树状数组

Oct 26, 2018, Chaigidel

约定

$\lg x = \log_2 x$

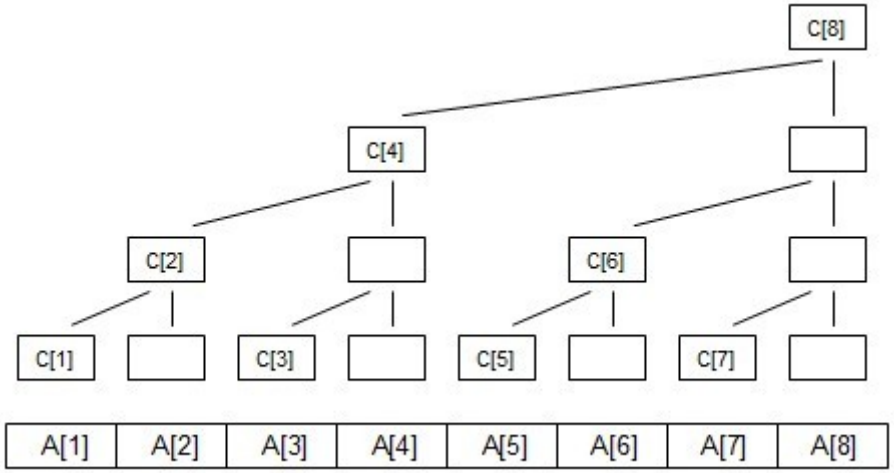
前缀和

有一数组 a[n]

