

树状数组

为什么需要树状数组？

当我们需要维护一个数组的前缀和 $S[i]=A[1]+A[2]+.....+A[i]$ 时，如果修改了任意一个 $A[i]$ ， $S[i]$ 都会发生变化。在最坏情况下，会需要 $O(n)$ 时间，引入树状数组后，修改和求和都是 $O(\log n)$ ，极大提高效率。

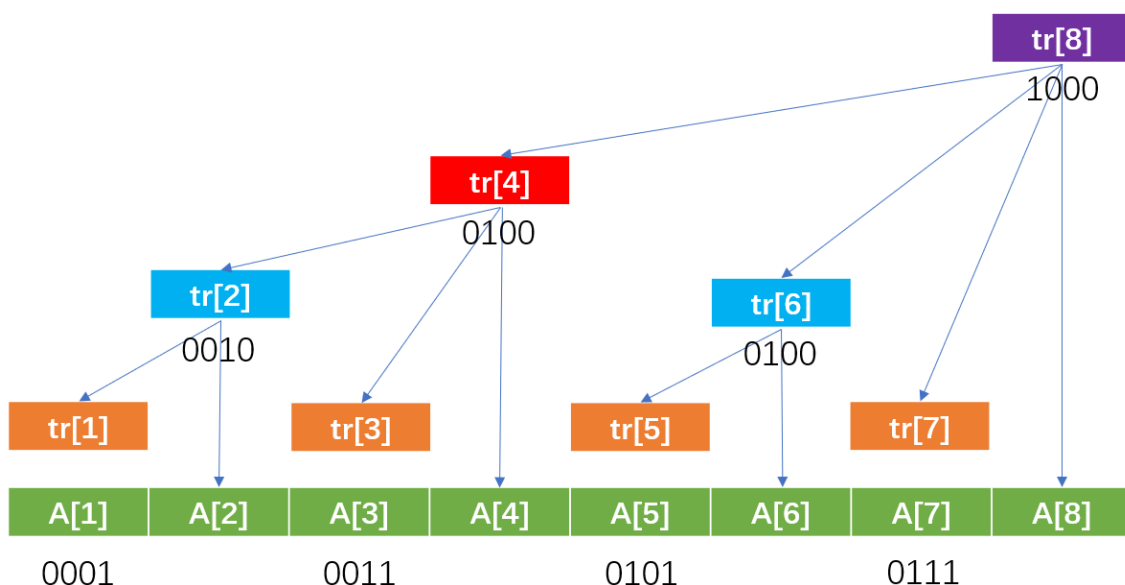
基本思想：根据任意正整数关于2的不重复次幂唯一分解性质，若一个正整数21的二进制表示为10101 = $2^4 + 2^2 + 2^0$ ，因此，区间[1,x]可以分成 $O(\log x)$ 个小区间：

- 长度为 2^4 的小区间 $[1, 2^4]$ ，即 $[1, 16]$ ；
- 长度为 2^2 的小区间 $[2^4 + 1, 2^4 + 2^2]$ ，即 $[17, 20]$ ；
- 长度为 2^0 的小区间 $[2^4 + 2^2 + 1, 2^4 + 2^2 + 2^0]$ ，即 $[21, 21]$ 。

这些子区间共同特点是：若区间结尾为R，则区间长度就是R的二进制分解下最小的1所在位置2的次幂，设为 $\text{lowbit}(R)$ 。

例如：

- $16=10000$ ，区间长度 $16=2^4$ ；
- $20=10100$ ，区间长度 $4=2^2$ ；
- $21=10101$ ，区间长度 $1=2^0$ 。



- $1=(0001)$ $\text{tr}[1]=A[1]$
- $2=(0010)$ $\text{tr}[2]=A[1]+A[2]$
- $3=(0011)$ $\text{tr}[3]=A[3]$
- $4=(0100)$ $\text{tr}[4]=A[1]+A[2]+A[3]+A[4]$
- $5=(0101)$ $\text{tr}[5]=A[5]$
- $6=(0110)$ $\text{tr}[6]=A[5]+A[6]$
- $7=(0111)$ $\text{tr}[7]=A[7]$
- $8=(1000)$ $\text{tr}[8]=A[1]+A[2]+A[3]+A[4]+A[5]+A[6]+A[7]+A[8]$

$$\text{tr}[i]=A[i-2^k+1]+A[i-2^k+2]+.....+A[i]$$

单点更新

例如：当前更改 $A[1]$ ，在 $A[1]$ 基础之上加 t

公众号：黑猫编程

网址：<https://noi.hioier.co>

- $1=(0001)$ $tr[1]+=t$ $1+lowbit(1)=2(0010)$
- $2=(0010)$ $tr[2]+=t$ $2+lowbit(2)=4(0100)$
- $4=(0100)$ $tr[4]+=t$ $4+lowbit(4)=8(1000)$
- $8=(1000)$ $tr[8]+=t$ $8+lowbit(8)=16(10000)$ 由于给定数组长度是8，而16超过8，因此不需要继续计算

```

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

```

查询前缀和

假定 $x=7$, $sum[7]=A[1]+A[2]+A[3]+A[4]+A[5]+A[6]+A[7]$

- $tr[7]=A[7]$
- $tr[6]=A[5]+A[6]$
- $tr[4]=A[1]+A[2]+A[3]+A[4]$

```

1 int sum(int x){
2     int res = 0;
3     while(x){
4         res += tr[x];
5         x -= lowbit(x);
6     }
7     return res;
8 }

```