

Contents

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

1 Basic

1.1 default code

```

1 #include <bits/stdc++.h>
1 #define PB push_back
1 #define MP make_pair
2 #define F first
2 #define S second
2 #define SZ(x) ((int)(x).size())
2 #define ALL(x) (x).begin(),(x).end()
3 #ifdef _DEBUG_
3 #define debug(...) printf(__VA_ARGS__)
5 #else
5 #define debug(...) 0
5 #endif
6 using namespace std;
6 typedef long long ll;
7 typedef pair<int,int> PII;
9 typedef vector<int> VI;
10
11 int main() {
12
12     return 0;
13 }
21

```

1.2 .vimrc

```

1 color torte
2 syn on
3 set guifont=Consolas:h16:
4 set number
5 set showcmd
6 set autoindent
7 set smartindent
8 set tabstop=4
9 set showmatch
10 set comments=s1:/*,mb:\ *,eLx:\ */
11 set backspace=indent,eol,start
12 set softtabstop=4
13 set shiftwidth=4
14
15 map <F9> <ESC>:w<CR>:!g++ % -o %< -O2 -std=c++0x<CR>
16 map <S-F9> <ESC>:w<CR>:!g++ % -o %< -O2 -D_DEBUG_ -std=c++0x<CR>
17 map <F5> <ESC>:!./%<<CR>
18 map <F6> <ESC>:w<CR>ggvG"+y
19 map <S-F5> <ESC>:!./%< < %<.in<CR>
20 imap <Home> <ESC>^i
21 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 }

```

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,qr;
43   bool BFS() {
44     memset(d,-1,v*sizeof(int));
45     ql=qr=0;
46     qu[qr++]=0;
47     d[0]=0;
48     while(ql<qr && d[v-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[qr++]=eg[u].to;
56       }
57     }
58   }
59   return d[v-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 }flow;
92
93 int main() {
94
95   return 0;
96 }

```

4 string

4.1 KMP

```

1 /**
2  Test OJ 265
3  trivial string matching
4
5  input:
6  abc
7  abccbabbbabc
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 abccbababbc
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]='\0';
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+i+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 <stdio>
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'
17         +26;
18 }
19
20 void insert(char *s,int num)
21 {
22     Trie *at=root;
23     while(*s)
24     {
25         if(!at->ch[c_i(*s)])
26             at->ch[c_i(*s)]=new (na++) Trie();
27         at=at->ch[c_i(*s)],s++;
28     }
29     str[num]=at;
30 }
31
32 Trie *q[1000100];
33 int ql,qr;
34
35 void init()
36 {
37     ql=qr=-1;
38     q[++qr]=root;
39     root->fail=NULL;
40     while(ql<qr)
41     {
42         Trie *n=q[++ql],*f;
43         for(int i=0;i<52;i++)
44         {
45             if(!n->ch[i])
46                 continue;
47             f=n->fail;
48             while(f && !f->ch[i])
49                 f=f->fail;

```

```

49         n->ch[i]->fail=f?f->ch[i]:root;
50         q[++qr]=n->ch[i];
51     }
52 }
53
54 void go(char *s)
55 {
56     Trie*p=root;
57     while(*s)
58     {
59         while(p && !p->ch[c_i(*s)])
60             p=p->fail;
61         p=p?p->ch[c_i(*s)]:root;
62         p->fi=1;
63         s++;
64     }
65 }
66
67 void AC()
68 {
69     for(int i=qr;i>0;i--)
70         q[i]->fail->c+=q[i]->c;
71 }
72
73 int main()
74 {
75     int T,q;
76     scanf("%d",&T);
77     while(T--)
78     {
79         na=trie;
80         root=new (na++) Trie();
81         scanf("%s",f);
82         scanf("%d",&q);
83         for(int i=0;i<q;i++)
84         {
85             scanf("%s",m);
86             insert(m,i);
87         }
88         init();
89         go(f);
90         for(int i=0;i<q;i++)
91             puts(str[i]->fi?"y":"n");
92     }
93     return 0;
94 }
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++)

```

