UESTC Oblivion模板

Fish

October 22, 2012

Contents

1 全局设定

1.1 GEdit

```
#!/bin/sh
CNAME=$GEDIT_CURRENT_DOCUMENT_NAME
COUT=${CNAME%%\.c*}
cd $GEDIT_CURRENT_DOCUMENT_DIR
if [ -a $COUT ]; then
    rm $COUT
fi
g++ -o $COUT $CNAME -O2 -Wall -g
if [ -a $COUT ]; then
    if [ -x $COUT ]; then
        echo ""
    else
        chmod 755 $COUT
    fi
    gnome-terminal -x /usr/bin/cb_console_runner ./$COUT
else
    echo "编译 $CNAME 失败，未生成 $COUT ! \n"
fi
```

2 数据结构

2.1 Size Balanced Tree

```
struct Node {
    T key;
    int size;
    Node* c[2];
} memo[MaxN], *cur, *nil, *root;

Node* New(T v) {
    cur->key = v, cur->size = 1;
    cur->c[0] = cur->c[1] = nil;
    return cur++;
}

struct Sbt {
    void init() {
        nil = cur = memo;
        root = nil = New(-1);
        nil->size = 0;
    }
    void rotate(Node*& t, int f) {
        Node* k = t->c[f ^ 1];
        t->c[f ^ 1] = k->c[f];
        k->c[f] = t;
        k->size = t->size;
        t->size = t->c[0]->size + t->c[1]->size + 1;
        t = k;
    }
    void keep(Node*& t, int f) {
        if (t == nil)
            return; //TLE
        else if (t->c[f]->c[f]->size > t->c[f ^ 1]->size)
            rotate(t, f ^ 1);
        else if (t->c[f]->c[f ^ 1]->size > t->c[f ^ 1]->size)
            rotate(t->c[f], f), rotate(t, f ^ 1);
        else
            return;
        for (int i = 0; i < 2; i++)
            keep(t->c[i], i);
        for (int i = 0; i < 2; i++)
            keep(t, i);
    }
    void insert(Node*& t, T v) {
        if (t == nil)
            t = New(v);
```

```
        else {
            t->size++;
            insert(t->c[v >= t->key], v);
            keep(t, v >= t->key);
        }
    }
    Node* del(Node*& t, T v) {
        Node* p;
        if (t == nil) return nil;
        t->size--;
        if (v == t->key || t->c[v > t->key] == nil) {
            if (t->c[0] != nil && t->c[1] != nil)
                p = del(t->c[0], v + 1), t->key = p->key;
            else
                p = t, t = t->c[t->c[0] == nil];
            return p;
        } else
            return del(t->c[v > t->key], v);
    }
};
```

2.2 Splay

```
struct Node {
    T key;
    int size;
    bool rev, same;
    Node *c[2], *p;
} node[MaxN], *q[MaxN], *st[MaxN], *nil, *root;
int top1, top2;
Node* New(T v) {
    Node* p;
    if (top2) p = st[--top2];
    else p = &node[top1++];
    p->key = v, p->size = 1, p->rev = p->same = false;
    p->c[0] = p->c[1] = p->p = nil;
    return p;
}
int num[MAX];
struct Splay {
    void init() {
        top1 = top2 = 0;
        nil = node;
        nil = New(-oo), nil->size = 0;
        root = New(-oo), root->c[1] = New(-oo);
        root->c[1]->p = root, update(root);
    }
    void rotate(Node* x, int f) {
        Node* y = x->p;
        pushdown(y), pushdown(x);
        y->c[f ^ 1] = x->c[f], x->p = y->p;
        if (x->c[f] != nil) x->c[f]->p = y;
        if (y->p != nil) y->p->c[y->p->c[1] == y] = x;
        x->c[f] = y, y->p = x;
        update(y);
    }
    void splay(Node* x, Node* f) {
        pushdown(x);
        while (x->p != f) {
            if (x->p->p == f) rotate(x, x->p->c[0] == x);
            else {
                Node* y = x->p;
                int t = (y->p->c[0] == y);
                if (y->c[t] == x) rotate(x, t ^ 1), rotate(x, t);
                else rotate(y, t), rotate(x, t);
            }
        }
        update(x);
        if (f == nil) root = x;
    }
    void select(int k, Node* f) {
        Node* x = root;
        int tmp;
        pushdown(x);
```

```
        while ((tmp = x->c[0]->size) != k) {
            if (k < tmp) x = x->c[0];
            else x = x->c[1], k -= tmp + 1;
            pushdown(x);
        }
        splay(x, f);
    }
    void clear(Node* x) {
        int f = 0, b = 0;
        if (x == nil) return;
        pushdown(x), q[b++] = x;
        while (f != b) {
            st[top2++] = q[f];
            if (q[f]->c[0] != nil) q[b++] = q[f]->c[0];
            if (q[f]->c[1] != nil) q[b++] = q[f]->c[1];
            f++;
        }
    }
    Node* make_tree(int l, int r, Node* f) {
        if (l > r) return nil;
        int mid = (l + r) >> 1;
        Node* p = New(num[mid]);
        p->c[0] = make_tree(l, mid - 1, p);
        p->c[1] = make_tree(mid + 1, r, p);
        p->p = f;
        update(p);
        return p;
    }
    Node* join(Node* x, Node* y) {
        if (x == y) return x;
        if (x == nil) return y;
        if (y == nil) return x;
        splay(x = getMaxMin(x, 1), nil);
        splay(y = getMaxMin(y, 0), nil);
        if (x->p != nil) return y; //in the same tree
        x->c[1] = y, y->p = x;
        splay(y, nil);
        return y;
    }
    Node* split(Node* t) {
        if (t == nil) return nil;
        splay(t, nil);
        t->c[0]->p = t->c[1]->p = nil;
        Node* ret = joint(t->c[0], t->c[1]);
        t->c[0] = t->c[1] = nil;
        clear(t);
        return ret;
    }
};
```

2.3 动态树

```
1
2  int n;
3  struct Node
4  {
5      Node *c[2], *p;
6      int key, s;
7      bool rev;
8  } memo[MaxN], *nil, *pt[MaxN], *st[MaxN];
9  struct Splay {
10     void init(Node* x) {
11         x->c[0] = x->c[1] = x->p = nil;
12         x->key = x->s = 0;
13         x->rev = false;
14     }
15     void init() {
16         pt[0] = nil = &memo[0];
17         // nil->size=0;
18         for (int i = 1; i <= n; i++)
19             pt[i] = &memo[i], init(pt[i]);
20     }
21     void rotate(Node* x, int f) {
22         Node *y = x->p, *z = y->p;
```

```
23          x->p = z, y->p = x;
24          if (z != nil) { //can not conbine
25              if (y == z->c[0]) z->c[0] = x;
26              else if (y == z->c[1]) z->c[1] = x;
27          }
28          y->c[f ^ 1] = x->c[f];
29          if (y->c[f ^ 1] != nil) y->c[f ^ 1]->p = y;
30          x->c[f] = y;
31          update(y);
32          update(x);
33      }
34      void splay(Node *x) {
35          int top = 1;
36          st[0] = x;
37          for (Node *q = x; !isroot(q);)
38              st[top++] = (q = q->p);
39          while (top)
40              pushdown(st[--top]);
41          while (!isroot(x)) {
42              Node *y = x->p;
43              if (isroot(y))
44                  rotate(x, y->c[0] == x);
45              else {
46                  int t = (y == y->p->c[0]);
47                  if (x == y->c[t]) rotate(x, t ^ 1), rotate(x, t);
48                  else rotate(y, t), rotate(x, t);
49              }
50          }
51      }
52      void pushdown(Node* x) {
53          if (x == nil) return;
54          if (x->rev) {
55              reverse(x->c[0]);
56              reverse(x->c[1]);
57              x->rev = false;
58          }
59      }
60      void reverse(Node* x) {
61          if (x != nil) {
62              swap(x->c[0], x->c[1]);
63              x->rev ^= 1;
64          }
65      }
66      void update(Node* x) {
67          if (x == nil) return;
68          x->s = x->c[0]->s + x->key + x->c[1]->s;
69      }
70      bool isroot(Node* x) {
71          return x->p == nil || (x->p->c[0] != x && x->p->c[1] != x);
72      }
73      Node* expose(Node* x) {
74          // return the root
75          Node* y;
76          for (y = nil; x != nil; x = x->p)
77              splay(x), x->c[1] = y, update(y = x);
78          return y;
79      }
80      void set(Node* x, int v) {
81          x->key = v, splay(x);
82      }
83      Node* getRoot(Node* x) {
84          return head(expose(x));
85      }
86      Node* head(Node* x) {
87          if (x == nil) return nil;
88          while (x->c[0] != nil)
89              pushdown(x), x = x->c[0];
90          splay(x);
91          return x;
92      }
93      void getPath(Node* x, Node* y) {
94          //path: v => u => u->c[1]
95          //to calculate edge's weight, u can not be included
96          // the expose() operator can not be ignored
97          Node* ry = head(expose(y)), *rx = head(expose(x));
98          if (rx != ry) puts("impossible");
99          else {
100             for (Node* u = y, *v = nil; u != nil; u = u->p) {
```

```
                splay(u);
                if (u->p == nil) {
                    printf("%d\n", v->s + u->key + u->c[1]->s);
                    return;
                }
                u->c[1] = v, update(v = u);
            }
        }
    }
    void setRoot(Node* x) {
        reverse(expose(x));
    }
    bool merge(Node* x, Node* y) {
        // y is x's father in a rooted tree
        Node* ry = head(expose(y)), *rx = head(expose(x));
        if (rx == ry) return false;
        else {
            setRoot(x);
            splay(x), x->p = y;
            return true;
        }
    }
    void cut(Node* x) {
        splay(x);
        if (x->c[0] != nil) x->c[0]->p = x->p, x->p = x->c[0] = nil;
        else x->p = nil;
    }
    Node* LCA(Node* x, Node* y) {
        Node *rx = head(expose(x));
        Node *ey = expose(y), *ry = head(ey);
        if (rx == ry) return ey;
        else return nil;
    }
};
```

ol

2.4 左偏树

```
struct Node {
    Node *left, *right, *parent;
    int dist;
    T val;
} node[MaxN], *root[MaxN];
int st[MaxN], top, K;

void init(Node*& root) {
    K = 0;
    top = 0;
    root = NULL;
}

Node* New(size_t x) {
    Node* ret;

    if (top) {
        ret = &node[st[--top]];
        if (ret->left != NULL) {
            st[top++] = ret->left - node;
        }
        if (ret->right != NULL) {
            st[top++] = ret->right - node;
        }
    } else {
        ret = &node[K++];
    }

    ret->left = ret->right = NULL;
    ret->parent = NULL;
    ret->dist = 0;
    ret->val = x;

    return ret;
```

```
}

void Delete(Node* x) {
    st[top++] = x - node;
}

Node* Find(Node* x) {
    if (x->parent != NULL) {
        x->parent = Find(x->parent);
    }
    return x->parent;
}

Node* merge(Node* x, Node* y) {
    if (x == NULL)
        return y;
    if (y == NULL)
        return x;
    if (x->val > y->val) //min_heap
        swap(x, y);
    x->right = merge(x->right, y);
    x->right->parent = x;
    if (x->left == NULL || x->right != NULL && x->left->dist < x->right->dist)
        swap(x->left, x->right);
    if (x->right == NULL)
        x->dist = 0;
    else
        x->dist = x->right->dist + 1;
    return x;
}

Node* insert(Node*& root, T v) {
    root = merge(root, New(v));
    return root;
}

Node* insert(Node*& root, Node*& v) {
    root = merge(root, v);
    return root;
}

T min(Node* root) {
    return root->val;
}

Node* pop(Node*& root) {
    Node* l = root->left;
    Node* r = root->right;

    if (l != NULL)
        l->parent = NULL;
    if (r != NULL)
        r->parent = NULL;
    root->left = NULL;
    root->right = NULL;
    Delete(root);

    root = merge(l, r);
    return root;
}

void Del(Node* x) {
    Node* q = x->parent;
    Node* p = merge(x->left, x->right);
    p->parent = q;
    if (q != NULL && q->left == x)
        q->left = p;
    else if (q != NULL && q->right == x)
        q->right = p;

    while (q != NULL) {
        if (q->left->dist < q->right->dist)
            swap(q->left, q->right);
        if (q->left->dist == q->right->dist + 1)
            return;
        q->dist++;
        p = q;
        q = q->parent;
```

```
    }
}

int main() {
    char s[100];
    int id, x;

    for (int i = 0; i < 10; i++)
        init(root[i]);

    while (scanf("%s%d%d", s, &id, &x) == 3) {
        if (s[0] == 'A')
            insert(root[id], x);
        else if (s[0] == 'M') {
            printf("%d\n", min(root[id]));
            pop(root[id]);
        } else if (s[0] == 'U')
            insert(root[id], root[x]);
    }

    return 0;
}
```

2.5 划分树

```
int sorted[MaxN], tr[Log][MaxN], cot[Log][MaxN];
ll pre_sum[MaxN], sum[Log][MaxN];
int n, q;

void init(int d, int l, int r) {
    if (l == r)
        return;
    int mid = l + r >> 1;
    int same = mid - l + 1;
    for (int i = l; i <= r; i++)
        same -= tr[d][i] < sorted[mid];
    int lp = l, rp = mid + 1;
    for (int i = l; i <= r; i++) {
        if (tr[d][i] < sorted[mid]) {
            tr[d + 1][lp++] = tr[d][i];
            sum[d][i] = sum[d][i - 1] + tr[d][i];
        } else if (tr[d][i] == sorted[mid] && same) {
            tr[d + 1][lp++] = tr[d][i];
            same--;
            sum[d][i] = sum[d][i - 1] + tr[d][i];
        } else {
            tr[d + 1][rp++] = tr[d][i];
            sum[d][i] = sum[d][i - 1];
        }
        cot[d][i] = cot[d][l - 1] + lp - l;
    }
    init(d + 1, l, mid);
    init(d + 1, mid + 1, r);
}

int read(int d, int L, int H, int l, int r, int k, ll& s) {
    if (l == r)
        return tr[d][l];
    int cnt = cot[d][r] - cot[d][l - 1];
    int mid = L + H >> 1;
    if (cnt >= k) {
        int delta = cot[d][l - 1] - cot[d][L - 1];
        return read(d + 1, L, mid, L + delta, L + delta + cnt - 1, k, s);
    } else {
        s += sum[d][r] - sum[d][l - 1];
        int delta = (l - L) - (cot[d][l - 1] - cot[d][L - 1]);
        k -= cnt;
        cnt = r - l + 1 - cnt;
        return read(d + 1, mid + 1, H, mid + 1 + delta, mid + delta + cnt, k, s);
    }
}

void init() {
    scanf("%d", &n);
```

```
    for (int i = 1; i <= n; i++) {
        scanf("%d", &tr[0][i]);
        sorted[i] = tr[0][i];
        pre_sum[i] = pre_sum[i - 1] + tr[0][i];
    }
    sort(sorted + 1, sorted + n + 1);
    init(0, 1, n);
}

void sol() {
    int l, r;
    ll ret, s, mid;
    scanf("%d", &q);
    while (q--) {
        scanf("%d%d", &l, &r);
        ret = s = 0;
        mid = read(0, 1, n, l + 1, r + 1, (r - l) / 2 + 1, s);
        ret += ll(r - l) / 2 * mid - s;
        s = pre_sum[r + 1] - pre_sum[l] - s - mid;
        ret += s - ll(r - l + 1) / 2 * mid;
        printf("%lld\n", ret);
    }
}
```