定义 $sum[x] = \sum_{i=1}^{x} a_i$

显然，$\sum_{i=l}^{r} = sum[r] - sum[l]$

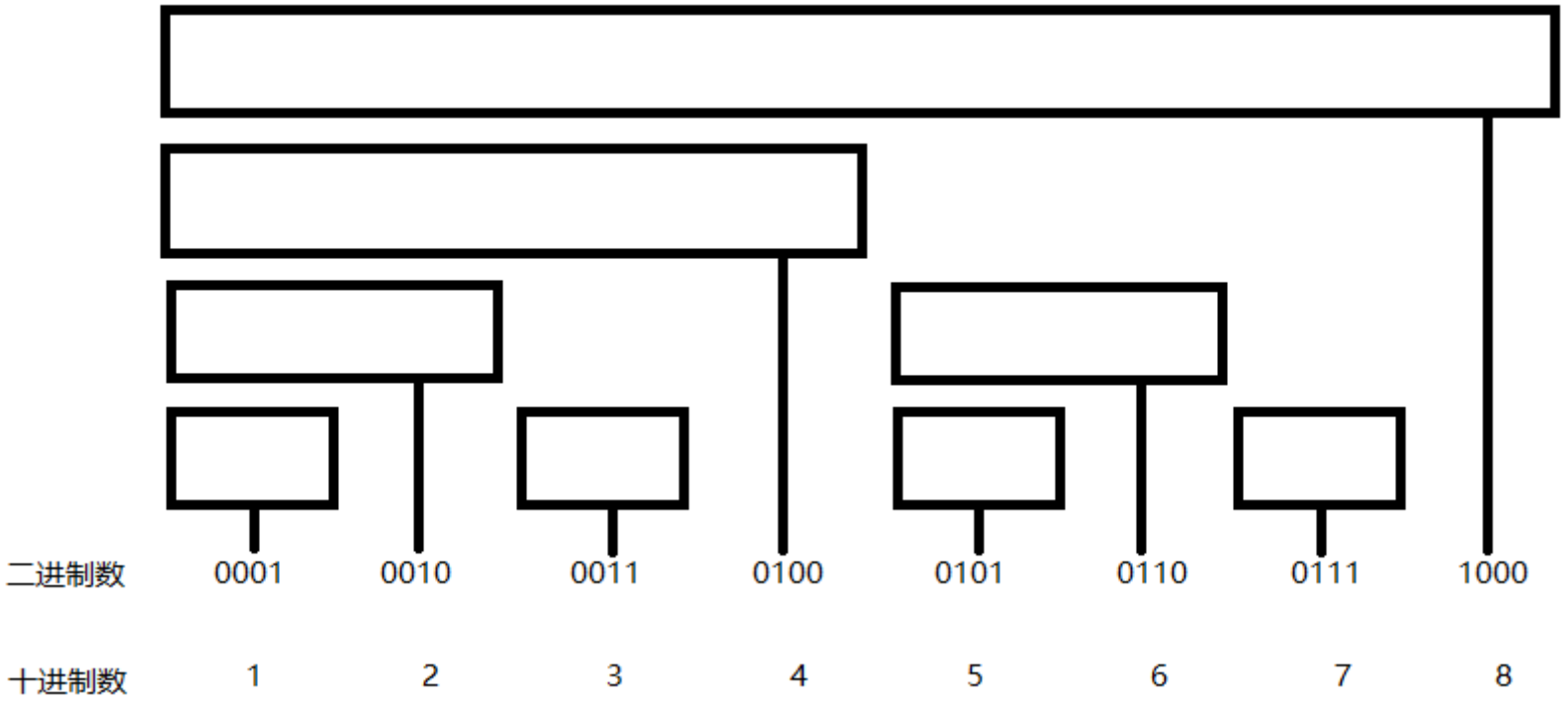
构造



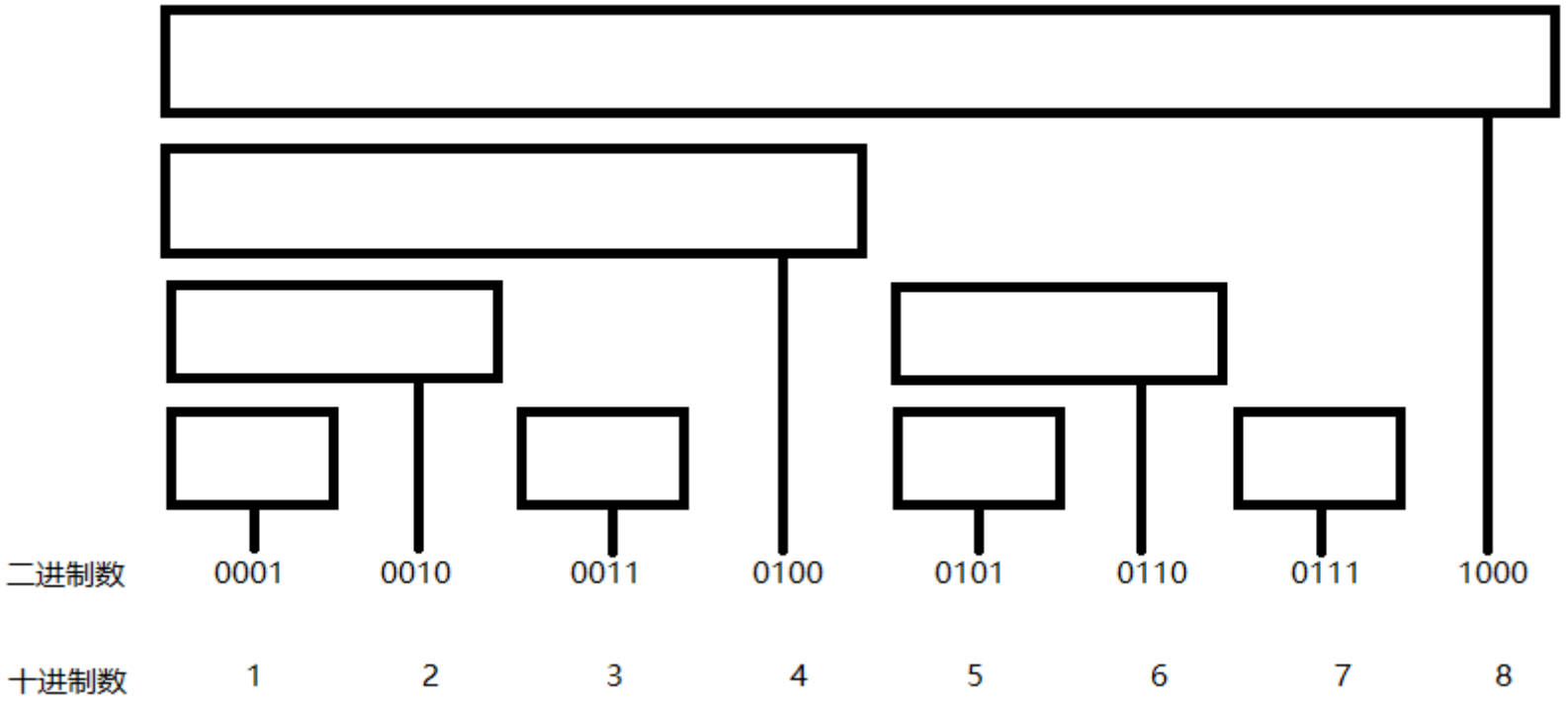
构造一个树型的结构，每个节点的值是子节点值的和

但我们 只存 一些特殊的点

仔细观察



