          del(p\rightarrow ch[0]);
154
          del(p\rightarrow ch[1]);
155
156
      int main() {
          scanf("%d%d" , &n , &m);
157
          for (int i = 1; i <= n; ++ i)
158
               scanf("%d" , &a[i]);
159
          Node *root = build(1 , n);
160
          return 0;
161
162
```

1.4 树状数组

```
int getKth(int k) {
1
         int x = 0 , i;
3
         for (i = 16 ; i >= 0 ; -- i)
             if (x + (1 << i) <= D && c[x + (1 << i)] < k) {
                 x += 1 << i;
                 k = c[x];
6
         return x + 1;
9
    struct CHU_2_BIT {
10
11
         int n:
         LL B[N] , C[N];
12
13
         void init(int size) {
             n = size:
14
15
             memset(B , 0 , n + 1 << 3);
16
             memset(C , 0 , n + 1 << 3);
17
         CHU_2_BIT() {}
18
19
         CHU_2_BIT(int size) {
20
             init(size);
         inline LL _sum(LL* c , int x) {
22
23
             LL res = 0;
             for (; x > 0; x = x \& -x)
                 res += c[x];
25
26
             return res;
27
28
         void add(int 1 , int r , LL w) {
             for (int i = 1; i \le n; i += i \& -i)
29
                 B[i] += w , C[i] += w * 1;
30
31
32
             for (int i = r ; i \le n ; i += i \& -i)
                 B[i] = w , C[i] = w * r;
33
34
35
         LL sum(int 1 , int r) {
             LL res = 0; — 1;
36
             res += (r + 1) * _sum(B , r) - _sum(C , r);
res -= (1 + 1) * _sum(B , 1) - _sum(C , 1);
37
38
39
             return res;
    }T;
41
```

1.5 Link-Cut-Tree

```
const int N = 100005:
1
    const int INF = -1 \ll 30;
 2
    struct Node {
        Node *ch[2] , *p , *fa;
 4
        int size;
        int val , add;
 6
7
        int same;
8
        pair<int , int> mx[2];
        bool rev;
9
10
        Node () {
            mx[0].first = mx[1].first = INF;
11
             size = val = add = 0;
12
13
             same = 1 << 30 , rev = 0;
14
15
        bool d() {
             return this == p->ch[1];
16
17
        void setc(Node *c , int d) {
18
            ch[d] = c;
19
             c->p = this;
20
21
        void addwei(int x) {
22
23
             if (same != 1 << 30)
24
                 same += x;
             add += x , val += x;
25
```

```
mx[0].first += x , mx[1].first += x;
                                                                     27
    void setsame(int x) {
                                                                     28
         same = val = x;
                                                                     29
         add = 0:
                                                                     30
         mx[0] = make_pair(x , size);
                                                                    31
         mx[1] = make_pair(INF , 0);
                                                                    32
                                                                    33
    void reverse() {
                                                                     34
        rev ^= 1;
                                                                    35
         swap(ch[0] , ch[1]);
                                                                    36
                                                                     37
    void pushdown();
                                                                    38
    void pushup() {
                                                                     39
         size = ch[0] - size + ch[1] - size + 1;
                                                                     40
         int j = 0 , k = 0 , l = 0;
                                                                     41
         for (int i = 0; i < 2; ++ i) {
             int x = \max(ch[0]->mx[j].first , ch[1]->mx[1].
                                                                     43
             first), y = 0;
             if (k < 1) x = max(x , val);
if (x == ch[0]->mx[j].first) y += ch[0]->mx[j
                                                                     45
             ++].second;
             if(k < 1 \&\& x == val) ++ y , ++ k;
                                                                     46
             if (x == ch[1] -> mx[1].first) y += ch[1] -> mx[1]
                                                                     47
             ++1.second;
             mx[i] = make_pair(x, y);
                                                                     48
                                                                     49
                                                                     50
}Tnull , *null = &Tnull;
                                                                    51
Node mem[N] , *C = mem;
                                                                    52
void Node::pushdown() {
                                                                    53
    if (rev) {
                                                                    54
         if (ch[0] != null)
             ch[0]->reverse();
                                                                    56
         if (ch[1] != null)
                                                                    57
             ch[1]->reverse();
         rev ^= 1;
                                                                    59
                                                                     60
    if (add) {
                                                                    61
         if (ch[0] != null)
                                                                     62
             ch[0]->addwei(add);
                                                                     63
         if (ch[1] != null)
                                                                    64
             ch[1]—>addwei(add);
                                                                     65
         add = 0;
                                                                    67
    if (same != 1 << 30) {
                                                                     68
         if (ch[0] != null)
                                                                     69
             ch[0]->setsame(same);
                                                                    70
         if (ch[1] != null)
                                                                    71
             ch[1]—>setsame(same);
                                                                     72
         same = 1 << 30;
                                                                    73
    }
                                                                    75
                                                                    76
Node* newnode(int v) {
                                                                    77
    Node *p = C ++;
                                                                     78
    p\to ch[0] = p\to ch[1] = p\to p = p\to fa = null;
                                                                    79
    p->size = 1;
                                                                     80
    p\rightarrow val = v;
                                                                    81
    p->same = 1 << 30 , p->rev = p->add = 0;
                                                                     82
    p\rightarrow mx[0] = make_pair(v, 1);
                                                                    83
    p\rightarrow mx[1] = make_pair(INF, 0);
                                                                     84
    return p;
                                                                     85
                                                                    86
void rotate(Node *t) {
                                                                    87
    Node *p = t \rightarrow p;
                                                                    88
    int d = t\rightarrow d();
                                                                    89
    p->p->setc(t , p->d());
    p->setc(t->ch[!d] , d);
                                                                    91
    t->setc(p , !d);
                                                                    92
    p->pushup();
                                                                    93
    t\rightarrow fa = p\rightarrow fa;
                                                                    94
                                                                    95
void update(Node *t) {
                                                                    96
    static Node* Stack[N];
                                                                    97
    int top = 0;
                                                                    98
    while (t != null) {
                                                                    99
         Stack[top ++] = t;
                                                                     100
         t = t \rightarrow p;
                                                                     101
                                                                    102
    for (int i = top - 1; i \ge 0; — i)
                                                                     103
         Stack[i]->pushdown();
                                                                     104
                                                                    105
```

```
106
      void splay(Node *t , Node *f = null) {
          update(t);
107
108
           while (t->p != f) {
               if (t\rightarrow p\rightarrow p == f)
109
                    rotate(t);
110
111
               else {
                    if (t->d() == t->p->d())
112
113
                         rotate(t->p) , rotate(t);
114
                         rotate(t) , rotate(t);
115
116
               }
117
           t->pushup();
118
119
120
      Node* expose(Node *x) {
          Node *y = null;
121
           while (x != null) {
122
123
               splay(x);
124
               Node *z = x \rightarrow ch[1];
               z\rightarrow p = null;
125
               z\rightarrow fa = x;
126
127
               x->setc(y , 1);
               y\rightarrow fa = null;
128
129
               x->pushup();
130
               y = x, x = x \rightarrow fa;
131
132
           return y;
133
      void setroot(Node *x) {
134
135
           expose(x);
136
           splay(x);
137
           x->reverse();
138
      void link(Node *x , Node *y) {
139
140
           setroot(x);
141
           x\rightarrow fa = y;
          expose(x);
142
143
144
      void cut(Node *x , Node *y) {
145
           setroot(x);
146
           expose(y);
147
          splay(x);
148
           x->setc(null , 1);
149
           x->pushup();
          y\rightarrow fa = y\rightarrow p = null;
150
151
      int n , m , pre[N] , mcnt , ca;
struct edge {
152
153
154
           int x , next;
      }e[N << 1];
155
      bool vis[N];
156
      Node *V[N];
157
      void work() {
158
           printf("Case #%d:\n" , ++ ca);
159
           scanf("%d%d" , &n , &m);
160
161
           C = mem:
162
           for (int i = 1; i <= n; ++ i) {
163
               int x;
               scanf("%d" , &x);
164
165
               V[i] = newnode(x);
166
167
168
           queue<int> Q;
          memset(vis , 0 , sizeof(vis));
169
170
           Q.push(1) , vis[1] = 1;
           while (!Q.empty()) {
171
172
               int x = Q.front(); Q.pop();
173
               for (int i = pre[x] ; ~i ; i = e[i].next) {
                    int y = e[i].x;
174
175
                    if (!vis[y]) {
176
                         V[y] \rightarrow fa = V[x];
                         vis[y] = 1;
177
178
                         Q.push(y);
179
                    }
180
               }
181
          }
      }
182
```

1.6 树边分治

```
int n , m , Q , pre[N] , mcnt;
struct edge {
                                                                  6
    int x , w , next;
}e[N << 2];
int tmp[N];
                                                                  8
bool color[N];
void build(vector<int>& child , int x , int l , int r) {
                                                                  10
    if (1 == r) {
                                                                  11
         int y = e[child[1]].x;
                                                                  12
         e[mcnt] = (edge) \{y, e[child[1]].w, tmp[x]\},
                                                                  13
         tmp[x] = mcnt ++
         e[mcnt] = (edge) \{x, e[child[1]].w, tmp[y]\},
                                                                  14
         tmp[y] = mcnt ++;
                                                                  15
         int mid = 1 + r >> 1;
                                                                  16
         int rt = ++ n ; color[rt] = 1;
                                                                  17
         e[mcnt] = (edge) \{x, 0, tmp[n]\}, tmp[n] = mcnt
        e[mcnt] = (edge) \{n, 0, tmp[x]\}, tmp[x] = mcnt
                                                                  19
         if (1 <= mid) build(child , rt , l , mid);</pre>
                                                                  20
         if (mid < r) build(child , rt , mid + 1 , r);</pre>
                                                                  21
                                                                  22
                                                                  23
void rebuild(int x , int fa) {
                                                                  24
    vector<int> child;
                                                                  25
    for (int i = pre[x] ; ~i ; i = e[i].next) {
                                                                  26
         int y = e[i].x;
                                                                  27
        if (y != fa) {
                                                                  28
             rebuild(y , x);
                                                                  29
             child.push_back(i);
                                                                  30
                                                                  31
                                                                  32
    if (!child.empty())
                                                                  33
        build(child , x , 0 , child.size() - 1);
                                                                  34
int s[N] , size;
                                                                  36
bool f[N];
                                                                  37
pair<int , int> Find(int x , int fa , int cnt) {
                                                                  38
                                                                  39
    s[x] = 1;
    pair<int , int> res = make_pair(1 << 30 , -1);
                                                                  40
    for (int i = pre[x] ; ~i ; i = e[i].next) {
                                                                  41
         int y = e[i].x;
                                                                  42
         if (!f[i >> 1] && y != fa) {
                                                                  43
             res = min(res , Find(y , x , cnt));
                                                                  44
             s[x] += s[y];
                                                                  45
             res = min(res , make_pair(max(s[y] , cnt - s[y
                                                                  46
             ]) , i));
        }
                                                                  47
                                                                  48
    return res;
                                                                  49
pair<int , int> res[N];
                                                                  51
int flag[N];
                                                                  52
vector< pair<int , int> > b[N];
                                                                  53
\label{eq:priority_queue} {\tt priority\_queue<} \ {\tt pair<} \ {\tt int>} \ {\tt > PQ[N << 1]};
                                                                  54
void Getdis(int x , int fa , int d , int id) {
                                                                  55
    b[x].push_back(make_pair(id , d));
                                                                  56
    if (!color[x])
                                                                  57
        PQ[id].push(make_pair(d , x));
                                                                  58
    ++ size:
                                                                  59
    for (int i = pre[x] ; ~i ; i = e[i].next) {
                                                                  60
         if (!f[i >> 1]) {
                                                                  61
             int y = e[i].x;
                                                                  62
             if (y != fa)
                                                                  63
                 Getdis(y , x , d + e[i].w , id);
                                                                  64
        }
                                                                  65
                                                                  67
void divide(int x , int cnt) {
                                                                  68
    if (cnt <= 1) return;</pre>
                                                                  69
    int k = Find(x , 0 , cnt).second;
                                                                  70
    f[k >> 1] = 1;
                                                                  71
                                                                  72
    size = 0 , PQ[k].push(make_pair(-1 << 29 , -1));
                                                                  73
    Getdis(e[k].x , 0 , 0 , k);
                                                                  74
    s[e[k].x] = size;
                                                                  75
                                                                  76
    size = 0 , PQ[k \land 1].push(make_pair(-1 << 29 , -1));
                                                                  77
    Getdis(e[k \land 1].x , 0 , 0 , k \land 1);
                                                                  78
    s[e[k \land 1].x] = size;
                                                                  79
                                                                  80
    res[k \gg 1] = make_pair(PQ[k].top().first + PQ[k \land 1].
```

```
top().first + e[k].w , k >> 1);
 82
 83
          divide(e[k].x , s[e[k].x]);
 84
          divide(e[k \land 1].x , s[e[k \land 1].x]);
 85
 86
     void down(int x) {
          int i = x , j = i << 1 | 1;</pre>
 87
          pair<int , int> t = res[i];
 88
          if (j + 1 < m \& res[j + 1] > res[j])
 89
              ++ j;
 90
          while (j < m && t < res[j]) {
 91
 92
              flag[res[j].second] = i , res[i] = res[j];
              i = j , j = i << 1 | 1;
 93
              if (j + 1 < m \&\& res[j + 1] > res[j])
 94
 95
                   ++ j;
 96
          res[i] = t , flag[t.second] = i;
 97
 98
     void up(int x) {
 99
          int i = x', j = (i + 1 >> 1) - 1;
pair<int , int> t = res[i];
100
101
102
          while (j \ge 0 \& res[j] < t) {
              flag[res[j].second] = i , res[i] = res[j];
103
104
              i = j, j = (i + 1 >> 1) - 1;
105
          res[i] = t , flag[t.second] = i;
106
107
     }
108
     void work() {
109
          int i , j , x , y , z;
110
111
          char str[10];
          scanf("%d",&n);
112
          memset(pre , -1 , sizeof(pre)) , mcnt = n * 6;
113
          for (i = 1 ; i < n ; ++ i) {
114
              scanf("%d%d%d", &x, &y, &z);
115
116
              e[mcnt] = (edge) \{y, z, pre[x]\}, pre[x] = mcnt
              e[mcnt] = (edge) \{x, z, pre[y]\}, pre[y] = mcnt
117
              ++;
118
119
          int cnt = n;
          mcnt = 0;
120
121
          memset(tmp , -1 , sizeof(tmp));
122
          rebuild(1 , 0);
123
          memcpy(pre , tmp , sizeof(tmp));
124
          divide(1 , n);
125
          m = mcnt >> 1;
126
          make_heap(res , res + m);

for (i = 0 ; i < m ; ++ i)
127
128
129
              flag[res[i].second] = i;
130
          scanf("%d",&Q);
131
132
          while (Q -) {
              scanf("%s" , str);
if (*str == 'A') {
133
134
                   if (!cnt)
135
                       puts("They have disappeared.");
136
137
                   else if (cnt == 1)
138
                       puts("0");
                   else
139
140
                       printf("%d\n" , max(0 , res[0].first));
141
              } else {
                   scanf("%d", &x);
142
                   if (color[x])
143
144
                       ++ cnt;
145
                   else
146
147
                  color[x] ^= 1;
148
                   for (i = 0 ; i < b[x].size() ; ++ i) {
149
                       j = b[x][i].first , z = b[x][i].second;
150
                       if (!color[x])
151
                            PQ[j].push(make_pair(z, x));
                       while (~PQ[j].top().second && color[PQ[j].
152
                       top().second])
153
                            PQ[j].pop();
                       res[flag[j >> 1]].first = PQ[j].top().
154
                       first + PQ[j \land 1].top().first + e[j].w;
155
                       down(flag[j >> 1]) , up(flag[j >> 1]);
                  }
156
157
              }
158
          }
159
```

```
160
int main() {
                                                                      161
    work();
                                                                      162
    return 0;
                                                                     163
}
                                                                     164
```

K-d Tree 1.7

```
struct Point3D {
    int x , y , z;
    bool operator < (const Point3D& R) const {</pre>
                                                                3
        if (x != R.x)
            return x < R.x;
        if (y != R.y)
             return y < R.y;
        return z < R.z;
                                                                8
}P[N];
                                                                 10
typedef pair<int , int> Point;
                                                                 11
typedef pair<int , int> Value;
Point a[N];
                                                                13
bool cmpX(const Point& A , const Point& B) {
                                                                 14
    return A < B;
                                                                15
                                                                16
bool cmpY(const Point& A , const Point& B) {
                                                                 17
    return make_pair(A.second , A.first) < make_pair(B.</pre>
                                                                18
    second , B.first);
                                                                 19
inline void add(Value& A , Value B) {
                                                                 20
    if (B.first > A.first)
                                                                 21
        A = B:
                                                                 22
    else if (B.first == A.first)
                                                                23
        A.second += B.second;
                                                                 24
                                                                 25
int cnt , root;
                                                                26
struct Node {
                                                                 27
    Point u , low , high;
                                                                 28
    Value val , mx;
                                                                29
    int c[2];
    Node () {}
                                                                 31
    Node(int K , Point p) {
                                                                32
        u = low = high = p;
                                                                33
        c[0] = c[1] = 0;
                                                                 34
        val = mx = make_pair(-1 << 30, 0);
                                                                 35
                                                                36
    void merge(const Node& R) {
                                                                 37
        low.first = min(low.first , R.low.first);
                                                                 38
        low.second = min(low.second , R.low.second);
                                                                 39
        high.first = max(high.first , R.high.first);
                                                                 40
        high.second = max(high.second , R.high.second);
                                                                 41
        add(mx , R.mx);
                                                                 42
}t[N];
                                                                 44
inline void update(int p) {
                                                                 45
    t[p].mx = t[p].val;
                                                                 46
                                                                 47
    t[p].high = t[p].low = t[p].u;
    if (t[p].c[0])
                                                                 48
        t[p].merge(t[t[p].c[0]]);
                                                                 49
    if (t[p].c[1])
                                                                50
        t[p].merge(t[t[p].c[1]]);
                                                                 51
                                                                52
void build(int& p , int k , int l , int r) {
                                                                53
    p = cnt ++;
                                                                 54
    int mid = 1 + r >> 1:
                                                                55
    nth\_element(a + 1 , a + mid , a + r , k ? cmpY : cmpX)
                                                                 56
    t[p] = Node(k , a[mid]);
                                                                57
    if (1 < mid)
        build(t[p].c[0] , k ^ 1 , l , mid);
                                                                 59
    if (mid + 1 < r)
                                                                 60
        build(t[p].c[1] , k \wedge 1 , mid + 1 , r);
    update(p);
                                                                 62
                                                                 63
LL cnt1 , cnt2;
                                                                 64
void query(int p , int k , const Point& P , Value& res) {
                                                                 65
                                                                 66
    if (!p) return;
    ++ cnt1;
                                                                 67
    if (t[p].high.first <= P.first && t[p].high.second <=</pre>
                                                                 68
        return add(res , t[p].mx);
                                                                 69
    if (t[p].u.first <= P.first && t[p].u.second <= P.</pre>
                                                                70
    second)
        add(res , t[p].val);
                                                                71
```

152

153

78

