ACM/ICPC at Wuhan University

Xioumu STL(code)

Created by xioumu, for OpenShield

目录

Graph 2
2-set 2
N*log(n) Dijkstra 3
DataStructure 4
数状数组 4
RMQ 4
后缀树 5
平衡树 6
线段树-扫描线矩形面积并 7
大根堆 8
DXL 8
Computational Geometry 10
凸包 10
线段相交 11
最近点对 11
线段与线段的距离 11
O(N^2)处理最少用几段弧完全覆盖一个圆 12
数论 12
miller_rabin_and_Pollard_rho 12
get_prime 14
Matrix 14
Gauss 15
GCD&扩展 GCD 16
辛普森积分 16
Others 16
O(n)求回文串 16
KMP 16
读入优化 17
乱七八糟 17
Vimrc 18

Graph

2-set

```
int n, m;
vector<int> e[maxn], g[maxn], op[maxn];
void add(vector<int> *e, int x, int y){
    e[x].push_back(y);
}
void get(int &x, inat &nx){
    if(x < 0){
        x = -x;
        nx = x + n;
    }
    else {
        nx = x;
        x += n;
    }
}
int sta[maxn], low[maxn], dfn[maxn], v[maxn], fen[maxn], du[maxn], co[maxn];
int top, num, fn;
void tar(vector<int> *e, int w){
    sta[++top] = w;
    low[w] = dfn[w] = ++num;
    v[w] = 1;
    rep (i, sz(e[w]) ) {
        int j = e[w][i];
        if(v[j] == 2) continue;
        if( dfn[j] == -1) tar(e, j);
        low[w] = min(low[w], low[j]);
    }

    if(dfn[w] == low[w]){
        fn++;
        do{
            fen[ sta[top] ] = fn;
            v[ sta[top] ] = 2;
            top--;
        }while( sta[top + 1] != w);
    }
}
bool shrink(vector <int> *e, vector <int> *g){ //1 -- 2 * n 缩点 把边反向 如果 ai, aj 在一个强连通 return false;
```

```
    memset(dfn, -1, sizeof(dfn));
    memset(low, -1, sizeof(low));
    memset(v, 0, sizeof(v));
    num = top = fn = 0;
    repf (i, 1, 2 * n)
        if(dfn[i] == -1){
            tar(e, i);
        }
    repf (i, 1, fn) {
        g[i].clear();
        op[i].clear();
    }
    memset(du, 0, sizeof(du));
    repf (i, 1, 2 * n){
        int ni;
        if(i > n) ni = i - n;
        else ni = i + n;
        if(fen[i] == fen[ni]) return false;
        add(op, fen[i], fen[ni]);
        rep (j, sz(e[i])){
            int k = e[i][j];
            if( fen[i] != fen[k] ){
                add(g, fen[k], fen[i]);
                du[ fen[i] ]++;
            }
        }
    }
    return true;
}
void updata(vector<int> *e, int w){
    if(co[w] != 0){
        return ;
    }
    co[w] = 2;
    rep (i, sz(e[w]) ){
        int j = e[w][i];
        du[j]--;
        updata(e, j);
    }
}
void dye(vector<int> *e){
    top = 0;
    repf (i, 1, fn)
        if( du[i] == 0)
            sta[++top] = i;
    memset(co, 0, sizeof(co));
    while(top != 0){
        int k = sta[top--];
        if( co[k] != 0) continue;
        else{
            co[k] = 1;
            rep (i, sz(op[k])){
                updata(e, op[k][i]);
            }
        }
        rep (i, sz(e[k])){
            int j = e[k][i];
            du[j]--;
            if(du[j] == 0)
                sta[++top] = j;
        }
    }
}
int main(){
    if( !shrink(e, g) ){
        printf("No\n");
    }
    else {
        printf("Yes\n");
        dye(g);
        vector<int> ans;
        repf (i, n + 1, 2 * n)
            if(co[ fen[i] ] == 1){
                ans.push back(i - n);
            }
        printf("%d", sz(ans));
        rep (i, sz(ans)){
            printf(" %d", ans[i]);
        }
        printf("\n");
    }

    return 0;
}
```

N*log(n) Dijkstra

```
long long v[MAXN],dis[MAXN],dui[MAXN],rear,front,dn,b[MAXN];
void up(long long x)
{
    long long i,j,k;
    i = x/2; j = x;
```

```
 6      while(i >= 1)
 7      {
 8        if(dis[ dui[j] ] < dis[ dui[i] ] ) { swap(&dui[j],&dui[i]);
swap(&b[ dui[j] ],&b[ dui[i] ]);  }
 9        else break;
10        j = i;
11        i /= 2;
12      }
13  }
14  void jin(long long a)
15  {
16      dui[++dn] = a;
17      b[a] = dn;
18      up(dn);
19  }
20  void chu(long long *a)
21  {
22      long long i,j,k;
23      *a = dui[1];
24      swap(&dui[1],&dui[dn]);
25      swap(&b[ dui[1] ],&b[ dui[dn] ]);
26      dn--;
27      i = 1;
28      while(i<=dn/2)
29      {  j = i*2;
30        if(j+1<=dn && dis[ dui[j] ] > dis[ dui[j+1] ]) j++;
31        if(dis[ dui[i] ] > dis[ dui[j] ]) { swap(&dui[i],&dui[j]);
swap(&b[ dui[i] ],&b[ dui[j] ]); }
32        else break;
33        i = j;
34      }
35  }
36  void dij(long long w)
37  {
38      long long i,j,k,r;
39      node *p;
40      memset(v,0,sizeof(v));
41      memset(dui,10,sizeof(dui));
42      /*for(i=1;i<=s4;i++) dis[i] = MAXNUM;*/
43      dn = 0;
44      dis[w] = 0;
45      for(i=1;i<=s4;i++) jin(i);
46      for(i=1;i<=(n-1)*(n-1)+3;i++)
47      {  chu(&k);  /*printf("%I64d:%I64d\n",k,dis[k]);*/
48        for(p=g[k];p;p=p->next)
49          if(dis[p->adj] > dis[k] + p->road)
50          {  dis[p->adj] = dis[k] + p->road;
51            up(b[p->adj]);
52          }
53      }
54  }
55
```