2.6 Dancing Links

2.6.1 普通

```

struct Node {
    Node *l, *r, *d, *u;
    int row, col;
} memo[MAX * MAX], *cur, *hr[MAX], *hc[MAX];
int cnt[MAX], st[MAX], ans, nC, nR;

void removeColumn(Node* c) {
    c->r->l = c->l;
    c->l->r = c->r;
    for (Node* i = c->d; i != c; i = i->d)
        for (Node* j = i->r; j != i; j = j->r) {
            j->d->u = j->u;
            j->u->d = j->d;
            cnt[j->col]--;
        }
}

void resumeColumn(Node* c) {
    for (Node* i = c->u; i != c; i = i->u)
        for (Node* j = i->l; j != i; j = j->l) {
            j->u->d = j;
            j->d->u = j;
            cnt[j->col]++;
        }
    c->r->l = c;
    c->l->r = c;
}

bool dfsExactly(const int& k) {
    if (hc[0]->r == hc[0]) {
        printf("%d", k);
        for (int i = 0; i < k; i++)
            printf(" %d", st[i]);
        puts("");
        return true;
    }

    int s = oo;
    Node* c = 0;
    for (Node* i = hc[0]->r; i != hc[0]; i = i->r)
        if (cnt[i->col] < s) s = cnt[i->col], c = i;

    removeColumn(c);
    for (Node* i = c->d; i != c; i = i->d) {
        st[k] = i->row;
```

```
47          for (Node* j = i->r; j != i; j = j->r)
48              removeColumn(hc[j->col]);
49          if (dfsExactly(k + 1)) return true;
50          for (Node* j = i->l; j != i; j = j->l)
51              resumeColumn(hc[j->col]);
52      }
53      resumeColumn(c);
54
55      return false;
56  }
57
58  Node* New(int r, int c) {
59      cur->l = cur->r = cur->u = cur->d = cur;
60      cur->row = r, cur->col = c;
61      return cur++;
62  }
63
64  void init(int r, int c) {
65      nR = r;
66      nC = c;
67      cur = memo;
68      for (int i = 0; i <= nC; i++)
69          hc[i] = New(0, i);
70      for (int i = 0; i < nC; i++)
71          hc[i]->r = hc[i + 1];
72      for (int i = 1; i <= nC; i++)
73          hc[i]->l = hc[i - 1];
74      hc[0]->l = hc[nC], hc[nC]->r = hc[0];
75      for (int i = 0; i <= nC; i++)
76          cnt[i] = 0;
77      for (int i = 0; i <= nR; i++)
78          hr[i] = NULL;
79  }
80
81  void add(const int& r, const int& c) {
82      Node* p = New(r, c);
83      cnt[c]++;
84      p->u = hc[c];
85      p->d = hc[c]->d;
86      if (!hr[r]) hr[r] = p;
87      p->l = hr[r], p->r = hr[r]->r;
88      p->r->l = p->l->r = p->u->d = p->d->u = p;
89  }
90
91  void removeNode(Node* c) {
92      for (Node* i = c->d; i != c; i = i->d) {
93          i->r->l = i->l, i->l->r = i->r;
94          cnt[i->col]--;
95
96      }
97  }
98
99  void resumeNode(Node* c) {
100     for (Node* i = c->u; i != c; i = i->u) {
101         i->r->l = i->l->r = i;
102         cnt[i->col]++;
103     }
104 }
105
106 int F() {
107     bool vis[MAX] = { 0 };
108     int ret = 0;
109
110     while (1) {
111         int s = oo;
112         Node* c = NULL;
113         for (Node* i = hc[0]->r; i != hc[0]; i = i->r)
114             if (!vis[i->col] && cnt[i->col] < s) {
115                 s = cnt[i->col];
116                 c = i;
117                 if (s <= 1) break;
118             }
119         if (!c) break;
120         ret++;
121         vis[c->col] = true;
122         for (Node* j = c->d; j != c; j = j->d)
123             for (Node* k = j->r; k != j; k = k->r)
124                 vis[k->col] = true;
```

```
    }

    return ret;
}

bool dfsMult(const int& k) {
    if (k + F() > ans) return false;
    if (hc[0]->r == hc[0]) return true;
    int s = oo;
    Node* c = 0;
    for (Node* i = hc[0]->r; i != hc[0]; i = i->r)
        if (cnt[i->col] < s) {
            s = cnt[i->col];
            c = i;
            if (cnt[i->col] <= 1) break;
        }

    for (Node* i = c->d; i != c; i = i->d) {
        removeNode(i);
        for (Node* j = i->r; j != i; j = j->r)
            removeNode(j);
        if (dfsMult(k + 1)) return true;
        for (Node* j = i->l; j != i; j = j->l)
            resumeNode(j);
        resumeNode(i);
    }

    return false;
}
```

2.6.2 混合

```
struct Node {
    Node *l, *r, *d, *u;
    int row, col;
} memo[MAX * MAX], *cur, *hr[MAX], *hc[MAX];
int cnt[MAX], st[MAX], nK, ans, nC, nR;
bool mp[MAX][MAX];
int lst[MAX][MAX][2];

Node* New(int r, int c) {
    cur->l = cur->r = cur->u = cur->d = cur;
    cur->row = r, cur->col = c;
    return cur++;
}

void init(int r, int c) {
    nR = r;
    nC = c;
    cur = memo;
    for (int i = 0; i <= nC; i++)
        hc[i] = New(0, i);
    for (int i = 0; i < nC; i++)
        hc[i]->r = hc[i + 1];
    for (int i = 1; i <= nC; i++)
        hc[i]->l = hc[i - 1];
    hc[0]->l = hc[nC], hc[nC]->r = hc[0];
    for (int i = 0; i <= nC; i++)
        cnt[i] = 0;
    for (int i = 0; i <= nR; i++) {
        hr[i] = NULL;
        for (int j = 0; j <= nC; j++)
            mp[i][j] = false;
    }
}

void add(const int& r, const int& c) {
    if (mp[r][c]) return;
    mp[r][c] = true;
    Node* p = New(r, c);
    cnt[c]++;
    p->u = hc[c];
    p->d = hc[c]->d;
```

```
43      if (!hr[r]) hr[r] = p;
44      p->l = hr[r], p->r = hr[r]->r;
45      p->r->l = p->l->r = p->u->d = p->d->u = p;
46  }
47
48  void remove(Node* c) {
49      if (c->col <= nK) {
50          for (Node* i = c->d; i != c; i = i->d)
51              i->r->l = i->l, i->l->r = i->r;
52      } else {
53          c->l->r = c->r, c->r->l = c->l;
54          for (Node* i = c->d; i != c; i = i->d)
55              for (Node* j = i->r; j != i; j = j->r)
56                  j->d->u = j->u, j->u->d = j->d, cnt[j->col]--;
57      }
58  }
59
60  void resume(Node* c) {
61      if (c->col <= nK) {
62          for (Node* i = c->u; i != c; i = i->u)
63              i->r->l = i->l->r = i;
64      } else {
65          for (Node* i = c->u; i != c; i = i->u)
66              for (Node* j = i->l; j != i; j = j->l)
67                  j->d->u = j->u->d = j, cnt[j->col]++;
68          c->l->r = c->r->l = c;
69      }
70  }
71
72  int F() {
73      bool vis[MAX] = { 0 };
74      int ret = 0;
75
76      while (1) {
77          int s = oo;
78          Node* c = NULL;
79          for (Node* i = hc[0]->r; i->col <= nK && i != hc[0]; i = i->r)
80              if (!vis[i->col] && cnt[i->col] < s) {
81                  s = cnt[i->col];
82                  c = i;
83                  if (s <= 1) break;
84              }
85          if (!c) break;
86          ret++;
87          vis[c->col] = true;
88          for (Node* j = c->d; j != c; j = j->d)
89              for (Node* k = j->r; k != j; k = k->r)
90                  vis[k->col] = true;
91      }
92
93      return ret;
94  }
95
96  bool dfs(const int& k) {
97      if (k + F() > ans) return false;
98      if (hc[0]->r == hc[0] || hc[0]->r->col > nK) return true;
99      int s = oo;
100     Node* c = NULL;
101     for (Node* i = hc[0]->r; i != hc[0] && i->col <= nK; i = i->r)
102         if (cnt[i->col] < s) {
103             s = cnt[i->col];
104             c = i;
105             if (cnt[i->col] <= 1) break;
106         }
107     for (Node* i = c->d; i != c; i = i->d) {
108         remove(i);
109         for (Node* j = i->r; j != i; j = j->r)
110             if (j->col <= nK) remove(j);
111         for (Node* j = i->r; j != i; j = j->r)
112             if (j->col > nK) remove(hc[j->col]);
113         if (dfs(k + 1)) return true;
114         for (Node* j = i->l; j != i; j = j->l)
115             if (j->col > nK) resume(hc[j->col]);
116         for (Node* j = i->l; j != i; j = j->l)
117             if (j->col <= nK) resume(j);
118         resume(i);
119     }
120
```

```
121     return false;
122 }
123
124 int n, m[MAX];
125
126 int doit() {
127     ans = 0;
128     while (!dfs(0))
129         ans++;
130     return ans;
131 }
```

2.7 表达式求值

```
1  #include <cstdio>
2  #include <cstring>
3  #include <cmath>
4  #include <cstdlib>
5  #include <numeric>
6  #include <set>
7  #include <map>
8  #include <queue>
9  #include <stack>
10 #include <cctype>
11 #include <utility>
12 #include <string>
13 #include <vector>
14 #include <limits>
15 #include <algorithm>
16
17 using namespace std;
18
19 typedef long long ll;
20 const int MAX = 4096;
21 const int Left = 0;
22 const int Right = 1;
23 const ll rbound = numeric_limits<int>::max();
24 const ll lbound = numeric_limits<int>::min();
25
26 ll abs64(ll x) {
27     if (x < 0) return -x;
28     else return x;
29 }
30
31 struct Parser {
32     static map<string, int> pred;
33     static map<string, int> ass;
34     map<string, int> mp;
35     ll val[MAX];
36     const char* pbuf;
37
38     inline int getId(const string& s) {
39         if (mp.find(s) == mp.end()) {
40             int id = mp.size(); //Careful
41             mp[s] = id;
42         }
43         return mp[s];
44     }
45
46     inline static ll calc(const ll& lhs, const string& op, const ll& rhs) {
47         ll ret;
48         if (op == "+") ret = lhs + rhs;
49         else if (op == "-") ret = lhs - rhs;
50         else if (op == "*") ret = lhs * rhs;
51         else if (op == "/" || op == "%") {
52             if (rhs == 0) throw 1;
53             else if (op == "/") ret = lhs / rhs;
54             else ret = (lhs < 0 ? -1: 1) * abs64(lhs) % abs64(rhs); //Warming
55         }
56         else if (op == "&&") ret = lhs && rhs;
57         else if (op == "||") ret = lhs || rhs;
58         else if (op == "^") ret = npow(lhs, rhs);
59         else throw 0;    //Be careful
60         if (ret < lbound || ret > rbound) throw ret;
```

```
61          return ret;
62      }
63
64      inline static ll npow(const ll& lhs, const ll& rhs) {
65          ll ret = 1;
66
67          if (rhs < 0) throw 1;
68          if (lhs == 0 && rhs == 0) throw 1;
69          if (lhs == 0 || lhs == 1) return lhs;
70          if (lhs == -1) return 1 - (rhs & 1) * 2;
71
72          for (int i = 0; i < rhs; i++)
73              ret = calc(ret, "*", lhs);
74          return ret;
75      }
76
77      inline static int isop(int c) {
78          return ispunct(c) && c != '(' && c != ')';
79      }
80
81      Parser() {
82          addOp("||", 0, Left);
83          addOp("&&", 1, Left);
84          addOp("=", 2, Left);
85          addOp("+", 3, Left);
86          addOp("-", 3, Left);
87          addOp("*", 5, Left);
88          addOp("%", 5, Left);
89          addOp("/", 5, Left);
90          addOp("^", 6, Right);
91      }
92
93      void reset(const char* s)
94          pbuf = s;
95
96      inline static void addOp(const string& s, int p, int a) {
97          pred[s] = p;
98          ass[s] = a;
99      }
100
101     ll exp(int p) {
102         ll a = P();
103         const char* ptr;
104
105         while (isop(*pbuf)) {
106             string op = "";
107             ptr = pbuf;
108             op += *ptr++;
109             // while (isop(*ptr))
110                 // op += *ptr++;
111             if (pred.find(op) == pred.end()) throw 0;
112             if (pred[op] >= p) {
113                 pbuf = ptr;
114                 int q = pred[op];
115                 if (ass[op] == Left) q++;
116                 ll b = exp(q);
117                 a = calc(a, op, b);
118             } else {
119                 break;
120             }
121         }
122
123         return a;
124     }
125
126     ll P() {
127         ll ret = 0;
128
129         if (*pbuf == '-') {
130             pbuf++;
131             ll r = P();
132             r = -r;
133             if (r < lbound || r > rbound) throw 0;
134             return r;
135         } else if (*pbuf == '(') {
136             pbuf++;
137             ret = exp(0);
138             if (*pbuf++ != ')') throw 0;
```

```
139                 return ret;
140             } else if (isalnum(*pbuf)) {
141                 string r = "";
142                 while (isalnum(*pbuf))
143                     r = r + *pbuf++;
144                 if ( isdigit (r[0]) ) {
145                     ret = 0;
146                     int len = r.length();
147                     for (int i = 0; i < len; i++) {
148                         ret = ret * 10 + r[i] - '0';
149                         if (ret < lbound || ret > rbound) throw 0;
150                     }
151                     return ret;
152                 } else {
153                     return val[getId(r)];
154                 }
155             } else throw 0;
156         }
157 };
158
159 map<string, int> Parser::pred;
160 map<string, int> Parser::ass;
161
162 int main() {
163     static char buf[MAX], s[MAX];
164     const char* pbuf;
165     char* ptr;
166     ll ret;
167     int T, cas = 1;
168
169     scanf("%d", &T);
170     gets(buf);
171     while (T--) {
172         gets(buf);
173         printf ("Case␣%d:␣", cas++);
174         pbuf = buf;
175         ptr = s;
176         while (*pbuf) {
177             if (!isspace(*pbuf))
178                 *ptr++ = *pbuf;
179             pbuf++;
180         }
181         *ptr = 0;
182         try {
183             Parser parser;
184             parser.reset(pbuf = s);
185             ret = parser.exp(0);
186             if (*parser.pbuf) throw 2;
187             printf ("%lld\n", ret);
188         } catch (...) {
189             puts("ERROR!");
190         }
191     }
192
193     return 0;
194 }
```

3 图论

3.1 2-SAT

考虑对两个对立的集合，或者多个物品，给定了两个物品的关系，常见的关系有选了A就要选择B，选了A就不能选择B，A必须选择，那么我们可以根据这些条件，让后构造一个有向图，如果选了A就不能选择B，那么我们就从A向B'连边，注意B'表示的是不选B，这样拆点跑强连通，如果有一个物品A满足A和A'在一个强连通分量中的话就是无解，否则有存在解。

对于两个对立集合的话就直接建边两个对立的点一个表示选择A，另外的就表示选择B，如果是多个物品的话就拆点，表示选择或者不选择。

2-SAT 一般结合二分来考察，这是因为2-SAT 大部分问题都是判定性问题。

对于构造解问题，可以通过反向枚举连通分量，也就是反着进行拓扑序查询，对于没有标记的连通分量，里面的点都可以选，然后将其对立的点所在的连通分量标记了，这样一定可以得到一组解。

