|  |
| --- |
|  |

|  |
| --- |
| ACM/ICPC at Wuhan University |
| Xioumu STL(code) |
| Created by xioumu, for OpenShield |

目录

[Graph 3](#_Toc369772215)

[2-sat 3](#_Toc369772216)

[N\*log(n) Dijkstra 4](#_Toc369772217)

[双联通分量 5](#_Toc369772218)

[KM 6](#_Toc369772219)

[DataStructure 7](#_Toc369772220)

[数状数组 7](#_Toc369772221)

[RMQ 7](#_Toc369772222)

[后缀树 7](#_Toc369772223)

[平衡树 8](#_Toc369772224)

[线段树-扫描线矩形面积并 9](#_Toc369772225)

[大根堆 11](#_Toc369772226)

[DXL 11](#_Toc369772227)

[Computational Geometry 15](#_Toc369772228)

[凸包 15](#_Toc369772229)

[线段相交 15](#_Toc369772230)

[最近点对 15](#_Toc369772231)

[线段与线段的距离 16](#_Toc369772232)

[O(N^2)处理最少用几段弧完全覆盖一个圆 16](#_Toc369772233)

[半平面交 17](#_Toc369772234)

[判断点是否在多边形内(old) 17](#_Toc369772235)

[判断点是否在多边形内(new) 18](#_Toc369772236)

[异面线段距离 18](#_Toc369772237)

[圆的面积并 19](#_Toc369772238)

[Math 21](#_Toc369772239)

[miller\_rabin\_and\_Pollard\_rho 21](#_Toc369772240)

[get\_prime 23](#_Toc369772241)

[Matrix 23](#_Toc369772242)

[Gauss 24](#_Toc369772243)

[GCD&扩展GCD 25](#_Toc369772244)

[辛普森积分 25](#_Toc369772245)

[Mobius 25](#_Toc369772246)

[logMod （a ^ x = b (mod n) ）(含逆元) 25](#_Toc369772247)

[Java 26](#_Toc369772248)

[MAP 26](#_Toc369772249)

[分数操作 26](#_Toc369772250)

[Others 27](#_Toc369772251)

[O(n)求回文串 27](#_Toc369772252)

[KMP 27](#_Toc369772253)

[Booth(int64乘int64余int64) 28](#_Toc369772254)

[读入优化 28](#_Toc369772255)

[乱七八糟 28](#_Toc369772256)

[Vimrc 30](#_Toc369772257)

# Graph

### 2-sat

**1 int n, m;**

**2 vector<int> e[maxn], g[maxn], op[maxn];**

**3 void add(vector<int> \*e, int x, int y){**

**4 e[x].push\_back(y);**

**5 }**

**6 void get(int &x, inat &nx){**

**7 if(x < 0){**

**8 x = -x;**

**9 nx = x + n;**

**10 }**

**11 else {**

**12 nx = x;**

**13 x += n;**

**14 }**

**15 }**

**16 int sta[maxn], low[maxn], dfn[maxn], v[maxn], fen[maxn], du[maxn], co[maxn];**

**17 int top, num, fn;**

**18 void tar(vector<int> \*e, int w){**

**19 sta[++top] = w;**

**20 low[w] = dfn[w] = ++num;**

**21 v[w] = 1;**

**22 rep (i, sz(e[w]) ) {**

**23 int j = e[w][i];**

**24 if(v[j] == 2) continue;**

**25 if( dfn[j] == -1) tar(e, j);**

**26 low[w] = min(low[w], low[j]);**

**27 }**

**28**

**29 if(dfn[w] == low[w]){**

**30 fn++;**

**31 do{**

**32 fen[ sta[top] ] = fn;**

**33 v[ sta[top] ] = 2;**

**34 top--;**

**35 }while( sta[top + 1] != w);**

**36 }**

**37 }**

**38 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;**

**39 memset(dfn, -1, sizeof(dfn));**

**40 memset(low, -1, sizeof(low));**

**41 memset(v, 0, sizeof(v));**

**42 num = top = fn = 0;**

**43 repf (i, 1, 2 \* n)**

**44 if(dfn[i] == -1){**

**45 tar(e, i);**

**46 }**

**47 repf (i, 1, fn) {**

**48 g[i].clear();**

**49 op[i].clear();**

**50 }**

**51 memset(du, 0, sizeof(du));**

**52 repf (i, 1, 2 \* n){**

**53 int ni;**

**54 if(i > n) ni = i - n;**

**55 else ni = i + n;**

**56 if(fen[i] == fen[ni]) return false;**

**57 add(op, fen[i], fen[ni]);**

**58 rep (j, sz(e[i])){**

**59 int k = e[i][j];**

**60 if( fen[i] != fen[k] ){**

**61 add(g, fen[k], fen[i]);**

**62 du[ fen[i] ]++;**

**63 }**

**64 }**

**65 }**

**66 return true;**

**67 }**

**68 void updata(vector<int> \*e, int w){**

**69 if(co[w] != 0){**

**70 return ;**

**71 }**

**72 co[w] = 2;**

**73 rep (i, sz(e[w]) ){**

**74 int j = e[w][i];**

**75 du[j]--;**

**76 updata(e, j);**

**77 }**

**78 }**

**79 void dye(vector<int> \*e){**

**80 top = 0;**

**81 repf (i, 1, fn)**

**82 if( du[i] == 0)**

**83 sta[++top] = i;**

**84 memset(co, 0, sizeof(co));**

**85 while(top != 0){**

**86 int k = sta[top--];**

**87 if( co[k] != 0) continue;**

**88 else{**

**89 co[k] = 1;**

**90 rep (i, sz(op[k])){**

**91 updata(e, op[k][i]);**

**92 }**

**93 }**

**94 rep (i, sz(e[k])){**

**95 int j = e[k][i];**

**96 du[j]--;**

**97 if(du[j] == 0)**

**98 sta[++top] = j;**

**99 }**

**100 }**

**101 }**

**102 int main(){**

**103 if( !shrink(e, g) ){**

**104 printf("No\n");**

**105 }**

**106 else {**

**107 printf("Yes\n");**

**108 dye(g);**

**109 vector<int> ans;**

**110 repf (i, n + 1, 2 \* n)**

**111 if(co[ fen[i] ] == 1){**

**112 ans.push\_back(i - n);**

**113 }**

**114 printf("%d", sz(ans));**

**115 rep (i, sz(ans)){**

**116 printf(" %d", ans[i]);**

**117 }**

**118 printf("\n");**

**119 }**

**120**

**121 return 0;**

**122 }**

### N\*log(n) Dijkstra

**1 long long v[MAXN],dis[MAXN],dui[MAXN],rear,front,dn,b[MAXN];**

**2 void up(long long x)**

**3 {**

**4 long long i,j,k;**

**5 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**

### 双联通分量

1 #include<cstdio>

2 #include<cstring>

3 #include<cstdlib>

4 #include<algorithm>

5 #define MAXN 1007

6 **using** **namespace** std;

7 **int** a[MAXN][MAXN],f[MAXN];

8 **int** n,m,ans;

9 **void** init(){

10 **int** i,j,k,r,w;

11 **for**(i=1;i<=n;i++)

12 **for**(j=i+1;j<=n;j++)

13 a[i][j] = a[j][i] = 1;

14 **for**(i=1;i<=m;i++){

15 scanf("%d %d",&r,&w);

16 a[r][w] = a[w][r] = 0;

17 }

18 }

19 **int** zhan[MAXN],top,v[MAXN],df[MAXN],low[MAXN],num;

20 **int** d[MAXN];

21 **bool** pan(**int** w) {

22 **int** i,j,k;

23 **for**(i=1;i<=n;i++){

24 **if**(a[w][i] && v[i] != 0){

25 **if**(v[i] == 1){

26 v[i] = (v[w]-1)%2 + 2;

27 **if**( !pan(i) ) **return** false;

28 }

29 **else** **if**( (v[w]-1)%2 + 2 != v[i])

30 **return** false;

31 }

32 }

33 **return** true;

34 }

35 **void** dfs(**int** w,**int** fa){

36 **int** i,j,k,r;

37 df[w] = low[w] = ++num;

38 zhan[++top] = w;

39 **for**(i=1;i<=n;i++)

40 **if**(a[w][i] && i != fa){

41 **if**(df[i] == 0){

42 dfs(i,w);

43 low[w] = min(low[w],low[i]);

44 **if**(low[i] >= df[w]){

45 memset(v,0,**sizeof**(v));

46 k = top;

47 **do**{

48 v[ zhan[top] ] = 1;

