# BIT讲解

## 引入

考虑在一个区间上进行单点修改，然后查询任意前缀和的操作，要求时间复杂度不能大于 $\mathcal{O}(n^2)$ ，有哪些数据结构或算法可以胜任。

很明显，暴力数据结构线段树是可行的，时间复杂度 $\mathcal{O}(n \log n)$ ，但是常数大到离谱，有测试表明甚至不如下面说的 $\mathcal{O}(n\sqrt{n})$ ，而且很容易没打好。

另外分块也可以，将整个区间分成 $\sqrt{n}$ 块，每个块 $\sqrt{n}$ 个数，查询区间就是拆成很多个完整的块以及剩下的头和尾，头尾元素均不超过 $\sqrt{n}$ ，完整块个数不超过 $\sqrt{n}$ ，只要把完整块的数据处理之后，时间复杂度就可以来到 $\mathcal{O}(n\sqrt{n})$。

以上的数据结构要么时间复杂度高，要么复杂，而且很容易注意到，其实他们都可以处理任意区间的相同问题，而不仅仅是前缀和区间的问题。

最最重要的是，他们都运用到了拆分区间的思想，所以。

# 雏形

## 查询

很明显，要做到 $\mathcal{O}(n \log n)$的时间复杂度，区间拆分要么是每次分成两半，要么二进制拆分，第一种是线段树的方式，所以我们考虑第二种。

考虑特殊样例吧。

像这样一个序列

```
1 2 3 4 5 6 7 8
```

考虑查询 $[1, 5]$ 。

$5_{10} = 101_2$

我们从低位开始减，具体原因后面解释，为了将原问题缩减为 $[1_2, 100_2]$ ，很明显要加上 $(100_2, 101_2]$ 区间。

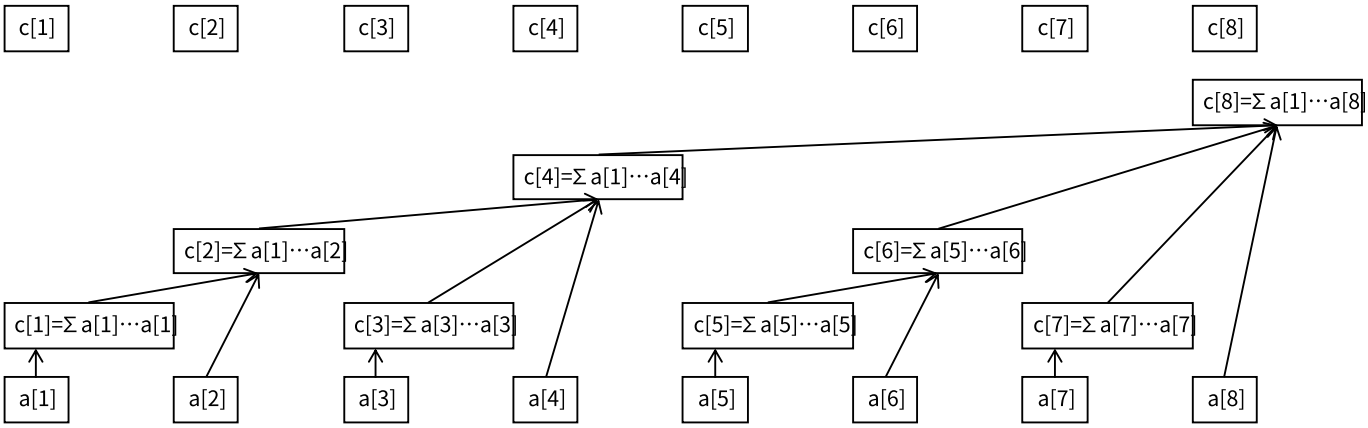
实际上，这种操作是在将右端点二进制形式下的最后一个 $1$ 消掉，然后将减小区间的数加到答案上，直到右端点为 $0$，无意义。

## 形式化

先不要心急讨论修改，我们先证明一些性质。

定义 $lowbit_i$ 为 $i$ 在二进制下将最后一个 $1$ 以及这个数位后面的 $0$ 拼起来的一个数，根据我们上面的信息，我们所加的区间一定是 $(i-lowbit_i,i]$ ，定义 $A$ 为原数组， $C$ 为处理后的树状数组，于是我们有 $C_i=\sum_{j=i-lowbit_i+1}^{i}A_i$ 。

考虑自下到上按照 $C_i$ 能管辖的范围，绘制出如下图。




给定一些结论

### 结论一

#### 结论

对于 $C_i$ ，一定有父亲 $C_{i+lowbit_i}$ 。

#### 证明

原结论的等同形式是对于 $i$ ，如果有一个最小的 $j$ 使得 $j-lowbit_j\le i-lowbit_i$ 。

这是很明显的， $i+lowbit_i$ 实际上是在将 $i$ 最后连续的 $1$ 去掉，把原来这些 $1$ 前的的 $0$ 变成 $1$ ，先令 $j$ 为 $i-lowbit_i$ , $j-lowbit_j$ 是将最后连续的 $1$ 去掉， $i-lowbit_i$ 是将最后一个 $1$ 去掉。

但是最小如何证明呢？ $i$ 加上一个数 $k$ $(0<k\le lowbit_i)$ ，实际上实在最后 $0$ 的部分加上若干个 $1$ ，再次减去 $lowbit_{i+k}$ 时，只会消掉一个 $1$ ，于是 $i+k-lowbit_{i+k}\ge i$ 。所以 $i+k-lowbit_{i+k}>i-lowbit_i$ 。

### 结论二

#### 结论

对于任意一个 $B_i$ ，若有 $1 \leq i \leq 2^n$ ，其祖先必定有 $B_{2^n}$ 。

#### 证明

当 $i = 1$ 时，结论显然。

并且若 $i = i \ \& \ 2^{n-1}$ 时命题成立（此处 $\&$ 为按位与）， 原命题一定成立，因为能走到 $2^{n-1}$，再跳一步直接进位到 $2^n$。

于是根据数学归纳法，原命题成立。Q.E.D.

### 修改

其实已经是顺水推舟的事情了，对于 $i$ 更改，只用从 $C_i$ 出发，一直向父亲走然后更改就行了，明显不可能更改 $C_j \ (j < i)$ ，而每个节点的父亲经证明就是 $C_{i+lowbit_i}$。

## 代码实现

### 关于lowbit

分离最低位采用二进制做法，$lowbit_i = i \ \& \ -i$ 。

因为计算机存储方式，取相反数是将所有位取反后加上 $1$ ，取反之后后面的 $0$ 变成 $1$ ，最后一个 $1$ 变成 $0$ ，在加上 $1$ 后，最后的 $1$ 全部进位为 $0$ 变回原来的，最后的 $0$ 接受进位后变成 $1$ ，不再向前进位而且也变回来了，于是按位与取相同部分即可。

### 查询

前面已经说得很清楚了，答案加上 $C_i$ 后，问题变成 $C_{i-lowbit_i}$ ，于是可以有：

```
int que(int x){
    int ans=0;
    while(x>0){
        ans=ans+tree[x];
        x-=lowbit(x);
    }
    return ans;
}
```

### 更改

树上一条链更改即可，每次跳父亲。

```
void add(int x,int y){
    while(x<=n){
        tree[x]+=y;
        x+=lowbit(x);
    }
    return ;
}
```