网址

题目详情 - 字胡串(string) - Super

题解 - 字胡串(string) - Super

数据

骗分代码

$\qquad$考虑找到每个区间最大值的位置，设区间 $[L,R]$ 表示该最大值 $a[x]$ 的位置向左向右找到的首个大于最大值 $a[x]$ 的位置，设区间 $[l,r]$ 表示该最大值 $a[x]$ 的位置 x 向左向右找到的第一个能使异或和 $g(S)$ 变大的位置，对答案的贡献是：

$ans+=(l-L)\times(R-pos+1)+(R-r)\times(pos-L+1)-(R-r)\times(l-L);$

$\qquad$可以用**倍增+线段树**来维护求出区间或和与区间最大值，然后再求 $[L,x-1]$ 和 $[x+1,R]$ 区间的贡献即可。

时间复杂度：

最优 $O(nlog^2n)$，当最大值总是在区间中心时取得；

最劣 $O(n^2log^2)$，当输入的自福串从小到大或从大到小排序后取得。

```
#include<algorithm>
#include<iostream>
#include<cstring>
#include<cstdio>
#include<cmath>
#include<vector>
#define _for(i,n) for(int i=1;i<=n;i++)
#define N 1012345
#define int long long
using namespace std;
inline int input(){
    char c;bool f=false;while(!isdigit(c=getchar()))f=c=='-';int x=c^48;
    while(isdigit(c=getchar())){x=(((x<<2)+x)<<1)+(c^48);}return f?-x:x;
}
int n,a[N],ans;

#define ls p<<1
#define rs p<<1|1
struct TR{
    int max,huo,pos;
}tr[N<<2|3];

void pushup(int p){
    tr[p].max=max(tr[ls].max,tr[rs].max);
    tr[p].huo=tr[ls].huo|tr[rs].huo;
    tr[p].pos=tr[ls].max==tr[p].max?tr[ls].pos:tr[rs].pos;
}

void build(int p,int l,int r){
    if(l==r){
```

```
        tr[p].max=tr[p].huo=a[l];
        tr[p].pos=l;
        return;
    }
    int mid=(l+r)>>1;
    build(ls,l,mid);
    build(rs,mid+1,r);
    pushup(p);
}

struct B{
    int x,id;
    bool operator<(const B y)const{return x<y.x;}
}b[N];
inline B ask_max(int p,int l,int r,int L,int R){
    if(L<=l&&r<=R) return {tr[p].max,tr[p].pos};
    B res={0,0};
    int mid=(l+r)>>1;
    if(L<=mid){
        B s=ask_max(ls,l,mid,L,R);
        if(s.x>res.x) res=s;
    }
    if(R>mid){
        B s=ask_max(rs,mid+1,r,L,R);
        if(s.x>res.x) res=s;
    }
    return res;
}

inline int ask_huo(int p,int l,int r,int L,int R){
    if(L<=l&&r<=R) return tr[p].huo;
    int mid=(l+r)>>1;
    int res=0;
    if(L<=mid) res=(res|ask_huo(ls,l,mid,L,R));
    if(R>mid) res=(res|ask_huo(rs,mid+1,r,L,R));
    return res;
}

void search(int L,int R){
    if(L>=R) return;
    //找到区间最大值位置
    B maxnum=ask_max(1,1,n,L,R);
    int l=maxnum.id,r=maxnum.id;
    int pos=maxnum.id;
//  printf("maxnum=a[%d]=%d\n",pos,maxnum.x);
//  printf("%d %d %d\n",L,R,pos);
    //寻找左侧首个满足 g(T)>f(T) 的位置
    for(int i=24;i>=0;i--){
        if(l-(1<<i)<L) continue;
        if(ask_huo(1,1,n,l-(1<<i),pos)>maxnum.x) continue;
        else l=l-(1<<i);
    }
    //寻找右侧首个满足 g(T)>f(T) 的位置
    for(int i=24;i>=0;i--){
        if(r+(1<<i)>R) continue;
        if(ask_huo(1,1,n,pos,r+(1<<i))>maxnum.x) continue;
        else r=r+(1<<i);
    }
```

```
    ans+=(l-L)*(R-pos+1)+(R-r)*(pos-L+1)-(R-r)*(l-L);
    search(L,pos-1);search(pos+1,R);
}

void solve(){
    _for(i,n) b[i]={a[i],i};
    sort(b+1,b+n+1);
    search(1,n);
}

signed main(){
    freopen("string.in", "r", stdin);
    freopen("string.out", "w", stdout);
    n=input();
    _for(i,n) a[i]=input();
    build(1,1,n);
    solve();
    printf("%lld",ans);
    return 0;
}
```

