

# Algorithm Library

CRatiQ

South China Normal University

June 1, 2025

Contents

数学	2
Set Xor-Min . . . . .	2
数据结构	3
半群 deque . . . . .	3
区间众数 . . . . .	4
李超树 . . . . .	5
Splay . . . . .	6

# 数学

## Set Xor-Min

维护一个集合  $S$ , 可以求  $\min_{y \in S} (x \oplus y)$ 。

```
1  struct SetXorMin
2  {
3      static constexpr int L=30;
4      int tot=0;
5      vector<array<int,2>> c;
6      vector<int> s;
7      set<i64> in;
8
9      SetXorMin() {}
10     SetXorMin(int n)
11     {
12         c.resize((n+1)*(L+1));
13         s.resize((n+1)*(L+1));
14     }
15
16     void insert(i64 x)
17     {
18         if (in.count(x))
19             return;
20         in.insert(x);
21         int p=0;
22         for (int i=L;i>=0;i--)
23         {
24             bool o=x>>i&1;
25             if (!c[p][o])
26                 c[p][o]=++tot;
27             s[p=c[p][o]]++;
28         }
29     }
30
31     void erase(i64 x)
32     {
33         if (!in.count(x))
34             return;
35         in.erase(x);
36         int p=0;
37         for (int i=L;i>=0;i--)
38         {
39             bool o=x>>i&1;
40             s[p=c[p][o]]--;
41         }
42     }
43
44     i64 QueryXorMin(i64 x)
45     {
46         int p=0;
47         i64 r=0;
48         for (int i=L;i>=0;i--)
49         {
50             bool o=x>>i&1;
51             if (s[c[p][o]])
52                 p=c[p][o];
53             else
54             {
55                 p=c[p][o^1];
56                 r|=1ll<<i;
57             }
58         }
59         return r;
60     }
61 };
```

# 数据结构

## 半群 deque

维护一个半群的 deque，支持前后增删及求和。

```
1  template <class T>
2  struct SWAG
3  {
4      vector<T> l,sl,r,sr;
5
6      void push_front(const T &o)
7      {
8          sl.push_back(sl.empty()?o:o+sl.back());
9          l.push_back(o);
10     }
11
12     void push_back(const T &o)
13     {
14         sr.push_back(sr.empty()?o:sr.back()+o);
15         r.push_back(o);
16     }
17
18     void pop_front()
19     {
20         if (!l.empty())
21         {
22             l.pop_back();
23             sl.pop_back();
24             return;
25         }
26         int n=r.size(),m;
27         if (m=n-1>>1)
28         {
29             l.resize(m);
30             sl.resize(m);
31             for (int i=1;i<=m;i++)
32                 l[m-i]=r[i];
33             sl[0]=l[0];
34             for (int i=1;i<m;i++)
35                 sl[i]=l[i]+sl[i-1];
36         }
37         for (int i=m+1;i<n;i++)
38             r[i-(m+1)]=r[i];
39         m=n-(m+1);
40         r.resize(m);
41         sr.resize(m);
42         if (m)
43         {
44             sr[0]=r[0];
45             for (int i=1;i<m;i++)
46                 sr[i]=sr[i-1]+r[i];
47         }
48     }
49
50     void pop_back()
51     {
52         if (!r.empty())
53         {
54             r.pop_back();
55             sr.pop_back();
56         }
57         else
58         {
59             int n=l.size(),m;
60             if (m=n-1>>1)
61             {
62                 r.resize(m);
63                 sr.resize(m);
64                 for (int i=1;i<=m;i++)
65                     r[m-i]=l[i];
```

```

66         sr[0]=r[0];
67         for (int i=1;i<m;i++)
68             sr[i]=sr[i-1]+r[i];
69     }
70     for (int i=m+1;i<n;i++)
71         l[i-(m+1)]=l[i];
72     m=n-(m+1);
73     l.resize(m);
74     sl.resize(m);
75     if (m)
76     {
77         sl[0]=l[0];
78         for (int i=1;i<m;i++)
79             sl[i]=l[i]+sl[i-1];
80     }
81 }
82 }
83
84 T ask()
85 {
86     assert(l.size()||r.size());
87     if (l.size()&&r.size())
88         return sl.back()+sr.back();
89     return l.size()?sl.back():sr.back();
90 }
91 };
92
93 struct Info
94 {
95     Z k,b;
96
97     Info operator + (const Info &o) const
98     {
99         return {k*o.k,b*o.k+o.b};
100     }
101 };
102
103 Z operator + (const Z &x,const Info &o)
104 {
105     return o.k*x+o.b;
106 }