3.2 改点堆

```
struct Edge {
    int to, w, id;
    Edge* nxt;
} memo[E], *cur, *g[V], *pree[V];
int q[V], d[V], pre[V], n, m;
int h[V], pos[V], K;
int s, t;

inline void init () {
    for (int i = 1; i <= n; i++)
        g[i] = NULL;
    cur = memo;
}

inline void add(int u, int v, int w, int id) {
    cur->to = v;
    cur->w = w;
    cur->id = id;
    cur->nxt = g[u];
    g[u] = cur++;
}

inline void sink(int k) {
    while (k <= K) {
        int idx = k, ls = k << 1, rs = k << 1 | 1;
        if (ls <= K && d[h[ls]] < d[h[idx]]) idx = ls;
        if (rs <= K && d[h[rs]] < d[h[idx]]) idx = rs;
        if (idx == k) break;
        else {
            swap(h[idx], h[k]);
            pos[h[idx]] = idx, pos[h[k]] = k;
            k = idx;
        }
    }
}

inline int getMin() {
    int ret = h[1];
    pos[h[1]] = -1;
    swap(h[1], h[K]);
    if (!--K) pos[h[1]] = 1;
    sink(1);
    return ret;
}

inline void swim(int k) {
    while (k > 1) {
        int p = k >> 1;
        if (d[h[k]] < d[h[p]]) {
```

```
            swap(h[k], h[p]);
            pos[h[k]] = k;
            pos[h[p]] = p;
            k >>= 1;
        }
        else break;
    }
}

inline void add(int k) {
    h[++K] = k;
    pos[k] = K;
    swim(K);
}

inline int dijkstra(const int& id = -1) {
    int u, v;
    K = 0;

    fill(d + 1, d + n + 1, oo);
    fill(pos + 1, pos + n + 1, -1);

    d[s] = 0;
    add(s);
    while (K) {
        u = getMin();
        if (u == t) break;
        for (Edge* it = g[u]; it; it = it->nxt) {
            v = it->to;
            if (it->id == id) continue;
            if (d[v] - d[u] > it->w) {
                d[v] = d[u] + it->w;
                if (id == -1) {
                    pre[v] = u;
                    pree[v] = it;
                }
                if (~pos[v]) swim(pos[v]);
                else add(v);
            }
        }
    }

    return d[t];
}
```

3.3 双连通

```
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <algorithm>

using namespace std;

const int MaxN = 20005;
const int MaxE = 400005;
const int MaxQ = 10005;
const int MaxM = MaxE * 2 + MaxQ * 2;

struct Edge {
    int v, id;
    Edge *nxt;
} memo[MaxM], *cur, *g[MaxN], *head[MaxN], *query[MaxN], *st[MaxM];
bool used[MaxE], cut[MaxN], vst[MaxN], bridge[MaxE];
int id[MaxN], low[MaxN], bel[MaxM];
int ret[MaxQ], dep[MaxN], p[MaxN];
int N, M, pF, cnt, K, top;

void add(Edge* g[], int u, int v, int id) {
    cur->v = v, cur->id = id;
    cur->nxt = g[u], g[u] = cur++;
    bridge[id] = false;
}

```

```
28  void dfs(int u) {
29      int v, cot = 0;
30      id[u] = low[u] = ++K;
31      for (Edge* it = g[u]; it; it = it->nxt) {
32          v = it->v;
33          if (used[it->id]) continue;
34          used[it->id] = true;
35          st[top++] = it;
36          if (!id[v]) {
37              dfs(v);
38              cot++;
39              low[u] = min(low[u], low[v]);
40              if (low[v] >= id[u]) {
41                  Edge* ptr;
42                  do {
43                      ptr = st[--top];
44                      bel[ptr->id] = cnt;
45                  } while (ptr != it);
46                  cnt++;
47                  if (u != pF) cut[u] = true;
48                  if (low[v] > id[u]) {
49                      // find the bridge
50                      bridge[it->id] = true;
51                  }
52              }
53          } else {
54              low[u] = min(low[u], id[v]);
55          }
56      }
57      if (u == pF && cot > 1) cut[u] = true;
58  }
59
60  void dfs(int u, int fa) {
61      int v;
62      vst[u] = true;
63      p[u] = fa;
64      for (Edge* it = head[u]; it; it = it->nxt) {
65          v = it->v;
66          if (!vst[v]) dfs(v, u);
67      }
68  }
69
70  int find(int x) {
71      if (x != p[x]) return p[x] = find(p[x]);
72      else return x;
73  }
74
75  void sol(int u) {
76      int v;
77      p[u] = u;
78      vst[u] = true;
79
80      for (Edge* it = query[u]; it; it = it->nxt) {
81          v = it->v;
82          if (vst[v]) {
83              int lca = find(v);
84              ret[it->id] = (dep[u] + dep[v] - dep[lca] * 2) / 2;
85          }
86      }
87
88      for (Edge* it = g[u]; it; it = it->nxt) {
89          v = it->v;
90          if (!vst[v]) {
91              dep[v] = dep[u] + 1;
92              sol(v);
93              p[v] = u;
94          }
95      }
96  }
97
98  int main() {
99      int Q;
100
101     while (scanf("%d%d", &N, &M) == 2 && (N || M)) {
102         memset(g, 0, sizeof(g));
103         memset(head, 0, sizeof(head));
104         memset(query, 0, sizeof(query));
105         for (int i = 0; i < N; i++)
```

```
            cut[i] = false, id[i] = 0;
        top = K = 0;
        cur = memo;
        cnt = N;
        for (int i = 0; i < M; i++) {
            int u, v;
            scanf("%d%d", &u, &v);
            u--, v--;
            add(g, u, v, i);
            add(g, v, u, i);
            used[i] = false;
        }
        for (int i = 0; i < N; i++)
            if (!id[i]) dfs(pF = i);
        for (int i = 0; i < N; i++)
            if (cut[i]) {
                for (Edge* it = g[i]; it; it = it->nxt) {
                    add(head, bel[it->id], i, -1);
                    add(head, i, bel[it->id], -1);
                }
            }
        for (int i = 0; i < cnt; i++)
            vst[i] = false, p[i] = -1;
        for (int i = 0; i < cnt; i++)
            if (!vst[i]) dfs(i, -1);
        for (int i = 0; i < cnt; i++)
            g[i] = NULL;
        cur = memo;
        for (int i = 0; i < cnt; i++) {
            vst[i] = false;
            if (p[i] != -1) {
                add(g, p[i], i, -1);
            }
        }
        scanf("%d", &Q);
        for (int i = 0; i < Q; i++) {
            int u, v;
            scanf("%d%d", &u, &v);
            u = bel[u - 1];
            v = bel[v - 1];
            add(query, u, v, i);
            add(query, v, u, i);
        }
        for (int i = 0; i < cnt; i++)
            if (!vst[i]) {
                dep[i] = 0;
                sol(i);
            }
        for (int i = 0; i < Q; i++) {
            printf("%d\n", ret[i]);
        }
    }
    return 0;
}
```

3.4 Sap

```
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<algorithm>
using namespace std;
const int V=220;
const int En=200000;
const int oo=0x3f3f3f3f;
struct Edge{int num,ne,c;}e[En];
int d[V],p[V],pre[V],low[V];
int gap[V],cur[V];
int N,K,st,ed;
void add(int x,int y,int c)
{
    e[K].num=y;e[K].c=c;
    e[K].ne=p[x];p[x]=K++;
```

```
    e[K].num=x;e[K].c=0;
    e[K].ne=p[y];p[y]=K++;
}
int sap()
{
    int ret=0;
    bool fail;
    for(int i=0;i<=N;i++)
    {
        low[i]=gap[i]=d[i]=0;
        cur[i]=p[i];
    }
    low[st]=oo;gap[0]=N;int u=st;
    while(d[st]<N)
    {
        fail=true;
        for(int i=cur[u];i!=-1;i=e[i].ne)
        {
            int v=e[i].num;cur[u]=i;
            if(e[i].c&&d[u]==d[v]+1)
            {
                pre[v]=i;
                low[v]=min(low[u],e[i].c);u=v;
                if(u==ed)
                {
                    do
                    {
                        e[pre[u]].c-=low[ed];
                        e[pre[u]^1].c+=low[ed];
                        u=e[pre[u]^1].num;
                    }while(u!=st);
                    ret+=low[ed];
                }
                fail=false;break;
            }
        }
        if(fail)
        {
            gap[d[u]]--;
            if(!gap[d[u]])return ret;
            d[u]=N;
            for(int i=p[u];i!=-1;i=e[i].ne)
            if(e[i].c)d[u]=min(d[u],d[e[i].num]+1);
            gap[d[u]]++;cur[u]=p[u];
            if(u!=st)u=e[pre[u]^1].num;
        }
    }
    return ret;
}
```

3.5 KM二分图加权匹配

3.5.1 邻接表

```
const int V=1200;
const int En=21000;
const int oo=1000000000;
struct Edge{int num,ne,w;}e[En];
int p[V],K;
void add(int x,int y,int z)
{
    e[K].num=y;e[K].w=z;
    e[K].ne=p[x];p[x]=K++;
}
bool sx[V],sy[V];
int lx[V],ly[V],mat[V];
bool path(int u)
{
    sx[u]=true;
    for(int i=p[u];i!=-1;i=e[i].ne)
    {
        int v=e[i].num;
        if(!sy[v]&&lx[u]+ly[v]==e[i].w)
```

```
20          {
21              sy[v]=true;
22              if(mat[v]==-1||path(mat[v]))
23              {mat[v]=u;return true;}
24          }
25      }
26      return false;
27  }
28  int N;
29  int KM()
30  {
31      int i,j;
32      for(i=0;i<N;i++)
33      {
34          lx[i]=-oo;
35          for(j=p[i];j!=-1;j=e[j].ne)
36          lx[i]=max(lx[i],e[j].w);
37      }
38      for(i=0;i<N;i++)ly[i]=0,mat[i]=-1;
39      for(int u=0;u<N;u++)
40      while(1)
41      {
42          for(i=0;i<N;i++)sx[i]=0,sy[i]=0;
43          if(path(u))break;
44          int dx=oo;
45          for(i=0;i<N;i++)if(sx[i])
46          for(j=p[i];j!=-1;j=e[j].ne)
47          if(!sy[e[j].num])
48          dx=min(dx,lx[i]+ly[e[j].num]-e[j].w);
49          if(dx==oo)return -1;
50          for(i=0;i<N;i++)if(sx[i])lx[i]-=dx;
51          for(i=0;i<N;i++)if(sy[i])ly[i]+=dx;
52      }
53      int ret=0;
54      for(i=0;i<N;i++)ret+=lx[i]+ly[i];
55      return -ret;
56  }
57  int _,ca,n,m,i,x,y,z,te;
58  int main()
59  {
60      scanf("%d",&_);ca=0;
61      while(_--)
62      {
63          ca++;
64          scanf("%d%d",&n,&m);N=n;
65          for(i=0;i<n;i++)p[i]=-1;K=0;
66          for(i=0;i<m;i++)
67          {
68              scanf("%d%d%d",&x,&y,&z);
69              x--;y--;
70              add(x,y,-z);add(y,x,-z);
71          }
72          te=KM();printf("Case␣%d:␣",ca);
73          if(te==-1)puts("NO");
74          else printf("%d\n",te);
75      }
76  }
```

3.5.2 邻接矩阵

```
1   const int V=410;
2   int w[V][V],lx[V],ly[V],mat[V];
3   bool sx[V],sy[V];
4   int N,M;
5   bool path(int u)
6   {
7       sx[u]=true;
8       for(int v=0;v<M;v++)
9       if(!sy[v]&&lx[u]+ly[v]==w[u][v])
10      {
11          sy[v]=true;
12          if(mat[v]==-1||path(mat[v]))
13          {mat[v]=u;return true;}
14      }
```

```
    return false;
}
const int oo=1000000000;
int KM()
{
    int i,j;
    for(i=0;i<N;i++)
    {
        lx[i]=-oo;
        for(j=0;j<M;j++)
        lx[i]=max(lx[i],w[i][j]);
    }
    for(i=0;i<M;i++)ly[i]=0;
    for(i=0;i<M;i++)mat[i]=-1;
    for(int u=0;u<N;u++)
    while(1)
    {
        for(i=0;i<N;i++)sx[i]=0;
        for(i=0;i<M;i++)sy[i]=0;
        if(path(u))break;
        int dx=oo;
        for(i=0;i<N;i++)if(sx[i])
        for(j=0;j<M;j++)if(!sy[j])
        dx=min(dx,lx[i]+ly[j]-w[i][j]);
        for(i=0;i<N;i++)if(sx[i])lx[i]-=dx;
        for(i=0;i<M;i++)if(sy[i])ly[i]+=dx;
    }
    int ret=0;
    for(i=0;i<N;i++)ret+=lx[i];
    for(i=0;i<M;i++)ret+=ly[i];
    return ret;
}
```

3.6 上下界最小流

```
const int V=600;
const int En=50000;
const int oo=0x3f3f3f3f;
struct Edge
{
    int num,ne,c;
}e[En];
int p[V],K;
void add(int x,int y,int c)
{
    e[K].num=y;e[K].c=c;
    e[K].ne=p[x];p[x]=K++;
    e[K].num=x;e[K].c=0;
    e[K].ne=p[y];p[y]=K++;
}
int d[V],cur[V],low[V],pre[V],gap[V],pree[V];
int st,ed,N;
int sap()
{
    int ret=0;
    bool fail;
    memset(gap,0,sizeof(gap));
    memset(low,0,sizeof(low));
    memset(d,0,sizeof(d));
    for(int i=0;i<N;i++)cur[i]=p[i];
    gap[0]=N;low[st]=oo;int u=st;
    while(d[st]<N)
    {
        fail=true;
        for(int i=cur[u];i!=-1;i=e[i].ne)
        {
            int v=e[i].num;cur[u]=i;
            if(e[i].c&&d[u]==d[v]+1)
            {
                pre[v]=u;pree[v]=i;
                low[v]=min(low[u],e[i].c);u=v;
                if(u==ed)
                {
                    do
```

```
                        {
                            e[pree[u]].c-=low[ed];
                            e[pree[u]^1].c+=low[ed];
                            u=pre[u];
                        }while(u!=st);
                        ret+=low[ed];
                    }
                    fail =false;break;
                }
            }
            if( fail )
            {
                gap[d[u]]--;
                if(!gap[d[u]]) return ret;
                d[u]=N;
                for(int i=p[u];i!=-1;i=e[i].ne)
                if(e[i].c)d[u]=min(d[u],d[e[i].num]+1);
                gap[d[u]]++;cur[u]=p[u];
                if(u!=st)u=pre[u];
            }
        }
        return ret;
}
struct ELF
{
    int u,v,lo;
}b[En];
int n,m,lb[V],ts,tt;
void solve()
{
    N=n+4;ts=0;tt=n+1;
    st=n+2;ed=n+3;
    memset(lb,0,sizeof(lb));
    int i,u,v;
    for(i=0;i<N;i++)p[i]=-1;K=0;
    for(i=0;i<m;i++)
    {
        u=b[i].u;
        v=b[i].v;
        lb[v]+=b[i].lo;
        lb[u]-=b[i].lo;
        add(u,v,oo-b[i].lo);
    }
    for(i=1;i<=n;i++)
    {
        add(ts,i,oo);
        add(i,tt,oo);
    }
    for(i=0;i<n+2;i++)
    {
        if(lb[i]>0)add(st,i,lb[i]);
        else add(i,ed,-lb[i]);
    }
    int ans=sap();
    add(tt,ts,oo);
    printf("%d\n",sap());
}
int _,ca,i;
int main()
{
    scanf("%d",&_);ca=0;
    while(_--)
    {
        ca++;
        scanf("%d%d",&n,&m);
        for(i=0;i<m;i++)
        {
            scanf("%d%d%d",&b[i].u,&b[i].v,&b[i].lo);
        }
        printf("Case #%d: ",ca);
        solve();
    }
}
```

