

# Fenwick Tree (Binary Index Tree)

日月卦長



# 經典題

- 給你一個長度為  $n$  的陣列  $a$ ，再給你  $q$  個操作，操作有兩種：
- $sum(ql, qr)$ :  
查詢  $a_{ql} + a_{ql+1} + \dots + a_{qr}$  的值
- $add(p, val)$ :  
將  $a_p += val$
- $1 \leq n, q \leq 10^6$

$a$

0	1	2	3	4	5	6	7
1	9	6	5	5	4	1	3

# 簡單！線段樹

- $4n$  的空間
- 多個複雜的遞迴函數(build, query, update,...)





# 維護前綴和

- 我們對陣列  $a$  維護他的前綴和陣列  $S$

- $S_0 = 0$

- $S_1 = a_1$

- $S_2 = a_1 + a_2$

- $S_3 = a_1 + a_2 + a_3$

- $S_4 = a_1 + a_2 + a_3 + a_4$

$$S_n = S_{n-1} + a_n$$

$$\sum_{i=ql}^{qr} a_i = S_{qr} - S_{ql-1}$$

# Fenwick Tree – 動態維護前綴和

- 又稱為 Binary Index Tree (BIT), 樹狀樹組
- 給定  $x$ ，有兩種操作：
  - 在  $O(\log n)$  的時間計算  $S_x$
  - 在  $O(\log n)$  的時間修改(增減)  $a_x$  的數值
- 空間複雜度  $O(n)$ ，常數非常小！
- 非常好寫！



# BIT 操作介紹

- $modify(x, val)$ :  
將  $a_x += val$
- $query(x)$ :  
計算  $a_1 + a_2 + \dots + a_x$
- ~~$build(n)$~~ :  
沒有這個操作，BIT 假設一開始陣列  $a$  裡面都是 0

# 重要函數 *lowbit(x)*

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

- *lowbit(x)* :  
非負整數  $x$  在二進位表示時，最靠右邊的 1 所對應的值。

- 範例：

$$20_{(10)} = 10100_{(2)}$$

其中的兩個 **bit** 分別表示  $2^4$  和  $2^2$ ，因此

$$\text{lowbit}(20) = 2^2 = 4$$



# BIT 結構

- BIT 用編號  $1 \sim n$  的陣列  $bit$  紀錄
- $bit_x$  記錄區間  $[x - lowbit(x) + 1, x]$  的總和，也就是

$$\sum_{i=x-lowbit(x)+1}^x a_i$$

$bit$  陣列

1	2	3	4	5	6	7	8	9
[1,1]	[1,2]	[3,3]	[1,4]	[5,5]	[5,6]	[7,7]	[1,8]	[9,9]



# BIT 結構

- 將紀錄的區間圖像化後  
看起來就是沒有右子樹的線段樹

[1,8]								
[1,4]								
[1,2]				[5,6]				
[1,1]		[3,3]		[5,5]		[7,7]		[9,9]

*bit* 陣列

1	2	3	4	5	6	7	8	9
[1,1]	[1,2]	[3,3]	[1,4]	[5,5]	[5,6]	[7,7]	[1,8]	[9,9]