```

## 区间众数

```

1  template <class T>
2  struct Mode
3  {
4      int n,ksz,m;
5      vector<T> b;
6      vector<vector<int>> pos,f;
7      vector<int> a,blk,id,l;
8
9      Mode(const vector<T> &c):n(c.size()),ksz(max<int>(1,sqrt(n))),
10         m((n+ksz-1)/ksz),b(c),pos(n),f(m,vector<int>(m)),a(n),blk(n),id(n),l(m+1)
11  {
12      sort(b.begin(),b.end());
13      b.erase(unique(b.begin(),b.end()),b.end());
14      for (int i=0;i<n;i++)
15      {
16          a[i]=lower_bound(b.begin(),b.end(),c[i])-b.begin();
17          id[i]=pos[a[i]].size();
18          pos[a[i]].push_back(i);
19      }
20      for (int i=0;i<n;i++)
21          blk[i]=i/ksz;
22      for (int i=0;i<=m;i++)
23          l[i]=min(i*ksz,n);
24
25      vector<int> cnt(b.size());
26      for (int i=0;i<m;i++)
27      {

```

```

28         cnt.assign(b.size(),0);
29         pair<int,int> cur={0,0};
30         for (int j=i;j<m;j++)
31         {
32             for (int k=l[j];k<l[j+1];k++)
33                 cur=max(cur,{++cnt[a[k]],a[k]});
34             f[i][j]=cur.second;
35         }
36     }
37 }
38
39 pair<T,int> ask(int L,int R)
40 {
41     int val=blk[L]==blk[R-1]?0:f[blk[L]+1][blk[R-1]-1],i;
42     int cnt=lower_bound(pos[val].begin(),pos[val].end(),R)-
43         lower_bound(pos[val].begin(),pos[val].end(),L);
44     for (int i=min(R,l[blk[L]+1])-1;i>=L;i--)
45     {
46         auto &v=pos[a[i]];
47         while (id[i]+cnt<v.size()&&v[id[i]+cnt]<R)
48             cnt++,val=a[i];
49         if (a[i]>val&&id[i]+cnt-1<v.size()&&v[id[i]+cnt-1]<R)
50             val=a[i];
51     }
52     for (int i=max(L,l[blk[R-1]]);i<R;i++)
53     {
54         auto &v=pos[a[i]];
55         while (id[i]>=cnt&&v[id[i]-cnt]>=L)
56             cnt++,val=a[i];
57         if (a[i]>val&&id[i]>=cnt-1&&v[id[i]-cnt+1]>=L)
58             val=a[i];
59     }
60     return {b[val],cnt};
61 }
62 };

```

## 李超树

```

1  constexpr i64 inf=9e18;
2
3  template <class Info>
4  struct SGT
5  {
6      int cnt=0;
7      vector<Info> a;
8      vector<int> ls,rs;
9      i64 z,y,L,R;
10
11      SGT(int n,i64 l,i64 r)
12      {
13          int N=(n+7)*64;
14          a.resize(N);
15          ls.resize(N);
16          rs.resize(N);
17          L=l,R=r,cnt=1;
18          a[1]={0,inf};
19      }
20
21  private:
22      void insert(int &p,i64 l,i64 r,Info v)
23      {
24          if (!p)
25          {
26              p=++cnt;
27              a[p]={0,inf};
28          }
29          i64 m=(l+r)>>1;
30          if (z<=l&&r<=y)
31          {
32              if (a[p].y(m)>v.y(m)) swap(a[p],v);
33              if (a[p].y(l)>v.y(l)) insert(ls[p],l,m,v);

