ACM/ICPC at Wuhan University

Xioumu STL(code)

Created by xioumu, for OpenShield

目录

Graph

2-sat

```
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;
    while(i >= 1)
    {
        if(dis[ dui[j] ] < dis[ dui[i] ] ) { swap(&dui[j],&dui[i]); swap(&b[ dui[j] ],&b[ dui[i] ]);  }
        else break;
        j = i;
        i /= 2;
    }
}
void jin(long long a)
{
    dui[++dn] = a;
    b[a] = dn;
    up(dn);
}
void chu(long long *a)
{
    long long i,j,k;
    *a = dui[1];
    swap(&dui[1],&dui[dn]);
    swap(&b[ dui[1] ],&b[ dui[dn] ]);
    dn--;
    i = 1;
    while(i<=dn/2)
    {  j = i*2;
        if(j+1<=dn && dis[ dui[j] ] > dis[ dui[j+1] ]) j++;
        if(dis[ dui[i] ] > dis[ dui[j] ]) { swap(&dui[i],&dui[j]); swap(&b[ dui[i] ],&b[ dui[j] ]); }
        else break;
        i = j;
    }
}
void dij(long long w)
{
    long long i,j,k,r;
    node *p;
    memset(v,0,sizeof(v));
    memset(dui,10,sizeof(dui));
    /*for(i=1;i<=s4;i++) dis[i] = MAXNUM;*/
    dn = 0;
    dis[w] = 0;
    for(i=1;i<=s4;i++) jin(i);
    for(i=1;i<=(n-1)*(n-1)+3;i++)
    {  chu(&k); /*printf("%I64d:%I64d\n",k,dis[k]);*/
       for(p=g[k];p;p=p->next)
         if(dis[p->adj] > dis[k] + p->road)
         {  dis[p->adj] = dis[k] + p->road;
            up(b[p->adj]);
         }
    }
}

```

双联通分量

```
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<algorithm>
#define MAXN 1007
using namespace std;
int a[MAXN][MAXN],f[MAXN];
int n,m,ans;
void init(){
    int i,j,k,r,w;
    for(i=1;i<=n;i++)
        for(j=i+1;j<=n;j++)
            a[i][j] = a[j][i] = 1;
    for(i=1;i<=m;i++){
        scanf("%d %d",&r,&w);
        a[r][w] = a[w][r] = 0;
    }
}
int zhan[MAXN],top,v[MAXN],df[MAXN],low[MAXN],num;
int d[MAXN];
bool pan(int w) {
    int i,j,k;
    for(i=1;i<=n;i++){
        if(a[w][i] && v[i] != 0){
            if(v[i] == 1){
                v[i] = (v[w]-1)%2 + 2;
                if( !pan(i) ) return false;
            }
            else if( (v[w]-1)%2 + 2 != v[i])
                return false;
        }
    }
    return true;
}
void dfs(int w,int fa){
    int i,j,k,r;
    df[w] = low[w] = ++num;
    zhan[++top] = w;
    for(i=1;i<=n;i++)
        if(a[w][i] && i != fa){
            if(df[i] == 0){
                dfs(i,w);
                low[w] = min(low[w],low[i]);
                if(low[i] >= df[w]){
                    memset(v,0,sizeof(v));
                    k = top;
                    do{
                        v[ zhan[top] ] = 1;
                        top--;
                    }while(zhan[top+1] != i);
                    v[w]=1;

                    if(!pan(w) ){
                        for(k=1;k<=n;k++)
                            if(v[k] >= 1) {
                                d[k] = 1;
                            }
                    }

                }
            }
            else low[w] = min(low[w],df[i]);
        }

}
void solve(){
    int i,j,k,r,w;
    ans = 0;
    memset(f,0,sizeof(f));
    top = num = 0;
    memset(df,0,sizeof(df));
    memset(low,0,sizeof(low));
    memset(v,0,sizeof(v));
    memset(d,0,sizeof(d));
    for(i=1;i<=n;i++){
        if(df[i] == 0){
            dfs(i,0);
        }
    }
    for(i=1;i<=n;i++)
        if(d[i] == 0){
            ans++;
        }
    printf("%d\n",ans);
}
int main() {
    while(scanf("%d %d",&n,&m) != EOF && n && m){
        init();
        solve();
    }
    return 0;
}
```

