

Contents

1	Basic	
1.1	default code	
1.2	.vimrc	
2	math	
2.1	ext gcd	
2.2	FFT other	
2.3	MillerRabin other	
3	flow	
3.1	dinic	
4	string	
4.1	KMP	
4.2	Z-value	
4.3	Z-value-palindrome	
4.4	Suffix Array($O(N \log N)$)	
4.5	Aho-Corasick	
4.6	Aho-Corasick-2016ioicamp	
5	graph	
5.1	Bipartite matching($O(N^3)$)	
6	data structure	
6.1	Treap	
6.2	copy on write treap	
6.3	copy on write segment tree	
6.4	Treap+(HOJ 92)	
6.5	Leftist Tree	
7	geometry	
7.1	Basic	
7.2	Smallest circle problem	

1 Basic

1.1 default code

```

1 #include <bits/stdc++.h>
2 #define PB push_back
3 #define MP make_pair
4 #define F first
5 #define S second
6 #define SZ(x) ((int)(x).size())
7 #define ALL(x) (x).begin(),(x).end()
8 #ifdef _DEBUG_
9     #define debug(...) printf(__VA_ARGS__)
10 #else
11     #define debug(...) (void)0
12 #endif
13 using namespace std;
14 typedef long long ll;
15 typedef pair<int,int> PII;
16 typedef vector<int> VI;
17
18 int main() {
19     return 0;
20 }

```

1.2 .vimrc

```

1 color torte
2 syn on
3 set guifont=Consolas:h16: nu sc ai si ts=4
   sm sts=4 sw=4
4
5 map <F9> <ESC>:w<CR>:!g++ % -o %< -O2 -Wall
   -Wno-unused-result -std=c++0x<CR>
6 map <S-F9> <ESC>:w<CR>:!g++ % -o %< -O2 -
   Wall -Wno-unused-result -D_DEBUG_ -std=c
   ++0x<CR>
7 map <F5> <ESC>:!./%<<CR>
8 map <F6> <ESC>:w<CR>ggvG"+y
9 map <S-F5> <ESC>:!./%< < %<.in<CR>
10 imap <Home> <ESC>^i
11 com INPUT sp %<.in

```

2 math

2.1 ext gcd

```

1 // find one solution (x,y) of ax+by=gcd(
   a,b)
2 void ext_gcd(int a,int b,int &g,int &x,int
   &y)
3 {
4     if(!b){ g=a; x=1; y=0; }
5     else{ ext_gcd(b, a%b, g, y, x); y -= x*(a
   /b); }
6 }

```

2.2 FFT other

```

1  /* FFT code from shik in CodeForces*/
2  /* zj a577*/
3  #include <bits/stdc++.h>
4  using namespace std;
5  const int N=300000;
6
7  const double PI=acos(-1.0);
8  struct vir{
9      double re,im;
10     vir( double _re=0, double _im=0 ):re(
11         _re),im(_im){}
12 };
13 vir operator +( vir a, vir b ) { return vir
14     (a.re+b.re,a.im+b.im); }
15 vir operator -( vir a, vir b ) { return vir
16     (a.re-b.re,a.im-b.im); }
17 vir operator *( vir a, vir b ) { return vir
18     (a.re*b.re-a.im*b.im,a.re*b.im+a.im*b.
19     re); }
20 vir x1[2*N],x2[2*N];
21
22 int rev( int x, int len ) {
23     int r=0,i;
24     for ( i=0;i<len; i++,x>>=1 ) r=(r<<1)
25         +(x&1);
26     return r;
27 }
28 void change( vir *x, int len, int loglen )
29 {
30     for ( int i=0; i<len; i++ )
31         if ( rev(i,loglen)<i ) swap(x[rev(i
32             ,loglen)],x[i]);
33 }
34 void fft( vir *x, int len, int loglen ) {
35     change(x,len,loglen);
36     int i,j,s,t=1;
37     for ( i=0; i<loglen; i++,t<<=1 ) {
38         for ( s=0; s<len; s+=t+t ) {
39             vir a,b,wo(cos(PI/t),sin(PI/t))
40                 ,wn(1,0);
41             for ( j=s; j<s+t; j++ ) {
42                 a=x[j]; b=x[j+t]*wn;
43                 x[j]=a+b; x[j+t]=a-b;
44                 wn=wn*wo;
45             }
46         }
47     }
48 }
49 void dit_fft( vir *x, int len, int loglen )
50 {
51     int i,j,s,t=len>>1;
52     for ( i=0; i<loglen; i++,t>>=1 ) {
53         for ( s=0; s<len; s+=t+t ) {
54             vir a,b,wn(1,0),wo(cos(PI/t),-
55                 sin(PI/t));
56             for ( j=s; j<s+t; j++ ) {
57                 a=x[j]+x[j+t]; b=(x[j]-x[j+
58                     t])*wn;
59                 x[j]=a; x[j+t]=b;
60                 wn=wn*wo;
61             }
62         }
63     }
64     change(x,len,loglen);
65     for ( i=0; i<len; i++ ) x[i].re/=len;
66 }
67
68 char a[N],b[N];
69 int ans[2*N];
70
71 int main()
72 {
73     int na,nb,len=1,loglen=0;
74     while( scanf( "%s%s",a,b)==2)
75     {
76         for(int i=2*N-1;i>=0;i--)
77             x1[i]=x2[i]=0.0;
78         for(na=0;a[na];na++);
79         for(nb=0;b[nb];nb++);
80         for(int i=na-1;i>=0;i--)
81             x1[i]=(double)(a[na-i-1]-'0');
82         for(int i=nb-1;i>=0;i--)
83             x2[i]=(double)(b[nb-i-1]-'0');
84         while(len<=2*max(na,nb)+5)
85         {
86             len*=2;
87             loglen++;
88         }
89         fft(x1,len,loglen);
90         fft(x2,len,loglen);
91         for(int i=0;i<len;i++)
92             x1[i]=x1[i]*x2[i];
93         dit_fft(x1,len,loglen);
94         for(int i=len-1;i>=0;i--)
95             ans[i]=(int)round(x1[i].re+0.01);
96         for(int i=0;i<len;i++)
97         {
98             if(ans[i]>=10)
99             {
100                 ans[i+1]+=ans[i]/10;
101                 ans[i]%=10;
102             }
103         }
104         bool zero=0;
105         for(int i=len-1;i>=0;i--)
106         {
107             //printf("%d\n",ans[i]);
108             if(zero)
109                 printf("%d",ans[i]);
110             else if(ans[i]>0)
111             {
112                 printf("%d",ans[i]);
113                 zero=1;
114             }
115         }
116         if(!zero)
117             printf("0");
118         puts("");
119     }
120     return 0;
121 }