49 top--;

50 }**while**(zhan[top+1] != i);

51 v[w]=1;

52

53 **if**(!pan(w) ){

54 **for**(k=1;k<=n;k++)

55 **if**(v[k] >= 1) {

56 d[k] = 1;

57 }

58 }

59

60 }

61 }

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

63 }

64

65 }

66 **void** solve(){

67 **int** i,j,k,r,w;

68 ans = 0;

69 memset(f,0,**sizeof**(f));

70 top = num = 0;

71 memset(df,0,**sizeof**(df));

72 memset(low,0,**sizeof**(low));

73 memset(v,0,**sizeof**(v));

74 memset(d,0,**sizeof**(d));

75 **for**(i=1;i<=n;i++){

76 **if**(df[i] == 0){

77 dfs(i,0);

78 }

79 }

80 **for**(i=1;i<=n;i++)

81 **if**(d[i] == 0){

82 ans++;

83 }

84 printf("%d\n",ans);

85 }

86 **int** main() {

87 **while**(scanf("%d %d",&n,&m) != EOF && n && m){

88 init();

89 solve();

90 }

91 **return** 0;

92 }

### KM

1 **struct** Graph {

2 **int** w[maxn][maxn], lx[maxn], ly[maxn], matx[maxn], maty[maxn], slk[maxn], n;

3 **bool** fx[maxn], fy[maxn];

4 **void** get\_max(**int** &x, **int** y) {

5 x = max(x, y);

6 }

7 **void** get\_min(**int** &x, **int** y) {

8 x = min(x, y);

9 }

10 **void** clear() {

11 memset(w, 0, **sizeof**(w));

12 n = 0;

13 }

14 **void** insert(**int** u, **int** v, **int** c) {

15 get\_max(n, max(u + 1, v + 1));

16 w[u][v] = c;

17 }

18 **int** match() {

19 memset(ly, 0, **sizeof**(ly));

20 **for** (**int** i = 0; i < n; ++i) {

21 lx[i] = -maxint;

22 **for** (**int** j = 0; j < n; ++j) {

23 get\_max(lx[i], w[i][j]);

24 }

25 }

26 memset(matx, -1, **sizeof**(matx));

27 memset(maty, -1, **sizeof**(maty));

28 **for** (**int** i = 0; i < n; ++i) {

29 memset(fx, false, **sizeof**(fx));

30 memset(fy, false, **sizeof**(fy));

31 **for** (**int** j = 0; j < n; j++)

32 slk[j] = maxint;

33 **if** (!dfs(i)) {

34 --i;

35 **int** p = maxint;

36 **for** (**int** j = 0; j < n; j++)

37 **if** (fy[j] == false)

38 p = min(p, slk[j]);

39 **for** (**int** j = 0; j < n; ++j) {

40 ly[j] += fy[j] \* p;

41 }

42 **for** (**int** k = 0; k < n; ++k) {

43 lx[k] -= fx[k] \* p;

44 }

45 }

46 }

47 **int** ans = 0;

48 **for** (**int** i = 0; i < n; ++i) {

49 ans += w[maty[i]][i];

50 }

51 **return** ans;

52 }

53 **bool** dfs(**int** u) {

54 fx[u] = 1;

55 **for** (**int** v = 0; v < n; ++v) {

56 **if** (lx[u] + ly[v] > w[u][v]) {

57 **if** (lx[u] + ly[v] - w[u][v] < slk[v])

58 slk[v] = lx[u] + ly[v] - w[u][v];

59 }

60 **else** **if** (lx[u] + ly[v] == w[u][v] && fy[v] == false) {

61 fy[v] = true;

62 **if** (maty[v] == -1 || dfs(maty[v])) {

63 matx[u] = v;

64 maty[v] = u;

65 **return** true;

66 }

67 }

68 }

69 **return** false;

70 }

71 }G;

72

# 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

1 **int** a[MAXN], height[MAXN], myrank[MAXN], sa[MAXN];

2 **int** wa[MAXN], wb[MAXN], wv[MAXN], wws[MAXN];

3 **int** rmq[100][MAXN];

4 **int** n;

5 **bool** cmp(**int** \*wb, **int** a, **int** b, **int** l, **int** n){

6 **int** r,w;

7 r = a + l >= n ? 0 : wb[a+l];

8 w = b + l >= n ? 0 : wb[b+l];

9 **return** wb[a] == wb[b] && r == w;

10 }

11 **void** getsa(**int** \*a, **int** n, **int** m, **int** \*sa){ //sa : 1 ~ n

12 **int** i,j,k,r,w,p;

13 **for**(i=0; i<=m; i++) wws[i] = 0;

14 **for**(i=0; i<n; i++) wws[ wa[i] = a[i] ]++;

15 **for**(i=1; i<=m; i++) wws[i] += wws[i-1];

16 **for**(i=n-1; i>=0; i--) sa[ --wws[ wa[i] ] ] = i;

17 **for**(j=1,p=1; j<n&&p<n; j\*=2,m=p){ //特别注意要写m=p

18 **for**(i=n-j,p=0; i<n; i++) wb[p++] = i;

19 **for**(i=0; i<n; i++) **if**(sa[i] >= j) wb[p++] = sa[i] - j;

20 **for**(i=0; i<=m; i++) wws[i] = 0;

21 **for**(i=0; i<n; i++) wv[i] = wa[ wb[i] ];

22 **for**(i=0; i<n; i++) wws[ wv[i] ]++;

23 **for**(i=1; i<=m; i++) wws[i] += wws[i-1];

24 **for**(i=n-1; i>=0; i--) sa[ --wws[ wv[i] ] ] = wb[i];

25 **for**(i=0; i<n; i++) wb[i] = wa[i];

26 **for**(i=1,p=1,wa[ sa[0] ] = 0; i<n; i++)

27 wa[ sa[i] ] = cmp(wb, sa[i], sa[i-1], j, n) ? p-1 : p++;

28 }

29 }

30 **void** getheight(**int** \*a, **int** \*sa, **int** n, **int** \*height){

31 **int** i,j,k,r,w;

32 k = 0;

33 **for**(i=0; i<=n; i++) myrank[ sa[i] ] = i;

34 **for**(i=0; i<n; height[ myrank[i++] ] = k)

35 **for**(k ? k-- : 0, j = sa[ myrank[i] - 1]; a[i+k] == a[j+k]; k++);

36 }

37 **void** getrmq(**int** \*height, **int** n, **int** rmq[100][MAXN]){

38 **int** i,j,k,r,m;

39 m = (**double**)log((**double**)n+1) / (**double**)log(2.0);

40 **for**(i=0; i<=m; i++)

41 **for**(j=0; j<=n; j++)

42 rmq[i][j] = 200000000;

43 **for**(i=0; i<=n; i++){

44 rmq[0][i] = height[i];

45 }

46 **for**(i=1; i<=m; i++)

47 **for**(j=0; j<=n - (1<<(i-1)) + 1; j++)

48 rmq[i][j] = min(rmq[i-1][j], rmq[i-1][j + ( 1 << (i-1) )]);

49 }

50 **int** find(**int** rmq[100][MAXN], **int** l, **int** r){

51 **if**(l > r) swap(l, r);

52 l++;

53 **int** m = (**double**)log((**double**)r-l+1)/(**double**)log(2.0);

54 **return** min(rmq[m][l], rmq[m][r - (1<<m) + 1]);

55 }

56 **int** main(){

57 **char** s[MAXN];

58 **int** i,j,k;

59 **while**(scanf(" %s",s) != EOF){

60 memset(a,0,**sizeof**(a));

61 n = strlen(s);

62 **for**(i=0; i<n; i++) a[i] = s[i];

63 a[n] = 200;

64 **for**(i=n+1; i<=n+n; i++) a[i] = s[n + n - i];

65 a[n+n+1] = 0; //源字符串长n + n，在末尾加0

66 getsa(a, n+n+2, 300, sa); //加0后字符串最后一个字符在n + n + 1

67 getheight(a, sa, n+n+1, height);

68 getrmq(height, n+n+1, rmq);

69 **int** ans = -1, ansb;

70 **for**(i=0; i<n; i++){

71 k = find(rmq, myrank[i], myrank[n + n - i]);

72 **if**(ans < 2\*k - 1){

73 ans = 2 \* k - 1;

74 ansb = i - k + 1;

75 }

76 k = find(rmq, myrank[i], myrank[n + n - i - 1]);

77 **if**(ans < (k-1) \* 2 ){

78 ans = (k-1) \* 2;

79 ansb = i - (k-2);