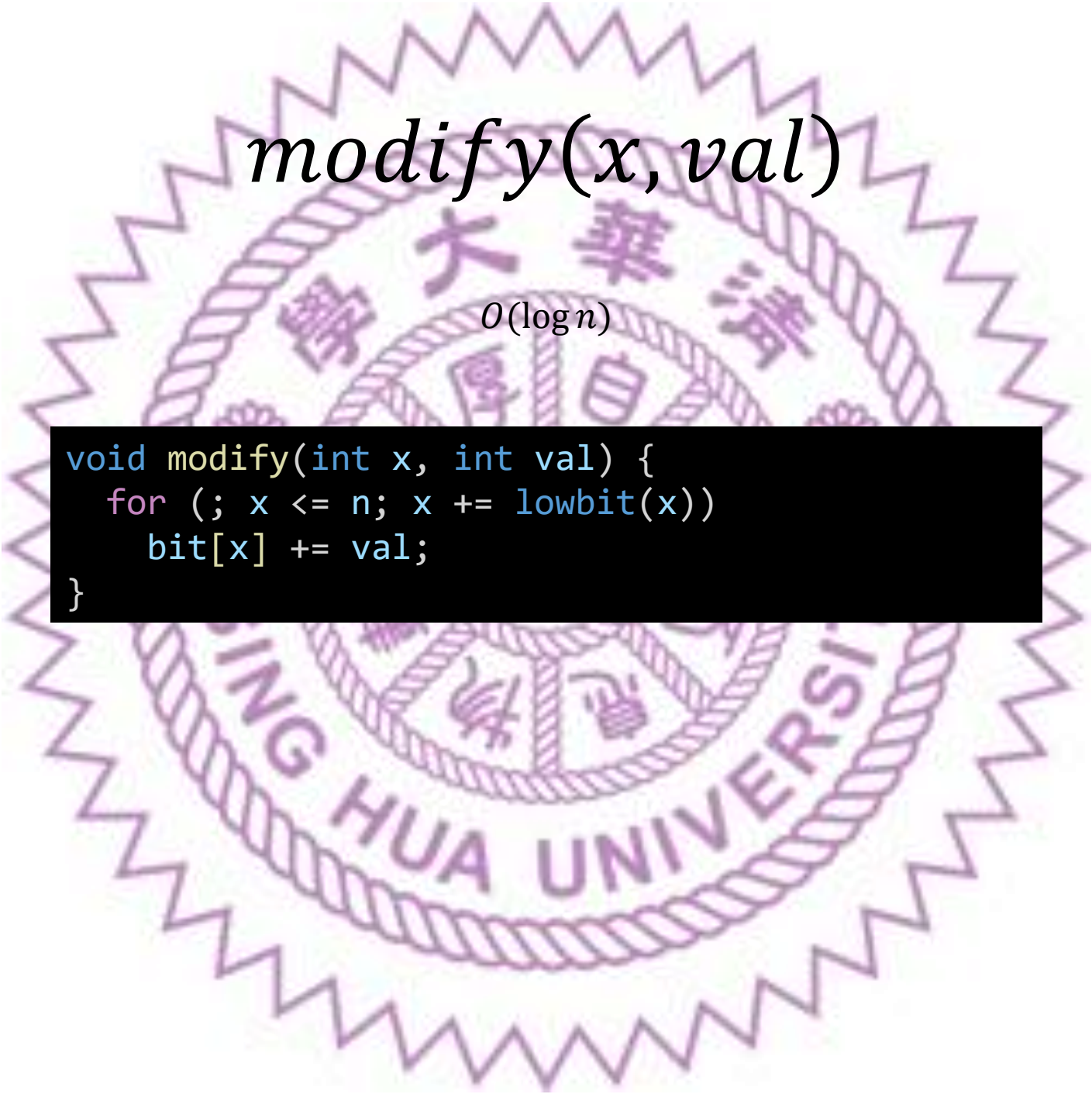
# $modify(x, val)$

- 以  $x = 5$  為例，修改  $a_5$  後需要更新的塊有： $bit_5, bit_6, bit_8$
- 特殊關係： $5 + lowbit(5) = 6$ ， $6 + lowbit(6) = 8$

[1,8]								
[1,4]								
[1,2]				[5,6]				
[1,1]		[3,3]		[5,5]		[7,7]		[9,9]

*bit* 陣列

1	2	3	4	5	6	7	8	9
[1,1]	[1,2]	[3,3]	[1,4]	[5,5]	[5,6]	[7,7]	[1,8]	[9,9]

A large, faint, purple watermark of the Tsinghua University seal is centered in the background. The seal is circular with a serrated outer edge. Inside, there are concentric circles containing the university's name in Chinese characters (清華大學) and English (TSINGHUA UNIVERSITY).

# *modify(x, val)*

$O(\log n)$

```
void modify(int x, int val) {  
    for (; x <= n; x += lowbit(x))  
        bit[x] += val;  
}
```



# $query(x)$

- 以  $x = 7$  為例， $a_1 + a_2 + \dots + a_7 = bit_4 + bit_6 + bit_7$
- 特殊關係： $7 - lowbit(7) = 6$ ， $6 - lowbit(6) = 4$

[1,8]								
[1,4]								
[1,2]				[5,6]				
[1,1]		[3,3]		[5,5]		[7,7]		[9,9]

*bit* 陣列

1	2	3	4	5	6	7	8	9
[1,1]	[1,2]	[3,3]	[1,4]	[5,5]	[5,6]	[7,7]	[1,8]	[9,9]

A large, faint, purple watermark of the Tsinghua University seal is centered in the background. The seal is circular with a serrated outer edge. Inside, there are concentric circles containing the university's name in Chinese characters (清華大學) and English (TSINGHUA UNIVERSITY).

