|  |
| --- |
|  |

|  |
| --- |
| ACM/ICPC at Wuhan University |
| Xioumu STL(code) |
| Created by xioumu v1.00 |

目录

[目录 2](#_Toc357174010)

[Graph 3](#_Toc357174011)

[Maxflow 3](#_Toc357174012)

[MinCostMaxFlow 3](#_Toc357174013)

[2-set 4](#_Toc357174014)

[N\*log(n) Dijkstra 5](#_Toc357174015)

[双强连通分量 6](#_Toc357174016)

[DataStructure 7](#_Toc357174017)

[数状数组 7](#_Toc357174018)

[RMQ 7](#_Toc357174019)

[平衡树 7](#_Toc357174020)

[线段树-扫描线矩形面积并 8](#_Toc357174021)

[大根堆 10](#_Toc357174022)

[DXL 10](#_Toc357174023)

[Aho-Corasick 12](#_Toc357174024)

[Computational Geometry 13](#_Toc357174025)

[凸包 13](#_Toc357174026)

[线段相交 13](#_Toc357174027)

[最近点对 13](#_Toc357174028)

[线段与线段的距离（线段与点的距离） 14](#_Toc357174029)

[O(N^2)处理最少用几段弧完全覆盖一个圆 14](#_Toc357174030)

[判断点是否在多边形内 15](#_Toc357174031)

[数论 15](#_Toc357174032)

[miller\_rabin\_and\_Pollard\_rho 15](#_Toc357174033)

[get\_prime 17](#_Toc357174034)

[Matrix 17](#_Toc357174035)

[Gauss 18](#_Toc357174036)

[GCD&扩展GCD 18](#_Toc357174037)

[辛普森积分 19](#_Toc357174038)

[欧拉函数 19](#_Toc357174039)

[Mobius反演 19](#_Toc357174040)

[String 19](#_Toc357174041)

[后缀树 19](#_Toc357174042)

[O(n)求回文串 20](#_Toc357174043)

[KMP 21](#_Toc357174044)

[Others 21](#_Toc357174045)

[读入优化 21](#_Toc357174046)

[布斯(Booth)乘法(int64乘int64余int64) 21](#_Toc357174047)

[乱七八糟 21](#_Toc357174048)

[JAVA 23](#_Toc357174049)

[Biginteger 23](#_Toc357174050)

[Vimrc 25](#_Toc357174051)

# Graph

### Maxflow

 1 struct Graph {  
 2     struct Adj {  
 3         int v, c, b;  
 4         Adj(int \_v, int \_c, int \_b):  
 5             v(\_v), c(\_c), b(\_b) {}  
 6         Adj(){}  
 7     };  
 8     int n, S, T, h[maxn], cnt[maxn];  
 9     vector<Adj> adj[maxn];  
10     void clear() {  
11         for (int i = 0; i < n; ++i) {  
12             adj[i].clear();  
13         }  
14         n = 0;  
15     }  
16     void insert(int u, int v, int c, int d = 0) {  
17         get\_max(n, max(u, v) + 1);  
18         adj[u].push\_back(Adj(v, c, adj[v].size()));  
19         adj[v].push\_back(Adj(u, c \* d, adj[u].size() - 1));  
20     }  
21     int maxflow(int \_S, int \_T) {  
22         S = \_S, T = \_T;  
23         fill(h, h + n, 0);  
24         fill(cnt, cnt + n, 0);  
25         int flow = 0;  
26         while (h[S] < n) {  
27             flow += dfs(S, maxint);  
28         }  
29         return flow;  
30     }  
31     int dfs(int u, int flow) {  
32         if (u == T) {  
33             return flow;  
34         }  
35         int minh = n - 1, ct = 0;  
36         for (vector<Adj>::iterator it = adj[u].begin(); flow && it != adj[u].end(); ++it) {  
37             if (it->c) {  
38                 if (h[it->v] + 1 == h[u]) {  
39                     int k = dfs(it->v, min(it->c, flow));  
40                     if (k) {  
41                         it->c -= k;  
42                         adj[it->v][it->b].c += k;  
43                         flow -= k;  
44                         ct += k;  
45                     }  
46                     if (h[S] >= n) {  
47                         return ct;  
48                     }  
49                 }  
50                 get\_min(minh, h[it->v]);  
51             }  
52         }  
53         if (ct) {  
54             return ct;  
55         }  
56         if (--cnt[h[u]] == 0) {  
57             h[S] = n;  
58         }  
59         h[u] = minh + 1;  
60         ++cnt[h[u]];  
61         return 0;  
62     }  
63 };

