

位运算问题整理

seeker

2023 年 1 月 28 日

摘要

本篇讲义收集了若干codeforces上的位运算问题，通过分析题目的方式，加深信息学竞赛选手位运算的理解。

1 引言

位运算问题通常伴随着与（AND）、或（OR）、异或（XOR）运算，且以异或运算为主（记作 \oplus ），由于其模型较为单一，涉及到的知识较少，通常以思维题的形式出现，通过位运算题目训练，可以较好的提升选手的问题转化与建模能力。

本文通过一些位运算题目，探讨一些思维构造性问题，帮助选手深入理解位运算。

2 Gardener and the Array

2.1 题目大意

给定数组 c ，定义

$$f(c) = c_1 | c_2 | \dots | c_n$$

其中， $|$ 表示或运算（OR），是否存在 c 的子数组 a 、 b ，使得 $f(a) = f(b)$ 。

2.2 数据范围

n 表示数组 c 的大小, $n \in (1, 1e5)$, $c_i \in (1, 2^{1e5})$

2.3 解题过程

考虑到 $f(c)$ 涉及按位或运算，因此将 c_i 化为二进制的形式。

由于子数组 a, b 不同的定义为存在不同元素，所以可以考虑使 a, b 包含大部分相同元素，只存在一个不同元素，考虑到 $f(a) \leq f(c)$ ，可以令 $a = c$ ，即寻找是否存在元素 c_i ，使得

$$f(c - c_i) = f(c)$$

为了满足上述式子， c_i 二进制位中每一位为1的位，在数组 c 中不能是唯一置1的，否则删去 c_i 后不能满足上式。在实际实现中还需要注意清空数组问题。

2.4 参考代码

Listing 1: Gardener and the Array参考代码

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  #define rep(i,a,b) for(ll i = a ; i <= b ; i++)
5  #define per(i,a,b) for(ll i = a ; i >= b ; i--)
6
7  const int N = 2e5+7;
8  int cnt[N];
9  vector<int>a[N];
10 void solve() {
11     int n;
12     cin >> n;
13     rep(i,1,n) {
14         int k,t;
15         scanf("%d",&k);
16         rep(j,1,k) {
17             scanf("%d",&t);
18             a[i].push_back(t);
19             cnt[t]++;
20         }
21     }
22     rep(i,1,n) {

```

```
23         int flag = 1;
24         for(auto &j:a[i])
25             if(cnt[j]==1) {
26                 flag = 0;
27                 break;
28             }
29         if(flag) {
30             puts("YES");
31             rep(i,1,n) {
32                 for(auto &j:a[i]) {
33                     cnt[j] = 0;
34                 }
35                 a[i].clear();
36             }
37             return ;
38         }
39     }
40     puts("NO");
41     rep(i,1,n) {
42         for(auto &j:a[i]) {
43             cnt[j] = 0;
44         }
45         a[i].clear();
46     }
47     return ;
48 }
49 int main() {
50     int tc;
51     cin >> tc;
52     while(tc--)
53         solve();
54 }
```

3 Orray

3.1 题目大意

给定长度为 n 的非负数组 a ，定义

$$b_i = a_1 | a_2 | \dots | a_i$$

试对着 a 重新排序，使得生成的数组 b 字典序最大。

注：当数组 x 字典序大于 y 时， x 和 y 中第一个不相同的位置 i ，满足 $x_i > y_i$ 。

3.2 数据范围

$$n \in [1, 2 \cdot 10^5], a_i \in [0, 10^9]$$

3.3 解题过程

为了使得字典序最大，第一个 ans_1 值一定是 a 中最大值。那么，为了让第二个 ans_2 值最大，需要使得除了 ans_1 置一位以外的位尽可能的大。因此，等同于在减去 ans_1 的置一位后的数组中寻找最大值，由此得到第二个 ans_2 值。

剩下的答案可以以此类推，即：在剩余数组中寻找并输出最大值，通过减去最大值的置一位更新剩余数组。这种方法看似是 $O(N^2)$ ，但由于每个数据位最多被删去一次，因此在删去全部的数据位后，剩余的数组就全是0了，可以将剩下的 a 直接输出了，实际复杂度是 $O(N \log N)$ 。

3.4 参考代码

Listing 2: Orray参考代码

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define rep(i,a,b) for(ll i = a ; i <= b ; i++)
4 #define per(i,a,b) for(ll i = a ; i >= b ; i--)
5 #define x first
6 #define y second
7 typedef long long ll;
```

```

8  const int N = 2e5+7;
9  int a[N]
10 void solve() {
11     int n;
12     cin >> n;
13     queue<pair<int,int> >b; //The first indicates the remaining
        value, the second indicates the subscript of the number
14     int maxn = -1,maxi;
15     rep(i,1,n) {
16         scanf("%d",&a[i]);
17         b.push({a[i],i});
18         maxn = max(a[i],maxn);
19         if(maxn==a[i]) maxi = i;
20     }
21     rep(i,1,n) { //Output the maximum value in b, and find the
        maximum value on the way to update array b
22         printf("%d ",a[maxi]);
23         queue<pair<int,int> > t;
24         int maxt = -1,maxti = -1;
25         while(b.size()) {
26             auto x = b.front();
27             b.pop();
28             if(x.y==maxi)
29                 continue;
30             x.x -= x.x&maxn;
31             t.push(x);
32             maxt = max(maxt,x.x);
33             if(maxt==x.x) maxti = x.y;
34         }
35         maxn = maxt;
36         maxi = maxti;
37         b = t;
38
39         if(maxn==0) {
40             while(b.size()) {
41                 auto x = b.front();
42                 b.pop();
43                 printf("%d ",a[x.y]);
44             }

