

原创 位运算+树状数组

2019-05-21 10:17:59 \_-Y--Y- 阅读数 35 文章标签: ACM 更多

版权声明: 本文为博主原创文章, 遵循 CC 4.0 BY-SA 版权协议, 转载请附上原文出处链接和本声明。  
本文链接: [https://blog.csdn.net/weixin\\_44410512/article/details/90401580](https://blog.csdn.net/weixin_44410512/article/details/90401580)

按位求与 (&)

0&0=0  
0&1=0  
1&0=0  
1&1=1

按位求或 (|)

0|0=0  
0|1=1  
1|0=1  
1|1=1

按位取反 (~)

~0=1  
~1=0

按位异或 (^)

0^0=0  
1^0=1  
0^1=1  
1^1=0

优先级

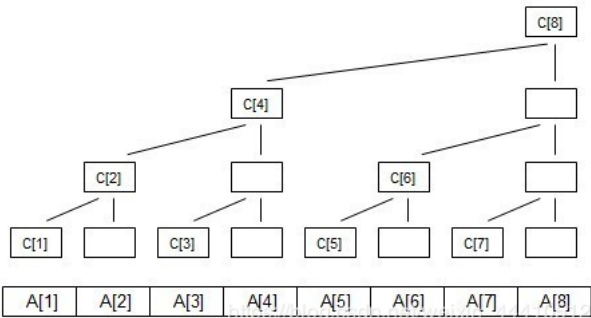
优先级	运算符	结合性
1	() [] .	从左到右
2	! + (正) - (负) ~ ++ --	从右向左
3	* / %	从左向右
4	+ (加) - (减)	从左向右
5	<< >> >>>	从左向右
6	< <= > >= instanceof	从左向右
7	= = ! =	从左向右
8	& (按位与)	从左向右
9	^	从左向右
10		从左向右
11	&&	从左向右
12		从左向右
13	?:	从右向左
14	= += -= *= /= % = &=  = ^= ~< > >> >>>=	从右向左

以下内容来自[https://blog.csdn.net/Small\\_Orange\\_glory/article/details/81290634](https://blog.csdn.net/Small_Orange_glory/article/details/81290634)

树状数组

树状数组核心就是

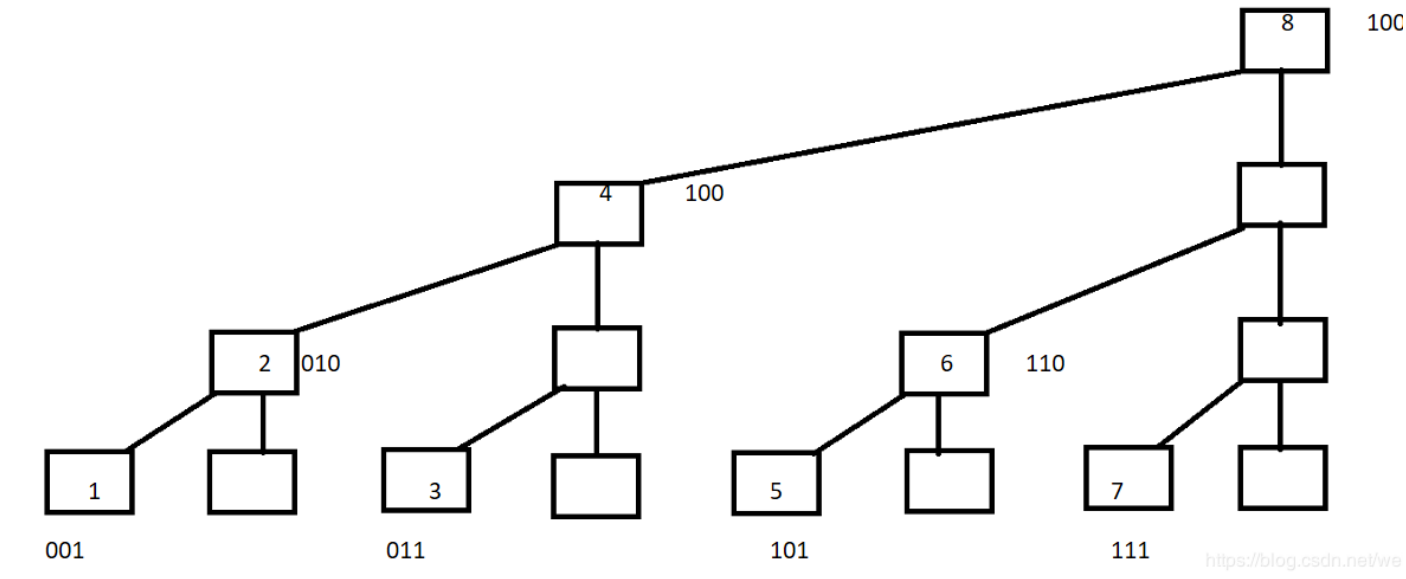
```
1 | int lowbit(int t){  
2 |     return t&(-t);  
3 | }
```