80 printf("\n");

81 }

82 }

83 **for**(i=ansb; i<ansb + ans; i++)

84 printf("%c",a[i]);

85 printf("\n");

86 }

87 **return** 0;

88 }

### 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);**

**79 t=insert(t,r);**

**80 }**

**81 fclose(input);**

**82 fclose(output);**

**83 return 0;**

**84 }**

**85**

**86**

### 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！**

**1 #include<cstdio>**

**2 #include<cstring>**

**3 #include<cstdlib>**

**4 #include<cmath>**

**5 #include<algorithm>**

**6 #include<string>**

**7 #include<vector>**

**8 using namespace std;**

**9 #define inf 1e-8**

**10 #define MAXN 2007**

**11 typedef long long int64;**

**12 int sgn(double x){**

**13 return x > inf ? 1: (x < -inf ? -1 : 0);**

**14 }**

**15 struct node{**

**16 double x,l,r;**

**17 int t;**

**18 node(double \_l, double \_r, double \_x,int \_t) : l(\_l), r(\_r), x(\_x), t(\_t) {}**

**19 bool operator < (const node &b) const {**

**20 return sgn(x- b.x) < 0;**

**21 }**

**22 };**

**23 vector<node> a;**

**24 int lazy[MAXN];**

**25 int cut[MAXN];**

**26 double fx[MAXN],fy[MAXN],sum[MAXN],num[MAXN],y[MAXN],ww[MAXN];**

**27 int n,m;**

**28 void init(){**

**29 int i,j,k,r,w;**

**30 double x1,y1,x2,y2;**

**31 double x[MAXN];**

**32 memset(lazy,0,sizeof(lazy));**

**33 m = 0;**

**34 a.clear();**

**35 for(i=0; i<n; i++){**

**36 scanf("%lf %lf %lf %lf",&x1,&y1,&x2,&y2);**

**37 a.push\_back( node(y1, y2, x1, 1) );**

**38 a.push\_back( node(y1, y2, x2, -1) );**

**39 y[++m] = y2;**

**40 x[m] = x1;**

**41 y[++m] = y1;**

**42 x[m] = x2;**

**43 }**

**44 sort(a.begin(), a.end());**

**45 sort(y+1, y+m+1);**

**46 fy[1] = y[1];**

**47 w = 1;**

**48 for(i=2; i<=m; i++){**

**49 if(sgn(y[i] - y[i-1]) != 0)**

**50 fy[++w] = y[i];**

**51 }**

**52 memcpy(y, fy, sizeof(y));**

**53 m = w;**

**54 memset(fy,0,sizeof(fy));**

**55 for(i=1; i<m; i++)**

**56 fy[i] = fy[i-1] + y[i+1] - y[i];**

**57**

**58 memset(num, 0, sizeof(num));**

**59 for(i=1; i<=m; i++)**

**60 num[i] = fy[i];**

**61 }**

**62 void getch(int t, int &lc, int &rc){**

**63 lc = t<<1;**

**64 rc = t<<1 | 1;**

**65 }**

**66 void add(int t, int ll, int rr, int l, int r, int h){**

**67 int lc,rc,mid;**

**68 if(rr < l || r < ll) return;**

**69 getch(t, lc, rc);**

**70 if(l <= ll && rr <= r){**

**71 cut[t] += h;**

**72 if(cut[t] >= 1){**

**73 sum[t] = num[rr] - num[ll-1];**

**74 }**

**75 else if(ll == rr) sum[t] = 0;**

**76 else sum[t] = sum[lc] + sum[rc];**

**77 return ;**

**78 }**

**79 mid = (ll + rr) >> 1;**

**80 add(lc, ll, mid, l, r, h);**

**81 add(rc, mid+1, rr, l, r, h);**

**82 if(cut[t] >= 1){**

**83 sum[t] = num[rr] - num[ll-1];**

**84 }**

**85 else sum[t] = sum[lc] + sum[rc];**

**86 }**

**87 int find(double yy){**

**88 int l,r,mid;**

**89 l = 1; r = m;**

**90 while(l <= r){**

**91 mid = (l + r) / 2;**

**92 if(sgn(y[mid] - yy) > 0) r = mid - 1;**

**93 else if(sgn(y[mid] - yy) < 0) l = mid + 1;**

**94 else return mid;**

**95 }**

**96 return -1;**

**97 }**

**98 void solve(){**

**99 int i,j,k,r,l,w;**

**100 memset(cut,0,sizeof(cut));**

**101 memset(sum,0,sizeof(sum));**

**102 memset(lazy,0,sizeof(lazy));**

**103 memset(ww,0,sizeof(ww));**

**104 double ans = 0;**

**105 for(i=0; i<(int)a.size()-1; i++){**

**106 l = find(a[i].l);**

**107 r = find(a[i].r) - 1;**

**108 if(l <= r) add(1, 1, m-1, l, r, a[i].t);**

**109 ans += sum[1] \* (a[i+1].x - a[i].x);**

**110 }**

**111 printf("Total explored area: %0.2f\n",ans);**

**112 }**

**113 int main(){**

**114 int ca = 1,ok=0;**

**115 while(scanf("%d",&n) != EOF && n){**

**116 if(ok == 1) printf("\n");**

**117 init();**

**118 printf("Test case #%d\n",ca++);**

**119 solve();**

**120 ok = 1;**

**121 }**

**122 return 0;**

**123 }**

### 大根堆

1 long dn=0; /\*大根堆\*/

2 void jia(long key)

3 { long i,j,k,m;

4 a[++dn]=key;

5 i=dn/2; j=dn;

6 while(i>=1)

7 { if(a[j]>a[i]) swap(&a[j],&a[i]);

8 else break;

9 j=i; i/=2;

10 }

11 }

12 void del()

13 { long i,j,k,m;

14 swap(&a[1],&a[dn]);

15 dn--;

16 i=1;

17 while(i<=dn/2)

18 { j=i\*2;

19 if(j+1<=dn&&a[j]<a[j+1]) j++;

20 if(a[i]<a[j]) swap(&a[i],&a[j]);

21 else break;

22 i=j;

23 }

24 }

25

### DXL

#### Suduke

1 const int maxn = 9 + 10;

2 int n = 9, m = 9, tn = 9;

3 class Graph {

4 public:

5 static const int maxn = 9 \* 9 \* 9 + 7;

6 static const int maxm = 1000 + 7;

7 static const int Max = maxn \* maxm + 10;

8 static const int sn = 9, sm = 9, stn = 9;

9 int adj[maxn][maxm], O[maxn]; //O[] is answer

10 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

#### Exact Cover

1 **class** Graph {

2 **public**:

3 **static** **const** **int** maxn = 1000 + 7;

4 **static** **const** **int** maxm = 1000 + 7;

5 **static** **const** **int** Max = maxn \* maxm + 10;

6 **int** adj[maxn][maxm];

7 **int** ans;

8 **void** init() {

9 n = m = 0;

10 memset(adj, 0, **sizeof**(adj));

11 }

12 **void** insert(**int** u, **int** v) {

13 n = max(n, u);

14 m = max(m, v);

15 adj[u][v] = 1;

16 }

17 **int** find\_ans() {

18 build\_dlx();

19 ans = -1;

20 **if** (dfs(0) ) {

21 **return** ans;

22 }

23 **return** -1;

24 }

25 **void** work(**int** n) {

26 **if**(n == -1) {

27 printf("NO\n");

28 **return** ;

29 }

30 printf("%d", n);

31 repf (i, 0, n - 1)

32 printf(" %d", O[i]);

33 printf("\n");

34 }

35 **private**:

36 **int** head;

37 **int** R[Max], L[Max], U[Max], D[Max], C[Max], H[Max];

38 **int** S[maxn], O[maxn];

39 **int** n, m, cnt, nm;

40

41 **void** add(**int** head, **int** tmp, **int** x) {

42 H[cnt] = head;

43 R[cnt] = tmp; L[cnt] = L[tmp];

44 L[tmp] = cnt; R[L[cnt]] = cnt;

45 U[cnt] = U[x]; D[cnt] = x;

46 D[U[x]] = cnt; U[x] = cnt;

47 C[cnt] = x; ++S[x];

48 ++cnt;

49 }

50 **void** build\_dlx() {

51 L[0] = R[0] = U[0] = D[0] = C[0] = H[0] = 0;

52 **for** (**int** i = 1; i <= m; i++) {

53 H[i] = 0;

54 L[i] = i - 1; R[i] = 0;

55 R[i - 1] = i; L[0] = i;

56 U[i] = D[i] = C[i] = i;

57 S[i] = 0;

58 }

59 cnt = m + 1;

