

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	Suffix Array($O(N \log N)$)	
4.4	Aho-Corasick	
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>
1 #define PB push_back
1 #define MP make_pair
1 #define F first
2 #define S second
3 #define SZ(x) ((int)(x).size())
3 #define ALL(x) (x).begin(),(x).end()
4 #ifdef _DEBUG_
4 #define debug(...) printf(__VA_ARGS__)
4 #else
5 #define debug(...) (void)0
6 #endif
7 using namespace std;
7 typedef long long ll;
8 typedef pair<int,int> PII;
8 typedef vector<int> VI;
8
10
11 int main() {
12     return 0;
13 }
14

```

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 bool isprime(ll n) {
52     if(n==1)
53         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     return ans;
92 }
93 }flow;
94 int main() {
95     return 0;
96 }

```

4 string

4.1 KMP

```

1  /**
2  Test OJ 265
3  trivial string matching
4
5  input:
6  abc
7  abccbabbabc
8
9  output:
10 0 8
11
12 */
13 #include <bits/stdc++.h>
14 #define PB push_back
15 #define F first
16 #define S second
17 #define SZ(x) ((int)(x).size())
18 #define MP make_pair
19 using namespace std;
20 typedef long long ll;
21 typedef pair<int,int> PII;
22 typedef vector<int> VI;
23
24 char S[500010],T[500010];

```

```

25 int K[500010];
26
27 int main()
28 {
29     gets(S);
30     gets(T);
31     K[0]=-1;
32     int a=-1;
33     for(int i=1;S[i];i++)
34     {
35         while(a!=-1 && S[a+1]!=S[i])
36             a=K[a];
37         if(S[a+1]==S[i])
38             a++;
39         K[i]=a;
40     }
41     VI ans;
42     a=-1;
43     for(int i=0;T[i];i++)
44     {
45         while(a!=-1 && S[a+1]!=T[i])
46             a=K[a];
47         if(S[a+1]==T[i])
48             a++;
49         if(!S[a+1])
50         {
51             ans.PB(i-a);
52             a=K[a];
53         }
54     }
55     bool first=1;
56     for(int u:ans)
57     {
58         if(first)
59             printf("%d",u),first=0;
60         else
61             printf(" %d",u);
62     }
63     puts("");
64     return 0;
65 }

```

4.2 Z-value

```

1  /**
2  Test OJ 265
3  trivial string matching
4
5  input:
6  abc
7  abccbabbabc
8
9  output:
10 0 8
11
12 */
13 #include <bits/stdc++.h>
14 #define pb push_back
15 #define F first
16 #define S second
17 #define SZ(x) ((int)(x).size())
18 #define MP make_pair
19 using namespace std;

```

```

20 typedef long long ll;
21 typedef pair<int,int> PII;
22 typedef vector<int> VI;
23
24 char S[1000010];
25 int Z[1000010];
26
27 int main()
28 {
29     int len=0,lenS;
30     gets(S);
31     for(;S[len];len++);
32     lenS=len;
33     gets(S+len+1);
34     for(len++;S[len];len++);
35     S[len]='*';
36     int bst=0;
37     Z[0]=0;
38     for(int i=1;i<len;i++)
39     {
40         if(Z[bst]+bst<i) Z[i]=0;
41         else Z[i]=min(Z[bst]+bst-i,Z[i-bst]);
42         while(S[Z[i]]==S[i+Z[i]]) Z[i]++;
43         if(Z[i]+i>Z[bst]+bst) bst=i;
44     }
45     bool first=1;
46     for(int i=lenS+1;i<len;i++)
47         if(Z[i]>=lenS)
48         {
49             if(first)
50                 printf("%d",i-lenS-1),first=0;
51             else
52                 printf(" %d",i-lenS-1);
53         }
54     puts("");
55     return 0;
56 }

```

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

```

1 // NTUJ448
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 const int SASIZE=2500000;
14 char in[500];
15 int S[SASIZE],from[SASIZE];
16 int R[SASIZE],SA[SASIZE],H[SASIZE];
17 int tR[SASIZE],tSA[SASIZE];
18 int cnt[SASIZE];
19 int num[4010];
20
21 int main()
22 {
23     int N;