```
72
          if (k) {
              query(t[p].c[0] , k \land 1 , P , res);
 73
 74
              if (t[p].u.second <= P.second)</pre>
 75
                  query(t[p].c[1] , k \land 1 , P , res);
 76
          } else {
 77
              query(t[p].c[0] , k ^ 1 , P , res);
              if (t[p].u.first <= P.first)</pre>
 78
 79
                  query(t[p].c[1] , k \land 1 , P , res);
 80
 81
 82
     void modify(int p , int k , const Point& P , const Value&
     val) {
          if (!p) return;
 83
          ++ cnt2;
 84
          if (t[p].u == P) {
 85
              t[p].val = val;
 86
 87
           else {
              if (k) {
 88
                   if (P.second <= t[p].u.second)</pre>
 29
 90
                       modify(t[p].c[0], k ^ 1, P, val);
                     (P.second >= t[p].u.second)
 91
 92
                       modify(t[p].c[1] , k ^ 1 , P , val);
 93
              } else
                  if (P.first <= t[p].u.first)</pre>
 94
 95
                       modify(t[p].c[0], k ^ 1, P, val);
                  if (P.first >= t[p].u.first)
 96
 97
                       modify(t[p].c[1], k ^ 1, P, val);
 98
              }
 99
100
          update(p);
101
     }
102
     void work() {
103
         int i , n;
scanf("%d",&n);
104
105
          for (i = 0; i < n; ++ i) {
106
              scanf("%d%d%d",&P[i].x , &P[i].y , &P[i].z);
107
108
              a[i] = make_pair(P[i].y, P[i].z);
109
          sort(P, P + n), sort(a, a + n);
110
          int m = unique(a, a + n) - a;
111
          cnt = 1, root = 0;
112
113
          build(root , 0 , 0 , m);
114
          Value ans(-1 << 30 , 0);
115
116
          for (i = 0 ; i < n ; ++ i) {
117
              Point p = make_pair(P[i].y, P[i].z);
              Value w = make_pair(0, 1);
118
              query(root , 0 , p , w);
119
              ++ w.first , add(ans , w);
120
121
              modify(root , 0 , p , w);
122
          printf("%d %d\n" , ans.first , ans.second & ((1 << 30))
123
            - 1));
124
     void getNearest(int p , int k) {
125
126
       if (!p) return;
127
       if (t[p].vis) {
              LL dis = dist(P , t[p].u);
128
              if (dis < res || (dis == res && t[p].o < t[ret].o)</pre>
129
130
                  res = dis , ret = p;
131
          if (k) {
132
133
              if (cmpY(P , t[p].u)) {
134
                  getNearest(t[p].c[0], k ^ 1);
                  if (sqr(P.second - t[p].u.second) \le res)
135
136
                       getNearest(t[p].c[1], k ^ 1);
137
              } else {
138
                  getNearest(t[p].c[1], k ^ 1);
                  if (sqr(P.second - t[p].u.second) <= res)
139
                       getNearest(t[p].c[0], k ^ 1);
140
141
          } else {
142
143
              if (cmpX(P , t[p].u)) {
                  getNearest(t[p].c[0] , k ^ 1);
if (sqr(P.first - t[p].u.first) <= res)</pre>
144
145
146
                       getNearest(t[p].c[1], k ^ 1);
147
              } else {
                  getNearest(t[p].c[1], k ^ 1);
148
149
                  if (sqr(P.first - t[p].u.first) <= res)</pre>
150
                       getNearest(t[p].c[0], k ^ 1);
151
              }
```

```
}
```

1.8 2D-Segment-Tree

```
\textbf{int} \ \ \text{mx}[\text{N} << 1][\text{N} << 1] \ , \ \ \text{mn}[\text{N} << 1][\text{N} << 1];
pair<int , int> A , B;
int val , LL , RR , Max , Min;
                                                                     3
inline int id(int 1 , int r) {return 1 + r | 1 != r;}
#define MID int mid = (l + r) >> 1
                                                                     6
#define Left 1 , mid
                                                                     7
#define Right mid + 1
                                                                     8
void QUERY(int p , int l , int r) {
                                                                     9
    int q = id(1 , r);
if (B.first <= 1 && r <= B.second) {</pre>
                                                                     10
                                                                     11
         Max = max(Max , mx[p][q]);
                                                                     12
         Min = min(Min , mn[p][q]);
                                                                     13
         return:
                                                                     14
    } MID:
                                                                     15
    if (B.first <= mid)</pre>
                                                                     16
         QUERY(p , Left);
                                                                     17
     if (B.second > mid)
                                                                     18
         QUERY(p , Right);
                                                                     19
                                                                     20
void query(int 1 , int r) {
                                                                     21
    int p = id(1, r);
                                                                     22
     if (A.first <= 1 && r <= A.second) {</pre>
                                                                     23
         QUERY(p , 1 , m);
                                                                     24
         return:
                                                                     25
    } MID;
                                                                     26
    if (A.first <= mid)</pre>
                                                                     27
         query(Left);
                                                                     28
     if (A.second > mid)
                                                                     29
         query(Right);
                                                                     30
void UPDATE(int p , int l , int r) {
                                                                     32
    int q = id(1, r);
                                                                     33
    if (1 == r) {
                                                                     35
         if (p & 1) {
              mx[p][q] = max(mx[LL][q], mx[RR][q]);
                                                                     36
             mn[p][q] = min(mn[LL][q], mn[RR][q]);
                                                                     37
         } else {
                                                                     38
             mx[p][q] = mn[p][q] = val;
                                                                     39
                                                                     40
         return:
                                                                     41
    } MID;
                                                                      42
    if (A.second <= mid)</pre>
                                                                     43
         UPDATE(p , Left);
                                                                      44
                                                                      45
    else
         UPDATE(p , Right);
                                                                     46
     mx[p][q] = max(mx[p][id(Left)], mx[p][id(Right)]);
                                                                      47
                                                                     48
    mn[p][q] = min(mn[p][id(Left)], mn[p][id(Right)]);
                                                                     49
void update(int 1 , int r) {
                                                                     50
    int p = id(1, r);
                                                                     51
    if (1 == r) {
                                                                     52
         UPDATE(p , 1 , m);
                                                                     53
         return;
                                                                     54
     } MID;
                                                                     55
    if (A.first <= mid)</pre>
                                                                     56
         update(Left);
                                                                     57
                                                                     58
         update(Right);
                                                                     59
    LL = id(Left) , RR = id(Right);
                                                                     60
    UPDATE(p , 1 , m);
                                                                     61
                                                                     62
void work()
                                                                     63
                                                                     64
    int i , j;
                                                                     65
    char str[5];
    scanf("%d%d",&n,&m);
                                                                     67
    memset(mx , 0x80 , sizeof(mx));
                                                                     68
    memset(mn , 0x7F , sizeof(mn));
                                                                     69
    for (i = 1 ; i \le n ; ++ i)
                                                                     70
         for (j = 1 ; j \le m ; ++ j) {
                                                                     71
              scanf("%d", &val);
                                                                     72
              A = make_pair(i, j);
                                                                     73
             update(1 , n);
                                                                     74
                                                                     75
    scanf("%d", &Q);
                                                                     76
    while (Q -
                 -) {
                                                                     77
         scanf("½s", str);
```

```
79
             if (*str == 'c') {
                 scanf("%d%d%d",&A.first , &A.second , &val);
80
81
                 update(1 , n);
82
            } else {
                 scanf("%d%d%d%d",&A.first , &B.first , &A.
83
                 second , &B.second);
                 Max = -1 << 30 , Min = 1 << 30;
84
85
                 query(1, n);
                 printf("%d %d\n" , Max , Min);
86
87
            }
88
    }
```

2 Graph Theory

2.1 Tarjan

```
int ncnt , scnt , bel[N] , low[N] , dfn[N];
 1
2
    int f[N];
    stack<int> S;
    void dfs(int x) {
 4
         int i , y;
         low[x] = dfn[x] = ++ ncnt;
         f[x] = 1 , S.push(x);
 8
         for (i = pre[x] ; ~i ; i = e[i].next) {
             y = e[i].x
 9
10
             if (!dfn[y]) {
11
                 dfs(y);
                 low[x] = min(low[x], low[y]);
12
             } else if (f[y])
13
14
                 low[x] = min(low[x], dfn[y]);
15
         if (low[x] == dfn[x]) {
16
             val[scnt] = 0;
17
             do {
18
                 i = S.top() , S.pop() , f[i] = 0;
20
                 bel[i] = scnt , val[scnt] += v[i];
21
             }while (i != x);
22
             ++ scnt;
23
24
25
    int dfn[N] , low[N] , ncnt;
26
27
    stack<int> S;
    int bel[M] , tmp[N];
28
29
    void dfs(int x , int fa) {
30
         dfn[x] = low[x] = ++ ncnt;
         for (int i = pre[x] ; ~i ; i = e[i].next) {
31
32
             int y = e[i].x;
33
             if (!dfn[y]) {
34
                 S.push(i);
                 dfs(y , i ^ 1);
low[x] = min(low[x] , low[y]);
35
36
                 if (low[y] > dfn[x]) {}//(x , y) is bridge
37
                 if (low[y] >= dfn[x]) {
38
39
                      ++ n; int j;
40
                      do {
                            = S.top() , S.pop();
41
42
                          if (tmp[e[j].x] != n)
                              E[m ++] = make_pair(n , e[j].x) ,
43
                              tmp[e[j].x] = n;
44
                          if (tmp[e[j ^ 1].x] != n)
                              E[m ++] = make_pair(n , e[j \land 1].x
45
                               ) , tmp[e[j \land 1].x] = n;
46
                          bel[j >> 1] = n;
47
                     } while (j != i);
48
49
             } else if (i != fa && dfn[y] < dfn[x])</pre>
50
                 S.push(i) , low[x] = min(low[x] , dfn[y]);
51
        }
    }
```

2.2 最小树形图

```
const int INF = 1e9;
const int N = 505;
int n;
int from[N][N + N];
int edge[N][N + N];
int sel[N + N], f[N + N], vis[N + N];
int getf(int x){
```

```
return f[x] == x ? x : f[x] = getf(f[x]);
                                                                   9
void liuzhu(){
                                                                   10
    f[1] = 1;
                                                                   11
    for (int i = 2; i <= n; ++ i) {
                                                                   12
         sel[i] = 1;
                                                                   13
         f[i] = i;
                                                                   14
        for (int j = 1 ; j <= n ; ++ j)
                                                                   15
             if (f[j] != i) {
                                                                   16
                 from[j][i] = i;
                                                                   17
                  if (edge[sel[i]][i] > edge[j][i])
                                                                   18
                      sel[i] = j;
                                                                   19
                                                                   20
                                                                   21
    int limit = n;
                                                                   22
    while(1) {
                                                                   23
        int prelimit = limit;
                                                                   24
        memset(vis , 0 , sizeof(vis));
for (int i = 2 ; i <= prelimit ; ++ i)</pre>
                                                                   25
                                                                   26
             if (f[i] == i && !vis[i]) {
                                                                   27
                 int j = i;
                                                                   28
                 while(j != 1 && !vis[j]) {
                                                                   29
                      vis[j] = i;
                                                                   30
                      j = getf(sel[j]);
                                                                   31
                                                                   32
                 if(j == 1 || vis[j] != i) continue;
                                                                   33
                 vector<int> C;
                                                                   34
                 int k = j;
                                                                   35
                 do {
                                                                   36
                      C.push_back(k);
                                                                   37
                      k = getf(sel[k]);
                                                                   38
                 } while(k != j);
                                                                   39
                  ++limit;
                                                                   40
                 for (int i = 1; i <= n; ++ i) {</pre>
                                                                   41
                      edge[i][limit] = INF;
                                                                   42
                      from[i][limit] = limit;
                                                                   44
                 f[limit] = vis[limit] = limit;
                                                                   45
                 for (int i = 0 ; i < (int)C.size() ; ++ i)</pre>
                                                                   46
                      int x = C[i];
                                                                   47
                      f[x] = limit;
                                                                   48
                      for (int j = 1; j \le n; ++ j) {
                                                                   49
                          if (edge[j][x] == INF)
                                                                   50
                               continue:
                                                                   51
                          if (edge[j][limit] > edge[j][x] -
                                                                   52
                          edge[sel[x]][x]) {
   edge[j][limit] = edge[j][x] -
                                                                   53
                               edge[sel[x]][x];
                               from[j][limit] = x;
                                                                   54
                          }
                                                                   55
                      }
                                                                   57
                 for (int j = 1; j <= n; ++ j)
                                                                   58
                      if (getf(j) == limit)
                                                                   59
                          edge[j][limit] = INF;
                                                                   60
                  sel[limit] = 1;
                                                                   61
                 for (int j = 1; j <= n; ++ j)
                                                                   62
                      if (edge[sel[limit]][limit] > edge[j][
                                                                   63
                      limit])
                          sel[limit] = j;
                                                                   64
                                                                   65
         if (prelimit == limit) break;
                                                                   66
                                                                   67
    for (int i = limit; i > 1; — i)
                                                                   68
         sel[from[sel[i]][i]] = sel[i];
                                                                   69
                                                                   70
```

2.3 全局最小割

```
pair<int , int> find() {
    int s = 0 , t = 0;
    for (int i = 1; i <= n; ++ i) {
                                                                 3
        d[i].w = 0;
        d[i].V.clear();
                                                                 5
        vis[i] = 0;
    for (int i = 1; i <= n; ++ i) {</pre>
                                                                 8
        int x = -1;
        for (int j = 1 ; j <= n ; ++ j)
                                                                 10
             if (!f[j] && !vis[j] && (!~x || d[x] < d[j]))</pre>
                                                                 11
                 x = j;
                                                                 12
        if (!~x) break;
                                                                 13
```

```
14
              vis[x] = 1 , s = t , t = x;
              for (int j = 1; j \le n; ++ j)
15
16
                  if (!f[j] && !vis[j])
17
                       d[j] += g[x][j];
18
19
         res = min(res , d[t]);
         return make_pair(s , t);
20
21
    }
22
    void global_minimum_cut() {
23
24
         memset(f , 0 , sizeof(f));
         for (int i = 1 ; i < n ; ++ i) {
    pair<int , int> t = find();
25
26
27
              int x = t.first , y = t.second;
28
              f[v] = 1;
              for (int i = 1 ; i <= n ; ++ i) {</pre>
29
                  g[x][i] += g[y][i];
30
31
                  g[i][x] += g[i][y];
              }
32
33
         }
    }
34
```

2.4 Hopcorft-Karp

```
int mx[N] , my[N];
1
    queue<int> que;
    int dx[N] , dy[N];
    bool vis[N];
    bool find(int x) {
6
         for (int i = pre[x] ; ~i ; i = e[i].next) {
8
             int y = e[i].x;
             if (!vis[y] && dy[y] == dx[x] + 1) {
9
10
                 vis[y] = 1;
                 if (!~my[y] || find(my[y])) {
11
12
                      mx[x] = y , my[y] = x;
                      return 1;
14
                 }
15
             }
16
17
         return 0;
18
    }
19
    int matching() {
20
21
         memset(mx , -1 , sizeof(mx));
        memset(my , -1 , sizeof(my));
22
23
         int ans = 0;
24
         while (1) {
             bool flag = 0;
25
             while (!que.empty()) que.pop();
26
27
             memset(dx , 0 , sizeof(dx));
28
             memset(dy , 0 , sizeof(dy));
             for (int i = 0; i < n; ++ i)
29
30
                  if (!~mx[i]) que.push(i);
             while (!que.empty()) {
31
32
                 int x = que.front(); que.pop();
                 for (int i = pre[x] ; ~i ; i = e[i].next) {
33
34
                      int y = e[i].x;
                      if (!dy[y]) {
35
36
                          dy[y] = dx[x] + 1 ;
37
                          if (~my[y])
                               que.push(my[y]) , dx[my[y]] = dy[y]
38
                               ] + 1;
39
                          else
                               flag = 1;
40
41
                      }
                 }
42
43
             if (!flag) break;
44
             memset(vis , 0 , sizeof(vis));
for (int i = 0 ; i < n ; ++ i)</pre>
45
46
                  if (!~mx[i] && find(i)) ++ ans;
48
49
         return ans;
    }
50
```

2.5 Dinic

```
int pre[N] , mcnt , s , t;
struct arc {
   int x , f , next;
} e[M];
```

```
void addarc(int x ,int y ,int z) {
    e[mcnt] = (arc) \{y , z , pre[x]\} , pre[x] = mcnt ++; e[mcnt] = (arc) \{x , 0 , pre[y]\} , pre[y] = mcnt ++;
                                                                       6
                                                                       7
\quad \textbf{int} \ d[N] \ , \ cur[N] \ , \ q[N]; \\
                                                                        9
bool BFS() {
                                                                        10
    memset(d, -1, sizeof(d));
                                                                       11
    int top = 0 , bot = -1;
                                                                       12
    q[++ bot] = t , d[t] = 1;
                                                                        13
    while (top != bot + 1) {
                                                                       14
         int x = q[top ++];
                                                                        15
         for (int i = pre[x] ; ~i ;i = e[i].next) {
                                                                       16
              int y = e[i].x;
                                                                       17
              if (!~d[y] && e[i ^ 1].f) {
                  d[y] = d[x] + 1 , q[++ bot] = y;
if (y == s) return 1;
                                                                        19
                                                                       20
                                                                        21
         }
                                                                        22
                                                                       23
    return 0;
                                                                        24
                                                                        25
int DFS(int x , int flow = 1 << 30) {</pre>
                                                                        26
    if (x == t || !flow) return flow;
                                                                       27
    int sum = 0 , u;
                                                                        28
    for (int& i = cur[x] ; ~i ; i = e[i].next) {
                                                                        29
         int y = e[i].x;
                                                                        30
         if (d[x] == d[y] + 1 && (u = DFS(y , min(flow , e[
                                                                        31
         il.f)))) {
              e[i].f = u , e[i ^ 1].f += u;
                                                                        32
              sum += u , flow -= u;
                                                                        33
              if (!flow) break;
                                                                        34
                                                                       35
                                                                        36
    if (!sum) d[x] = -1;
                                                                        37
    return sum:
                                                                       38
int dinic() {
                                                                        40
    int ans = 0;
                                                                        41
    while (BFS()) {
                                                                        42
         memcpy(cur , pre , sizeof(cur));
                                                                        43
         ans += DFS(s);
                                                                        44
                                                                       45
    return ans;
                                                                        46
```

2.6 费用流