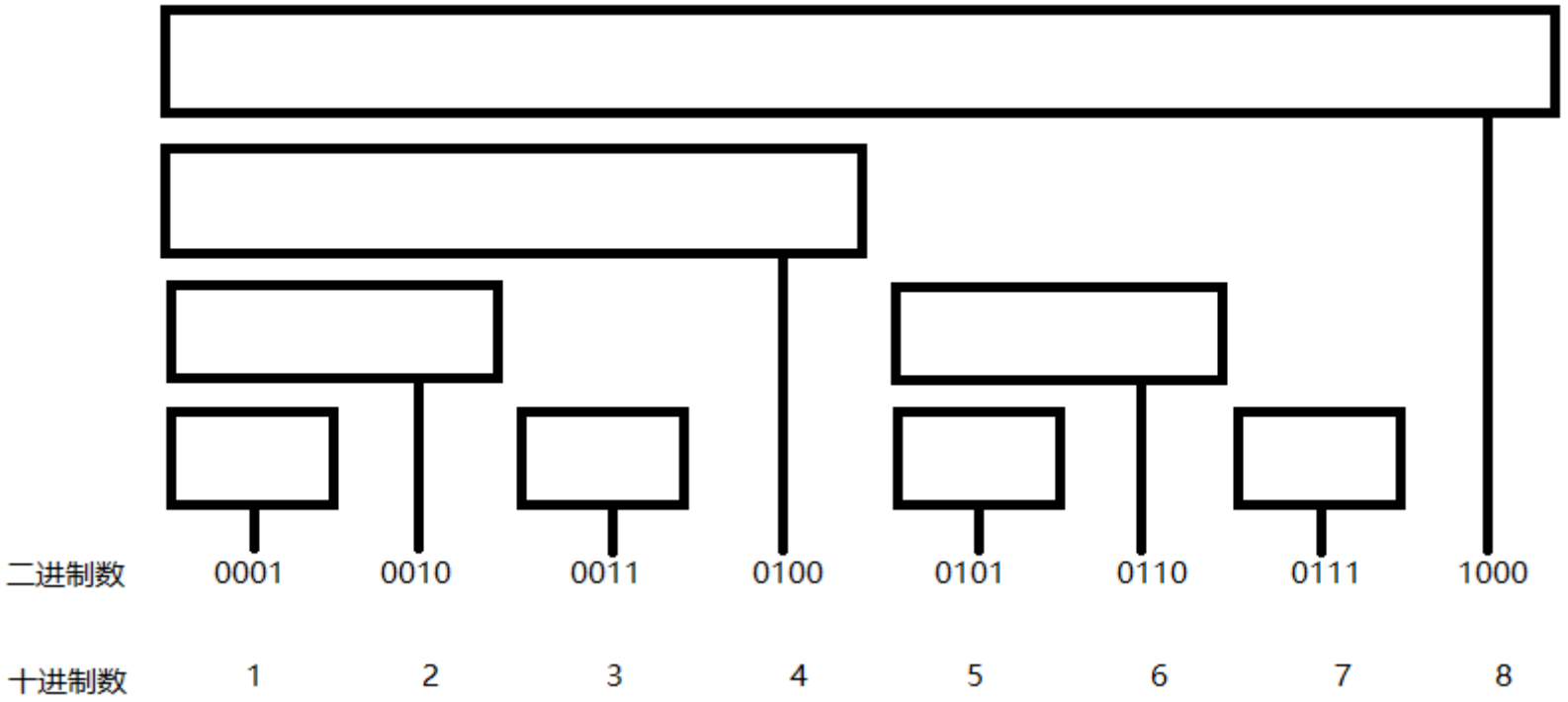
仔细观察



$g(x)$ 表示 x 的二进制末尾0的个数

$lowbit(x) = 2^{g(x)}$

仔细观察

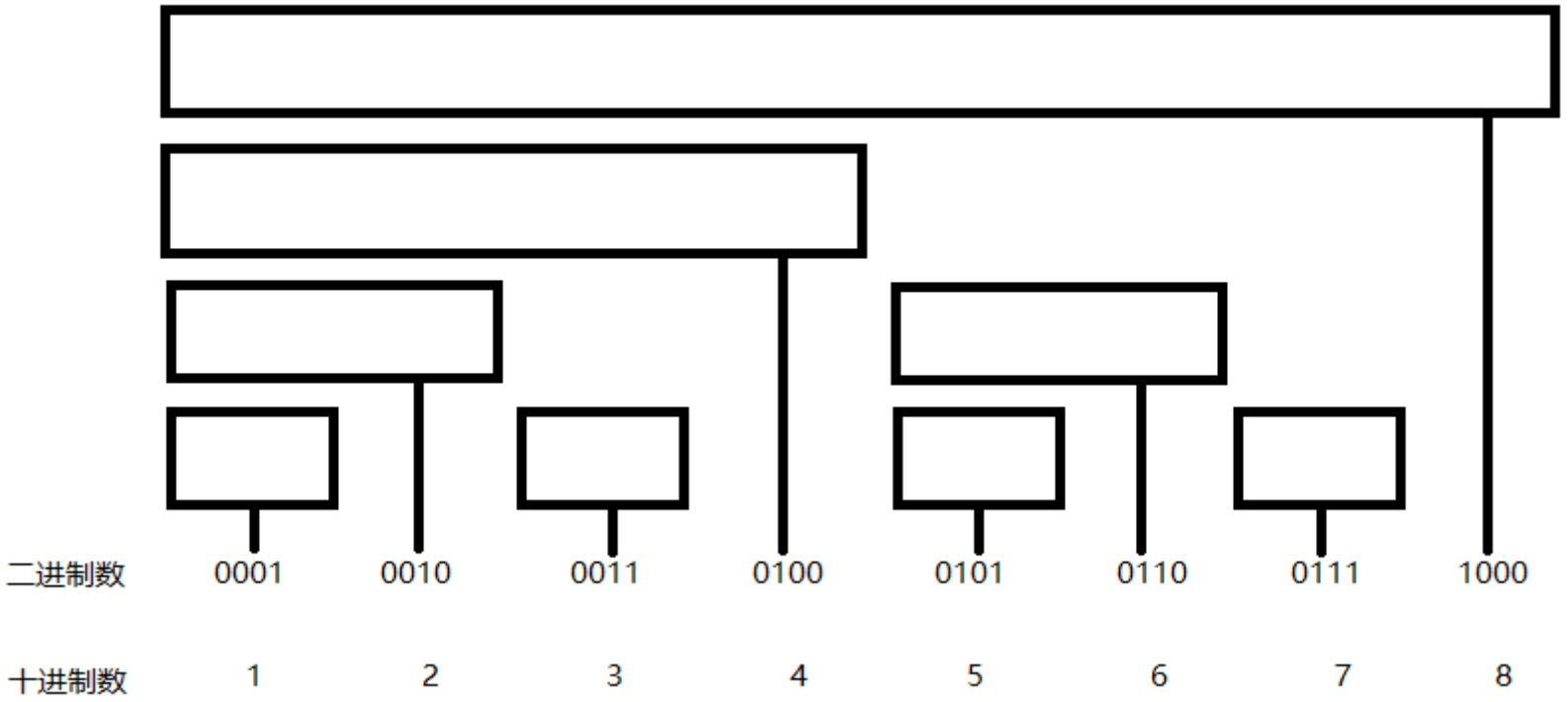


$g(x)$ 表示 x 的二进制末尾0的个数

$lowbit(x) = 2^{g(x)}$

第 x 段区间长度为 $lowbit(x)$ ？