```

```

24 while(scanf("%d",&N)==1 && N)
25 {
26     int len=0,maxR=0;
27     for(int i=0;i<N;i++)
28     {
29         scanf("%s",in);
30         for(int j=0;in[j];j++)
31         {
32             from[len]=i;
33             S[len++]=in[j]-'a';
34         }
35         from[len]=N;
36         S[len++]=i+50;
37     }
38     memset(R,-1,sizeof(R));
39     memset(cnt,0,sizeof(cnt));
40     for(int i=0;i<len;i++)
41     {
42         R[i]=S[i];
43         maxR=max(maxR,R[i]);
44     }
45     for(int i=0;i<len;i++)
46         cnt[R[i]+1]++;
47     for(int i=1;i<=maxR;i++)
48         cnt[i]+=cnt[i-1];
49     for(int i=0;i<len;i++)
50         SA[cnt[R[i]]++]=i;
51     /* for(int i=0;i<len;i++)
52         printf("R[%d]=%d, SA[%d]=%d\n",i,R[i],i,SA[i]); */
53     for(int i=1;i<len;i*=2)
54     {
55         memset(cnt,0,sizeof(int)*(maxR+10));
56         memcpy(tSA,SA,sizeof(int)*(len+10));
57         memcpy(tR,R,sizeof(int)*(len+10));
58         for(int j=0;j<len;j++)
59             cnt[R[j]+1]++;
60         for(int j=1;j<=maxR;j++)
61             cnt[j]+=cnt[j-1];
62         for(int j=len-i;j<len;j++)
63             SA[cnt[R[j]]++]=j;
64         for(int j=0;j<len;j++)
65         {
66             int k=tSA[j]-i;
67             if(k<0)
68                 continue;
69             SA[cnt[R[k]]++]=k;
70         }
71         int num=0;
72         maxR=0;
73         R[SA[0]]=num;
74         for(int j=1;j<len;j++)
75         {
76             if(tR[SA[j-1]]<tR[SA[j]] || tR[SA[j]-1]+i<tR[SA[j]+i])
77                 num++;
78             R[SA[j]]=num;
79             maxR=max(maxR,R[SA[j]]);
80         }
81         /* puts("-----");
82         for(int i=0;i<len;i++)
83             printf("R[%d]=%d, SA[%d]=%d\n",i,R[i],i,SA[i]); */
84     }
85     memset(H,0,sizeof(H));

```

```

86 for(int i=0;i<len;i++)
87 {
88     if(R[i]==0)
89         continue;
90     int &t=H[R[i]];
91     if(i>0)
92         t=max(0,H[R[i-1]]-1);
93     while(S[i+t]==S[SA[R[i]-1]+t]) t++;
94 }
95 /*for(int i=0;i<len;i++)
96     printf("R[%d]=%d, SA[%d]=%d\n",i,R[i],i,SA[i]);
97 for(int i=0;i<len;i++)
98     printf("%3d %3d %s\n",H[i],SA[i],S+SA[i]);*/
99 /*for(int i=0;i<len;i++)
100 {
101     printf("%3d %3d %d|",H[i],SA[i],from[i]);
102     for(int j=SA[i];j<len;j++)
103         printf("%2d ",S[j]);
104     puts("");
105 }*/
106 memset(num,0,sizeof(num));
107 int anslen=0,ansfrom=-1;
108 int get=0;
109 deque<PII> deq;
110 /* for(int i=0;i<len;i++)
111     printf("%d:%d\n",i,from[i]);*/
112 for(int l=0,r=0;r<len;r++)
113 {
114     if(from[SA[r]]<N && num[from[SA[r]]]==0)
115         get++;
116     num[from[SA[r]]]++;
117     while(deq.size()>0 && deq.back().F>=H[r]) deq.pop_back();
118     deq.pb(MP(H[r],r));
119     while(num[from[SA[l]]]>1)
120     {
121         num[from[SA[l]]]--;
122         l++;
123     }
124     while(deq.size()>0 && deq.front().S<=l) deq.pop_front();
125     if(get==N && deq.front().F>anslen)
126         anslen=deq.front().F, ansfrom=SA[l];
127 }
128 //printf("(%d)\n",anslen);
129 if(anslen==0)
130     puts("IDENTITY LOST");
131 else
132 {
133     for(int i=ansfrom;i<ansfrom+anslen;i++)
134         putchar(S[i]+'a');
135     puts("");
136 }
137 }
138 return 0;
139 }

