

区间，数组，矩阵和树状数组 Interval, Array, Matrix & Binary Indexed Tree

课程版本 v5.0 主讲 令狐冲



扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: [ninechapter](#)

微博: <http://www.weibo.com/ninechapter>

知乎: <http://zhuankan.zhihu.com/jiuzhang>

官网: <http://www.jiuzhang.com>

本章节, 我们将要学习和区间, 数组, 矩阵有关的一些问题, 以及一个最近兴起的数据结构——树状数组(Binary Indexed Tree)

请在随课教程中先修如下知识:

- 如何用 Comparator 对区间进行排序
- 在排好序的区间序列中插入一个段新区间
- 快速选择算法 Quick Select(返回第三章复习)
- K 路归并算法(K-way Merge Algorithm)
- 子数组与前缀和(Subarray & Prefix Sum)
- 如何用位运算来数二进制中 1 的个数

大纲:

- Merge Two / K Sorted Arrays / Intervals 相关算法即拓展题
- Median of Unsorted / Two Sorted / K Sorted Arrays 相关算法及拓展题
- 通过 Range Sum Query 学习 Binary Indexed Tree

Merge Two Sorted Arrays

<http://www.lintcode.com/problem/merge-two-sorted-arrays>

<http://www.jiuzhang.com/solutions/merge-two-sorted-arrays/>

Follow up I

把小数组 Merge 到有足够空余空间的大数组里

<http://www.lintcode.com/problem/merge-sorted-array/>

<http://www.jiuzhang.com/solutions/merge-sorted-array/>

Follow Up II

<http://www.lintcode.com/problem/merge-two-sorted-interval-lists/>

<http://www.jiuzhang.com/solutions/merge-two-sorted-interval-lists/>

归并 2 个有序的区间序列

Follow up III

归并 K 个数组

<http://www.lintcode.com/problem/merge-k-sorted-arrays/>

<http://www.jiuzhang.com/solutions/merge-k-sorted-arrays/>

k 路归并算法 (外排序算法 External Sorting)

Follow up IV

<http://www.lintcode.com/problem/merge-k-sorted-interval-lists/>

<http://www.jiuzhang.com/solutions/merge-k-sorted-interval-lists/>

归并 K 个有序区间序列

Intersection of Two Arrays

<http://www.lintcode.com/problem/intersection-of-two-arrays/>

<http://www.jiuzhang.com/solutions/intersection-of-two-arrays/>

求两个数组的交集

Follow up I

<http://www.lintcode.com/problem/intersection-of-two-arrays-ii/>

<http://www.jiuzhang.com/solutions/intersection-of-two-arrays-ii/>

如何有重复如何处理？

Follow up II

<http://www.lintcode.com/problem/intersection-of-arrays/>

<http://www.jiuzhang.com/solutions/intersection-of-arrays/>

多个数组如何处理？

不用 HashMap 怎么做？

Sparse Matrix Multiplication

<http://www.lintcode.com/problem/sparse-matrix-multiplication>

<http://www.jiuzhang.com/solutions/sparse-matrix-multiplication>

简化版：如何加速稀疏向量 (Sparse Vector) 的乘法？

与 Interval 有关的重要算法

将在《九章算法强化班》中讲解

扫描线算法 (Sweep Line Algorithm)

Median

求数组的中位数

Follow up I: 没排序数组的第 k 大

用我们之前学过的 Quick Select 算法就可以轻松解决

啥不记得？去随课教程看第三章双指针算法

Median of Two Sorted Arrays

<http://www.lintcode.com/problem/median-of-two-sorted-arrays>

<http://www.jiuzhang.com/solutions/median-of-two-sorted-arrays>

FindMedian = FindKth($k=n/2$)

Follow up I

<http://www.lintcode.com/problem/median-of-k-sorted-arrays/>

<http://www.jiuzhang.com/problem/median-of-k-sorted-arrays/>

求 K 个排序数组的中位数

Follow up II

假如 K 个排序数组位于不同的机器上
如何利用机器的并发性提高算法效率？

更多二分答案算法

将在《九章算法强化班》中讲解

Binary Search on Answer

休息 5 分钟

Maximum Subarray

<http://www.lintcode.com/problem/maximum-subarray/>

<http://www.jiuzhang.com/solution/maximum-subarray/>

换个皮: [Best time to buy and sell stock](#)

Submatrix Sum

<http://www.lintcode.com/problem/submatrix-sum/>

<http://www.jiuzhang.com/solutions/submatrix-sum/>

Maximum Submatrix

<http://www.lintcode.com/problem/maximum-submatrix/>

<http://www.jiuzhang.com/solutions/maximum-submatrix/>

如何转换成 Maximum Subarray ?

