樹狀數組
**Fenwick Tree**

# Binary Indexed Tree

Pei-yih Ting @NTOUCSE

# 樹狀數組 (Binary Indexed Tree)

- 這是一個資料結構書裡沒講的結構, 可是不是 trivial 的

# 樹狀數組 (Binary Indexed Tree)

- 這是一個資料結構書裡沒講的結構, 可是不是 trivial 的
- 用來解很多程式問題: UVa11525,12086,1428,11423,501,10909,12532
  1406, 11990, 12345, 11610,
  12356, 11495, APCS10510#3,
  APCS10910#4, ZJ a693, d794,
  c576, d788, d539, d847, d799,
  b687

# 樹狀數組 (Binary Indexed Tree)

- 這是一個資料結構書裡沒講的結構, 可是不是 trivial 的
- 用來解很多程式問題: UVa11525,12086,1428,11423,501,10909,12532 1406, 11990, 12345, 11610, 12356, 11495, APCS10510#3, APCS10910#4, ZJ a693, d794, c576, d788, d539, d847, d799, b687
- 重點是功能很明確, 實作很簡單

# 樹狀數組 (Binary Indexed Tree)

- 這是一個資料結構書裡沒講的結構, 可是不是 trivial 的
- 用來解很多程式問題: UVa11525,12086,1428,11423,501,10909,12532 1406, 11990, 12345, 11610,
- 重點是功能很明確, 實作很簡單

有一個 n 個元素的數列 $a_1, a_2, \ldots, a_n$, 數列裡某些元素會不斷地更動, 你需要在每次更動後以 O(log n) 的時間裡掌握部份連續元素的和 $a_s+a_{s+1}+\ldots+a_e$, s 和 e 可任意指定

# 樹狀數組 (Binary Indexed Tree)

- 這是一個資料結構書裡沒講的結構, 可是不是 trivial 的
- 用來解很多程式問題: UVa11525,12086,1428,11423,501,10909,12532
  1406, 11990, 12345, 11610,
- 重點是功能很明確, 實作很簡單

有一個 n 個元素的數列 $a_1, a_2, \ldots, a_n$, 數列裡某些元素會不斷地更動, 你需要在每次更動後以 O(log n) 的時間裡掌握部份連續元素的和 $a_s+a_{s+1}+\ldots+a_e$, s 和 e 可任意指定

- trivial 的作法在每次更動後都需要 O(n) 的時間來計算部份和

# 樹狀數組 (Binary Indexed Tree)

- 這是一個資料結構書裡沒講的結構, 可是不是 trivial 的
- 用來解很多程式問題: UVa11525,12086,1428,11423,501,10909,12532
  1406, 11990, 12345, 11610,
- 重點是功能很明確, 實作很簡單

有一個 n 個元素的數列 $a_1, a_2, \ldots, a_n$, 數列裡某些元素會不斷地更動, 你需要在每次更動後以 O(log n) 的時間裡掌握部份連續元素的和 $a_s+a_{s+1}+\ldots+a_e$, s 和 e 可任意指定

- trivial 的作法在每次更動後都需要 O(n) 的時間來計算部份和
- 樹狀數組結構裡不存放數列的元素 $a_1, a_2, \ldots, a_n$ 了

# 樹狀數組 (Binary Indexed Tree)

- 這是一個資料結構書裡沒講的結構, 可是不是 trivial 的
- 用來解很多程式問題: UVa11525,12086,1428,11423,501,10909,12532
  1406, 11990, 12345, 11610,
- 重點是功能很明確, 實作很簡單

> 有一個 n 個元素的數列 $a_1, a_2, \ldots, a_n$, 數列裡某些元素會不斷地更動, 你需要在每次更動後以 O(log n) 的時間裡掌握部份連續元素的和 $a_s+a_{s+1}+\ldots+a_e$, s 和 e 可任意指定

- trivial 的作法在每次更動後都需要 O(n) 的時間來計算部份和
- 樹狀數組結構裡不存放數列的元素 $a_1, a_2, \ldots, a_n$ 了
- 只存放部份元素的和 $s_1, s_2, \ldots, s_n$ (還是有 n 個)

# 樹狀數組 (Binary Indexed Tree)

- 這是一個資料結構書裡沒講的結構, 可是不是 trivial 的
- 用來解很多程式問題: UVa11525,12086,1428,11423,501,10909,12532 1406, 11990, 12345, 11610,
- 重點是功能很明確, 實作很簡單