KM

```
struct Graph {
    int w[maxn][maxn], lx[maxn], ly[maxn], matx[maxn], maty[maxn], slk[maxn], n;
    bool fx[maxn], fy[maxn];
    void get_max(int &x, int y) {
        x = max(x, y);
    }
    void get_min(int &x, int y) {
        x = min(x, y);
    }
    void clear() {
        memset(w, 0, sizeof(w));
        n = 0;
    }
    void insert(int u, int v, int c) {
        get_max(n, max(u + 1, v + 1));
        w[u][v] = c;
    }
    int match() {
        memset(ly, 0, sizeof(ly));
        for (int i = 0; i < n; ++i) {
            lx[i] = -maxint;
            for (int j = 0; j < n; ++j) {
                get_max(lx[i], w[i][j]);
            }
        }
        memset(matx, -1, sizeof(matx));
        memset(maty, -1, sizeof(maty));
        for (int i = 0; i < n; ++i) {
            memset(fx, false, sizeof(fx));
            memset(fy, false, sizeof(fy));
            for (int j = 0; j < n; j++)
                slk[j] = maxint;
            if (!dfs(i)) {
                --i;
                int p = maxint;
                for (int j = 0; j < n; j++)
                    if (fy[j] == false)
                        p = min(p, slk[j]);
                for (int j = 0; j < n; ++j) {
                    ly[j] += fy[j] * p;
                }
                for (int k = 0; k < n; ++k) {
                    lx[k] -= fx[k] * p;
                }
            }
        }
        int ans = 0;
        for (int i = 0; i < n; ++i) {
            ans += w[maty[i]][i];
        }
        return ans;
    }
    bool dfs(int u) {
        fx[u] = 1;
        for (int v = 0; v < n; ++v) {
            if (lx[u] + ly[v] > w[u][v]) {
                if (lx[u] + ly[v] - w[u][v] < slk[v])
                    slk[v] = lx[u] + ly[v] - w[u][v];
            }
            else if (lx[u] + ly[v] == w[u][v] && fy[v] == false) {
                fy[v] = true;
                if (maty[v] == -1 || dfs(maty[v])) {
                    matx[u] = v;
                    maty[v] = u;
                    return true;
                }
            }
        }
        return false;
    }
}G;

```

DataStructure

数状数组

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

RMQ

```
void getrmq(int *height, int n, int rmq[50][MAXN]){
    int i,j,k,r,w,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] = MAXNUM;
    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[50][MAXN], int l, int r){
    int m = (double)log((double)r - l + 1) / (double)log(2.0);
    return min(rmq[m][l], rmq[m][r - (1<<m) + 1]);
}

```

后缀树

```
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;
}
void getsa(int *a, int n, int m, int *sa){  //sa : 1 ~ n
    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; //源字符串长 n + n, 在末尾加 0
        getsa(a, n+n+2, 300, sa);  //加 0 后字符串最后一个字符在 n + n + 1
        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);
            printf("\n");
            }
        }
        for(i=ansb; i<ansb + ans; i++)
            printf("%c",a[i]);
        printf("\n");
    }
    return 0;
}
```