DataStructure

数状数组

```
1  int f[maxn];
 2  int lowb(int t) { return t & (-t); }
 3  void add(int *f, int i, int value){   // index : 1 ~ n
 4      for(; i < n; f[i] += value, i += lowb(i) );
 5  }
 6  int getsum(int *f, int i){
 7      int s = 0;
 8      for(; i > 0; s += f[i], i -= lowb(i));
 9      return s;
10  }
```

RMQ

```
1  void getrmq(int *height, int n, int rmq[50][MAXN]){
 2      int i,j,k,r,w,m;
 3      m = (double)log((double)n + 1) / (double)log(2.0);
 4      for(i=0; i<=m; i++)
 5          for(j=0; j<=n; j++)
 6              rmq[i][j] = MAXNUM;
 7      for(i=0; i<=n; i++)  rmq[0][i] = height[i];
 8      for(i=1; i<=m; i++)
 9          for(j=0; j<=n - (1<<(i-1)) + 1; j++)
10              rmq[i][j] = min(rmq[i-1][j], rmq[i-1][j + (1 << (i - 1) ) ]);
11  }
12  int find(int rmq[50][MAXN], int l, int r){
13      int m = (double)log((double)r - l + 1) / (double)log(2.0);
14      return min(rmq[m][l], rmq[m][r - (1<<m) + 1]);
15  }
16
```

```
// (后缀树 最长回文子串)
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<algorithm>
#include<string>
using namespace std;
#define inf 1e-8
#define MAXN 2007
typedef long long int64;
int a[MAXN], height[MAXN], myrank[MAXN], sa[MAXN];
int wa[MAXN], wb[MAXN], wv[MAXN], wws[MAXN];
int rmq[100][MAXN];
int n;
bool cmp(int *wb, int a, int b, int l, int n){
    int r,w;
    r = a + l >= n ? 0 : wb[a+l];
    w = b + l >= n ? 0 : wb[b+l];
    return wb[a] == wb[b] && r == w;
}
      //格挡符号要加最大的符号,如：200.  末尾要加最小的符号,如：0.
void getsa(int *a, int n, int m, int *sa){  //sa: 1~n, a: 0 ~ n-1,a[n]=0
    int i,j,k,r,w,p;
    for(i=0; i<=m; i++)  wws[i] = 0;
    for(i=0; i<n; i++)  wws[ wa[i] = a[i] ]++;
    for(i=1; i<=m; i++) wws[i] += wws[i-1];
    for(i=n-1; i>=0; i--)   sa[ --wws[ wa[i] ] ] = i;
    for(j=1,p=1; j<n&&p<n; j*=2,m=p){       //特别注意要写 m=p
        for(i=n-j,p=0; i<n; i++)  wb[p++] = i;
        for(i=0; i<n; i++)  if(sa[i] >= j)  wb[p++] = sa[i] - j;
        for(i=0; i<=m; i++)   wws[i] = 0;
        for(i=0; i<n; i++)  wv[i] = wa[ wb[i] ];
        for(i=0; i<n; i++)  wws[ wv[i] ]++;
        for(i=1; i<=m; i++)  wws[i] += wws[i-1];
        for(i=n-1; i>=0; i--)   sa[ --wws[ wv[i] ] ] = wb[i];
        for(i=0; i<n; i++) wb[i] = wa[i];
        for(i=1,p=1,wa[ sa[0] ] = 0; i<n; i++)
            wa[ sa[i] ] = cmp(wb, sa[i], sa[i-1], j, n) ? p-1 : p++;
    }
}
void getheight(int *a, int *sa, int n, int *height){
    int i,j,k,r,w;
    k = 0;
    for(i=0; i<=n; i++)  myrank[ sa[i] ] = i;
    for(i=0; i<n; height[ myrank[i++] ] = k)
        for(k ? k-- : 0, j = sa[ myrank[i] - 1]; a[i+k] == a[j+k]; k++);
}
void getrmq(int *height, int n, int rmq[100][MAXN]){
    int i,j,k,r,m;
    m = (double)log((double)n+1) / (double)log(2.0);
    for(i=0; i<=m; i++)
        for(j=0; j<=n; j++)
            rmq[i][j] = 200000000;
    for(i=0; i<=n; i++){
        rmq[0][i] = height[i];
    }
    for(i=1; i<=m; i++)
        for(j=0; j<=n - (1<<(i-1)) + 1; j++)
            rmq[i][j] = min(rmq[i-1][j], rmq[i-1][j + ( 1 << (i-1) )]);
}
int find(int rmq[100][MAXN], int l, int r){
    if(l > r)  swap(l, r);
    l++;
    int m = (double)log((double)r-l+1)/(double)log(2.0);
    return min(rmq[m][l], rmq[m][r - (1<<m) + 1]);
}
int main(){
    char s[MAXN];
    int i,j,k;
    while(scanf(" %s",s) != EOF){
        memset(a,0,sizeof(a));
        n = strlen(s);
        for(i=0; i<n; i++)  a[i] = s[i];
        a[n] = 200;
        for(i=n+1; i<=n+n; i++)  a[i] = s[n + n - i];
        a[n+n+1] = 0;
        getsa(a, n+n+2, 300, sa);
        getheight(a, sa, n+n+1, height);
        getrmq(height, n+n+1, rmq);
        int ans = -1, ansb;
        for(i=0; i<n; i++){
            k = find(rmq, myrank[i], myrank[n + n - i]);
            if(ans < 2*k - 1){
                ans = 2 * k - 1;
                ansb = i - k + 1;
            }
            k = find(rmq, myrank[i], myrank[n + n - i - 1]);
            if(ans < (k-1) * 2 ){
                ans = (k-1) * 2;
                ansb = i - (k-2);
```

```
 96            printf("\n");
 97            }
 98        }
 99        for(i=ansb; i<ansb + ans; i++)
100            printf("%c",a[i]);
101        printf("\n");
102    }
103    return 0;
104 }
```