3.7 上下界最大流

```
const int V=1500;
const int En=900000;
const int inf=0x3f3f3f3f;
struct Edge
{
    int num,ne;
    int c;
}e[En];
int p[V],K;
void add(int x,int y,int c)
{
    e[K].num=y;e[K].c=c;
    e[K].ne=p[x];p[x]=K++;
    e[K].num=x;e[K].c=0;
    e[K].ne=p[y];p[y]=K++;
}
int d[V],pre[V],pree[V],gap[V],cur[V];
int N,st,ed;
int low[V];
int sap()
{
    int ret=0;
    bool fail;
    for(int i=0;i<=N;i++)
    {
        d[i]=0;
        gap[i]=0;
        cur[i]=p[i];
        low[i]=0;
    }
    low[st]=inf;gap[0]=N;int u=st;
    while(d[st]<N)
    {
        fail=true;
        for(int i=cur[u];i!=-1;i=e[i].ne)
        {
            int v=e[i].num;cur[u]=i;
            if(e[i].c&&d[u]==d[v]+1)
            {
                pre[v]=u;pree[v]=i;
                low[v]=min(low[u],e[i].c);u=v;
                if(u==ed)
                {
                    do
                    {
                        e[pree[u]].c-=low[ed];
                        e[pree[u]^1].c+=low[ed];
                        u=pre[u];
                    }while(u!=st);
                    ret+=low[ed];
                }
                fail=false;break;
            }
        }
        if(fail)
        {
            gap[d[u]]--;
            if(!gap[d[u]])return ret;
            d[u]=N;
            for(int i=p[u];i!=-1;i=e[i].ne)
            if(e[i].c)d[u]=min(d[u],d[e[i].num]+1);
            gap[d[u]]++;cur[u]=p[u];
            if(u!=st)u=pre[u];
        }
    }
    return ret;
}
int n,m,s,t;
struct Elf{int u,v,lo,up;}b[12000];
int lb[12000];
int doit()
{
    int i;
    N=n+2;st=n;ed=n+1;
```

```
    for(i=0;i<N;i++)p[i]=-1;K=0;
    for(i=0;i<n;i++)lb[i]=0;
    for(i=0;i<m;i++)
    {
        lb[b[i].u]-=b[i].lo;
        lb[b[i].v]+=b[i].lo;
        add(b[i].u,b[i].v,b[i].up-b[i].lo);
    }
    for(i=0;i<n;i++)
    {
        if(lb[i]>0)add(st,i,lb[i]);
        else add(i,ed,-lb[i]);
    }
    add(t,s,inf);
    int te=sap();
    for(i=p[st];i!=-1;i=e[i].ne)
    if(e[i].c!=0)return -1;
    st=s;ed=t;te=sap();
    return te;
}
```

3.8 全局最小割

```
using namespace std;
#define inf 100000000
bool visit[502],com[502];
int map[502][502],W[502],s,t;
int maxadj(int N,int V)
{
    int CUT;
    memset(visit,0,sizeof(visit));
    memset(W,0,sizeof(W));
    for(int i=0;i<N;i++)
    {
        int Num=0,Max=-inf;
        for(int j=0;j<V;j++)
        if(!com[j]&&!visit[j]&&W[j]>Max){Max=W[j];Num=j;}
        visit[Num]=true;s=t;t=Num;CUT=W[t];
        for(int j=0;j<V;j++)
        if(!com[j]&&!visit[j])W[j]+=map[Num][j];
    }
    return CUT;
}
int stoer(int V)
{
    int Mincut=inf;int N=V;
    memset(com,0,sizeof(com));
    for(int i=0;i<V-1;i++)
    {
        int Cut;s=0,t=0;
        Cut=maxadj(N,V);N--;
        if(Cut<Mincut)Mincut=Cut;
        com[t]=true;
        for(int j=0;j<V;j++)
        if(!com[j])
        {map[j][s]+=map[j][t];map[s][j]+=map[t][j];}
    }
    return Mincut;
}
```

3.9 最小树形图

```
const int V=1200;
const int En=2100000;
struct Elf{int u,v,len;}b[En];
const int oo=1000000000;
int ret;
int N,M,Root;//点数，边数，根，默认从开始0
int id[V],pre[V],cnt,vis[V];
int in[V];
```

```
bool TreeMST()
{
    ret=0;
    int i,u,v;
    while(1)
    {
        for(i=0;i<N;i++)
        in[i]=oo;
        memset(pre,-1,sizeof(pre));
        for(i=0;i<M;i++)
        {
            u=b[i].u;
            v=b[i].v;
            if(b[i].len<in[v]&&u!=v)
            {
                pre[v]=u;
                in[v]=b[i].len;
            }
        }
        for(i=0;i<N;i++)
        {
            if(i==Root)continue;
            if(pre[i]==-1)return false;
        }
        in[Root]=0;
        cnt=0;
        memset(id,-1,sizeof(id));
        memset(vis,-1,sizeof(vis));
        for(i=0;i<N;i++)
        {
            ret+=in[i];v=i;
            while(vis[v]!=i&&id[v]==-1&&v!=Root)
            {vis[v]=i;v=pre[v];}
            if(v!=Root&&id[v]==-1)
            {
                for(u=pre[v];u!=v;u=pre[u])
                id[u]=cnt;
                id[v]=cnt++;
            }
        }
        if(cnt==0)return true;
        for(i=0;i<N;i++)
        if(id[i]==-1)id[i]=cnt++;
        for(i=0;i<M;i++)
        {
            v=b[i].v;
            b[i].u=id[b[i].u];
            b[i].v=id[b[i].v];
            if(b[i].u!=b[i].v)
            b[i].len-=in[v];
        }
        N=cnt;
        Root=id[Root];
    }
    return true;
}
```

3.10 带花树

```
#include <stdio.h>
#include <string.h>
#include <algorithm>
#include <vector>
#define maxn 300
#define maxm 90010

using namespace std;

int match[maxn];                        //标记是否匹配
int st[maxn],aim[maxm],nxt[maxm],ln;    //边表
int q[maxn];                            //队列bfs
int level[maxn];                        //离根深度的奇偶性
vector<int> ar[maxn];                   //存每个点到根的路径
vector<int> a;                          //找到的一条增广路
```

```
int n;
void init ()
{
    for(int i=0;i<n;i++)st[i]=-1;ln=0;
}
void in_edge(int x,int y){
    aim[ln]=y;
    nxt[ln]=st[x];
    st[x]=ln++;
}
int lca(int p,int q){                    //求和的最近公共祖先pq
    int ret=0;
    while (ret<ar[p].size() && ret<ar[q].size() && ar[p][ret]==ar[q][ret]) ret++;
    return ret-1;
}
int FindAlterRoad(int sp){
    int qn=1;
    memset(level,-1,sizeof(level));
    level[q[0]=sp]=1;
    ar[sp].clear();
    ar[sp].push_back(sp);
    for (int p=0;p<qn;p++){
        int x=q[p];
        for (int i=st[x];i!=-1;i=nxt[i]){
            int u=aim[i];
            if (match[u]==u) continue;
            if (level[u]==-1){           //是未访问的点u
                if (match[u]==-1){       //是未匹配的u找到增广路,
                    a=ar[x];
                    a.push_back(u);
                    return 1;
                }else{                   //是已匹配的点u
                    int v=match[u];
                    if (level[v]!=-1) continue;
                    ar[v]=ar[x];
                    ar[v].push_back(u);
                    ar[v].push_back(v);
                    level[u]=0;
                    level[v]=1;
                    q[qn++]=v;
                }
            }else
            if (level[u]==1){            //和同为偶点ux形成花.
                int root=lca(u,x);
                vector<int> tmp=ar[x];
                for (int i=ar[u].size()-1;i>root;i--){
                    int y=ar[u][i];
                    tmp.push_back(y);
                    if (level[y]==0){
                        level[y]=1;
                        ar[y]=tmp;
                        level[y]=1;
                        q[qn++]=y;
                    }
                }
                tmp=ar[u];
                for (int i=ar[x].size()-1;i>root;i--){
                    int y=ar[x][i];
                    tmp.push_back(y);
                    if (level[y]==0){
                        level[y]=1;
                        ar[y]=tmp;
                        level[y]=1;
                        q[qn++]=y;
                    }
                }
            }
        }
    }
    return 0;
}
int MaximumMatch(){
    int ret=0;                           //最大匹配数
    memset(match,-1,sizeof(match));
    for (int i=0;i<n;i++)
        if (match[i]==-1)
            if (FindAlterRoad(i)){
                for (int i=0;i<a.size();i+=2){
```

```
94                          int u=a[i],v=a[i+1];
95                          match[u]=v;
96                          match[v]=u;
97                      }
98                      ret++;
99                  }else match[i]=i;
100     return ret;
101 }
```

4 动态规划

4.1 四边形不等式

```
int dp[MAX][MAX],sum[MAX],s[MAX][MAX],p[MAX];

/*
let k = (i+j)/2
w[i][j]=(p[k]-p[i])+(p[k]-p[i+1])+...+(p[k]-p[k-1])+(p[k+1]-p[k])+...+(p[j]-p[k])
=p[k]*(k-i)-p[k]*(j-k)+(p[k+1]+p[k+2]+...+p[j])-(p[i]+p[i+1]+...+p[k-1])
=p[k]*(2*k-i-j)+(sum[j]-sum[k])-(sum[k-1]-sum[i-1])
*/
int w(int i,int j)
{
    int k=(i+j)>>1;
    return p[k]*(2*k-i-j)+(sum[j]-sum[k])-(sum[k-1]-sum[i-1]);
}

int main()
{
    int n,k,st,ed,tmp;


    while(~scanf("%d%d",&n,&k))
    {
        for(int i=1; i<=n; i++)p[i]=get();
        for(int i=0; i<=k; i++)
        {
            for(int j=0; j<=n; j++)
            {
                dp[i][j]=oo;
            }
        }
        for(int i=0; i<=n; i++)s[0][i]=0;
        dp[0][0]=sum[0]=0;
        for(int i=1; i<=n; i++)sum[i]=sum[i-1]+p[i];
        for(int i=1; i<=k; i++)
        {
            for(int j=n; j>=i-1; j--)
            {
                st=s[i-1][j];
                if(j==n)ed=n;
                else ed=s[i][j+1];
                for(int k=st; k<=ed; k++)
                {
                    tmp=dp[i-1][k]+w(k+1,j);
                    if(tmp<dp[i][j])
                    {
                        dp[i][j]=tmp;
                        s[i][j]=k;
                    }
                }
            }
        }

        put(dp[k][n]);
        puts("");
    }

    return 0;
}
```

4.2 斜率优化

```
LL dp[500010],s[500010],a[500010];
LL f1(int x,int y)
{
    return (dp[x]-dp[y]+s[x]*s[x]-s[y]*s[y]);
}
LL f2(int x,int y)
{
```

```
    return 2*(s[x]-s[y]);
}
int n,m,i,h,t,now,q[500010];
int main()
{
    while(~scanf("%d%d",&n,&m))
    {
        s[0]=0;
        for(i=1; i<=n; i++)
        {
            scanf("%lld",&a[i]);
            s[i]=s[i-1]+a[i];
        }
        dp[0]=0;
        dp[1]=a[1]*a[1]+m;
        h=1;
        t=0;
        q[1]=1;
        q[0]=0;
        for(i=2; i<=n; i++)
        {
            while(h-t>0&&f1(q[t],q[t+1])>=s[i]*f2(q[t],q[t+1]))
                t++;
            now=q[t];
            dp[i]=dp[now]+(s[i]-s[now])*(s[i]-s[now])+m;
            while(h-t>0&&f1(i,q[h])*f2(q[h],q[h-1])<=f2(i,q[h])*f1(q[h],q[h-1]))
                h--;
            q[++h]=i;
        }
        printf("%lld\n",dp[n]);
    }
}
```

5 字符串

5.1 KMP

```
struct KMP
{
    static const int MaxN = 1005;
    char s[MaxN], t[MaxN];//t-pattern
    int F[MaxN];

    void build(char *s)
    {
        int m = strlen(s);
        F[0] = -1;
        for (int i = 1, j = -1; i <= m; i++)
        {
            while (j >= 0 && s[j] != s[i - 1])
                j = F[j];
            F[i] = ++j;
            if (s[i] == s[F[i]])
                F[i] = F[F[i]];
        }
    }
    int sol()
    {
        build(t);
        int n = strlen(s), m = strlen(t), match = 0;
        for (int i = 0, j = 0; i < n; i++)
        {
            while (j >= 0 && s[i] != t[j])
                j = F[j];
            ++j;
            if (j == m)
            {
                j = F[j];
                match++;
            }
        }
        return match;
    }
}kmp;
```

5.2 扩展KMP

```
void e_kmp(char *s, char *t, int *has, int *e_has)//是模式串t
{
    //是和自身匹配的结果，是和匹配的结果e_hasthasst
    int sp, p, mx, tn;//是以上的mx k + F[k] - 1 指针，也就是最大范围指针,是对于的下标spk
    for (sp = p = mx = 0; s[p] > 0; p++)//是中考虑的当前位置ps
    {
        //如果当前范围为 p ，或者由 s[p-sp] 得到的结果可以延伸到 mx 位置并有可能继续延伸
        if (mx == p || p + e_has[p - sp] >= mx)
        {
            for (tn = mx - p; s[mx] == t[tn]; tn++)//能继续匹配就延伸
                mx++;
            has[sp = p] = mx - p;//更新 sp 和 has[p]
            if (mx == p)//有可能 mx=p 且 has[p]=0 ，当前位置和的前缀一个字符都匹配不了t
                sp = mx = p + 1;//我们已考虑的范围还是要前移
        }
        else //没有超出当前范围，直接读取答案
            has[p] = e_has[p - sp];
    }
}
int main()
{
    gets(s);gets(t);
    t[tn] = -1;
    e_has[0] = tn;
    //t 和自身匹配的过程和 s 和 t 匹配的过程是完全相同的，因此可以调用同一个函数
    //但是 t 不能从起始位置和自己匹配，否则没法保证之前说的 k>0
    e_kmp(t + 1, t, e_has + 1, e_has);
```

```
    e_kmp(s, t, has, e_has);
}
```

5.3 最小表示法

```
int get_sub_string(char* s, int n, int f) {
    int ret, i = 0, j = 1, k = 0;
    int tmp;
    //f = -1 cycle min expression

    while (i < n && j < n && k < n) {
        tmp = s[i + k] - s[j + k];
        if (tmp == 0)
            k++;
        else {
            if (tmp * f < 0)
                i += k + 1;
            else
                j += k + 1;
            if (i == j)
                j++;
            k = 0;
        }
    }

    ret = min(i, j);
    char c = s[ret + n];
    s[ret + n] = 0;
    strcpy(t, s + ret);
    s[ret + n] = c;
    return ret + 1;
}
```

