

# 线性基

定义：

张成：

- $T \subseteq S$  所有这样是子集  $T$  的异或和组成的集合称为  $S$  的张成记作  $\text{span}(S)$

线性相关：

- 存在一个元素  $s_j$  可以用其它若干元素异或起来得到
- 或：存在一个子集的异或和为0

线性基：

称  $B$  是  $S$  的线性基当且仅当：

1.  $S \subseteq \text{span}(B)$  即  $S$  是  $B$  的张成的子集。
2.  $B$  是线性无关的。

线性基的长度：

- 集合  $B$  中元素的个数

线性基的基本性质：

1.  $B$  是极小的满足线性基性质的集合，它的任何真子集都不可能是线性基；
2.  $S$  中的任意元素都可以唯一表示为  $B$  中若干个元素异或起来的结果。

构造：

线性基是动态构造的，我们只需要从空的  $a$  开始，每次考虑在一个已存在的线性基中插入一个数  $t$  即可。

构造流程：

对于每个数  $x$  从高位到低位扫描，扫到第  $i$  位为1时，若  $a_i$  上有数字  $x = x \oplus a_i$  并向下继续扫，若没有数字  $a_i = x$  结束扫描。

```
inline void insert(ll x){
    for(int i=50;i>=0;i--if(x&(1ll<<i)){
        if(a[i]) x^=a[i];
        else{a[i]=x;break ;}
    }
}
```

}

## 大致证明：

### 未成功插入线性基：

考虑时候不能插入进去呢，显然就是它在尝试插入时异或若干个之后变成了0  $\rightarrow$  新插入的数 $x$ 可由当前的线性基异或表示  $\rightarrow$  所有数都能由当前的线性基异或表示。

### 成功插入线性基：

假设 $x$ 成功插入到了 $a_i$ 显然，它在插入前可能异或若干个，那么就有：

$$x \oplus a_j \oplus a_k \oplus \dots = a_i$$

$$\text{即：} a_i \oplus a_j \oplus a_k \oplus \dots = x$$

$\rightarrow$  新插入的数 $x$ 可由当前的线性基异或表示  $\rightarrow$  所有数都能由当前的线性基异或表示。

## 性质：

构造出来的线性基满足：

1. 若 $a_i=0$ 
  - 只有满足 $j>i$ 的 $a_j$ 的第 $i$ 位才可能为1
2. 若 $a_i \neq 0$ 
  - 整个 $a$ 数组只有 $a_i$ 的第 $i$ 位二进制位1
  - $a_i$ 的更高位( $>i$ 的位数)一定为0
  - $a_i$ 的更低位( $<i$ 的位数)可能为1

## 合并：

将一个线性基中的所有元素插入到另一个线性基中。（线性基的个数最多为二进制的位数）

## 删除：

若 $x$ 在线性基外：直接删掉 若 $x$ 在线性基内：

没有在线性基中的数，一定是因为线性基中的若干个可以异或得到他，那么可以记录一下不在线性基中的数都是由线性基中的哪些数异或得到的，那么每一个线性基外的数对应一个集合 $S$ 这个集合内就是线性基中那些异或起来可以得到他的数。

假如线性基外的某一个数的 $S$ 中包含 $x$ 那么就不需要删除 $x$ 把这个数删除即可。原因是这个数在线性基中是可以代替 $x$ 的，那么就当这个数代替了 $x$ 然后 $x$ 被删除了，然后把线性基中的 $x$ 当做这个数即可，这样的话线性基不会有变化。（实现起来并不需要维护集合 $S$ 而是直接维护有哪些数可以代替线性基中的数就好了）

假如 $x$ 不被线性基外的任何一个数的 $S$ 包含，那么就另外造一个集合 $P$ 记录线性基中这个数插入进

来的时候异或过哪些数。然后找到线性基中最小的并且包含 $x$ 的数，让他异或线性基中其他包含 $x$ 的数即可（包括自己，自己异或完自己后自己这一位就变成了0），这样就能消除 $x$ 在线性基中的影响（事实上就等价于用最小的数代替了它）。