```

```

34         else if (a[p].y(r)>v.y(r)) insert(rs[p],m+1,r,v);
35         return;
36     }
37     if (z<=m) insert(ls[p],l,m,v);
38     if (y>m) insert(rs[p],m+1,r,v);
39 }
40 public:
41     void insert(i64 l,i64 r,const Info &v)
42     {
43         z=l,y=r;
44         int p=1;
45         insert(p,L,R,v);
46     }
47
48     i64 QueryMin(i64 p)
49     {
50         i64 res=a[1].y(p),l=L,r=R,x=1;
51         while (l<r)
52         {
53             i64 m=(l+r)>>1;
54             if (p<=m)
55                 x=ls[x],r=m;
56             else
57                 x=rs[x],l=m+1;
58             if (!x) return res;
59             res=min(res,a[x].y(p));
60         }
61         return res;
62     }
63 };
64
65 struct Info
66 {
67     i64 k,b;
68
69     i64 y(const i64 &x) const { return k*x+b; }
70 };

```

## Splay

```

1  template <class Info,class Tag>
2  struct Splay
3  {
4      #define _rev
5      struct Node
6      {
7          Node *c[2],*f;
8          int siz;
9          Info s,v;
10         Tag t;
11
12         Node():c{},f(0),siz(1),s(),v(),t() {}
13         Node(Info x):c{},f(0),siz(1),s(x),v(x),t() {}
14
15         void operator += (const Tag &o)
16         {
17             s+=o,v+=o,t+=o;
18         }
19         #ifdef _rev
20         void swap() { swap(c[0],c[1]); }
21         #endif
22
23         void pushup()
24         {
25             if (c[0])
26                 s=c[0]->s+v,siz=c[0]->siz+1;
27             else s=v,siz=1;
28             if (c[1])
29                 s=s+c[1]->s,siz+=c[1]->siz;
30         }
31     }

```

```

32     void pushdown()
33     {
34         for (auto x:c)
35             if (x)
36                 *x+=t;
37         t=Tag();
38     }
39
40     void zigzag()
41     {
42         Node *y=f,*z=y->f;
43         bool isl=y->c[0]==this;
44         if (z) z->c[z->c[1]==y]=this;
45         f=z,y->f=this;
46         y->c[isl^1]=c[isl];
47         if (c[isl]) c[isl]->f=y;
48         c[isl]=y;
49         y->pushup();
50     }
51
52     //only used for makeroot
53     void splay(Node *tg)
54     {
55         for (Node *y=f;y!=tg;zigzag(),y=f)
56             if (Node *z=y->f;z!=tg)
57                 (z->c[1]==y^y->c[1]==this?this:y)->zigzag();
58         pushup();
59     }
60
61     void clear()
62     {
63         for (Node *x:c)
64             if (x)
65                 x->clear();
66         delete this;
67     }
68 };
69
70 Node *rt;
71 int shift;
72
73 Splay()
74 {
75     rt=new Node;
76     rt->c[1]=new Node;
77     rt->c[1]->f=rt;
78     rt->siz=2;
79 }
80
81 Splay(vector<Info> &a,int l,int r)
82 {
83     shift=l-1;
84     rt=new Node;
85     rt->c[1]=new Node;
86     rt->c[1]->f=rt;
87     if (l<r)
88     {
89         rt->c[1]->c[0]=build(a,l,r);
90         rt->c[1]->c[0]->f=rt->c[1];
91     }
92     rt->c[1]->pushup();
93     rt->pushup();
94 }
95
96 Node *build(vector<Info> &a,int l,int r)
97 {
98     if (l==r) return 0;
99     int m=(l+r)>>1;
100     Node *x=new Node(a[m]);
101     x->c[0]=build(a,l,m);
102     x->c[1]=build(a,m+1,r);