5.4 AC自动机

```
//自动机AC

const int NODE = 105;
const int CH = 26;
int chd[NODE][CH] , sz;
int word[NODE], fail[NODE], Que[NODE], sw[300];

const int Inf = 10000000;
const int MOD = 20090717;
char str [1005];
int dp[2][1<<12][NODE];
int N, M, K;

void Ins(char *a, int val)
{
    int p = 0;
    for(; *a ; a ++)
    {
        int c = sw[*a];
        if (!chd[p][c])
        {
            memset(chd[sz] , 0, sizeof(chd[sz]));
            word[sz] = 0;
            chd[p][c] = sz ++;
        }
        p = chd[p][c];
    }
    word[p] = (1<<val);
}

void AC()
{
    int *s = Que , *e = Que;
    for( int i = 0 ; i < CH ; i ++ )
        if (chd[0][i])
```

```
        {
            fail [ chd[0][i] ] = 0;
            *e++ = chd[0][i];
        }
    while(s != e)
    {
        int p = *s++;
        for(int i = 0; i < CH; i++)
        {
            if(chd[p][i])
            {
                int v = chd[p][i];
                *e++ = v;
                fail [v] = chd[fail [p]][i];
                word[v] |= word[fail [v]];
                //对word[v] 按word[ fail [v]] 里的内容进行处理
            }
            else
            {
                chd[p][i] = chd[fail [p]][i];
            }
        }
    }
}

int main()
{
    for(int i = 0; i < CH; i++)sw['a'+i] = i;
    fail [0] = 0;
    word[0] = 0;
    //下面两句每次都必须初始化
    memset(chd[0], 0, sizeof(chd[0]));
    sz = 1;

    Ins(str, i);
    AC();


    return 0;
}
```

5.5 后缀数组

```
/*后缀数组常见错误:

1， 求 h 数组的时候， h[rank[i]]=k 写成 h[i]=k;
2， DA 里面 x[sa[0]] = 0 写成 = ; 1
m 过大请改成
bool cmpX(int p1, int p2)
{
    return ax[p1] < ax[p2];
}
for(i = 0; i < n; i++)sa[i] = i;
sort(sa,sa+n,cmpX);
for(i = 0; i < n; i++)x[i] = r[i];
*/
//1，普通的带 rmq
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;

const int MaxN = 2010;
int ax[MaxN],sa[MaxN],yo[2][MaxN],Cnt[MaxN],Rank[MaxN],h[MaxN];
bool cmpSa(int *r, int i, int j, int len)
{
    return r[i]==r[j] && r[i+len]==r[j+len];
}
void DA(int *sa, int *r, int n, int m)
{
    int *x=yo[0], *y=yo[1], i,p,len;

    for(int i = 0; i < m; i++)Cnt[i] = 0;
```

```
    for(int i = 0; i < n; i++)Cnt[x[i]=r[i]]++;
    for(int i = 1; i < m; i++)Cnt[i] += Cnt[i-1];
    for(int i = n-1; i>=0; i--)
        sa[--Cnt[x[i]]] = i;

    for(len=1,p=0; p<n; m=p,len*=2)
    {
        for(i=n-len,p=0; i<n; i++)y[p++] = i;
        for(i=0; i < n; i++)
            if(sa[i] >= len)
                y[p++] = sa[i]-len;

        for(i = 0; i < m; i++)Cnt[i]=0;
        for(i = 0; i < n; i++)Cnt[x[y[i]]]++;
        for(i = 1; i < m; i++)Cnt[i] += Cnt[i-1];
        for(i = n-1; i >= 0; i--)
            sa[--Cnt[x[y[i]]]] = y[i];
        for(swap(x,y),x[sa[0]]=0,p=i=1; i<n; i++)
            x[sa[i]] = cmpSa(y,sa[i],sa[i-1],len)?(p-1):(p++);
    }
}

void calH(int *sa, int *r, int n)
{
    int j;
    for(int i = 0; i < n; i++)Rank[sa[i]] = i;
    for(int i = 0, k = 0; i < n; h[Rank[i]]=k, i++)
    {
        if(Rank[i]==0)continue;
        for((k?k--:k),j=sa[Rank[i]-1]; r[i+k]==r[j+k]; k++)
            ;
    }
}

#define two(i) (1<<(i))
char s[MaxN];
int rmq[20][MaxN], mm[MaxN];
void init(int n)
{
    int i;
    for(mm[0] = -1, i = 1; i <= n; i++)
        mm[i] = ((i & (i - 1))==0)? mm[i-1]+1 : mm[i-1];
    for(i = 0; i < n; i++) rmq[0][i] = h[i];
    for(i = 1; i < 20; i++)
        for(int j = 0; j < n; j++)
        {
            rmq[i][j] = rmq[i-1][j];
            if(j+two(i-1) < n)
                rmq[i][j] = min(rmq[i][j], rmq[i-1][j+two(i-1)]);
        }
}

int query(int L, int R)
{
    if(L>R)swap(L,R);
    L++;
    int k = mm[R-L+1];
    return min(rmq[k][L],rmq[k][R-two(k)+1]);
}

int main()
{
    int n;
    while(scanf("%s",s)==1)
    {
        n = strlen(s);
        for(int i = 0; i < n; i++)
            ax[i] = s[i];
        ax[n] = 200;
        for(int i = 0; i < n; i++)
            ax[i+n+1] = ax[n-1-i];
        int N = 2*n+2;
        ax[N-1] = 0;
        DA(sa,ax,N,250);
        calH(sa,ax,N);
        init(N);
        int ret = 0, id=0;
        for(int i = 0; i < n; i++)
```

```
        {
            int L = Rank[i], R = Rank[2*n+1-i];
            int t = query(L,R);
            if(ret < t*2)
                ret = t*2, id = i-t;
            R = Rank[2*n-i];
            t = query(L,R);
            if(ret < t*2-1)
                ret = t*2-1, id = i-t+1;
        }
        for(int i = id; i < id+ret; i++)
            printf("%c",s[i]);
        puts("");
    }

    return 0;
}


//2，略恶心的重复次数最多子串（要求字典序最小）
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;

const int MaxN = 200050;

int ax[MaxN],sa[MaxN],yo[2][MaxN],Cnt[MaxN],Rank[MaxN],h[MaxN];
bool cmpSa(int *r, int i, int j, int len)
{
    return r[i]==r[j] && r[i+len]==r[j+len];
}
void DA(int *sa, int *r, int n, int m)
{
    int *x=yo[0], *y=yo[1], i, len, p;

    for(i = 0; i < m; i++)Cnt[i] = 0;
    for(i = 0; i < n; i++)Cnt[x[i] = r[i]]++;
    for(i = 1; i < m; i++)Cnt[i] += Cnt[i-1];
    for(i = n-1; i >= 0; i--)sa[--Cnt[x[i]]] = i;

    for(len=1,p=0; p<n; m=p, len*=2)
    {
        for(p=0, i=n-len; i<n; i++) y[p++] = i;
        for(i = 0; i < n; i++)
            if(sa[i] >= len)
                y[p++] = sa[i]-len;

        for(i = 0; i < m; i++)Cnt[i] = 0;
        for(i = 0; i < n; i++)Cnt[x[y[i]]]++;
        for(i = 1; i < m; i++)Cnt[i] += Cnt[i-1];
        for(i = n-1; i >= 0; i--)
            sa[--Cnt[x[y[i]]]] = y[i];
        for(swap(x,y), p=i=1, x[sa[0]]=0; i<n; i++)
            x[sa[i]] = cmpSa(y,sa[i],sa[i-1],len)?(p-1):(p++);
    }
}

void calH(int *sa, int *r, int n)
{
    int j;
    for(int i = 0; i < n; i++)Rank[sa[i]] = i;
    for(int i = 0, k = 0; i < n; h[Rank[i]]=k, i++)
    {
        if(Rank[i]==0)continue;
        for((k?k--:k),j=sa[Rank[i]-1]; r[i+k]==r[j+k]; k++)
            ;
    }
}

char s[MaxN];
int N, mm[MaxN], rmq[2][20][MaxN];
#define two(i) (1<<(i))
void init()
{
    for(int i = 0; i < N; i++)
    {
        rmq[0][0][i] = h[i];
```

```
        rmq[1][0][i] = Rank[i];
    }
    for(int k = 0; k < 2; k++)
    for(int i = 1; i < 20; i++)
        for(int j = 0; j < N; j++)
        {
            rmq[k][i][j] = rmq[k][i-1][j];
            if(j+two(i-1) < N)
                rmq[k][i][j] = min(rmq[k][i][j], rmq[k][i-1][j+two(i-1)]);
        }
}

int query(int i, int L, int R)
{
    if(L>R)swap(L,R);
    if(i==0)L++;
    int k = mm[R-L+1];
    return min(rmq[i][k][L],rmq[i][k][R-two(k)+1]);
}

int main()
{
    mm[0] = -1;
    for(int i = 1; i < MaxN; i++)
        mm[i] = (i&(i-1))==0?(mm[i-1]+1):mm[i-1];

    int cas = 0;
    while(scanf("%s",s)==1)
    {
        if(s[0]=='#')break;
        int n = strlen(s);
        for(int i = 0; i < n; i++)
        {
            ax[i] = s[i];
            ax[i+n+1] = s[n-1-i];
        }
        ax[n] = 200;
        ax[2*n+1] = 0;
        N = 2*n+2;
        DA(sa,ax,N,300);
        calH(sa,ax,N);
        init();

        int mx = 0, id = 0, mxl = 0;
        for(int L = 1; L <= n; L++)
        {
            for(int i = 0; i+L < n; i+=L)
            {
                int t1 = query(0,Rank[i],Rank[i+L]);
                int t2 = query(0,Rank[2*n+1-i],Rank[2*n+1-(i+L)]);
                int x = t1+t2;
                int t = (t1+t2)/L+1;
                int tid = sa[query(1,i-t2,i-t2+x%L)];
                if(mx < t || (mx==t && Rank[tid]<Rank[id]))
                    mx = t, mxl = L, id = tid;
            }
        }
        printf("Case %d: ",++cas);
        for(int i = id; i < id+mx*mxl; i++)
            printf("%c",s[i]);
        puts("");
    }

    return 0;
}
```

5.6 后缀自动机

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <iostream>
#include <climits>
#include <numeric>
```

```
#define foreach(e,x) for(__typeof(x.begin()) e=x.begin();e!=x.end();++e)
#define REP(i,n) for(int i=0;i<n;++i)
using namespace std;
const int MAX_M = 9;

struct State {
    State*suf, *go[26], *next;
    int val;
    int l[MAX_M];
    State() :
            suf(0), next(0) {
        memset(go, 0, sizeof go);
        memset(l, 0, sizeof l);
    }
};

const int MAX_N = 100000 + 10;
State statePool[MAX_N * 2], *cur,*root,*last;
State* firstVal[MAX_N] = { };

State*newState(int val) {
    cur->val = val;
    cur->next = firstVal[val];
    firstVal[val] = cur;
    return cur++;
}

void init() {
    cur = statePool;
    root = last = newState(0);
}

void extend(int w) {
    State*p = last;
    State*np = newState(p->val + 1);
    while (p && !p->go[w])
        p->go[w] = np, p = p->suf;
    if (!p)
        np->suf = root;
    else {
        State*q = p->go[w];
        if (p->val + 1 == q->val) {
            np->suf = q;
        } else {
            State*nq = newState(p->val + 1);
            memcpy(nq->go, q->go, sizeof q->go);
            nq->suf = q->suf;
            q->suf = nq;
            np->suf = nq;
            while (p && p->go[w] == q)
                p->go[w] = nq, p = p->suf;
        }
    }
    last = np;
}

char buf[MAX_N];
int main() {
// freopen("in", "r", stdin);
    init();
    int L;
    scanf("%s", buf);
    L = strlen(buf);
    for (char*pt = buf; *pt; ++pt)
        extend(*pt - 'a');

    int id;
    for (id = 0; scanf("%s", buf) != EOF; ++id) {
        int l = 0;
        State*t = root;
        for (char*pt = buf; *pt; ++pt) {
            int w = *pt - 'a';
            while (t && !t->go[w]) {
                t = t->suf;
                l = t ? t->val : 0;
            }
            if (!t) {
                t = root;
```

```
                l = 0;
            } else {
                t = t->go[w];
                t->l[id] = max(t->l[id], ++l);
            }
        }
    }

    int ans = 0;
    for (int i = L; i >= 0; --i) {
        for (State*p = firstVal[i]; p; p = p->next) {
            int ret = p->val;
            for (int j = 0; j < id; ++j) {
                ret = min(ret, p->l[j]);
                if (p->suf)
                    p->suf->l[j] = max(p->suf->l[j], p->l[j]);
            }
            ans = max(ans, ret);
        }
    }

    printf("%d\n", ans);
    return 0;
}
```

5.7 线性回文

```
//线性回文
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
const int MaxN = 110050;
char s[MaxN];
char ss[MaxN*2];
int P[MaxN*2];

#define Min(a,b) ((a)<(b)?(a):(b))
#define Max(a,b) ((a)>(b)?(a):(b))

int main()
{
    while(scanf("%s",s)==1)
    {
        int n = strlen(s);
        for(int i = 0; i < n; i++)
        {
            ss[i*2] = '#';
            ss[i*2+1] = s[i];
        }
        ss[n*2] = '#';
        int ret = 0, a=0;
        P[0] = 0;
        for(int i = 1; i <= 2*n; i++)
        {
            if(2*a-i>=0 && P[2*a-i] < 2*a-i-(a-P[a]))P[i] = P[2*a-i];
            else
            {
                int st = 2*a-i-(a-P[a])+1;
                int ed = Min(i, n*2-i);
                int j = st;
                for(; j <= ed; j++)
                    if(ss[i-j] != ss[i+j])break;
                P[i] = j-1;
            }
            if(P[i]+i > a)a = i;
            ret = Max(ret, P[i]);
        }
        printf("%d\n",ret);
    }

    return 0;
}
```

6 计算几何

6.1 半平面交

```
struct Line
{
    pt a, b;
    double k, c; //k = dy/dx c = constant value)
    Line() { }
    Line(const pt& a, const pt& b) : a(a), b(b) { }
    bool operator<(const Line& l) const
    {
        if (dbcmp(k, l.k)) return k < l.k;
        else return c < l.c;
    }
    void set()
    {
        k = atan2(a.y - b.y, a.x - b.x);
        if (dbcmp(a.x, b.x)) c = (a * b) / fabs(a.x - b.x);
        else c = (a * b) / fabs(a.y - b.y);
    }
};

Line l[MAX], q[MAX];

bool add(double a, double b, double c, vector<Line>& l)
{
    //set half-panel ax + by < c
    if (sgn(c))
    {
        if (sgn(a) == 0 && sgn(b) == 0)
        {
            if (sgn(c) < 0)
                return false;
        }
        else if (sgn(b))
        {
            P p1(0, c / b);
            P p2(1, (c - a) / b);
            if (sgn(c / b) > 0)
            {
                if (c < 0)
                    l.push_back(Line(p1, p2));
                else
                    l.push_back(Line(p2, p1));
            }
            else
            {
                if (c > 0)
                    l.push_back(Line(p1, p2));
                else
                    l.push_back(Line(p2, p1));
            }
        }
        else
        {
            P p1(c / a, 0);
            P p2(c / a, 1);
            if (sgn(c / a) > 0)
            {
                if (c > 0)
                    l.push_back(Line(p1, p2));
                else
                    l.push_back(Line(p2, p1));
            }
            else
            {
                if (c < 0)
                    l.push_back(Line(p1, p2));
                else
                    l.push_back(Line(p2, p1));
            }
        }
    }
    else
```