```
int S , T , pre[N] , mcnt;
struct arc {
    int x , f , c , next;
} e[M];
void addarc(int x , int y , int z , int c) {
                                                               6
    e[mcnt] = (arc) \{y, z, c, pre[x]\}, pre[x] = mcnt
    e[mcnt] = (arc) \{x, 0, -c, pre[y]\}, pre[y] = mcnt
                                                               8
                                                               10
int maxflow , ans , d[N] , h[N];
                                                                11
bool f[N]:
                                                               12
bool Dijkstra() {
                                                               13
    priority_queue< pair<int , int> > Q;
                                                               14
    memset(d , 0x3f , sizeof(d));
                                                               15
    d[T] = 0; Q.push(make_pair(-d[T] , T));
                                                               16
    while (!Q.empty()) {
                                                                17
        int x = Q.top().second , w = -Q.top().first; Q.pop
                                                               18
        ();
        if (w > d[x]) continue;
                                                                19
        if (x == S) {
                                                               20
            for (int i = 0; i <= T; ++ i) {
                                                                21
                h[i] += d[i];
                                                               22
                                                               23
            return 1;
                                                               24
                                                               25
        for (int i = pre[x] ; ~i ; i = e[i].next) {
                                                               26
            int y = e[i].x, z = e[i \land 1].c + h[x] - h[y];
                                                               27
            if (e[i \land 1].f \&\& d[x] + z < d[y]) {
                                                               28
                 d[y] = d[x] + z;
                                                               29
                 Q.push(make_pair(-d[y], y));
                                                               30
                                                               31
        }
                                                               32
```

33

```
34
         return 0;
35
36
     int dfs(int x , int flow = 1 << 30) {</pre>
         if (x == T) {
37
              maxflow += flow , ans += h[S] * flow;
38
39
              return flow;
         } f[x] = 1; int sum = 0 , u;
40
         for (int i = pre[x] ; ~i ; i = e[i].next) {
41
42
              int y = e[i].x, u;
              if (e[i].f \&\& !f[y] \&\& h[x] == e[i].c + h[y]) {
43
44
                   u = dfs(y , min(flow , e[i].f));
                   e[i].f = u , e[i ^ 1].f += u;
45
                   flow = u, sum += u;
46
47
                   if (!flow) break;
48
              }
49
         return sum;
50
51
52
     void MincostMaxflow() {
         //memset(h, 0, sizeof(h));
queue<int> Q;// 无负权边可选
53
54
55
         memset(f , 0 , sizeof(f));
         memset(h , 0x3f , sizeof(h));
56
         h[T] = 0 , f[T] = 1 , Q.push(T);
57
         while (!Q.empty()) {
58
              int x = Q.front(); Q.pop() , f[x] = 0;
for (int i = pre[x] ; ~i ; i = e[i].next){
59
60
                  int y = e[i].x , z = e[i ^ 1].c;
if (e[i ^ 1].f && h[y] > h[x] + z){
61
62
                       h[y] = h[x] + z;
63
                       if (!f[y]) {
64
                            Q.push(y);
65
                            f[y] = 1;
66
67
                       }
68
                  }
69
              }
70
71
         maxflow = 0 , ans = 0;
72
         while (Dijkstra()) {
73
              do {
74
                   memset(f , 0 , sizeof(f));
              } while (dfs(S));
75
76
         } // while (Dijkstra());
     }
```

2.7 一般图匹配

```
const int N = 505;
2
3
    int S[N], Q[N], *Top = Q , idx;
    int n, m;
    int f[N] , pre[N] , nxt[N] , vis[N];
    vector<int> e[N];
 6
    int getf(int x) {
        return x == f[x] ? x : f[x] = getf(f[x]);
9
10
    int LCA(int x, int y) {
11
        idx ++;
12
        x = getf(x);
        y = getf(y);
13
        while (1) {
14
15
             if (x)
                 if (vis[x] == idx)
16
17
                     return x;
                 vis[x] = idx;
18
19
                 x = getf(pre[nxt[x]]);
20
21
             swap(x, y);
22
23
24
    void blossom(int x, int y, int 1) {
25
        while (getf(x) != 1) {
26
             pre[x] = y;
27
             if(S[nxt[x]] == 1) {
                 *Top ++;
28
29
                 S[*Top = nxt[x]] = 0;
30
             f[x] = f[nxt[x]] = 1;
31
32
             y = nxt[x];
             x = pre[y];
33
34
35
   void match(int x) {
36
```

```
for (int i = 1 ; i <= n ; ++ i)</pre>
                                                                     37
         f[i] = i;
                                                                     38
    memset(S, -1, sizeof(S));
                                                                     39
    memset(Q, 0, sizeof(Q));
                                                                     40
    S[*(Top = Q + 1) = x] = 0;
                                                                     41
    for(int *i = Q + 1; *i; *i++) {
                                                                     42
         for (auto &g : e[*i]) {
                                                                     43
             if (S[g] == -1) {
                                                                     44
                  pre[g] = *i, S[g] = 1;
                                                                     45
                  if (!nxt[g]) {
                                                                     46
                       for (int u = g, v = *i, lst; v; u =
                                                                     47
                       lst, v = pre[u])
                           lst = nxt[v], nxt[v] = u, nxt[u] =
                                                                     48
                      return;
                                                                     49
                                                                     50
                  *Top++, S[*Top = nxt[g]] = 0;
                                                                     51
             } else if(!S[g] && getf(g) != getf(*i)) {
   int 1 = LCA(g, *i);
                                                                     52
                                                                    53
                  blossom(g, *i, 1);
                                                                     54
                  blossom(*i, g, 1);
                                                                    55
                                                                    56
         }
                                                                    57
    }
                                                                    58
                                                                     59
int main() {
                                                                    60
    scanf("%d%d" , &n , &m);
                                                                     61
    for (int i = 0; i < m; ++ i) {
                                                                     62
         int \times , y;
                                                                    63
         scanf("%d%d" , &x , &y);
                                                                     64
         e[x].push_back(y);
                                                                     65
         e[y].push_back(x);
                                                                    66
                                                                     67
    for (int i = 1 ; i <= n ; ++ i) {
                                                                     68
         if (!nxt[i]) {
                                                                     69
             match(i);
                                                                     71
                                                                    72
    int ans = 0;
                                                                    73
    for(int i = 1; i <= n; i++) ans += nxt[i] != 0;</pre>
                                                                     74
    printf("%d\n", ans / 2);
for(int i = 1; i <= n; i++) printf("%d ", nxt[i]);</pre>
                                                                    75
                                                                    76
    putchar('\n');
                                                                    77
                                                                    78
    return 0;
                                                                     79
```

2.8 Kuhn-Munkras

```
\textbf{int} \ g[N][N] \ , \ lx[N] \ , \ ly[N] \ , \ match[N] \ , \ slack[N];
                                                                          1
bool fx[N] , fy[N];
bool find(int x) {
                                                                          .3
     fx[x] = 1;
     for (int y = 0; y < n; ++ y) {
                                                                          5
         if (fy[y]) continue;
                                                                          6
          if (1x[x] + 1y[y] == g[x][y]) {
               fy[y] = 1;
                                                                          8
               if (!~match[y] || find(match[y])) {
                                                                          9
                   match[y] = x;
                                                                          10
                   return 1:
                                                                          11
                                                                           12
          } else {
                                                                          13
              slack[y] = min(slack[y], lx[x] + ly[y] - g[x]
                                                                          14
               ][y]);
         }
                                                                           15
                                                                          16
     return 0;
                                                                          17
                                                                          18
void update() {
                                                                          19
    int delta = 1 << 30;</pre>
                                                                          20
     for (int i = 0 ; i < n ; ++ i)
                                                                          21
          if (!fy[i])
    \label{eq:delta} \mbox{delta = min(delta , slack[i]);} \\ \mbox{for (int } i = 0 \ ; \ i < n \ ; \ ++ \ i) \ \{ \\ \mbox{}
                                                                          23
                                                                          24
          if (fx[i]) lx[i] -= delta;
                                                                          25
         if (fy[i])
                                                                          26
              ly[i] += delta;
                                                                          27
                                                                          28
              slack[i] -= delta;
                                                                          29
                                                                           30
                                                                          31
int Kuhn_Munkras() {
                                                                          32
     for (int i = 0; i < n; ++ i) {
                                                                          33
         match[i] = -1, lx[i] = ly[i] = 0;
                                                                          34
```

```
35
             for (int j = 0; j < n; ++ j) {
                 lx[i] = max(lx[i] , g[i][j]);
36
37
38
39
        for (int i = 0 ; i < n ; ++ i) {</pre>
40
             for (int j = 0; j < n; ++ j)
                 slack[j] = 1 << 30;
41
42
             while (1) {
                 for (int j = 0; j < n; ++ j)
43
                     fx[j] = fy[j] = 0;
44
45
                 if (find(i))
46
                     break:
                 update();
47
48
49
        int ans = 0;
50
        for (int i = 0; i < n; ++ i)
52
             ans += g[match[i]][i];
53
        return ans;
    }
```

2.9 dominator-tree

```
1
    int n , m;
 2
    struct edge {
 3
        int x , next;
 4
    } e[N << 5];
    int mcnt:
    int pre[N] , bre[N] , tree[N];
 6
    int mstamp , mvis[N];
int *prec, *succ;
    vector<int> mord;
    vector<int> buf[N];
    int buf2[N];
11
    int num[N] , fs[N], mins[N] , fa[N] , dom[N], sem[N];;
12
    void dfs(int u) {
        mvis[u] = mstamp;
14
15
         num[u] = mord.size();
16
         mord.push_back(u);
17
         for (int i = succ[u] ; ~i ; i = e[i].next) {
             int v = e[i].x;
18
19
             if (mvis[v] != mstamp) {
20
                 fa[v] = u;
21
                 dfs(v);
             }
22
23
24
    int find(int u) {
25
26
         if (u != fs[u]) {
27
             int v = fs[u];
             fs[u] = find(fs[u]);
28
             if (mins[v] != -1 \&\& num[sem[mins[v]]] < num[sem[
29
             mins[u]]) {
30
                 mins[u] = mins[v];
31
32
33
         return fs[u];
34
35
    void merge(int u, int v) {
36
37
    void mark(int source) { // prec = bre, succ = pre;
38
39
        mord.clear();
40
         ++mstamp;
41
         dfs(source);
42
         for (int i = 0; i < (int)mord.size(); ++i) {</pre>
             int u = mord[i];
43
             fs[u] = u;
44
45
             mins[u] = -1;
             buf2[u] = -1;
46
47
48
         for (int i = (int)mord.size() - 1; i > 0; --i) {
49
             int u = mord[i], p = fa[u];
             sem[u] = p;
50
             for (int j = prec[u]; ~j ; j = e[j].next) {
51
                 int v = e[j].x;
52
                 if (mvis[v] != mstamp)
53
54
                      continue;
                 if (num[v] > num[u]) {
56
                      find(v);
57
                      v = sem[mins[v]];
58
                 if (num[v] < num[sem[u]]) {
59
```

```
sem[u] = v;
              }
                                                                       61
                                                                       62
         buf[sem[u]].push_back(u);
                                                                       63
         mins[u] = u;
                                                                       64
         merge(u, p);
                                                                       65
         while (buf[p].size()) {
                                                                       66
              int v = buf[p].back();
                                                                       67
              buf[p].pop_back();
                                                                       68
              find(v);
                                                                       69
              if (sem[v] == sem[mins[v]]) {
                                                                       70
                   dom[v] = sem[v];
                                                                       71
              } else {
                                                                       72
                   buf2[v] = mins[v];
                                                                       73
                                                                       74
         }
                                                                       75
                                                                       76
    dom[mord[0]] = mord[0];
                                                                       77
    for (int i = 0; i < (int)mord.size(); ++i) {</pre>
                                                                       78
         int u = mord[i];
                                                                       79
         if (~buf2[u]) {
                                                                       80
              dom[u] = dom[buf2[u]];
                                                                       81
                                                                       82
         if (u != source) {
                                                                       83
              //printf("%d dom %d\n" , dom[u] , u);
//e[mcnt] = (edge) {u , tree[dom[u]]};
                                                                       84
                                                                       85
              //tree[dom[u]] = mcnt ++;
                                                                       86
              res[dom[u]] = 1;
                                                                       87
         }
                                                                       88
    }
                                                                       89
}
                                                                       90
```

2.10 Gomory-Hu-tree

```
void divide(int 1 , int r) {
                                                                     1
    if (1 == r) return;
                                                                    2
    random_shuffle(a + 1 , a + r + 1);
    for (int i = 0 ; i < mcnt ; i += 2)</pre>
                                                                     4
         e[i].f = e[i \land 1].f = (e[i].f + e[i \land 1].f) / 2;
                                                                     5
    s = a[1], t = a[r];
    E[m ++] = (edge) \{s, t, -dinic()\};
                                                                    7
    int ns = 0 , nt = 0;
for (int i = 1 ; i <= r ; ++ i)</pre>
                                                                    8
                                                                    9
         if (!~d[a[i]])
                                                                     10
             T[nt ++] = a[i];
                                                                     11
         else
                                                                    12
             S[ns ++] = a[i];
                                                                     13
    for (int i = 0; i < ns; ++ i)
                                                                     14
         a[1 + i] = S[i];
                                                                    15
    for (int i = 0 ; i < nt ; ++ i)</pre>
                                                                    16
         a[1 + ns + i] = T[i];
                                                                    17
    divide(l, l + ns - 1);
                                                                    18
    divide(l + ns, r);
                                                                     20
}
```

2.11 最大团搜索

```
\textbf{int} \ \textbf{n} \ , \ \textbf{mc[N]} \ , \ \textbf{list[N][N]} \ , \ \textbf{len[N]} \ , \ \textbf{ans;}
                                                                                1
bool g[N][N] , found;
void dfs(int size) {
     int i , j , k;
     if (!len[size]) {
                                                                                6
           if (size > ans)
                ans = size , found = 1;
                                                                                8
           return:
     for (k = 0 ; k < len[size] && !found ; ++ k) {
    if (size + len[size] - k <= ans)</pre>
                                                                                11
                                                                                12
                break;
                                                                                13
           i = list[size][k];
                                                                                14
           if (size + mc[i] <= ans)</pre>
                                                                                15
                break;
                                                                                16
           for (j = k + 1, len[size + 1] = 0; j < len[size]
                                                                                17
            ; ++ j)
                if (g[i][list[size][j]])
                                                                                18
                     list[size + 1][len[size + 1] ++] = list[
                                                                                19
                     size][j];
           dfs(size + 1);
                                                                                20
     }
                                                                                21
                                                                                22
void max_cluster() {
                                                                                23
```

```
24
         int i , j;
         mc[n] = ans = 1;
25
26
         for (i = n - 1; i; --i) {
             found = 0 , len[1] = 0;
for (j = i + 1 ; j <= n ; ++ j)
27
28
                  if (g[i][j])
29
                      list[1][len[1] ++] = j;
30
             dfs(1);
31
32
             mc[i] = ans;
33
34
    void work() {
35
         for (int i = 1; i <= n; ++ i)
36
37
             for (int j = 1 ; j \le n ; ++ j)
                 scanf("%d",&g[i][j]);
38
39
         max_cluster();
         cout << ans << endl;
40
41
42
    // 极大团枚举 0(3^{n/3})
43
    int trail_zero(ULL s) {
44
45
         return s ? __builtin_ctzll(s) : 64;
46
    bool BronKerbosch(const vector<ULL> &g, ULL cur, ULL allow
47
         if (allow == 0 && forbid == 0) {
48
             for (int i = 0; i < n; ++ i) {
    printf("%d" , (int) (cur >> i & 1));
49
50
51
             puts("");
52
             return false;
53
54
         if (allow == 0) return false;
         int pivot = trail_zero(allow | forbid);
56
57
         ULL z = allow \& \neg g[pivot];
         for (size_t u = trail_zero(z); u < g.size(); u +=</pre>
         trail_zero(z >> (u + 1)) + 1) {
             if (BronKerbosch(g, cur | (1ULL << u), allow & g[u
59
             ], forbid & g[u])) return true;
             allow ^= 1ULL << u; forbid |= 1ULL << u;
60
61
         return false:
62
63
    //BronKerbosch(g , 0 , (1ULL \ll n) - 1 , 0);
```

3 Mathematics

3.1 高斯消元

```
int rank = 0;
     for (int i = 0 ; i < n ; ++ i) {</pre>
 2
 3
          int pivot = rank;
          for (int j = rank + 1 ; j < m ; ++ j)
    if (fabs(a[j][i]) > fabs(a[pivot][i]))
                     pivot = j;
          if (fabs(a[pivot][i]) < 1e-10)</pre>
                continue;
 8
          for (int j = 0 ; j < n ; ++ j)
 9
                swap(a[rank][j] , a[pivot][j]);
10
          double tmp = a[rank][i];
11
          for (int j = 0 ; j < n ; ++ j)
a[rank][j] /= tmp;
12
13
          for (int k = 0; k < m; ++ k) {
14
                if (k != rank) {
15
                     double times = a[k][i];
16
                     for (int j = 0 ; j < n ; ++ j) {
    a[k][j] -= a[rank][j] * times;</pre>
17
18
19
20
               }
21
22
          ++ rank;
23
     }
```

3.2 行列式

```
for (int k = 1; k \le n; ++ k) {
                A[i][k] = A[j][k] * t;
                                                            8
                 swap(A[i][k], A[j][k]);
                                                            9
                                                            10
            ans = -ans:
                                                            11
                                                            12
                                                            13
    if (!A[i][i])
                                                            14
        return 0;
                                                            15
    ans *= A[i][i];
                                                            16
                                                            17
return ans;
                                                            18
```

3.3 FFT