Range Sum Query Immutable

<http://www.lintcode.com/problem/range-sum-query-immutable/>

<http://www.jiuzhang.com/solutions/range-sum-query-immutable/>

Range Sum Query 2D Immutable

<http://www.lintcode.com/problem/range-sum-query-2d-immutable/>

<http://www.jiuzhang.com/solutions/range-sum-query-2d-immutable>

Range Sum Query Mutable

<http://www.lintcode.com/problem/range-sum-query-mutable/>

<http://www.jiuzhang.com/solutions/range-sum-query-mutable/>

Binary Indexed Tree

又名:Fenwick Tree 中文名:树状数组 简写:BIT

基于“前缀和”信息来实现——

$\text{Log}(n)$ 修改任意位置值

$\text{Log}(n)$ 查询任意区间和

Binary Indexed Tree 的特性

功能特性:

对于一个有 N 个数的数组, 支持如下功能:

$\text{update}(\text{index}, \text{val})$ // $\log N$ 的时间内更新数组中一个位置上的值

$\text{getPrefixSum}(k)$ // $\log(K)$ 的时间内获得数组中前 K 个数的和

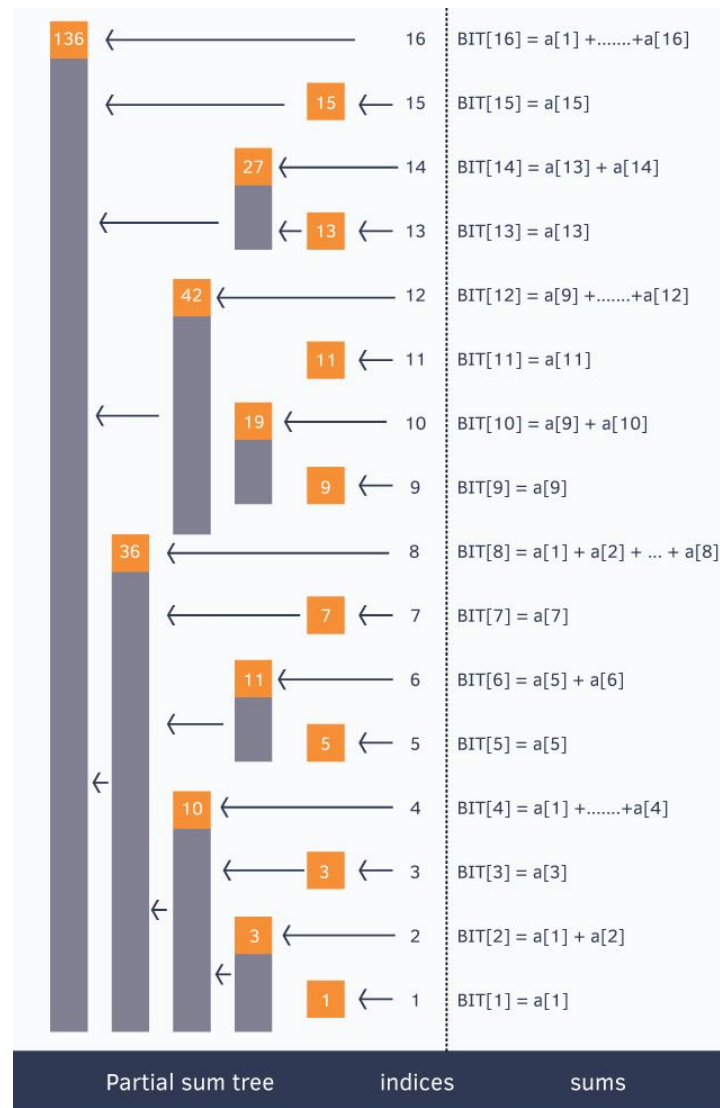
问: 如何用 $\text{getPrefixSum}(k)$ 实现 $\text{getRangeSum}(x, y)$?