# *query(x)*

$O(\log n)$

```
long long query(int x) {  
    long long sum = 0;  
    for (; x; x -= lowbit(x))  
        sum += bit[x];  
    return sum;  
}
```

# 包裝後 完整程式碼

```
class BIT {
    int n;
    vector<long long> bit;

public:
    void init(int _n) {
        n = _n;
        bit.resize(n);
        for (auto &b : bit) b = 0;
    }
    long long query(int x) const {
        long long sum = 0;
        for (; x; x -= lowbit(x))
            sum += bit[x];
        return sum;
    }
    void modify(int x, int val) {
        for (; x <= n; x += lowbit(x))
            bit[x] += val;
    }
};
```



## 拓展到二維

```
class BIT2D {
    int m;
    vector<BIT> bit1D;

public:
    void init(int _m, int _n) {
        m = _m;
        bit1D.resize(m);
        for (auto &b : bit1D) b.init(_n);
    }
    long long query(int x, int y) const {
        long long sum = 0;
        for (; x; x -= lowbit(x))
            sum += bit1D[x].query(y);
        return sum;
    }
    void modify(int x, int y, int val) {
        for (; x <= m; x += lowbit(x))
            bit1D[x].modify(y, val);
    }
};
```

# 經典題－逆序數對

- 給一個數列  $a_1, a_2, a_3, \dots, a_n$   
有多少對  $(i, j)$ ，滿足  $1 \leq i < j \leq n$  但  $a_i > a_j$ ？
- 已知  $n \leq 100000, 1 \leq a_i \leq n$

# 方法1：在 merge sort 過程中順便求

- 與本章節無關所以不討論
- 但是要知道有這個方法





## 方法2：值域 BIT

- 對於每個  $1 \leq x \leq n$   
計算  $a_1 \sim a_{x-1}$  中有多少數字 **大於**  $a_x$
- 這些數量總和就是逆序數對

陣列  $a$

1	2	3	4	5	6
4	3	2	2	3	1

$x = 5$

- 用一個值域陣列  $b$  紀錄  
 $a_1 \sim a_{x-1}$  中每個數字出現幾次
- $b_1 \sim b_{a_x}$  的前綴和  
代表 **小於等於**  $a_x$  的數字出現幾次

陣列  $b$

1	2	3	4
0	2	1	1

$a_1 \sim a_4$  中小於等於  $a_5$  的有 3 個  
大於  $a_5$  的有  $(5 - 1) - 3 = 1$  個

# 陣列 $b$ 可以用 BIT 輕易維護

```
vector<int> v(n);
for(int i = 0; i < n; ++i) cin >> v[i];
BIT bit; /* 如果數值範圍太大要記得做離散化 */
bit.init(*max_element(v.begin(), v.end()));
int ans = 0;
for(int i = 0; i < n; ++i) {
    ans += i - bit.query(v[i]);
    bit.modify(v[i], 1);
}
cout << ans << '\n';
```

## 經典題 2

- 給你一個長度為  $n$  的陣列  $a$ ，再給你  $q$  個操作，操作有兩種：

- $sum(ql, qr)$ :  
查詢  $a_{ql} + a_{ql+1} + \dots + a_{qr}$  的值

- $add(ql, qr, val)$ :  
將  $a_{ql}, a_{ql+1}, \dots, a_{qr}$  都加上  $val$

- $1 \leq n, q \leq 10^6$

線段樹+懶惰標記?

$a$

1	2	3	4	5	6	7	8
1	9	6	5	5	4	1	3



# 維護差分

- 我們對陣列  $a$  維護他的差分陣列  $D$

- $D_1 = a_1$

- $D_2 = a_2 - a_1$

- $D_3 = a_3 - a_2$

- $D_4 = a_4 - a_3$

$$D_n = a_n - a_{n-1}$$

$$\sum_{i=1}^x D_i = a_x$$

# 性質

$$\begin{aligned}\sum_{i=1}^x a_i &= \sum_{i=1}^x \sum_{y=1}^i D_y \\ &= x \times D_1 + (x-1) \times D_2 + \cdots + 1 \times D_x \\ &= \sum_{i=1}^x (x-i+1) \times D_i \\ &= (x+1) \sum_{i=1}^x D_i - \sum_{i=1}^x i \times D_i\end{aligned}$$