```
void FFT(Complex P[], int n, int oper) {
    for (int i = 1, j = 0; i < n - 1; i ++) {
    for (int s = n; j \land = s >>= 1, \sim j \& s;);
         if (i < j) {
              swap(P[i], P[j]);
    for (int d = 0; (1 << d) < n; d++) {
   int m = 1 << d, m2 = m * 2;</pre>
                                                                         8
                                                                         9
         double p0 = pi / m * oper;
                                                                         10
         Complex unit_p0(cos(p0) , sin(p0));
                                                                         11
         for (int i = 0; i < n; i += m2) {
                                                                         12
              Complex unit(1 , 0);

for (int j = 0; j < m; j++) {
                                                                         13
                                                                         14
                   Complex &P1 = P[i + j + m], &P2 = P[i + j]
                                                                         15
                   Complex t = unit * P1;
                                                                         16
                   P1 = P2 - t;
                                                                         17
                   P2 = P2 + t;
                                                                         18
                   unit = unit * unit_p0;
                                                                         19
              }
                                                                         20
         }
                                                                         21
                                                                         23
void NTT(int P[], int n, int oper) {
                                                                         24
    for (int i = 1, j = 0; i < n - 1; i++) {
for (int s = n; j \land = s >>= 1, \neg j \& s;);
                                                                         25
                                                                         26
         if (i < j) {
                                                                         27
              swap(P[i], P[j]);
                                                                         28
                                                                         29
    }//998244353 1004535809
                                                                         30
    for (int d = 0; (1 << d) < n; d++) {</pre>
                                                                         31
         int m = 1 << d, m2 = m << 1;</pre>
                                                                         32
         int unit_p0 = power(G , Q - 1 >> d + 1);
                                                                         33
         if (oper == -1)
                                                                         34
              unit_p0 = inverse(unit_p0);
                                                                         35
         for (int i = 0; i < n; i += m2) {</pre>
                                                                         36
              int unit = 1;
                                                                         37
              for (int j = 0; j < m; j++) {
                                                                         38
                   int &P1 = P[i + j + m], &P2 = P[i + j];
int t = (LL)unit * P1 % Q;
                                                                         39
                                                                         40
                   P1 = P2 - t + Q;
                                                                         41
                   if (P1 >= Q) P1 -= Q;
                                                                         42
                   P2 = P2 + t;
                                                                         43
                   if (P2 >= 0) P2 -= 0;
                                                                         44
                   unit = (LL)unit * unit_p0 % Q;
                                                                         45
              }
                                                                         46
         }
                                                                         47
                                                                         48
                                                                         49
void FFT(int P[] , int n , int oper) {
                                                                         50
    if (n == 1) return;
    int m = 0;
                                                                         52
    for (int i = 0 ; i < n ; i += 3) tmp[m ++] = P[i];</pre>
                                                                         53
    for (int i = 1; i < n; i += 3) tmp[m ++] = P[i];</pre>
    for (int i = 2 ; i < n ; i += 3) tmp[m ++] = P[i];</pre>
                                                                         55
    memcpy(P , tmp , n << 2) , m = n / 3;
                                                                         56
    FFT(P , m , oper);
                                                                         57
    FFT(P + m , m , oper);
FFT(P + m + m , m , oper);
                                                                         58
                                                                         59
    int unit_p0 = hash[oper * n];
                                                                         60
    int unit = 1;
                                                                         61
    for (int i = 0, j = 0; i < n; ++ i) {
                                                                         62
         tmp[i] = P[j] + (LL)unit * (P[m + j] + (LL)unit *
                                                                         63
         P[m + m + j] % Q) % Q;
         tmp[i] %= Q;
                                                                         64
         unit = (LL)unit * unit_p0 % Q;
                                                                         65
```

```
66
             if (++ j == m)
67
                 j = 0;
68
69
        memcpy(P, tmp, n << 2);
70
71
    void FWT(int a[] , int len , int oper) {
72
         for (int k = 0 ; 1 << k < len ; ++ k) {
             for (int i = 0 ; i < len ; ++ i) {</pre>
73
                 if (~i >> k & 1) {
74
                     int j = i ^ (1 << k);
75
76
                     int x = (a[i] + Q - a[j]) % Q;
                     int y = (a[i] + a[j]) % Q;
77
                     if (oper == -1) {
78
                          x = (Q - x) \% Q;
79
                          swap(x , y);
80
81
                     a[i] = x;
82
83
                     a[j] = y;
84
                 }
85
            }
86
        }
87
```

3.4 Euler 筛

```
int m , n;
    bool f[N];
     int prime[N] , tot;
     int mu[N] , phi[N];
     void init(int n) {
 5
         int i , j , x;
 6
         mu[1] = 1;
         for (i = 2; i <= n; ++ i) {
 8
 9
              if (!f[i]) {
                   prime[tot ++] = i ;
10
11
                   phi[i] = i - 1, h[i] = 1;
                   mu[i] = -1;
13
              for (j = 0 ; j < tot ; ++ j) {
    x = i * prime[j];</pre>
14
15
                   if (x > n) break;
16
17
                   f[x] = 1;
18
                   if (i % prime[j] == 0) {
19
                        phi[x] = phi[i] * prime[j];
20
                       mu[x] = 0;
                       break:
21
22
                   } else {
                       phi[x] = phi[i] * (prime[j] - 1);
23
                       mu[x] = -mu[i];
24
25
                   }
26
              }
27
28
29
     LL solve(int n , int m) {
30
         LL ans = 0;
31
         if (n > m) swap(n, m);
         for (int i = 1 , x ; i <= n ; i = x + 1) {
    x = min(n / (n / i) , m / (m / i));</pre>
32
33
              ans += (LL) (sum[x] - sum[i - 1])^* (n / i)^* (m / i)^*
34
               i);
35
         return ans:
36
37
     }
```

3.5 自适应 simpson 积分

```
double F(double x) {
         return sqrt(1 + 4 * a * a * x * x);
 2
 3
    double simpson(double a , double b) {   double c = (a + b) * 0.5;   return (F(a) + 4 * F(c) + F(b)) * (b - a) / 6;
 5
 6
     double asr(double a , double b , double eps , double A) {
8
9
         double c = (a + b) * 0.5;
         double L = simpson(a , c) , R = simpson(c , b);
10
         if (fabs(L + R - A) <= 15 * eps)
11
12
              return L + R + (L + R - A) / 15;
         return asr(a , c , eps / 2 , L) + asr(c , b , eps / 2
13
14
   double cal(double L , double R) {
15
```

```
return asr(L , R , 1e-5 , simpson(L , R)); | 16
|}
```

3.6 离散对数 BSGS

```
//S * P^k = T, m = sqrt(S), I = P^m
                                                              1
map<unsigned int , int> hash;
unsigned int E = S;
for (i = 0 ; i < m ; ++ i) {
    if (E == T) {
        `printf("‰\n" , i);
        return;
    if (!hash.count(E))
                                                              9
        hash[E] = i;
    E = E * P;
                                                              11
                                                              12
for (i = 1 ; i < m ; ++ i) {
                                                              13
    T = T * I;
                                                              14
    if (hash.count(T)) {
                                                              15
        printf("%u\n" , i * m + hash[T]);
                                                              16
                                                              17
        return;
                                                               18
                                                              19
puts("poor sisyphus");
                                                               20
```

3.7 素性测试启发式分解

```
inline LL mod_mul(LL a , LL b , LL Q){
    return (a * b - (LL)((long double)a * b / Q) * Q) % Q;
                                                               2
                                                               3
inline LL myrand() {
    return rand() << 30 | rand();
                                                               5
                                                               6
LL mod_exp(LL a , LL x , LL n) {
    LL ret = 1;
                                                               8
    while(x) {
                                                               9
        if(x & 1)
                                                               10
            ret = mod_mul(ret , a , n);
                                                               11
                                                               12
        a = mod_mul(a , a , n) , x >>= 1;
                                                               13
    return ret:
                                                               14
                                                               15
bool Rabin_Miller(LL n) { //素性测试
                                                               16
    LL k = 0, i, j, m, a;
                                                               17
    if (n < 2) return 0;
                                                               18
    if (n == 2) return 1;
                                                               19
    if (~n & 1) return 0;
                                                               20
                                                               21
    m = n - 1;
    while(~m & 1)
                                                               22
        m >>= 1 , ++ k;
    for(i = 0 ; i < 20; ++ i) {
                                                               24
        a = myrand() \% (n - 2) + 2;
                                                               25
        a = mod_exp(a, m, n);
                                                               26
        if (a == 1)
                                                               27
            continue;
                                                               28
        for (j = 0 ; j < k ; ++ j) {
                                                               29
            if (a == n - 1)
                                                               30
                break:
                                                               31
            a = mod_mul(a, a, n);
                                                               32
                                                               33
        if (j < k)
                                                               34
            continue;
                                                               35
        return 0;
                                                               36
                                                               37
    return 1:
                                                               38
                                                               39
inline LL func(LL x , LL n) {
                                                               40
    return (mod_mul(x , x , n) + 1) % n;
                                                               41
LL Pollard(LL n) { //启发式分解
                                                               43
                                                                44
    LL i , x , y , p;
    if (Rabin_Miller(n))
                                                               45
                                                               46
        return n;
    if(~n & 1)
                                                                47
        return 2;
                                                               48
    for(i = 1 ; i < 20 ; ++ i) {
                                                               49
        x = i;
                                                               50
        y = func(x, n);
                                                               51
        p = \underline{gcd(y - x, n)};
                                                               52
        while(p == 1) {
                                                               53
            x = func(x, n);
                                                               54
```

```
y = func(func(y , n) , n);
55
                   p = \underline{gcd((y - x + n) \% n, n) \% n};
56
57
              if(p == 0 || p == n)
58
                   continue;
59
60
              return p;
61
62
     void factor(LL n , vector<int>& ans) {
63
         LL x = Pollard(n);
64
65
         if(x == n) {
              ans.push_back(x);
66
67
              return:
68
         factor(x , ans);
factor(n / x , ans);
69
70
```

3.8 线性递推数列

```
LL n :
     int d[N] , c[N] , t[N] , Deg;
     struct Poly {
 3
         int a[N];
 5
         Poly() {
              memset(a , 0 , sizeof(a));
 8
         int& operator [] (int x) {
              return a[x];
10
     inline void add(int& A , int B) {
12
         A += B;
13
         if (A >= Q)
14
              A -= Q;
15
16
     Poly operator * (Poly& X , Poly& Y) {
17
         int i , j; Poly ans;
18
19
         for (i = 0 ; i < Deg ; ++ i)
              for (j = 0 ; j < Deg ; ++ j)
20
         add(ans[i + j], (LL)X[i] * Y[j] % Q);

for (i = Deg + Deg - 2; i >= Deg; — i) {
21
22
              for (j = 1 ; j <= Deg ; ++ j)
23
                  add(ans[i-j] \ , \ (LL)ans[i] \ * \ c[j] \ \% \ Q);
24
25
              ans[i] = 0;
26
27
         return ans;
28
     void work() {
29
         memset(c , 0 , sizeof(c));
         memset(d , 0 , sizeof(d));
31
32
         Deg = 2;
         d[0] = 3 , d[1] = 4;
         c[2] = 1 , c[1] = 2;
34
35
         /* c 为转移关系 d 为初值 Deg 为阶数*/
36
         //Fi = 2 * Fi-1 + 1 * Fi - 2
//F0 = 3 , F1 = 4 , F2 = 11 , F3 = 26...
37
38
         Poly ans , P;
39
         P[1] = 1 , ans[0] = 1;
40
         while (n) {
41
              if (n & 1)
42
                  ans = ans * P;
43
              P = P * P , n >>= 1;
44
45
         int res = 0;
         for (int i = 0 ; i < Deg ; ++ i) {
    add(res , (LL)d[i] * ans[i] % Q);</pre>
47
48
49
50
         printf("%d\n" , res);
51
```

3.9 拉格朗日插值

```
for (int i = 0 ; i < n ; ++ i)
    val[i] = cal(i);
for (int i = 0 ; i < n ; ++ i) {
    int cur = 0 , nxt = 1;
    memset(f[cur] , 0 , sizeof(f[cur]));
    f[cur][0] = 1;
    for (int j = 0 ; j < n ; ++ j) {
        if (i != j) {</pre>
```

```
x = inverse((i - j + Q) % Q);
              y = (LL)(Q - j) * x % Q;
//printf("%d %d : %d %d\n"
                                                                         10
                                                                         11
              memset(f[nxt] , 0 , sizeof(f[nxt]));
for (int k = 0 ; k < n ; ++ k) {
                                                                        12
                                                                        13
                   if (f[cur][k]) {
                                                                         14
                        f[nxt][k] += (LL)f[cur][k] * y % Q;
                                                                        15
                        f[nxt][k]  %= Q;
                                                                         16
                        f[nxt][k + 1] += (LL)f[cur][k] * x % Q
                                                                         17
                        f[nxt][k + 1] \% = Q;
                                                                         18
                   }
                                                                         19
                                                                        20
              swap(cur , nxt);
                                                                         21
         }
                                                                         22
                                                                        23
     for (int j = 0; j < n; ++ j)
         L[i][j] = f[cur][j];
                                                                         25
                                                                        26
memset(res , 0 , sizeof(res));
                                                                         27
for (int i = 0 ; i < n ; ++ i)
                                                                         28
     for (int x = 0; x < n; ++ x) {
                                                                         29
         res[x] += (LL)L[i][x] * val[i] % Q;
                                                                        30
          res[x] \%= Q;
                                                                         31
                                                                         32
                                                                         33
// 特殊的,一个K阶多项式已知前K+1项求第n项 O(Klogn)
                                                                         34
// 例: 求前n个自然数的K次方和,多项式是K+1阶
                                                                         35
scanf("%d%d" , &n , &K);
                                                                        36
for (int i = 1; i \le K + 1; ++ i) {
                                                                         37
     f[i] = (f[i-1] + power(i, K)) % Q;
                                                                         38
                                                                        39
if (n <= K + 1) {
                                                                         40
    printf("%d\n", f[n]);
                                                                         41
     return:
                                                                         42
int A = 1 , B = 1;
                                                                         44
for (int i = 0; i \le K + 1; ++ i)

A = (LL)A * (n - i) % Q;
                                                                         45
                                                                         46
for (int i = 1; i <= K + 1; ++ i)
B = (LL)B * (Q - i) % Q;
                                                                         47
                                                                         48
int res = 0;
                                                                         49
for (int i = 0 ; i <= K + 1; ++ i) {
  int C = (LL)A * inverse((LL)(n - i) * B % Q) % Q;
  res += (LL)f[i] * C % Q , res %= Q;</pre>
                                                                         50
                                                                         51
                                                                        52
     if (i == K + 1) break;
                                                                         53
     B = (LL)B * (i + 1) % Q * inverse(Q - (K + 1 - i)) % Q
                                                                        54
                                                                         55
printf("%d\n" , res);
                                                                         56
```

4 Geometry

4.1 基础 2D 几何

```
const double eps = 1e-10 , pi = acos(-1.0);
inline int dcmp(double x) {
    return (x > eps) - (x < -eps);
struct Point {
    double x , y;
    Point (double x = 0 , double y = 0) : x(x) , y(y) {}
                                                               8
    void input() {
        scanf("%lf%lf",&x,&y);
                                                               10
                                                               11
    bool operator < (const Point& R) const{</pre>
        if (dcmp(x - R.x) == 0)
                                                               13
            return dcmp(y - R.y) < 0;
                                                               14
        return dcmp(x - R.x) < 0;
                                                               16
    bool operator == (const Point& R) const{
                                                               17
        return dcmp(x - R.x) == 0 \&\& dcmp(y - R.y) == 0;
                                                               18
                                                               19
    Point operator + (const Point& R) const{
                                                               20
        return Point(x + R.x, y + R.y);
                                                               21
                                                               22
    Point operator — (const Point& R) const{
                                                               23
        return Point(x - R.x, y - R.y);
                                                               24
                                                               25
    Point operator * (const double& R) const{
                                                               26
        return Point(x * R , y * R);
                                                               27
```

Point P , V; // P + Vt

double angle;

Line () {}

```
28
        Point operator / (const double& R) const{
29
30
            return Point(x / R , y / R);
31
32
        double operator ^ (const Point& R) const{
            return x * R.y - y * R.x;
33
34
        double operator % (const Point& R) const{
35
            return x * R.x + y * R.y;
36
37
38
        double len() {
            return sqrt(*this % *this);
39
40
41
    };
    // 向量的极角, [-pi, pi)
42
    double Angle(Point V) {
43
        return atan2(V.y , V.x);
45
    // 两个向量的夹角,不分正负[0,pi)
46
    double Angle(Point A , Point B) {
47
        return acos((A % B) / A.len() / B.len());
48
49
    // 逆时针旋转
50
    Point Rotate(Point A , double rad) {
51
        double Sin = sin(rad) , Cos = cos(rad);
52
        return Point(A.x * Cos - A.y * Sin , A.x * Sin + A.y
53
54
    // 向量的单位法向量, 利用旋转得到
55
    Point Normal(Point A) {
56
57
        double L = A.len();
        return Point(-A.y / L , A.x / L);
58
59
    // 直线交点, v和w为两个直线的方向向量,
60
    // 设交点的参数为P+vt,Q+wt,连立方程解t
61
    // 线段,射线对这个t的参数有限制,很好理解。
    Point GetLineIntersection(Point P , Point v , Point Q ,
63
    Point w) {
64
        Point u = P - Q;
        double t1 = (w \wedge u) / (v \wedge w);
65
66
        return P + v * t1;
67
    ·// 点到直线有向距离,这里直线是用两个点表示的
68
    double DistancePointToLine(Point P , Point A , Point B) {
69
70
        Point v = B - A;
71
        return (v \land (P - A)) / v.len();
72
    // 点到线段距离,就是上面的代码判断一下P在AB上投影的位置。
73
    double DistancePointToSegment(Point P , Point A , Point B)
75
        if (A == B) return (P - A).len();
        Point v1 = B - A, v2 = P - A, v3 = P - B;
        if (dcmp(v1 % v2) < 0) return v2.len();</pre>
77
78
        if (dcmp(v1 % v3) > 0) return v3.len();
        return fabs(v1 ^ v2) / v1.len();
79
80
81
    // 返回点在直线上的投影
    Point GetLineProjection(Point P , Point A , Point B) {
82
83
        Point v = B - A;
        return A + v * (v % (P - A) / (v % v));
84
85
86
    // 判断线段是否相交,没有考虑共线的情况。
87
    bool SegmentProperIntersection(Point a1 , Point a2 , Point
     b1 , Point b2) {
88
        double c1 = (a2 - a1) \land (b1 - a1);
89
        double c2 = (a2 - a1) \land (b2 - a1);
        double c3 = (b2 - b1) \land (a1 - b1);
90
        double c4 = (b2 - b1) \land (a2 - b1);
        return dcmp(c1) * dcmp(c2) < 0 && dcmp(c3) * dcmp(c4)
92
93
    // 点是否在线段上,判定方式为到两个端点的方向是否不一致。
94
    bool OnSegment(Point P , Point a1 , Point a2) {
95
        double len = (P - a1).len();
96
        if (dcmp(len) == 0) return true;
97
98
        a1 = a1 - P, a2 = a2 - P;
        return dcmp((a1 ^ a2) / len) == 0 && dcmp(a1 % a2) <=
99
100
```

4.2 直线与圆