总之，由于如果修改了线性基中的某一位会影响到一些比它小的位，所以一般不能修改，要么改最小的并且不会影响到下面的位。

## 应用：

### 查询 $x$ 是否在集合的张成中：

对数字 $x$ 从高位到低位扫描，若该位为1  $x = x \oplus a_i$

若 $x$ 最后异或结果为0则 $x$ 在当前集合的张成中，否则不在

```
inline bool check(ll x){
    for(int i=50;i>=0;i--)
        if(x&(1ll<<i)) x^=a[i];
    return x==0;
}
```

### 求数集能异或出的最大值：

使用贪心的思想:

```
inline ll getMax(){
    ll ans=0;
    for(int i=50;i>=0;i--)
        ans=max(ans,ans^a[i]);
    return ans;
}
```

证明：

$a_i$ 的第 $i$ 位一定是1，而后面扫描到的 $a_{i-1}, a_{i-2}, \dots, a_0$ 第 $i$ 位一定不是1，而如果高位不是1低位再多1也弥补不了，所以当前只需要考虑第 $i$ 位的情况，若 $ans$ 的第 $i$ 位为0异或后为1会变大就进行异或，若为1异或后为0会变小就不进行异或。

### 求数集能异或出的最小值：

显然是最小的一个 $a_i$  最小的 $a_i$  异或其它数字都会变大。

### 求能异或出的第 $k$ 小值：

从一个序列中取任意个元素进行异或，求能异或出的所有数字中第 $k$ 小的那个。

首先对线性基进行改造，保证只有 $a_i$ 的第 $i$ 位是1，其它的第 $i$ 位都为0。

改造方法是若 $a_i$ 的第 $j$ 位为1( $j < i$ )  $a_i = a_i \oplus a_j$

求解过程：将 $k$ 转化为二进制，如果 $k$ 的第 $i$ 位为1  $ans$ 异或上线性基中的第 $i$ 个元素。（不是直接异或 $a_{i-1}$  见代码）

```
void init(){//处理线性基
    for(int i=1;i<=50;i++)
        for(int j=0;j<i;j++)
            if(a[i]&(1ll<<j))
                a[i]^=a[j];
}
ll k_th(ll k){
    if(k==1&&tot<n)
        return 0;
    /*特判一下,假如k=1
    并且原来的序列可以异或出0,
    就要返回0 tot表示线性基中的元素个数,
    n表示序列长度*/
    if(tot<n) k--;
    //类似上面,去掉0的情况,因为线性基中只能异或出不为0的解
    init();
    ll ans=0;
    for(int i=0;i<=50;i++)if(a[i]){
        if(k&1) ans^=a[i];
        k>>=1;
    }
    return ans;
}
```

证明（或者说是理解）：其实处理完之后的线性基其实也还是原序列的一个线性基，回想上面的线性基处理过程，可以发现，处理完之后，线性基中的元素，作用其实都是提供自己最高位上的1，那么只要使提供出来的1可以和  $k$  的二进制的每一位上的1对应，那么求出的  $ans$  一定就是第  $k$  小的。

题目：

## 洛谷4570 [BJWC2011]元素

题意：给定一些矿石编号和矿石价值，问在保证矿石编号的任意子集的异或和不为0的情况下，最大价值是多少。

题解：很显然的贪心，按价值从大到小排序维护线性基。

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<set>
#include<queue>
#include<vector>
#include<cmath>
#include<cstdlib>
#include<map>
#include<stack>
```

```
using namespace std;
typedef long long ll;
typedef pair<int,ll> pi;
const int N=1005;
ll getint(){
    char c;bool flag=0;ll num=0ll;
    while((c=getchar())<'0' || c>'9')if(c=='-')flag=1;
    while(c>='0'&&c<='9'){num=num*10+c-48;c=getchar();}
    if(flag) num=-num;
    return num;
}
int n,ans;pi x[N];ll a[65];
inline bool insert(ll x){
    for(int i=60;i>=0;i--)if(x&(1ll<<i)){
        if(a[i]) x^=a[i];
        else{a[i]=x;break ;}
    }
    return x;
}
int main(){
    n=getint();
    for(int i=1;i<=n;i++){
        x[i].second=getint();
        x[i].first=-getint();
    }
    sort(x+1,x+n+1);
    for(int i=1;i<=n;i++)
        if(insert(x[i].second))
            ans-=x[i].first;
    cout<<ans<<endl;
}
```

From:  
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:  
<https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:famerwzyuki:%E7%BA%BF%E6%80%A7%E5%9F%BA&rev=1590745598>

Last update: 2020/05/29 17:46