```

74         if(match[i]==-1)
75         {
76             fill(vis,vis+N,0);
77             if(DFS(i))
78                 ans++;
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) {

```

```

47 t->sum = sum(t->l) + sum(t->r) + t->val;
48 t->sz = sz(t->l) + sz(t->r) + 1;
49 }
50
51 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);

```

```

111         split(t, l - 1, tl, t);
112         split(t, r - l + 1, t, tr);
113         add(t, x);
114         t = merge(tl, merge(t, tr));
115     }
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
22 Treap* make(Treap* t) {
23     return new Treap(*t);
24 }
25
26 Treap* make(char _val) {
27     return new Treap(_val);
28 }
29
30 void print_ref(Treap* t) {
31     if(!t) return;
32     print_ref(t->l);
33     printf("%d ", t->refs);
34     print_ref(t->r);
35 }
36
37 void print(Treap* t) {
38     if(!t) return;
39     print(t->l);
40     putchar(t->val);
41     print(t->r);
42 }
43
44 void takeRef(Treap* t) {
45     if(t) t->refs++;
46 }
47
48 void dropRef(Treap* t) {
49     if(t) {
50         char c = t->val;
51         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);
119     }
120 }
121
122 void print_inorder(Treap* t) {
123     if(!t) return;
124     putchar(t->val);
125     print_inorder(t->l);
126     print_inorder(t->r);
127 }
128
129 char s[N];
130
131 int main() {
132     int m;
133     scanf("%d", &m);
134     scanf("%s", s);
135     int n = strlen(s);
136     int q;
137     scanf("%d", &q);
138
139     Treap* t = NULL;
140     for(int i = 0; i < n; i++) {
141         Treap *a = t, *b = make(s[i]);
142         t = merge(a, b);
143         dropRef(a);
144         dropRef(b);
145     }
146
147     while(q--) {
148         int l, r, x;
149         scanf("%d%d%d", &l, &r, &x);
150         r++;
151
152         Treap *a, *b, *c, *d;
153         a = b = c = d = NULL;
154         split(t, l, a, b);
155         dropRef(a);
156         split(b, r-l, c, d);
157         dropRef(b);
158         dropRef(d);
159         split(t, x, a, b);
160         dropRef(t);
161         Treap* t2 = merge(c, b);
162         dropRef(b);
163         dropRef(c);
164         t = merge(a, t2);
165         dropRef(a);
166         dropRef(t2);
167
168         if(t->sz > m) {
169             Treap* t2 = NULL;
170             split(t, m, t2, a);
171             dropRef(a);
172             dropRef(t);
173             t = t2;
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 | };

```

```

56 |
57 | struct Query {
58 |     int op, l, r, k, c, v;
59 |
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
83 |                 [i].l, &qs[i].r, &qs[i].k);
84 |             else scanf("%d%d", &qs[i].c, &qs[i].
85 |                 v);
86 |
87 |             if(qs[i].op == 2) vec2.push_back(qs[i
88 |                 ].v);
89 |         }
90 |         sort(vec2.begin(), vec2.end());
91 |         vec2.resize(unique(vec2.begin(), vec2.
92 |             end())-vec2.begin());
93 |         for(int i = 1; i <= n; i++) arr[i] =
94 |             lower_bound(vec2.begin(), vec2.end()
95 |                 , arr[i]) - vec2.begin();
96 |         int mn = 0, mx = vec2.size()-1;
97 |
98 |         for(int i = 0; i <= n; i++) t[i] = NULL
99 |             ;
100 |         t[0] = new (Seg::pmem++) Seg();
101 |         t[0] = t[0]->init(mn, mx);
102 |         int ptr = 0;
103 |         for(int i = 1; i <= n; i++) {
104 |             t[i] = t[i-1]->add(arr[i], mn, mx);
105 |         }
106 |         for(int i = 0; i < q; i++) {
107 |             int op = qs[i].op;
108 |             if(op == 1) {
109 |                 int l = qs[i].l, r = qs[i].r, k =
110 |                     qs[i].k;
111 |                 printf("%d\n", vec2[query(t[l-1], t
112 |                     [r], k, mn, mx)]);
113 |             }
114 |             if(op == 2) {
115 |                 continue;
116 |             }
117 |             if(op == 3) puts("7122");
118 |         }
119 |     }
120 | }

```