```
Line (Point A , Point B) {
        P = A, V = B - A;
        angle = atan2(V.y , V.x);
    bool operator < (const Line& R) const {</pre>
        return angle < R.angle;</pre>
                                                                10
                                                               11
    Point point(double t){
                                                               12
        return P + V * t;
                                                               13
                                                               14
struct Circle {
                                                               16
    Point 0:
                                                               17
    double r;
    Circle () {}
                                                               19
    Circle (Point _0 , double _r) {0 = _0 , r = _r;}
                                                               20
    Point point(double arc) {
                                                                21
        return Point(0.x + cos(arc) * r , 0.y + sin(arc) *
                                                               22
                                                                23
    void input() {
                                                               24
        0.input() , scanf("%lf",&r);
                                                                25
                                                               26
    void print() {
                                                               27
        printf("(\%f,\%f,\%f)\n", 0.x + eps, 0.y + eps, r
                                                               28
        + eps);
                                                               29
    }
};
                                                               30
                                                               31
// 判定直线与圆相交
                                                               32
// 方法为连立直线的参数方程与圆的方程,很好理解
                                                               33
// t1, t2为两个参数, sol为点集。有了参数, 射线线段什么的也
                                                               34
很方便
\textbf{int} \ \ \texttt{getLineCircleIntersection}(\texttt{Line} \ \texttt{L} \ \ , \ \ \textbf{Circle} \ \texttt{C} \ \ , \ \ \textbf{double} \&
                                                                35
t1 , double& t2 , vector<Point>& sol) {
    double a = L.V.x, b = L.P.x - C.0.x, c = L.V.y, d =
                                                                36
     L.P.y - C.O.y;
    double e = a * a + c * c , f = 2 * (a * b + c * d) , g
                                                                37
     = b * b + d * d - C.r * C.r;
    double delta = f * f - 4 * e' * g;
                                                                38
    if (dcmp(delta) < 0) return 0;</pre>
                                                                39
    if (dcmp(delta) == 0) {
                                                                40
        t1 = t2 = -f / (2 * e);
                                                                41
        sol.push_back(L.point(t1));
                                                                42
        return 1:
                                                               43
                                                                44
    t1 = (-f - sqrt(delta)) / (e + e);
                                                                45
    t2 = (-f + sqrt(delta)) / (e + e);
                                                                46
    sol.push_back(L.point(t1)) , sol.push_back(L.point(t2)
    );
    return 2;
                                                                48
                                                                49
// 判定圆和圆之间的关系
                                                               50
// 内含, 内切, 相交, 重合, 外切, 相离
                                                               51
int getCircleCircleIntersection(Circle C1 , Circle C2 ,
                                                               52
vector<Point>& sol) {
    double d = (C1.0 - C2.0).len();
                                                                53
    if (dcmp(d) == 0) { //同心
                                                               54
        if (dcmp(C1.r - C2.r) == 0)//重合
                                                               55
            return -1;
                                                               56
        return 0;//内含
                                                               57
                                                               58
    if (dcmp(C1.r + C2.r - d) < 0) return 0;//相离
                                                               59
    if (dcmp(fabs(C1.r - C2.r) - d) > 0) return 0;//内含
                                                               60
    double a = Angle(C2.0 - C1.0); // acos內可能越界
    double p = (C1.r * C1.r + d * d - C2.r * C2.r) / (2 * double) / (2 * double)
                                                               62
    C1.r * d);
    p = max(-1.0, min(1.0, p));
                                                                63
    double da = acos(p);
                                                                64
    Point P1 = C1.point(a - da), P2 = C1.point(a + da);
                                                                65
    sol.push back(P1);
                                                               66
    if (P1 == P2) return 1; //切
                                                               67
    sol.push_back(P2);
                                                                68
    return 2:
                                                               69
                                                               70
// 过点p到圆C的切线。返回切线条数,sol里为方向向量
                                                               71
int getTangents(Point P, Circle C, vector<Point>& sol) {
                                                               72
    Point u = C.0 - P;
                                                               73
    double dist = u.len();
                                                               74
    if(dist < C.r) return 0;// 园内
                                                               75
```

```
76
         if(dcmp(dist - C.r) == 0) {// p在圆上,只有一条切线
 77
             sol.push_back(Rotate(u, pi/2));
 78
             return 1;
 79
         } else {
             double ang = asin(C.r / dist);
 80
             sol.push_back(Rotate(u, +ang));
 81
             sol.push_back(Rotate(u, -ang));
 82
 83
             return 2:
 84
 85
     //两个圆的公切线,对应切点存在ab里面
 86
     int getTangents(Circle A , Circle B , Point* a , Point* b)
 87
 88
         int cnt = 0;
 89
         if (A.r < B.r)
 90
             swap(A, B), swap(a, b);
         double dist = (A.0 - B.0).len(), dr = A.r - B.r, sr
 91
         = A.r + B.r:
 92
         if (dcmp(dist - dr) < 0) // 内含
             return 0;
 93
         double base = Angle(B.0 - A.0);
 94
 95
         if (dcmp(dist) == 0 \&\& dcmp(A.r - B.r) == 0)
             return -1;//重合
 96
         if (dcmp(dist - dr) == 0) {//内切
 97
 98
             a[cnt] = A.point(base);
             b[cnt] = B.point(base);
 99
100
             return 1:
101
         double ang = acos(dr / dist);//非上述情况,两条外公切
102
103
         a[cnt] = A.point(base + ang) , b[cnt] = B.point(base +
          ang) , ++ cnt;
         a[cnt] = A.point(base - ang), b[cnt] = B.point(base -
104
          ang) , ++ cnt;
         if (dcmp(dist - sr) == 0) {// 外切,中间一条内公切线
105
             a[cnt] = A.point(base) , b[cnt] = B.point(pi +
106
             base) , ++ cnt;
107
         } else if (dcmp(dist - sr) > 0) {
             ang = acos(sr / dist);//相离,两条内公切线
108
             a[cnt] = A.point(base + ang) , b[cnt] = B.point(pi
109
              + base + ang) , ++ cnt;
             a[cnt] = A.point(base - ang) , b[cnt] = B.point(pi
110
              + base - ang) , ++ cnt;
111
112
         return cnt:
113
114
     // 外接圆,三根中线交点
     Circle CircumscribedCircle(Point A , Point B , Point C) {
115
         Point D = (B + C) / 2, d = Normal(B - C);
116
         Point E = (A + C) / 2, e = Normal(A - C);
117
         Point P = GetLineIntersection(D , d , E , e);
118
         return Circle(P , (C - P).len());
119
120
     // 内接圆, 黑科技
121
122
     Circle InscribedCircle(Point A , Point B , Point C) {
123
         double a = (B - C).len(), b = (A - C).len(), c = (A - C).len()
          - B).len();
         Point P = (A * a + B * b + C * c) / (a + b + c);
124
         return Circle(P , fabs(DistancePointToLine(P , A , B))
125
     }
126
```

4.3 点在多边形内判定

```
for (int i = 0; i < n; ++ i)
1
        if (OnSegment(P , p[i] , p[i + 1]))
3
            return 0:
    int res = 0;
    for (int i = 0; i < n; ++ i) {
6
        Point a = p[i] , b = p[i + 1];
7
        if (a.y > b.y) swap(a, b);
        if (dcmp((a - P) \land (b - P)) < 0 \&\& dcmp(a.y - P.y) < 0
         && dcmp(b.y - P.y) >= 0)
            res ^= 1;
9
10
   return res;
11
```

4.4 2D 凸包相关

```
inline LL OnLeft(Point P , Point A , Point B) {
   return (B - A) ^ (P - A);
}
```

```
/****** Naive 凸包 2.0 O(n+m) *******/
int top = 0;
                                                               5
for (int i = 0; i < n; ++ i) {
                                                                6
    while (top > 1 \&\& OnLeft(p[i], s[top - 2], s[top -
    1]) <= 0) {
                                                               8
         top;
    s[top ++] = p[i];
                                                                10
                                                                11
int tmp = top;
                                                               12
for (int i = n - 2; i \ge 0; — i) {
                                                                1.3
    while (top > tmp && OnLeft(p[i], s[top - 2], s[top -
     1]) <= 0) {
         top;
                                                                15
                                                               16
    s[top ++] = p[i];
                                                               17
                                                                18
if (n > 1)
                                                                19
                                                               20
21
Vec.clear();
                                                                22
Point cur = a[0] + b[0];
                                                               23
for (int i = 0 , j = 0 ; i < n || j < m ; ) {</pre>
                                                                24
    if (i < n && (j == m || ((a[i + 1] - a[i]) ^ (b[j + 1])
                                                                25
      - b[j])) >= 0)) {
        cur = cur + a[i + 1] - a[i];
                                                                26
        ++ i:
                                                                27
    } else {
                                                                28
        cur = cur + b[j + 1] - b[j];
                                                               29
                                                               30
        ++ j;
                                                               31
    Vec.push_back(make_pair(cur , 1));
                                                               32
                                                                33
 ***** 点在凸多边形内判定 0(logn) ******/
                                                                34
bool InConvex(Point q) {
                                                                35
    if (OnLeft(q , p[0] , p[1]) < 0 || OnLeft(q , p[0] , p
    [n-1]) > 0)
        return 0;
                                                                37
    int l = 2, r = n - 1;
                                                                38
    while (1 < r) {
                                                                39
        int mid = 1 + r >> 1;
                                                                40
        \textbf{if} \ (\texttt{OnLeft}(\texttt{q} \ , \ \texttt{p[0]} \ , \ \texttt{p[mid]}) \mathrel{<=} \texttt{0}) \ \{
                                                                41
            r = mid:
                                                                42
        } else {
                                                                43
            l = mid + 1;
                                                                44
                                                                45
                                                                46
    return OnLeft(q , p[r-1] , p[r]) >= 0;
                                                                47
                                                                48
 ****** 点到凸多边形的切线 0(logn) ******/
                                                                49
#define above(b , c) (OnLeft(b , q , c) > 0)
                                                               50
#define below(b , c) (OnLeft(b , q , c) < 0)
int getRtangent(Point q) { // find max
                                                               52
    int ret = 0;
                                                               53
    int l = 1, r = n - 1;
                                                               54
    while (1 <= r) {
                                                               55
        int dnl = above(p[l], p[l + 1]);
                                                               56
        int mid = 1 + r \gg 1;
                                                               57
        int dnm = above(p[mid] , p[mid + 1]);
                                                               58
        if (dnm) {
                                                                59
            if (above(p[mid], p[ret])) {
                                                               60
                 ret = mid;
                                                                61
                                                                62
                                                               63
        if (dnl) {
                                                                64
            if (above(p[1], p[ret])) {
                                                                65
                ret = 1;
                                                                66
                                                                67
            if (dnm && above(p[mid] , p[l])) {
                                                                68
                 r = mid - 1;
                                                                69
            } else {
                                                                70
                1 = mid + 1:
                                                                71
                                                               72
        } else {
                                                               73
            if (!dnm && above(p[mid] , p[1])) {
                                                               74
                 l = mid + 1;
                                                                75
            } else {
                                                               76
                 r = mid - 1;
                                                               77
                                                                78
        }
                                                               79
                                                               80
    return ret;
                                                               81
```

82

```
83
     int getLtangent(Point q) { // find min
          int ret = 0;
 84
 85
          int 1 = 1 , r = n - 1;
          while (1 <= r) {
 86
              int dnl = below(p[l], p[l-1]);
 87
              int mid = 1 + r + 1 >> 1;
 88
              int dnm = below(p[mid] , p[mid - 1]);
 89
 90
              if (dnm) {
                  if (below(p[mid], p[ret])) {
 91
 92
                      ret = mid;
 93
 94
              if (dnl) {
 95
                  if (below(p[l], p[ret])) {
 96
 97
                      ret = 1;
 98
                  if (dnm && below(p[mid] , p[1])) {
 99
                      l = mid + 1;
100
101
                  } else {
102
                      r = mid - 1;
103
104
              } else {
                  if (!dnm && below(p[mid] , p[l])) {
105
106
                      r = mid - 1;
107
                  } else {
                      l = mid + 1;
108
109
110
              }
         }
111
          return ret;
112
113
       ***** 直线对凸多边形的交点 O(logn) *****/
114
     double arc[N] , sum[N];
     void init() {
116
         for (int i = 0; i < n; ++ i) {
117
118
             p[i + n] = p[i];
          p[n + n] = p[0];
119
120
          for (int i = 0 ; i < n + n ; ++ i) {</pre>
121
              sum[i + 1] = sum[i] + (p[i] ^ p[i + 1]);
122
123
          for (int i = 0 ; i < n ; ++ i) {</pre>
              int j = (i + 1) \% n;
124
125
              arc[i] = atan2(p[j].y - p[i].y , p[j].x - p[i].x);
126
              if (i && arc[i] < arc[i - 1]) {
                  arc[i] += pi + pi;
127
128
129
         }
130
     int getseg(Point P , Point V , int l , int r) {
131
132
            1:
         while (1 < r) {
133
              int mid = 1 + r + 1 >> 1;
134
              if ((V \land (p[mid] - P)) < 0) {
135
136
                  1 = mid;
137
              } else {
                  r = mid - 1;
138
139
140
141
         return 1;
142
     void work(Point A , Point B) {
143
144
          if (B < A) {
145
              swap(A , B);
146
147
          double al = atan2(B.y - A.y , B.x - A.x);
148
          if (al < arc[0]) al += pi + pi;</pre>
149
          int Left = (lower\_bound(arc , arc + n , al) - arc) % n
          double ar = atan2(A.y - B.y , A.x - B.x);
150
151
          if (ar < arc[0]) ar += pi + pi;
152
          int Right = lower_bound(arc , arc + n , ar) - arc;
          int down = getseg(A , B - A , Left , Right);
153
154
          int up = getseg(B , A - B , Right , Left + n);
          if (down < Left || up < Right) {</pre>
155
              puts("0.000000");
156
157
          } else {
              Point D = GetLineIntersection(A , B - A , p[down]
158
               p[down + 1] - p[down]);
              Point U = GetLineIntersection(B , A - B , p[up] ,
159
              p[up + 1] - p[up]);
              //printf("%f %f / %f %f\n" , D.x , D.y , U.x , U.y
160
161
              double area = (D \land p[down + 1]) + (sum[up] - sum[
```

```
down + 1]) + (p[up] ^ U) + (U ^ D);
printf("%.6f\n" , min(sum[n] — area , area) / 2); 162
}
}
```

4.5 半平面交

```
typedef vector<Point> Polygon;
//用有向直线AB的左半平面切割 O(n)
                                                                  3
Polygon CutPolygon(const Polygon& poly , Point A , Point B
) {
    Polygon newpoly;
                                                                 5
    int n = poly.size();
                                                                 6
    for (int i = 0; i < n; ++ i) {
        const Point &C = poly[i] , &D = poly[(i + 1) % n];
                                                                 8
         if (dcmp((B - A) \land (C - A)) >= 0)
             newpoly.push_back(C);
                                                                 10
         if (dcmp((B - A) \land (C - D)) != 0) {
                                                                 11
             double t = ((B - A) \land (C - A)) / ((D - C) \land (B))
                                                                  12
              — A)):
                                                                 13
             if (dcmp(t) > 0 \&\& dcmp(t - 1) < 0)
                 newpoly.push_back(C + (D - C) * t);
                                                                 14
        }
                                                                 15
    return newpoly;
                                                                 17
                                                                 18
inline bool Onleft(Line L , Point P) {
                                                                 20
    return (L.V \land (P - L.P)) > 0;
                                                                 21
Point GetLineIntersection(Line A , Line B) {
                                                                 23
    Point u = A.P - B.P;
                                                                  24
    double t = (B.V \wedge u) / (A.V \wedge B.V);
                                                                 25
    return A.point(t);
                                                                 26
                                                                 27
Point p[N];
                                                                 28
Line q[N];
                                                                  29
int HalfPlaneIntersection(Line* L , int n , Point* Poly) {
                                                                 30
    sort(L, L + n);
                                                                 31
    int top = 0 , bot = 0;
                                                                  32
                                                                 33
    q[0] = L[0];
    for (int i = 1; i < n; ++ i) {
                                                                 34
        while (top < bot && !Onleft(L[i], p[bot - 1])) -
                                                                  35
        while (top < bot && !Onleft(L[i] , p[top])) ++ top</pre>
                                                                  36
        q[++ bot] = L[i];
                                                                  37
         if (dcmp(L[i].V \land q[bot - 1].V) == 0) {
                                                                  38
               - bot:
                                                                 39
             if (Onleft(q[bot] , L[i].P))
                                                                 40
                 q[bot] = L[i];
                                                                  41
                                                                 42
         if (top < bot)</pre>
                                                                 43
             p[bot - 1] = GetLineIntersection(q[bot - 1]),
             q[bot]);
                                                                  45
    while (top < bot && !Onleft(q[top], p[bot - 1])) —
                                                                  46
    bot
    if (bot - top <= 1) return 0;</pre>
                                                                  47
    p[bot] = GetLineIntersection(q[bot] , q[top]);
                                                                 48
    int m = 0;
                                                                  49
    for (int i = top ; i <= bot ; ++ i) Poly[m ++] = p[i];</pre>
    return m:
                                                                 51
                                                                  52
```

4.6 圆面积相关

```
/*****圆和多边形求交****/
double sector_area(Point A , Point B , double R) {
    double theta = Angle(A) - Angle(B);
    while (theta < 0) theta += pi + pi;</pre>
    while (theta >= pi + pi) theta -= pi + pi;
    theta = min(theta , pi + pi - theta);
    return R * R * theta;
\frac{1}{a[n]} = a[0]
double cal(double R) {
    double area = 0;
                                                               10
    for (int i = 0 ; i < n ; ++ i) {</pre>
                                                                11
        double t1 = 0 , t2 = 0 , delta;
                                                               12
        Line L = Line(a[i] , a[i + 1]);
                                                               13
        int cnt = getLineCircleIntersection(L , Circle(
                                                                14
        Point(0 , 0) , R) , t1 , t2);
```

