8，位运算整章注解

2018年4月6日 15:26

算术运算的常见操作符：
+ - * / %
位运算的常见操作符：
&按位与
| 按位或
^ 按位异或
~ 按位取反
<< 左移，右侧补0
>>右移，左侧补符号位
>>>右移，左侧补0

问题：不安全网页的黑名单包含100亿个黑名单网页，每个网页的url最多占用64字节，现在想要实现一种网页过滤系统，可以根据网页的url判断其是否在安全黑名单上，请设计该系统，要求该系统有万分之一以下的判断失误率，并且使用的额外空间不要超过30G

解法：

常规的解法是将黑名单存入哈希表or数据库，再查询
但这种情况下，100亿个64字节的url，需要6400亿byte，大约需要640G空间，是不满足题目要求的。

因此，当面试者遇到网页黑名单系统，垃圾邮件过滤系统，爬虫网址重复判断系统时，又看到系统容忍一定的失误率，但对空间的要求比较严格，这里可能是面试官希望用到布隆过滤器。

什么是布隆过滤器呢？
布隆过滤器可精准的代表一个集合，可以精准的判断某一元素是否在此集合，精准程度由用户的具体设计决定。做到100%的正确是不能的。但其优势在于，可以用很少的空间，并且做到精准率较高

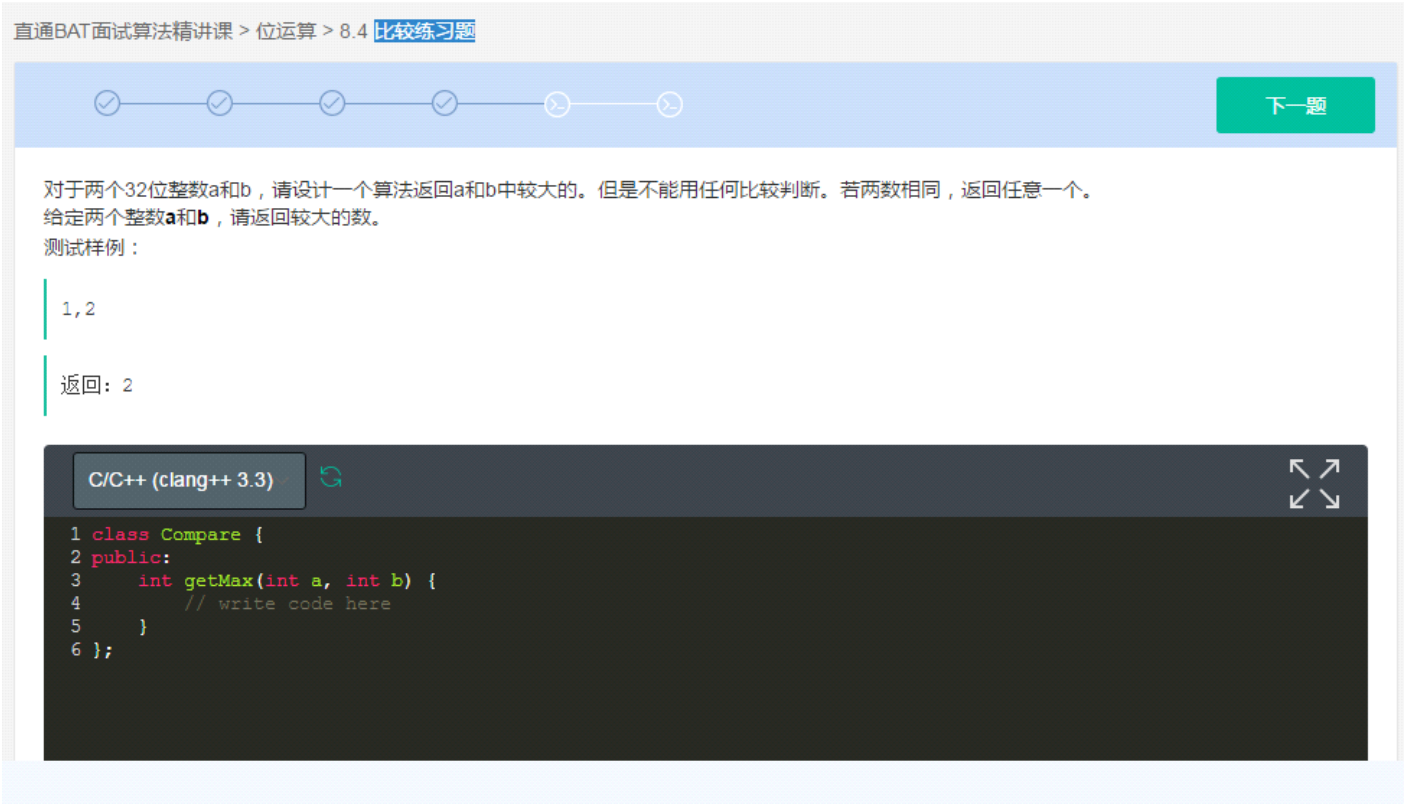
我们定义一定长度的bit数组，其中每位置0置1主要看该位置的下标是否出现过。

比如这里100亿个url，每个url可以将其用hash函数转为一个k整数值，将其存至2^32次方长的bit数组下标里，对应的下标位置1，即可。

现在有一个新的url a，查询时，只需要将其通过同样的hash函数进行映射，然后判断这个映射值得下标是否被涂黑即可。

其犯错类型是宁可错杀三千，不可放过一个，即出现过得一定会被判断出来，但没出现过得，也有可能进行误判。

1，



/*
问题：给定两个32位整数a，b，返回ab中较大的，但不能用任何比较判断

解法：先比较a与b两个数的符号，

符号不同，直接返回符号为正的那个数。

如果a为0或正，那么b为负(sa==1,sb==0)，则返回a;

如果a为负，那么b为0或正(sa==0,sb==1)，则返回b;

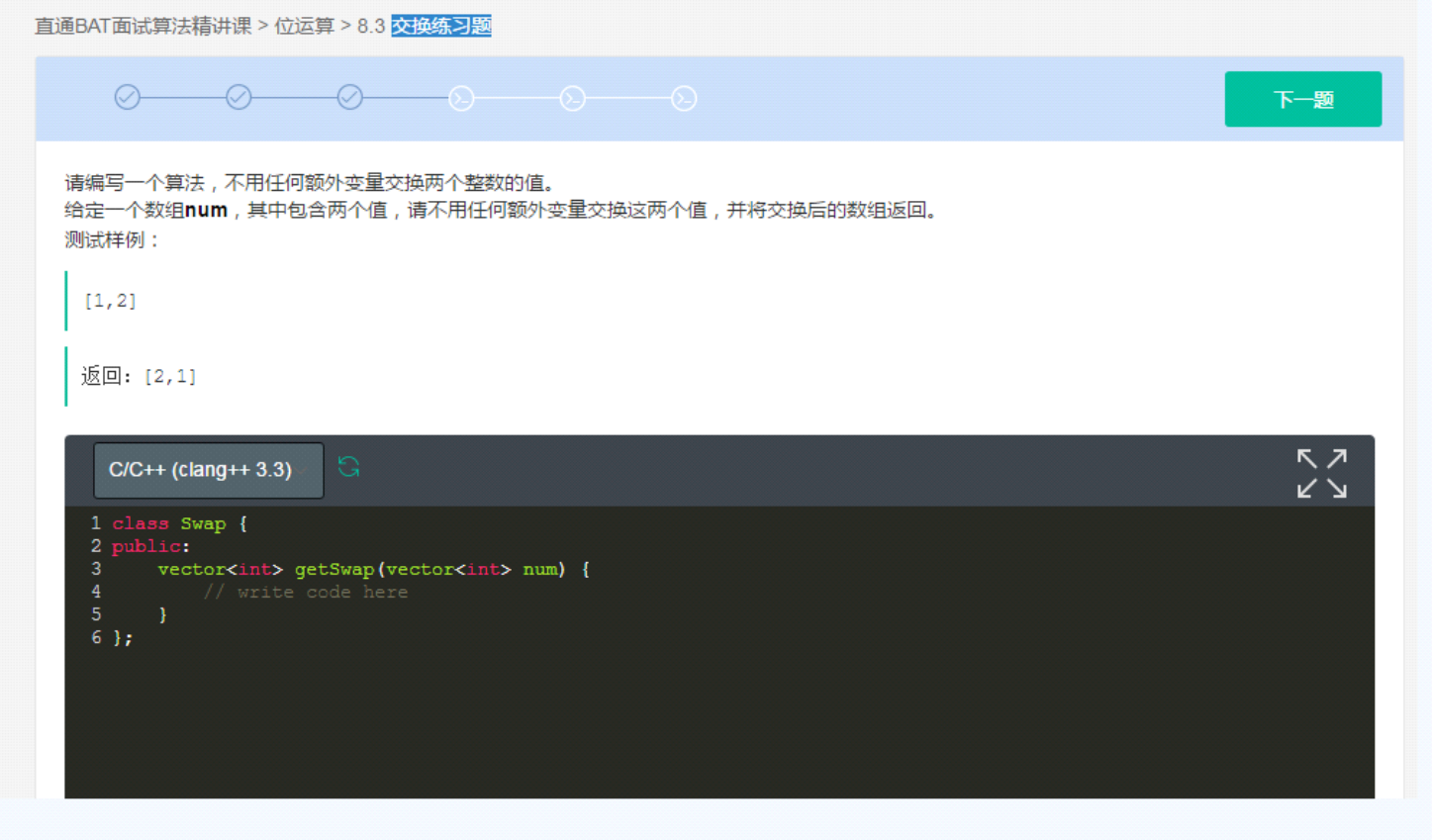
如果符号相同，这种情况下,a-b的值绝对不会溢出，那么就看c的符号。

如果c=a-b,为正返回a;