平衡树

```
/* 小的在左，大的在右。 */
#include"stdio.h"
#define NEWS (avltree *)malloc(sizeof(avltree))
typedef struct avltree
{ struct avltree *rc,*lc;
  long height,data,h,gao;
}avltree;
 FILE *input,*output;
long max(long a,long b) { if(a>b) return a; else return b;}
long min(long a,long b) { if(a<b) return a; else return b;}
long mheight(avltree *t){ if(t==NULL) return 0; else return
t->height;  }
avltree *singleft(avltree *t)
{ avltree *a;
  a=t->lc;
  t->lc=a->rc;
  a->rc=t;

  t->height=max(mheight(t->lc),mheight(t->rc))+1;
  a->height=max(mheight(a->lc),mheight(a->rc))+1;
  return a;
}
avltree *singright(avltree *t)
{ avltree *p;
  p=t->rc;
  t->rc=p->lc;
  p->lc=t;

  t->height=max(mheight(t->lc),mheight(t->rc))+1;
  p->height=max(mheight(p->lc),mheight(p->rc))+1;
  return p;
}
avltree *douleft(avltree *t)
{ t->lc=singright(t->lc);
  t=singleft(t);
  return t;
}
avltree *douright(avltree *t)
{ t->rc=singleft(t->rc);
  t=singright(t);
  return t;
}
avltree *insert(avltree *t,long key)
{ long i,j,k,r,w;
  avltree *p;
  if(t==NULL)
  { p=NEWS;
    p->height=1;
    p->data=key;
    p->lc=p->rc=NULL;
    return p;
  }
  if(key>t->data)
  { t->rc=insert(t->rc,key);
    if(mheight(t->rc) - mheight(t->lc) ==2)
    { if(key>t->rc->data) t=singright(t);
      else t=douright(t);
    }
  }
  else if(key<t->data)
  { t->lc=insert(t->lc,key);
    if(mheight(t->lc) - mheight(t->rc) ==2)
    { if(key<t->lc->data) t=singleft(t);
      else t=douleft(t);
    }
  }
  t->height=max(mheight(t->lc),mheight(t->rc))+1;
  return t;
}

int main()
{ long i,j,k,r,w,n;
  avltree *t=NULL;
  FILE *input,*output;
  input=fopen("avl.in","r");
  output=fopen("avl.out","w");
  fscanf(input,"%ld",&n);
  for(i=1;i<=n;i++)
  { 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];

       void init() {
           n = m = 0;
           memset(adj, 0, sizeof(adj));
       }
       void insert(int u, int v) {
           u++, v++;
           n = max(n, u);
           m = max(m, v);
           adj[u][v] = 1;
```

```
    }
    int find_ans() {
        build_dlx();
        ans = -1;
        if (dfs(0) ) {
            return ans;
        }
        return -1;
    }
    void out_ans(int ans) {
        if(ans == -1) {
            printf("NO\n");
            return ;
        }
        //printf("%d", n);
        repf (i, 0, ans - 1) {
            int x, y, ty;
            O[i]--;
            x = O[i] / (sm * stn);
            y = (O[i] % (sm * stn) ) / stn;
            ty = (O[i] % (stn));
            //printf("%d %d %d\n", x, y, ty);
            sudoku[x][y] = ty + 1;
        }
        rep (i, sn)
            rep (j, sm)
                printf("%d",sudoku[i][j]);
        printf("\n");
    }
private:
    int head;
    int R[Max], L[Max], U[Max], D[Max], C[Max], H[Max];
    int S[maxn];
    int n, m, cnt, nm;

    void add(int head, int tmp, int x) {
        H[cnt] = head;
        R[cnt] = tmp; L[cnt] = L[tmp];
        L[tmp] = cnt; R[L[cnt]] = cnt;
        U[cnt] = U[x]; D[cnt] = x;
        D[U[x]] = cnt; U[x] = cnt;
        C[cnt] = x; ++S[x];
        ++cnt;
    }
    void build_dlx() {
        L[0] = R[0] = U[0] = D[0] = C[0] = H[0] = 0;
        for (int i = 1; i <= m; i++) {
            H[i] = 0;
            L[i] = i - 1; R[i] = 0;
            R[i - 1] = i; L[0] = i;
            U[i] = D[i] = C[i] = i;
            S[i] = 0;
        }
        cnt = m + 1;
        for (int i = 1; i <= n; i++) {
            int tmp = Max - 1;
            L[tmp] = R[tmp] = U[tmp] = D[tmp] = C[tmp] = tmp;
            for (int j = 1; j <= m; j++)
                if(adj[i][j])   {
                    add(i, tmp, j);
                }
            L[R[tmp]] = L[tmp];
            R[L[tmp]] = R[tmp];
        }
    }
    void remove(const int &c) {
        R[L[c]] = R[c];
        L[R[c]] = L[c];
        for (int i = D[c]; i != c; i = D[i]) {
            for (int j = R[i]; j != i; j = R[j]) {
                U[D[j]] = U[j];
                D[U[j]] = D[j];
                --S[C[j]];
            }
        }
    }

    void resume(const int &c) {
        for (int i = D[c]; i != c; i = D[i]) {
            for (int j = R[i]; j != i; j = R[j]) {
                U[D[j]] = j;
                D[U[j]] = j;
                ++S[C[j]];
            }
        }
        R[L[c]] = c;
```

```
            L[R[c]] = c;
        }

        bool dfs(const int &k) {
            if (R[0] == 0){
                ans = k;
                return true;
            }
            int s(maxint), c;
            for (int i = R[0]; i != 0; i = R[i]) {
                if (S[i] < s) {
                    c = i;
                    s = S[i];
                }
            }
            remove(c);
            for (int i = D[c]; i != c; i = D[i]) {
                O[k] = H[i]; //
                for (int j = R[i]; j != i; j = R[j]) remove(C[j]);
                if (dfs(k + 1)) return true;
                for (int j = L[i]; j != i; j = L[j]) resume(C[j]);
            }
            resume(c);
            return false;
        }
}G;
char in[maxn * maxn];
int a[maxn][maxn];

void add(int x, int y, int ty) {
    int l_id = x * m * tn + y * tn + ty;
    //printf("%d %d %d %d\n", x, y, ty, l_id);
    int bn =  ((x / 3) * 3 + y / 3);
    G.insert(l_id, x * m + y);
    G.insert(l_id, x * tn + ty + n * m);         //row
    G.insert(l_id, n * tn + y * tn + ty + n * m);   //vertical
    G.insert(l_id, n * tn + m * tn + bn * tn + ty + n * m);  //block
}
int main(){
    while (scanf("%s", in) == 1) {
        if (in[0] == 'e')    break;
        rep (i, n)
            rep (j, m)
                if (in[i * m + j] == '.')    a[i][j] = 0;
                else  a[i][j] = in[i * m + j] - '0';

        G.init();
        rep (i, n)
            rep (j, m) {
                if(a[i][j] == 0) {
                    repf (k, 1, 9)
                        add(i, j, k - 1);
                }
                else add(i, j, a[i][j] - 1);
            }
        int ans = G.find_ans();
        G.out_ans(ans);
    }
    return 0;
}