```
15
             Point X = L.point(t1), Y = L.point(t2);
             16
             i + 1].len() - R) <= 0;
17
             if (f1 && f2)
                 delta = fabs(a[i] \wedge a[i + 1]);
18
             else if (!f1 && f2) {
19
                 delta = sector\_area(a[i] , X , R) + fabs(X ^ a
20
                 [i + 1]);
             } else if (f1 && !f2) {
21
                 delta = fabs(a[i] \land Y) + sector\_area(Y , a[i +
22
23
             } else {
                 if (cnt > 1 && 0 < t1 && t1 < 1 && 0 < t2 &&
24
                 t2 < 1) {
                     delta = sector\_area(a[i] , X , R) +
25
                      sector\_area(Y , a[i + 1] , R) + fabs(X ^ Y)
                 } else {
26
27
                     delta = sector\_area(a[i], a[i + 1], R);
28
29
30
             area += delta * dcmp(a[i] ^ a[i + 1]);
31
         return area / 2:
32
33
     '********* 圆 交/ 弁******/
34
    void getarea() { // 计算圆并的重心,必要的时候可以去除有包
35
     含关系的圆
        for (int i = 0; i < n; ++ i) {
36
37
             vector< pair<double , int> > Vec;
38
             int cnt = 1;
             Vec.push_back({0 , 0});
Vec.push_back({2 * pi , 0});
39
40
             for (int j = 0; j < n; ++ j) {
    double dist = (c[j].0 - c[i].0).len();</pre>
41
42
                 if (dcmp(dist) == 0 \&\& dcmp(c[i].r - c[j].r)
                 == 0) {
                     if (i < j) {
44
45
                          ++ cnt;
46
47
                     continue;
48
49
                 if (dcmp(dist - c[j].r - c[i].r) >= 0) {
50
51
                 if (dcmp(dist + c[j].r - c[i].r) <= 0) { // j}
                 in i
53
                     continue:
54
                 if (dcmp(dist + c[i].r - c[j].r) <= 0) { // i}
55
                 in j
                     ++ cnt;
                     continue;
57
58
                 double an = atan2(c[j].0.y - c[i].0.y, c[j].0
59
                 .x - c[i].0.x);
                 double p = (c[i].r * c[i].r + dist * dist - c[
60
                 j].r * c[j].r) / (2 * c[i].r * dist);
                 double da = acos(max(-1.0 , min(1.0 , p)));
61
                 double L = an - da, R = an + da;
63
                 //printf("%d : %f %f\n" , j , L , R);
64
65
                 if (L < 0) L += 2 * pi;
                 if (R < 0) R += 2 * pi;
66
                 if (L >= 2 * pi) L -= 2 * pi;
67
68
                 if (R >= 2 * pi) R -= 2 * pi;
                    (L < R) {
69
                     Vec.push_back({L , 1});
                     Vec.push_back(\{R, -1\});
71
72
                 } else {
                     Vec.push\_back({0 , 1});
73
                     Vec.push_back(\{R, -1\});
74
75
                     Vec.push_back({L , 1});
                     Vec.push_back(\{2 * pi', -1\});
76
77
78
             sort(Vec.begin() , Vec.end());
79
             for (int j = 0; j + 1 < Vec.size(); ++ j) {
    //printf("%d : %d %f\n" , j , cnt , Vec[j].</pre>
80
81
                 first);
82
                 cnt += Vec[j].second;
                 if (cnt == 1) {
83
84
                     double delta = Vec[j + 1].first - Vec[j].
```

```
first;
                if (dcmp(delta) <= 0)</pre>
                                                                          85
                     continue;
                                                                          86
                double SIN = sin(delta / 2);
                                                                          87
               Point W = Point(0 , 4 * c[i].r * SIN * SIN
                                                                          88
               * SIN / (3 * (delta - sin(delta))));

W = Rotate(W, (Vec[j + 1].first + Vec[j].

first - pi) / 2) + c[i].0;
                                                                          89
                double area = c[i].r * c[i].r * (delta -
                                                                          90
                sin(delta));
                sx -= area * W.x;
                                                                          91
               sy -= area * W.y;
                                                                          92
               s -= area:
                                                                          93
               Point A = c[i].point(Vec[j].first) , B = c
                                                                          95
                [i].point(Vec[j + 1].first);
                area = (A \wedge B);
                                                                          96
               sx = area * (A.x + B.x) / 3;

sy = area * (A.y + B.y) / 3;
                                                                          97
                                                                          98
                s = area;
                                                                          99
          }
                                                                          100
     }
                                                                          101
}
                                                                          102
                                                                          103
```

平面划分 4.7

```
void work() {
    scanf("%d" , &n);
    for (int i = 0 ; i < n ; ++ i) {</pre>
        L[i].input();
        P[i] = L[i];
    int m = n;
    for (int i = 0; i + 1 < n; ++ i)
                                                                8
        for (int j = i + 1; j + 1 < n; ++ j) {
            if (dcmp((P[i + 1] - P[i]) ^ (P[j + 1] - P[j])
                                                                 10
                 P[m ++] = GetLineIntersection(P[i] , P[i +
                  1] - P[i] , P[j] , P[j + 1] - P[j]);
                                                                 12
    sort(P, P + m);
                                                                 13
    m = unique(P, P + m) - P;
                                                                 14
    memset(pre , -1 , sizeof(pre));
                                                                 15
    set< pair<int , int> > Hash;
                                                                16
    for (int i = 0; i + 1 < n; ++ i) {
                                                                17
        vector< pair <Point , int> > V;
                                                                18
        for (int j = 0 ; j < m ; ++ j)</pre>
                                                                19
            if (OnSegment(P[j] , L[i] , L[i + 1]))
                                                                20
                 V.push_back(make_pair(P[j] , j));
                                                                21
        sort(V.begin() , V.end());
                                                                22
        for (int j = 0 ; j + 1 < V.size() ; ++ j) {</pre>
                                                                 23
            int x = V[j].second, y = V[j + 1].second;
                                                                 24
            if (!Hash.count(make_pair(x , y))) {
                                                                25
                Hash.insert(make_pair(x , y));
                                                                26
                 e[mcnt] = (edge) \{y, pre[x]\}, pre[x] =
                                                                27
                 mcnt ++;
                                                                 28
            if (!Hash.count(make_pair(y , x))) {
                                                                29
                Hash.insert(make_pair(y , x));
                                                                 30
                e[mcnt] = (edge) \{x, pre[y]\}, pre[y] =
                                                                31
                 mcnt ++;
            }
                                                                 32
        }
                                                                33
                                                                34
    for (int x = 0 ; x < m ; ++ x) {
                                                                 35
        vector< pair<double , int> > V;
                                                                36
        for (int i = pre[x] ; ~i ; i = e[i].next) {
                                                                 37
            int v = e[i].x;
                                                                 38
            V.push\_back(make\_pair((P[y] - P[x]).arg(), i)
                                                                39
                                                                 40
        sort(V.begin() , V.end());
                                                                 41
        for (int i = 0 ; i < V.size() ; ++ i) {</pre>
                                                                 42
            int j = (i + 1) % V.size();
                                                                 43
            Next[V[j].second \land 1] = V[i].second;
                                                                 44
                                                                45
                                                                46
    double res = 0;
                                                                 47
    for (int i = 0 ; i < mcnt ; ++ i) {</pre>
                                                                48
        if (!vis[i]) {
                                                                 49
            int \times = i;
                                                                50
```

double area = 0:

51

```
52
                  while (!vis[x]) {
53
                      vis[x] = 1;
54
                      area += (P[e[x \land 1].x] \land P[e[x].x]);
55
                      x = Next[x];
56
                  if (x == i \&\& dcmp(area) > 0)
57
                      res += area;
58
59
             }
60
         printf("%.8f\n" , res / 2);
61
62
```

基础 3D 几何 4.8

```
const double eps = 1e-8 , pi = acos(-1.0);
    inline int dcmp(double x) {
3
        return (x > eps) - (x < -eps);
 4
    struct Point {
5
        double x , y , z;
        Point () \{x = y = z = 0;\}
8
        Point (double _x , double _y , double _z) {
            x = _x , y = _y , z = _z;
10
11
        void input() {
            scanf("%lf%lf%lf" , &x , &y , &z);
12
13
14
        bool operator < (const Point &R) const {</pre>
            if (dcmp(x - R.x) != 0)
15
16
                return x < R.x;
17
            if (dcmp(y - R.y) != 0)
                return y < R.y;</pre>
18
            return z < R.z;
19
20
        bool operator == (const Point &R) const {
21
            return dcmp(x - R.x) == 0 \&\& dcmp(y - R.y) == 0 \&\&
             dcmp(z - R.z) == 0;
23
24
        Point operator + (const Point& R) const {
25
            return Point(x + R.x, y + R.y, z + R.z);
26
27
        Point operator - (const Point& R) const {
28
            return Point(x - R.x , y - R.y , z - R.z);
29
        Point operator * (const double& R) const {
30
31
            return Point(x * R , y * R , z * R);
32
        Point operator / (const double& R) const {
33
            return Point(x / R , y / R , z / R);
34
35
        double operator % (const Point& R) const {
36
            return x * R.x + y * R.y + z * R.z;
37
38
39
        Point operator ^ (const Point& R) const {
            return Point(y * R.z - z * R.y , z * R.x - x * R.z
              , x * R.y - y * R.x);
41
        inline double len() {
42
            return sqrt(*this % *this);
43
44
45
46
    Point GetLinePlaneProjection(Point A , Point P , Point n)
        double t = (n \% (P - A)) / (n \% n);
47
        return A + n * t; // t * n.len() 是距离
48
49
    } // 直线平面投影
    Point GetLinePlaneIntersection(Point A , Point V , Point P
50
        double t = (n \% (P - A)) / (n \% V);
51
        return A + V * t;
52
    } // 直线平面交点
    inline double area(Point A , Point B , Point C) {
54
55
        return ((B - A) \land (C - A)).len();
56
    bool PointinTri(Point P) {
57
        double area1 = area(P , a[0] , a[1]);
58
        double area2 = area(P , a[1] , a[2]);
59
        double area3 = area(P , a[2] , a[0]);
60
61
        return dcmp(area1 + area2 + area3 - area(a[0], a[1],
         a[2])) == 0;
62
63
    double GetLineIntersection(Point P , Point v , Point Q ,
    Point w) {
```

```
//共面时使用
    Point u = P - Q;
                                                              65
    Point delta = v \wedge w , cross = w \wedge u;
                                                              66
    if (dcmp(delta.z) != 0)
                                                             67
        return cross.z / delta.z;
                                                             68
    else if (dcmp(delta.y) != 0)
                                                             69
       return cross.y / delta.y;
                                                             70
    else if (dcmp(delta.x) != 0)
                                                             71
       return cross.x / delta.x;
                                                             72
    else {
                                                             73
        return 1e60;
                                                             74
    }
                                                              75
                                                             76
                                                             77
//a点绕0b向量逆时针旋转弧度angle. cossin可预先计算
                                                             78
Point Rotate(Point a, Point b, double angle) {
                                                             79
   static Point e1 ,e2 , e3;
                                                              80
    b = b / b.len() , e3 = b;
                                                             81
    double lens = a % e3;
                                                             82
    e1 = a - e3 * lens;
                                                             83
    if (dcmp(e1.len()) > 0)
                                                             84
        e1 = e1 / e1.len();
                                                             85
    else
                                                             86
        return a;
                                                             87
    e2 = e1 ^ e3;
                                                             88
    double x1 = a \% e2 , y1 = a \% e1 , x2 , y2;
                                                             89
    x2 = x1 * cos(angle) - y1 * sin(angle);
                                                             90
    y2 = x1 * sin(angle) + y1 * cos(angle);
                                                             91
    return e3 * lens + e1 * y2 + e2 * x2;
                                                             92
                                                             93
                                                             94
   绕任意轴(过原点)逆时针旋转(注意要把轴向量归一化,不
                                                             95
   然会在"点在轴上"这个情况下出问题)
   rotate x v z d
                                                             96
   | (1-\cos(d))*x*x+\cos(d)
                               (1-\cos(d))*x*y+\sin(d)*z
                                                             97
   (1-\cos(d))*x*z-\sin(d)*y
                             0 |
   | (1-\cos(d))^*y^*x-\sin(d)^*z
                               (1-\cos(d))*y*y+\cos(d)
                                                             98
   (1-\cos(d))*y*z+\sin(d)*x
                            0
   (1-\cos(d))*z*x+\sin(d)*y
                               (1-\cos(d))*z*y-\sin(d)*x
                                                             99
                             0 |
   (1-\cos(d))*z*z+\cos(d)
                                          0
                                                              100
                                          1 |
                                                              101
```

凸包 3D 4.9

```
double mix(const Point &a, const Point &b, const Point &c)
    return a % (b ^ c);
const int N = 305:
int mark[N][N];
Point info[N];
                                                               6
int n , cnt;
double area(int a, int b, int c) {
                                                               9
    return ((info[b] - info[a]) ^ (info[c] - info[a])).len
                                                               10
    ();
                                                               11
double volume(int a, int b, int c, int d) {
                                                               12
    return mix(info[b] - info[a], info[c] - info[a], info[
                                                               13
    d] - info[a]);
                                                               14
struct Face {
                                                               15
    int v[3];
                                                               16
    Face() {}
                                                               17
    Face(int a, int b, int c) {
                                                               18
        v[0] = a , v[1] = b , v[2] = c;
                                                               20
    int& operator [] (int k) {
                                                               21
        return v[k];
                                                               23
                                                               24
vector <Face> face;
                                                               25
inline void insert(int a, int b, int c) {
                                                               26
                                                               27
    face.push_back(Face(a, b, c));
                                                               28
void add(int v) {
                                                               29
    vector <Face> tmp;
                                                               30
    int a, b, c;
                                                               31
                                                               32
    cnt ++
    for (int i = 0; i < face.size()
                                     ; ++ i) {
                                                               33
        a = face[i][0] , b = face[i][1] , c = face[i][2];
```

```
35
             if (dcmp(volume(v, a, b, c)) < 0)
                 mark[a][b] = mark[b][a] = mark[b][c] = mark[c
36
                 ][b] = mark[c][a] = mark[a][c] = cnt;
37
                 tmp.push_back(face[i]);
38
39
         face = tmp;
40
         for (int i = 0; i < tmp.size(); ++ i) {</pre>
41
             a = face[i][0] , b = face[i][1] , c = face[i][2];
42
             if (mark[a][b] == cnt) insert(b, a, v);
43
44
             if (mark[b][c] == cnt) insert(c, b, v);
45
             if (mark[c][a] == cnt) insert(a, c, v);
46
47
48
    int Find() {
        for (int i = 2; i < n; ++ i) {</pre>
49
             Point ndir = (info[0] - info[i]) \land (info[1] - info
             [i]);
             if (ndir == Point())
51
                 continue;
52
             swap(info[i], info[2]);
53
54
             for (int j = i + 1; j < n; j++)
                 if (dcmp(volume(0, 1, 2, j)) != 0) {
55
                     swap(info[j], info[3]);
56
                     insert(0, 1, 2);
57
                     insert(0, 2, 1);
58
59
                     return 1;
60
61
         return 0;
62
63
    void work() {
64
         for (int i = 0; i < n; ++ i)
65
             info[i].input();
66
67
         sort(info, info + n);
         n = unique(info, info + n) - info;
68
         face.clear();
69
70
         random_shuffle(info, info + n);
         if (Find()) {
71
             memset(mark, 0, sizeof(mark));
72
73
             cnt = 0;
             for (int i = 3; i < n; ++ i) add(i);</pre>
74
75
             vector<Point> Ndir;
             for (int i = 0; i < face.size(); ++i) {</pre>
76
                 Point p = (info[face[i][0]] - info[face[i
77
                 ][1]]) ^{(info[face[i][2]] - info[face[i][1]])}
                 p = p / p.len();
78
                 Ndir.push_back(p);
79
80
             sort(Ndir.begin(), Ndir.end());
81
             int ans = unique(Ndir.begin(), Ndir.end()) - Ndir.
             begin();
83
             printf("%d\n", ans);
         } else {
84
             printf("1\n");
85
86
87
    }
```

5 String

5.1 Ext-KMP

```
f[0] = m;
    for (int i = 1 , k = 0 ; i < m ; ++ i) {
   int len = k + f[k], l = f[i - k];</pre>
 3
         if (i > 1 \&\& 1 < len - i)
 5
              f[i] = 1;
         else {
 6
              int j = i > 1? max(0 , len - i) : 0;
              while(i + j < m && t[j] == t[i + j])
 8
9
                  ++ j;
              f[i] = j , k = i;
10
11
12
    for (int i = 0 , k = 0 ; i < n ; ++ i) {</pre>
13
         int len = k + L[k] , l = f[i - k];
14
15
         if (i > 0 \&\& 1 < len - i)
              L[i] = 1;
16
         else {
17
              int j = i > 0? max(0, len - i): 0;
18
              while (j < m \&\& t[j] == s[(i + j) \% n])
19
```

```
++ j;

L[i] = j , k = i;

}

}
```

5.2 manacher

```
s[0] = '%';
                                                                    1
for (int i = 0 ; str[i] ; ++ i) {
    s[len ++] = '#';
                                                                    2
    s[len ++] = str[i];
    s[len ++] = '#';
s[len] = 0;
int id = 0 , mx = 0;
for (int i = 0 ; i < len ; ++ i) {</pre>
                                                                    8
    p[i] = mx > i ? min(p[id + id - i], mx - i) : 1;
    while (s[i + p[i]] == s[i - p[i]]) ++ p[i];
                                                                    10
    if (i + p[i] > mx)
                                                                    11
         mx = i + p[i] , id = i;
                                                                    12
```

5.3 最小表示法

```
int MinR(char *str) {
    int i = 0 , j = 1 , k = 0 , len = strlen(str);
                                                                2
    while (i < len && j < len && k < len) {
        int cmp = str[(j + k) \% len] - str[(i + k) \% len];
        if (!cmp)
             ++ k;
        else {
                                                                7
            if (cmp > 0)
                                                                8
                j += k + 1;
            else i += k + 1;
                                                                10
            if (i == j) ++ j;
                                                                11
            k = 0;
                                                                12
        }
                                                                13
                                                                14
    return min(i , j);
                                                                15
```

5.4 后缀数组