```
72      {
73          if (sgn(a))
74          {
75              P p1(0, 0);
76              P p2(-b / a, 1);
77              if (a > 0)
78                  l.push_back(Line(p1, p2));
79              else
80                  l.push_back(Line(p2, p1));
81          }
82          else
83          {
84              P p1(0, 0);
85              P p2(-1, 0);
86              if (sgn(b))
87              {
88                  if (b > 0)
89                      l.push_back(Line(p1, p2));
90                  else
91                      l.push_back(Line(p2, p1));
92              }
93              else
94                  return false;
95          }
96      }
97
98      return true;
99  }
100
101 Polygon run(int n)
102 {
103     //Time: O(NlogN)
104     int f, b;
105     Polygon ret;
106
107     //the line vector must be anti-clockwise
108     //cut the right side and the left side gets left
109
110     if (n < 3)
111         return ret;
112
113     b = 1;
114     for (int i = 0; i < n; i++)
115         l[i].set();
116     sort(l, l + n);
117     for (int i = 1; i < n; i++)
118         if (dbcmp(l[i].k, l[i - 1].k))
119             l[b++] = l[i];
120     n = b;
121     f = b = 0;
122     q[b] = l[0];
123     q[++b] = l[1];
124     for (int i = 2; i < n; i++)
125     {
126         if (! line_ins (q[b], q[b - 1]) || ! line_ins (q[f], q[f + 1]))
127             return ret;
128         while (f != b && xmult(l[i].b, ins_point (q[b], q[b - 1]), l[i].a) < 0)
129             b--;
130         while (f != b && xmult(l[i].b, ins_point (q[f], q[f + 1]), l[i].a) < 0)
131             f++;
132         q[++b] = l[i];
133     }
134     while (f != b && xmult(q[f].b, ins_point (q[b], q[b - 1]), q[f].a) < 0)
135         b--;
136     while (f != b && xmult(q[b].b, ins_point (q[f], q[f + 1]), q[b].a) < 0)
137         f++;
138     if (b <= f + 1)
139         return ret;
140     for (int i = f; i < b; i++)
141         ret.push_back( ins_point (q[i], q[i + 1]));
142     if (f < b + 1)
143         ret.push_back( ins_point (q[f], q[b]));
144     Polygon:: iterator it = unique(ret.begin(), ret.end());
145     ret.erase( it, ret.end());
146     if (*ret.begin() == ret.back())
147         ret.pop_back();
148     return ret;
149 }
```

```
150
151 vector<pt> cut(const vector<pt>& vt, const Line& l)
152 {
153     //Time: O(N ^ 2)
154     vector<P> ret[3];
155     Line side;
156     P p;
157     int n, cur, pre;
158
159     ret[LEFT].clear();
160     ret[RIGHT].clear();
161     ret[ONLINE].clear();
162     n = vt.size();
163     if (n == 0) return vt;
164     pre = cur = relation(vt[0], l);
165
166     for (int i = 0; i < n; i++)
167     {
168         cur = relation(vt[(i + 1) % n], l);
169         if (cur == pre)
170             ret[cur].push_back(vt[(i + 1) % n]);
171         else
172         {
173             side.a = vt[i];
174             side.b = vt[(i + 1) % n];
175             p = ins_point(side, l);
176             ret[pre].push_back(p);
177             ret[cur].push_back(p);
178             ret[cur].push_back(vt[(i + 1) % n]);
179             pre = cur;
180         }
181     }
182
183     if(ret[LEFT].size() == 0u) return ret[ONLINE];
184     return ret[LEFT];
185 }
```

6.2 多边形相关

```
1  int insidePolygon(const Polygon& poly, P p)
2  {
3      // p inside simple polygon
4      // -1 - inside
5      // 0 - boarder
6      // 1 - outside
7      int rel;
8      Line ray, side;
9      int n = poly.size();
10     rel = 0;
11     ray.a = p;
12     ray.b.y = p.y;
13     ray.b.x = -oo;
14     for (int i = 0; i < n; i++)
15     {
16         side.a = poly[i];
17         side.b = poly[(i + 1) % n];
18         if (on_seg(p, side))
19             return 0;
20         if (dbcmp(side.a.y, side.b.y) == 0)
21             continue;
22         if (on_seg(side.a, ray))
23             if (dbcmp(side.a.y, side.b.y) > 0) rel++;
24         else if (on_seg(side.b, ray))
25             if (dbcmp(side.b.y, side.a.y) > 0) rel++;
26         else if (seg_ins(ray, side))
27             rel++;
28     }
29     return ((rel % 2 == 1) ? -1 : 1);
30 }
31
32 bool InsidePolygon(const Polygon& poly, Line L)
33 {
34     // seg in polygon
35     bool ret;
```

```
    Points pts;
    P p;
    Line side;
    ret = ((insidePolygon(poly, L.a) != 1) && (insidePolygon(poly, L.b) != 1));
    if (!ret) return false;
    int n = poly.size();
    for (int i = 0; i < n; i++)
    {
        side.a = poly[i];
        side.b = poly[(i + 1) % n];
        if (on_seg(L.a, side)) pts.push_back(L.a);
        else if (on_seg(L.b, side)) pts.push_back(L.b);
        else if (on_seg(side.a, L)) pts.push_back(side.a);
        else if (on_seg(side.b, L)) pts.push_back(side.b);
        else if (seg_ins(side, L)) return false;
    }
    sort(pts.begin(), pts.end());
    for (int i = 1; i < (int) pts.size(); i++)
    {
        if (pts[i - 1] != pts[i])
        {
            p.x = (pts[i - 1].x + pts[i].x) / 2.0;
            p.y = (pts[i - 1].y + pts[i].y) / 2.0;
            if (insidePolygon(poly, p) == 1)
            {
                return false;
            }
        }
    }
    return true;
}

P center(const Polygon& poly)
{
    P p, p0, p1, p2, p3;
    double m, m0;
    p1 = poly[0], p2 = poly[1];
    p.x = p.y = m = 0;
    for (int i = 2; i < (int) poly.size(); i++)
    {
        p3 = poly[i];
        p0 = (p1 + p2 + p3) / 3.0;
        m0 = p1 * p2 + p2 * p3 + p3 * p1;
        if (!sgn(m + m0)) m0 += eps;
        p = (p * m + p0 * m0) / (m + m0);
        m += m0, p2 = p3;
    }
    return p;
}
```

6.3 扫描线

6.3.1 建树

```
const int MaxN = 50005;
const double eps = 1e-8;

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

int dbcmp(const double& x, const double& y)
{
    return sgn(x - y);
}

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

enum
{
```

```
21      down, up
22  };
23  double tx; //now x
24  struct Circle
25  {
26      double x, y, r;
27      int ret;
28
29      void init()
30      {
31          scanf("%lf%lf%lf", &x, &y, &r);
32      }
33
34      double getY(const int& side)
35      {
36          double dy = sqrt(sqr(r) - sqr(tx - x));
37          if (side == up) return y + dy;
38          else return y - dy;
39      }
40  } c[MaxN];
41
42  struct Node
43  {
44      int id;
45      int side;
46      Node()
47      {
48      }
49      Node(const int& id, const int& side) :
50          id(id), side(side)
51      {
52      }
53      bool operator<(const Node& node) const
54      {
55          double y1 = c[id].getY(side);
56          double y2 = c[node.id].getY(node.side);
57          if (dbcmp(y1, y2)) return y1 < y2;
58          return side < node.side;
59      }
60  };
61  set<Node> st;
62  typedef set<Node>::iterator Ptr;
63
64  struct Event
65  {
66      double x;
67      int id;
68
69      Event()
70      {
71      }
72      Event(const double& x, const int& id) :
73          x(x), id(id)
74      {
75      }
76      bool operator<(const Event& e) const
77      {
78          return x < e.x;
79      }
80  } e[MaxN * 2];
81  bool U[MaxN];
82  int n, en;
83
84  int doit()
85  {
86      int ret = 1;
87      for (int i = 0; i < en; i++)
88      {
89          tx = e[i].x;
90          if (U[e[i].id]) st.erase(Node(e[i].id, down)), st.erase(Node(e[i].id, up));
91          else
92          {
93              Ptr it = st.insert(Node(e[i].id, down)).first;
94              Ptr l = it, r = it;
95              if (l-- == st.begin() || ++r == st.end()) c[e[i].id].ret = 1;
96              else
97              {
98                  if (r->id == l->id) c[it->id].ret = c[r->id].ret + 1;
```

```
99                  else c[it->id].ret = max(c[l->id].ret, c[r->id].ret);
100             }
101             st.insert(Node(e[i].id, up));
102             U[e[i].id] ^= 1;
103         }
104     }
105     for (int i = 0; i < n; i++)
106         ret = max(ret, c[i].ret);
107     return ret;
108 }
109
110 int run()
111 {
112
113     while (scanf("%d", &n) == 1)
114     {
115         en = 0;
116         //st.clear();
117         for (int i = 0; i < n; i++)
118         {
119             U[i] = false;
120             c[i].init();
121             e[en++] = Event(c[i].x - c[i].r, i);
122             e[en++] = Event(c[i].x + c[i].r, i);
123         }
124         sort(e, e + en);
125         printf("%d\n", doit());
126     }
127
128     return 0;
129 }
```

6.3.2 圆交

```
1  #include <cstdio>
2  #include <cstring>
3  #include <cstdlib>
4  #include <cmath>
5  #include <set>
6  #include <algorithm>
7
8  using namespace std;
9
10 const int MaxN = 50005;
11 const double eps = 1e-8;
12
13 double x[MaxN], y[MaxN], R[MaxN];
14 int left[MaxN], right[MaxN], up[MaxN], rank[MaxN];
15 double mid;
16
17 double sqr(double x)
18 {
19     return x * x;
20 }
21
22 int sgn(double x)
23 {
24     return x < -eps ? -1 : x > eps;
25 }
26
27 bool comp_left(const int& i, const int& j)
28 {
29     return x[i] - R[i] < x[j] - R[j];
30 }
31
32 bool comp_right(const int& i, const int& j)
33 {
34     return x[i] + R[i] < x[j] + R[j];
35 }
36
37 bool comp_rank(const int& i, const int& j)
38 {
39     if (sgn(y[i] - y[j])) return y[i] < y[j];
40     else return x[i] < x[j];
```

```
}

bool doit(const int& i, const int& j)
{
    return sgn(sqr(x[up[i]] - x[up[j]]) + sqr(y[up[i]] - y[up[j]]) - sqr(R[up[i]] + R[up[j]] + mid * 2.0)) < 0;
}

int n;
set<int> st;
typedef set<int>::iterator Ptr;

bool check()
{
    st.clear();
    int l = 0, r = 0, id;
    while (l < n || r < n)
    {
        if (r == n || (l != n && sgn((x[right[r]] + R[right[r]] + mid) - (x[left[l]] - R[left[l]] - mid)) >= 0))
        {
            Ptr it = st.insert(id = rank[left[l++]]).first;
            Ptr pl = it, pr = it;
            if (pl-- != st.begin()) if (doit(id, *pl)) return false;
            if (++pr != st.end()) if (doit(id, *pr)) return false;
        }
        else
        {
            st.erase(rank[right[r++]]);
        }
    }
    return true;
}

double doit()
{
    double l = 0, r = hypot(x[0] - x[1], y[0] - y[1]) - R[0] - R[1];
    for (int i = 0; i < n; i++)
        left[i] = right[i] = up[i] = i;
    sort(left, left + n, comp_left);
    sort(right, right + n, comp_right);
    sort(up, up + n, comp_rank);
    for (int i = 0; i < n; i++)
        rank[up[i]] = i;
//  for (int i = 0; i < 100; i++)
    while (r - l > eps)
    {
        mid = (l + r) / 2.0;
        if (check()) l = mid;
        else r = mid;
    }
    return l + r;
}

int run()
{
    int T;

    freopen("in.txt", "r", stdin);

    scanf("%d", &T);
    while (T--)
    {
        scanf("%d", &n);
        for (int i = 0; i < n; i++)
            scanf("%lf%lf%lf", &x[i], &y[i], &R[i]);
        printf("%.6f\n", doit());
    }

    return 0;
}
```

6.4 圆相关

```
double CommonArea(const Circle & A, const Circle & B)
{
```

```
    double s = 0.0;
    const Circle & M = (A.r > B.r) ? A : B;
    const Circle & N = (A.r > B.r) ? B : A;
    double D = (M.o - N.o).norm();
    if ((D < M.r + N.r) && (D > M.r - N.r))
    {
        double cosM = (M.r * M.r + D * D - N.r * N.r) / (2.0 * M.r * D);
        double cosN = (N.r * N.r + D * D - M.r * M.r) / (2.0 * N.r * D);
        double alpha = 2.0 * acos(cosM);
        double beta = 2.0 * acos(cosN);
        double TM = 0.5 * M.r * M.r * sin(alpha);
        double TN = 0.5 * N.r * N.r * sin(beta);
        double FM = (alpha / (2.0 * PI)) * area(M);
        double FN = (beta / (2.0 * PI)) * area(N);
        s = FM + FN - TM - TN;
    }
    else if (D <= M.r - N.r)
    {
        s = area(N);
    }
    return s;
}

int ins(const Circle& A, const Circle& B, pt &p1, pt &p2)
{
    double dis, ang, da, alpha;
    pt tp;
    tp = B.o - A.o;
    dis = (A.o - B.o).norm();
    ang = atan2(tp.y, tp.x);
    if(sgn(dis - A.r - B.r) > 0 || sgn(dis - fabs(A.r-B.r)) < 0)
        return 0;
    if(sgn(dis - A.r - B.r) == 0 || sgn(dis - fabs(A.r-B.r)) == 0)
    {
        p1 = A.o + pt(cos(ang), sin(ang)) * A.r;
        return 1;
    }
    da = acos(( sqr(A.r) + sqr(dis) - sqr(B.r) ) / ( 2 * A.r * dis ));
    alpha = ang + da;
    p1 = A.o + pt(cos(alpha), sin(alpha)) * A.r;
    alpha = ang - da;
    p2 = A.o + pt(cos(alpha), sin(alpha)) * A.r;
    return 2;
}

bool inCircle (const pt& p, const Circle& c)
{
    return dbcmp((p - c.o).norm(), c.r) <= 0;
}

bool SegCir(const Line& l, const Circle& c)
{
    // check whether segment intersects circle or not
    if ( inCircle (l.a, c) || inCircle (l.b, c))
    {
        return true;
    }
    if (dbcmp(Distance(c.o, l), c.r) > 0)
    {
        return false;
    }
    pt dir = l.b - l.a;
    swap(dir.x, dir.y);
    dir.x *= -1;
    Line line(c.o, c.o + dir);
    return xmult(line.a, l.a, line.b) * xmult(line.a, l.b, line.b) <= 0;
}

//Common area of circles and polygon
void add(const pt& u, const pt& v, const double& r, pt* p, int& n) {
    // cirlce (0,0) x-=xo...
    double a = (v - u) & (v - u);
    double b = 2.0 * ((v - u) & u);
    double c = (u & u) - r * r;
    double d = b * b - 4.0 * a * c;

    p[n++] = u;
    if (sgn(d) < 0) {
```

```
81          return;
82      }
83      d = sqrt(fabs(d));
84      double t1 = (-b + d) / (2 * a);
85      double t2 = (-b - d) / (2 * a);
86      if (t1 > t2) {
87          swap(t1, t2);
88      }
89      if (sgn(t1) > 0 && dbcmp(t1, 1) < 0) {
90          p[n++] = u + (v - u) * t1;
91      }
92      if (sgn(t2) > 0 && dbcmp(t2, 1) < 0 && dbcmp(t2, t1)) {
93          p[n++] = u + (v - u) * t2;
94      }
95  }
96
97  double area(const pt& u, const pt& v, const double& r) {
98      if (dbcmp(hypot((u.x + v.x) / 2.0, (u.y + v.y) / 2.0), r) < 0) {
99          return 0.5 * (u * v);
100     } else {
101         double t = atan2(v.y, v.x) - atan2(u.y, u.x);
102         while (t > PI)
103             t -= 2 * PI;
104         while (t < -PI)
105             t += 2 * PI;
106         return 0.5 * sqr(r) * t;
107     }
108 }
```

6.5 三维凸包