### MinCostMaxFlow

 1 **struct** Graph{  
 2     **struct** Adj {  
 3         **int** v, c, w, b;  
 4         Adj(**int** \_v, **int** \_c, **int** \_w, **int** \_b):v(\_v), c(\_c), w(\_w), b(\_b) {};  
 5     }\*st[maxn];  
 6     vector<Adj> adj[maxn];  
 7     **int** n;

int NEXT(int s, int n) {

return s % n;

}  
 8     **void** clear() {  
 9         **for** (**int** i = 0; i < n; ++i) {  
10             adj[i].clear();  
11         }  
12         n = 0;  
13     }  
14     **void** insert(**int** u, **int** v, **int** c, **int** w, **int** d = 0) {  
15         get\_max(n, max(u, v) + 1);  
16         adj[u].push\_back(Adj(v, c, w, adj[v].size()));  
17         adj[v].push\_back(Adj(u, 0, -w, adj[u].size() - 1));  
18         **if** (d) {  
19             adj[v].push\_back(Adj(u, c, w, adj[u].size()));  
20             adj[u].push\_back(Adj(v, 0, -w, adj[v].size() - 1));  
21         }  
22     }  
23     pair<**int**, **int**> mcmf(**int** S, **int** T) {  
24         **int** d;  
25         **int** flow = 0, cost = 0;  
26         **while** ((d = bell(S, T))) {  
27             flow += d;  
28             **for** (**int** v = T; v != S; v = adj[st[v]->v][st[v]->b].v) {  
29                 cost += st[v]->w \* d;  
30                 st[v]->c -= d;  
31                 adj[st[v]->v][st[v]->b].c += d;  
32             }  
33         }  
34         **return** make\_pair(flow, cost);  
35     }  
36     **int** bell(**int** S, **int** T) {  
37         **int** d[maxn], bfs[maxn], hash[maxn];  
38         fill(hash, hash + n, 0);  
39         fill(d, d + n, maxint);  
40         hash[S] = 1; d[S] = 0; bfs[0] = S;  
41         **for** (**int** s = 0, t = 1; s != t; hash[bfs[s]] = 0, s = NEXT(s + 1, n)) { //NEXT(s, n) = s % n  
42             **int** v = bfs[s];  
43             **for** (vector<Adj>::iterator it = adj[v].begin(); it != adj[v].end(); ++it) {  
44                 **if** (it->c > 0 && d[v] + it->w < d[it->v]) {  
45                     d[it->v] = d[v] + it->w;  
46                     st[it->v] = &(\*it);  
47                     **if** (hash[it->v] == 0) {  
48                         hash[it->v] = 1;  
49                         bfs[t] = it->v;  
50                         t = NEXT(t + 1, n);  
51                     }  
52                 }  
53             }  
54         }  
55         **if**(d[T] == maxint) {  
56             **return** 0;  
57         }  
58         **int** ans = maxint;  
59         **for** (**int** v = T; v != S; v = adj[st[v]->v][st[v]->b].v) {  
60             get\_min(ans, st[v]->c);  
61         }  
62         **return** ans;  
63     }  
64 };

### 2-set

**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**

### 双强连通分量

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 {

23 **int** i,j,k;

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

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

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

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

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

29 }

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

31 **return** false;

32 }

33 }

34 **return** true;

35 }

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

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

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

39 zhan[++top] = w;

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

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

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

43 dfs(i,w);

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

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

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

47 k = top;

48 **do**{

49 v[ zhan[top] ] = 1;

50 top--;

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

52 v[w]=1;

53

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

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

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

57 d[k] = 1;

58 }

59 }

60

61 }

62 }

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

64 }

65

66 }

67 **void** solve(){

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

69 ans = 0;

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

71 top = num = 0;

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

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

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

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

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

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

78 dfs(i,0);

79 }

80 }

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

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

83 //printf("%d\n",i);

84 ans++;

85 }

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

87 }

88 **int** main()

89 {

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

91 init();

92 solve();

93 }

94 **return** 0;