```

Exact Cover

```
class Graph {
    public:
        static const int maxn = 1000 + 7;
        static const int maxm = 1000 + 7;
        static const int Max = maxn * maxm + 10;
        int adj[maxn][maxm];
        int ans;
        void init() {
            n = m = 0;
            memset(adj, 0, sizeof(adj));
        }
        void insert(int u, int v) {
            n = max(n, u);
            m = max(m, v);
            adj[u][v] = 1;
        }
        int find_ans() {
            build_dlx();
            ans = -1;
            if (dfs(0) ) {
```

```cpp
            return ans;
        }
        return -1;
    }
    void work(int n) {
        if(n == -1) {
            printf("NO\n");
            return ;
        }
        printf("%d", n);
        repf (i, 0, n - 1)
            printf(" %d", O[i]);
        printf("\n");
    }
private:
    int head;
    int R[Max], L[Max], U[Max], D[Max], C[Max], H[Max];
    int S[maxn], O[maxn];
    int n, m, cnt, nm;

    void add(int head, int tmp, int x) {
        H[cnt] = head;
        R[cnt] = tmp; L[cnt] = L[tmp];
        L[tmp] = cnt; R[L[cnt]] = cnt;
        U[cnt] = U[x]; D[cnt] = x;
        D[U[x]] = cnt; U[x] = cnt;
        C[cnt] = x; ++S[x];
        ++cnt;
    }
    void build_dlx() {
        L[0] = R[0] = U[0] = D[0] = C[0] = H[0] = 0;
        for (int i = 1; i <= m; i++) {
            H[i] = 0;
            L[i] = i - 1; R[i] = 0;
            R[i - 1] = i; L[0] = i;
            U[i] = D[i] = C[i] = i;
            S[i] = 0;
        }
        cnt = m + 1;
        for (int i = 1; i <= n; i++) {
            int tmp = Max - 1;
            L[tmp] = R[tmp] = U[tmp] = D[tmp] = C[tmp] = tmp;
            for (int j = 1; j <= m; j++)
                if(adj[i][j])   {
                    add(i, tmp, j);
                }
            L[R[tmp]] = L[tmp];
            R[L[tmp]] = R[tmp];
        }
    }
    void remove(const int &c) {
        R[L[c]] = R[c];
        L[R[c]] = L[c];
        for (int i = D[c]; i != c; i = D[i]) {
            for (int j = R[i]; j != i; j = R[j]) {
                U[D[j]] = U[j];
                D[U[j]] = D[j];
                --S[C[j]];
            }
        }
    }

    void resume(const int &c) {
        for (int i = D[c]; i != c; i = D[i]) {
            for (int j = R[i]; j != i; j = R[j]) {
                U[D[j]] = j;
                D[U[j]] = j;
                ++S[C[j]];
            }
        }
        R[L[c]] = c;
        L[R[c]] = c;
    }

    bool dfs(const int &k) {
        if (R[0] == 0){
            ans = k;
            return true;
        }
        int s(maxint), c;
        for (int i = R[0]; i != 0; i = R[i]) {
            if (S[i] < s) {
                c = i;
                s = S[i];
            }
        }
```

```
            remove(c);
            for (int i = D[c]; i != c; i = D[i]) {
                O[k] = H[i];
                for (int j = R[i]; j != i; j = R[j]) remove(C[j]);
                if (dfs(k + 1)) return true;
                for (int j = L[i]; j != i; j = L[j]) resume(C[j]);
            }
            resume(c);
            return false;
        }
}G;

```

Computational Geometry

凸包

```
bool operator < (const point &p) const{
    if(sgn(x - p.x) != 0) return x < p.x;
    else return y < p.y;
}
void convex(vector <point> a, vector <point> &tu){ //顺时针
    point hu[maxn], hd[maxn];
    int n = a.size(), un, dn;
    sort(a.begin(), a.end());
    hu[0] = hd[0] = a[0];
    hu[1] = hd[1] = a[1];
    un = dn = 1;
    for(int i = 2; i < n; i++){
        for(; un > 0 && sgn( (hu[un] - hu[un - 1]) * (a[i] - hu[un] )) >= 0; un--);
        for(; dn > 0 && sgn( (hd[dn] - hd[dn - 1]) * (a[i] - hd[dn] )) <= 0; dn--);
        hu[++un] = a[i];
        hd[++dn] = a[i];
    }
    tu.clear();
    for(int i = 0; i <= un - 1; i++) tu.push_back(hu[i]);
    for(int i = dn; i >= 1; i--) tu.push_back(hd[i]);
}

```

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);

```

半平面交

```
struct line {
    point p, v;
    double ang;
    line() {}
    line(point p, point v) : p(p), v(v) { ang = atan2(v.y, v.x); }
    bool operator < (const line &l) const { return ang < l.ang; }
};

//点 p 在有向直线 l 的左边（线上不算）
bool onLeft(line l, point p) {
    return sgn(l.v * (p - l.p)) > 0;
}

//二直线交点，假设交点唯一存在
point getIntersection(line a, line b) {
    point u = a.p - b.p;
    double t = (b.v * u) / (a.v * b.v);
    return a.p + a.v * t;
}

point p[maxn];
line q[maxn];
int halfPlane(vector<line> l, vector<point> &poly)
{ //l:anti-clockwise
    int n = sz(l);
    sort(l.begin(), l.end());
    int first, last;
    q[first = last = 0] = l[0];
    for (int i = 1; i < n; i++) {
        while (first < last && !onLeft(l[i], p[last - 1])) last--;
        while (first < last && !onLeft(l[i], p[first])) first++;
        q[++last] = l[i];
        if (sgn(q[last].v * q[last - 1].v) == 0) {
            last--;
            if (onLeft(q[last], l[i].p)) q[last] = l[i];
        }
        if (first < last) p[last - 1] = getIntersection(q[last - 1], 
q[last]);
    }
    while (first < last && !onLeft(q[first], p[last - 1])) last--;

    poly.clear();
    if (last - first <= 1) return 0;
    p[last] = getIntersection(q[last], q[first]);
    int m = 0;
    for (int i = first; i <= last; i++) {
        poly.push_back(p[i]);
        m++;
    }
    return m;
}
```

判断点是否在多边形内(old)