```

```
45         return;
46     }
47 }
48 }
49
50
51 int main() {
52     int tc;
53     cin >> tc;
54     while(tc--)
55         solve(), cout << endl;
56 }
```

4 Even Subarrays

4.1 题目大意

给定长度为 n 的数组 a ，计算数对 (i, j) 的数量，满足

$$x = a_i \oplus a_{i+1} \oplus \dots \oplus a_j$$

，其中 x 是有偶数个因子数字。

4.2 数据范围

$$n \in [2, 2 \cdot 10^5], a_i \in [1, n]$$

4.3 解题过程

首先考虑满足偶数因子的数字问题，由于一个非1数字的因子至少包含1和其本身，所以有奇数个因子的数字一定是完全平方数。

本题转换为，计算有多少对数对的异或和是完全平方数，由于 $a_i \in [1, 2 \cdot 10^5]$ ，所以其异或和得到的完全平方数小于262144，可以预处理1000多个完全平方数。

现在考虑如何枚举数对，设 $f(x) = a_1 \oplus a_2 \oplus \dots \oplus a_x$ ，通过异或的性质可以得到：

$$\begin{aligned}(i, j) &= a_i \oplus a_{i+1} \oplus \dots \oplus a_j \\ &= f(j) \oplus f(i-1)\end{aligned}\tag{1}$$

令 s 表示任意完全平方数，如果 $(i, j) = s$ ，则 $f(j) \oplus f(i-1) = s$ ，通过异或性质得 $s \oplus f(j) = f(i-1)$ 。因此，对于 $i \in [1, n]$ ，我们可以枚举 s ，通过记录之前的 $f(x)$ 的方式，寻找是否存在异或和，满足

$$f(i) \oplus s = f(j), i > j$$

枚举 f 中再枚举完全平方数 s ，因此时间复杂度为 $O(N \cdot \sqrt{2 \cdot 10^5}) = O(N\sqrt{N})$

4.4 参考代码

Listing 3: Even Subarrays参考代码

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  #define rep(i,a,b) for(ll i = a ; i <= b ; i++)
5  #define per(i,a,b) for(ll i = a ; i >= b ; i--)
6  #define x first
7  #define y second
8
9  const int N = 4e5+7;
10
11 ll a[N*2], cnt[N*2];
12 ll n;
13 ll mp[N*2];
14 vector<ll>x;
15 void solve() {
16     cin >> n;
17     rep(i,1,n) {
18         cin >> a[i];
19     }
20     ll sum = 0;
21     ll ans = 0;
22     rep(i,1,n) {

```