60 **for** (**int** i = 1; i <= n; i++) {

61 **int** tmp = Max - 1;

62 L[tmp] = R[tmp] = U[tmp] = D[tmp] = C[tmp] = tmp;

63 **for** (**int** j = 1; j <= m; j++)

64 **if**(adj[i][j]) {

65 add(i, tmp, j);

66 }

67 L[R[tmp]] = L[tmp];

68 R[L[tmp]] = R[tmp];

69 }

70 }

71 **void** remove(**const** **int** &c) {

72 R[L[c]] = R[c];

73 L[R[c]] = L[c];

74 **for** (**int** i = D[c]; i != c; i = D[i]) {

75 **for** (**int** j = R[i]; j != i; j = R[j]) {

76 U[D[j]] = U[j];

77 D[U[j]] = D[j];

78 --S[C[j]];

79 }

80 }

81 }

82

83 **void** resume(**const** **int** &c) {

84 **for** (**int** i = D[c]; i != c; i = D[i]) {

85 **for** (**int** j = R[i]; j != i; j = R[j]) {

86 U[D[j]] = j;

87 D[U[j]] = j;

88 ++S[C[j]];

89 }

90 }

91 R[L[c]] = c;

92 L[R[c]] = c;

93 }

94

95 **bool** dfs(**const** **int** &k) {

96 **if** (R[0] == 0){

97 ans = k;

98 **return** true;

99 }

100 **int** s(maxint), c;

101 **for** (**int** i = R[0]; i != 0; i = R[i]) {

102 **if** (S[i] < s) {

103 c = i;

104 s = S[i];

105 }

106 }

107 remove(c);

108 **for** (**int** i = D[c]; i != c; i = D[i]) {

109 O[k] = H[i];

110 **for** (**int** j = R[i]; j != i; j = R[j]) remove(C[j]);

111 **if** (dfs(k + 1)) **return** true;

112 **for** (**int** j = L[i]; j != i; j = L[j]) resume(C[j]);

113 }

114 resume(c);

115 **return** false;

116 }

117 }G;

118

# 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

**1 判线段相交，求交点**

**2 bool jiaodian(point a,point b,point c,point d,point &e)**

**3 {**

**4 double d1 = (b-a) \* (c-a), d2 = (b-a) \* (d-a),**

**5 d3 = (d-c) \* (a-c), d4 = (d-c) \* (b-c);**

**6 if(sgn(d1)\*sgn(d2) > 0)**

**7 return false;**

**8 e = point( (c.x\*d2 - d.x\*d1) / (d2-d1) ,**

**9 (c.y\*d2 - d.y\*d1) / (d2-d1) );**

**10 return true;**

**11 }**

**12**

### 最近点对

**1 bool cmpy(const point &a, const point &b){**

**2 if( sgn(a.y - b.y) != 0) return a.y < b.y;**

**3 else return a.x < b.x;**

**4 }**

**5 bool cmpx(const point &a, const point &b){**

**6 if( sgn(a.x - b.x) != 0) return a.x < b.x;**

**7 else return a.y < b.y;**

**8 }**

**9 point tempt[maxn], a[maxn];**

**10 int n;**

**11 void get\_min(point \*a, int l, int r, double &d){**

**12 int n = r - l + 1;**

**13 if(n == 1) { return;}**

**14 if(n <= 3){**

**15 repf(i, l, r - 1){**

**16 d = min(d, (a[i] - a[(i + 1)]).len());**

**17 }**

**18 d = min(d, (a[r] - a[l]).len());**

**19 }**

**20 else{**

**21 double d1, d2, d3;**

**22 d1 = d2 = d3 = 1e100;**

**23 int mid = (l + r) >> 1;**

**24 get\_min(a, l, mid, d1);**

**25 get\_min(a, mid + 1, r, d2);**

**26 d = min(d1, d2);**

**27 int k = 0, num = 6;**

**28 repf (i, l, r)**

**29 if( fabs(a[i].x - a[mid].x) <= d)**

**30 tempt[k++] = a[i];**

**31 sort(tempt, tempt + k, cmpy);**

**32 rep (i, k)**

**33 for(int j = i + 1; j < k && tempt[j].y - tempt[i].y < d; j++){**

**34 d = min(d, (tempt[j] - tempt[i]).len());**

**35 }**

**36 }**

**37 }**

**38 int main(){**

**39 while(scanf("%d", &n) == 1 && n){**

**40 rep(i, n){**

**41 point p;**

**42 p.input();**

**43 a[i] = p;**

**44 }**

**45 sort(a, a + n, cmpx);**

**46 double ans = 1e100;**

**47 get\_min(a, 0, n - 1, ans);**

**48 printf("%.2f\n", ans / 2);**

**49 }**

**50 return 0;**

**51 }**

### 线段与线段的距离

1 double get\_dis(point a, point sb, point eb) {

2 return min( (a - sb).len(), (a - eb).len());

3 }

4 double dis(point a, point b, point c) {

5 double mul = ( (a - b) ^ (c - b) ) / (c - b).len();

6 point dir = (c - b).set();

7 point mid = dir \* mul + b;

8 if( sgn((mid - b) ^ (c - b) ) >= 0 && sgn((mid - c) ^ (b - c)) >= 0) {

9 return fabs((a - b) \* (c - b) / (c - b).len());

10 }

11 else return get\_dis(a, b, c);

12 }

13 double dis(int a, int b) { //线段tp[a]sp[a], tp[b]sp[b]

14 double res = min( dis(tp[a], tp[b], sp[b]), dis(sp[a], tp[b], sp[b]));

15 res = min(res, min(dis(tp[b], tp[a], sp[a]), dis(sp[b], tp[a], sp[a])));

16 return res;

17 }

18

### O(N^2)处理最少用几段弧完全覆盖一个圆

1 struct node {

2 double be, en; //开始的角度 与 结束的角度 （-pi ~ pi)

3 node (double \_be = 0, double \_en = 0) : be(\_be), en(\_en){

4 }

5 bool operator < (const node &b) const {

6 return sgn(be - b.be) < 0;

7 }

8 } a[maxn], b[maxn];

9

10 node change(node p, double ang) { //将角度转换成从ang度开始，需要转动多少度

11 double be = p.be, en = p.en;

12 be -= ang;

13 while(sgn(be) < 0) be += 2 \* pi;

14 en -= ang;

15 while(sgn(en) < 0) en += 2 \* pi;

16 if(sgn(en - be) < 0) en += 2 \* pi;

17 return node(be, en);

18 }

19

20

21 sort(a, a + n);

22 rep (i, n)

23 a[i + n] = a[i];

24 int ans = maxint;

25 rep (i, n) {

26 rep (j, n) {

27 b[j] = change(a[i + j], a[i].be);

28 }

29 int res = 0, k = 0;

30 double old = 0;

31 while(k < n && sgn(old - 2 \* pi) < 0) {

32 double next = old;

33 while(k < n && sgn(b[k].be - old) <= 0) {

34 if(sgn(b[k].en - next) > 0)

35 next = b[k].en;

36 k++;

37 }

38 if(sgn(next - old) == 0 ) k = n + 1;

39 res++;

40 old = next;

41

42 }

43 if(sgn(old - 2 \* pi) < 0) {

44 continue;

45 }

46 ans = min(ans, res);

47 }

48 if(ans == maxint) ans = -1;

49 printf("%d\n", ans);

50

### 半平面交

1 **struct** line {

2 point p, v;

3 **double** ang;

4 line() {}

5 line(point p, point v) : p(p), v(v) { ang = atan2(v.y, v.x); }

6 **bool** **operator** < (**const** line &l) **const** { **return** ang < l.ang; }

7 };

8

9 //点p在有向直线l的左边（线上不算）

10 **bool** onLeft(line l, point p) {

11 **return** sgn(l.v \* (p - l.p)) > 0;

12 }

13

14 //二直线交点，假设交点唯一存在

15 point getIntersection(line a, line b) {

16 point u = a.p - b.p;

17 **double** t = (b.v \* u) / (a.v \* b.v);

18 **return** a.p + a.v \* t;

19 }

20

21 point p[maxn];

22 line q[maxn];

23 **int** halfPlane(vector<line> l, vector<point> &poly) { //l:anti-clockwise

24 **int** n = sz(l);

25 sort(l.begin(), l.end());

26 **int** first, last;

27 q[first = last = 0] = l[0];