仔细观察



$g(x)$ 表示 x 的二进制末尾0的个数

$lowbit(x) = 2^{g(x)}$

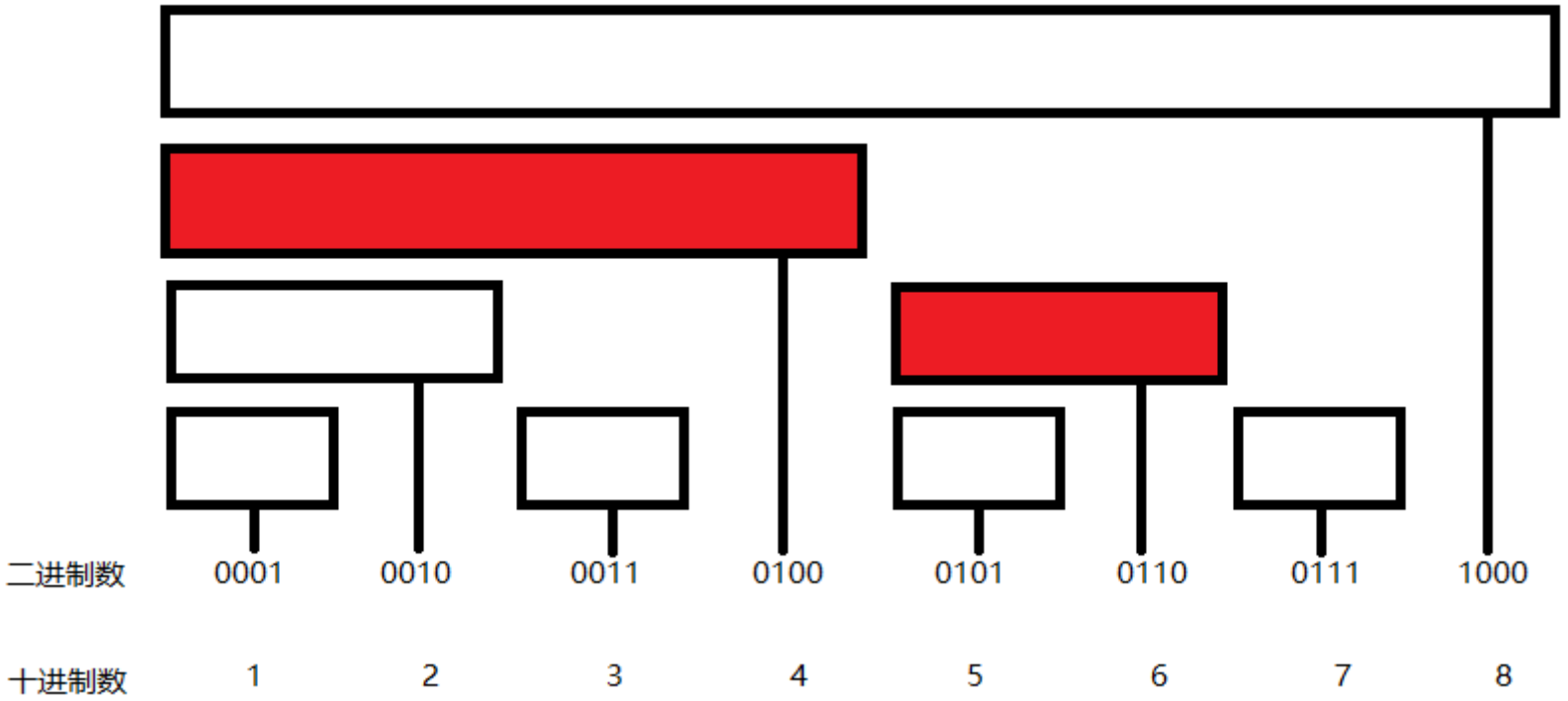
第 x 段区间长度为 $lowbit(x)$ ？

大胆猜测，不用证明

构造出来了 可是有什么用呢

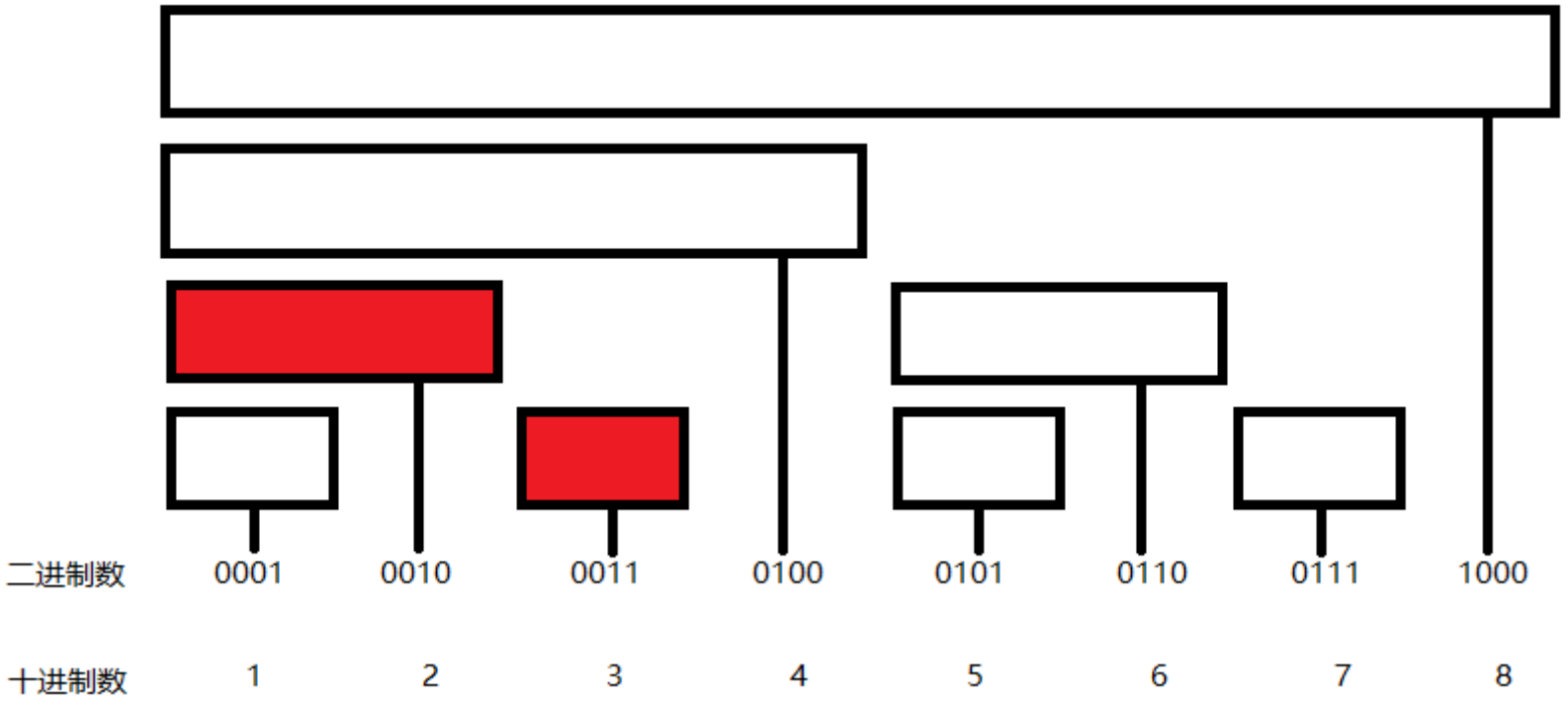
构造出来了 可是有什么用呢

查询 sum[6]