平衡树

```
1 /* 小的在左，大的在右。 */
 2 #include"stdio.h"
 3 #define NEWS (avltree *)malloc(sizeof(avltree))
 4 typedef struct avltree
 5 { struct avltree *rc,*lc;
 6   long height,data,h,gao;
 7 }avltree;
 8   FILE *input,*output;
 9 long max(long a,long b) { if(a>b) return a; else return b;}
10 long min(long a,long b) { if(a<b) return a; else return b;}
11 long mheight(avltree *t){ if(t==NULL) return 0; else return
t->height;  }
12 avltree *singleft(avltree *t)
13 { avltree *a;
14   a=t->lc;
15   t->lc=a->rc;
16   a->rc=t;
17
18   t->height=max(mheight(t->lc),mheight(t->rc))+1;
19   a->height=max(mheight(a->lc),mheight(a->rc))+1;
20   return a;
21 }
22 avltree *singright(avltree *t)
23 { avltree *p;
24   p=t->rc;
25   t->rc=p->lc;
26   p->lc=t;
27
28   t->height=max(mheight(t->lc),mheight(t->rc))+1;
29   p->height=max(mheight(p->lc),mheight(p->rc))+1;
30   return p;
31 }
32 avltree *douleft(avltree *t)
33 { t->lc=singright(t->lc);
34   t=singleft(t);
35   return t;
36 }
37 avltree *douright(avltree *t)
38 { t->rc=singleft(t->rc);
39   t=singright(t);
40   return t;
41 }
42 avltree *insert(avltree *t,long key)
43 { long i,j,k,r,w;
44   avltree *p;
45   if(t==NULL)
46   { p=NEWS;
47     p->height=1;
48     p->data=key;
49     p->lc=p->rc=NULL;
50     return p;
51   }
52   if(key>t->data)
53   { t->rc=insert(t->rc,key);
54     if(mheight(t->rc) - mheight(t->lc) ==2)
55     { if(key>t->rc->data) t=singright(t);
56       else t=douright(t);
57     }
58   }
59   else if(key<t->data)
60   { t->lc=insert(t->lc,key);
61     if(mheight(t->lc) - mheight(t->rc) ==2)
62     { if(key<t->lc->data) t=singleft(t);
63       else t=douleft(t);
64     }
65   }
66   t->height=max(mheight(t->lc),mheight(t->rc))+1;
67   return t;
68 }
69
70 int main()
71 { long i,j,k,r,w,n;
72   avltree *t=NULL;
73   FILE *input,*output;
74   input=fopen("avl.in","r");
75   output=fopen("avl.out","w");
76   fscanf(input,"%ld",&n);
77   for(i=1;i<=n;i++)
78   { fscanf(input,"%ld",&r);
```

```
    t=insert(t,r);
  }
  fclose(input);
  fclose(output);
  return 0;
}
```

线段树-扫描线矩形面积并

//注意线段树中的每个点要代表一个左闭右开的区间!

```
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<algorithm>
#include<string>
#include<vector>
using namespace std;
#define inf 1e-8
#define MAXN 2007
typedef long long int64;
int sgn(double x){
    return x > inf ? 1: (x < -inf ? -1 : 0);
}
struct node{
    double x,l,r;
    int t;
    node(double _l,double _r,double _x,int _t) : l(_l),r(_r),x(_x),
t(_t) {}
    bool operator < (const node &b) const {
        return sgn(x- b.x) < 0;
    }
};
vector<node> a;
int lazy[MAXN];
int cut[MAXN];
double fx[MAXN],fy[MAXN],sum[MAXN],num[MAXN],y[MAXN],ww[MAXN];
int n,m;
void init(){
    int i,j,k,r,w;
    double x1,y1,x2,y2;
    double x[MAXN];
    memset(lazy,0,sizeof(lazy));
    m = 0;
    a.clear();
    for(i=0; i<n; i++){
        scanf("%lf %lf %lf %lf",&x1,&y1,&x2,&y2);
        a.push_back( node(y1, y2, x1, 1) );
        a.push_back( node(y1, y2, x2, -1) );
        y[++m] = y2;
        x[m] = x1;
        y[++m] = y1;
        x[m] = x2;
    }
    sort(a.begin(), a.end());
    sort(y+1, y+m+1);
    fy[1] = y[1];
    w = 1;
    for(i=2; i<=m; i++){
        if(sgn(y[i] - y[i-1]) != 0)
            fy[++w] = y[i];
    }
    memcpy(y, fy, sizeof(y));
    m = w;
    memset(fy,0,sizeof(fy));
    for(i=1; i<m; i++)
        fy[i] = fy[i-1] + y[i+1] - y[i];

    memset(num, 0, sizeof(num));
    for(i=1; i<=m; i++)
        num[i] = fy[i];
}
void getch(int t, int &lc, int &rc){
    lc = t<<1;
    rc = t<<1 | 1;
}
void add(int t, int ll, int rr, int l, int r, int h){
    int lc,rc,mid;
    if(rr < l || r < ll) return;
    getch(t, lc, rc);
    if(l <= ll && rr <= r){
        cut[t] += h;
        if(cut[t] >= 1){
            sum[t] = num[rr] - num[ll-1];
        }
        else if(ll == rr) sum[t] = 0;
        else sum[t] = sum[lc] + sum[rc];
        return ;
    }
    mid = (ll + rr) >> 1;
```

```
    add(lc, ll, mid, l, r, h);
    add(rc, mid+1, rr, l, r, h);
    if(cut[t] >= 1){
        sum[t] = num[rr] - num[ll-1];
    }
    else sum[t] = sum[lc] + sum[rc];
}
int find(double yy){
    int l,r,mid;
    l = 1; r = m;
    while(l <= r){
        mid = (l + r) / 2;
        if(sgn(y[mid] - yy) > 0) r = mid - 1;
        else if(sgn(y[mid] - yy) < 0) l = mid + 1;
        else return mid;
    }
    return -1;
}
void solve(){
    int i,j,k,r,l,w;
    memset(cut,0,sizeof(cut));
    memset(sum,0,sizeof(sum));
    memset(lazy,0,sizeof(lazy));
    memset(ww,0,sizeof(ww));
    double ans = 0;
    for(i=0; i<(int)a.size()-1; i++){
        l = find(a[i].l);
        r = find(a[i].r) - 1;
        if(l <= r) add(1, 1, m-1, l, r, a[i].t);
        ans += sum[1] * (a[i+1].x - a[i].x);
    }
    printf("Total explored area: %0.2f\n",ans);
}
int main(){
    int ca = 1,ok=0;
    while(scanf("%d",&n) != EOF && n){
        if(ok == 1) printf("\n");
        init();
        printf("Test case #%d\n",ca++);
        solve();
        ok = 1;
    }
    return 0;
}
```

