

Contents

1	Basic
1.1	default code
1.2	.vimrc
2	math
2.1	ext gcd
3	flow
3.1	dinic
4	string
4.1	KMP
4.2	Z-value
4.3	Suffix Array($O(N\log N)$)
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+(H0J 92)
7	geometry

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(...) 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
20     return 0;
21 }
```

1.2 .vimrc

```

1 color torte
2 syn on
3 set guifont=Consolas:h16:
4 set number
5 set showcmd
6
7 " use indentation of previous line
8 set autoindent
9 " use intelligent indentation for C
10 set smartindent
11 " configure tabwidth and insert spaces
    instead of tabs
12 set tabstop=4          " tab width is 4
    spaces
13 set expandtab          " expand tabs to
    spaces
14 set showmatch
15 " intelligent comments
16 set comments=s1:/*,mb:\ *,elx:\ */
17 set backspace=indent,eol,start
18 set softtabstop=4
19 set shiftwidth=4
20
21 map <F9> <ESC>:w<CR>:!g++ % -o %< -O2 -std=
    c++0x<CR>
22 map <S-F9> <ESC>:w<CR>:!g++ % -o %< -O2 -
    D_DEBUG_ -std=c++0x<CR>
23 map <F5> <ESC>:!. /%<<CR>
24 map <F6> <ESC>:w<CR>ggvG"+y
25 map <S-F5> <ESC>:!. /%< < %<.in<CR>
26 imap <Home> <ESC>^i
27 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 ,
     maxflow=c
34     eg[ei]=E(b,c);
35     e[a].PB(ei);
36     ei++;
37     eg[ei]=E(a,0);
38     e[b].PB(ei);
39     ei++;
40   }
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]
         ].co==0)
70         continue;
71       if((temp=go(eg[u[i]].to,min(p,eg[u[i]
         ].co)))>0)
72         continue;
73       eg[u[i]].co-=temp;
74       eg[u[i]^1].co+=temp;
75       ptr[n]=i;
76       return temp;
77     }
78     ptr[n]=SZ(u);
79     return 0;
80   }
81   int max_flow() {
82     int ans=0,temp;
83     while(BFS()) {
84       for(int i=0;i<v;i++)
85         ptr[i]=0;
86       while((temp=go(0,INF))>0)
87         ans+=temp;
88     }
89     return ans;
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 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[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\n",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<= 39 {
125     1) deq.pop_front(); 40     if(match[u]==-1 || (!vis[match[u]]&&DFS
126     if(get==N && deq.front().F>anslen) (match[u]))
127     anslen=deq.front().F, ansfrom=SA[1 41     {
128     ]; 42         match[u]=x;
129 } 43         match[x]=u;
130 //printf("(%d)\n",anslen); 44         return 1;
131 if(anslen==0) 45     }
132 puts("IDENTITY LOST"); 46 }
133 else 47 return 0;
134 { 48 }
135 for(int i=ansfrom;i<ansfrom+anslen;i 49
136     ++); 50 int main()
137 putchar(S[i]+'a'); 51 {
138 puts(""); 52     int N;
139 } 53     while(scanf("%d",&N)==1)
140 } 54     {
141 return 0; 55         odd.clear();
142 } 56         even.clear();
143 } 57         for(int i=0;i<N;i++)
144 } 58             e[i].clear();
145 } 59         for(int i=0;i<N;i++)
146 } 60             {
147 } 61                 scanf("%d",in+i);
148 } 62                 if(in[i]%2==0)
149 } 63                     even.pb(i);
150 } 64                 else
151 } 65                     odd.pb(i);
152 } 66             }
153 } 67         for(int i:even)
154 } 68             for(int j:odd)
155 } 69                 if(is(1ll*in[i]*in[i]+1ll*in[j]*in[
156 } 70                     j]) && __gcd(in[i],in[j])==1)
157 } 71                     e[i].pb(j), e[j].pb(i);
158 } 72         int ans=0;
159 } 73         fill(match,match+N,-1);
160 } 74         for(int i=0;i<N;i++)
161 } 75             if(match[i]==-1)
162 } 76                 {
163 } 77                     fill(vis,vis+N,0);
164 } 78                     if(DFS(i))
165 } 79                         ans++;
166 } 80                 }
167 } 81             printf("%d\n",ans);
168 } 82         }
169 } 83     return 0;
170 }

```

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 <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) {
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 {
61         push(b);
62         b->l = merge(a, b->l);
63         pull(b);
64         return b;
65     }
66 }
67
68 void split(Treap* t, int k, Treap *&a,
69           Treap *&b) {
70     if(!t) a = b = NULL;
71     else if(sz(t->l) < k) {
72         a = t;
73         push(a);
74         split(t->r, k - sz(t->l) - 1, a->r, b);
75         pull(a);
76     }
77     else {
78         b = t;
79         push(b);
80         split(t->l, k, a, b->l);
81         pull(b);
82     }
83 }
84
85 int main() {
86     srand(105105);
87
88     int n, q;
89     scanf("%d%d", &n, &q);
90
91     Treap *t = NULL;
92     for(int i = 0; i < n; i++) {
93         ll tmp;
94         scanf("%lld", &tmp);
95         t = merge(t, make(tmp));
96     }
97
98     while(q--) {
99         char c;
100         int l, r;
101         scanf("\n%c %d %d", &c, &l, &r);
102
103         Treap *tl = NULL, *tr = NULL;
104         if(c == 'Q') {
105             split(t, l - 1, tl, t);
106             split(t, r - l + 1, t, tr);
107             printf("%lld\n", sum(t));
108             t = merge(tl, merge(t, tr));
109         }
110         else {
111             ll x;
112             scanf("%lld", &x);
113             split(t, l - 1, tl, t);
114             split(t, r - l + 1, t, tr);
115             add(t, x);
116             t = merge(tl, merge(t, tr));
117         }
118     }
119     return 0;
120 }