> 有一個 n 個元素的數列 $a_1, a_2, \ldots, a_n$, 數列裡某些元素會不斷地更動, 你需要在每次更動後以 O(log n) 的時間裡掌握部份連續元素的和 $a_s+a_{s+1}+\ldots+a_e$, s 和 e 可任意指定

- trivial 的作法在每次更動後都需要 O(n) 的時間來計算部份和
- 樹狀數組結構裡不存放數列的元素 $a_1, a_2, \ldots, a_n$ 了
- 只存放部份元素的和 $s_1, s_2, \ldots, s_n$ (還是有 n 個)

  其中 $s_1=a_1$, $s_2=a_1+a_2$, $s_3=a_3$, $s_4=a_1+a_2+a_3+a_4$, …, $s_n=\ldots$

# 樹狀數組 (Binary Indexed Tree)

- 這是一個資料結構書裡沒講的結構, 可是不是 trivial 的
- 用來解很多程式問題: UVa11525,12086,1428,11423,501,10909,12532
  1406, 11990, 12345, 11610,
- 重點是功能很明確, 實作很簡單

> 有一個 n 個元素的數列 $a_1, a_2, \ldots, a_n$, 數列裡某些元素會不斷地更動, 你需要在每次更動後以 O(log n) 的時間裡掌握部份連續元素的和 $a_s+a_{s+1}+\ldots+a_e$, s 和 e 可任意指定

- trivial 的作法在每次更動後都需要 O(n) 的時間來計算部份和
- 樹狀數組結構裡不存放數列的元素 $a_1, a_2, \ldots, a_n$ 了
- 只存放部份元素的和 $s_1, s_2, \ldots, s_n$ (還是有 n 個)

  其中 $s_1=a_1$, $s_2=a_1+a_2$, $s_3=a_3$, $s_4=a_1+a_2+a_3+a_4$, …, $s_n=\ldots$

  什麼?

# 樹狀數組 (Binary Indexed Tree)

- 這是一個資料結構書裡沒講的結構, 可是不是 trivial 的
- 用來解很多程式問題: UVa11525,12086,1428,11423,501,10909,12532
  1406, 11990, 12345, 11610,
- 重點是功能很明確, 實作很簡單

有一個 n 個元素的數列 $a_1, a_2, \ldots, a_n$, 數列裡某些元素會不斷地更動, 你需要在每次更動後以 **O(log n)** 的時間裡掌握部份連續元素的和 $a_s+a_{s+1}+\ldots+a_e$, **s** 和 **e** 可任意指定

- trivial 的作法在每次更動後都需要 **O(n)** 的時間來計算部份和
- 樹狀數組結構裡不存放數列的元素 $a_1, a_2, \ldots, a_n$ 了
- 只存放部份元素的和 $s_1, s_2, \ldots, s_n$ (還是有 n 個)

  其中 $s_1=a_1$, $s_2=a_1+a_2$, $s_3=a_3$, $s_4=a_1+a_2+a_3+a_4$, …, $s_n=\ldots$

  什麼? 這是什麼規則?

# 樹狀數組 (Binary Indexed Tree)

- 這是一個資料結構書裡沒講的結構, 可是不是 trivial 的
- 用來解很多程式問題: UVa11525,12086,1428,11423,501,10909,12532 1406, 11990, 12345, 11610,
- 重點是功能很明確, 實作很簡單

> 有一個 n 個元素的數列 $a_1, a_2, \ldots, a_n$, 數列裡某些元素會不斷地更動, 你需要在每次更動後以 O(log n) 的時間裡掌握部份連續元素的和 $a_s+a_{s+1}+\ldots+a_e$, s 和 e 可任意指定

- trivial 的作法在每次更動後都需要 O(n) 的時間來計算部份和
- 樹狀數組結構裡不存放數列的元素 $a_1, a_2, \ldots, a_n$ 了
- 只存放部份元素的和 $s_1, s_2, \ldots, s_n$ (還是有 n 個)

  其中 $s_1=a_1$, $s_2=a_1+a_2$, $s_3=a_3$, $s_4=a_1+a_2+a_3+a_4$, …, $s_n=\ldots$

  什麼? 這是什麼規則? 會不會太複雜了啊 !!!