```

4.4 Aho-Corasick

```

1 #include <cstdio>
2 #include <cstring>
3 #include <new>
4
5 struct Trie{
6     int c;
7     Trie *fail,*ch[52];
8     Trie():c(0){memset(ch,0,sizeof(ch));}
9 }trie[1000100];
10
11 char m[1010],f[100100];
12 Trie *str[1010],*na,*root;
13
14 inline int c_i(char a)
15 {
16     return (a>='A' && a<='Z') ? a-'A' : a-'a'+26;
17 }
18
19 void insert(char *s,int num)
20 {
21     Trie *at=root;
22     while(*s)
23     {
24         if(!at->ch[c_i(*s)])
25             at->ch[c_i(*s)]=new (na++) Trie();
26         at=at->ch[c_i(*s)],s++;
27     }
28     str[num]=at;
29 }
30
31 Trie *q[1000100];
32 int ql,qr;
33
34 void init()
35 {
36     ql=qr=-1;
37     q[++qr]=root;
38     root->fail=NULL;
39     while(ql<qr)
40     {
41         Trie *n=q[++ql],*f;
42         for(int i=0;i<52;i++)
43         {
44             if(!n->ch[i])
45                 continue;
46             f=n->fail;
47             while(f && !f->ch[i])
48                 f=f->fail;
49             n->ch[i]->fail=f?f->ch[i]:root;
50             q[++qr]=n->ch[i];
51         }
52     }
53 }
54
55 void go(char *s)
56 {
57     Trie*p=root;
58     while(*s)
59     {
60         while(p && !p->ch[c_i(*s)])
61             p=p->fail;

```

```

62     p=p?p->ch[c_i(*s)]:root;
63     p->fi=1;
64     s++;
65 }
66 }
67
68 void AC()
69 {
70     for(int i=qr;i>0;i--)
71         q[i]->fail->c+=q[i]->c;
72 }
73
74 int main()
75 {
76     int T,q;
77     scanf("%d",&T);
78     while(T--)
79     {
80         na=trie;
81         root=new (na++) Trie();
82         scanf("%s",f);
83         scanf("%d",&q);
84         for(int i=0;i<q;i++)
85         {
86             scanf("%s",m);
87             insert(m,i);
88         }
89         init();
90         go(f);
91         for(int i=0;i<q;i++)
92             puts(str[i]->fi?"y":"n");
93     }
94     return 0;
95 }

```

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         printf("%d\n",ans);
83     }
84     return 0;
85 }

```


6 data structure

6.1 Treap

```

1 #include <stdlib>
2 #include <stdio>
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) {
53     if(!a || !b) return a ? a : b;
54     else if(a->pri > b->pri) {
55         push(a);
56         a->r = merge(a->r, b);
57         pull(a);
58         return a;
59     }
60     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("%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         }
117     }
118     return 0;
119 }

```

6.2 copy on write treap