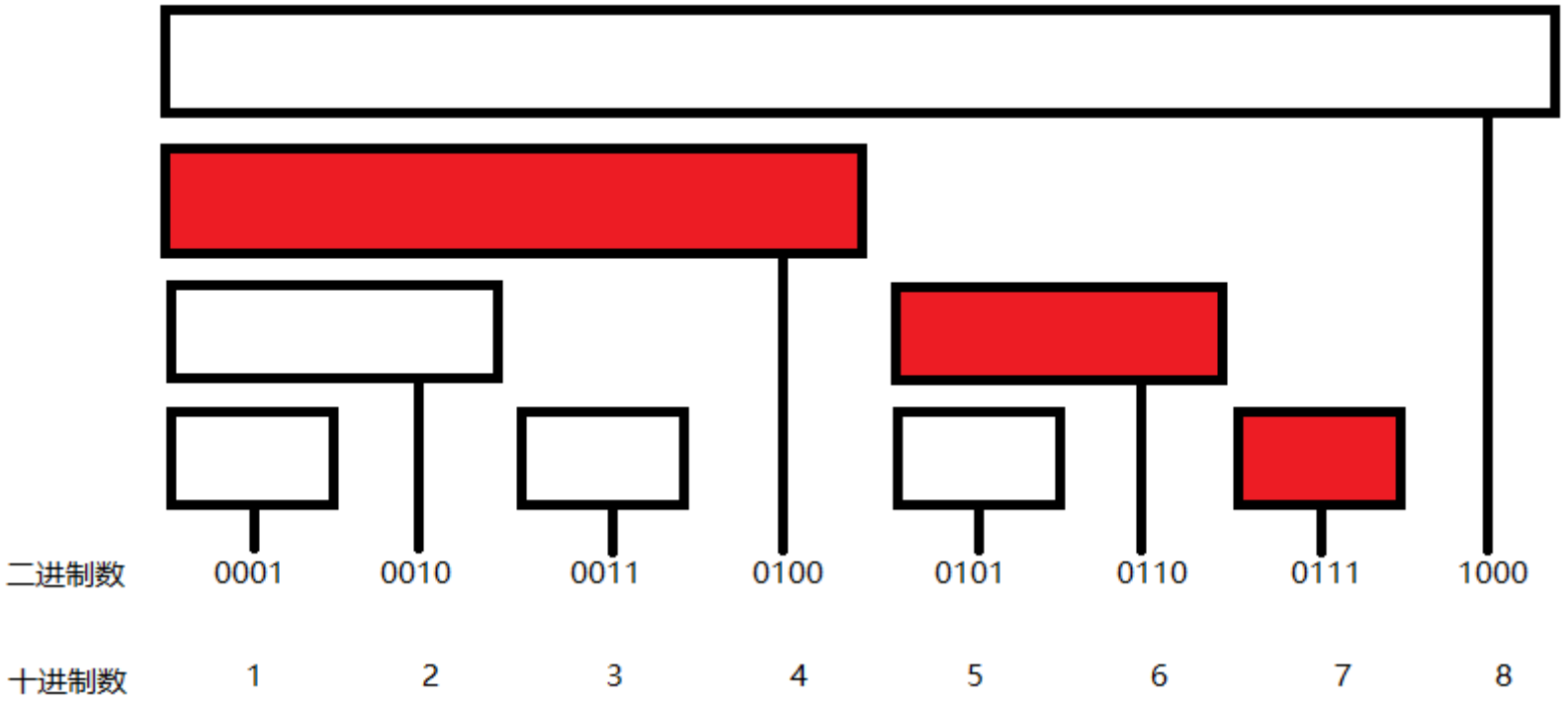
构造出来了 可是有什么用呢

查询 sum[3]



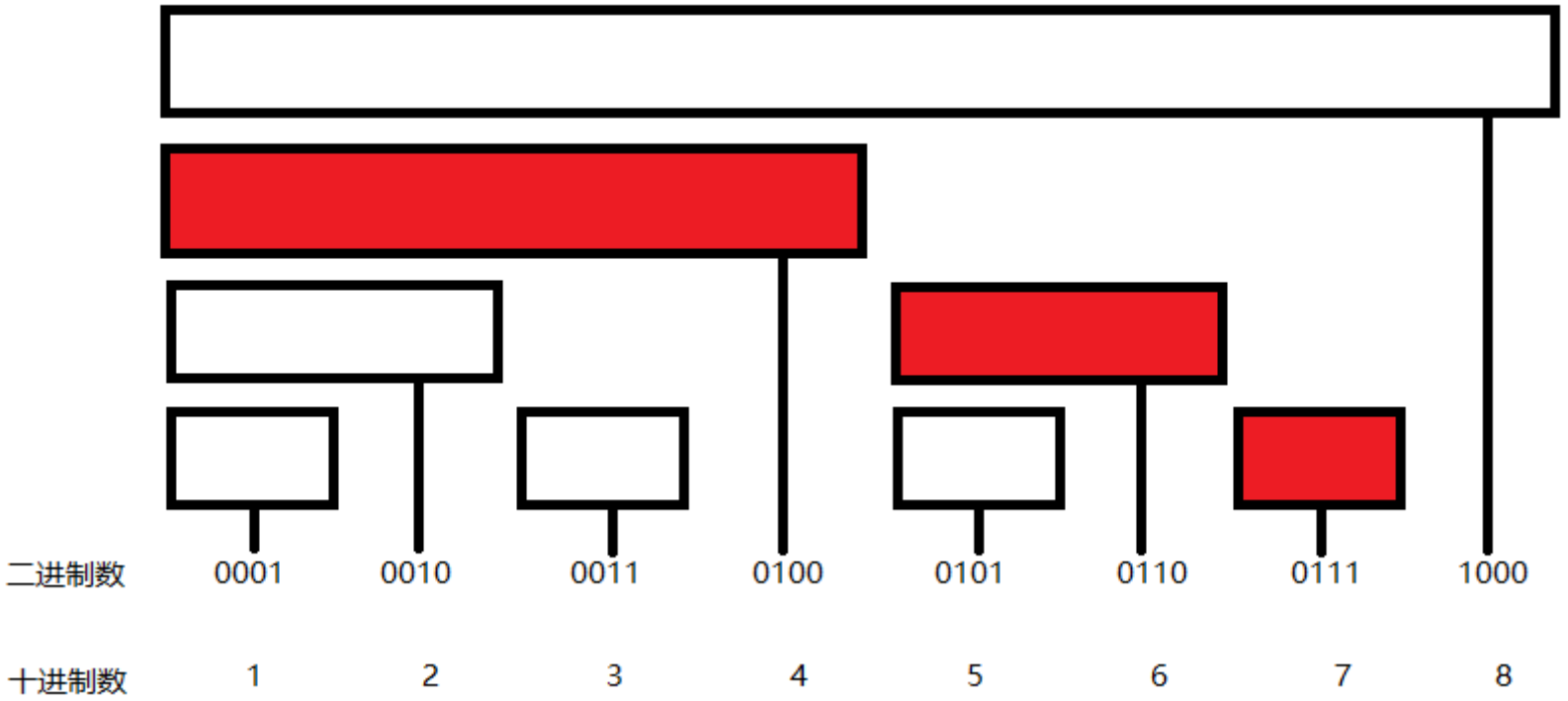
构造出来了 可是有什么用呢

查询 sum[7]