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：

$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：

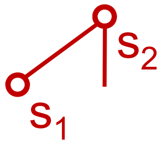
$s_1$

$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：



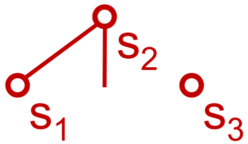

$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

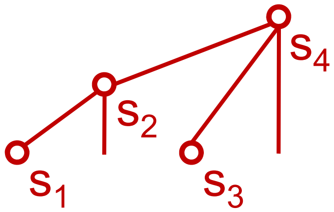
$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

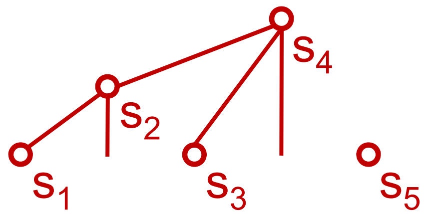
$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

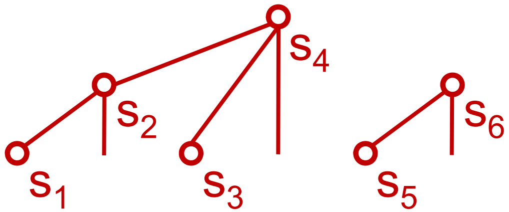
$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

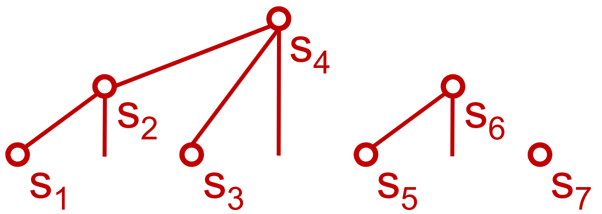
$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

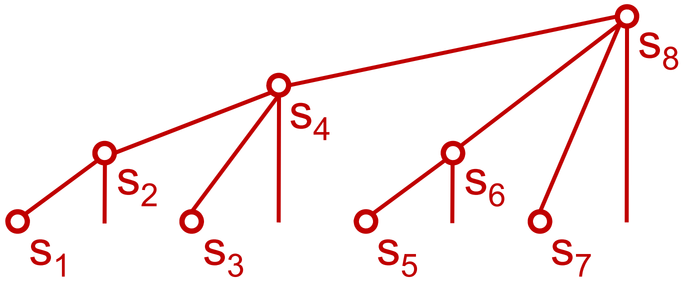
$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

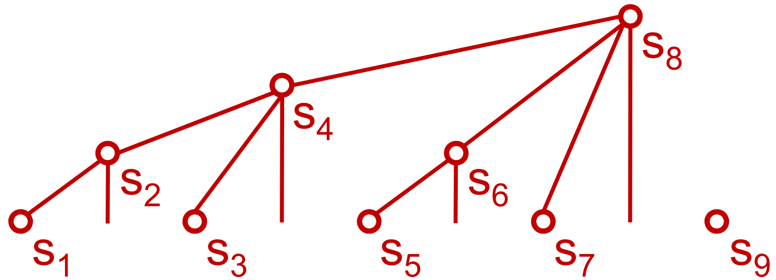
$s_7=a_7$

$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

- $\mathbf{s_i}$ 是那個子樹裡所有 $\mathbf{a_i}$ 的和

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

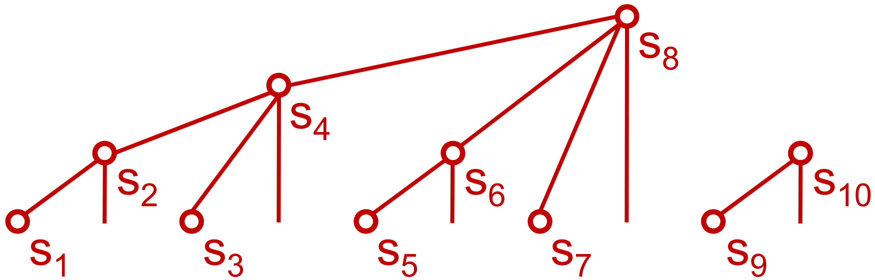
$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