28 **for** (**int** i = 1; i < n; i++) {

29 **while** (first < last && !onLeft(l[i], p[last - 1])) last--;

30 **while** (first < last && !onLeft(l[i], p[first])) first++;

31 q[++last] = l[i];

32 **if** (sgn(q[last].v \* q[last - 1].v) == 0) {

33 last--;

34 **if** (onLeft(q[last], l[i].p)) q[last] = l[i];

35 }

36 **if** (first < last) p[last - 1] = getIntersection(q[last - 1], q[last]);

37 }

38 **while** (first < last && !onLeft(q[first], p[last - 1])) last--;

39

40 poly.clear();

41 **if** (last - first <= 1) **return** 0;

42 p[last] = getIntersection(q[last], q[first]);

43 **int** m = 0;

44 **for** (**int** i = first; i <= last; i++) {

45 poly.push\_back(p[i]);

46 m++;

47 }

48 **return** m;

49 }

### 判断点是否在多边形内(old)

1 **double** trim(**double** d, **double** l = 1.0) {

2 **return** d > l ? l : (d < -l ? -l : d);

3 }

4 **int** get\_position(**const** point& p, **const** point\* pol, **int** n) {

5 **double** ang = 0;

6 **for** (**int** i = 0; i < n; ++i) {

7 **if** (pol[i] == p) **return** 0; //在点上

8 point p1 = pol[i] - p, p2 = pol[(i + 1) % n] - p;

9 **double** c = (p1 ^ p2) / (p1.len() \* p2.len());

10 c = trim(c);

11 ang += sgn(p1 \* p2) \* acos(c);

12 }

13 ang = abs(ang);

14 **return** ang < 0.5 \* pi ? -1 : (ang < 1.5 \* pi ? 0 : 1);

15 }

### 判断点是否在多边形内(new)

1 **bool** onSegment(**const** point &p, **const** point &s, **const** point &e) {

2 **if** (p == s || p == e) **return** true;

3 **if** (sgn((p - s) \* (e - s)) == 0 && sgn((s - p) ^ (e - p)) <= 0)

4 **return** true;

5 **return** false;

6 }

7 **int** get\_position(**const** point &p, point \*pol, **int** n) {

8 **int** wn = 0;

9 **for** (**int** i = 0; i < n; i++) {

10 **if** (onSegment(p, pol[i], pol[(i + 1) % n])) **return** 0; //on the segment

11 **int** k = sgn((pol[(i + 1) % n] - pol[i]) \* (p - pol[i]));

12 **int** d1 = sgn(pol[i].y - p.y);

13 **int** d2 = sgn(pol[(i + 1) % n].y - p.y);

14 **if** (k > 0 && d1 <= 0 && d2 > 0) wn++;

15 **if** (k < 0 && d2 <= 0 && d1 > 0) wn--;

16 }

17 **if** (wn != 0) **return** 1; //inner

18 **else** **return** -1; //outter

19 }

### 异面线段距离

1 //返回直线距离的平方,返回一个分数。 node是分数类

2 node gao3(point p, point a, point b) { //点到线段的距离

3 **if** (p == a || p == b) **return** node(0);

4 **if** (a == b) **return** node((p - a) ^ (p - a));

5 point v1 = b - a, v2 = p - a, v3 = p - b;

6 **if** ((v1 ^ v2) < 0) {

7 **return** node(v2 ^ v2);

8 }

9 **else** **if** ((v1 ^ v3) > 0) {

10 **return** node(v3 ^ v3);

11 }

12 **else** {

13 **return** node((v1 \* v2) ^ (v1 \* v2)) / node(v1 ^ v1);

14 }

15 }

16 node gao2(point a, point b, point c, point d) { //当线段之间没有垂线的距离

17 node res = gao3(c, a, b);

18 res = min(res, gao3(d, a, b));

19 res = min(res, gao3(a, c, d));

20 res = min(res, gao3(b, c, d));

21 **return** res;

22 }

23

24 **bool** ok(node x) {

25 **if** (x.zi \* x.mu < 0) **return** false;

26 **if** (x.zi < 0)

27 x.zi \*= -1, x.mu \*= -1;

28 **return** x.zi <= x.mu;

29 }

30

31 node gao(point a, point b, point l, point r) { //线段与线段之间的距离

32 lint x0, y0, z0, x1, y1, z1, x2, y2, z2, x3, y3, z3;

33 x0 = a.x, y0 = a.y, z0 = a.z;

34 x1 = b.x, y1 = b.y, z1 = b.z;

35 x2 = l.x, y2 = l.y, z2 = l.z;

36 x3 = r.x, y3 = r.y, z3 = r.z;

37 lint a1 = x1 - x0, a2 = x2 - x3, a3 = x0 - x2,

38 a4 = y1 - y0, a5 = y2 - y3, a6 = y0 - y2,

39 a7 = z1 - z0, a8 = z2 - z3, a9 = z0 - z2;

40 lint A = a1 \* a1 + a4 \* a4 + a7 \* a7,

41 B = a2 \* a2 + a5 \* a5 + a8 \* a8,

42 C = 2 \* (a1 \* a2 + a4 \* a5 + a7 \* a8),

43 D = 2 \* (a1 \* a3 + a4 \* a6 + a7 \* a9),

44 E = 2 \* (a2 \* a3 + a5 \* a6 + a8 \* a9),

45 F = a3 \* a3 + a6 \* a6 + a9 \* a9;

46 **if** ((a1 \* a5 == a2 \* a4 && a1 \* a8 == a2 \* a7 && a4 \* a8 == a5 \* a7))

47 **return** gao2(a, b, l, r);

48

49 lint Y = C \* D - 2 \* A \* E,

50 X = C \* E - 2 \* B \* D;

51 **if** (!ok(node(Y, 4 \* A \* B - C \* C)) || !ok(node(X, 4 \* A \* B - C \* C)))

52 **return** gao2(a, b, l, r);

53 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);

54 **return** node(S, (4 \* A \* B - C \* C) \* (4 \* A \* B - C \* C));

55 }

56

### 圆的面积并

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

2 **const** **int** zy[] = {1, 0, -1, 0};

3

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

5 **void** get\_min(**double**& x, **double** y) { x = min(x, y); }

6 **void** get\_max(**double**& x, **double** y) { x = max(x, y); }

7 **struct** P {

8 **double** x, y;

9 P() {}

10 P(**double** \_x, **double** \_y): x(\_x), y(\_y) {}

11 P **operator** + (**const** P &a) **const** { **return** P(x + a.x, y + a.y); }

12 P **operator** - (**const** P &a) **const** { **return** P(x - a.x, y - a.y); }

13 P **operator** \* (**const** **double** &m) **const** { **return** P(x \* m, y \* m); }

14 P **operator** / (**const** **double** &m) **const** { **return** P(x / m, y / m); }

15 P set(**const** **double** &m) **const** {

16 **double** len = length();

17 **return** P(x \* m / len, y \* m / len);

18 }

19 P turn(**const** **double** &m) **const** {

20 **double** c = cos(m), s = sin(m);

21 **return** P(x \* c - y \* s, x \* s + y \* c);

22 }

23 **bool** **operator** == (**const** P &p) **const** { **return** sgn(x - p.x) == 0 && sgn(y - p.y) == 0;

24 }

25 **double** length() **const** {

26 **return** sqrt(x \* x + y \* y);

27 }

28 **double** dist(**const** P &a) **const** {

29 **return** sqrt(SQR(x - a.x) + SQR(y - a.y));

30 }

31 **double** cross(**const** P &a, **const** P &b) **const** { **return** (a.x - x) \* (b.y - y) - (a.y - y) \* (b.x - x); }

32 **double** cross(**const** P &a) **const** { **return** x \* a.y - y \* a.x; }

33 **double** dot(**const** P &a, **const** P &b) { **return** (a.x - x) \* (b.x - x) + (a.y - y) \* (b.y - y); }

34 **void** input() { scanf("%lf%lf", &x, &y); }

35 **void** output() **const** { printf("(%lf, %lf)\n", x, y); }

36 P trunc(**double** l) **const** {

37 **double** r = l / length();

38 **return** P(x \* r, y \* r);

39 }

40 P turn\_left() **const** { **return** P(-y, x); }

41 P rotate\_left(**double** ang) **const** {

42 **double** c = cos(ang), s = sin(ang);

43 **return** P(x \* c - y \* s, y \* c + x \* s);

44 }

45 P turn\_right() **const** { **return** P(y, -x); }

46 P rotate\_right(**double** ang) **const** {

47 **double** c = cos(ang), s = sin(ang);