构造出来了 可是有什么用呢

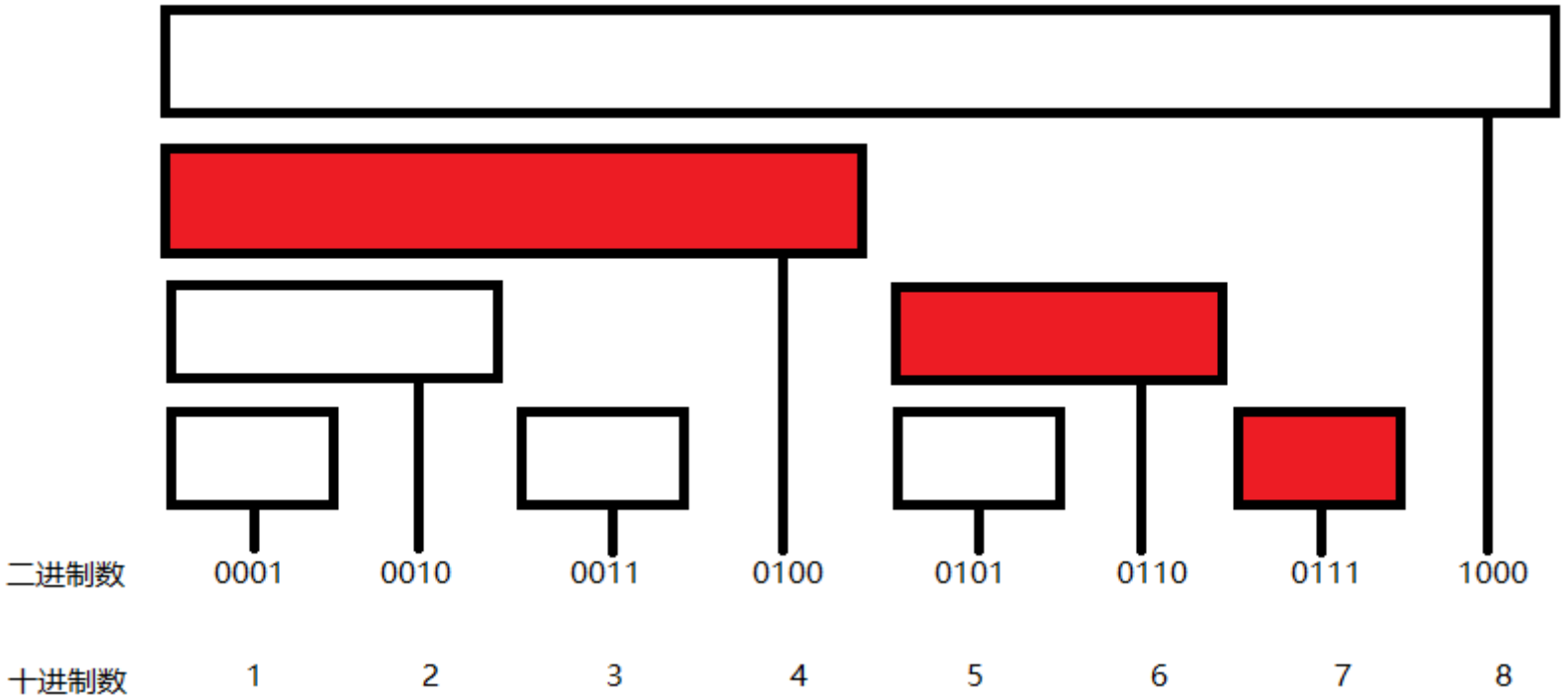
查询 sum[7]



似乎是每次加一个子树？ 子树覆盖的长度是多少？

构造出来了 可是有什么用呢

查询 sum[7]

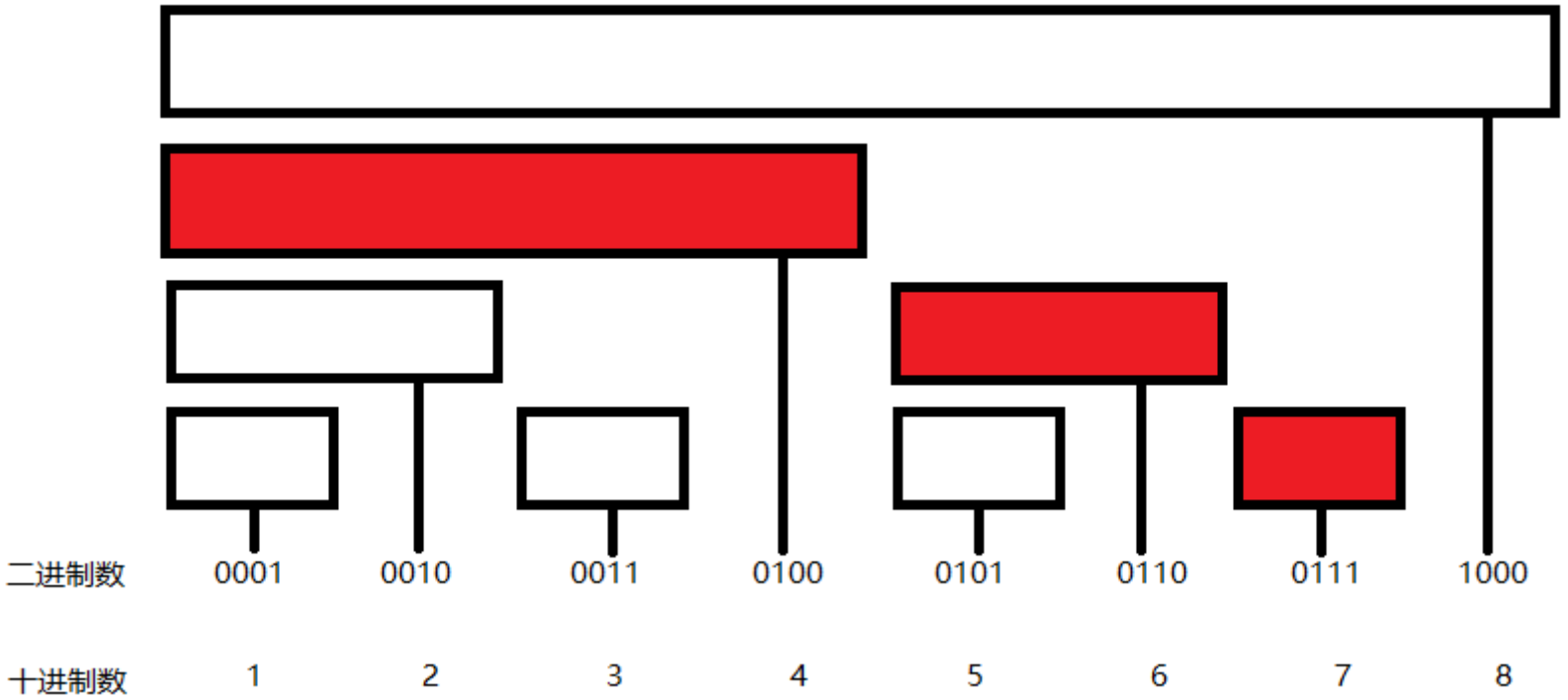


似乎是每次加一个子树？ 子树覆盖的长度是多少？

$lowbit(x)$

构造出来了 可是有什么用呢

查询 sum[7]



似乎是每次加一个子树？ 子树覆盖的长度是多少？

$lowbit(x)$

```
int ret;
for (; x > 0; x -= lowbit(x) )
    ret += c[x];
```

https://visualgo.net/en/fenwicktree

代码写出来了 可是 lowbit 怎么求呢

我会位运算！

```
int lowbit(int x){
    int ret = 1;
    while (!(x&1)){
        ret <<= 1;
        x >>= 1;
    }
    return ret;
}
```

代码写出来了 可是 lowbit 怎么求呢

我会位运算！

```
int lowbit(int x){
    int ret = 1;
    while (!(x&1)){
        ret <<= 1;
        x >>= 1;
    }
    return ret;
}
```