```

1 #include <stdlib>
2 #include <stdio>
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
69 void pull(Treap* t) {
70     t->sz = sz(t->l) + sz(t->r) + 1;
71 }
72
73 Treap* merge(Treap* a, Treap* b) {
74     if(!a || !b) {
75         Treap* t = a ? make(a) : make(b);
76         t->refs = 0;
77         takeRef(t->l);
78         takeRef(t->r);
79         return t;
80     }
81
82     Treap* t;
83     if( rnd(a->sz+b->sz) < a->sz ) {
84         t = make(a);
85         t->refs = 0;
86         t->r = merge(a->r, b);
87         takeRef(t->l);
88         takeRef(t->r);
89     }
90     else {
91         t = make(b);
92         t->refs = 0;
93         t->l = merge(a, b->l);
94         takeRef(t->l);
95         takeRef(t->r);
96     }
97
98     pull(t);
99     return t;
100 }
101
102 void split(Treap* t, int k, Treap* &a,
103 Treap* &b) {
104     if(!t) a = b = NULL;
105     else if(sz(t->l) < k) {
106         a = make(t);
107         a->refs = 0;
108         split(a->r, k-sz(t->l)-1, a->r, b);
109         takeRef(a->l);
110         takeRef(a->r);
111         pull(a);
112     }
113     else {
114         b = make(t);
115         b->refs = 0;
116         split(b->l, k, a, b->l);
117         takeRef(b->l);
118         takeRef(b->r);
119         pull(b);
120     }
121 }
122
123 void print_inorder(Treap* t) {
124     if(!t) return;
125     putchar(t->val);
126     print_inorder(t->l);
127     print_inorder(t->r);
128 }

```

```

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 }

```

6.3 copy on write segment tree

```

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
63     bool operator<(const Query b) const {
64         return c < b.c;
65     }
66 } qs[Q];
67
68 int arr[N];
69 Seg *t[N];
70 vector<int> vec2;
71
72 int main() {
73     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%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     t[0] = new (Seg::pmem++) Seg();
94     t[0] = t[0]->init(mn, mx);
95     int ptr = 0;
96     for(int i = 1; i <= n; i++) {
97         t[i] = t[i-1]->add(arr[i], mn, mx);
98     }
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 return 0;
116 }
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, rsum, mx_sum;
12     Treap *l, *r;
13
14     Treap() {}
15     Treap(int _val) : pri(rand()), sz(1), val(_val), chg(INF), rev(0), sum(_val), lsum(_val), rsum(_val), mx_sum(_val), l(NULL), r(NULL) {}
16 };
17
18 int sz(Treap* t) {return t ? t->sz : 0;}
19 int sum(Treap* t) {
20     if(!t) return 0;
21     if(t->chg == INF) return t->sum;
22     else return t->chg*t->sz;
23 }
24 int lsum(Treap* t) {
25     if(!t) return -INF;
26     if(t->chg != INF) return max(t->chg, (t->chg)*(t->sz));
27     if(t->rev) return t->rsum;
28     return t->lsum;
29 }
30 int rsum(Treap* t) {
31     if(!t) return -INF;
32     if(t->chg != INF) return max(t->chg, (t->chg)*(t->sz));
33     if(t->rev) return t->lsum;
34     return t->rsum;
35 }
36 int mx_sum(Treap* t) {
37     if(!t) return -INF;
38     if(t->chg != INF) return max(t->chg, (t->chg)*(t->sz));
39     return t->mx_sum;
40 }
41
42 void push(Treap* t) {
43     if(t->chg != INF) {
44         t->val = t->chg;
45         t->sum = (t->sz) * (t->chg);
46         t->lsum = t->rsum = t->mx_sum = max(t->sum, t->val);
47         if(t->l) t->l->chg = t->chg;
48         if(t->r) t->r->chg = t->chg;
49         t->chg = INF;
50     }
51     if(t->rev) {
52         swap(t->l, t->r);
53         if(t->l) t->l->rev ^= 1;
54         if(t->r) t->r->rev ^= 1;
55         t->rev = 0;
56     }
57 }
58
59 void pull(Treap* t) {
60

```