```
\begin{array}{l} \textbf{int} \ sa[N] \ , \ t1[N] \ , \ t2[N] \ , \ c[N]; \\ \textbf{int} \ Rank[N] \ , \ height[N]; \end{array}
void buildsa(char *s , int n , int m) {
   int i , k , p , a1 , a2 , *x = t1 , *y = t2;
   memset(c , 0 , m << 2);</pre>
                                                                                      3
      for (i = 0 ; i < n ; ++ i) ++ c[x[i] = s[i]];
                                                                                      6
     for (i = 1 ; i < m ; ++ i) c[i] += c[i - 1];
for (i = n - 1 ; i >= 0 ; -- i) sa[-- c[x[i]]] = i;
                                                                                      8
      for (k = 1, p = 0; k < n; k <<= 1, p = 0) {
           for (i = n - k ; i < n ; ++ i) y[p ++] = i;
for (i = 0 ; i < n ; ++ i) if (sa[i] >= k) y[p ++]
                                                                                      10
                                                                                      11
             = sa[i] - k;
           memset(c , 0 , m << 2);
                                                                                      12
           for (i = 0 ; i < n ; ++ i) ++ c[x[y[i]]];
                                                                                      13
           for (i = 1 ; i < m ; ++ i) c[i] += c[i - 1];
                                                                                      14
           for (i = n - 1; i \ge 0; -i) sa[--- c[x[y[i]]]]
                                                                                      15
           = y[i];
           swap(x , y) , p = 1 , x[sa[0]] = 0;
for (i = 1; i < n; ++ i) {
    a1 = sa[i-1] + k < n ? y[sa[i-1] + k] :
                                                                                      16
                                                                                      17
                                                                                      18
                 -1:
                 a2 = sa[i] + k < n ? y[sa[i] + k] : -1;
                                                                                      19
                 x[sa[i]] = (y[sa[i-1]] = y[sa[i]] & a1 = a2) ? p - 1 : p ++;
                                                                                      20
                                                                                      21
           if (p \ge n) break; m = p;
                                                                                      22
                                                                                      23
      for (i = 0 ; i < n ; ++ i) Rank[sa[i]] = i;
                                                                                      24
     for (i = 0 , k = 0; i < n ; ++ i) {
   if (k) — k; if (!Rank[i]) continue;</pre>
                                                                                      25
                                                                                      26
           int j = sa[Rank[i] - 1];
                                                                                      27
           while (s[i + k] == s[j + k]) ++ k;
                                                                                      28
           height[Rank[i]] = k;
                                                                                      29
                                                                                     30
                                                                                      31
 /******Suffix array for Trie *****/
                                                                                      32
int f[18][N] , s[N] , dep[N];
                                                                                     33
int sa[N] , t1[N] , t2[N] , c[N];
                                                                                     34
int Rank[18][N] , rnk[N];
                                                                                      35
inline int LCP(int x , int y) {
                                                                                      36
```

```
37
        int len = 0;
        for (int i = 17; i \ge 0; — i)
38
39
             if (Rank[i][x] && Rank[i][x] == Rank[i][y]) {
40
                 len += 1 << i;
                 x = f[i][x];
41
                 y = f[i][y];
42
            }
43
        return len:
44
45
    void buildsa(int *s , int n , int m) {
46
        int i , j , k , p , a1 , a2;
int *x = t1 , *y = t2;
47
48
        memset(c , 0 , m + 1 << 2);
49
        for (i = 1; i \le n; ++ i) ++ c[x[i] = s[i]];
50
        for (i = 1 ; i \le m ; ++ i) c[i] += c[i - 1];
51
        for (i = n ; i \ge 1 ; -- i) sa[c[x[i]] --] = i;
52
        for (k = 1, j = 0, p = 0; k < n; k <<= 1, ++ j) {
             memset(c , 0 , m + 1 << 2);
54
             for (i = 1 ; i \le n ; ++ i) Rank[j][i] = x[i];
55
             for (i = 1; i <= n; ++ i) ++ c[x[f[j][i]]];</pre>
56
             for (i = 1 ; i \le m ; ++ i) c[i] += c[i-1];
57
58
             for (i = n ; i >= 1 ; -- i) y[c[x[f[j][i]]] --] =
59
            memset(c , 0 , m + 1 << 2);
             for (i = 1 ; i \le n ; ++ i) ++ c[x[y[i]]];
60
             for (i = 1 ; i \le m ; ++ i) c[i] += c[i - 1];
61
62
             for (i = n ; i \ge 1 ; -- i) sa[c[x[y[i]]] --] = y[
             i];
             swap(x , y) , p = 1 , x[sa[1]] = 1;
63
             for (i = 2; i <= n; ++ i) {
64
                 a1 = y[f[j][sa[i - 1]]];
65
                 a2 = y[f[j][sa[i]]];
66
                 x[sa[i]] = (y[sa[i-1]] == y[sa[i]] && a1 ==
                 a2) ? p : ++ p;
             }
68
69
            m = p;
70
71
        for (i = 1; i <= n; ++ i) rnk[sa[i]] = i;</pre>
    }
```

5.5 后缀自动机

```
int root , last , nodecnt;
    int u[N << 1][26] , val[N << 1] , f[N << 1];</pre>
     inline int newnode(int _val) {
         ++ nodecnt:
         memset(u[nodecnt] , 0 , sizeof(u[nodecnt]));
val[nodecnt] = _val , f[nodecnt] = 0;
 5
         return nodecnt;
 8
    void extend(int c) {
9
         int p = last , np = newnode(val[p] + 1);
10
         while (p \&\& u[p][c] == 0)
11
             u[p][c] = np , p = f[p];
12
         if (p == 0)
13
             f[np] = root;
14
         else {
15
             int q = u[p][c];
16
             if (val[p] + 1 == val[q]) {
17
18
                  f[np] = q;
             } else {
                  int nq = newnode(val[p] + 1);
20
21
                  memcpy(u[nq], u[q], sizeof(u[q]));
22
                  f[nq] = f[q];
                  f[q] = f[np] = nq;
23
                  while (p \&\& u[p][c] == q)
24
                      u[p][c] = nq , p = f[p];
25
26
             }
27
         last = np;
28
29
30
    void work() {
         nodecnt = 0;
31
32
         root = last = newnode(0);
    }
```

5.6 后缀树

```
const int INF = 10000000000, C = 26, N = 100005;
int pos;
int text[N];
struct Node {
```

```
int 1, r;
    Node *suf, *ch[C];
                                                                   6
    int dgr;
    Node *fa:
    Node (int l = -1, int r = INF) : l(1), r(r) {
                                                                   10
         suf = fa = NULL;
                                                                   11
        memset(ch, 0, sizeof(ch));
                                                                   12
         dqr = 0;
                                                                    13
                                                                   14
    Node* addEdge(Node *t) {
                                                                   15
         int c = text[t\rightarrow 1];
                                                                   16
         dgr += !ch[c];
                                                                   17
         ch[c] = t;
         t->fa = this;
                                                                   19
         return t;
                                                                   20
    int len() {
                                                                    22
         return min(r, pos + 1) -1;
                                                                   23
                                                                   25
};
                                                                   26
int top;
                                                                   27
Node pool[N << 1];
                                                                   28
Node *root, *nxtSuf, *cur;
                                                                    29
int remCnt, curP, curLen;
                                                                   30
long long size;
                                                                   31
queue<Node*> leaves;
                                                                    32
void init() {
                                                                   33
    top = 0, pos = -1;
                                                                   34
    remCnt = 0, curP = 0, curLen = 0;
                                                                   35
    nxtSuf = NULL;
                                                                   36
    root = cur = new(pool + (top++)) Node(-1, -1);
                                                                    37
    size = 0:
                                                                    38
    while (leaves.size()) {
                                                                   39
         leaves.pop();
                                                                    40
                                                                    41
                                                                    42
void link(Node *u) {
                                                                    43
    if (nxtSuf) {
                                                                    44
         nxtSuf->suf = u;
                                                                    45
                                                                   46
    nxtSuf = u;
                                                                    47
                                                                    48
bool walk(Node *u) {
                                                                    49
    int len = u->len();
                                                                   50
    if (curLen >= len) {
                                                                   51
        curP += len;
                                                                   52
         curLen -= len;
                                                                   53
        cur = u;
                                                                   54
        return true;
                                                                   55
    return false;
                                                                   57
                                                                   58
void extend(int c) {
                                                                   59
    text[++pos] = c;
                                                                   60
    nxtSuf = NULL;
                                                                    61
    ++remCnt;
                                                                   62
    while (remCnt) {
                                                                   63
         curP = curLen ? curP : pos;
                                                                    64
         int curE = text[curP];
                                                                   65
         if (!cur->ch[curE]) {
                                                                    66
             leaves.push(cur->addEdge(new(pool + (top++))
                                                                    67
             Node(pos)));
             link(cur);
                                                                    68
         } else {
                                                                    69
             Node *nxt = cur->ch[curE];
                                                                   70
             if (walk(nxt)) {
                                                                    71
                  continue;
                                                                   72
                                                                   73
             if (\text{text[nxt}\rightarrow 1 + \text{curLen]} == c) {
                                                                   74
                                                                   75
                  ++curl en:
                 link(cur);
                                                                   76
                 break;
                                                                   77
                                                                   78
             Node *split = new(pool + (top++)) Node(nxt \rightarrow 1,
                                                                    79
              nxt->1 + curLen);
             cur->addEdge(split);
                                                                    80
             leaves.push(split\rightarrowaddEdge(new(pool + (top++))
                                                                   81
              Node(pos)));
             nxt->l += curLen;
                                                                    82
             split->addEdge(nxt);
                                                                   83
             link(split);
                                                                   84
```

```
85
                -remCnt;
 86
 87
              if (cur == root && curLen > 0) {
                  curP = pos - (--curLen);
 88
              } else {
 89
 90
                  cur = cur->suf ? cur->suf : root;
 91
 92
          size += leaves.size();
 93
 94
 95
     void finish() {
          nxtSuf = NULL;
 96
          for (int i = 0; i < top; ++i) {
 97
              if (pool[i].r == INF) {
 98
                  link(pool + i);
 99
100
101
          while (remCnt > 0) {
102
103
              if (curLen) {
                  int curE = text[curP];
104
                  Node *nxt = cur->ch[curE];
105
106
                  if (walk(nxt)) {
107
                       continue;
108
                  Node *split = new(pool + (top++)) Node(nxt->1,
109
                   nxt->1 + curLen);
                  leaves.push(cur->addEdge(split));
110
111
                  nxt->1 += curLen;
                  split->addEdge(nxt);
112
                  link(split);
113
114
              } else {
                  leaves.push(cur);
115
                  link(cur);
117
118
               —remCnt;
              if (cur == root && curLen > 0) {
119
                   —curLen:
120
121
                  curP = pos - remCnt + 1;
122
              } else {
                  cur = cur->suf ? cur->suf : root;
123
124
125
          if (nxtSuf != root) {
126
127
              link(root);
128
129
130
     void eraseUp(Node *&u) {
          size -= u->len();
131
132
          int ch = text[u->1];
          u = u->fa;
133
          u\rightarrow ch[ch] = NULL;
134
135
          —(u->dgr);
136
137
     void erase() {
         Node *u = leaves.front();
138
          leaves.pop();
139
140
          while (u->dgr == 0 && u != cur) {
              eraseUp(u);
141
142
143
          if (u == cur) {
              if (cur->dgr == 0 && curLen == 0) {
144
145
                  int len = u->len();
                  curLen = len;
curP = pos - len + 1;
146
147
                  cur = cur->fa;
148
149
                  eraseUp(u);
150
151
              if (curLen) {
                  int curE = text[curP];
152
153
                  if (!cur->ch[curE]) {
                       Node *leaf = new(pool + (top++)) Node(pos
154
                        - curLen + 1);
                       leaves.push(cur->addEdge(leaf));
155
                       size += leaf->len();
156
                       —remCnt;
157
                       if (cur == root && curLen > 0) {
158
                           curP = pos - (--curLen) + 1;
159
                       } else {
160
                           cur = cur->suf ? cur->suf : root;
161
162
163
                       while (curLen && walk(cur->ch[text[curP]])
                       ) {
164
                           continue:
```

```
}
                                                                  165
             }
                                                                  166
        }
                                                                  167
    }
                                                                  168
                                                                  169
int n;
                                                                  170
char s[N], buf[N];
                                                                  171
int ord[N], stop, sord[N << 1];
                                                                  172
void dfs(Node *u) {
                                                                  173
    sord[u - pool] = stop++;
                                                                  174
    for (int i = 0; i < C; ++i) {
                                                                  175
        if (u->ch[i]) {
                                                                  176
             dfs(u->ch[i]);
                                                                  177
                                                                  178
    }
                                                                  179
                                                                  180
void getOrd() {
                                                                  181
    init();
                                                                  182
    for (int i = 0; i < n; ++i) {
                                                                  183
        extend(s[i] - 'a');
                                                                  184
                                                                  185
    finish();
                                                                  186
    stop = 0;
                                                                  187
    dfs(root);
                                                                  188
    int i = 0;
                                                                  189
    while (leaves.size()) {
                                                                  190
        ord[i++] = sord[leaves.front() - pool];
                                                                  191
        leaves.pop();
                                                                  192
    }
                                                                  193
                                                                  194
long long res[N];
                                                                  195
int main() {
                                                                  196
    while (scanf("%s", s) == 1) {
                                                                  197
        n = strlen(s);
                                                                  198
        getOrd();
                                                                  199
        int q , 1;
                                                                  200
        scanf("%d%d", &q , &l);
                                                                  201
        long long ans = 0;
                                                                  202
        int pos = 0;
                                                                  203
        init();
                                                                  204
        for (int i = 0; i < n; ++i) {</pre>
                                                                  205
             extend(s[i] - 'a');
                                                                  206
             if (i >= 1) {
                                                                  207
                 erase();
                                                                  208
                                                                  209
             if (i >= l - 1) {
                                                                  210
                 res[i-l+1] = size;
                                                                  211
                 if (size > ans || (size == ans && ord[i -
                                                                  212
                 1 + 1] < ord[pos])) {</pre>
                      ans = size;
                                                                  213
                     pos = i - 1 + 1;
                                                                  214
                 }
                                                                  215
             }
                                                                  216
                                                                  217
        while (q —) {
                                                                  218
             int x:
                                                                  219
             scanf("%d" ,&x);
                                                                  220
             printf("%lld\n", res[--x]);
                                                                  221
        }
                                                                  222
                                                                  223
                                                                  224
    return 0;
                                                                  225
```

5.7 回文树

```
struct PalinTree {
    char str[N];
    int n;
    int u[N][26];
    int len[N] , f[N] , cnt[N];
    int nodecnt , root;
                                                                  6
    void init() {
        scanf("%s"
                    , str);
        n = strlen(str);
        nodecnt = 2:
                                                                  10
        len[1] = -1, len[2] = 0;
                                                                  11
        f[1] = 0, f[2] = 1;
                                                                  12
        memset(u[1] , 0 , sizeof(u[1]));
                                                                  13
        memset(u[2] \ , \ 0 \ , \ \textbf{sizeof}(u[2]));
                                                                  14
        root = 1;
                                                                  15
        for (int i = 0; i < n; ++ i)
                                                                  16
             extend(i, str[i] - 'a');
                                                                  17
```

```
18
        void extend(int i , int c) {
19
20
            int p = root;
            while (str[i-1-len[p]] != str[i])
21
22
                p = f[p];
            int& pp = u[p][c];
23
            if (!pp) {
24
25
                pp = ++ nodecnt;
                 len[pp] = len[p] + 2;
                cnt[pp] = 0;
27
28
                memset(u[pp] , 0 , sizeof(u[pp]));
                 int q = f[p];
29
                 while (q \&\& str[i-1-len[q]] != str[i])
30
                     q = f[q];
31
                 f[pp] = q ? u[q][c] : 2;
32
33
            ++ cnt[pp];
            root = pp;
35
36
    }
```

Other 6

6.1emacs

```
(add-hook 'c++-mode-hook 'linux-cpp-mode)
    (defun linux-cpp-mode()
      (define_key c++_mode_map [return] 'newline_and_indent)
 3
      (interactive)
      (c—set—style "K&R")
      (c-toggle-auto-state)
      (setq c-basic-offset 4)
8
      (show-paren-mode t)
      (setq show-paren-style 'parentheses)
 9
10
    (setq-default indent-tabs-mode nil)
    (global—linum—mode t)
```

6.2 Dancing Links

```
int U[M] , D[M] , L[M] , R[M], col[M] , row[M];
1
    int cnt , p[N] , s[N];
 4
    #define FOR(i,A,s) for (int i = A[s]; i != s; i = A[i])
    //由矩阵建立十字链表
    cnt = n + 1;
6
    for (i = 0 ; i \le n ; i ++)
        L[i] = i - 1, R[i] = i + 1;
    memset(s , 0 , sizeof(s));
    L[0] = n , R[n] = 0;
    for (i = 1; i <= n; i ++)
11
12
        p[i] = i;
    for (i = 1 ; i \le n ; ++ i) {
13
        x = y = -1;
14
        for (j = 1; j <= n; j ++)
15
             if (g[i][j]) {
16
17
                 if (x == -1)
                     x = cnt , y = cnt;
19
                     L[cnt] = y , R[y] = cnt , y = cnt;
20
                 D[p[j]] = cnt, U[cnt] = p[j], p[j] = cnt;
21
                 col[cnt] = j, row[cnt] = i, s[j] ++;
22
23
                 ++ cnt;
24
        L[x] = y, R[y] = x;
25
26
    for (i = 1 ; i <= n ;i ++)
   D[p[i]] = i , U[i] = p[i];</pre>
27
28
    //可重复覆盖:
30
    void remove(int c) {
        FOR(i,D,c) L[R[i]] = L[i] , R[L[i]] = R[i];
31
32
    void resume(int c) {
33
34
        FOR(i,U,c) L[R[i]] = R[L[i]] = i;
35
    int H() {
36
37
        int val = 0; bool u[N] = {0};
        FOR(i,R,0) if (!u[i]) {
38
39
             ++ val;
             FOR (j,D,i) FOR(k,R,j)
40
                 u[col[k]] = 1;
41
```