# 性質

$$\sum_{i=1}^x a_i = (x+1) \sum_{i=1}^x D_i - \sum_{i=1}^x i \times D_i$$

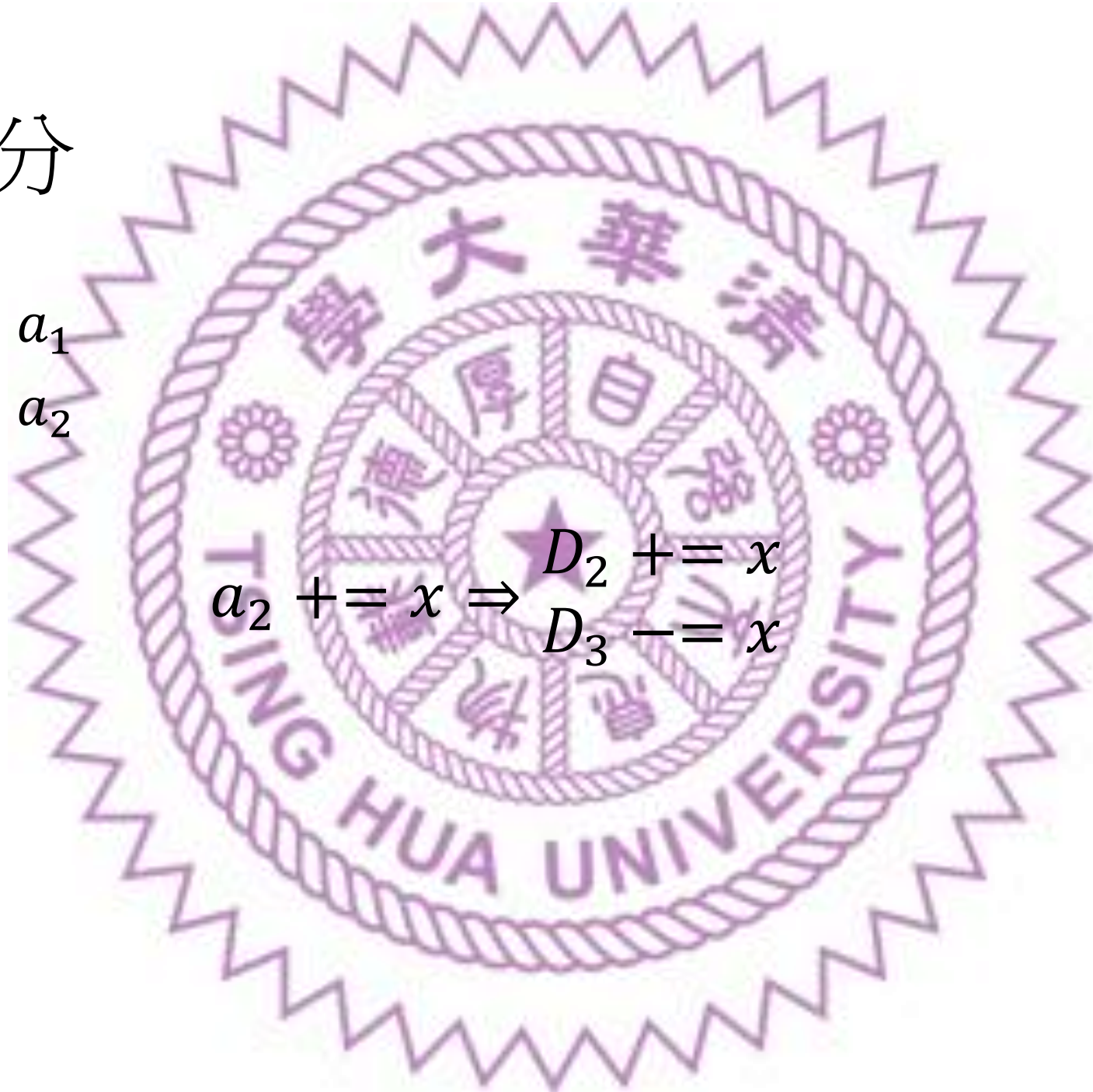
用兩顆 BIT 就可以維護的東西



# 維護差分

- $D_2 = a_2 - a_1$
- $D_3 = a_3 - a_2$

$$a_2 \pm x \Rightarrow \begin{matrix} D_2 \pm x \\ D_3 \mp x \end{matrix}$$



# 區間加值

- $a_{ql} += x$
- $a_{ql+1} += x$
- ...
- $a_{qr} += x$



- $D_{ql} += x$
- $D_{ql+1} += x - x$
- ...
- $D_{qr} += x - x$
- $D_{qr+1} += -x$



$D_{ql}$	$+= x$
$D_{qr+1}$	$-= x$

# 區間修改 +查詢

```
class RangeAddBIT {
    int n;
    BIT D, xD;

public:
    void init(int _n) {
        n = _n;
        D.init(n);
        xD.init(n);
    }
    void add(int ql, int qr, int val) { // [ql, qr] 加值
        D.modify(ql, val);
        xD.modify(ql, ql * val);
        if (qr < n) {
            D.modify(qr + 1, -val);
            xD.modify(qr + 1, -(qr + 1) * val);
        }
    }
    long long query(int x) { // 查詢 [1,x] 總和
        return (x + 1) * D.query(x) - xD.query(x);
    }
};
```