95 }

# 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**

### 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

### Aho-Corasick

 1 **class** Trie {  
 2 **public**:  
 3     **const** **static** **int** st = 'A', en = 'z';  
 4     **const** **static** **int** m = en - st + 1;  
 5     **const** **static** **int** maxn = 10002;  
 6     **int** d[maxn][m];     //graph  
 7     **int** t[maxn];        //state  
 8     **int** p[maxn];        //suffix  
 9     **int** n, len;  
10     **void** init() {  
11         len = 1;  
12         n = 0;  
13         t[0] = 0;  
14         memset(d[0], -1, **sizeof**(d[0]));  
15     }  
16     **void** insert(**char** \*s, **int** id) {  
17         **int** i;  
18         **for**(i = 0; \*s; ++s) {  
19             **int** &k = d[i][\*s - st];  
20             **if**(k == -1) {  
21                 k = len++;  
22                 memset(d[k], -1, **sizeof**(d[k]));  
23                 t[k] = 0;  
24             }  
25             i = k;  
26         }  
27         t[i] |= 1<<id;  
28     }  
29     **void** bfs() {  
30         **int** i;  
31         queue<**int**> q;  
32         q.push(0);  
33         p[0] = 0;  
34         **while**(!q.empty()) {  
35             **int** k = q.front();  
36             q.pop();  
37             **for**(i = 0; i < m; i++) {  
38                 **int** &j = d[k][i];  
39                 **if**(-1 == j) {  
40                     j = d[p[k]][i];  
41                     **if**(j == -1)j=0;  
42                 }  
43                 **else** {  
44                     **if** (k) p[j] = d[p[k]][i];  
45                     **else**  p[j] = 0;  
46                     t[j] |= t[p[j]];  
47                     q.push(j);  
48                 }  
49             }  
50         }  
51     }  
52 };

# 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（线段与点的距离）

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

### 判断点是否在多边形内

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

2 **double** ang = 0;

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

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

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

6 to\_normal(c);

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

8 }

9 ang = abs(ang);

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

11 }

# 数论

### 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)) return 0;**

**119 u64 t = Pollard(a), p;**

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

**121 else return t;**

**122 }**

**123**

**124 int main(void)**

**125 {**

**126 u64 l, a, t;**

**127 limit = 1;**

**128 limit = limit << 63; //动态化分段使用**

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

**130 {**

**131 m = 0;**

**132 while(a > 1)**

**133 {**

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

**135 t = prime(a);**

**136 factor[m++] = t;**

**137 a /= t;**

**138 }**

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

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

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

**142 }**

**143 return 0;**

**144 }**

### 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() {

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 }

### 欧拉函数

1 **void** getPhi() {

2 clr(phi);

3 phi[1] = 1;

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

5 **if** (phi[i] == 0) {

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

7 **if** (phi[j] == 0)

8 phi[j] = j;

9 phi[j] = phi[j] / i \* (i - 1);

10 }

11 }

12 }

13 }

### Mobius反演

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 }

# String

### 后缀树

1 // (后缀树\_最长回文子串)

7 #include<cstdio>

8 #include<cstring>

9 #include<cstdlib>

10 #include<cmath>

11 #include<algorithm>

12 #include<string>

13 using namespace std;

14 #define inf 1e-8

15 #define MAXN 2007

16 typedef long long int64;

17 int a[MAXN], height[MAXN], myrank[MAXN], sa[MAXN];

18 int wa[MAXN], wb[MAXN], wv[MAXN], wws[MAXN];

19 int rmq[100][MAXN];

20 int n;

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

22 int r,w;

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

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

25 return wb[a] == wb[b] && r == w;

26 }

//格挡符号要加最大的符号,如：200. 末尾要加最小的符号,如: 0.

27 void getsa(int \*a, int n, int m, int \*sa){ //sa: 1~n, a: 0 ~ n-1,a[n]=0

28 int i,j,k,r,w,p;

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

44 }

45 }

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

47 int i,j,k,r,w;

48 k = 0;

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

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

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

52 }

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

54 int i,j,k,r,m;

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

56 for(i=0; i<=m; i++)

57 for(j=0; j<=n; j++)

58 rmq[i][j] = 200000000;

59 for(i=0; i<=n; i++){

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

61 }

62 for(i=1; i<=m; i++)

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

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

65 }

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

67 if(l > r) swap(l, r);

68 l++;

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

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

71 }

72 int main(){

73 char s[MAXN];

74 int i,j,k;

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

76 memset(a,0,sizeof(a));

77 n = strlen(s);

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

79 a[n] = 200;

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

81 a[n+n+1] = 0;

82 getsa(a, n+n+2, 300, sa);

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

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

85 int ans = -1, ansb;

86 for(i=0; i<n; i++){

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

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

89 ans = 2 \* k - 1;

90 ansb = i - k + 1;

91 }

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

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

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

95 ansb = i - (k-2);

96 printf("\n");

97 }

98 }

99 for(i=ansb; i<ansb + ans; i++)

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

101 printf("\n");

102 }

103 return 0;

104 }

### 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

# Others

### 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**

### 布斯(Booth)乘法(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 }

### 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);

l 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);

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

========随机打乱数组顺序======

random\_shuffle ( a.begin(), a.end() );

# JAVA

### Biginteger

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 }

# Vimrc

在用户名根目录下创建.vimrc,或者进vim,打”:e .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 torte  
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**