实现特性:

虽然名字叫做 Tree, 但是是用数组(Array)存储的

BIT 是一棵**多叉树**, **父子关系代表包含关系**

BIT的第0位空出来, 没有用上



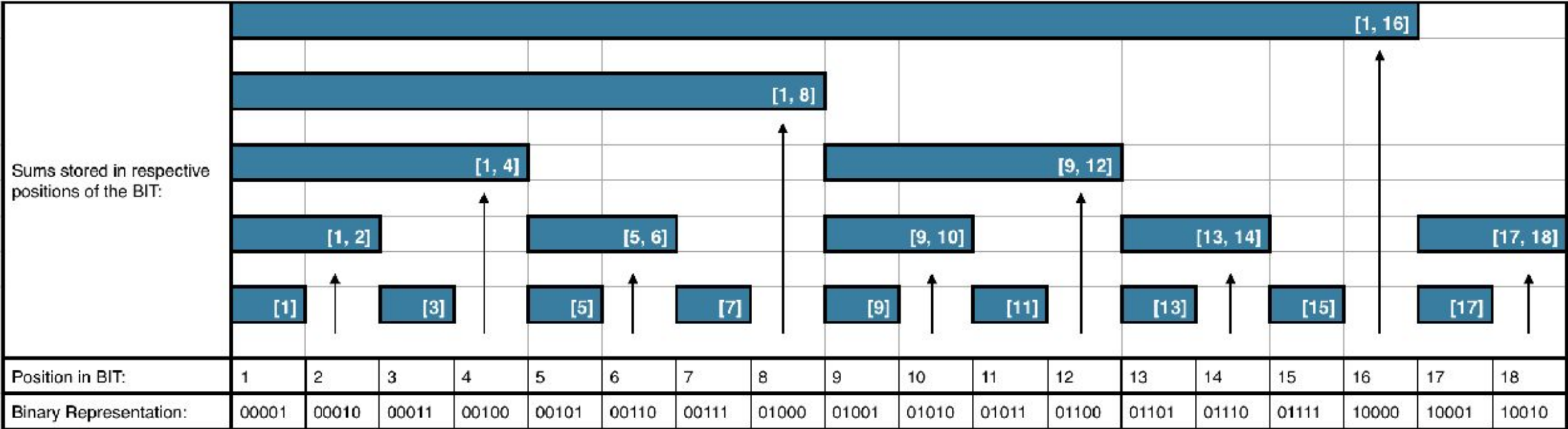
The value in the enclosed box represents BIT[index].

假设原数组为 A, 树状数组为 BIT

$BIT[16] = A[1] + A[2] \dots + A[16]$

$BIT[15] = A[15]$

...