大根堆

```
long dn=0;          /*大根堆*/
void jia(long key)
{ long i,j,k,m;
  a[++dn]=key;
  i=dn/2; j=dn;
  while(i>=1)
  { if(a[j]>a[i]) swap(&a[j],&a[i]);
    else break;
    j=i; i/=2;
  }
}
void del()
{ long i,j,k,m;
  swap(&a[1],&a[dn]);
  dn--;
  i=1;
  while(i<=dn/2)
  { j=i*2;
    if(j+1<=dn&&a[j]<a[j+1]) j++;
    if(a[i]<a[j]) swap(&a[i],&a[j]);
    else break;
    i=j;
  }
}
```

DXL

Suduke

```
const int maxn = 9 + 10;
int n = 9, m = 9, tn = 9;
class Graph {
    public:
        static const int maxn = 9 * 9 * 9 + 7;
        static const int maxm = 1000 + 7;
        static const int Max = maxn * maxm + 10;
        static const int sn = 9, sm = 9, stn = 9;
        int adj[maxn][maxm], O[maxn]; //O[] is answer
        int ans, sudoku[20][20];
```

```
11
12      void init() {
13          n = m = 0;
14          memset(adj, 0, sizeof(adj));
15      }
16      void insert(int u, int v) {
17          u++, v++;
18          n = max(n, u);
19          m = max(m, v);
20          adj[u][v] = 1;
21      }
22      int find_ans() {
23          build_dlx();
24          ans = -1;
25          if (dfs(0) ) {
26              return ans;
27          }
28          return -1;
29      }
30      void out_ans(int ans) {
31          if(ans == -1) {
32              printf("NO\n");
33              return ;
34          }
35          //printf("%d", n);
36          repf (i, 0, ans - 1) {
37              int x, y, ty;
38              O[i]--;
39              x = O[i] / (sm * stn);
40              y = (O[i] % (sm * stn) ) / stn;
41              ty = (O[i] % (stn));
42              //printf("%d %d %d\n", x, y, ty);
43              sudoku[x][y] = ty + 1;
44          }
45          rep (i, sn)
46              rep (j, sm)
47                  printf("%d",sudoku[i][j]);
48          printf("\n");
49      }
50  private:
51      int head;
52      int R[Max], L[Max], U[Max], D[Max], C[Max], H[Max];
53      int S[maxn];
54      int n, m, cnt, nm;
55
56      void add(int head, int tmp, int x) {
57          H[cnt] = head;
58          R[cnt] = tmp; L[cnt] = L[tmp];
59          L[tmp] = cnt; R[L[cnt]] = cnt;
60          U[cnt] = U[x]; D[cnt] = x;
61          D[U[x]] = cnt; U[x] = cnt;
62          C[cnt] = x; ++S[x];
63          ++cnt;
64      }
65      void build_dlx() {
66          L[0] = R[0] = U[0] = D[0] = C[0] = H[0] = 0;
67          for (int i = 1; i <= m; i++) {
68              H[i] = 0;
69              L[i] = i - 1; R[i] = 0;
70              R[i - 1] = i; L[0] = i;
71              U[i] = D[i] = C[i] = i;
72              S[i] = 0;
73          }
74          cnt = m + 1;
75          for (int i = 1; i <= n; i++) {
76              int tmp = Max - 1;
77              L[tmp] = R[tmp] = U[tmp] = D[tmp] = C[tmp] = tmp;
78              for (int j = 1; j <= m; j++)
79                  if(adj[i][j])  {
80                      add(i, tmp, j);
81                  }
82              L[R[tmp]] = L[tmp];
83              R[L[tmp]] = R[tmp];
84          }
85      }
86      void remove(const int &c) {
87          R[L[c]] = R[c];
88          L[R[c]] = L[c];
89          for (int i = D[c]; i != c; i = D[i]) {
90              for (int j = R[i]; j != i; j = R[j]) {
91                  U[D[j]] = U[j];
92                  D[U[j]] = D[j];
93                  --S[C[j]];
94              }
95          }
96      }
97
98      void resume(const int &c) {
99          for (int i = D[c]; i != c; i = D[i]) {
100             for (int j = R[i]; j != i; j = R[j]) {
101                 U[D[j]] = j;
102                 D[U[j]] = j;
103                 ++S[C[j]];
104             }
```

```
105             }
106             R[L[c]] = c;
107             L[R[c]] = c;
108         }
109
110         bool dfs(const int &k) {
111             if (R[0] == 0){
112                 ans = k;
113                 return true;
114             }
115             int s(maxint), c;
116             for (int i = R[0]; i != 0; i = R[i]) {
117                 if (S[i] < s) {
118                     c = i;
119                     s = S[i];
120                 }
121             }
122             remove(c);
123             for (int i = D[c]; i != c; i = D[i]) {
124                 O[k] = H[i]; //
125                 for (int j = R[i]; j != i; j = R[j]) remove(C[j]);
126                 if (dfs(k + 1)) return true;
127                 for (int j = L[i]; j != i; j = L[j]) resume(C[j]);
128             }
129             resume(c);
130             return false;
131         }
132 }G;
133 char in[maxn * maxn];
134 int a[maxn][maxn];
135
136 void add(int x, int y, int ty) {
137     int l_id = x * m * tn + y * tn + ty;
138     //printf("%d %d %d %d\n", x, y, ty, l_id);
139     int bn =  ((x / 3) * 3 + y / 3);
140     G.insert(l_id, x * m + y);
141     G.insert(l_id, x * tn + ty + n * m);          //row
142     G.insert(l_id, n * tn + y * tn + ty + n * m);   //vertical
143     G.insert(l_id, n * tn + m * tn + bn * tn + ty + n * m);  //block
144 }
145 int main(){
146     while (scanf("%s", in) == 1) {
147         if (in[0] == 'e')    break;
148         rep (i, n)
149             rep (j, m)
150                 if (in[i * m + j] == '.')   a[i][j] = 0;
151                 else  a[i][j] = in[i * m + j] - '0';
152
153         G.init();
154         rep (i, n)
155             rep (j, m) {
156                 if(a[i][j] == 0) {
157                     repf (k, 1, 9)
158                         add(i, j, k - 1);
159                 }
160                 else add(i, j, a[i][j] - 1);
161             }
162         int ans = G.find_ans();
163         G.out_ans(ans);
164     }
165     return 0;
166 }
167
```