48 **return** P(x \* c + y \* s, y \* c - x \* s);

49 }

50 };

51

52 **double** dist2(**const** P &a, **const** P &b) {

53 **return** SQR(a.x - b.x) + SQR(a.y - b.y);

54 }

55 **double** dist(**const** P &a, **const** P &b) {

56 **return** sqrt(SQR(a.x - b.x) + SQR(a.y - b.y));

57 }

58 **double** cross(**const** P &a, **const** P &b, **const** P &c) {

59 **return** (b.x - a.x) \* (c.y - a.y) - (b.y - a.y) \* (c.x - a.x);

60 }

61 **double** dmul(**const** P &a, **const** P &b, **const** P &c) {

62 **return** (b.x - a.x) \* (c.x - a.x) + (b.y - a.y) \* (c.y - a.y);

63 }

64

65 **int** NEXT(**int** x, **int** n) {

66 **return** x % n;

67 }

68

69 **struct** C {

70 P mid;

71 **double** r;

72 C(**const** P &\_mid, **const** **double** &\_r)

73 :mid(\_mid), r(\_r) {}

74 C() {}

75 **bool** **operator** == (**const** C &a) **const** {

76 **return** mid == a.mid && sgn(r - a.r) == 0;

77 }

78 **bool** in(**const** C &a) **const** {

79 **return** sgn(r + dist(mid, a.mid) - a.r) <= 0;

80 }

81 **const** C &input() {

82 mid.input();

83 scanf("%lf", &r);

84 **return** \***this**;

85 }

86 **const** C &output() **const** {

87 printf("P: %.12lf %.12lf R: %.12lf\n", mid.x, mid.y, r);

88 }

89 };

90 **double** cal\_angle(**const** C &c, **const** P &a, **const** P &b) {

91 **double** k = dmul(c.mid, a, b) / SQR(c.r);

92 get\_min(k, 1.0);

93 get\_max(k, -1.0);

94 **return** acos(k);

95 }

96 **double** cal\_area(**const** C &c, **const** P &a, **const** P &b) {

97 **return** SQR(c.r) \* cal\_angle(c, a, b) / 2 - cross(c.mid, a, b) / 2;

98 }

99 **struct** cmp {

100 P mid;

101 cmp(**const** P &\_mid)

102 :mid(\_mid) {}

103 **bool** **operator** () (**const** P &a, **const** P &b) {

104 **return** atan2(a.y - mid.y, a.x - mid.x) < atan2(b.y - mid.y, b.x - mid.x);

105 }

106 };

107 **bool** circles\_intersection(**const** C &a, **const** C &b, P &c1, P &c2) {

108 **double** dd = dist(a.mid, b.mid);

109 **if** (sgn(dd - (a.r + b.r)) >= 0) {

110 **return** false;

111 }

112 **double** l = (dd + (SQR(a.r) - SQR(b.r)) / dd) / 2;

113 **double** h = sqrt(SQR(a.r) - SQR(l));

114 c1 = a.mid + (b.mid - a.mid).trunc(l) + (b.mid - a.mid).turn\_left().trunc(h);

115 c2 = a.mid + (b.mid - a.mid).trunc(l) + (b.mid - a.mid).turn\_right().trunc(h);

116 **return** true;

117 }

118 **bool** cover(**const** C &c, **const** P &a, **const** P &b, **const** vector<C> &cir) {

119 P p = c.mid + ((a + b) / 2 - c.mid).trunc(c.r);

120 **for** (vector<C>::const\_iterator it = cir.begin(); it != cir.end(); ++it) {

121 **if** (sgn(dist2(p, it->mid) - SQR(it->r)) < 0) {

122 **return** true;

123 }

124 }

125 **return** false;

126 }

127 **double** cal\_area(**const** vector<C> &in) {

128 vector<C> cir;

129 **for** (**int** i = 0; i < SZ(in); ++i) {

130 **if** (sgn(in[i].r) == 0) {

131 **continue**;

132 }

133 **bool** flag = false;

134 **for** (**int** j = i + 1; j < SZ(in); ++j) {

135 **if** (in[i] == in[j]) {

136 flag = true;

137 **break**;

138 }

139 }

140 **if** (flag) {

141 **continue**;

142 }

143 **for** (**int** j = 0; j < SZ(in); ++j) {

144 **if** (!(in[i] == in[j]) && in[i].in(in[j])) {

145 flag = true;

146 **break**;

147 }

148 }

149 **if** (flag) {

150 **continue**;

151 }

152 cir.push\_back(in[i]);

153 }

154 vector<vector<P> > point\_on\_circle(SZ(cir));

155 **for** (**int** i = 0; i < SZ(cir); ++i) {

156 **for** (**int** z = 0; z < 4; ++z) {

157 point\_on\_circle[i].push\_back(cir[i].mid + P(zx[z], zy[z]).trunc(cir[i].r));

158 }

159 }

160 **for** (**int** i = 0; i < SZ(cir); ++i) {

161 **for** (**int** j = i + 1; j < SZ(cir); ++j) {

162 P a, b;

163 **if** (circles\_intersection(cir[i], cir[j], a, b)) {

164 point\_on\_circle[i].push\_back(a);

165 point\_on\_circle[i].push\_back(b);

166 point\_on\_circle[j].push\_back(a);

167 point\_on\_circle[j].push\_back(b);

168 }

169 }

170 }

171 **for** (**int** i = 0; i < SZ(cir); ++i) {

172 sort(point\_on\_circle[i].begin(), point\_on\_circle[i].end(), cmp(cir[i].mid));

173 point\_on\_circle[i].erase(unique(point\_on\_circle[i].begin(), point\_on\_circle[i].end()), point\_on\_circle[i].end());

174 }

175 **double** ans = 0;

176 **for** (**int** i = 0; i < SZ(cir); ++i) {

177 **for** (**int** j = 0; j < SZ(point\_on\_circle[i]); ++j) {

178 **const** P &a = point\_on\_circle[i][j];

179 **const** P &b = point\_on\_circle[i][NEXT(j + 1, SZ(point\_on\_circle[i]))];

180 **if** (!cover(cir[i], a, b, cir)) {

181 ans += cross(P(0, 0), a, b) / 2;

182 ans += cal\_area(cir[i], a, b);

183 }

184 }

185 }

186 **return** ans;

187 }

188

# Math

### miller\_rabin\_and\_Pollard\_rho

1 //miller\_rabin大数检测+Pollard P素因子分解

2 //输入 a<2^63

3 //加大MAX 可以保证分解的成功率

4 #include <stdlib.h>

5 #include <stdio.h>

6

7 **typedef** **unsigned** \_\_int64 u64;

8

9 #define MAX 100

10 #define MAXN 30

11

12 u64 len, dig, limit;

13 u64 mod(u64 a, u64 b, u64 n)

14 {

15 **if**(!a) **return** 0;

16 **else** **return** (((a & dig) \* b) % n + (mod(a >> len, b, n) << len) % n) % n;

17 }

18

19 u64 by(u64 a, u64 b, u64 n)

20 {

21 u64 p;

22 p = 8, len = 61;

23 **while**(p < n)

24 {

25 p <<= 4;

26 len -= 4;

27 }

28 dig = ((limit / p) << 1) - 1; //动态划分段

29 **return** mod(a, b, n);

30 }

31

32 u64 random(**void**)

33 {

34 u64 a;

35 a = rand();

36 a \*= rand();

37 a \*= rand();

38 a \*= rand();

39 **return** a;

40 }

41

42 //Miller\_Rabin

43 u64 square\_multiply(u64 x, u64 c, u64 n)

44 {

45 u64 z = 1;

46 **while**(c)

47 {

48 **if**(c % 2 == 1) z = by(z, x, n);

49 x = by(x,x,n);

50 c = (c >> 1);

51 }

52 **return** z;

53 }

54

55 **bool** Miller\_Rabin(u64 n)

56 {

57 **if**(n < 2) **return** false;

58 **if**(n == 2) **return** true;

59 **if**(!(n & 1)) **return** false;

60 u64 k = 0, i, j, m, a;

61 m = n - 1;

62 **while**(m % 2 == 0) m = (m >> 1), k++;

63 **for**(i = 0; i < MAX; i++)

64 {

65 a = square\_multiply(random() % (n - 1) + 1, m, n);//平方乘

66 **if**(a == 1) **continue**;

67 **for**(j = 0; j < k; j++)

68 {

69 **if**(a == n - 1) **break**;

70 a = by(a, a, n);

71 }

72 **if**(j < k) **continue**;

73 **return** false ;