$s_9=a_9$

- $\mathbf{s_i}$ 是那個子樹裡所有 $\mathbf{a_i}$ 的和

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

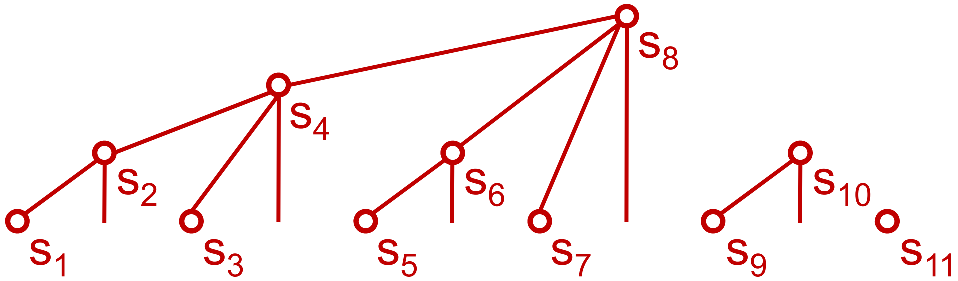
$s_9=a_9$

$s_{10}=a_9+a_{10}$

- **$s_i$** 是那個子樹裡所有 **$a_i$** 的和

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

$s_9=a_9$

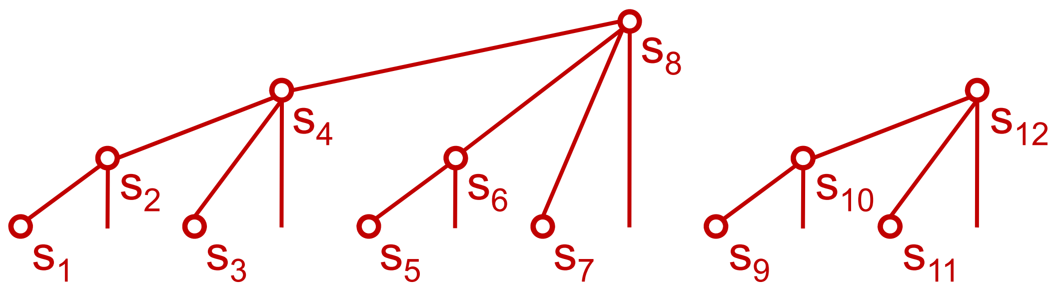
$s_{10}=a_9+a_{10}$

$s_{11}=a_{11}$

- $\mathbf{s_i}$ 是那個子樹裡所有 $\mathbf{a_i}$ 的和

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

$s_9=a_9$

$s_{10}=a_9+a_{10}$

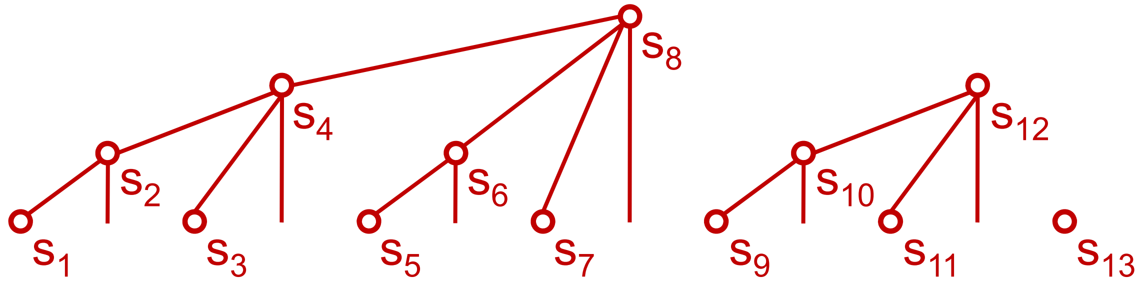
$s_{11}=a_{11}$

$s_{12}=a_9+a_{10}+a_{11}+a_{12}$