C[i]代表 子树的叶子结点的权值之和// 这里以求和举例

如图可以知道

$C[1] = A[1];$   
 $C[2] = A[1] + A[2];$   
 $C[3] = A[3];$   
 $C[4] = A[1] + A[2] + A[3] + A[4];$   
 $C[5] = A[5];$   
 $C[6] = A[5] + A[6];$   
 $C[7] = A[7];$   
 $C[8] = A[1] + A[2] + A[3] + A[4] + A[5] + A[6] + A[7] + A[8];$



将C[]数组的结点序号转化为二进制

$1 = (001) \quad C[1] = A[1];$   
 $2 = (010) \quad C[2] = A[1] + A[2];$   
 $3 = (011) \quad C[3] = A[3];$   
 $4 = (100) \quad C[4] = A[1] + A[2] + A[3] + A[4];$   
 $5 = (101) \quad C[5] = A[5];$   
 $6 = (110) \quad C[6] = A[5] + A[6];$   
 $7 = (111) \quad C[7] = A[7];$   
 $8 = (1000) \quad C[8] = A[1] + A[2] + A[3] + A[4] + A[5] + A[6] + A[7] + A[8];$

向上建树

```
1 void update(int x,int val){
2     while(x<=n){
3         t[x]+=val;
4         x+=x&(-x);
5     }
6 }
```

例如二进制数 0 0 0 1 向上建树

原码 0 0 0 1 (1)

反码 1 1 1 0

补码 1 1 1 1

$x \& (-x)$  0 0 0 1

原码 0 0 1 0 (2)

反码 1 1 0 1

补码 1 1 1 0

$x \& (-x)$  0 0 1 0

原码 0 1 0 0 (4)

反码 1 0 1 1

补码 1 1 0 0

$x \& (-x)$  0 1 0 0

例如 二进制 0 0 1 1

原码 0 0 1 1 (3)

反码 1 1 0 0

补码 1 1 0 1

$x \& (-x)$  0 0 0 1

原码 0 1 0 0 (4)

反码 1 0 1 1

补码 1 1 0 0

$x \& (-x)$  0 1 0 0

原码 1 0 0 0 (8)

反码 0 1 1 1

补码 1 0 0 0

$x \& (-x)$  1 0 0 0

例如 二进制 0 1 0 1

原码 0 1 0 1 (5)

反码 1 0 1 0

补码 1 0 1 1

$x \& (-x)$  0 0 0 1

原码 0 1 1 0 (6)

反码 1 0 0 1

补码 1 0 1 0

$x \& (-x)$  0 0 1 0

区间查找 (1~x)

```

1 | int sum(int x){
2 |     int ans=0;
3 |     while(x>=1){
4 |         ans+=t[x];
5 |         x-=x&(-x);
6 |     }
7 |     return ans;
8 | }
```

## 例一

<https://cn.vjudge.net/problem/POJ-2299>

## AC code

```

1 | #include <iostream>
2 | #include <algorithm>
3 | #include <cstring>
4 | #include <cstdio>
5 | using namespace std;
```

```
6  #define maxn 500010
7  int n;
8  int a[maxn];
9  int t[maxn*4];
10 struct node{
11     int x;
12     int id;
13 }f[maxn];
14 bool cmp(node a, node b){
15     return a.x<b.x;
16 }
17 void update(int x,int val){
18     while(x<=n){
19         t[x]+=val;
20         x+=x&(-x);
21     }
22 }
23 int sum(int x){
24     int ans=0;
25     while(x>=1){
26         ans+=t[x];
27         x-=x&(-x);
28     }
29     return ans;
30 }
31 int main(){
32     while(~scanf("%d", &n),n){
33         for(int i=1;i<=n;i++){
34             scanf("%d", &f[i].x);
35             f[i].id=i;
36         }
37         sort(f+1,f+n+1,cmp);
38         for(int i=1;i<=n;i++) a[i]=f[i].id;
39         memset(t,0,sizeof(t));
40         long long ans=0;
41         for(int i=1;i<=n;i++){
42             update(a[i],1);
43             ans+=(i-sum(a[i]));
44         }
45         printf("%lld\n", ans);
46     }
47     return 0;
48 }
```

有 0 个人打赏

文章最后发布于: 201

©2019 CSDN 皮肤主题: 大白 设计师: CSDN官方博客