74 }

75 **return** true;

76 }

77

78 //Pollard p,只找出一个因子。

79 u64 gcd(u64 a, u64 b)

80 {

81 **return** b == 0 ? a : gcd(b, a % b);

82 }

83

84 //用公式f(x) = x^2 + 1检验碰撞。

85 u64 f(u64 x, u64 n)

86 {

87 **return** (by(x, x, n) + 1) % n;

88 }

89

90 //分解不到，return 0

91 u64 Pollard(u64 n)

92 {

93 **if**(n <= 2) **return** 0;

94 **if**(!(n & 1)) **return** 2; //必不可少

95 u64 i, p, x, xx;

96 **for**(i = 1; i < MAX; i++)

97 {

98 x = random() % n; //或者直接用 x = i

99 xx = f(x, n);

100 p = gcd((xx + n - x) % n , n);

101 **while**(p == 1)

102 {

103 x = f(x, n);

104 xx = f(f(xx, n), n);

105 p = gcd((xx + n - x) % n, n) % n;

106 }

107 **if**(p)**return** p;

108 }

109 **return** 0;

110 }

111

112 /////////////////////////////////////////////////////////

113 u64 factor[MAXN], m;

114 /////////////////////////////////////////////////////

115 //分解质数因子

116 u64 prime(u64 a)

117 {

118 **if**(Miller\_Rabin(a) || a == 0) **return** 0;

119 u64 t = Pollard(a), p;

120 **if**(p = prime(t)) **return** p;

121 **else** **return** t;

122 }

123

124

125 //622057148 155514287 会跪

126 **int** main(**void**)

127 {

128 u64 l, a, t;

129 limit = 1;

130 limit = limit << 63; //动态化分段使用

131 **while**(scanf("%I64u", &a) != EOF)

132 {

133 m = 0;

134 **while**(a > 1)

135 {

136 **if**(Miller\_Rabin(a)) **break**;

137 t = prime(a);

138 **if** (t == 0) **break**;

139 factor[m++] = t;

140 a /= t;

141 }

142 **if**(a > 0) factor[m++] = a;

143 **for**(l = 0; l < m; l++)

144 printf("%I64u\n", factor[l]);

145 }

146 **return** 0;

147 }

### get\_prime

**1 int prime[664588], cnt = 0;**

**2 void makePrime() {**

**3 for (int i = 2; i < maxn; ++i) {**

**4 if (!f[i]) {**

**5 prime[cnt++] = i;**

**6 }**

**7 for (int j = 0; (int64)i \* prime[j] < maxn; ++j) {**

**8 f[i \* prime[j]] = true;**

**9 if (i % prime[j] == 0) {**

**10 break;**

**11 }**

**12 }**

**13 }**

**14 }**

### Matrix

1 **struct** matrix {

2 **double** ar[maxa][maxa];

3 **int** n, m ; // n \* m; 0 ~ n - 1, 0 ~ m - 1;

4 matrix() {

5 n = 4; //n

6 m = 4; //m

7 memset(ar, 0, **sizeof**(ar));

8 }

9 **void** clear() {

10 rep (i, n)

11 rep (j, m)

12 ar[i][j] = 0;

13 }

14 **void** set\_one() { //记得先给N,M赋值

15 rep (i, n)

16 rep (j, m)

17 ar[i][j] = 0;

18 rep (i, min(n, m))

19 ar[i][i] = 1;

20 }

21 **void** output() {

22 printf("%d %d\n", n, m);

23 rep(i, n) {

24 rep(j, m)

25 printf("%.3f ", ar[i][j]);

26 printf("\n");

27 }

28 printf("\n");

29 }

30 };

31 matrix **operator** \* (**const** matrix &a, **const** matrix &b) {

32 matrix c;

33 **if**(a.m != b.n) printf("a.m != b.n\n");

34 c.clear();

35 c.n = a.n;

36 c.m = b.m;

37 rep (i, a.n)

38 rep (j, b.m)

39 rep (k, a.m) {

40 c.ar[i][j] += a.ar[i][k] \* b.ar[k][j]; //mod

41 }

42 **return** c;

43 }

44

#### 二&三维旋转

**计算机生成了可选文字: 平移：
饮ty仪**

**计算机生成了可选文字: 拉伸：
b**

**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)**

**计算机生成了可选文字: 旋转：
州
、．l...lee了
0001
r"，承，一。
．人州1一。＋.tS
！月：才，(，一0一凡’
戈0
通洲。（1一O一才声
c＋可（l一C)
人才，(1一0＋汉：S
0
月：.':(1一O＋人s
凡人（1一O一月：S
c＋对（l一O
0**

**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++)**

**13 if(map[i][k] && i != k)**

**14 { for(j=k;j<=30;j++)**

**15 map[i][j] ^= map[k][j];**

**16 }**

**17 }**

**18**

**19 for(k=29;k>=0;k--)**

**20 { ans[k] = map[k][30];**

**21 for(i=0;i<=30 && !map[k][i];i++) ;**

**22 if(i == 30) return 0;**

**23 for(i=k+1;i<30;i++)**

**24 ans[k] ^= map[k][i] \* ans[i];**

**25 //ans[k] ^= map[k][k];**

**26 }**

**27 }**

### GCD&扩展GCD

**1 long long Gcd(long long a,long long b)**

**2 {**

**3 for(long long t=a%b;t; a=b,b=t,t=a%b); return b;**

**4 }**

**5 long long ExpandGcd(long long a, long long b, long long &d, long long &x, long long &y)**

**6 {**

**7 if( b ) { ExpandGcd( b, a%b , d, y, x); y -= a/b \* x; }**

**8 else { d = a; x = 1; y = 0; }**

**9 }**

**10**

### 辛普森积分

1 double f(double x) {

2 return x;

3 }

4 double sps(double l, double r){

5 return (f(l) + f(r) + f((l+r)/2)\*4)/6 \* (r - l);

6 }

7 double sps2(double l, double r, int dep){

8 //printf("%lf %lf %d\n", l, r, dep);

9 double cur = sps(l, r), mid = (l + r)/2;

10 double y = sps(l, mid) + sps(mid, r);

11 if(sgn(cur-y) == 0 && dep > 9) return cur;

12 return sps2(l, mid, dep+1) + sps2(mid, r, dep+1);

13 }

1 **void** gcd(lint a, lint b, lint& d, lint& x, lint& y) {

2 **if** (!b) { d = a; x = 1; y = 0; }

3 **else** {gcd(b, a % b, d, y, x); y -= x \* (a / b); }

4 }

5

6 lint inv(lint a, lint n) {

7 lint d, x, y;

8 gcd(a, n, d, x, y);

9 **return** d == 1 ? (x + n) % n : -1;

10 }

### Mobius

1 lint v[maxn];

2 lint mob[maxP];

3 **void** getMobius() {

4 memset(mob, 0, **sizeof**(mob));

5 memset(v, 0, **sizeof**(v));

6 mob[1] = 1;

7 **for** (lint i = 2; i < maxn; i++) {

8 **if** (v[i] == 0) {

9 **for** (lint j = i + i; j < maxn; j += i) {

10 v[j] = 1;

11 mob[j] = mob[j / i] \* -1;

12 }

13 mob[i] = -1;

14 }

15 }

16 }

### logMod （a ^ x = b (mod n) ）(含逆元)

1 **void** gcd(lint a, lint b, lint &d, lint &x, lint &y) {

2 **if** (!b) {d = a; x = 1; y = 0;}

3 **else** { gcd(b, a % b, d, y, x); y -= x \* (a / b);}

4 }

5

6 lint inv(lint a, lint n) {

7 lint d, x, y;

8 gcd(a, n, d, x, y);

9 **return** d == 1 ? (x + n) % n : -1;

10 }

11

12 lint mulMod(lint a, lint b, lint m = mod) { // a \* b % m;

13 **return** a \* b % m;

14 }

15

16 lint powMod(lint a, lint b, lint m = mod) { // a ^ b % m;

17 lint res = 1;

18 **while** (b != 0) {

19 **if** (b & 1) {

20 res = (res \* a) % m;

21 }

22 a = (a \* a) % m;

23 b >>= 1;

24 }

25 **return** res;

26 }

27

28

29 lint logMod(lint a, lint b, lint n = mod) { //a ^ x = b (mod n)

30 lint m, v, e = 1, i;

31 m = (**int**)sqrt(n + 0.5);

32 v = inv(powMod(a, m, n), n);

33 map <lint, lint > x;

34 x.clear();

35 x[1] = 0;