```
double trim(double d, double l = 1.0) {
    return d > l ? l : (d < -l ? -l : d);
}
int get_position(const point& p, const point* pol, int n) {
    double ang = 0;
    for (int i = 0; i < n; ++i) {
        if (pol[i] == p) return 0; //在点上
        point p1 = pol[i] - p, p2 = pol[(i + 1) % n] - p;
        double c = (p1 ^ p2) / (p1.len() * p2.len());
        c = trim(c);
        ang += sgn(p1 * p2) * acos(c);
    }
    ang = abs(ang);
    return ang < 0.5 * pi ? -1 : (ang < 1.5 * pi ? 0 : 1);
}
```

判断点是否在多边形内(new)

```
bool onSegment(const point &p, const point &s, const point &e) {
    if (p == s || p == e) return true;
    if (sgn((p - s) * (e - s)) == 0 && sgn((s - p) ^ (e - p)) <= 0)
        return true;
    return false;
}
int get_position(const point &p, point *pol, int n) {
    int wn = 0;
    for (int i = 0; i < n; i++) {
        if (onSegment(p, pol[i], pol[(i + 1) % n])) return 0; //on the segment
        int k = sgn((pol[(i + 1) % n] - pol[i]) * (p - pol[i]));
        int d1 = sgn(pol[i].y - p.y);
        int d2 = sgn(pol[(i + 1) % n].y - p.y);
        if (k > 0 && d1 <= 0 && d2 > 0) wn++;
        if (k < 0 && d2 <= 0 && d1 > 0) wn--;
    }
    if (wn != 0) return 1; //inner
    else return -1; //outter
}
```

异面线段距离

```
//返回直线距离的平方,返回一个分数。 node 是分数类
node gao3(point p, point a, point b) { //点到线段的距离
    if (p == a || p == b) return node(0);
    if (a == b) return node((p - a) ^ (p - a));
    point v1 = b - a, v2 = p - a, v3 = p - b;
    if ((v1 ^ v2) < 0) {
        return node(v2 ^ v2);
    }
    else if ((v1 ^ v3) > 0) {
        return node(v3 ^ v3);
    }
    else {
        return node((v1 * v2) ^ (v1 * v2)) / node(v1 ^ v1);
    }
}
node gao2(point a, point b, point c, point d) { //当线段之间没有垂线的距离
    node res = gao3(c, a, b);
    res = min(res, gao3(d, a, b));
    res = min(res, gao3(a, c, d));
    res = min(res, gao3(b, c, d));
    return res;
}

bool ok(node x) {
    if (x.zi * x.mu < 0) return false;
    if (x.zi < 0)
        x.zi *= -1, x.mu *= -1;
    return x.zi <= x.mu;
}

node gao(point a, point b, point l, point r) { //线段与线段之间的距离
    lint x0, y0, z0, x1, y1, z1, x2, y2, z2, x3, y3, z3;
    x0 = a.x, y0 = a.y, z0 = a.z;
    x1 = b.x, y1 = b.y, z1 = b.z;
    x2 = l.x, y2 = l.y, z2 = l.z;
    x3 = r.x, y3 = r.y, z3 = r.z;
    lint a1 = x1 - x0, a2 = x2 - x3, a3 = x0 - x2,
         a4 = y1 - y0, a5 = y2 - y3, a6 = y0 - y2,
         a7 = z1 - z0, a8 = z2 - z3, a9 = z0 - z2;
    lint  A = a1 * a1 + a4 * a4 + a7 * a7,
         B = a2 * a2 + a5 * a5 + a8 * a8,
         C = 2 * (a1 * a2 + a4 * a5 + a7 * a8),
         D = 2 * (a1 * a3 + a4 * a6 + a7 * a9),
        E = 2 * (a2 * a3 + a5 * a6 + a8 * a9),
        F = a3 * a3 + a6 * a6 + a9 * a9;
    if ((a1 * a5 == a2 * a4 && a1 * a8 == a2 * a7 && a4 * a8 == a5 * a7))
        return gao2(a, b, l, r);

    lint Y = C * D - 2 * A * E,
        X = C * E - 2 * B * D;
    if (!ok(node(Y, 4 * A * B - C * C)) || !ok(node(X, 4 * A * B - C * C)))
        return gao2(a, b, l, r);
    lint S = A * X * X + B * Y * Y + C * X * Y + D * X * (4 * A * B - C * C) + E * Y * (4 * A * B - C * C) + F * (4 * A * B - C * C) * (4 * A * B - C * C);
    return node(S, (4 * A * B - C * C) * (4 * A * B - C * C));
}

```

圆的面积并

