Algorithm Library

CReatiQ

South China Normal University

June 1, 2025

Contents

数学

Set Xor-Min

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

```
struct SetXorMin
{
    static constexpr int L=30;
    int tot=0;
    vector<array<int,2>> c;
    vector<int> s;
    set<i64> in;

    SetXorMin() {}
    SetXorMin(int n)
    {
        c.resize((n+1)*(L+1));
        s.resize((n+1)*(L+1));
    }

    void insert(i64 x)
    {
        if (in.count(x))
            return;
        in.insert(x);
        int p=0;
        for (int i=L;i>=0;i--)
        {
            bool o=x>>i&1;
            if (!c[p][o])
                c[p][o]=++tot;
            s[p=c[p][o]]++;
        }
    }

    void erase(i64 x)
    {
        if (!in.count(x))
            return;
        in.erase(x);
        int p=0;
        for (int i=L;i>=0;i--)
        {
            bool o=x>>i&1;
            s[p=c[p][o]]--;
        }
    }

    i64 QueryXorMin(i64 x)
    {
        int p=0;
        i64 r=0;
        for (int i=L;i>=0;i--)
        {
            bool o=x>>i&1;
            if (s[c[p][o]])
                p=c[p][o];
            else
            {
                p=c[p][o^1];
                r|=1ll<<i;
            }
        }
        return r;
    }
};
```

数据结构

半群 deque

维护一个半群的 deque，支持前后增删及求和。

```cpp
template <class T>
struct SWAG
{
    vector<T> l,sl,r,sr;

    void push_front(const T &o)
    {
        sl.push_back(sl.empty()?o:o+sl.back());
        l.push_back(o);
    }

    void push_back(const T &o)
    {
        sr.push_back(sr.empty()?o:sr.back()+o);
        r.push_back(o);
    }

    void pop_front()
    {
        if (!l.empty())
        {
            l.pop_back();
            sl.pop_back();
            return;
        }
        int n=r.size(),m;
        if (m=n-1>>1)
        {
            l.resize(m);
            sl.resize(m);
            for (int i=1;i<=m;i++)
                l[m-i]=r[i];
            sl[0]=l[0];
            for (int i=1;i<m;i++)
                sl[i]=l[i]+sl[i-1];
        }
        for (int i=m+1;i<n;i++)
            r[i-(m+1)]=r[i];
        m=n-(m+1);
        r.resize(m);
        sr.resize(m);
        if (m)
        {
            sr[0]=r[0];
            for (int i=1;i<m;i++)
                sr[i]=sr[i-1]+r[i];
        }
    }

    void pop_back()
    {
        if (!r.empty())
        {
            r.pop_back();
            sr.pop_back();
        }
        else
        {
            int n=l.size(),m;
            if (m=n-1>>1)
            {
                r.resize(m);
                sr.resize(m);
                for (int i=1;i<=m;i++)
                    r[m-i]=l[i];
                sr[0]=r[0];
                for (int i=1;i<m;i++)
                    sr[i]=sr[i-1]+r[i];
            }
            for (int i=m+1;i<n;i++)
                l[i-(m+1)]=l[i];
            m=n-(m+1);
            l.resize(m);
            sl.resize(m);
            if (m)
            {
                sl[0]=l[0];
                for (int i=1;i<m;i++)
                    sl[i]=l[i]+sl[i-1];
            }
        }
    }

    T ask()
    {
        assert(l.size()||r.size());
        if (l.size()&&r.size())
            return sl.back()+sr.back();
        return l.size()?sl.back():sr.back();
    }
};

struct Info
{
    Z k,b;

    Info operator + (const Info &o) const
    {
        return {k*o.k,b*o.k+o.b};
    }
};

Z operator + (const Z &x,const Info &o)
{
    return o.k*x+o.b;
}
```