- **$s_i$** 是那個子樹裡所有 **$a_i$** 的和

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

$s_9=a_9$

$s_{10}=a_9+a_{10}$

$s_{11}=a_{11}$

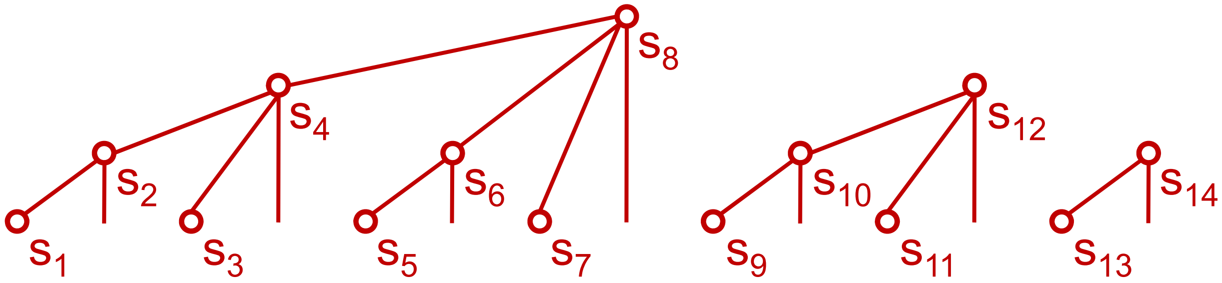
$s_{12}=a_9+a_{10}+a_{11}+a_{12}$

$s_{13}=a_{13}$

- $\mathbf{s_i}$ 是那個子樹裡所有 $\mathbf{a_i}$ 的和

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

$s_9=a_9$

$s_{10}=a_9+a_{10}$

$s_{11}=a_{11}$

$s_{12}=a_9+a_{10}+a_{11}+a_{12}$

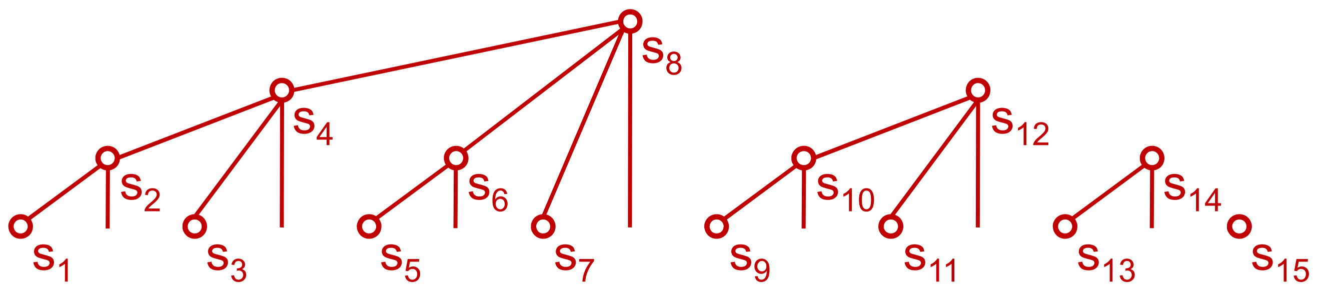
$s_{13}=a_{13}$

$s_{14}=a_{13}+a_{14}$

- $s_i$ 是那個子樹裡所有 $a_i$ 的和

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