```

112     vec2.clear();
113     Seg::pmem = Seg::mem;
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
23 int sz(Treap* t) {return t ? t->sz : 0;}
24 int sum(Treap* t) {
25     if(!t) return 0;
26     if(t->chg == INF) return t->sum;
27     else return t->chg*t->sz;
28 }
29 int lsum(Treap* t) {
30     if(!t) return -INF;
31     if(t->chg != INF) return max(t->chg,
32         (t->chg)*(t->sz));
33     if(t->rev) return t->rsum;
34     return t->lsum;
35 }
36 int rsum(Treap* t) {
37     if(!t) return -INF;
38     if(t->chg != INF) return max(t->chg,
39         (t->chg)*(t->sz));
40     if(t->rev) return t->lsum;
41     return t->rsum;
42 }
43 int mx_sum(Treap* t) {
44     if(!t) return -INF;
45     if(t->chg != INF) return max(t->chg,
46         (t->chg)*(t->sz));
47     return t->mx_sum;
48 }
49 void push(Treap* t) {
50     if(t->chg != INF) {
51         t->val = t->chg;
52         t->sum = (t->sz) * (t->chg);
53     }
54 }
55 void pull(Treap* t) {
56     t->sz = sz(t->l)+sz(t->r)+1;
57     t->sum = sum(t->l)+sum(t->r)+t->val;
58     t->lsum = max(lsum(t->l), sum(t->l)+max(
59         0, lsum(t->r))+t->val);
60     t->rsum = max(rsum(t->r), sum(t->r)+max(
61         0, rsum(t->l))+t->val);
62     t->mx_sum = max(max(mx_sum(t->l),
63         mx_sum(t->r)), max(0, rsum(t->l))+
64         max(0, lsum(t->r))+t->val);
65 }
66 Treap* merge(Treap* a, Treap* b) {
67     if(!a || !b) return a ? a : b;
68     if(a->pri > b->pri) {
69         push(a);
70         a->r = merge(a->r, b);
71         pull(a);
72         return a;
73     }
74     else {
75         push(b);
76         b->l = merge(a, b->l);
77         pull(b);
78         return b;
79     }
80 }
81 void split(Treap* t, int k, Treap* &a,
82     Treap* &b) {
83     if(!t) {
84         a = b = NULL;
85         return;
86     }
87     push(t);
88     if(sz(t->l) < k) {
89         a = t;
90         push(a);
91         split(t->r, k-sz(t->l)-1, a->r, b);
92         pull(a);
93     }
94     else {
95         b = t;
96         push(b);
97         split(t->l, k, a, b->l);
98         pull(b);
99     }
100 }
101 void del(Treap* t) {
102     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(t1, 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--)
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--)
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
35         &P2,const PT &T2) // intersection

```

```

30 {
31     if(fabs(T1%T2)<eps)
32         return PT(INF,INF);
33     dou u=((P2-P1)%T2)/(T1%T2);
34     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,&
54                 to.x,&to.y);
55             for(int i=0;i<N;i++)
56                 if(fabs((conv[i]-conv[i+1])%to)>eps)
57                 {
58                     // printf("M:%d i=%d\n",M,i);
59                     PT at=inter(conv[i],conv[i]-conv[
60                         i+1],cat,to);
61                     if((conv[i]-at)*(conv[i+1]-at)<
62                         eps && (at-cat)*to>-eps)
63                         ans=max(ans,(cat-at).len());
64                 }
65             printf("%.4f\n",ans);
66         }
67     }
68     return 0;
69 }

```

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     }
87 }

```

```
78     for(int j = 1; j < i; j++) {
79         if((p[j]-o).len() <= r) continue;
80
81         b = p[j];
82         c = (PT) {-1.0, -1.0};
83         update(a, b, c, o, r);
84
85         for(int k = 0; k < j; k++) {
86             if((p[k]-o).len() <= r) continue;
87
88             c = p[k];
89             update(a, b, c, o, r);
90         }
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 }
106 }
```