区间众数

```
template <class T>
struct Mode
{
    int n,ksz,m;
    vector<T> b;
    vector<vector<int>> pos,f;
    vector<int> a,blk,id,l;

    Mode(const vector<T> &c):n(c.size()),ksz(max<int>(1,sqrt(n))),
        m((n+ksz-1)/ksz),b(c),pos(n),f(m,vector<int>(m)),a(n),blk(n),id(n),l(m+1)
    {
        sort(b.begin(),b.end());
        b.erase(unique(b.begin(),b.end()),b.end());
        for (int i=0;i<n;i++)
        {
            a[i]=lower_bound(b.begin(),b.end(),c[i])-b.begin();
            id[i]=pos[a[i]].size();
            pos[a[i]].push_back(i);
        }
        for (int i=0;i<n;i++)
            blk[i]=i/ksz;
        for (int i=0;i<=m;i++)
            l[i]=min(i*ksz,n);

        vector<int> cnt(b.size());
        for (int i=0;i<m;i++)
        {
```

```cpp
            cnt.assign(b.size(),0);
            pair<int,int> cur={0,0};
            for (int j=i;j<m;j++)
            {
                for (int k=l[j];k<l[j+1];k++)
                    cur=max(cur,{++cnt[a[k]],a[k]});
                f[i][j]=cur.second;
            }
        }
    }

    pair<T,int> ask(int L,int R)
    {
        int val=blk[L]==blk[R-1]?0:f[blk[L]+1][blk[R-1]-1],i;
        int cnt=lower_bound(pos[val].begin(),pos[val].end(),R)-
                lower_bound(pos[val].begin(),pos[val].end(),L);
        for (int i=min(R,l[blk[L]+1])-1;i>=L;i--)
        {
            auto &v=pos[a[i]];
            while (id[i]+cnt<v.size()&&v[id[i]+cnt]<R)
                cnt++,val=a[i];
            if (a[i]>val&&id[i]+cnt-1<v.size()&&v[id[i]+cnt-1]<R)
                val=a[i];
        }
        for (int i=max(L,l[blk[R-1]]);i<R;i++)
        {
            auto &v=pos[a[i]];
            while (id[i]>=cnt&&v[id[i]-cnt]>=L)
                cnt++,val=a[i];
            if (a[i]>val&&id[i]>=cnt-1&&v[id[i]-cnt+1]>=L)
                val=a[i];
        }
        return {b[val],cnt};
    }
};
```

李超树

```cpp
constexpr i64 inf=9e18;

template <class Info>
struct SGT
{
    int cnt=0;
    vector<Info> a;
    vector<int> ls,rs;
    i64 z,y,L,R;

    SGT(int n,i64 l,i64 r)
    {
        int N=(n+7)*64;
        a.resize(N);
        ls.resize(N);
        rs.resize(N);
        L=l,R=r,cnt=1;
        a[1]={0,inf};
    }

private:
    void insert(int &p,i64 l,i64 r,Info v)
    {
        if (!p)
        {
            p=++cnt;
            a[p]={0,inf};
        }
        i64 m=(l+r)>>1;
        if (z<=l&&r<=y)
        {
            if (a[p].y(m)>v.y(m)) swap(a[p],v);
            if (a[p].y(l)>v.y(l)) insert(ls[p],l,m,v);
            else if (a[p].y(r)>v.y(r)) insert(rs[p],m+1,r,v);
            return;
        }
        if (z<=m) insert(ls[p],l,m,v);
        if (y>m) insert(rs[p],m+1,r,v);
    }
public:
    void insert(i64 l,i64 r,const Info &v)
    {
        z=l,y=r;
        int p=1;
        insert(p,L,R,v);
    }

    i64 QueryMin(i64 p)
    {
        i64 res=a[1].y(p),l=L,r=R,x=1;
        while (l<r)
        {
            i64 m=(l+r)>>1;
            if (p<=m)
                x=ls[x],r=m;
            else
                x=rs[x],l=m+1;
            if (!x) return res;
            res=min(res,a[x].y(p));
        }
        return res;
    }
};

struct Info
{
    i64 k,b;

    i64 y(const i64 &x) const { return k*x+b; }
};
```