```
return val;
                                                                 43
                                                                 44
bool dfs(int d) {
                                                                 45
    if (d + H() > K)return 0 ;
                                                                 46
    if (R[0] == 0) {
                                                                 47
        return 1;
                                                                 48
                                                                 49
    int c = R[0];
                                                                 50
    FOR(i,R,0) if (s[i] < s[c]) c = i;
                                                                51
    FOR(i,D,c) {
                                                                 52
        remove(i);
                                                                 53
        FOR(j,R,i) remove(j);
                                                                54
        if (dfs(d + 1))
            return 1;
                                                                56
        FOR(j,L,i) resume(j);
                                                                57
        resume(i);
                                                                 59
    return 0:
                                                                 60
                                                                 61
//精确覆盖
                                                                 62
void remove(int c) {
                                                                 63
    L[R[c]] = L[c], R[L[c]] = R[c];
                                                                 64
    FOR(i,D,c) FOR(j,R,i)
                                                                 65
        U[D[j]] = U[j], D[U[j]] = D[j], — s[col[j]];
                                                                 66
                                                                 67
void resume(int c) {
                                                                 68
    FOR(i,U,c) FOR(j,L,i)
                                                                 69
        U[D[j]] = D[U[j]] = j , ++ s[col[j]];
                                                                70
    L[R[c]] = R[L[c]] = c;
                                                                71
                                                                 72
                                                                73
bool dfs(int d) {
                                                                 74
    if (R[0] == 0)
                                                                 75
        return 1:
                                                                76
    int c = R[0];
    FOR(i,R,0) if (s[i] < s[c]) c = i;
                                                                 78
    remove(c);
                                                                79
    FOR(i,D,c) {
                                                                 80
        ans.pb(row[i]);
                                                                 81
        FOR(j,R,i) remove(col[j]);
                                                                 82
        if (dfs(d + 1)) return 1;
                                                                83
        FOR(j,L,i) resume(col[j]);
                                                                 84
        ans.pop_back();
                                                                 85
                                                                86
    resume(c);
                                                                87
    return 0;
                                                                 88
                                                                89
```

6.3unorderedmap

```
template<typename T1 , typename T2> struct hashmap {
   const static int MOD = 99991;
    const static int Size = 500005;
    int pre[MOD] , mcnt;
    struct node {
        T1 key;
        T2 val
        int next;
    } e[Size];
    void clear() {
                                                               10
        memset(pre , -1 , sizeof(pre));
                                                                11
        mcnt = 0;
                                                                12
                                                               13
    void insert(const T1& K , const T2& V) {
                                                                14
        int x = K \% MOD;
                                                                15
        e[mcnt] = (node) \{K, V, pre[x]\};
                                                               16
        pre[x] = mcnt ++;
                                                                18
    int find(const T1 &K) {
                                                                19
        int x = K \% MOD;
        for (int i = pre[x] ; ~i ; i = e[i].next)
                                                                21
            if (e[i].key == K)
                                                                22
                return i;
                                                                23
        return -1;
                                                                24
                                                                25
    T2& operator [] (const T1 &x){
                                                               26
        int i = find(x);
                                                               27
        if (!~i){
                                                                28
            insert(x, 0);
                                                               29
            return e[mcnt - 1].val;
                                                               30
                                                               31
        return e[i].val;
```

32

6.4 插头 DP

```
inline int getpos(int x , int k) {
1
2
      return x \gg k + k \& 3;
3
    inline int setpos(int x , int k , int v) {
      return (x \& \sim (3 << k + k)) | (v << k + k);
6
    }
    void work() {
      int res = -1 << 30;
9
      for (int i = 0; i < n; ++ i)
10
        for (int j = 0; j < m; ++ j) {
11
          scanf("%d" , &a[i][j]);
12
13
          res = max(res , a[i][j]);
14
15
      int cur = 0 , nxt = 1;
      f[cur].clear();
16
      f[cur][0] = 0;
17
18
      for (int i = 0; i < n; ++ i) {
        for (int j = 0; j < m; ++ j) {
19
20
          f[nxt].clear();
21
          for (int it = 0 ; it < f[cur].mcnt ; ++ it) {</pre>
22
             int k = f[cur].e[it].key;
23
             int w = f[cur].e[it].val;
             int L = getpos(k , j);
25
26
             int U = getpos(k , j + 1);
             int num = getpos(k , m + 1);
27
             //printf("%d %d %d %d : %d\n" , i , j , k , num ,
28
             w);
29
             if (!L && !U)
               f[nxt][k] = max(f[nxt][k], w);
30
             w += a[i][j];
             if (!L && !U) {
32
33
               if (j + 1 < m) {
                 int K = setpos(k , j , 1);
K = setpos(K , j + 1 , 2);
34
35
                 f[nxt][K] = max(f[nxt][K], w);
36
37
               if (num < 2) {
38
                 int K = setpos(k , m + 1 , num + 1);
                 K = setpos(K , j , 3);
f[nxt][K] = max(f[nxt][K] , w);
40
41
                 if (j + 1 < m) {
                   int K = setpos(k , m + 1 , num + 1);
43
44
                   K = setpos(K, j + 1, 3);
                   f[nxt][K] = max(f[nxt][K], w);
45
46
                 }
47
             } else {
48
               static int match[N];
49
               static int S[N];
               int top = 0;
51
               for (int 1 = 0; 1 \le m; ++ 1) {
52
                 int x = getpos(k, 1);
53
                 if (x == 1)
54
                   S[top ++] = 1;
                 if (x == 2) {
56
57
                     top;
                   match[S[top]] = 1;
                   match[1] = S[top];
59
60
61
               if (L && !U) {
62
                 if (num < 2) {
63
                   int K = setpos(k , m + 1 , num + 1);
64
65
                   K = setpos(K , j , 0);
                   if (L != 3) K = setpos(K , match[j] , 3);
                   f[nxt][K] = max(f[nxt][K], w);
67
68
                 if (j + 1 < m) {
69
                   int K = setpos(k , j , U);
70
```

```
K = setpos(K, j + 1, L);
            f[nxt][K] = max(f[nxt][K], w);
                                                             72
                                                             73
          f[nxt][k] = max(f[nxt][k], w);
                                                             74
                                                             75
        if (!L && U) {
                                                             76
          if (num < 2) {
                                                             77
            int K = setpos(k , m + 1 , num + 1);
                                                             78
            K = setpos(K, j + 1, 0);
                                                             79
            if (U != 3) K = setpos(K, match[j + 1], 3)
                                                             80
            f[nxt][K] = max(f[nxt][K], w);
                                                             81
                                                             82
          if (j + 1 < m) {
                                                             83
            f[nxt][k] = max(f[nxt][k], w);
                                                             84
                                                             85
          int K = setpos(k , j , U);
                                                             86
          K = setpos(K, j + 1, L);
                                                             87
          f[nxt][K] = max(f[nxt][K], w);
                                                             88
                                                             89
        if (L && U) {
                                                             90
          int K = setpos(k , j , 0);
                                                             91
          K = setpos(K, j + 1, 0);
                                                             92
          if (L == 3 || U == 3) {
                                                             93
            if (L != 3)
                                                             94
              K = setpos(K, match[j], 3);
                                                             95
            if (U != 3)
                                                             96
              K = setpos(K, match[j + 1], 3);
                                                             97
            f[nxt][K] = max(f[nxt][K], w);
                                                             98
          } else {
                                                             99
            if (L == U) {
                                                             100
              if (L == 1 && U == 1) {
                                                             101
                K = setpos(K, match[j + 1], 1);
                                                             102
                                                             103
              if (L == 2 && U == 2) {
                                                             104
                K = setpos(K, match[j], 2);
                                                             105
                                                             106
              f[nxt][K] = max(f[nxt][K], w);
                                                             107
            } else if (L == 2 && U == 1) {
                                                             108
              f[nxt][K] = max(f[nxt][K], w);
                                                             109
                                                             110
                                                             111
        }
                                                             112
     }
                                                             113
                                                             114
    swap(cur , nxt);
                                                             115
                                                             116
  f[nxt].clear();
                                                             117
  for (int it = 0 ; it < f[cur].mcnt ; ++ it) {</pre>
    int k = f[cur].e[it].key;
                                                             119
    int w = f[cur].e[it].val;
                                                             120
    int num = getpos(k , m + 1);
                                                             121
    k = setpos(k , m + 1 , 0) << 2;
                                                             122
    f[nxt][setpos(k , m + 1 , num)] = w;
                                                             123
                                                             124
  swap(cur , nxt);
                                                             125
                                                             126
res = max(res , f[cur][2 << m + m + 2]);
                                                             127
cout << res << endl;
                                                             128
                                                             129
```

6.5 压位 LCS

```
typedef unsigned long long LL;
const int N = 1005;
const int B = 64;
const int M = (N + B - 1) / B + 5;
int n , m;
char s[N],
            t[N];
LL c[26][M];
LL f[2][M] , X[M];
void work() {
                                                                   10
    n = strlen(s);
                                                                   11
    m = strlen(t);
                                                                   12
    memset(c , 0 , sizeof(c));
                                                                   13
    for (int i = 0; i < n; ++ i)
                                                                   14
         c[s[i] - 'a'][i >> 6] |= 1ULL << (i & 63);
                                                                   15
    int L = (n + B - 1) / B;
                                                                   16
    int cur = 0 , nxt = 1;
                                                                   17
    memset(f , 0 , sizeof(f));
                                                                   18
    for (int i = 0 ; i < m ; ++ i) {
   int id = t[i] - 'a';</pre>
                                                                   19
                                                                   20
```

```
21
            for (int j = 0; j < L; ++ j)
                 X[j] = f[cur][j] \mid c[id][j];
22
23
            for (int j = 0, x = 1; j < L; ++ j) {
                 int y = f[cur][j] >> 63 & 1;
24
                 f[cur][j] \iff 1 , f[cur][j] \mid = x;
25
26
27
            memcpy(f[nxt] , X , sizeof(X));
28
            for (int j = 0, x = 0; j < L; ++ j) {
29
                 if (f[nxt][j] < x + f[cur][j]) {</pre>
30
                     f[nxt][j] = x + f[cur][j];
31
                     x = 1;
32
                 } else {
33
                     f[nxt][j] = x + f[cur][j];
34
35
                     x = 0;
36
                 f[nxt][j] ^= X[j];
37
38
                 f[nxt][j] &= X[j];
39
40
            swap(cur , nxt);
41
42
        int ans = 0;
        for (int i = 0; i < n; ++ i)
43
            if (f[cur][i >> 6] >> (i & 63) & 1)
44
45
        printf("%d\n" , ans);
46
47
    }
```

6.6 bitset 区间询问

```
int u[N] , v[N];
2
    struct Range {
         int key[N] ,
                       val[N];
3
         bitset<N> w[N / B + 1][N / B + 1];
         void init() { // u有序, B可以开大一点
int m = (n + B - 1) / B;
5
6
             for (int i = 0; i < n; ++ i) {
8
                  key[i] = u[i];
9
                  val[i] = v[i];
10
11
             for (int i = 0 ; i < m ; ++ i) {</pre>
                 int L = i * B , R = min(n , L + B);
for (int j = L ; j < R ; ++ j) {</pre>
12
13
14
                      w[i][i].set(v[j]);
15
16
             for (int i = 0 ; i < m ; ++ i) {</pre>
17
18
                  for (int j = i + 1; j < m; ++ j) {
                      w[i][j] = w[i][j-1] | w[j][j];
19
20
21
         } // 提取key在一个区间内的val构成的bitset
22
         bitset<N> get(int 1 , int r) {
23
             l = lower\_bound(key , key + n , l) - key;
24
25
             r = upper\_bound(key , key + n , r) - key;
             int 11 = 1 / B , rr = r / B;
26
             bitset<N> ret;
27
             if (ll == rr) {
28
                  for (int i = 1 ; i < r ; ++ i)</pre>
29
30
                      ret.set(val[i]);
31
             } else {
                 ret |= w[ll + 1][rr - 1];
32
                  int R = (ll + 1) * B , L = rr * B;
33
                  for (int i = 1; i < R; ++ i)
34
35
                      ret.set(val[i]);
                  for (int i = L ; i < r ; ++ i)</pre>
36
37
                      ret.set(val[i]);
38
39
             return ret:
40
        }
    };
41
```

6.7 转转转

```
bool cmp(const pair<Point , int> &AA , const pair<Point ,
    int> &BB) {
    const Point &A = AA.first;
    const Point &B = BB.first;
    if (A.sign() != B.sign())
        return A.sign() < B.sign();
    return (A ^ B) > 0;
}
```

```
int n , D , id[N][2];
                                                                    8
Point a[N][2] , d[N];
                                                                    9
int f[N];
                                                                    10
vector<int> add[N] , del[N];
                                                                    11
int getf(int x) {
                                                                    12
    return f[x] == x ? x : f[x] = getf(f[x]);
                                                                    13
                                                                    14
Point P , V;
                                                                    15
double len;
                                                                    16
inline double distance(int i) {
                                                                    17
    Point u = P - a[i][0], w = a[i][1] - a[i][0]; return len * (w \wedge u) / (V \wedge w);
                                                                    18
                                                                    19
                                                                    20
struct segment {
                                                                    21
    bool operator () (const int &x , const int &y) {
                                                                    22
         double dx = distance(x);
                                                                    23
         double dy = distance(y);
                                                                    24
         return dx < dy;
                                                                    25
         /*if (fabs(dx - dy) > 1e-6)
                                                                    26
             return dx < dy;
                                                                    27
             return x < y;*/
                                                                    28
    }
                                                                    29
                                                                    30
int main() {
    scanf("%d" , &n);
                                                                    31
                                                                    32
    for (int i = 0 ; i < n ; ++ i) {</pre>
                                                                    33
         for (int j = 0 ; j < 2 ; ++ j) {
    scanf("%d%d" , &a[i][j].x ,</pre>
                                                                    34
                            , &a[i][j].x , &a[i][j].y);
                                                                    35
             d[D ++] = a[i][j];
                                                                    36
         }
                                                                    37
                                                                    38
    sort(d, d + D);
                                                                    39
    D = unique(d, d + D) - d;
                                                                    40
                                                                    41
    for (int i = 0 ; i < n ; ++ i) {</pre>
                                                                    42
         int x = lower\_bound(d, d + D, a[i][0]) - d;
         int y = lower_bound(d, d + D, a[i][1]) - d;
                                                                    44
         id[i][0] = x , id[i][1] = y;
                                                                    45
         E.push_back(make_pair(0 , make_pair(x , y)));
                                                                    46
    }
                                                                    47
                                                                    48
    for (int i = 0; i < D; ++ i) {
                                                                    49
         vector< pair<Point , int> > Vec;
                                                                    50
         for (int j = 0 ; j < D ; ++ j) {</pre>
                                                                    51
             if (i != j)
                                                                    52
                  Vec.push_back(make_pair(d[j] - d[i], j));
                                                                    53
                                                                    54
         for (int j = 0 ; j < D ; ++ j) {
                                                                    55
             add[j].clear();
                                                                    56
             del[j].clear();
                                                                    57
                                                                    58
         set<int , segment> Hash;
         sort(Vec.begin() , Vec.end() , cmp);
                                                                    60
                                                                    61
         P = d[i];
                                                                    62
         V = Point(1, 0);
                                                                    63
         len = 1;
                                                                    64
         for (int j = 0; j < n; ++ j) {
                                                                    65
             int &x = id[j][0] , &y = id[j][1];
                                                                    66
             if (x == i || y == i)
                                                                    67
                  continue:
                                                                    68
             if (((d[x] - d[i]) \land (d[y] - d[i])) < 0) {
                                                                    69
                                                                    70
                  swap(x, y);
                  swap(a[j][0] , a[j][1]);
                                                                    71
                                                                    72
             if (d[y].y >= d[i].y && d[x].y < d[i].y) {</pre>
                                                                    73
                  Hash.insert(j);
                                                                    74
                  add[x].push_back(j);
                                                                    75
                                                                    76
                  del[y].push_back(j);
                                                                    77
                  add[x].push_back(j);
                                                                    78
                                                                    79
                  del[y].push_back(j);
                                                                    80
         }
                                                                    81
                                                                    82
         for (int j = 0 ; j < (int)Vec.size() ; ++ j) {</pre>
                                                                    83
             V = Vec[j].first;
                                                                    84
             len = sqrt(V.len());
                                                                    85
              int x = Vec[j].second;
                                                                    86
             for (auto &k : del[x])
                                                                    87
                  Hash.erase(k);
                                                                    88
             bool flag = 0;
                                                                    89
             if (!Hash.empty()) {
                                                                    90
```

```
91
                       int k = *Hash.begin();
                       \textbf{if} \ (((a[k][1] - a[k][0]) \ ^ \ (d[x] - a[k
 92
                       ][0])) >= 0)
 93
                            flag = 1;
                   } else {
 94
 95
                       flag = 1;
 96
                   if (flag) {
 97
 98
                       //E.push\_back(make\_pair((d[i] - d[x]).len
                        () , make_pair(i , x)));
 99
100
                   for (auto &k : add[x])
                       Hash.insert(k);
101
102
              }
103
104
          return 0;
```

6.8 Time-travel

```
int del[N];
1
    void divide(const vector<int> &A) {
         if (A.size() <= 1)
3
             return;
4
         // 负数询问, 否则偶数插入奇数删除
 5
         vector<int> P , Q;
6
         int r = A.size() , mid = r / 2;
for (int i = 0 ; i < mid ; ++ i)</pre>
7
8
             P.push_back(A[i]);
9
10
         divide(P);
11
         P.clear();
         for (int i = 0; i < r; ++ i)
12
13
             if (A[i] > 0 && (A[i] & 1))
14
                  del[A[i] >> 1] = 1;
         for (int i = 0 ; i < mid ; ++ i)</pre>
15
             if (A[i] > 0 && (~A[i] & 1)) {
                  if (!del[A[i] >> 1])
17
18
                      P.push_back(A[i] >> 1);
19
                      del[A[i] >> 1] = 2;
20
21
         for (int i = mid ; i < r ; ++ i)</pre>
22
             if (A[i] < 0)
23
24
                  Q.push_back(-A[i]);
         update(P , Q);
25
26
         Q.clear();
         int c1 = 0 , c2 = 0;
for (int i = r - 1 ; i >= mid ; — i) {
27
28
             if (A[i] > 0 \&\& (A[i] \& 1) \&\& del[A[i] >> 1] == 2)
29
                  Q.push\_back(A[i] ^ 1);
30
31
                  ++ c1;
             } else if (A[i] < 0) {
32
33
                  Q.push_back(A[i]);
                  ++ c2;
34
35
             }
36
         for (int i = 0 ; i < r ; ++ i)
37
             if (A[i] > 0 && (A[i] & 1))
38
39
                  del[A[i] >> 1] = 0;
         if (c1 && c2)
40
             divide(Q);
41
42
         P.clear();
43
         for (int i = mid ; i < r ; ++ i)</pre>
44
             P.push_back(A[i]);
         divide(P);
45
    }
46
```