```

2.3 MillerRabin other

```

1  /* Miller Rabin code from ioicamp */
2  #include <bits/stdc++.h>
3  #define PB push_back
4  #define MP make_pair

```

```

5 #define F first
6 #define S second
7 #define SZ(x) ((int)(x).size())
8 #define ALL(x) (x).begin(),(x).end()
9 #ifdef _DEBUG_
10 #define debug(...) printf(__VA_ARGS__)
11 #else
12 #define debug(...) 0
13 #endif
14 using namespace std;
15 typedef long long ll;
16 typedef pair<int,int> PII;
17 typedef vector<int> VI;
18
19 ll mul(ll a, ll b, ll n) {
20     ll r = 0;
21     a %= n, b %= n;
22     while(b) {
23         if(b&1) r = (a+r>=n ? a+r-n : a+r);
24         a = (a+a>=n ? a+a-n : a+a);
25         b >>= 1;
26     }
27     return r;
28 }
29
30 ll bigmod(ll a, ll d, ll n) {
31     if(d==0) return 1LL;
32     if(d==1) return a % n;
33     return mul(bigmod(mul(a, a, n), d/2, n),
34                d%2?a:1, n);
35 }
36
37 const bool PRIME = 1, COMPOSITE = 0;
38 bool miller_rabin(ll n, ll a) {
39     if(__gcd(a, n) == n) return PRIME;
40     if(__gcd(a, n) != 1) return COMPOSITE;
41     ll d = n-1, r = 0, res;
42     while(d%2==0) { ++r; d/=2; }
43     res = bigmod(a, d, n);
44     if(res == 1 || res == n-1) return PRIME;
45     while(r--) {
46         res = mul(res, res, n);
47         if(res == n-1) return PRIME;
48     }
49     return COMPOSITE;
50 }
51
52 bool isprime(ll n) {
53     if(n==1) return COMPOSITE;
54     ll as[7] = {2, 325, 9375, 28178, 450775,
55                9780504, 1795265022};
56     for(int i=0; i<7; i++)
57         if(miller_rabin(n, as[i]) == COMPOSITE)
58             return COMPOSITE;
59     return PRIME;
60 }

```

3 flow

3.1 dinic

```

1 #include <bits/stdc++.h>

```

```

2 #define PB push_back
3 #define MP make_pair
4 #define F first
5 #define S second
6 #define SZ(x) ((int)(x).size())
7 using namespace std;
8 typedef long long ll;
9 typedef pair<int,int> PII;
10 typedef vector<int> VI;
11
12 /***/
13 // dinic
14 const int MAXV=300;
15 const int MAXE=10000;
16 const int INF=(int)1e9+10;
17
18 struct E{
19     int to,co;//capacity
20     E(int t=0,int c=0):to(t),co(c){}
21 }eg[2*MAXE];
22
23 // source:0 sink:n-1
24 struct Flow{
25     VI e[MAXV];
26     int ei,v;
27     void init(int n) {
28         v=n;
29         ei=0;
30         for(int i=0;i<n;i++)
31             e[i]=VI();
32     }
33     void add(int a,int b,int c) { //a to b ,
34         // maxflow=c
35         eg[ei]=E(b,c);
36         e[a].PB(ei);
37         ei++;
38         eg[ei]=E(a,0);
39         e[b].PB(ei);
40         ei++;
41     }
42     int d[MAXV],qu[MAXV],ql,qv;
43     bool BFS() {
44         memset(d,-1,v*sizeof(int));
45         ql=qv=0;
46         qu[qv++]=0;
47         d[0]=0;
48         while(ql<qv && d[qv-1]==-1) {
49             int n=qu[ql++];
50             VI &v=e[n];
51             for(int i=v.size()-1;i>=0;i--) {
52                 int u=v[i];
53                 if(d[eg[u].to]==-1 && eg[u].co>0) {
54                     d[eg[u].to]=d[n]+1;
55                     qu[qv++]=eg[u].to;
56                 }
57             }
58         }
59         return d[qv-1]!=-1;
60     }
61     int ptr[MAXV];
62     int go(int n,int p) {
63         if(n==v-1)
64             return p;
65         VI &u=e[n];

```

```

66     int temp;
67     for(int i=ptr[n];i<SZ(u);i++)
68     {
69         if(d[n]+1!=d[eg[u[i]].to] || eg[u[i]
70             ].co==0)
71             continue;
72         if((temp=go(eg[u[i]].to,min(p,eg[u[i]
73             ].co)))==0)
74             continue;
75         eg[u[i]].co-=temp;
76         eg[u[i]^1].co+=temp;
77         ptr[n]=i;
78         return temp;
79     }
80     ptr[n]=SZ(u);
81     return 0;
82 }
83 int max_flow() {
84     int ans=0,temp;
85     while(BFS()) {
86         for(int i=0;i<v;i++)
87             ptr[i]=0;
88         while((temp=go(0,INF))>0)
89             ans+=temp;
90     }
91 }
92 }
93 int main() {
94     return 0;
95 }
96 }

```

4 string

4.1 KMP

```

1 void KMP_build(const char *S,int *F) {
2     int p=F[0]=-1;
3     for(int i=1;S[i];i++) {
4         while(p!=-1 && S[p+1]!=S[i])
5             p=F[p];
6         if(S[p+1]==S[i])
7             p++;
8         F[i]=p;
9     }
10 }
11
12 VI KMP_match(const char *S,const int *F,
13     const char *T) {
14     VI ans;
15     int p=-1;
16     for(int i=0;T[i];i++) {
17         while(p!=-1 && S[p+1]!=T[i])
18             p=F[p];
19         if(S[p+1]==T[i])
20             p++;
21         if(!S[p+1]) {
22             ans.pb(i-p);
23             p=F[p];
24         }
25     }
26 }

```

```

24     }
25     return ans;
26 }

```

4.2 Z-value

```

1 void Z_build(const char *S,int *Z) {
2     Z[0]=0;
3     int bst=0;
4     for(int i=1;S[i];i++) {
5         if(Z[bst]+bst<i) Z[i]=0;
6         else Z[i]=min(Z[bst]+bst-i,Z[i-bst]);
7         while(S[Z[i]]==S[i+Z[i]]) Z[i]++;
8         if(Z[i]+i>Z[bst]+bst) bst=i;
9     }
10 }