Computational Geometry

凸包

```
1 bool operator < (const point &p) const{
2     if(sgn(x - p.x) != 0) return x < p.x;
3     else return y < p.y;
4 }
5 void convex(vector <point> a, vector <point> &tu){ //顺时针
6     point hu[maxn], hd[maxn];
7     int n = a.size(), un, dn;
8     sort(a.begin(), a.end());
9     hu[0] = hd[0] = a[0];
10    hu[1] = hd[1] = a[1];
11    un = dn = 1;
12    for(int i = 2; i < n; i++){
13        for(; un > 0 && sgn( (hu[un] - hu[un - 1]) * (a[i] - hu[un] )) >=
0; un--);
14        for(; dn > 0 && sgn( (hd[dn] - hd[dn - 1]) * (a[i] - hd[dn] ))
<= 0; dn--);
15        hu[++un] = a[i];
16        hd[++dn] = a[i];
17    }
18    tu.clear();
19    for(int i = 0; i <= un - 1; i++) tu.push_back(hu[i]);
20    for(int i = dn; i >= 1; i--) tu.push_back(hd[i]);
21 }
```

```
22
23
```

线段相交

判线段相交，求交点

```
bool jiaodian(point a,point b,point c,point d,point &e)
{
    double d1 = (b-a) * (c-a), d2 = (b-a) * (d-a),
          d3 = (d-c) * (a-c), d4 = (d-c) * (b-c);
    if(sgn(d1)*sgn(d2) > 0)
       return false;
    e = point( (c.x*d2 - d.x*d1) / (d2-d1) ,
              (c.y*d2 - d.y*d1) / (d2-d1) );
    return true;
}
```

最近点对

```
bool cmpy(const point &a, const point &b){
    if( sgn(a.y - b.y) != 0) return a.y < b.y;
    else return a.x < b.x;
}
bool cmpx(const point &a, const point &b){
    if( sgn(a.x - b.x) != 0) return a.x < b.x;
    else return a.y < b.y;
}
point tempt[maxn], a[maxn];
int n;
void get_min(point *a, int l, int r, double &d){
    int n = r - l + 1;
    if(n == 1) { return;}
    if(n <= 3){
       repf(i, l, r - 1){
          d = min(d, (a[i] - a[(i + 1)]).len());
       }
       d = min(d, (a[r] - a[l]).len());
    }
    else{
       double d1, d2, d3;
       d1 = d2 = d3 = 1e100;
       int mid = (l + r) >> 1;
       get_min(a, l, mid, d1);
       get_min(a, mid + 1, r, d2);
       d = min(d1, d2);
       int k = 0, num = 6;
       repf (i, l, r)
          if( fabs(a[i].x - a[mid].x) <= d)
             tempt[k++] = a[i];
       sort(tempt, tempt + k, cmpy);
       rep (i, k)
          for(int j = i + 1; j < k && tempt[j].y - tempt[i].y < d;
j++){
             d = min(d, (tempt[j] - tempt[i]).len());
          }
    }
}
int main(){
    while(scanf("%d", &n) == 1 && n){
       rep(i, n){
          point p;
          p.input();
          a[i] = p;
       }
       sort(a, a + n, cmpx);
       double ans = 1e100;
       get min(a, 0, n - 1, ans);
       printf("%.2f\n", ans / 2);
    }
    return 0;
}
```