$s_9=a_9$

$s_{10}=a_9+a_{10}$

$s_{11}=a_{11}$

$s_{12}=a_9+a_{10}+a_{11}+a_{12}$

$s_{13}=a_{13}$

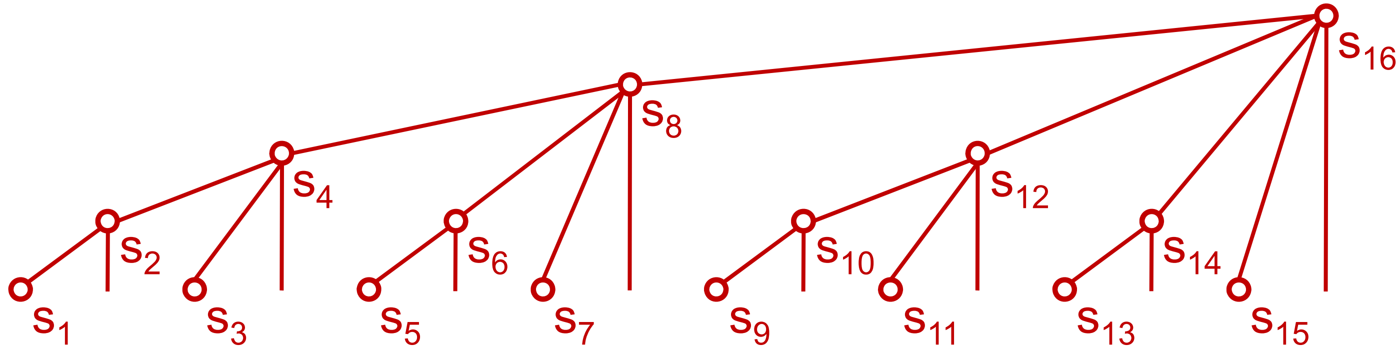
$s_{14}=a_{13}+a_{14}$

$s_{15}=a_{15}$

- $\mathbf{s_i}$ 是那個子樹裡所有 $\mathbf{a_i}$ 的和

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

$s_9=a_9$

$s_{10}=a_9+a_{10}$

$s_{11}=a_{11}$

$s_{12}=a_9+a_{10}+a_{11}+a_{12}$

$s_{13}=a_{13}$

$s_{14}=a_{13}+a_{14}$

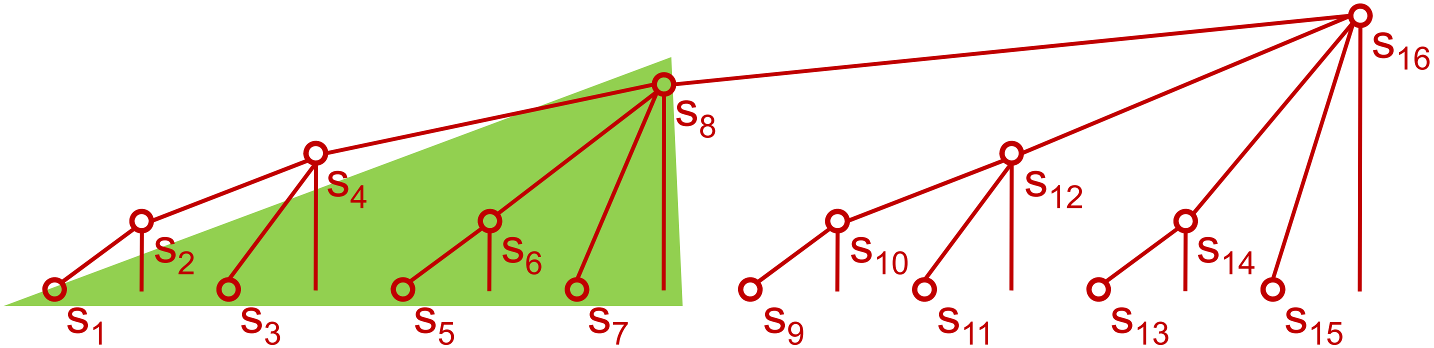
$s_{15}=a_{15}$

$s_{16}=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8+a_9+a_{10}+a_{11}+a_{12}+a_{13}+a_{14}+a_{15}+a_{16}$