```

4.3 Z-value-palindrome

```

1 // AC code of NTUJ1871
2 #include <bits/stdc++.h>
3 #define pb push_back
4 #define F first
5 #define S second
6 #define SZ(x) ((int)(x).size())
7 #define MP make_pair
8 using namespace std;
9 typedef long long ll;
10 typedef pair<int,int> PII;
11 typedef vector<int> VI;
12
13 char in[100100];
14 char s[200100];
15 int z[200100];
16
17 int main()
18 {
19     while(gets(in))
20     {
21         int len=1;
22         for(int i=0;in[i];i++)
23         {
24             s[len++]='*';
25             s[len++]=in[i];
26         }
27         s[len]=0;
28         z[0]=0;
29         z[1]=0;
30         int bst=1;
31         for(int i=1;i<len;i++)
32         {
33             z[i]=min(bst+z[bst]-i,z[bst+bst-i]);
34             while(s[i+z[i]+1]==s[i-z[i]-1])
35                 z[i]++;
36             if(z[i]+i>bst+z[bst])
37                 bst=i;
38         }
39         /*for(int i=1;i<len;i++)
40             putchar(s[i]);
41         puts("");
42         for(int i=1;i<len;i++)

```

```

43     printf("%d",z[i]);
44     puts("");*/
45     bool yes=0;
46     for(int i=3;i<len;i+=2)
47         if(z[(i+1)/2]==i/2 && z[(i+len)
48             /2]==(len-i-1)/2)
49             yes=1;
50     if(yes)
51         puts("www");
52     else
53         puts("vvvvvvv");
54     return 0;
55 }

```

4.4 Suffix Array($O(N\log N)$)

```

1 const int SASIZE=100020; // >= (max length
   of string + 20)
2 struct SA{
3     char S[SASIZE]; // put target string into
   S[0:(len-1)]
4     // you can change the type of S into int
   if required
5     // if the string is in int, please avoid
   number < 0
6     int R[SASIZE*2],SA[SASIZE];
7     int tR[SASIZE*2],tSA[SASIZE];
8     int cnt[SASIZE],len; // set len
   before calling build()
9     int H[SASIZE];
10
11 void build_SA() {
12     int maxR=0;
13     for(int i=0;i<len;i++)
14         R[i]=S[i];
15     for(int i=0;i<len;i++)
16         R[len+i]=-1;
17     memset(cnt,0,sizeof(cnt));
18     for(int i=0;i<len;i++)
19         maxR=max(maxR,R[i]);
20     for(int i=0;i<len;i++)
21         cnt[R[i]+1]++;
22     for(int i=1;i<=maxR;i++)
23         cnt[i]+=cnt[i-1];
24     for(int i=0;i<len;i++)
25         SA[cnt[R[i]]++]=i;
26     for(int i=1;i<len;i*=2)
27     {
28         memset(cnt,0,sizeof(int)*(maxR+10));
29         memcpy(tSA,SA,sizeof(int)*(len+10));
30         memcpy(tR,R,sizeof(int)*(len+i+10));
31         for(int j=0;j<len;j++)
32             cnt[R[j]+1]++;
33         for(int j=1;j<=maxR;j++)
34             cnt[j]+=cnt[j-1];
35         for(int j=len-i;j<len;j++)
36             SA[cnt[tR[j]]++]=j;
37         for(int j=0;j<len;j++)
38         {
39             int k=tSA[j]-i;
40             if(k<0)
41                 continue;

```

```

42         SA[cnt[R[k]]++]=k;
43     }
44     int num=0;
45     maxR=0;
46     R[SA[0]]=num;
47     for(int j=1;j<len;j++)
48     {
49         if(tR[SA[j-1]]<tR[SA[j]] || tR[SA[j]
50             -1]+i<tR[SA[j]+i])
51             num++;
52         R[SA[j]]=num;
53         maxR=max(maxR,R[SA[j]]);
54     }
55 }
56 void build_H() {
57     memset(H,0,sizeof(int)*(len+10));
58     for(int i=0;i<len;i++)
59     {
60         if(R[i]==0)
61             continue;
62         int &t=H[R[i]];
63         if(i>0)
64             t=max(0,H[R[i-1]]-1);
65         while(S[i+t]==S[SA[R[i]-1]+t]) t++;
66     }
67 }
68 }sa;

```

4.5 Aho-Corasick

```

1 // AC code of UVa 10679
2 #include <cstdio>
3 #include <cstring>
4 #include <new>
5
6 struct Trie {
7     int c;
8     bool fi=0;
9     Trie *fail,*ch[52];
10     Trie():c(0){memset(ch,0,sizeof(ch));}
11 }trie[1000100];
12
13 char m[1010],f[100100];
14 Trie *str[1010],*na,*root;
15
16 inline int c_i(char a) {
17     return (a>='A' && a<='Z') ? a-'A' : a-'a'
18         +26;
19 }
20 void insert(char *s,int num) {
21     Trie *at=root;
22     while(*s) {
23         if(!at->ch[c_i(*s)])
24             at->ch[c_i(*s)]=new (na++) Trie();
25         at=at->ch[c_i(*s)],s++;
26     }
27     str[num]=at;
28 }
29
30 Trie *q[1000100];
31 int ql,qr;

```

```

32
33 void init() {
34     ql=qr=-1;
35     q[++qr]=root;
36     root->fail=NULL;
37     while(ql<qr) {
38         Trie *n=q[++ql],*f;
39         for(int i=0;i<52;i++) {
40             if(!n->ch[i])
41                 continue;
42             f=n->fail;
43             while(f && !f->ch[i])
44                 f=f->fail;
45             n->ch[i]->fail=f?f->ch[i]:root;
46             q[++qr]=n->ch[i];
47         }
48     }
49 }
50
51 void go(char *s) {
52     Trie*p=root;
53     while(*s) {
54         while(p && !p->ch[c_i(*s)])
55             p=p->fail;
56         p=p->ch[c_i(*s)]:root;
57         p->fi=1;
58         s++;
59     }
60 }
61
62 void AC() {
63     for(int i=qr;i>0;i--)
64         q[i]->fail->c+=q[i]->c;
65 }
66
67 int main() {
68     int T,q;
69     scanf("%d",&T);
70     while(T--) {
71         na=trie;
72         root=new (na++) Trie();
73         scanf("%s",f);
74         scanf("%d",&q);
75         for(int i=0;i<q;i++) {
76             scanf("%s",m);
77             insert(m,i);
78         }
79         init();
80         go(f);
81         for(int i=0;i<q;i++)
82             puts(str[i]->fi?"y":"n");
83     }
84     return 0;
85 }