```

6.2 copy on write treap

```

1 #include <cstdlib>
2 #include <cstdio>
3 #include <algorithm>
4 #include <limits>
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     Treap() {}
16     Treap(char _val):
17         val(_val), sz(1), refs(0), l(NULL),
18         r(NULL) {}
19 };
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     }
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);

```



```

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 } 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     else scanf("%d%d", &qs[i].c, &qs[i]. 16         pri(rand()), sz(1), val(_val), chg(
      v);                                     INF), rev(0), sum(_val), lsum(
84                                           _val), rsum(_val), mx_sum(_val),
85     if(qs[i].op == 2) vec2.push_back(qs[i] 17     l(NULL), r(NULL) {});
      ].v);
86 }
87 sort(vec2.begin(), vec2.end());
88 vec2.resize(unique(vec2.begin(), vec2. 18
      end())-vec2.begin());
89 for(int i = 1; i <= n; i++) arr[i] = 19 int sz(Treap* t) {return t ? t->sz : 0;}
      lower_bound(vec2.begin(), vec2.end() 20 int sum(Treap* t) {
      , arr[i]) - vec2.begin();              21     if(!t) return 0;
90 int mn = 0, mx = vec2.size()-1;          22     if(t->chg == INF) return t->sum;
91                                           23     else return t->chg*t->sz;
92 for(int i = 0; i <= n; i++) t[i] = NULL 24 }
      ;
93 t[0] = new (Seg::pmem++) Seg();           25 int lsum(Treap* t) {
94 t[0] = t[0]->init(mn, mx);               26     if(!t) return -INF;
95 int ptr = 0;                             27     if(t->chg != INF) return max(t->chg,
96 for(int i = 1; i <= n; i++) {            28     (t->chg)*(t->sz));
97     t[i] = t[i-1]->add(arr[i], mn, mx);  29     if(t->rev) return t->rsum;
98 }                                         30     return t->lsum;
99                                           31 }
100 for(int i = 0; i < q; i++) {             32 int rsum(Treap* t) {
101     int op = qs[i].op;                   33     if(!t) return -INF;
102     if(op == 1) {                         34     if(t->chg != INF) return max(t->chg,
103         int l = qs[i].l, r = qs[i].r, k = 35     (t->chg)*(t->sz));
            qs[i].k;                       36     if(t->rev) return t->lsum;
104         printf("%d\n", vec2[query(t[l-1], t 37     return t->rsum;
            [r], k, mn, mx)]);
105     }
106     if(op == 2) {                         38 }
107         continue;
108     }
109     if(op == 3) puts("7122");
110 }
111 vec2.clear();
112 Seg::pmem = Seg::mem;
113 }
114 return 0;
115 }
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
20 int sum(Treap* t) {
21     if(!t) return 0;
22     if(t->chg == INF) return t->sum;
23     else return t->chg*t->sz;
24 }
25
26 int lsum(Treap* t) {
27     if(!t) return -INF;
28     if(t->chg != INF) return max(t->chg,
29     (t->chg)*(t->sz));
30     if(t->rev) return t->rsum;
31     return t->lsum;
32 }
33
34 int rsum(Treap* t) {
35     if(!t) return -INF;
36     if(t->chg != INF) return max(t->chg,
