树状数组介绍

主讲人：邓哲也

树状数组的引入

与线段树类似，树状数组也是用于动态查询区间信息、支持修改的数据结构。

与线段树相比，它更简洁，实现起来也更方便，但是它能维护的信息比较有限。

树状数组的引入

树状数组是在 1994 年由 Peter M. Fenwick 在他的论文《A New Data Structure for Cumulative Frequency Tables》中率先提出的，所以又称 Fenwick Tree。

树状数组的引入

树状数组处理的问题一般有如下形式：

给定一个数组 a[1..n]，支持以下两种操作：

1. 修改：给 a[i] 加上 v
2. 查询：询问 a[1] + a[2] + … + a[i]

树状数组的思想

每个正整数都可以表示为若干个 2 的幂次之和。

类似的，每次求前缀和，我们也希望将区间 [1,i] 分解成 $\log_2 i$ 个子集的和。

也就是如果 i 的二进制表示中如果有 k 个 1，我们就希望将其分解为 k 个子集之和。

树状数组的思想

基于这种思想，我们可以构造一张表格：

内容代表该“子集”所包含的 a 数组的元素。

例如，下标为 8 的子集包含了 a[1..8]，而下标为 5 的子集只包含 a[5]。

下标	1	2	3	4	5	6	7	8	9	10	11	12
内容	1	1..2	3	1..4	5	5..6	7	1..8	9	9..10	11	9..12

树状数组的思想

下表就是一个实际的例子：

用 sum 来表示子集和。

比如求 7 的前缀和，只需要计算 sum[7]+sum[6]+sum[4]

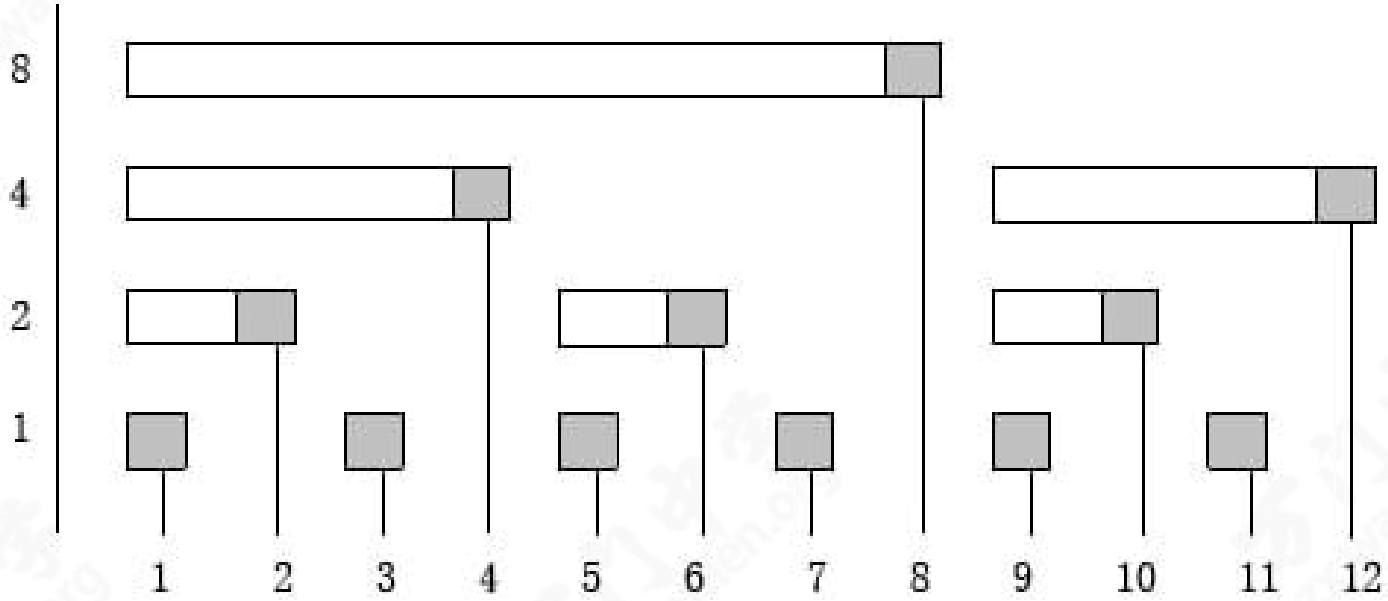
求 10 的前缀和，只需要计算 sum[10]+sum[8]

下标	1	2	3	4	5	6	7	8	9	10	11	12
a数组	1	2	0	1	0	0	2	2	1	2	1	3
前缀和	1	3	3	4	4	4	6	8	9	11	12	15
sum	1	3	0	4	0	0	2	8	1	3	1	7

树状数组的思想

我们可以用下图来更直观的理解：

深色方块代表自己下标对应的值 a[i]，浅色代表还要维护的别的下标对应的值 a[k .. i-1]



树状数组的实现

现在留下的问题就是，子集要如何划分。

观察下表。

下标	1	2	3	4	5	6	7	8
下标的二进制	1	10	11	100	101	110	111	1000
内容	1	1..2	3	1..4	5	5..6	7	1..8
元素个数	1	2	1	4	1	2	1	8
元素个数的二进制	1	10	1	100	1	10	1	1000

可以发现，元素个数的二进制就是下标的二进制表示中最低位的 1 所在的位置对应的数。

树状数组的实现

如何求一个数 x 的二进制最低非0位对应的数？

朴素方法：$O(\log_2 x)$枚举

树状数组的实现

如何求一个数 x 的二进制最低非0位对应的数？

Lowbit(x) 函数！

C(x) = x - (x and (x - 1))

x and (x - 1) 将 x 最低位的 1 以及后面所有的 0 都变成了 0，然后再被 x 减去，这就是我们要的数。

树状数组的实现

如何求一个数 x 的二进制最低非0位对应的数？

Lowbit(x) 函数！

C(x) = x and -x

计算机里存储整数用的是数字的补码。

正数的补码就是本身的二进制码。

相反数的补码等于其反码加一。

假设 x 的二进制为 a1b（a为01串，b全是0）。

-x 就是 (~a)1b，两者 and 就得到了 1b，就是我们想要的数。

下节课再见