线段与线段的距离

```
double get_dis(point a, point sb, point eb) {
    return min( (a - sb).len(), (a - eb).len());
}
double dis(point a, point b, point c) {
    double mul = ( (a - b) ^ (c - b) ) / (c - b).len();
    point dir = (c - b).set();
    point mid = dir * mul + b;
    if( sgn((mid - b) ^ (c - b) ) >= 0 && sgn((mid - c) ^ (b - c)) >=
0) {
       return fabs((a - b) * (c - b) / (c - b).len());
    }
    else return get_dis(a, b, c);
}
double dis(int a, int b) { //线段 tp[a]sp[a], tp[b]sp[b]
```

```
    double res =  min( dis(tp[a], tp[b], sp[b]), dis(sp[a], tp[b],
sp[b]));
    res = min(res, min(dis(tp[b], tp[a], sp[a]), dis(sp[b], tp[a],
sp[a])));
    return res;
}
```

O(N^2)处理最少用几段弧完全覆盖一个圆

```
struct node {
    double be, en; //开始的角度 与 结束的角度 （-pi ~ pi）
    node (double  be = 0, double  en = 0) : be( be), en( en){
    }
    bool operator < (const node &b) const {
        return sgn(be - b.be) < 0;
    }
} a[maxn], b[maxn];

node change(node p, double ang) {  //将角度转换成从 ang 度开始，需要转动
多少度
    double be = p.be, en = p.en;
    be -= ang;
    while(sgn(be) < 0) be += 2 * pi;
    en -= ang;
    while(sgn(en) < 0) en += 2 * pi;
    if(sgn(en - be) < 0) en += 2 * pi;
    return node(be, en);
}


        sort(a, a + n);
        rep (i, n)
            a[i + n] = a[i];
        int ans = maxint;
        rep (i, n) {
            rep (j, n) {
                b[j] =  change(a[i + j], a[i].be);
            }
            int res = 0, k = 0;
            double old = 0;
            while(k < n && sgn(old - 2 * pi) < 0) {
                double next = old;
                while(k < n && sgn(b[k].be - old) <= 0) {
                    if(sgn(b[k].en - next) > 0)
                        next = b[k].en;
                    k++;
                }
                if(sgn(next - old) == 0 ) k = n + 1;
                res++;
                old = next;

            }
            if(sgn(old - 2 * pi) < 0)  {
                continue;
            }
            ans = min(ans, res);
        }
        if(ans == maxint) ans = -1;
        printf("%d\n", ans);

```

数论

miller_rabin_and_Pollard_rho

```
//miller_rabin 大数检测+Pollard P 素因子分解
//输入 a<2^63
//加大 MAX 可以保证分解的成功率
#include <stdlib.h>
#include <stdio.h>

typedef unsigned __int64 u64;

#define MAX 100
#define MAXN 30

u64 len, dig, limit;
u64 mod(u64 a, u64 b, u64 n)
{
    if(!a) return 0;
    else return (((a & dig) * b) % n + (mod(a >> len, b, n) << len) %
n) % n;
}

u64 by(u64 a, u64 b, u64 n)
{
```

```
    u64 p;
    p = 8, len = 61;
    while(p < n)
    {
        p <<= 4;
        len -= 4;
    }
    dig = ((limit / p) << 1) - 1; //动态划分段
    return mod(a, b, n);
}

u64 random(void)
{
    u64 a;
    a = rand();
    a *= rand();
    a *= rand();
    a *= rand();
    return a;
}

//Miller_Rabin
u64 square_multiply(u64 x, u64 c, u64 n)
{
    u64 z = 1;
    while(c)
    {
        if(c % 2 == 1) z = by(z, x, n);
        x = by(x,x,n);
        c = (c >> 1);
    }
    return z;
}

bool Miller_Rabin(u64 n)
{
    if(n < 2) return false;
    if(n == 2) return true;
    if(!(n & 1)) return false;
    u64 k = 0, i, j, m, a;
    m = n - 1;
    while(m % 2 == 0) m = (m >> 1), k++;
    for(i = 0; i < MAX; i++)
    {
        a = square_multiply(random() % (n - 1) + 1, m, n);//平方乘
        if(a == 1) continue;
        for(j = 0; j < k; j++)
        {
            if(a == n - 1) break;
            a = by(a, a, n);
        }
        if(j < k) continue;
        return false ;
    }
    return true;
}

//Pollard p,只找出一个因子。
u64 gcd(u64 a, u64 b)
{
    return b == 0 ? a : gcd(b, a % b);
}

//用公式 f(x) = x^2 + 1 检验碰撞。
u64 f(u64 x, u64 n)
{
    return (by(x, x, n) + 1) % n;
}

//分解不到，return 0
u64 Pollard(u64 n)
{
    if(n <= 2) return 0;
    if(!(n & 1)) return 2; //必不可少
    u64 i, p, x, xx;
    for(i = 1; i < MAX; i++)
    {
        x = random() % n; //或者直接用 x = i
        xx = f(x, n);
        p = gcd((xx + n - x) % n , n);
        while(p == 1)
        {
            x = f(x, n);
            xx = f(f(xx, n), n);
            p = gcd((xx + n - x) % n, n) % n;
        }
        if(p)return p;
    }
    return 0;
}

////////////////////////////////////////////////////////
u64 factor[MAXN], m;
////////////////////////////////////////////////////////
```

```
//分解质数因子
u64 prime(u64 a)
{
    if(Miller Rabin(a)) return 0;
    u64 t = Pollard(a), p;
    if(p = prime(t)) return p;
    else return t;
}

int main(void)
{
    u64 l, a, t;
    limit = 1;
    limit = limit << 63; //动态化分段使用
    while(scanf("%I64u", &a) != EOF)
    {
        m = 0;
        while(a > 1)
        {
            if(Miller_Rabin(a)) break;
            t = prime(a);
            factor[m++] = t;
            a /= t;
        }
        if(a > 0) factor[m++] = a;
        for(l = 0; l < m; l++)
            printf("%I64u\n", factor[l]);
    }
    return 0;
}
```

get_prime

```
int prime[664588], cnt = 0;
void makePrime() {
    for (int i = 2; i < maxn; ++i) {
        if (!f[i]) {
            prime[cnt++] = i;
        }
        for (int j = 0; (int64)i * prime[j] < maxn; ++j) {
            f[i * prime[j]] = true;
            if (i % prime[j] == 0) {
                break;
            }
        }
    }
}
```