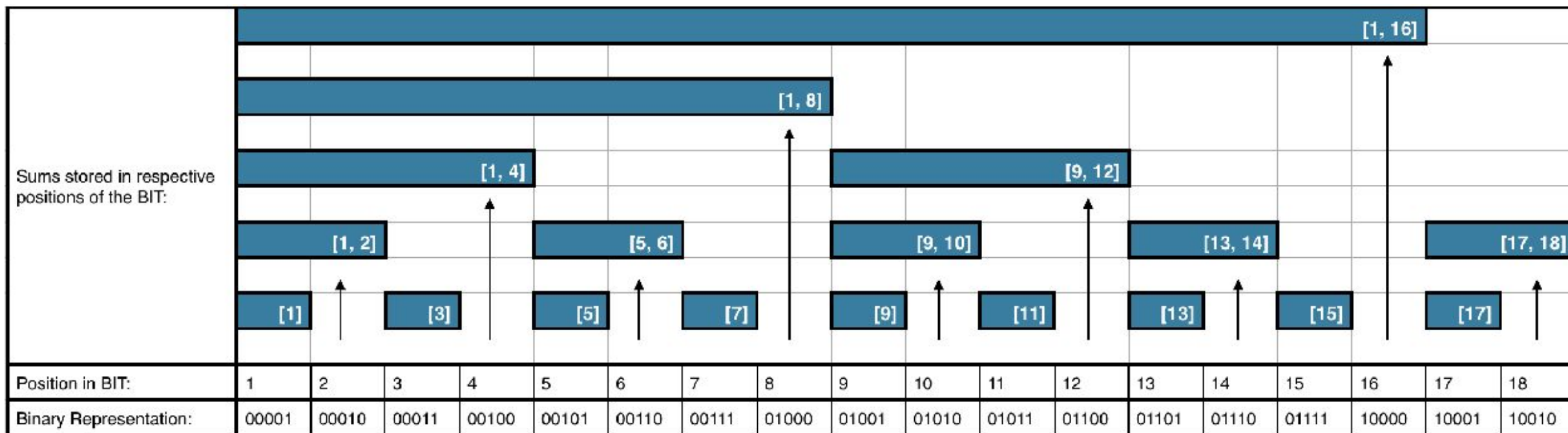
如何求前缀和？

$\text{getPrefixSum}(16) = \text{BIT}[16]$

$\text{getPrefixSum}(15) = \text{BIT}[15] + \text{BIT}[14] + \text{BIT}[12] + \text{BIT}[8]$

$\text{getPrefixSum}(14) = \text{BIT}[14] + \text{BIT}[12] + \text{BIT}[8]$

$\text{getPrefixSum}(13) = \text{BIT}[13] + \text{BIT}[12] + \text{BIT}[8]$



01111	01110	01100	01000
-------	-------	-------	-------

$\text{getPrefixSum}(15) = \text{BIT}[15] + \text{BIT}[14] + \text{BIT}[12] + \text{BIT}[8]$

10010	10000
-------	-------

$\text{getPrefixSum}(18) = \text{BIT}[18] + \text{BIT}[16]$

10101	10100	10000
-------	-------	-------

$\text{getPrefixSum}(21) = \text{BIT}[21] + \text{BIT}[20] + \text{BIT}[16]$

猜猜看, $\text{getPrefixSum}(k)$ 等于 BIT 数组中哪些数之和是如何计算的?