```

61     t->sz = sz(t->l)+sz(t->r)+1;          121
62     t->sum = sum(t->l)+sum(t->r)+t->val;    122
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);
66 }                                          129
67                                          130
68 Treap* merge(Treap* a, Treap* b) {       131
69     if(!a || !b) return a ? a : b;      132
70     if(a->pri > b->pri) {                  133
71         push(a);                          134
72         a->r = merge(a->r, b);              135
73         pull(a);                          136
74         return a;                          137
75     }                                     138
76     else {                                139
77         push(b);                          140
78         b->l = merge(a, b->l);              141
79         pull(b);                          142
80         return b;                          143
81     }                                     144
82 }                                          145
83                                          146
84 void split(Treap* t, int k, Treap* &a,   147
        Treap* &b) {                      148
85     if(!t) {                              149
86         a = b = NULL;                     150
87         return ;                          151
88     }                                     152
89     push(t);                              153
90     if(sz(t->l) < k) {                      154
91         a = t;                             155
92         push(a);                          156
93         split(t->r, k-sz(t->l)-1, a->r, b); 157
94         pull(a);                          158
95     }                                     159
96     else {                                160
97         b = t;                             161
98         push(b);                          162
99         split(t->l, k, a, b->l);            163
100        pull(b);                          164
101    }                                     165
102 }                                          166
103                                          167
104 void del(Treap* t) {                      168
105     if(!t) return;                       169
106     del(t->l);                             170
107     del(t->r);                             171
108     delete t;                             172
109 }                                          173
110                                          174
111 int main() {                              175
112     srand(7122);                          176
113                                          177
114     int n, m;                             178
115     scanf("%d%d", &n, &m);                179
116                                          180
117     Treap* t = NULL;                      181
118     for(int i = 0; i < n; i++) {           182
119         int x;                            183
120         scanf("%d", &x);

```

```

        t = merge(t, new Treap(x));
    }

    while(m--) {
        char s[15];
        scanf("%s", s);

        Treap *t1 = NULL, *tr = NULL, *t2 =
            NULL;

        if(!strcmp(s, "INSERT")) {
            int p, k;
            scanf("%d%d", &p, &k);
            for(int i = 0; i < k; i++) {
                int x;
                scanf("%d", &x);
                t2 = merge(t2, new Treap(x)
                    );
            }
            split(t, p, t1, tr);
            t = merge(t1, merge(t2, tr));
        }

        if(!strcmp(s, "DELETE")) {
            int p, k;
            scanf("%d%d", &p, &k);
            split(t, p-1, t1, t);
            split(t, k, t, tr);
            del(t);
            t = merge(t1, tr);
        }

        if(!strcmp(s, "MAKE-SAME")) {
            int p, k, l;
            scanf("%d%d%d", &p, &k, &l);
            split(t, p-1, t1, t);
            split(t, k, t, tr);
            if(t) t->chg = l;
            t = merge(t1, merge(t, tr));
        }

        if(!strcmp(s, "REVERSE")) {
            int p, k;
            scanf("%d%d", &p, &k);
            split(t, p-1, t1, t);
            split(t, k, t, tr);
            if(t) t->rev ^= 1;
            t = merge(t1, merge(t, tr));
        }

        if(!strcmp(s, "GET-SUM")) {
            int p, k;
            scanf("%d%d", &p, &k);
            split(t, p-1, t1, t);
            split(t, k, t, tr);
            printf("%d\n", sum(t));
            t = merge(t1, merge(t, tr));
        }

        if(!strcmp(s, "MAX-SUM")) {
            printf("%d\n", mx_sum(t));
        }
    }

    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;
40 }
41
42 PT conv[500],cat,to;
43
44 int main()
45 {
46     int T,N,M;
47     scanf("%d",&T);
48     while(T--)
49     {

```

```

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

```

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             for(int j = 1; j < i; j++) {
86                 if((p[j]-o).len() <= r) continue;
87
88                 b = p[j];
89                 c = (PT) {-1.0, -1.0};
90                 update(a, b, c, o, r);
91             }
92         }
93     }
94
95     #ifdef test
96     printf("i=%d\n", i);
97     printf("a=(%.1f, %.1f)\n", a.x, a.y);
98     printf("b=(%.1f, %.1f)\n", b.x, b.y);
99     printf("c=(%.1f, %.1f)\n", c.x, c.y);
100    printf("o=(%.1f, %.1f)\n", o.x, o.y);
101    printf("r=%.1f\n", r);
102    puts("-----");
103    #endif // test
104 }
105
106 printf("%.3f\n", r);
107 }

```