```
const int zx[] = {0, 1, 0, -1};
const int zy[] = {1, 0, -1, 0};

int sgn(double x) { return (x > eps) - (x < -eps); }
void get_min(double& x, double y) { x = min(x, y); }
void get_max(double& x, double y) { x = max(x, y); }
struct P {
    double x, y;
    P() {}
    P(double _x, double _y): x(_x), y(_y) {}
    P operator + (const P &a) const { return P(x + a.x, y + a.y); }
    P operator - (const P &a) const { return P(x - a.x, y - a.y); }
    P operator * (const double &m) const { return P(x * m, y * m); }
    P operator / (const double &m) const { return P(x / m, y / m); }
    P set(const double &m) const {
        double len = length();
        return P(x * m / len, y * m / len);
    }
    P turn(const double &m) const {
        double c = cos(m), s = sin(m);
        return P(x * c - y * s, x * s + y * c);
    }
    bool operator == (const P &p) const { return sgn(x - p.x) == 0 && sgn(y - p.y) == 0;
    }
    double length() const {
        return sqrt(x * x + y * y);
    }
    double dist(const P &a) const {
        return sqrt(SQR(x - a.x) + SQR(y - a.y));
    }
    double cross(const P &a, const P &b) const { return (a.x - x) * (b.y - y) - (a.y - y) * (b.x - x); }
    double cross(const P &a) const { return x * a.y - y * a.x; }
    double dot(const P &a, const P &b) { return (a.x - x) * (b.x - x) + (a.y - y) * (b.y - y); }
    void input() { scanf("%lf%lf", &x, &y); }
    void output() const { printf("(%lf, %lf)\n", x, y); }
    P trunc(double l) const {
        double r = l / length();
        return P(x * r, y * r);
    }
    P turn_left() const { return P(-y, x); }
    P rotate_left(double ang) const {
        double c = cos(ang), s = sin(ang);
        return P(x * c - y * s, y * c + x * s);
    }
    P turn_right() const { return P(y, -x); }
    P rotate_right(double ang) const {
        double c = cos(ang), s = sin(ang);
        return P(x * c + y * s, y * c - x * s);
    }
};

double dist2(const P &a, const P &b) {
    return SQR(a.x - b.x) + SQR(a.y - b.y);
}
double dist(const P &a, const P &b) {
    return sqrt(SQR(a.x - b.x) + SQR(a.y - b.y));
}
double cross(const P &a, const P &b, const P &c) {
    return (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (c.x - a.x);
}
double dmul(const P &a, const P &b, const P &c) {
    return (b.x - a.x) * (c.x - a.x) + (b.y - a.y) * (c.y - a.y);
```

```cpp
}

int NEXT(int x, int n) {
    return x % n;
}

struct C {
    P mid;
    double r;
    C(const P &_mid, const double &_r)
        :mid(_mid), r(_r) {}
    C() {}
    bool operator == (const C &a) const {
        return mid == a.mid && sgn(r - a.r) == 0;
    }
    bool in(const C &a) const {
        return sgn(r + dist(mid, a.mid) - a.r) <= 0;
    }
    const C &input() {
        mid.input();
        scanf("%lf", &r);
        return *this;
    }
    const C &output() const {
        printf("P: %.12lf %.12lf R: %.12lf\n", mid.x, mid.y, r);
    }
};
double cal_angle(const C &c, const P &a, const P &b) {
    double k = dmul(c.mid, a, b) / SQR(c.r);
    get_min(k, 1.0);
    get_max(k, -1.0);
    return acos(k);
}
double cal_area(const C &c, const P &a, const P &b) {
    return SQR(c.r) * cal_angle(c, a, b) / 2 - cross(c.mid, a, b) /
2;
}
struct cmp {
    P mid;
    cmp(const P &_mid)
        :mid(_mid) {}
    bool operator () (const P &a, const P &b) {
        return atan2(a.y - mid.y, a.x - mid.x) < atan2(b.y - mid.y,
b.x - mid.x);
    }
};
bool circles_intersection(const C &a, const C &b, P &c1, P &c2) {
    double dd = dist(a.mid, b.mid);
    if (sgn(dd - (a.r + b.r)) >= 0) {
        return false;
    }
    double l = (dd + (SQR(a.r) - SQR(b.r)) / dd) / 2;
    double h = sqrt(SQR(a.r) - SQR(l));
    c1 = a.mid + (b.mid - a.mid).trunc(l) + (b.mid -
a.mid).turn_left().trunc(h);
    c2 = a.mid + (b.mid - a.mid).trunc(l) + (b.mid -
a.mid).turn_right().trunc(h);
    return true;
}
bool cover(const C &c, const P &a, const P &b, const vector<C> &cir)
{
    P p = c.mid + ((a + b) / 2 - c.mid).trunc(c.r);
    for (vector<C>::const_iterator it = cir.begin(); it != cir.end();
++it) {
        if (sgn(dist2(p, it->mid) - SQR(it->r)) < 0) {
            return true;
        }
    }
    return false;
}
double cal_area(const vector<C> &in) {
    vector<C> cir;
    for (int i = 0; i < SZ(in); ++i) {
        if (sgn(in[i].r) == 0) {
            continue;
        }
        bool flag = false;
        for (int j = i + 1; j < SZ(in); ++j) {
            if (in[i] == in[j]) {
                flag = true;
                break;
            }
        }
        if (flag) {
            continue;
        }
        for (int j = 0; j < SZ(in); ++j) {
            if (!(in[i] == in[j]) && in[i].in(in[j])) {
                flag = true;
                break;
            }
        }
        if (flag) {
            continue;
        }
        cir.push_back(in[i]);
    }
    vector<vector<P> > point_on_circle(SZ(cir));
    for (int i = 0; i < SZ(cir); ++i) {
        for (int z = 0; z < 4; ++z) {
            point_on_circle[i].push_back(cir[i].mid + P(zx[z], zy[z]).trunc(cir[i].r));
        }
    }
    for (int i = 0; i < SZ(cir); ++i) {
        for (int j = i + 1; j < SZ(cir); ++j) {
            P a, b;
            if (circles_intersection(cir[i], cir[j], a, b)) {
                point_on_circle[i].push_back(a);
                point_on_circle[i].push_back(b);
                point_on_circle[j].push_back(a);
                point_on_circle[j].push_back(b);
            }
        }
    }
    for (int i = 0; i < SZ(cir); ++i) {
        sort(point_on_circle[i].begin(), point_on_circle[i].end(), cmp(cir[i].mid));
        point_on_circle[i].erase(unique(point_on_circle[i].begin(), point_on_circle[i].end()), point_on_circle[i].end());
    }
    double ans = 0;
    for (int i = 0; i < SZ(cir); ++i) {
        for (int j = 0; j < SZ(point_on_circle[i]); ++j) {
            const P &a = point_on_circle[i][j];
            const P &b = point_on_circle[i][NEXT(j + 1, SZ(point_on_circle[i]))];
            if (!cover(cir[i], a, b, cir)) {
                ans += cross(P(0, 0), a, b) / 2;
                ans += cal_area(cir[i], a, b);
            }
        }
    }
    return ans;
}

```

Math

miller_rabin_and_Pollard_rho