- $\mathbf{s_i}$ 是那個子樹裡所有 $\mathbf{a_i}$ 的和

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

$s_9=a_9$

$s_{10}=a_9+a_{10}$

$s_{11}=a_{11}$

$s_{12}=a_9+a_{10}+a_{11}+a_{12}$

$s_{13}=a_{13}$

$s_{14}=a_{13}+a_{14}$

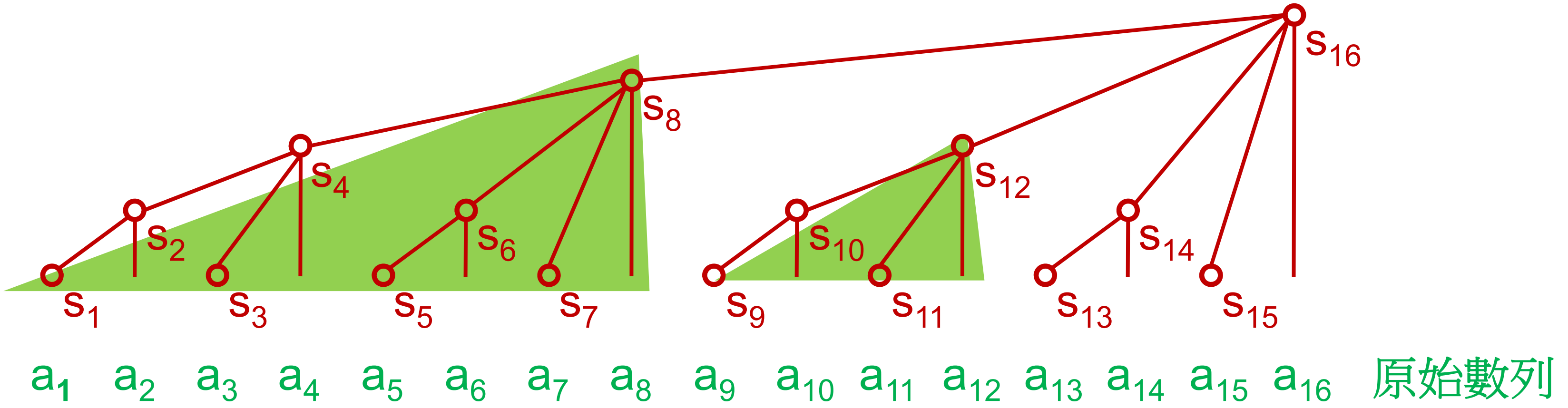
$s_{15}=a_{15}$

$s_{16}=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8+a_9+a_{10}+a_{11}+a_{12}+a_{13}+a_{14}+a_{15}+a_{16}$

- 所有子樹依照一種特別的遞迴架構定義出來
- $\mathbf{s_i}$ 是那個子樹裡所有 $\mathbf{a_i}$ 的和

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

$s_9=a_9$

$s_{10}=a_9+a_{10}$

$s_{11}=a_{11}$

$s_{12}=a_9+a_{10}+a_{11}+a_{12}$

$s_{13}=a_{13}$

$s_{14}=a_{13}+a_{14}$

$s_{15}=a_{15}$

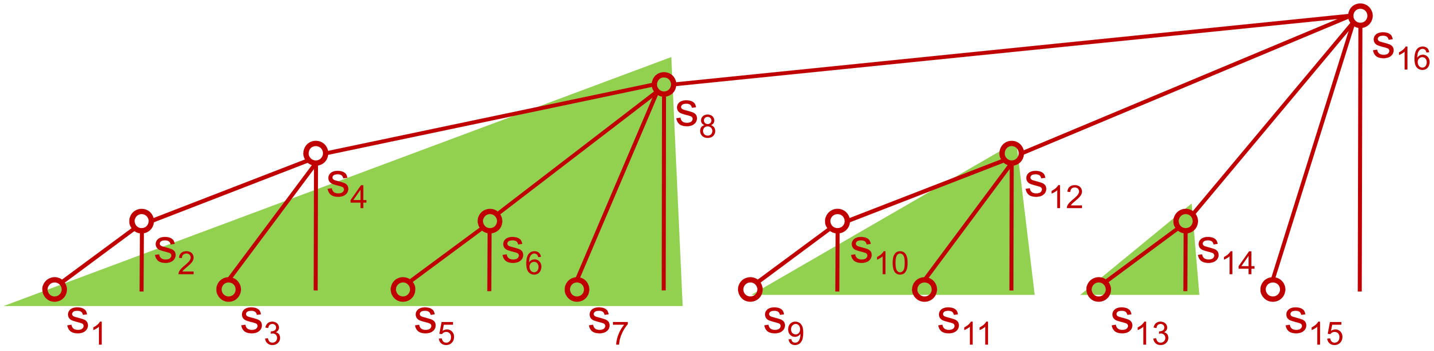
$s_{16}=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8+a_9+a_{10}+a_{11}+a_{12}+a_{13}+a_{14}+a_{15}+a_{16}$