```
1   #include <cstdio>
2   #include <cstring>
3   #include <cstdlib>
4   #include <cmath>
5   #include <ctime>
6   #include <algorithm>
7
8   using namespace std;
9
10  const int MaxN = 505;
11  const int MaxF = MaxN * 4;
12  const double eps = 1e-7;
13
14  struct pt3
15  {
16      double x, y, z;
17
18      pt3()
19      {
20      }
21
22      pt3(double _x, double _y, double _z) :
23              x(_x), y(_y), z(_z)
24      {
25      }
26
27      pt3 operator-(const pt3& p1) const
28      {
29          return pt3(x - p1.x, y - p1.y, z - p1.z);
30      }
31
32      pt3 operator*(const pt3& p) const
33      {
34          return pt3(y * p.z - z * p.y, z * p.x - x * p.z, x * p.y - y * p.x);
35      }
36
37      double operator^(const pt3& p) const
38      {
39          return x * p.x + y * p.y + z * p.z;
40      }
41
42      void init ()
43      {
```

```
44          scanf("%lf%lf%lf", &x, &y, &z);
45      }
46  };
47
48  struct _3DCH
49  {
50
51      struct fac
52      {
53          int a, b, c;
54          bool ok;
55      };
56
57      int n;
58      pt3 P[MaxN];
59
60      int cnt;
61      fac F[MaxF];
62      int to[MaxN][MaxN];
63
64      double vlen(pt3 a)
65      {
66          return sqrt(a.x * a.x + a.y * a.y + a.z * a.z);
67      }
68
69      double area(pt3 a, pt3 b, pt3 c)
70      {
71          return vlen((b - a) * (c - a));
72      }
73
74      double volume(pt3 a, pt3 b, pt3 c, pt3 d)
75      {
76          return (b - a) * (c - a) ^ (d - a);
77      }
78
79      double ptof(pt3 &p, fac & f)
80      {
81          pt3 m = P[f.b] - P[f.a], n = P[f.c] - P[f.a], t = p - P[f.a];
82          return (m * n) ^ t;
83      }
84
85      void deal(int p, int a, int b)
86      {
87          int f = to[a][b];
88          fac add;
89          if (F[f].ok)
90          {
91              if (ptof(P[p], F[f]) > eps) dfs(p, f);
92              else
93              {
94                  add.a = b, add.b = a, add.c = p, add.ok = 1;
95                  to[p][b] = to[a][p] = to[b][a] = cnt;
96                  F[cnt++] = add;
97              }
98          }
99      }
100
101     void dfs(int p, int cur)
102     {
103         F[cur].ok = 0;
104         deal(p, F[cur].b, F[cur].a);
105         deal(p, F[cur].c, F[cur].b);
106         deal(p, F[cur].a, F[cur].c);
107     }
108
109     bool same(int s, int t)
110     {
111         pt3 &a = P[F[s].a], &b = P[F[s].b], &c = P[F[s].c];
112         return fabs(volume(a, b, c, P[F[t].a])) < eps && fabs(volume(a, b, c, P[F[t].b])) < eps && fabs(volume(a, b,
     c, P[F[t].c])) < eps;
113     }
114
115     void construct()
116     {
117         if (n < 4) return;
118
119         for (int i = 2; i < n; i++)
120         {
```

```
            if (vlen((P[0] - P[1]) * (P[1] - P[i])) > eps)
            {
                swap(P[2], P[i]);
                break;
            }
        }

        for (int i = 3; i < n; i++)
        {
            if (fabs((P[0] - P[1]) * (P[1] - P[2]) ^ (P[0] - P[i])) > eps)
            {
                swap(P[3], P[i]);
                break;
            }
        }

        srand(time(NULL));
        random_shuffle(P + 4, P + n);

        cnt = 0;
        fac add;
        for (int i = 0; i < 4; i++)
        {
            add.a = (i + 1) % 4, add.b = (i + 2) % 4, add.c = (i + 3) % 4, add.ok = 1;
            if (ptof(P[i], add) > 0) swap(add.b, add.c);
            to[add.a][add.b] = to[add.b][add.c] = to[add.c][add.a] = cnt;
            F[cnt++] = add;
        }

        for (int i = 4; i < n; i++)
        {
            for (int j = 0; j < cnt; j++)
            {
                if (F[j].ok && ptof(P[i], F[j]) > eps)
                {
                    dfs(i, j);
                    break;
                }
            }
        }
        int tmp = cnt;
        cnt = 0;
        for (int i = 0; i < tmp; i++)
        {
            if (F[i].ok)
            {
                F[cnt++] = F[i];
            }
        }
    }

    // surface area
    double area()
    {
        double ret = 0.0;
        for (int i = 0; i < cnt; i++)
        {
            ret += area(P[F[i].a], P[F[i].b], P[F[i].c]);
        }
        return ret / 2.0;
    }

    double volume()
    {
        pt3 O(0, 0, 0);
        double ret = 0.0;
        for (int i = 0; i < cnt; i++)
        {
            ret += volume(O, P[F[i].a], P[F[i].b], P[F[i].c]);
        }
        return fabs(ret / 6.0);
    }

    int facetCnt_tri ()
    {
        return cnt;
    }
```

```
199     int facetCnt()
200     {
201         int ans = 0;
202         for (int i = 0; i < cnt; i++)
203         {
204             bool nb = 1;
205             for (int j = 0; j < i; j++)
206             {
207                 if (same(i, j))
208                 {
209                     nb = 0;
210                     break;
211                 }
212             }
213             ans += nb;
214         }
215         return ans;
216     }
217 };
218
219 int run()
220 {
221     _3DCH hull;
222     while (scanf("%d", &hull.n) == 1)
223     {
224         for (int i = 0; i < hull.n; i++)
225             hull.P[i].init();
226         hull.construct();
227         printf("%.3f\n", hull.area());
228     }
229     return 0;
230 }
```

6.6 三维变换

```
1  #include <cstdio>
2  #include <cstring>
3  #include <cstdlib>
4  #include <cmath>
5  #include <algorithm>
6
7  using namespace std;
8
9  const double eps = 1e-6;
10 const double pi = acos(-1.0);
11
12 struct Mat {
13     double a[4][4];
14     int r, c;
15     Mat() {
16         r = c = 0;
17     }
18     Mat(int r, int c) :
19             r(r), c(c) {
20         for (int i = 0; i < r; i++)
21             for (int j = 0; j < c; j++)
22                 a[i][j] = 0;
23     }
24     Mat operator*(const Mat& other) const {
25         Mat ret(r, other.c);
26         for (int i = 0; i < r; i++)
27             for (int j = 0; j < other.c; j++)
28                 for (int k = 0; k < c; k++)
29                     ret.a[i][j] += a[i][k] * other.a[k][j];
30         return ret;
31     }
32     Mat pow(int k) {
33         Mat ret(r, c);
34         Mat a = *this;
35         for (int i = 0; i < r; i++)
36             ret.a[i][i] = 1.0;
37
38         while (k) {
39             if (k & 1)
```

```
40                  ret = ret * a;
41              a = a * a;
42              k >>= 1;
43          }
44
45          return ret;
46      }
47  } A;
48
49  Mat rot(double x, double y, double z, double theta) {
50      Mat ret(4, 4);
51      double a[3];
52      double cosA, sinA, s;
53      theta *= pi / 180.0;
54      s = sqrt(x * x + y * y + z * z);
55      cosA = cos(theta);
56      sinA = sin(theta);
57      x /= s;
58      y /= s;
59      z /= s;
60      a[0] = x;
61      a[1] = y;
62      a[2] = z;
63      for (int i = 0; i < 4; i++)
64          ret.a[i][i] = 1.0;
65
66      for (int i = 0; i < 3; i++) {
67          ret.a[i][i] = (1 - cosA) * a[i] * a[i] + cosA;
68          ret.a[i][(i + 2) % 3] = (1 - cosA) * a[i] * a[(i + 2) % 3]
69                  + a[(i + 1) % 3] * sinA;
70          ret.a[i][(i + 1) % 3] = (1 - cosA) * a[i] * a[(i + 1) % 3]
71                  - a[(i + 2) % 3] * sinA;
72      }
73
74      return ret;
75  }
76
77  Mat dfs(int k) {
78      double x, y, z, theta;
79      char op[5];
80      Mat ret(4, 4), tmp;
81
82      for (int i = 0; i < 4; i++)
83          ret.a[i][i] = 1.0;
84      while (scanf("%s", op) == 1) {
85          if (op[0] == 'e')
86              break;
87          tmp = Mat(4, 4);
88          for (int i = 0; i < 4; i++)
89              tmp.a[i][i] = 1.0;
90          if (op[0] == 'r') {
91              if (op[1] == 'e') {
92                  int t;
93                  scanf("%d", &t);
94                  tmp = dfs(t);
95              } else {
96                  // 轴 (x,y,z) ，角度 theta ，逆时针
97                  scanf("%lf%lf%lf%lf", &x, &y, &z, &theta);
98                  tmp = rot(x, y, z, theta);
99              }
100         } else if (op[0] == 't') {
101             // (x, y, z) -> (x + a[0][3], y + a[1][3], z + a[2][3])
102             for (int i = 0; i < 3; i++)
103                 scanf("%lf", &tmp.a[i][3]);
104         } else {
105             // (x, y, z) -> (a[0][0] * x, a[1][1] * y, a[2][2] * z)
106             for (int i = 0; i < 3; i++)
107                 scanf("%lf", &tmp.a[i][i]);
108         }
109         ret = tmp * ret;
110     }
111
112     return ret.pow(k);
113 }
114
115 void doit(double x, double y, double z) {
116     Mat ret(4, 1);
117     ret.a[0][0] = x;
```

```
118     ret.a[1][0] = y;
119     ret.a[2][0] = z;
120     ret.a[3][0] = 1.0;
121     ret = A * ret;
122     printf("%.2f %.2f %.2f\n", ret.a[0][0] + eps, ret.a[1][0] + eps,
123             ret.a[2][0] + eps);
124 }
125
126 int main() {
127     int n;
128     double x, y, z;
129
130     while (scanf("%d", &n) == 1 && n) {
131         A = dfs(1);
132         while (n--) {
133             scanf("%lf%lf%lf", &x, &y, &z);
134             doit(x, y, z);
135         }
136         puts("");
137     }
138
139     return 0;
140 }
```

7 数学

7.1 数论相关

```
void getphi()
{
    for(int i=2;i<=N;i++)
    {
        if (!ph[i]) {p[np++]=i;ph[i]=i-1;}
        for(int j=0;p[j]<=N/i;j++)
        {
            ph[i*p[j]]=ph[i]*(i%p[j]?p[j]-1:p[j]);
            if (i%p[j]==0) break;
        }
    }
}

void exgcd(int a,int b,int &x,int &y)
{
    if(b) exgcd(b,a%b,y,x),y-=x*(a/b);
    else x=1,y=0;
}
int inv(int a,int m)
{
    int x,y;
    exgcd(a,m,x,y);
    return (x+m)%m;
}
```

7.2 模线性方程组

```
const int MAX=1200;
typedef long long LL;
LL a[MAX],b[MAX];
LL x,y;

LL gcd(LL a,LL b)
{
    if(b==0)
    {
        x=1;
        y=0;
        return a;
    }
    LL ret=gcd(b,a%b);
    LL t=x;
    x=y;
    y=t-a/b*y;
    return ret;
}

bool run(LL ai,LL bi,LL aj,LL bj,LL& a,LL& b)
{
    LL g=gcd(bi,bj);
    LL mod;
    b=bi/g*bj;
    mod=bj/g;
    if((aj-ai)%g) return false;
    a=ai+bi*((aj-ai)/g*x%mod)%b;
    a=(a%b+b)%b;
    return true;
}

bool run(int n,LL& ret)
{
    LL aa,bb;
    if(n==1)
    {
        ret=a[0];
        return true;
    }
```

```
    else
    {
        if (!run(a[0], b[0], a[1], b[1], aa, bb))
        {
            return false;
        }
        for(int i=2; i<n; i++)
        {
            if (!run(aa, bb, a[i], b[i], aa, bb))
            {
                return false;
            }
        }
        ret=aa;
        return true;
    }
}

int main()
{
    int n;
    LL ret;

    while(~scanf("%d",&n))
    {
        for(int i=0; i<n; i++)
        {
            scanf("%I64d%I64d",&b[i],&a[i]);
            a[i]%=b[i];
        }
        if (run(n, ret))
        {
            printf ("%I64d\n",ret);
        }
        else
        {
            puts("-1");
        }

    }

    return 0;
}
```

7.3 行列式

```

/*
 * 行列式，模数为质数可用逆元
 */

#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;

typedef long long LL;
const int MaxN = 610;
int n,m,g[MaxN][MaxN],in[MaxN],out[MaxN],idx[MaxN];
int K,st[MaxN],ed[MaxN];
LL mod, dp[MaxN], a[MaxN][MaxN];

LL sol(int u)
{
    if (dp[u] != -1)return dp[u];
    LL &ret = dp[u];
    ret = 0;
    for(int i = 0; i < n; i++)
        if (g[i][u])
        {
            ret = (ret+g[i][u]*sol(i)%mod)%mod;
        }
    return ret;
}
```

```
LL Pow(LL x, LL n)
{
    LL ret = 1;
    while(n>0)
    {
        if (n&1)ret = ret*x%mod;
        x = x*x%mod;
        n/=2;
    }
    return ret;
}
LL cal()
{
    LL ret = 1;
    int neg = 0;
    for(int i = 0; i < K; i++)
    {
        int j = i;
        for(; j<K && a[j][i]==0; j++);
        if (j==K)return 0;
        if (j!=i)
        {
            neg ^= 1;
            for(int k = 0; k < K; k++)
                swap(a[i][k], a[j][k]);
        }
        for(j = i+1; j < K; j++)
        {
            LL c = (mod-a[j][i])*Pow(a[i][i], mod-2)%mod;
            for(int k = i; k < K; k++)
                a[j][k] = (a[j][k]+c*a[i][k]%mod)%mod;
        }
        ret = ret*a[i][i]%mod;
    }
    if (neg)ret = (mod-ret)%mod;
    return ret;
}

int main()
{
    scanf("%d%d%I64d",&n,&m,&mod);
    for(int i = 0; i < m; i++)
    {
        int a,b;
        scanf("%d%d",&a,&b);a--;b--;
        g[a][b]++;
        in[b]++; out[a]++;
    }
    int p1=0,p2=0;
    for(int i = 0; i < n; i++)
    {
        if (!in[i])
            st[p1++] = i;
        if (!out[i])
            ed[p2++] = i;
    }
    K = p1;
    for(int i = 0; i < K; i++)
    {
        for(int j = 0; j < n; j++)dp[j] = -1;
        dp[st[i]] = 1;
        for(int j = 0; j < K; j++)
            a[i][j] = sol(ed[j]);
    }
    LL ret = cal();
    printf("%I64d\n",ret%mod);

    return 0;
}
/*
 * 行列式，模数不一定为质数，辗转相减
 */
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;

typedef long long LL;
```

```
107 int n,P;
108 LL a[210][210];
109
110 int main()
111 {
112     while(scanf("%d%d",&n,&P)==2)
113     {
114         for(int i = 0; i < n; i++)
115             for(int j = 0; j < n; j++)
116                 scanf("%lld",&a[i][j]);// a[i][j]%=P;
117         LL ret = 1;
118         for(int i = 0; i < n; i++)
119         {
120             for(int j = i+1; j < n; j++)
121             {
122                 while(a[j][i])
123                 {
124                     LL t = a[i][i]/a[j][i];
125                     if(t)
126                     {
127                         for(int k = i; k < n; k++)
128                             a[i][k] = (a[i][k]-t*a[j][k])%P;
129                     }
130                     for(int k = i; k < n; k++)
131                         swap(a[i][k], a[j][k]);
132                     ret = -ret;
133                 }
134             }
135             if(a[i][i]==0){ret=0;break;}
136             ret = ret*a[i][i]%P;
137         }
138         printf("%lld\n",(ret+P)%P);
139     }
140
141     return 0;
142 }
```

7.4 高斯消元