看起来就很慢

代码写出来了 可是 lowbit 怎么求呢

```
int lowbit(int x){
    return x&(-x);
}
```

蛤？

代码写出来了 可是 lowbit 怎么求呢

```
int lowbit(int x){
    return x&(-x);
}
```

蛤？

由于 $x > 0$

举个例子

x = 010011000 第一位符号位

+x 补码是 010011000

-x 原码是 110011000

-x 反码是 101100111

-x 补码是 101101000

由于内部储存的是补码

x & (-x) = 1000

我用前缀和直接搞定的事为啥用树状数组？

我用前缀和直接搞定的事为啥用树状数组？

前缀和的动态修改复杂度为 $O(n)$

我用前缀和直接搞定的事为啥用树状数组？

前缀和的动态修改复杂度为 $O(n)$

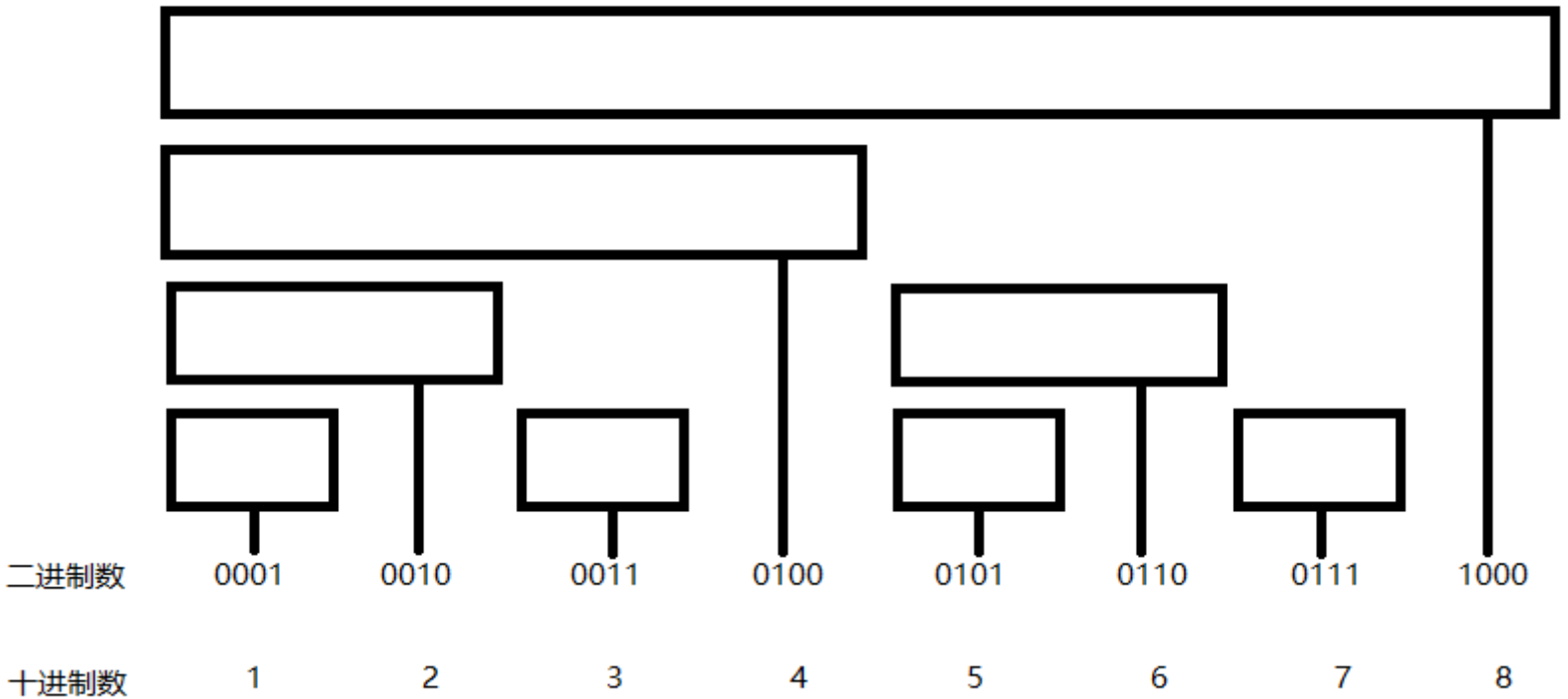
树状数组 $O(\lg n)$！

我用前缀和直接搞定的事为啥用树状数组？

前缀和的动态修改复杂度为 $O(n)$

树状数组 $O(\lg n)$!

举个栗子



修改 a[6] 只需修改 c[6]，c[8]

修改 a[1] 只需修改 c[1]，c[2]，c[4]，c[8]

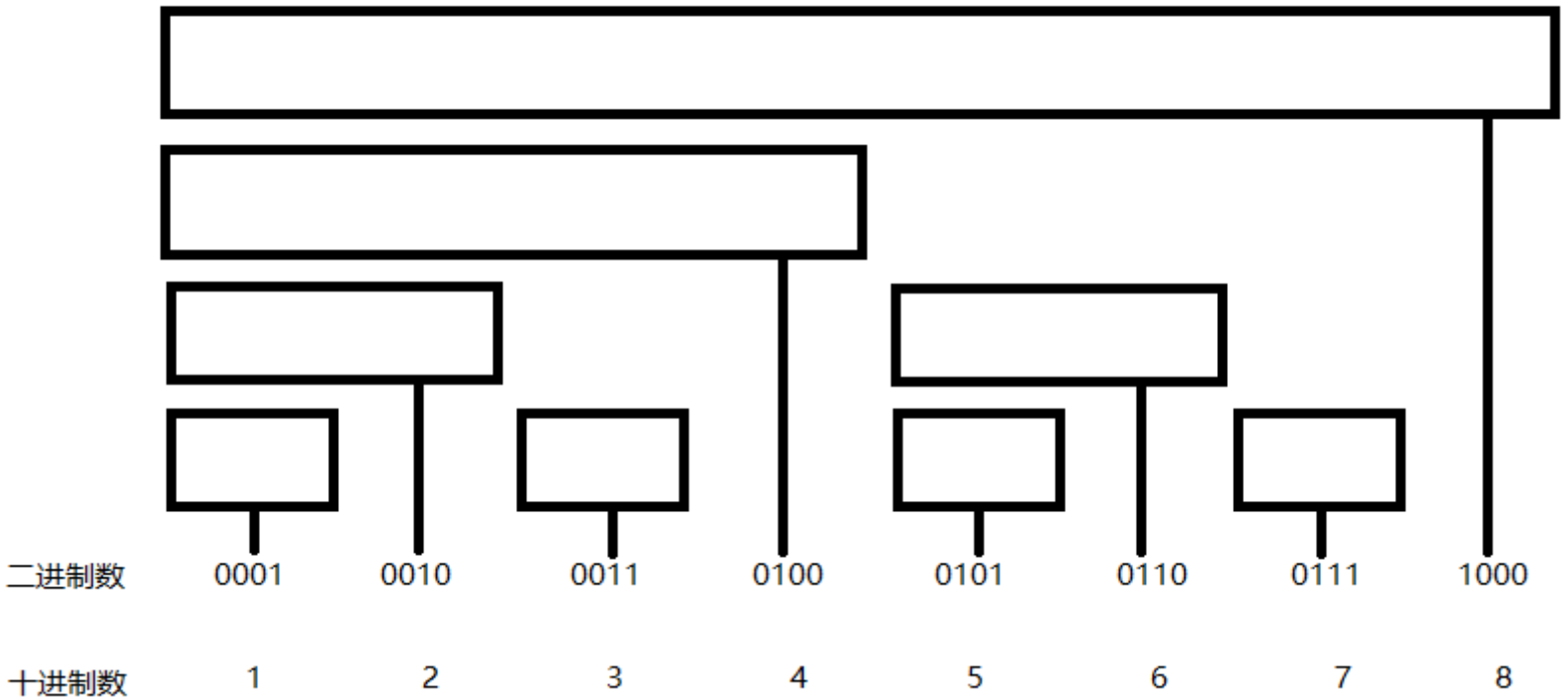
修改 a[3] 只需修改 c[4]，c[8]