Matrix

```
struct matrix {
    double ar[maxa][maxa];
    int n, m ; // n * m; 0 ~ n - 1, 0 ~ m - 1;
    matrix() {
        n = 4; //n
        m = 4; //m
        memset(ar, 0, sizeof(ar));
    }
    void clear() {
        rep (i, n)
            rep (j, m)
                ar[i][j] = 0;
    }
    void set one() {
        rep (i, n)
            rep (j, m)
                ar[i][j] = 0;
        rep (i, min(n, m))
            ar[i][i] = 1;
    }
    void output() {
        printf("%d %d\n", n, m);
        rep(i, n) {
            rep(j, m)
                printf("%.3f ", ar[i][j]);
            printf("\n");
        }
        printf("\n");
    }
};
matrix operator * (const matrix &a, const matrix &b) {
    matrix c;
    if(a.m != b.n) printf("a.m != b.n\n");
    c.clear();
    c.n = a.n;
    c.m = b.m;
    rep (i, a.n)
        rep (j, b.m)
        rep (k, a.m) {
            c.ar[i][j] += a.ar[i][k] * b.ar[k][j];   //mod
```

```
41         }
42     return c;
43 }
44
```

二&三维旋转

平移：

$$\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ tx & ty & tz & 1 \end{pmatrix}$$

拉伸：

$$\begin{pmatrix} a & & & \\ & b & & \\ & & c & \\ & & & 1 \end{pmatrix}$$

C = cos(angle), S = sin(angle).

绕(0, 0, 0) - (X, Y, Z) 向量顺时针旋转 angle (即从(x,y,z)向(0,0,0)点看,顺时针旋转)

旋转：

$$\begin{pmatrix} C+A_x^2(1-C) & A_xA_y(1-C)-A_zS & A_xA_z(1-C)+A_yS & 0 \\ A_xA_y(1-C)+A_zS & C+A_y^2(1-C) & A_yA_z(1-C)-A_xS & 0 \\ A_xA_z(1-C)-A_yS & A_yA_z(1-C)+A_xS & C+A_z^2(1-C) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
matrix get_rotate(double x, double y, double z, double d) {
    matrix now;
    now.set_one();
    d = -d / 180.0 * pi;
    double c = cos(d), s = sin(d);
    double l = sqrt(x * x + y * y + z * z);
    x /= l, y /= l, z /= l;
    now.ar[0][0] = c + x * x * (1 - c);
    now.ar[0][1] = x * y * (1 - c) - z * s;
    now.ar[0][2] = x * z * (1 - c) + y * s;

    now.ar[1][0] = x * y * (1 - c) + z * s;
    now.ar[1][1] = c + y * y * (1 - c);
    now.ar[1][2] = y * z * (1 - c) - x * s;

    now.ar[2][0] = x * z * (1 - c) - y * s;
    now.ar[2][1] = y * z * (1 - c) + x * s;
    now.ar[2][2] = c + z * z * (1 - c);
    now.ar[3][3] = 1;
    return now;
}
```

Gauss

```
1 int gauss(int map[40][40],int ans[40])
 2 {
 3    int i,j,k,r,w;
 4    for(k=0;k<30;k++)
 5    {  i = k;
 6       while(i<30 && map[i][k] == 0)  i++;
 7       if(i == 30) continue;
 8       if(i > k)
 9       {  for(j=0;j<=30;j++)
10             swap(map[i][j],map[k][j]);
11       }
12       for(i=0;i<30;i++)
```

```
        if(map[i][k] && i != k)
        {  for(j=k;j<=30;j++)
             map[i][j] ^= map[k][j];
        }
   }

   for(k=29;k>=0;k--)
   {  ans[k] = map[k][30];
      for(i=0;i<=30 && !map[k][i];i++) ;
      if(i == 30) return 0;
      for(i=k+1;i<30;i++)
        ans[k] ^= map[k][i] * ans[i];
      //ans[k] ^= map[k][k];
   }
}
```

GCD&扩展 GCD

```
long long Gcd(long long a,long long b)
{
    for(long long t=a%b;t; a=b,b=t,t=a%b); return b;
}
long long ExpandGcd(long long a, long long b, long long &d, long long
&x, long long &y)
{
    if( b ) { ExpandGcd( b, a%b , d, y, x); y -= a/b * x; }
    else { d = a; x = 1; y = 0; }
}

```

辛普森积分

```
double f(double x) {
    return x;
}
double sps(double l, double r){
    return (f(l) + f(r) + f((l+r)/2)*4)/6 * (r - l);
}
double sps2(double l, double r, int dep){
    //printf("%lf %lf %d\n", l, r, dep);
    double cur = sps(l, r), mid = (l + r)/2;
    double y = sps(l, mid) + sps(mid, r);
    if(sgn(cur-y) == 0 && dep > 9) return cur;
    return sps2(l, mid, dep+1) + sps2(mid, r, dep+1);
}
```

Others

O(n)求回文串