```

4.6 Aho-Corasick-2016ioicamp

```

1 // AC code of 2016ioicamp 54
2 #include <bits/stdc++.h>
3 #define PB push_back
4 #define MP make_pair
5 #define F first
6 #define S second

```

```

7 #define SZ(x) ((int)(x).size())
8 #define ALL(x) (x).begin(),(x).end()
9 #ifdef _DEBUG_
10     #define debug(...) printf(__VA_ARGS__)
11 #else
12     #define debug(...) (void)0
13 #endif
14 using namespace std;
15 typedef long long ll;
16 typedef pair<int,int> PII;
17 typedef vector<int> VI;
18
19 const int MAXNM=100010;
20 int pp[MAXNM];
21
22 const int sizz=100010;
23 int nx[sizz][26],spt;
24 int fl[sizz],efl[sizz],ed[sizz];
25 int len[sizz];
26 int newnode(int len_=0) {
27     for(int i=0;i<26;i++)nx[spt][i]=0;
28     ed[spt]=0;
29     len[spt]=len_;
30     return spt++;
31 }
32 int add(char *s,int p) {
33     int l=1;
34     for(int i=0;s[i];i++) {
35         int a=s[i]-'a';
36         if(nx[p][a]==0) nx[p][a]=newnode(l);
37         p=nx[p][a];
38         l++;
39     }
40     ed[p]=1;
41     return p;
42 }
43 int q[sizz],qs,qe;
44 void make_fl(int root) {
45     fl[root]=efl[root]=0;
46     qs=qe=0;
47     q[qe++]=root;
48     for(;qs!=qe;) {
49         int p=q[qs++];
50         for(int i=0;i<26;i++) {
51             int t=nx[p][i];
52             if(t==0) continue;
53             int tmp=fl[p];
54             for(;tmp&&nx[tmp][i]==0;) tmp=fl[tmp];
55             fl[t]=tmp?nx[tmp][i]:root;
56             efl[t]=ed[fl[t]]?fl[t]:efl[fl[t]];
57             q[qe++]=t;
58         }
59     }
60 }
61 char s[MAXNM];
62 char a[MAXNM];
63
64 int dp[MAXNM][4];
65
66 void mmax(int &a,int b) {
67     a=max(a,b);
68 }
69
70 void match(int root) {

```

```

71 int p=root;
72 for(int i=1;s[i];i++) {
73     int a=s[i]-'a';
74     for(;p&nx[p][a]==0;p=f1[p]);
75     p=p?nx[p][a]:root;
76     for(int j=1;j<=3;j++)
77         dp[i][j]=dp[i-1][j];
78     for(int t=p;t;t=efl[t]) {
79         if(!led[t])
80             continue;
81         for(int j=1;j<=3;j++)
82             mmax(dp[i][j],dp[i-len[t]][j-1]+(pp
83                 [i]-pp[i-len[t]]));
84     }
85 }
86
87 int main() {
88     int T;
89     scanf("%d",&T);
90     while(T--) {
91         int n,m;
92         scanf("%d%d",&n,&m);
93         scanf("%s",s+1);
94         for(int i=1;i<=n;i++)
95             scanf("%d",pp+i);
96         for(int i=1;i<=n;i++)
97             pp[i]+=pp[i-1];
98         spt=1;
99         int root=newnode();
100        for(int i=0;i<m;i++) {
101            scanf("%s",a);
102            add(a,root);
103        }
104        make_f1(root);
105        for(int i=1;i<=n;i++)
106            dp[i][1]=dp[i][2]=dp[i][3]=0;
107        match(root);
108        printf("%d\n",dp[n][3]);
109    }
110    return 0;
111 }

```

5 graph

5.1 Bipartite matching($O(N^3)$)

```

1 // NTUJ1263
2 #include <bits/stdc++.h>
3 #define pb push_back
4 #define F first
5 #define S second
6 #define SZ(x) ((int)(x).size())
7 #define MP make_pair
8 using namespace std;
9 typedef long long ll;
10 typedef pair<int,int> PII;
11 typedef vector<int> VI;
12
13 bool is(ll x)
14 {
15     ll l=1,r=2000000,m;

```

```

16 while(l<=r)
17 {
18     m=(l+r)/2;
19     if(m*m==x)
20         return 1;
21     if(m*m<x)
22         l=m+1;
23     else
24         r=m-1;
25 }
26 return 0;
27 }
28
29 VI odd,even;
30 int in[300];
31 VI e[300];
32 int match[300];
33 bool vis[300];
34
35 bool DFS(int x)
36 {
37     vis[x]=1;
38     for(int u:e[x])
39     {
40         if(match[u]==-1 || (!vis[match[u]]&&DFS
41             (match[u])))
42         {
43             match[u]=x;
44             match[x]=u;
45             return 1;
46         }
47     }
48     return 0;
49 }
50
51 int main()
52 {
53     int N;
54     while(scanf("%d",&N)==1)
55     {
56         odd.clear();
57         even.clear();
58         for(int i=0;i<N;i++)
59             e[i].clear();
60         for(int i=0;i<N;i++)
61         {
62             scanf("%d",in+i);
63             if(in[i]%2==0)
64                 even.pb(i);
65             else
66                 odd.pb(i);
67         }
68         for(int i:even)
69             for(int j:odd)
70                 if(is(1ll*in[i]*in[i]+1ll*in[j]*in[
71                     j]) && __gcd(in[i],in[j])==1)
72                     e[i].pb(j), e[j].pb(i);
73         int ans=0;
74         fill(match,match+N,-1);
75         for(int i=0;i<N;i++)
76             if(match[i]==-1)
77             {
78                 fill(vis,vis+N,0);
79                 if(DFS(i))
80                     ans++;
81             }
82     }
83     printf("%d\n",ans);
84 }