```

23         ans += n-i+1;
24     }
25     rep(i,1,n) {
26         sum ^= a[i];
27         if(mp[sum])
28             ans--;
29         for(auto &j:x) {
30             ans -= cnt[j^sum];
31         }
32         cnt[sum]++;
33     }
34     cout << ans << endl;
35     sum = 0;
36     rep(i,1,n) {
37         sum ^= a[i];
38         cnt[sum] = 0;
39     }
40 }
41 int main() {
42     rep(i,0,N) {
43         if(i*i>=N)
44             break;
45         mp[i*i] = 1;
46         x.push_back(i*i);
47     }
48     int tc;
49     cin >> tc;
50     while(tc--)
51         solve();
52 }

```

5 Vlad and a Pair of Numbers

5.1 题目大意

给定 x ，问是否存在 a, b ，满足

$$x = a \oplus b = \frac{a \cdot b}{2}$$

5.2 数据范围

$$x \in [1, 2^{29}]$$

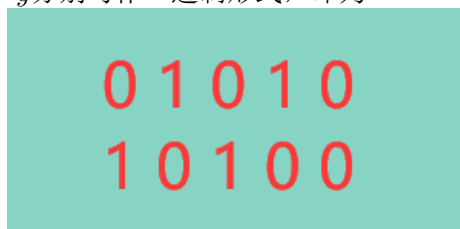
要求答案满足 $a, b \in [0, 2^{29}]$

5.3 解题过程

首先观察到, $a + b = 2 \cdot x + x\%2$, 证明如下:

如果 $a \oplus b$ 的最后一位为1, 则 a 与 b 仅有一个最后一位为1, 因此 $a + b$ 的最后一位为1; 如果 $a \oplus b$ 最后一位为0, 则 a 与 b 最后一位相同, 无论为1还是为0, $a + b$ 的结果都为0。因此, $a + b = 2 \cdot a \oplus b + x\%2$ 。

令 $y = a + b$, 将 x, y 二进制化, 依据上述结论按位从高位向低位分析。以 $x = 10$ 为例, x, y 分别写作二进制形式, 即为



$$\begin{array}{r} 01010 \\ 10100 \end{array}$$

由于 y 是 x 向高位移动产生的, 所以一定是 y 的最高位为1, x 此位为0, 为了满足条件, 需要向 a 和 b 的低位借位。

我们首先思考什么情况不存在 a, b 满足条件。假设我们在某低位已经欠高位一位数了 (俗称欠债):

- 当 x 此位为0时, 我们可以使 a, b 都置1来向高位进位来还清债务, 此时无论 y 是1还是0都有机会处理掉: 如果是1的话, 就相当于再向低位借位; 如果是0的话, 就已经满足条件了;
- 当 x 为1时, 由于 $a+b$ 一定为 $0+1$ 的状态, 所以靠这一位是没法还债的, 只能依靠于下一位进位, 才能加起来变成10, 也就是说, 这一位的和一定是0, 此时, 如果 y 要求1的话, 便无法满足条件了。

Corollary 5.1 在已经被借位的情况下, $x = 1$ 且 $y = 0$ 时 ab 不存在解。

“还债”的事情讨论完了, 我们再讨论一下导致“欠债”的情况。

- 当 x 为0时，考虑到 ab 相同，相加的结果一定是0，所以当 y 等于1的时候会导致“欠债”，需要借位。
- 当 x 为1时，考虑到 ab 不同，结果一定是1，当 y 等于0时会导致“欠债”。

由此，可以得出答案。但是依靠分类讨论的解法，我们忽视了一个条件： x 和 y 是错位的相同的，事实上，依靠这个条件，我们可以用一个更简单的结论去总结上述情况：

Corollary 5.2 x 中不能包含连续的1

考虑到： x 和 y 是错位的相同的，那么 x 中连续的1一定会导致“欠债”状态下的 $x = 1, y = 0$ ，或者是直到结尾也无法还债的情况 $x = 1, y = 1$ ，所以一定不满足条件。

5.4 参考代码

Listing 4: Vlad and a Pair of Numbers参考代码

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define rep(i,a,b) for(ll i = a ; i <= b ; i++)
4  #define per(i,a,b) for(ll i = a ; i >= b ; i--)
5  #define x first
6  #define y second
7  typedef long long ll;
8  const int N = 2e5+7;
9
10 void solve() {
11     ll x;
12     cin >> x ;
13     ll y = x*2;
14     if(x%2) y+=1;
15     ll a = 0, b = 0;
16     ll t = 1;
17     rep(i,1,31) t*=2ll;
18     bool flag = 0;
19     per(i,32,1) {

```

```
20         if(flag) {
21             if((x&t)==0) {
22                 flag = 0;
23                 a+=t,b+=t;
24             } else if(y&t) {
25                 puts("-1");
26                 return ;
27             }
28         }
29         if(x&t) {
30             a+=t;
31         }
32         if((x&t)==0 && (y&t)) {
33             flag = 1;
34         }
35         t/=2;
36     }
37     cout << a << " " << b << endl;
38 }
39
40
41 int main() {
42     int tc;
43     cin >> tc;
44     while(tc--)
45         solve();
46 }
```