36 **for** (lint i = 1; i < m; i++) {

37 e = mulMod(e, a, n);

38 **if** (!x.count(e)) x[e] = i;

39 }

40 **for** (lint i = 0; i < m; i++) {

41 **if** (x.count(b)) **return** i \* m + x[b];

42 b = mulMod(b, v, n);

43 }

44 **return** -1;

45 }

# Java

### MAP

1 **public** **static** Map<BigInteger, BigInteger> dic;

2 **public** **static** BigInteger gao(BigInteger n) {

3 **if** (dic.containsKey(n) == false) {

4 dic.put(n, res3);

5 }

6 **return** dic.get(n);

7 }

8 **public** **static** **void** main(String[] args) {

9 dic = **new** HashMap();

10 dic.clear();

11 gao(x);

12 }

### 分数操作

1 import java.io.\*;

2 import java.math.\*;

3 import java.util.\*;

4

5 **public** **class** Main {

6 **public** **final** **static** **int** maxn = 50 + 10;

7 **public** **final** **static** **int** lim\_m = 50;

8 **public** **final** **static** BigInteger ZERO = **new** BigInteger("0");

9 **public** **final** **static** BigInteger ONE = **new** BigInteger("1");

10

11 **public** **static** **void** updata(**int** i, **int** j, **int** r, **int** w, BigInteger[][] mu, BigInteger[][] zi, BigInteger scope) {

12 BigInteger nmu = mu[r][w].multiply(scope);

13 BigInteger new\_mu = mu[i][j].multiply(nmu);

14 BigInteger new\_zi = mu[i][j].multiply(zi[r][w]);

15 new\_zi = new\_zi.add( nmu.multiply(zi[i][j]) );

16

17 //BigInteger h = new\_mu.gcd(new\_zi);

18 mu[i][j] = new\_mu;

19 zi[i][j] = new\_zi.divide(h);

20 }

21

22 **public** **static** **void** main(String[] args) {

23 BigInteger[][] mu = **new** BigInteger[maxn][maxn], zi = **new** BigInteger[maxn][maxn];

24 BigInteger scope = ZERO;

25 **int**[] x = **new** **int**[maxn];

26 **int** n, m, a, b;

27 Scanner cin = **new** Scanner(System.in);

28 **while** (cin.hasNextInt()) {

29 n = cin.nextInt();

30 m = cin.nextInt();

31 a = cin.nextInt();

32 b = cin.nextInt();

33 **for** (**int** i = 0; i < n; i++)

34 x[i] = cin.nextInt();

35

36 **for** (**int** i = 0; i <= n + 1; i++)

37 **for** (**int** j = 0; j <= lim\_m; j++)

38 mu[i][j] = ONE;

39 **for** (**int** i = 0; i <= n + 1; i++)

40 **for** (**int** j = 0; j <= lim\_m; j++)

41 zi[i][j] = ZERO;

42 zi[0][0] = ONE;

43 scope = scope.valueOf(b - a + 1);

44

45 **for** (**int** i = 0; i < n; i++) {

46 **for** (**int** j = a; j <= b; j++) {

47 **int** dis = Math.abs(x[i] - j);

48 **for** (**int** k = 0; k <= lim\_m; k++)

49 **if** (k - dis >= 0) {

50 updata(i + 1, k, i, k - dis, mu, zi, scope);

51 }

52 }

53 }

54

55 BigInteger ans\_mu = ONE, ans\_zi = ZERO;

56 **for** (**int** i = 0; i <= m; i++) {

57 ans\_zi = ans\_zi.multiply(mu[n][i]);

58 ans\_zi = ans\_zi.add(zi[n][i].multiply(ans\_mu));

59 ans\_mu = ans\_mu.multiply(mu[n][i]);

60 BigInteger h = ans\_zi.gcd(ans\_mu);

61 ans\_zi = ans\_zi.divide(h);

62 ans\_mu = ans\_mu.divide(h);

63 }

64

65 System.out.println(ans\_zi + "/" + ans\_mu);

66 }

67 }

68 }

# Others

### O(n)求回文串

**1 void getff()**

**2 {**

**3 long i,j,k,r,w,id,am,mx;**

**4 long p;**

**5 memset(s,0,sizeof(s));**

**6 memset(ff,0,sizeof(ff));**

**7 n = strlen(b);**

**8 s[0] = '#';**

**9 for(i=1;i<=2\*n;i++)**

**10 if(i%2 == 1) s[i] = b[i/2];**

**11 else s[i] = '#';**

**12 m = 2\*n; w = j = id = am= mx = 0;**

**13 p = 1;**

**14 while(p < m)**

**15 { if(mx > p) { ff[p] = min( ff[ id-(p-id) ] , ff[id] - (p-id));}**

**16 else ff[p] = 1;**

**17**

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

**19**

**20 if(ff[p] + p > mx)**

**21 { mx = ff[p] + p;**

**22 id = p;**

**23 }**

**24**

**25 p++;**

**26 }**

**27 for(i=1;i<=m;i++) ff[i]--;**

**28 }**

### KMP

1 /\*==================================================\*

2 | KMP 匹配算法O(M+N)

3 | CALL: res=kmp(str, pat); 原串为str; 模式为pat(长为P);

4 \\*==================================================\*/

5 int fail[P];

6 int kmp(char\* str, char\* pat){

7 int i, j, k;

8 memset(fail, -1, sizeof (fail));

9 for (i = 1; pat[i]; ++i) {

10 for (k=fail[i-1]; k>=0 && pat[i]!=pat[k+1];

11 k=fail[k]);

12 if (pat[k + 1] == pat[i]) fail[i] = k + 1;

13 }

14 i = j = 0;

15 while ( str[i] && pat[j] ){ // By Fandywang

16 if ( pat[j] == str[i] ) ++i, ++j;

17 else if (j == 0)++i;//第一个字符匹配失败，从str 下个字符开始

18 else j = fail[j-1]+1; }

19 if( pat[j] ) return -1;

20 else return i-j;

21 }

22

### Booth(int64乘int64余int64)

1 **inline** **long** **long** mul(**long** **long** lhs, **long** **long** rhs) {

2 **long** **long** lhs2 = lhs % 100000;

3 **long** **long** rhs2 = rhs % 100000;

4 **return** ((lhs / 100000 \* rhs2 + rhs / 100000 \* lhs2) \* 100000 + lhs2 \* rhs2) % MOD;

5 }

6

### 读入优化

1 **int** scanf(**int** &num)

2 {

3 **char** in;

4 **while**((in=getchar())!=EOF && (in>'9' || in<'0'));

5 **if**(in==EOF) **return** 0;

6 num=in-'0';

7 **while**(in=getchar(),in>='0' && in<='9') num\*=10,num+=in-'0';

8 **return** 1;

9 }

10

11 **int** scanf(**int** &num) { //负数

12 **char** in;

13 **int** op = 1;

14 **while** ((in = getchar()) != EOF && !(('0' <= in && in <= '9') || in == '-'));

15 **if** (in == EOF) **return** 0;

16 **if** (in == '-') {

17 op = -1;

18 in = getchar();

19 }

20 num = in - '0';

21 **while** (in = getchar(), in >= '0' && in <= '9') num \*= 10, num += in - '0';

22 num \*= op;

23 **return** 1;

24 }

### 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

**1 source $VIMRUNTIME/mswin.vim  
 2 behave mswin  
 3 imap <cr> <cr><left><right>  
 4 imap <c-]> {<cr>}<c-o>O<left><right>  
 5 imap <c-d> <c-o>dd  
 6 map <f6> =a{  
 7 map <c-t> :tabnew<cr>  
 8 syn on  
 9 colo desert  
10 set gfn=Courier\ 10\ Pitch\ 12  
11 set ru nu et sta nowrap ar acd ww=<,>,[,] sw=4 ts=4 cin noswf  
12   
13 map <f10> :call CR2()<cr><space>  
14 func CR2()  
15 exec "update"  
16 exec "!xterm -fn 10\*20 -e \"g++ %<.cpp -Wall -o %< && time ./%< ; read -n 1\""  
17 endfunc  
18 map <f9> :call CR()<cr><space>  
19 func CR()  
20 exec "update"  
21 exec "!xterm -fn 10\*20 -e \"g++ %<.cpp -Wall -o %< && time ./%< < %<.in ; read -n 1\""  
22 endfunc  
23   
24 map<f4> :call AddComment()<cr>  
25 func AddComment()  
26     if (getline('.')[0] == '/')  
27         normal ^xx  
28     else  
29         normal 0i//  
30     endif  
31 endfunc**