```



```

79     }
80     printf("%d\n",ans);
81 }
82 return 0;
83 }

```

6 data structure

6.1 Treap

```

1 #include <cstdlib>
2 #include <cstdio>
3 #include <algorithm>
4
5 using namespace std;
6
7 typedef long long ll;
8
9 const int N = 100000 + 10;
10
11 struct Treap {
12     static Treap mem[N], *pmem;
13
14     int sz, pri;
15     ll val, sum, add;
16     Treap *l, *r;
17
18     Treap() {}
19     Treap(ll _val):
20         l(NULL), r(NULL), sz(1), pri(rand()),
21         val(_val), sum(_val), add(0) {}
22 } Treap::mem[N], *Treap::pmem = Treap::mem;
23
24 Treap* make(ll val) {
25     return new (Treap::pmem++) Treap(val);
26 }
27
28 inline int sz(Treap *t) {
29     return t ? t->sz : 0;
30 }
31
32 inline ll sum(Treap *t) {
33     return t ? t->sum + t->add * sz(t) : 0;
34 }
35
36 inline void add(Treap *t, ll x) {
37     t->add += x;
38 }
39
40 void push(Treap *t) {
41     t->val += t->add;
42     if(t->l) t->l->add += t->add;
43     if(t->r) t->r->add += t->add;
44     t->add = 0;
45 }
46
47 void pull(Treap *t) {
48     t->sum = sum(t->l) + sum(t->r) + t->val;
49     t->sz = sz(t->l) + sz(t->r) + 1;
50 }
51
52 Treap* merge(Treap *a, Treap *b) {

```

```

52 if(!a || !b) return a ? a : b;
53 else if(a->pri > b->pri) {
54     push(a);
55     a->r = merge(a->r, b);
56     pull(a);
57     return a;
58 }
59 else {
60     push(b);
61     b->l = merge(a, b->l);
62     pull(b);
63     return b;
64 }
65 }
66
67 void split(Treap* t, int k, Treap *&a,
68     Treap *&b) {
69     if(!t) a = b = NULL;
70     else if(sz(t->l) < k) {
71         a = t;
72         push(a);
73         split(t->r, k - sz(t->l) - 1, a->r, b);
74         pull(a);
75     }
76     else {
77         b = t;
78         push(b);
79         split(t->l, k, a, b->l);
80         pull(b);
81     }
82 }
83
84 int main() {
85     srand(105105);
86
87     int n, q;
88     scanf("%d%d", &n, &q);
89
90     Treap *t = NULL;
91     for(int i = 0; i < n; i++) {
92         ll tmp;
93         scanf("%lld", &tmp);
94         t = merge(t, make(tmp));
95     }
96
97     while(q--) {
98         char c;
99         int l, r;
100         scanf("\n%c %d %d", &c, &l, &r);
101
102         Treap *tl = NULL, *tr = NULL;
103         if(c == 'Q') {
104             split(t, l - 1, tl, t);
105             split(t, r - l + 1, t, tr);
106             printf("%lld\n", sum(t));
107             t = merge(tl, merge(t, tr));
108         }
109         else {
110             ll x;
111             scanf("%lld", &x);
112             split(t, l - 1, tl, t);
113             split(t, r - l + 1, t, tr);
114             add(t, x);
115             t = merge(tl, merge(t, tr));
116         }

```



```

116 }
117
118 return 0;
119 }

```

6.2 copy on write treap

```

1 #include <cstdlib>
2 #include <cstdio>
3 #include <algorithm>
4 #include <climits>
5 #include <cstring>
6
7 using namespace std;
8
9 const int N = 1000000 + 10;
10
11 struct Treap {
12     char val;
13     int sz, refs;
14     Treap *l, *r;
15
16     Treap() {}
17     Treap(char _val):
18         val(_val), sz(1), refs(0), l(NULL),
19         r(NULL) {}
20 };
21 Treap* make(Treap* t) {
22     return new Treap(*t);
23 }
24
25 Treap* make(char _val) {
26     return new Treap(_val);
27 }
28
29 void print_ref(Treap* t) {
30     if(!t) return;
31     print_ref(t->l);
32     printf("%d ", t->refs);
33     print_ref(t->r);
34 }
35
36 void print(Treap* t) {
37     if(!t) return;
38     print(t->l);
39     putchar(t->val);
40     print(t->r);
41 }
42
43 void takeRef(Treap* t) {
44     if(t) t->refs++;
45 }
46
47 void dropRef(Treap* t) {
48     if(t) {
49         char c = t->val;
50         t->refs--;
51         if(t->refs <= 0) {
52             dropRef(t->l);
53             dropRef(t->r);
54             delete t;
55         }

```

```

56     }
57 }
58
59 int sz(Treap* t) {
60     return t ? t->sz : 0;
61 }
62
63 int rnd(int m) {
64     static int x = 851025;
65     return (x = (x*0xdefaced+1) & INT_MAX)
66         % m;
67 }
68 void pull(Treap* t) {
69     t->sz = sz(t->l) + sz(t->r) + 1;
70 }
71
72 Treap* merge(Treap* a, Treap* b) {
73     if(!a || !b) {
74         Treap* t = a ? make(a) : make(b);
75         t->refs = 0;
76         takeRef(t->l);
77         takeRef(t->r);
78         return t;
79     }
80
81     Treap* t;
82     if( rnd(a->sz+b->sz) < a->sz ) {
83         t = make(a);
84         t->refs = 0;
85         t->r = merge(a->r, b);
86         takeRef(t->l);
87         takeRef(t->r);
88     }
89     else {
90         t = make(b);
91         t->refs = 0;
92         t->l = merge(a, b->l);
93         takeRef(t->l);
94         takeRef(t->r);
95     }
96
97     pull(t);
98     return t;
99 }
100
101 void split(Treap* t, int k, Treap* &a,
102 Treap* &b) {
103     if(!t) a = b = NULL;
104     else if(sz(t->l) < k) {
105         a = make(t);
106         a->refs = 0;
107         split(a->r, k-sz(t->l)-1, a->r, b);
108         takeRef(a->l);
109         takeRef(a->r);
110         pull(a);
111     }
112     else {
113         b = make(t);
114         b->refs = 0;
115         split(b->l, k, a, b->l);
116         takeRef(b->l);
117         takeRef(b->r);
118         pull(b);