```
1  #include <cstdio>
2  #include <cstdlib>
3  #include <cstring>
4  #include <cmath>
5  #include <algorithm>
6
7  using namespace std;
8
9  const double eps = 1e-8;
10 const int MaxN = 50;
11 double memo[MaxN][MaxN], *a[MaxN];
12 int n;
13 //N * N 的矩阵， memo[0 .. N - 1][0 .. N - 1] 是系数, a[0 .. N][N] 是常数项
14 //如果要M * N 的矩阵的话，自己改改吧，如果是整数上的高斯消元，比如说异或
15 //版的，照着改一下就行了
16
17
18 double Gauss() {
19     double ans[MaxN] = {0};
20     double tp;
21     int r, c;
22
23     for (int i = 0; i < n; i++)
24         a[i] = memo[i];
25
26     for (r = c = 0; r < n && c < n; r++, c++) {
27         for (int i = r + 1; i < n; i++)
28             if (fabs(a[i][c]) > fabs(a[r][c]))
29                 swap(a[i], a[r]);
30
31         if (fabs(a[r][c]) < eps) {
32             r--;
33             continue;
34         }
35         for (int i = r + 1; i < n; i++) {
```

```
            tp = a[i][c] / a[r][c];
            if (fabs(tp) < eps)
                continue;
            for (int j = c; j <= n; j++)
                a[i][j] -= tp * a[r][j];
        }
    }

    for (int i = r; i < n; i++)
        if (fabs(a[i][n]) > eps)
            return -1; //无解，如果答案有可能是 -1 的话，这里返回值要改，可以用一个
                       //全局变量来表示是否有解
    for (r--, c--; r >=0 && c >= 0; r--, c--) {
        while (fabs(a[r][c]) < eps)
            c--;
        for (int i = c + 1; i < n; i++)
            a[r][n] -= a[r][i] * ans[i];
        ans[c] = a[r][n] / a[r][c];
    }
    return ans[0];   //默认返回的值x0
}

int main() {
    return 0;
}
```

7.5 Miller Rabin

```
const int Mtime = 12;
typedef unsigned long long ull;
using namespace std;

ull pfac[1005];
int npf;

ull gcd(ull a, ull b)
{
    if (b) return gcd(b, a % b);
    else return a;
}
ull fmul(ull a, ull b, ull m)
{
    ull t = a, ans = 0;
    while (b)
    {
        if (b & 1) ans = (ans + t) % m;
        t = (t + t) % m;
        b >>= 1;
    }
    return ans;
}
ull fexp(ull a, ull b, ull m)
{
    ull t = a, ans = 1;
    while (b)
    {
        if (b & 1) ans = fmul(ans, t, m);
        t = fmul(t, t, m);
        b >>= 1;
    }
    return ans;
}
ull Pollard(ull n)
{
    ull x, y, d, c, i;
    while (1)
    {
        c = rand() % n;
        for (x = 1, i = 0; i < 3; i++) x = x * rand() % n;
        y = x;
        for (i = 2; ; i++)
        {
            x = (fmul(x, x, n) + c) % n;
            d = gcd(n, n + y - x);
```

```
47              if (d != 1 && d != n) return d;
48              if (y == x) break;
49              if (i == (i & -i)) y = x;
50          }
51      }
52  }
53  bool Miller(ull n, int t = Mtime)
54  {
55      if (n < 2) return false; // Warning
56      ull p2 = (n - 1) & (1 - n);
57      ull u = (n - 1) / p2;
58      if (n == 2 || n == 3 || n== 5 || n == 7) return false;
59      if (n % 2 == 0 || n % 3 == 0 || n % 5 == 0 || n % 7 == 0) return true;
60      for (int i = 0; i < t; i++)
61      {
62          ull a = rand() % (n - 1) + 1;
63          ull x = fexp(a, u, n);
64          for (ull j = 1; j < p2; j <<= 1)
65          {
66              ull nx = fmul(x, x, n);
67              if (nx == 1 && x != 1 && x != n - 1) return true;
68              x = nx;
69          }
70          if (x != 1) return true;
71      }
72      return false; //Be carefull
73  }
74  void getpfac(ull n)
75  {
76      if (! Miller(n))
77      {
78          pfac[npf++] = n;
79          return;
80      }
81      ull p = Pollard(n);
82      getpfac(n / p);
83      getpfac(p);
84  }
```

7.6 离散对数

```
1   //离散对数
2   #include <cstdio>
3   #include <cstring>
4   #include <cmath>
5   #include <algorithm>
6   using namespace std;
7
8
9   typedef long long LL;
10
11  struct Hash
12  {
13      static const int MOD = 100007;
14      static const int MaxN = 100005;
15      struct Node
16      {
17          LL k, v;//A^k = v
18          Node *nxt;
19      }buf[MaxN], *g[MaxN], *pt;
20      void init ()
21      {
22          memset(g,0,sizeof(g));
23          pt = buf;
24      }
25      LL find(LL v)
26      {
27          for(Node *now = g[v%MOD]; now; now = now->nxt)
28              if(now->v == v)
29                  return now->k;
30          return -1;
31      }
32      void Ins(LL k, LL v)
33      {
```

```
34          if(find(v) != -1)return;
35          pt->k = k;
36          pt->v = v;
37          pt->nxt = g[v % MOD];
38          g[v % MOD] = pt++;
39      }
40
41  }hash;
42
43  LL gcd(LL x, LL y)
44  {
45      return y==0?x:gcd(y,x%y);
46  }
47
48  LL e_gcd(LL a, LL b, LL &x, LL &y)
49  {
50      if(b==0)
51      {
52          x = 1; y = 0;
53          return a;
54      }
55      LL ret = e_gcd(b, a%b, y, x);
56      y = y - a/b*x;
57      return ret;
58  }
59
60  LL Baby(LL A, LL B, LL C)//A^x = B (mod C)
61  {
62      B %= C; A %= C;
63      LL x = 1%C, y;
64      for(int i = 0; i <= 64; i++)
65      {
66          if(x==B)return i;
67          x = x*A % C;
68      }
69
70      LL D = 1%C, g;
71      int cnt = 0;
72      while((g = gcd(A,C)) != 1)
73      {
74          if(B%g) return -1;
75          cnt++;
76          C /= g;
77          B /= g;
78          D = A/g * D % C;
79      }
80      hash.init();
81      int m = (int)sqrt(C);
82      LL Am = 1%C; hash.Ins(0,Am);
83      for(int i = 1; i <= m; i++)
84      {
85          Am = Am*A % C;
86          hash.Ins(i,Am);
87      }
88      for(int i = 0; i <= m; i++)
89      {
90          //D*x = B (mod C), D*x + C*y = B
91          g = e_gcd(D,C,x,y);
92          x = (x*B/g%C+C)%C;
93          LL k = hash.find(x);
94          if(k != -1) return i*m+k+cnt;
95          D = D*Am % C;
96      }
97      return -1;
98  }
99
100 int main()
101 {
102     int A,B,C;
103     while(scanf("%d%d%d",&A,&C,&B) == 3 && (A+B+C))
104     {
105         if(B>=C)
106         {
107             puts("Orz,I␣'␣cant␣find␣D!");
108             continue;
109         }
110         LL ret = Baby(A,B,C);
111         if(ret == -1)puts("Orz,I␣'␣cant␣find␣D!");
```

```
112         else  printf("%I64d\n",ret);
113     }
114
115     return 0;
116 }
```

7.7 FFT

```
1  /*
2     注意如果有负数，先取模，最后看是不是小于 P / 2 ，如果不是的话，就是负的。
3     对于有负数的情况 P 要足够大。
4     常见题型：求 \sum a[i]*b[( i+j)%n] ，考虑把 b 反转，然后乘法
5  */
6  #include <cstdio>
7  #include <cstdlib>
8  #include <cstring>
9  #include <cmath>
10 #include <algorithm>
11 #pragma comment(linker, "/STACK:102400000,102400000")
12
13 using namespace std;
14
15 typedef long long LL;
16 const int MaxN = 1 << 19; // 1 << (logN + 2(3))
17 const int MaxD = 25; // logN
18 const int D = 1; // 10 进制 1 位
19 const int Inf = 0x3F3F3F3F;
20
21 #define inv(n) Pow(n, P - 2, P)
22
23 LL P;
24 LL _g[MaxD];
25 int BIT_CNT;
26
27 LL Pow(LL a, LL b, LL c)
28 {
29     LL ret = 1 % c;
30     while (b > 0)
31     {
32         if (b & 1) ret = ret * a % c;
33         a = a * a % c;
34         b >>= 1;
35     }
36     return ret;
37 }
38
39 bool is_prime(LL n)
40 {
41     LL i;
42     for (i = 2; i * i <= n; ++i)
43         if (n % i == 0) return false;
44     return true;
45 }
46
47
48 LL getP(LL Lim)
49 {
50     // P = C * 2^21 + 1, P >= Lim
51     LL c, t;
52     for (c = 3; ; ++c)
53     {
54         t = c << 21 | 1;
55         if (t >= Lim && is_prime(t)) return t;
56     }
57     return -1;
58 }
59
60 bool is_g(LL a, LL p)
61 {
62     LL i, p0 = p - 1;
63     for (i = 1; i * i <= p0; ++i)
64     {
65         if (p0 % i == 0)
66         {
```

```
67              if (Pow(a, i, p) == 1 && i < p0) return false;
68              if (Pow(a, p0 / i, p) == 1 && p0 / i < p0) return false;
69          }
70      }
71      return true;
72  }
73
74  LL getG(LL p)
75  {
76      LL g;
77      for (g = 2; !is_g(g, p); ++g)
78          /* empty */;
79      return g;
80  }
81
82  void get_g(LL G, LL p, int blim, LL _g[]) // blim logN 上限，一般加 1 到 2
83  {
84      int i;
85      LL j;
86      for (i = 0; i < blim; ++i)
87      {
88          j = 1LL << i;
89          _g[i] = Pow(G, (p - 1) / j, p);
90      }
91  }
92
93  int reverse(int j)
94  {
95      int i, k = 0;
96      for (i = 0; i < BIT_CNT; ++i)
97          if (j & (1 << i))
98              k |= 1 << (BIT_CNT - i - 1);
99      return k;
100 }
101
102 void FFT(LL x[], int n)
103 {
104     int i, j, m, i0, j0;
105     LL t0, t1, tt;
106     for (m = 1; m <= BIT_CNT; ++m)
107     {
108         i0 = 1 << m;
109         j0 = i0 >> 1;
110         for (i = 0; i < n; i += i0)
111             for (j = 0, tt = 1; j < j0; ++j, tt = tt * _g[m] % P)
112             {
113                 t0 = tt;
114                 t1 = x[i + j + j0] * t0 % P;
115                 t0 = (x[i + j] + t1) % P;
116                 t1 = (x[i + j] - t1) % P;
117                 if (t1 < 0) t1 += P;
118                 x[i + j] = t0;
119                 x[i + j + j0] = t1;
120             }
121     }
122 }
123
124 void conv(LL a[], LL b[], int n)
125 {
126     int i;
127     FFT(a, n);
128     FFT(b, n);
129     for (i = 0; i < n; ++i)
130         b[i] = a[i] * b[i] % P;
131     for (i = 0; i < n; ++i)
132         a[reverse(i)] = b[i == 0 ? 0 : n - i];
133     FFT(a, n);
134     for (i = 0; i < n; ++i)
135         a[i] = a[i] * inv(n) % P;
136 }
137
138 char sA[MaxN], sB[MaxN];
139 LL a[MaxN], b[MaxN], ans[MaxN];
140 int n;
141
142 void init ()
143 {
144     P = getP(1000000000);
```

```
    get_g(getG(P), P, 21, _g);
}

void get()
{
    int i, j;
    int c = 0, k = 0;
    LL av, bv, t = 1;
    av = bv = 0;
    int on = (n + D - 1) / D;
    for (BIT_CNT = 1; on + on > (1 << BIT_CNT); ++BIT_CNT)
        ;

    // carefull !
    memset(a, 0, sizeof(a));
    memset(b, 0, sizeof(b));

    for (i = n - 1; i >= 0; --i)
    {
        av = av + t * (sA[i] - '0');
        bv = bv + t * (sB[i] - '0');
        ++c;
        if (c == D || i == 0)
        {
            j = reverse(k);
            a[j] = av;
            b[j] = bv;
            ++k;
            c = av = bv = 0;
            t = 1;
        }
        else
            t *= 10;
    }
    n = 1 << BIT_CNT;
}

void sol()
{
    int i, j = 0, k;
    conv(a, b, n);
    for (i = 0; i < n; ++i)
    {
        k = a[i] + j;
        ans[i] = k % 10;
        j = k / 10;
    }
    for (i = n - 1; i > 0 && ans[i] == 0; --i)
        ;
    printf("%lld", ans[i--]);
    for ( ; i >= 0; --i)
        printf("%lld", ans[i]);
    puts("");
}

int main()
{
    freopen("in.txt", "r", stdin);

    init();
    while (scanf("%s%s", sA, sB) == 2)
    {
        int la = strlen(sA);
        int lb = strlen(sB);
        n = max(la, lb);
        for (int i = 0; i < n; ++i)
        {
            if (i < la) sA[n - 1 - i] = sA[la - 1 - i];
            else sA[n - 1 - i] = '0';
            if (i < lb) sB[n - 1 - i] = sB[lb - 1 - i];
            else sB[n - 1 - i] = '0';
        }
        sA[n] = sB[n] = 0;
        get();
        sol();
    }

    return 0;
```

```
223 }
```

7.8 其它公式

7.8.1 正多面体顶点着色

正四面体$N=\frac{n^4+11n^2}{24}$
正六面体$N=\frac{n^8+17n^4+6n^2}{24}$
正八面体$N=\frac{n^6+3n^4+12n^3+8n^2}{24}$
正十二面体$N=\frac{n^{20}+15n^{10}+20n^8+24n^4}{60}$
正二十面体$N=\frac{n^{12}+15n^6+44n^4}{60}$

7.8.2 求和公式

$\sum k=\frac{n(n+1)}{2}$
$\sum 2k-1=n^2$
$\sum k^2=\frac{n(n+1)(2n+1)}{6}$
$\sum (2k-1)^2=\frac{n(4n^2-1)}{3}$
$\sum k^3=\left(\frac{n(n+1)}{2}\right)^2$
$\sum (2k-1)^3=n^2\left(2n^2-1\right)$
$\sum k^4=\frac{n(n+1)(2n+1)\left(3n^2+3n-1\right)}{30}$
$\sum k^5=\frac{n^2(n+1)^2\left(2n^2+2n-1\right)}{12}$
$\sum k\left(k+1\right)=\frac{n(n+1)(n+2)}{3}$
$\sum k\left(k+1\right)\left(k+2\right)=\frac{n(n+1)(n+2)(n+3)}{4}$
$\sum k\left(k+1\right)\left(k+2\right)\left(k+3\right)=\frac{n(n+1)(n+2)(n+3)(n+4)}{5}$

7.8.3 几何公式

1. **球扇形**
   全面积$T=\pi r\left(2h+r_0\right)$，$h$为球冠高，$r_0$为球冠底面半径。
   体积$V=\frac{2\pi r_0^2 h}{3}$
2. **几何重心**
   $G_x=\frac{\int xf(x)dx}{\int f(x)dx}$