Splay

```
template <class Info,class Tag>
struct Splay
{
#define _rev
    struct Node
    {
        Node *c[2],*f;
        int siz;
        Info s,v;
        Tag t;

        Node():c{},f(0),siz(1),s(),v(),t() {}
        Node(Info x):c{},f(0),siz(1),s(x),v(x),t() {}

        void operator += (const Tag &o)
        {
            s+=o,v+=o,t+=o;
#ifdef _rev
            if (o.rev) swap(c[0],c[1]);
#endif
        }

        void pushup()
        {
            if (c[0])
                s=c[0]->s+v,siz=c[0]->siz+1;
            else s=v,siz=1;
            if (c[1])
                s=s+c[1]->s,siz+=c[1]->siz;
        }

        void pushdown()
        {
            for (auto x:c)
                if (x)
                    *x+=t;
            t=Tag();
        }

        void zigzag()
        {
            Node *y=f,*z=y->f;
            bool isl=y->c[0]==this;
            if (z) z->c[z->c[1]==y]=this;
            f=z,y->f=this;
            y->c[isl^1]=c[isl];
            if (c[isl]) c[isl]->f=y;
            c[isl]=y;
            y->pushup();
        }

        //only used for makeroot
        void splay(Node *tg)
        {
            for (Node *y=f;y!=tg;zigzag(),y=f)
                if (Node *z=y->f;z!=tg)
                    (z->c[1]==y^y->c[1]==this?this:y)->zigzag();
            pushup();
        }

        void clear()
        {
            for (Node *x:c)
                if (x)
                    x->clear();
            delete this;
        }
    };

    Node *rt;
    int shift;

    Splay()
    {
        rt=new Node;
        rt->c[1]=new Node;
        rt->c[1]->f=rt;
        rt->siz=2;
    }

    Splay(vector<Info> &a,int l,int r)
    {
        shift=l-1;
        rt=new Node;
        rt->c[1]=new Node;
        rt->c[1]->f=rt;
        if (l<r)
        {
            rt->c[1]->c[0]=build(a,l,r);
            rt->c[1]->c[0]->f=rt->c[1];
        }
        rt->c[1]->pushup();
        rt->pushup();
    }

    Node *build(vector<Info> &a,int l,int r)
    {
        if (l==r) return 0;
        int m=(l+r)>>1;
        Node *x=new Node(a[m]);
        x->c[0]=build(a,l,m);
        x->c[1]=build(a,m+1,r);
        for (Node *y:x->c)
            if (y) y->f=x;
        x->pushup();
        return x;
    }

    void makeroot(Node *u,Node *tg)
    {
        if (!tg) rt=u;
        u->splay();
    }

    void findKth(int k,Node *tg)
    {
        Node *x=rt;
        while (1)
        {
            x->pushdown();
            int res=x->c[0]?x->c[0]->siz:0;
            if (res+1==k)
            {
                x->splay(tg);
                if (!tg) rt=x;
                return;
            }
            if (res>=k) x=x->c[0];
            else x=x->c[1],k-=res+1;
        }
    }

    void split(int l,int r)
    {
        findKth(l,0);
        findKth(r+2,rt);
    }

#ifdef _rev
    void reverse(int l,int r)
    {
        l-=shift;
        r-=shift+1;
        if (l>r) return;
        split(l,r);
        *(rt->c[1]->c[0])+=Tag(1);
    }
#endif

    //insert before pos
    void insert(int pos,Info x)
    {
        pos-=shift;
        split(pos,pos-1);
        rt->c[1]->c[0]=new Node(x);
        rt->c[1]->c[0]->f=rt->c[1];
        rt->c[1]->pushup();
        rt->pushup();
    }

    void insert(int pos,vector<Info> &a,int l,int r)
    {
        pos-=shift;
        split(pos,pos-1);
        rt->c[1]->c[0]=build(a,l,r);
        rt->c[1]->c[0]->f=rt->c[1];
        rt->c[1]->pushup();
        rt->pushup();
    }

    void erase(int pos)
    {
        pos-=shift;
        split(pos,pos);
        delete rt->c[1]->c[0];
        rt->c[1]->c[0]=0;
        rt->c[1]->pushup();
        rt->pushup();
    }

    void erase(int l,int r)
    {
        l-=shift,r-=shift+1;
        if (l>r) return;
        split(l,r);
        rt->c[1]->c[0]->clear();
        rt->c[1]->c[0]=0;
        rt->c[1]->pushup();
        rt->pushup();
    }

    void modify(int pos,Info x)
    {
        pos-=shift;
        findKth(pos+1,0);
        rt->v=x;
        rt->pushup();
    }

    void rangeApply(int l,int r,Tag w)
    {
        l-=shift,r-=shift+1;
        if (l>r) return;
        split(l,r);
        Node *x=rt->c[1]->c[0];
        *x+=w;
        rt->c[1]->pushup();
        rt->pushup();
    }

    Info rangeQuery(int l,int r)
    {
        l-=shift,r-=shift+1;
        split(l,r);
        return rt->c[1]->c[0]->s;
    }

    ~Splay() { rt->clear(); }
#undef _rev
};

struct Tag
{
    bool rev=0;

    Tag() {}
    Tag(bool c):rev(c) {}

    void operator += (const Tag &o)
    {
        rev^=o.rev;
    }
};

struct Info
{
    i64 x=0;

    void operator += (const Tag &o) const
    {

    }

    Info operator + (const Info &o) const
    {
        return {x+o.x};
    }
};
```