我用前缀和直接搞定的事为啥用树状数组？

前缀和的动态修改复杂度为 $O(n)$

树状数组 $O(\lg n)$!

举个栗子



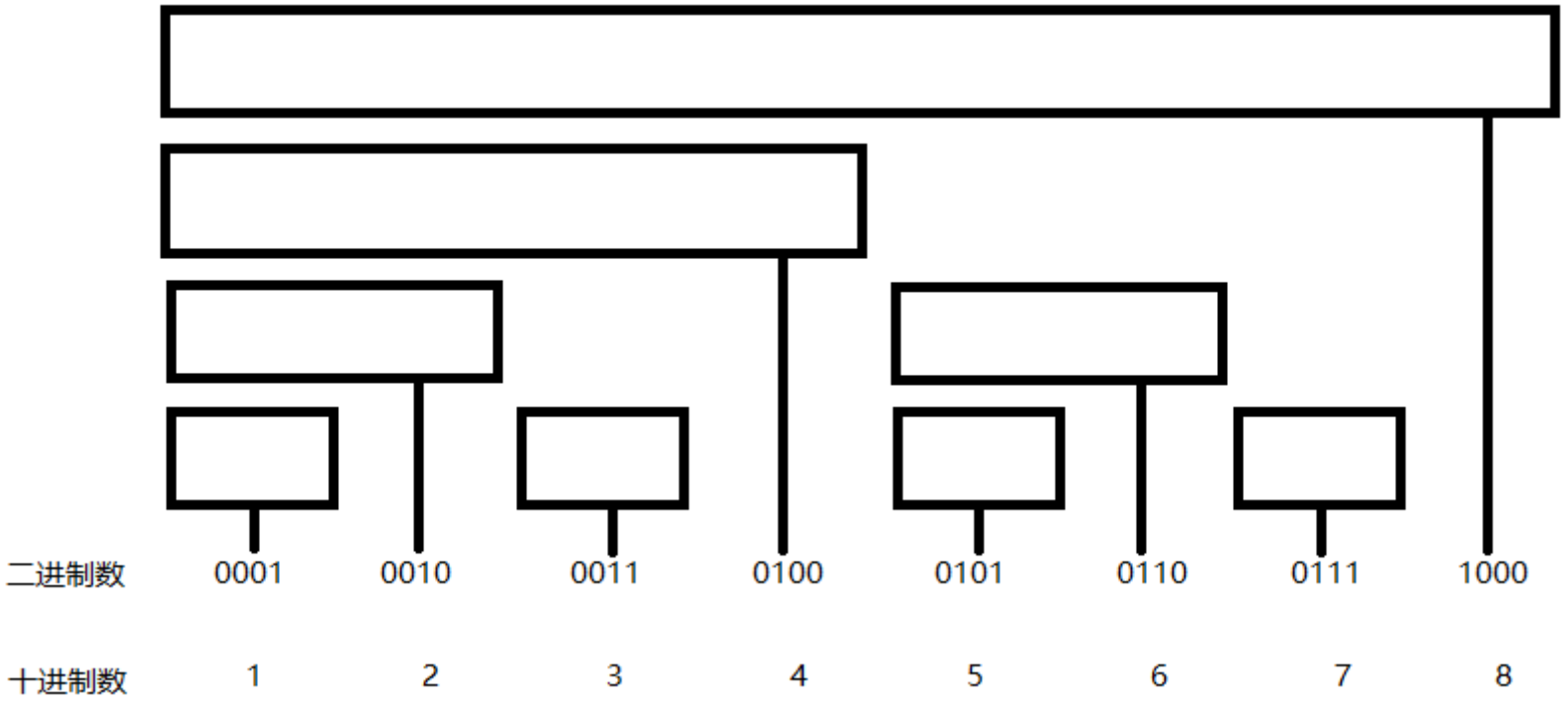
修改 a[6] 只需修改 c[6]，c[8]

修改 a[1] 只需修改 c[1]，c[2]，c[4]，c[8]

修改 a[3] 只需修改 c[4]，c[8]

似乎又和 lowbit(x) 有不为人知的关系？

我用前缀和直接搞定的事为啥用树状数组？



修改 a[6] 只需修改 c[6]，c[8]

修改 a[1] 只需修改 c[1]，c[2]，c[4]，c[8]

修改 a[3] 只需修改 c[4]，c[8]

```
for (; x <= n; x += lowbit(x))
    c[x] += add;
```

https://visualgo.net/en/fenwicktree

综上

```
int a[N], c[N];
int lowbit(int x) { return x&-x; }
void change(int pos, int x){
    int add = x - a[pos];
    a[pos] = x;
    for (int i = pos; i <= n; i += lowbit(i))
        c[i] += add;
}
void build(int n){
    for (int i = 1; i <= n; ++i)
        add(i, a[i]);
}
int sum(int n){
    int ret = 0;
    for (int i = n; i > 0; i -= lowbit(i))
        ret += c[i];
    return ret;
}
int query(int l, int r){
    return sum(r) - sum(l);
}
```