```

6.3 copy on write segment tree

```

119 }
120
121 void print_inorder(Treap* t) {
122     if(!t) return;
123     putchar(t->val);
124     print_inorder(t->l);
125     print_inorder(t->r);
126 }
127
128 char s[N];
129
130 int main() {
131     int m;
132     scanf("%d", &m);
133     scanf("%s", s);
134     int n = strlen(s);
135     int q;
136     scanf("%d", &q);
137
138     Treap* t = NULL;
139     for(int i = 0; i < n; i++) {
140         Treap *a = t, *b = make(s[i]);
141         t = merge(a, b);
142         dropRef(a);
143         dropRef(b);
144     }
145
146     while(q--) {
147         int l, r, x;
148         scanf("%d%d%d", &l, &r, &x);
149         r++;
150
151         Treap *a, *b, *c, *d;
152         a = b = c = d = NULL;
153         split(t, l, a, b);
154         dropRef(a);
155         split(b, r-l, c, d);
156         dropRef(b);
157         dropRef(d);
158         split(t, x, a, b);
159         dropRef(t);
160         Treap* t2 = merge(c, b);
161         dropRef(b);
162         dropRef(c);
163         t = merge(a, t2);
164         dropRef(a);
165         dropRef(t2);
166
167         if(t->sz > m) {
168             Treap* t2 = NULL;
169             split(t, m, t2, a);
170             dropRef(a);
171             dropRef(t);
172             t = t2;
173         }
174     }
175
176     print(t);
177     putchar('\n');
178
179     return 0;
180 }

```

```

1 #include <cstdlib>
2 #include <cstdio>
3 #include <algorithm>
4 #include <vector>
5
6 using namespace std;
7
8 const int N = 50000 + 10;
9 const int Q = 10000 + 10;
10
11 struct Seg {
12     static Seg mem[N*80], *pmem;
13
14     int val;
15     Seg *tl, *tr;
16
17     Seg() :
18         tl(NULL), tr(NULL), val(0) {}
19
20     Seg* init(int l, int r) {
21         Seg* t = new (pmem++) Seg();
22         if(l != r) {
23             int m = (l+r)/2;
24             t->tl = init(l, m);
25             t->tr = init(m+1, r);
26         }
27         return t;
28     }
29
30     Seg* add(int k, int l, int r) {
31         Seg* _t = new (pmem++) Seg(*this);
32         if(l==r) {
33             _t->val++;
34             return _t;
35         }
36
37         int m = (l+r)/2;
38         if(k <= m) _t->tl = tl->add(k, l, m);
39         else _t->tr = tr->add(k, m+1, r);
40
41         _t->val = _t->tl->val + _t->tr->val;
42         return _t;
43     }
44 } Seg::mem[N*80], *Seg::pmem = mem;
45
46 int query(Seg* ta, Seg* tb, int k, int l,
47     int r) {
48     if(l == r) return l;
49
50     int m = (l+r)/2;
51
52     int a = ta->tl->val;
53     int b = tb->tl->val;
54     if(b-a >= k) return query(ta->tl, tb->tl,
55         k, l, m);
56     else return query(ta->tr, tb->tr, k
57         -(b-a), m+1, r);
58 };
59
60 struct Query {
61     int op, l, r, k, c, v;
62 }

```

```

60 bool operator<(const Query b) const {
61     return c < b.c;
62 }
63 } qs[Q];
64 int arr[N];
65 Seg *t[N];
66 vector<int> vec2;
67
68 int main() {
69     int T;
70     scanf("%d", &T);
71
72     while(T--) {
73         int n, q;
74         scanf("%d%d", &n, &q);
75
76         for(int i = 1; i <= n; i++) {
77             scanf("%d", arr+i);
78             vec2.push_back(arr[i]);
79         }
80         for(int i = 0; i < q; i++) {
81             scanf("%d", &qs[i].op);
82             if(qs[i].op == 1) scanf("%d%d%d", &qs[i].l, &qs[i].r, &qs[i].k);
83             else scanf("%d", &qs[i].c, &qs[i].v);
84
85             if(qs[i].op == 2) vec2.push_back(qs[i].v);
86         }
87         sort(vec2.begin(), vec2.end());
88         vec2.resize(unique(vec2.begin(), vec2.end())-vec2.begin());
89         for(int i = 1; i <= n; i++) arr[i] = lower_bound(vec2.begin(), vec2.end(), arr[i]) - vec2.begin();
90         int mn = 0, mx = vec2.size()-1;
91
92         for(int i = 0; i <= n; i++) t[i] = NULL;
93
94         t[0] = new (Seg::pmem++) Seg();
95         t[0] = t[0]->init(mn, mx);
96         int ptr = 0;
97         for(int i = 1; i <= n; i++) {
98             t[i] = t[i-1]->add(arr[i], mn, mx);
99         }
100         for(int i = 0; i < q; i++) {
101             int op = qs[i].op;
102             if(op == 1) {
103                 int l = qs[i].l, r = qs[i].r, k = qs[i].k;
104                 printf("%d\n", vec2[query(t[l-1], t[r], k, mn, mx)]);
105             }
106             if(op == 2) {
107                 continue;
108             }
109             if(op == 3) puts("7122");
110         }
111         vec2.clear();
112         Seg::pmem = Seg::mem;
113     }
114 }
115

```

```

116 return 0;
117 }

```

6.4 Treap+(H0J 92)

```

1 #include <cstdlib>
2 #include <cstdio>
3 #include <algorithm>
4 #include <cstring>
5
6 using namespace std;
7
8 const int INF = 103456789;
9
10 struct Treap {
11     int pri, sz, val, chg, rev, sum, lsum,
12         rsum, mx_sum;
13     Treap *l, *r;
14
15     Treap() {}
16     Treap(int _val) :
17         pri(rand()), sz(1), val(_val), chg(
18             INF), rev(0), sum(_val), lsum(
19                 _val), rsum(_val), mx_sum(_val),
20                 l(NULL), r(NULL) {}
21
22     int sz(Treap* t) {return t ? t->sz : 0;}
23     int sum(Treap* t) {
24         if(!t) return 0;
25         if(t->chg == INF) return t->sum;
26         else return t->chg*t->sz;
27     }
28     int lsum(Treap* t) {
29         if(!t) return -INF;
30         if(t->chg != INF) return max(t->chg,
31             (t->chg)*(t->sz));
32         if(t->rev) return t->rsum;
33         return t->lsum;
34     }
35     int rsum(Treap* t) {
36         if(!t) return -INF;
37         if(t->chg != INF) return max(t->chg,
38             (t->chg)*(t->sz));
39         if(t->rev) return t->lsum;
40         return t->rsum;
41     }
42     int mx_sum(Treap* t) {
43         if(!t) return -INF;
44         if(t->chg != INF) return max(t->chg,
45             (t->chg)*(t->sz));
46         return t->mx_sum;
47     }
48     void push(Treap* t) {
49         if(t->chg != INF) {
50             t->val = t->chg;
51             t->sum = (t->sz) * (t->chg);
52             t->lsum = t->rsum = t->mx_sum = max(
53                 t->sum, t->val);
54             if(t->l) t->l->chg = t->chg;
55             if(t->r) t->r->chg = t->chg;
56             t->chg = INF;
57         }
58     }
59