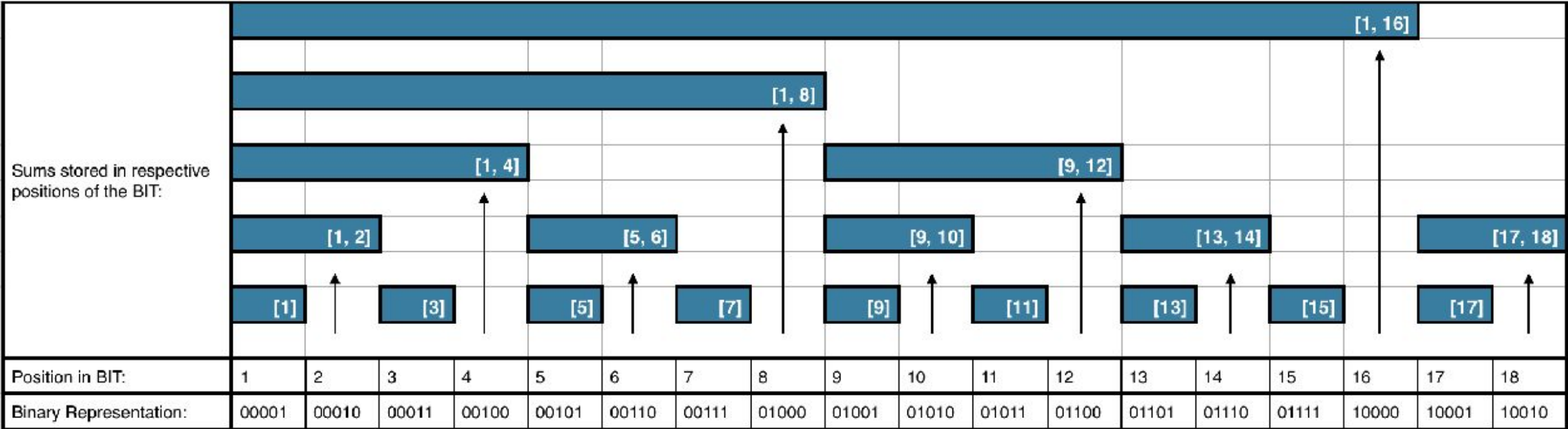
lowbit 最后一个二进制位

$$\text{lowbit}(x) = x \& (-x)$$

$$\begin{array}{r} \dots 01010110000\dots \\ \& \dots 10101010000\dots \\ \hline \dots 00000010000\dots \end{array}$$

如何更新？

- 当 $A[x]$ 被更新时，哪些 BIT 中的数会受到影响？
- 更改了 $A[1]$ ，受到影响的是 $BIT[1], BIT[2], BIT[4], BIT[8], BIT[16] \dots$
- 更改了 $A[5]$ ，受到影响的是 $BIT[5], BIT[6], BIT[8], BIT[16] \dots$
- 更改了 $A[9]$ ，受到影响的是 $BIT[9], BIT[10], BIT[12], BIT[16] \dots$



01111

10000

update(15) => BIT[15], BIT[16]

00001

00010

00100

01000

10000

update(1) => BIT[1], BIT[2], BIT[4], BIT[8], BIT[16]

01001

01010

01100

10000

update(9) => BIT[9], BIT[10], BIT[12], BIT[16]

这里 => 符号代表影响到的BIT里需要被更新的位置
猜猜看, 更改序列是如何得到的?

BIT的两个操作总结

getPrefixSum(x)

不断的做 $x = x - \text{lowbit}(x)$ 直到 $x = 0$

update(x, val)

- 计算出 $\text{delta} = \text{val} - A[x]$ 也就是增量
- 从 x 开始, 不断的将 $\text{BIT}[x] += \text{delta}$, 然后 $x = x + \text{lowbit}(x)$, 直到 $x > N$

```
public void update(int index, int val) {
    int delta = val - arr[index];
    arr[index] = val;

    for (int i = index + 1; i <= arr.length; i = i + lowbit(i)) {
        bit[i] += delta;
    }
}

public int getPrefixSum(int index) {
    int sum = 0;
    for (int i = index + 1; i > 0; i = i - lowbit(i)) {
        sum += bit[i];
    }
    return sum;
}

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

Range Sum Query 2D Mutable

<http://www.lintcode.com/problem/range-sum-query-2d-mutable/>

<http://www.jiuzhang.com/solutions/range-sum-query-2d-mutable/>

2D BIT 可能难以想象, 但是代码很容易记

Binary Indexed Tree 事实上就是一个有**部分区段累加和数组**

把原先我们累加的方式从：

```
for (int i = index; i >= 0; i = i - 1) sum += arr[i];
```

改成了

```
for (int i = index+1; i >= 1; i = i - lowbit(i)) sum += bit[i];
```

这样我们很容易将这个算法拓展到 2D, 3D ...

线段树 Segment Tree

可以取代 Binary Indexed Tree, 功能更强大的数据结构

九章微课堂《线段树入门》

<http://www.jiuzhang.com/tutorial/segment-tree>

线段树 (Segment Tree) 特别万能
考得不多，但是如果能掌握，可以一口气解决一大堆问题

	Binary Indexed Tree	Segment Tree	Heap	Balanced BST
区间和	$O(\log N)$	$O(\log N)$	不支持	不支持
区间最大值/最小值	不支持	$O(\log N)$	不支持	不支持
所有数最大值/最小值	$O(\log N)$	取值 $O(1)$ 更新 $O(\log N)$	取值 $O(1)$ 更新 $O(\log N)$	$O(\log N)$
比某个数大的最小值	不支持	$\log(\text{Range})$	不支持	$O(\log N)$
比某个数小的最大值	不支持	$\log(\text{Range})$	不支持	$O(\log N)$
实现难度	简单，记不住	中等，容易记住	中等，老写错	难，不会写