```cpp
//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;
```

```c
        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;
}
```

```cpp
//////////////////////////////////////////////////////////////////////
u64 factor[MAXN], m;
/////////////////////////////////////////////////////////////////
//分解质数因子
u64 prime(u64 a)
{
    if(Miller_Rabin(a) || a == 0) return 0;
    u64 t = Pollard(a), p;
    if(p = prime(t)) return p;
    else return t;
}


//622057148 155514287 会跪
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);
            if (t == 0) break;
            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

```cpp
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

```cpp
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() { //记得先给 N,M 赋值
        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
        }
    return c;
}

```

二&三维旋转

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)

```
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

```
int gauss(int map[40][40],int ans[40])
{
   int i,j,k,r,w;
   for(k=0;k<30;k++)
   {  i = k;
      while(i<30 && map[i][k] == 0)  i++;
      if(i == 30) continue;
      if(i > k)
      {  for(j=0;j<=30;j++)
            swap(map[i][j],map[k][j]);
      }
      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);
}
void gcd(lint a, lint b, lint& d, lint& x, lint& y)  {
    if (!b) { d = a; x = 1; y = 0; }
    else {gcd(b, a % b, d, y, x); y -= x * (a / b); }
}

lint inv(lint a, lint n) {
    lint d, x, y;
    gcd(a, n, d, x, y);
    return d == 1 ? (x + n) % n : -1;
}
```

Mobius

```
lint v[maxn];
lint mob[maxP];
void getMobius() {
    memset(mob, 0, sizeof(mob));
    memset(v, 0, sizeof(v));
    mob[1] = 1;
    for (lint i = 2; i < maxn; i++) {
        if (v[i] == 0) {
            for (lint j = i + i; j < maxn; j += i) {
                v[j] = 1;
                mob[j] = mob[j / i] * -1;
            }
            mob[i] = -1;
        }
    }
}
```

logMod （a ^ x = b (mod n) ）(含逆元)

```
void gcd(lint a, lint b, lint &d, lint &x, lint &y) {
    if (!b) {d = a; x = 1; y = 0;}
    else { gcd(b, a % b, d, y, x); y -= x * (a / b);}
}

lint inv(lint a, lint n) {
    lint d, x, y;
    gcd(a, n, d, x, y);
    return d == 1 ? (x + n) % n : -1;
}

lint mulMod(lint a, lint b, lint m = mod) { // a * b % m;
    return a * b % m;
}

lint powMod(lint a, lint b, lint m = mod) { // a ^ b % m;
    lint res = 1;
    while (b != 0) {
        if (b & 1) {
            res = (res * a) % m;
        }
        a = (a * a) % m;
        b >>= 1;
    }
    return res;
}


lint logMod(lint a, lint b, lint n = mod) { //a ^ x = b (mod n)
    lint m, v, e = 1, i;
    m = (int)sqrt(n + 0.5);
    v = inv(powMod(a, m, n), n);
    map <lint, lint > x;
    x.clear();
    x[1] = 0;
    for (lint i = 1; i < m; i++) {
        e = mulMod(e, a, n);
        if (!x.count(e)) x[e] = i;
    }
    for (lint i = 0; i < m; i++) {
        if (x.count(b)) return i * m + x[b];
        b = mulMod(b, v, n);
    }
    return -1;
}
```