```

```

51     }
52     if(t->rev) {
53         swap(t->l, t->r);
54         if(t->l) t->l->rev ^= 1;
55         if(t->r) t->r->rev ^= 1;
56         t->rev = 0;
57     }
58 }
59
60 void pull(Treap* t) {
61     t->sz = sz(t->l)+sz(t->r)+1;
62     t->sum = sum(t->l)+sum(t->r)+t->val;
63     t->lsum = max(lsum(t->l), sum(t->l)+max
123     (0, lsum(t->r))+t->val);
124
64     t->rsum = max(rsum(t->r), sum(t->r)+max
125     (0, rsum(t->l))+t->val);
126
65     t->mx_sum = max(max(mx_sum(t->l),
127     mx_sum(t->r)), max(0, rsum(t->l))+
128     max(0, lsum(t->r))+t->val);
129
66 }
67
68 Treap* merge(Treap* a, Treap* b) {
69     if(!a || !b) return a ? a : b;
70     if(a->pri > b->pri) {
71         push(a);
72         a->r = merge(a->r, b);
73         pull(a);
74         return a;
75     }
76     else {
77         push(b);
78         b->l = merge(a, b->l);
79         pull(b);
80         return b;
81     }
82 }
83
84 void split(Treap* t, int k, Treap* &a,
Treap* &b) {
85     if(!t) {
86         a = b = NULL;
87         return ;
88     }
89     push(t);
90     if(sz(t->l) < k) {
91         a = t;
92         push(a);
93         split(t->r, k-sz(t->l)-1, a->r, b);
94         pull(a);
95     }
96     else {
97         b = t;
98         push(b);
99         split(t->l, k, a, b->l);
100        pull(b);
101    }
102 }
103
104 void del(Treap* t) {
105     if(!t) return;
106     del(t->l);
107     del(t->r);
108     delete t;
109 }
110
111 int main() {
112     srand(7122);
113
114     int n, m;
115     scanf("%d%d", &n, &m);
116
117     Treap* t = NULL;
118     for(int i = 0; i < n; i++) {
119         int x;
120         scanf("%d", &x);
121         t = merge(t, new Treap(x));
122     }
123
124     while(m--) {
125         char s[15];
126         scanf("%s", s);
127
128         Treap *t1 = NULL, *tr = NULL, *t2 =
NULL;
129
130         if(!strcmp(s, "INSERT")) {
131             int p, k;
132             scanf("%d%d", &p, &k);
133             for(int i = 0; i < k; i++) {
134                 int x;
135                 scanf("%d", &x);
136                 t2 = merge(t2, new Treap(x)
);
137             }
138             split(t, p, t1, tr);
139             t = merge(t1, merge(t2, tr));
140         }
141
142         if(!strcmp(s, "DELETE")) {
143             int p, k;
144             scanf("%d%d", &p, &k);
145             split(t, p-1, t1, t);
146             split(t, k, t, tr);
147             del(t);
148             t = merge(t1, tr);
149         }
150
151         if(!strcmp(s, "MAKE-SAME")) {
152             int p, k, l;
153             scanf("%d%d%d", &p, &k, &l);
154             split(t, p-1, t1, t);
155             split(t, k, t, tr);
156             if(t) t->chg = l;
157             t = merge(t1, merge(t, tr));
158         }
159
160         if(!strcmp(s, "REVERSE")) {
161             int p, k;
162             scanf("%d%d", &p, &k);
163             split(t, p-1, t1, t);
164             split(t, k, t, tr);
165             if(t) t->rev ^= 1;
166             t = merge(t1, merge(t, tr));
167         }
168
169         if(!strcmp(s, "GET-SUM")) {
170             int p, k;
171             scanf("%d%d", &p, &k);
172             split(t, p-1, t1, t);
173             split(t, k, t, tr);

```

```

174         printf("%d\n", sum(t));
175         t = merge(tl, merge(t, tr));
176     }
177
178     if(!strcmp(s, "MAX-SUM")) {
179         printf("%d\n", mx_sum(t));
180     }
181 }
182
183 return 0;
184 }

```

6.5 Leftist Tree

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct Left{
5     Left *l,*r;
6     int v,h;
7     Left(int v_) : v(v_), h(1), l(0), r(0) {}
8 };
9
10 int height(Left *p)
11 {
12     return p ? p -> h : 0 ;
13 }
14
15 Left* combine(Left *a,Left *b)
16 {
17     if(!a || !b) return a ? a : b ;
18     Left *p ;
19     if( a->v > b->v)
20     {
21         p = a;
22         p -> r = combine( p -> r , b );
23     }
24     else
25     {
26         p = b;
27         p -> r = combine( p -> r , a );
28     }
29     if( height( p->l ) < height( p->r ) )
30         swap( p->l , p->r );
31     p->h = min( height( p->l ) , height( p->r ) ) + 1;
32     return p;
33 }
34 Left *root;
35
36 void push(int v)
37 {
38     //printf("push-%d\n",v);
39     Left *p = new Left(v);
40     root = combine( root , p );
41     //puts("end");
42 }
43 int top()
44 {
45     return root? root->v : -1;
46 }
47 void pop()
48 {

```

```

49     if(!root) return;
50     Left *a = root->l , *b = root->r ;
51     delete root;
52     root = combine( a , b );
53 }
54 void clear(Left* &p)
55 {
56     if(!p)
57         return;
58     if(p->l) clear(p->l);
59     if(p->r) clear(p->r);
60     delete p;
61     p = 0 ;
62 }
63
64
65 int main()
66 {
67     int T,n,x,o,size;
68     bool bst,bqu,bpq;
69     scanf("%d",&T);
70     while(T-->0)
71     {
72         bst=bqu=bpq=1;
73         stack<int> st;
74         queue<int> qu;
75         clear(root);
76         size=0;
77         scanf("%d",&n);
78         while(n-->0)
79         {
80             scanf("%d%d",&o,&x);
81             if(o==1)
82                 st.push(x),qu.push(x),push(x),size++;
83             else if(o==2)
84             {
85                 size--;
86                 if(size<0)
87                     bst=bqu=bpq=0;
88                 if(bst)
89                 {
90                     if(st.top()!=x)
91                         bst=0;
92                     st.pop();
93                 }
94                 if(bqu)
95                 {
96                     if(qu.front()!=x)
97                         bqu=0;
98                     qu.pop();
99                 }
100                 if(bpq)
101                 {
102                     // printf("(%d)\n",top());
103                     if(top()!=x)
104                         bpq=0;
105                     pop();
106                 }
107             }
108         }
109         int count=0;
110         if(bst)
111             count++;
112         if(bqu)

```