```
void getff()
{
    long i,j,k,r,w,id,am,mx;
    long p;
    memset(s,0,sizeof(s));
    memset(ff,0,sizeof(ff));
    n = strlen(b);
    s[0] = '#';
    for(i=1;i<=2*n;i++)
      if(i%2 == 1) s[i] = b[i/2];
      else s[i] = '#';
    m = 2*n;   w = j = id = am= mx = 0;
    p = 1;
    while(p < m)
    {  if(mx > p) { ff[p] = min( ff[ id-(p-id) ] , ff[id] - (p-id));}
       else ff[p] = 1;

       for(;s[p + ff[p]] == s[p - ff[p]]; ff[p]++);

       if(ff[p] + p > mx)
       {  mx = ff[p] + p;
          id = p;
       }

       p++;
    }
    for(i=1;i<=m;i++) ff[i]--;
}
```

KMP

```
/*=====================================================*
 | KMP 匹配算法 O(M+N)
 | CALL: res=kmp(str, pat);  原串为 str; 模式为 pat(长为 P);
 \*=====================================================*/
```

```
int  fail[P];
int  kmp(char* str, char* pat){
    int  i, j, k;
    memset(fail, -1, sizeof (fail));
    for  (i = 1; pat[i]; ++i) {
        for  (k=fail[i-1]; k>=0 && pat[i]!=pat[k+1];
              k=fail[k]);
        if (pat[k + 1] == pat[i]) fail[i] = k + 1;
    }
    i = j = 0;
    while ( str[i] && pat[j] ){  // By Fandywang
        if ( pat[j] == str[i] ) ++i, ++j;
        else if (j == 0)++i;//第一个字符匹配失败，从 str 下个字符开始
        else  j = fail[j-1]+1;    }
    if(  pat[j] )  return  -1;
    else return i-j;
}

```

读入优化

```
int scanf(int &num)
{
    char in;
    while((in=getchar())!=EOF && (in>'9' || in<'0'));
    if(in==EOF) return 0;
    num=in-'0';
    while(in=getchar(),in>='0' && in<='9') num*=10,num+=in-'0';
    return 1;
}

```

乱七八糟

```
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<algorithm>
#include<string>
#include<map>
#include<set>
#include<iostream>
#include<vector>
#include<queue>
using namespace std;
#define sz(v) ((int)(v).size())
#define rep(i, n) for (long long i = 0; i < (n); ++i)
#define repf(i, a, b) for (long long i = (a); i <= (b); ++i)
#define repd(i, a, b) for (long long i = (a); i >= (b); --i)
#define clr(x) memset(x,0,sizeof(x))
#define clrs( x , y ) memset(x,y,sizeof(x))
#define out(x) printf(#x" %d\n", x)
typedef long long lint;
const double esp = 1e-8;

queue<int> bfs; q.push(x);q.front();q.pop();q.empty();

Reverse ( string )  功能颠倒字符串
resize(n)  初始化数组长度

=============优先队列======================
struct Type
{
    int x,y;
};
struct cmp    //top()为最大值
{
    bool operator()(const Type &a,const Type &b)
    {
        return (a.x<b.y);
    }
};

priority_queue< Type,vector<Type>,cmp > q;
priority_queue<int> q; q.push(x); q.top(); q.pop();

=================map======================
map <string, int> mp;
map <string, int>::iterator it;
int find(char ss[]){
    int i;
    string s(ss);
    it = mp.find(s);
    if( it == mp.end() )    return mp[s] = ++nn;
    else return it->second;
}
================================================
ceil()  返回大于或者等于指定表达式的最小整数
```

floor() 即取不大于 x 的最大整数
都是返回 int 形

==========

```
#define myabs(x) ((x) > 0 ? (x) : -(x))

#include <sstream>
stringstream::stringstream(string str);
stringstream ss(com[i]);

reverse(str.begin(),str.end()); 字符串反转
reverse(s[i], s[i] + strlen(s[i]));
s.erase(k, j); 从 k 开始删 j 个字符
substring 连续子串
subsequence 非连续子串
system();

==========
istream& getline ( istream &is , string &str , char delim );
istream& getline ( istream& , string& );

sscanf(s,"%d",a);

next_permutation(vec.begin(), vec.end()); 下一个排列
template <typename T>     //模板函数
  bool compare(const T &p){
                        return p < value;
                }

==========VIM==========
sp a.in 分割并打开
Tabb
Tabn
tabnew


====map===
map.begin()最大
map.rbegin()最小
mp.erase()删

===读入===
#include<sstream>
gets(ss);
string s(ss),tmp;
stringstream io;
io << s;
io >> recname[i];
while(io >> tmp) {
      sec[i].push_back(tmp);
}
==========
startsWith

=====离散==========

sort(v.begin(), v.end());
v.erase(unique(v.begin(), v.end()), v.end());

========随机打乱数组顺序======
random_shuffle ( a.begin(), a.end() );
```

Vimrc

gedit ~/.vimrc //命令

```
source $VIMRUNTIME/mswin.vim
behave mswin
imap <cr> <cr><left><right>
imap <c-]> {<cr>}<c-o>O<left><right>
imap <c-d> <c-o>dd
map <f6> =a{
map <c-t> :tabnew<cr>
syn on
colo torte
set gfn=Courier\ 10\ Pitch\ 12
set ru nu et sta nowrap ar acd ww=<,>,[,] sw=4 ts=4 cin noswf

map <f10> :call CR2()<cr><space>
func CR2()
exec "update"
exec "!xterm -fn 10*20 -e \"g++ %<.cpp -Wall -o %< && time ./%< ; read -n 1\""
endfunc
map <f9> :call CR()<cr><space>
func CR()
exec "update"
exec "!xterm -fn 10*20 -e \"g++ %<.cpp -Wall -o %< && time ./%< < %<.in ; read -n 1\""
endfunc

map<f4> :call AddComment()<cr>
func AddComment()
```

```
26     if (getline('.')[0] == '/')
27         normal ^xx
28     else
29         normal 0i//
30     endif
31 endfunc
```