Java

MAP

```
public static Map<BigInteger, BigInteger> dic;
public static BigInteger gao(BigInteger n) {
    if (dic.containsKey(n) == false) {
        dic.put(n, res3);
    }
    return dic.get(n);
}
public static void main(String[] args) {
    dic = new HashMap();
    dic.clear();
    gao(x);
}
```

分数操作

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

public class Main {
    public final static int maxn = 50 + 10;
    public final static int lim_m = 50;
    public final static BigInteger ZERO = new BigInteger("0");
    public final static BigInteger ONE = new BigInteger("1");

    public static void updata(int i, int j, int r, int w, BigInteger[][] mu, BigInteger[][] zi, BigInteger scope) {
        BigInteger nmu = mu[r][w].multiply(scope);
        BigInteger new_mu = mu[i][j].multiply(nmu);
        BigInteger new_zi = mu[i][j].multiply(zi[r][w]);
        new_zi = new_zi.add( nmu.multiply(zi[i][j]) );

        //BigInteger h = new_mu.gcd(new_zi);
        mu[i][j] = new_mu;
        zi[i][j] = new_zi.divide(h);
    }

    public static void main(String[] args) {
        BigInteger[][] mu = new BigInteger[maxn][maxn], zi = new BigInteger[maxn][maxn];
        BigInteger scope = ZERO;
        int[] x = new int[maxn];
        int n, m, a, b;
        Scanner cin = new Scanner(System.in);
        while (cin.hasNextInt()) {
            n = cin.nextInt();
            m = cin.nextInt();
            a = cin.nextInt();
            b = cin.nextInt();
            for (int i = 0; i < n; i++)
                x[i] = cin.nextInt();

            for (int i = 0; i <= n + 1; i++)
                for (int j = 0; j <= lim_m; j++)
                    mu[i][j] = ONE;
            for (int i = 0; i <= n + 1; i++)
            for (int j = 0; j <= lim_m; j++)
                zi[i][j] = ZERO;
        zi[0][0] = ONE;
        scope = scope.valueOf(b - a + 1);

        for (int i = 0; i < n; i++) {
            for (int j = a; j <= b; j++) {
                int dis = Math.abs(x[i] - j);
                for (int k = 0; k <= lim_m; k++)
                    if (k - dis >= 0) {
                        updata(i + 1, k, i, k - dis, mu, zi, scope);
                    }
            }
        }

        BigInteger ans_mu = ONE, ans_zi = ZERO;
        for (int i = 0; i <= m; i++) {
            ans_zi = ans_zi.multiply(mu[n][i]);
            ans_zi = ans_zi.add(zi[n][i].multiply(ans_mu));
            ans_mu = ans_mu.multiply(mu[n][i]);
            BigInteger h = ans_zi.gcd(ans_mu);
            ans_zi = ans_zi.divide(h);
            ans_mu = ans_mu.divide(h);
        }

        System.out.println(ans_zi + "/" + ans_mu);
    }
  }
}
```

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;
}

```

Booth(int64 乘 int64 余 int64)

```
inline long long mul(long long lhs, long long rhs) {
   long long lhs2 = lhs % 100000;
   long long rhs2 = rhs % 100000;
   return ((lhs / 100000 * rhs2 + rhs / 100000 * lhs2) * 100000 + lhs2 * rhs2) % MOD;
}

```

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;
}

int scanf(int &num) { //负数
    char in;
    int op = 1;
    while ((in = getchar()) != EOF && !(('0' <= in && in <= '9') || in == '-'));
    if (in == EOF) return 0;
    if (in == '-') {
        op = -1;
        in = getchar();
    }
    num = in - '0';
    while (in = getchar(), in >= '0' && in <= '9') num *= 10, num += in - '0';
    num *= op;
    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 (int i = 0; i < (n); ++i)
#define repf(i, a, b) for (int i = (a); i <= (b); ++i)
#define repd(i, a, b) for (int 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;
const int maxint = -1u>>1;

int sgn(double x) {
    return (x > eps) - (x < -eps);
}

===============================================================

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， set======================
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;
}

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

set< pair<int, int> > st;
set< pair<int, int> >::reverse_iterator it
it = st.rbegin()

=====================================================
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(); 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


===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() );
```

================

```
sprintf(ch,"%.15lf\n",ans);
```
把数字转成字符串

Exp(x)　e 的 x 次方

====long double======

windowns 下不能输出 long double

Linux %Lf

=============栈空间================

```
#pragma comment(linker, "/STACK:102400000,102400000")
```

=============合并=========

```
accumulate(numbers.begin(), numbers.end(), init);
```

===============hash_map==============

```
#include <utility>
#include <ext/hash_map>
hash_map<int,int> mp ;
```

===============流=========

```
ios::sync_with_stdio(false);
```

===========数学函数==============

```
hypot(float x, float y)
```
对于给定的直角三角形的两个直角边，求其斜边的长度

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 desert
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()
    if (getline('.')[0] == '/')
        normal ^xx
    else
        normal 0i//
    endif
endfunc
```