```

113     count++;
114     if(bpq)
115         count++;
116
117     if(count>1)
118         puts("not sure");
119     else if(count==0)
120         puts("impossible");
121     else if(bst)
122         puts("stack");
123     else if(bqu)
124         puts("queue");
125     else if(bpq)
126         puts("priority queue");
127 }
128 return 0;
129 }

```

7 geometry

7.1 Basic

```

1 // correct code of NPSC2013 senior-final pF
2
3 #include <bits/stdc++.h>
4 #define pb push_back
5 #define F first
6 #define S second
7 #define SZ(x) ((int)(x).size())
8 #define MP make_pair
9 using namespace std;
10 typedef long long ll;
11 typedef pair<int,int> PII;
12 typedef vector<int> VI;
13
14 typedef double dou;
15 struct PT{
16     dou x,y;
17     PT(dou x_=0.0,dou y_=0.0): x(x_),y(y_) {}
18     PT operator + (const PT &b) const {
19         return PT(x+b.x,y+b.y); }
20     PT operator - (const PT &b) const {
21         return PT(x-b.x,y-b.y); }
22     PT operator * (const dou &t) const {
23         return PT(x*t,y*t); }
24     dou operator * (const PT &b) const {
25         return x*b.x+y*b.y; }
26     dou operator % (const PT &b) const {
27         return x*b.y-b.x*y; }
28     dou len2() const { return x*x+y*y; }
29     dou len() const { return sqrt(len2()); }
30 };
31
32 const dou INF=1e12;
33 const dou eps=1e-8;
34 PT inter(const PT &P1,const PT &T1,const PT &P2,const PT &T2) // intersection
35 {
36     if(fabs(T1%T2)<eps)
37         return PT(INF,INF);
38     dou u=((P2-P1)%T2)/(T1%T2);
39     return P1+T1*u;

```

```

35 }
36
37 PT conv[500],cat,to;
38
39 int main()
40 {
41     int T,N,M;
42     scanf("%d",&T);
43     while(T--)
44     {
45         scanf("%d%d",&N,&M);
46         for(int i=0;i<N;i++)
47             scanf("%Lf%Lf",&conv[i].x,&conv[i].y)
48             ;
49         conv[N]=conv[0];
50         dou ans=0.0;
51         while(M--)
52         {
53             scanf("%Lf%Lf%Lf%Lf",&cat.x,&cat.y,&to.x,&to.y);
54             for(int i=0;i<N;i++)
55                 if(fabs((conv[i]-conv[i+1])%to)>eps)
56                 {
57                     // printf("M:%d i=%d\n",M,i);
58                     PT at=inter(conv[i],conv[i]-conv[i+1],cat,to);
59                     if((conv[i]-at)*(conv[i+1]-at)<eps && (at-cat)*to>-eps)
60                         ans=max(ans,(cat-at).len());
61                 }
62             printf("%.4f\n",ans);
63         }
64     }
65     return 0;

```

7.2 Smallest circle problem

```

1 #include <cstdlib>
2 #include <cstdio>
3 #include <algorithm>
4 #include <cmath>
5
6 // #define test
7
8 using namespace std;
9
10 const int N = 1000000 + 10;
11
12 struct PT {
13     double x, y;
14
15     PT() {}
16     PT(double x, double y):
17         x(x), y(y) {}
18     PT operator+(const PT &b) const {
19         return (PT) {x+b.x, y+b.y};
20     }
21     PT operator-(const PT &b) const {
22         return (PT) {x-b.x, y-b.y};
23     }
24     PT operator*(const double b) const {

```

```

25     return (PT) {x*b, y*b};
26 }
27 PT operator/(const double b) const {
28     return (PT) {x/b, y/b};
29 }
30 double operator%(const PT &b) const {
31     return x*b.y - y*b.x;
32 }
33
34 double len() const {
35     return sqrt(x*x + y*y);
36 }
37 PT T() const {
38     return (PT) {-y, x};
39 }
40 } p[N];
41
42 void update(PT a, PT b, PT c, PT &o, double
43     &r) {
44     if(c.x < 0.0) o = (a+b) / 2.0;
45     else {
46         PT p1 = (a+b)/2.0, p2 = p1 + (b-a).T();
47         PT p3 = (a+c)/2.0, p4 = p3 + (c-a).T();
48         double a123 = (p2-p1)*(p3-p1), a124 = (
49             p2-p1)*(p4-p1);
50         if(a123 * a124 > 0.0) a123 = -a123;
51         else a123 = abs(a123), a124 = abs(a124
52             );
53         o = (p4*a123 + p3*a124) / (a123 + a124)
54             ;
55     }
56     r = (a-o).len();
57 }
58
59 int main() {
60     //freopen("C:/Users/S11/Desktop/pb.in", "
61         r", stdin);
62
63     srand(7122);
64
65     int m, n;
66     while(scanf("%d%d", &m, &n)) {
67         if(!n && !m) return 0;
68
69         for(int i = 0; i < n; i++) scanf("%Lf%
70             Lf", &p[i].x, &p[i].y);
71
72         for(int i = 0; i < n; i++)
73             swap(p[i], p[rand() % (i+1)]);
74
75         PT a = p[0], b = p[1], c(-1.0, -1.0), o
76             = (a+b) / 2.0;
77         double r = (a-o).len();
78         for(int i = 2; i < n; i++) {
79             if((p[i]-o).len() <= r) continue;
80
81             a = p[i];
82             b = p[0];
83             c = (PT) {-1.0, -1.0};
84             update(a, b, c, o, r);
85
86             for(int k = 0; k < j; k++) {
87                 if((p[k]-o).len() <= r) continue;
88
89                 c = p[k];
90                 update(a, b, c, o, r);
91             }
92
93             #ifdef test
94             printf("i=%d\n", i);
95             printf("a=(%.1f, %.1f)\n", a.x, a.y);
96             printf("b=(%.1f, %.1f)\n", b.x, b.y);
97             printf("c=(%.1f, %.1f)\n", c.x, c.y);
98             printf("o=(%.1f, %.1f)\n", o.x, o.y);
99             printf("r=%.1f\n", r);
100             puts("-----");
101             #endif // test
102         }
103     }
104     printf("%.3f\n", r);
105 }

```