正解代码

$\qquad$对于 $50pts$, 直接$ O\left(n^{2}\right)$ 枚举区间, $O(1)$ 判断是否合法即可。

$\qquad$对于 $100 pts$, 考虑一个区间, 只要这个区间的任意值非最大值有一位不与最大值相同, 那么这个区间就是 合法区间。

$\qquad$找出所有值左右第一个大于它的位置, 那么以它为最大值的区间就固定在这一段中。只要再 找出这个区间中左右第一个有一位不与最大值相同的值的位置, 那么这个位置左边的所有位置都 可以与最大值右边的位置构成一个合法区间。右边也同理。 可以用单调栈找出左右第一个大于它的位置, 再用 $O\left(n \log a_{i}\right)$ *的时间处理出左右第一个某一位为当前数超集的地方（备注：就是找到首个使当前数异或能变大的数的位置，用 $bef*{i,j}$ 与 $aft_{i,j}$ 表示 i 的上一个与下一个转化为二进制后第 j 位为 1 的位置，可预处理）*, 然后就可以 $O(n)$ 统计答案了, 注意处理值相同的情况。

$\qquad$时间复杂度 $O\left(n \log a_{i}\right)$ 。

```
#include <bits/stdc++.h>
using namespace std;
inline int read(){
    char ch=getchar();int i=0,f=1;
    while(!isdigit(ch)){if(ch=='-')f=-1;ch=getchar();}
    while(isdigit(ch)){i=(i<<1)+(i<<3)+ch-'0';ch=getchar();}
    return i*f;
}
const int N=1e6+50;
int n,a[N],l[N],r[N],st[N],pos[N],top,diff_l[N],diff_r[N],mx[35];
int mxpos[35];

int main(){
    freopen("string.in","r",stdin);
    freopen("string.out","w",stdout);
    n=read();
    for(int i=1;i<=n;i++)a[i]=read();
    for(int i=1;i<=n;i++){
        while(top&&st[top]<=a[i])top--;
        l[i]=pos[top]+1;
```

```
        st[++top]=a[i];pos[top]=i;
    }
    top=0; pos[top]=n+1;
    for(int i=n;i>=1;i--){
        while(top&&st[top]<a[i]) top--;
        r[i]=pos[top]-1;
        st[++top]=a[i];pos[top]=i;
    }
    for(int i=1;i<=n;i++){
        int p=0;
        for(int j=0;(1ll<<j)<=a[i];j++){
            if((1ll<<j)&a[i])mx[j]=max(mx[j],i);
            else p=max(p,mx[j]);
        }
        diff_l[i]=p;
    }
    fill(mx,mx+32+1,n+1);
    for(int i=n;i>=1;i--) {
        int p=n+1;
        for(int j=0;(1ll<<j)<=a[i];j++) {
            if((1ll<<j)&a[i]) mx[j]=min(mx[j],i);
            else p=min(p,mx[j]);
        }
        diff_r[i]=p;
    }
    long long ans=0;
    for(int i=1;i<=n;i++) {
        if(diff_l[i]>=l[i])
            ans+=1ll*(diff_l[i]-l[i]+1)*(r[i]-i+1);
        if(diff_r[i]<=r[i])
            ans+=1ll*(r[i]-diff_r[i]+1)*(i-l[i]+1);
        if(diff_l[i]>=l[i]&&diff_r[i]<=r[i])
            ans-=1ll*(r[i]-diff_r[i]+1)*(diff_l[i]-l[i]+1);
    }
    cout<<ans<<endl;
}
```