```



```

103     for (Node *y:x->c)
104         if (y) y->f=x;
105     x->pushup();
106     return x;
107 }
108
109 void makeroot(Node *u,Node *tg)
110 {
111     if (!tg) rt=u;
112     u->splay();
113 }
114
115 void findKth(int k,Node *tg)
116 {
117     Node *x=rt;
118     while (1)
119     {
120         x->pushdown();
121         int res=x->c[0]?x->c[0]->siz:0;
122         if (res+1==k)
123         {
124             x->splay(tg);
125             if (!tg) rt=x;
126             return;
127         }
128         if (res>=k) x=x->c[0];
129         else x=x->c[1],k-=res+1;
130     }
131 }
132
133 void split(int l,int r)
134 {
135     findKth(l,0);
136     findKth(r+2,rt);
137 }
138
139 #ifdef _rev
140 void reverse(int l,int r)
141 {
142     l-=shift;
143     r-=shift+1;
144     if (l>r) return;
145     split(l,r);
146     *(rt->c[1]->c[0])+=Tag(1);
147 }
148 #endif
149
150 //insert before pos
151 void insert(int pos,Info x)
152 {
153     pos-=shift;
154     split(pos,pos-1);
155     rt->c[1]->c[0]=new Node(x);
156     rt->c[1]->c[0]->f=rt->c[1];
157     rt->c[1]->pushup();
158     rt->pushup();
159 }
160
161 void insert(int pos,vector<Info> &a,int l,int r)
162 {
163     pos-=shift;
164     split(pos,pos-1);
165     rt->c[1]->c[0]=build(a,l,r);
166     rt->c[1]->c[0]->f=rt->c[1];
167     rt->c[1]->pushup();
168     rt->pushup();
169 }
170
171 void erase(int pos)
172 {
173     pos-=shift;

```

```

174     split(pos,pos);
175     delete rt->c[1]->c[0];
176     rt->c[1]->c[0]=0;
177     rt->c[1]->pushup();
178     rt->pushup();
179 }
180
181 void erase(int l,int r)
182 {
183     l-=shift,r-=shift+1;
184     if (l>r) return;
185     split(l,r);
186     rt->c[1]->c[0]->clear();
187     rt->c[1]->c[0]=0;
188     rt->c[1]->pushup();
189     rt->pushup();
190 }
191
192 void modify(int pos,Info x)
193 {
194     pos-=shift;
195     findKth(pos+1,0);
196     rt->v=x;
197     rt->pushup();
198 }
199
200 void rangeApply(int l,int r,Tag w)
201 {
202     l-=shift,r-=shift+1;
203     if (l>r) return;
204     split(l,r);
205     Node *x=rt->c[1]->c[0];
206     *x+=w;
207     rt->c[1]->pushup();
208     rt->pushup();
209 }
210
211 Info rangeQuery(int l,int r)
212 {
213     l-=shift,r-=shift+1;
214     split(l,r);
215     return rt->c[1]->c[0]->s;
216 }
217
218 ~Splay() { rt->clear(); }
219 #undef _rev
220 };
221
222 struct Tag
223 {
224     bool rev=0;
225
226     Tag() {}
227     Tag(bool c):rev(c) {}
228
229     void operator += (const Tag &o)
230     {
231         rev^=o.rev;
232     }
233 };
234
235 struct Info
236 {
237     i64 x=0;
238
239     void operator += (const Tag &o) const
240     {
241
242     }
243
244     Info operator + (const Info &o) const

```

```
245     {  
246         return {x+o.x};  
247     }  
248 };
```