37     (t->chg)*(t->sz));
38     if(t->rev) return t->lsum;
39     return t->rsum;
40 }
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
49 void push(Treap* t) {
50     if(t->chg != INF) {
51         t->val = t->chg;
52         t->sum = (t->sz) * (t->chg);
53         t->lsum = t->rsum = t->mx_sum = max
54         (t->sum, t->val);
55         if(t->l) t->l->chg = t->chg;
56         if(t->r) t->r->chg = t->chg;
57         t->chg = INF;
58     }
59
60     if(t->rev) {
61         swap(t->l, t->r);
62         if(t->l) t->l->rev ^= 1;
63         if(t->r) t->r->rev ^= 1;
64         t->rev = 0;
65     }
66 }
67
68 void pull(Treap* t) {
69     t->sz = sz(t->l)+sz(t->r)+1;
70     t->sum = sum(t->l)+sum(t->r)+t->val;
71     t->lsum = max(lsum(t->l), sum(t->l)+max
72     (0, lsum(t->r))+t->val);
73     t->rsum = max(rsum(t->r), sum(t->r)+max
74     (0, rsum(t->l))+t->val);
75     t->mx_sum = max(max(mx_sum(t->l),
76     mx_sum(t->r)), max(0, rsum(t->l))+
77     max(0, lsum(t->r))+t->val);
78 }
79
80 Treap* merge(Treap* a, Treap* b) {
81     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,
85 Treap* &b) {
86     if(!t) {
87         a = b = NULL;
88         return ;
89     }
90     push(t);
91     if(sz(t->l) < k) {
92         a = t;
93         push(a);
94         split(t->r, k-sz(t->l)-1, a->r, b);
95         pull(a);
96     }
97     else {
98         b = t;
99         push(b);
100        split(t->l, k, a, b->l);
101        pull(b);
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
134             for(int i = 0; i < k; i++) {
135                 int x;
136                 scanf("%d", &x);
137                 t2 = merge(t2, new Treap(x)
138                     );
139             }
140             split(t, p, t1, tr);
141             t = merge(t1, merge(t2, tr));
142         }
143
144         if(!strcmp(s, "DELETE")) {
145             int p, k;
146             scanf("%d%d", &p, &k);
147             split(t, p-1, t1, t);
148             split(t, k, t, tr);
149             del(t);
150             t = merge(t1, tr);
151         }
152
153         if(!strcmp(s, "MAKE-SAME")) {
154             int p, k, l;
155             scanf("%d%d%d", &p, &k, &l);
156             split(t, p-1, t1, t);
157             split(t, k, t, tr);
158             if(t) t->chg = l;
159             t = merge(t1, merge(t, tr));
160         }
161
162         if(!strcmp(s, "REVERSE")) {
163             int p, k;
164             scanf("%d%d", &p, &k);
165             split(t, p-1, t1, t);
166             split(t, k, t, tr);
167             if(t) t->rev ^= 1;
168             t = merge(t1, merge(t, tr));
169         }
170
171         if(!strcmp(s, "GET-SUM")) {
172             int p, k;
173             scanf("%d%d", &p, &k);
174             split(t, p-1, t1, t);
175             split(t, k, t, tr);
176             printf("%d\n", sum(t));
177             t = merge(t1, merge(t, tr));
178         }
179
180         if(!strcmp(s, "MAX-SUM")) {
181             printf("%d\n", mx_sum(t));
182         }
183     }
184     return 0;
}

```

7 geometry