如果c=a-b,为负返回b;
*/

来自 <http://tool.oschina.net/highlight>

屏幕剪辑的捕获时间: 2018/4/6 15:27,2,
2 ,



屏幕剪辑的捕获时间: 2018/4/6 15:27,3，

```
/*
如何不用任何额外的变量，交换两个整数的值
a = a^b;
b = a^b;//a^b^b = a
a = a^b;//a^b^a = b
*/

#include<bits/stdc++.h>
using namespace std;
class Solution
{
public:
    void solution(int &x, int &y)
    {
        x = x^y;
        y = x^y;
        x = x^y;
    }
};

int main()
{
    int x = 1;
    int y = 2;
    Solution s;
    s.solution(x,y);
    cout<<x<<" "<<y<<endl;
    return 0;
}
```

来自 <http://tool.oschina.net/highlight>

3，

屏幕剪辑的捕获时间: 2018/4/6 15:28

```
/*
问题：给定一个整形数组arr，其中只有两个数出现了奇数次，其他的数都出现了偶数次，要求打印这两个奇数
时间复杂度要求o(n)，空间o(1)

同8_4的思路，定义一个额外变量eo，假设出现奇数次的数为a，b。

最终eo的值为a^b.然后遍历eo的每一位，找到第一个为1的位，自然可知，a，b在此位上定有一个为1，一个为0
不妨假设a在此位为1
再定义一个eo‘的变量，其亦或arr中所有此位为1的数，从而得到的值定为a
然后eo^eo',得到的值即为b。
*/
#include<bits/stdc++.h>
using namespace std;


class Solution
{
public:
    vector<int> solution(vector<int> arr,int n)
    {
        vector<int> res;

        int eo = 0;//找到两个奇数的亦或和
        for(int i = 0;i<n;i++)
            eo ^= arr[i];

        int flag = 0;//找到eo里第一位不是0的位置
        int temp = eo;
        for(int i = 0;i<32;i++)
        {
            if(((temp>>i )& 1) == 1)
            {
                flag = i;
                break;
            }
        }

        int eo1 = 0;
        //oe1只亦或flag位为1的数组元素
        for(int i= 0;i<n;i++)
```

```
        {
            if(((arr[i]>>flag)&1) == 1)
            {
                eo1^=arr[i];
            }
        }
        eo = eo^eo1;
        res.push_back(eo<eo1?eo:eo1);//按从小到大的顺序放
        res.push_back(eo>eo1?eo:eo1);
        return res;
    }
};

int main()
{
    int n = 8;
    int a[n] ={1,5,5,5,2,2,4,4};
    vector<int> arr(a,a+n);
    Solution s;
    vector<int> res = s.solution(arr,n);

    vector<int>::iterator iter;
    for(iter = res.begin();iter!=res.end();iter++)
        cout<<*iter<<" ";
    cout<<endl;
    return 0;
}
```

来自 <http://tool.oschina.net/highlight>

4，

屏幕剪辑的捕获时间: 2018/4/6 15:28

```
/*
问题：给定一个整形数组arr，其中只有一个数出现了奇数次，其他的数都出现了偶数次，要求打印这
个数
时间复杂度要求o(n)，空间o(1)

解法：
由于任何数与0亦或是其本身，
任何数和其本身亦或为0.
定义一个额外变量oe = 0，遍历数组，将其中每个数与oe亦或，最后oe的值即为哪个单独出现的数
这是因为亦或运算，满足交换律和结合律
如 arr = cbdaabc
```

由于交换律和结合率，oe按从左到右的顺序亦或arr和直接亦或aabbbccd是没有区别的
也就是说，最后剩下的那个，肯定是出现奇数次的

拓展，如何设置一种加密过程，完成对明文的加解密工作？
答:这里可以用亦或，比如明文为text,密码为pw，即可对两者按位亦或得到密文，解密时再亦或密码即可
如果明文长，密码短，对密码重复使用即可。
*/

```
#include<bits/stdc++.h>
using namespace std;
class Solution
{
public:
    int solution(vector<int> arr,int n)
    {
        int oe = 0;
        for(int i = 0;i<n;i++)
            oe^=arr[i];
        return oe;
    }
};


int main()
{
    int n = 7;
    int a[7] = {1,1,1,2,1,5,5};
    vector<int> arr(a,a+n);
    Solution s;
    cout<<s.solution(arr,n)<<endl;
    return 0;
}
```

来自 <http://tool.oschina.net/highlight>