- 所有子樹依照一種特別的遞迴架構定義出來

- **$s_i$** 是那個子樹裡所有 **$a_i$** 的和

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

$s_9=a_9$

$s_{10}=a_9+a_{10}$

$s_{11}=a_{11}$

$s_{12}=a_9+a_{10}+a_{11}+a_{12}$

$s_{13}=a_{13}$

$s_{14}=a_{13}+a_{14}$

$s_{15}=a_{15}$

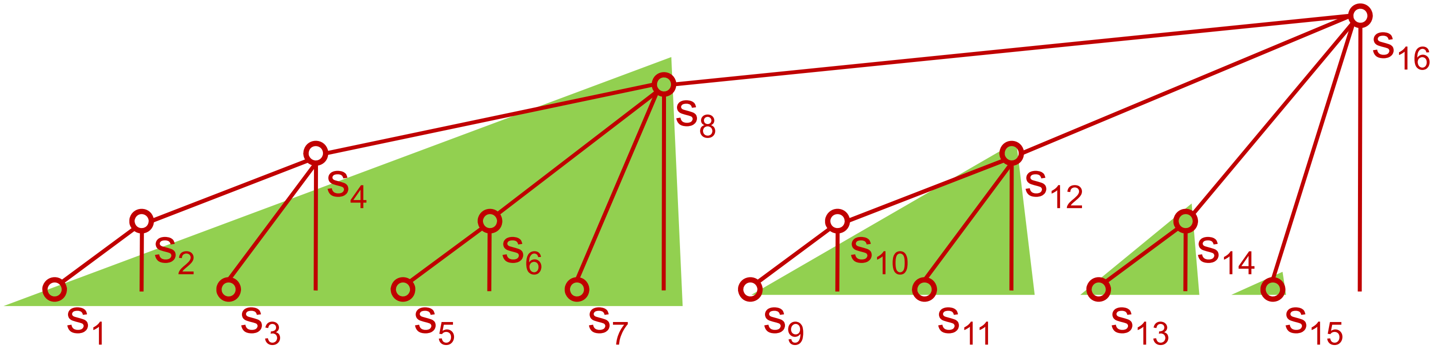
$s_{16}=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8+a_9+a_{10}+a_{11}+a_{12}+a_{13}+a_{14}+a_{15}+a_{16}$

- 所有子樹依照一種特別的遞迴架構定義出來

- $\mathbf{s_i}$ 是那個子樹裡所有 $\mathbf{a_i}$ 的和

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

$s_9=a_9$

$s_{10}=a_9+a_{10}$

$s_{11}=a_{11}$

$s_{12}=a_9+a_{10}+a_{11}+a_{12}$

$s_{13}=a_{13}$

$s_{14}=a_{13}+a_{14}$

$s_{15}=a_{15}$

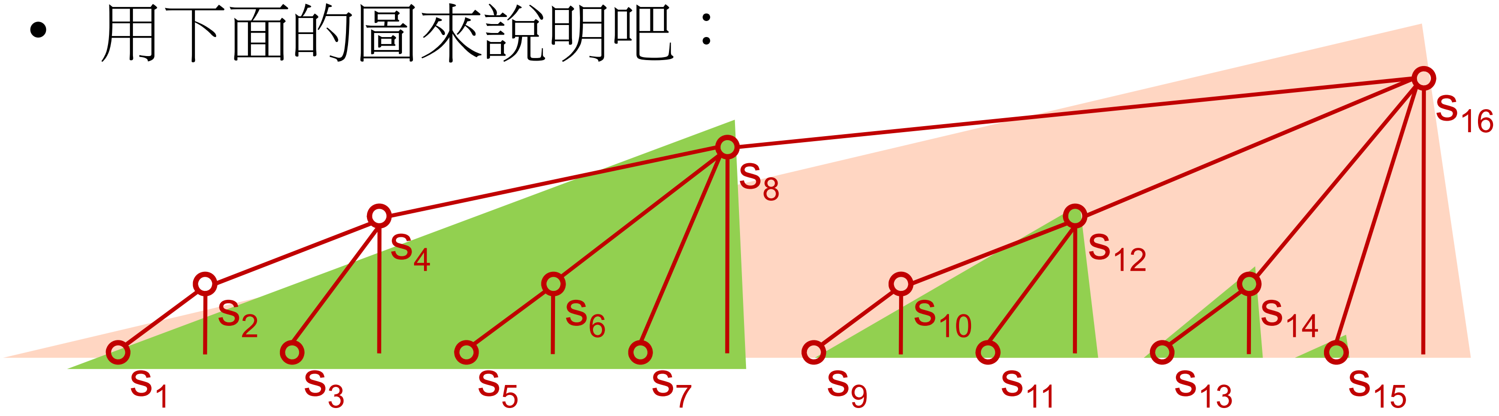
$s_{16}=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8+a_9+a_{10}+a_{11}+a_{12}+a_{13}+a_{14}+a_{15}+a_{16}$

- 所有子樹依照一種特別的遞迴架構定義出來

- $s_i$ 是那個子樹裡所有 $a_i$ 的和

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：




第二次出現某一高度時拔高一層

$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

$s_9=a_9$

$s_{10}=a_9+a_{10}$

$s_{11}=a_{11}$

$s_{12}=a_9+a_{10}+a_{11}+a_{12}$

$s_{13}=a_{13}$

$s_{14}=a_{13}+a_{14}$

$s_{15}=a_{15}$

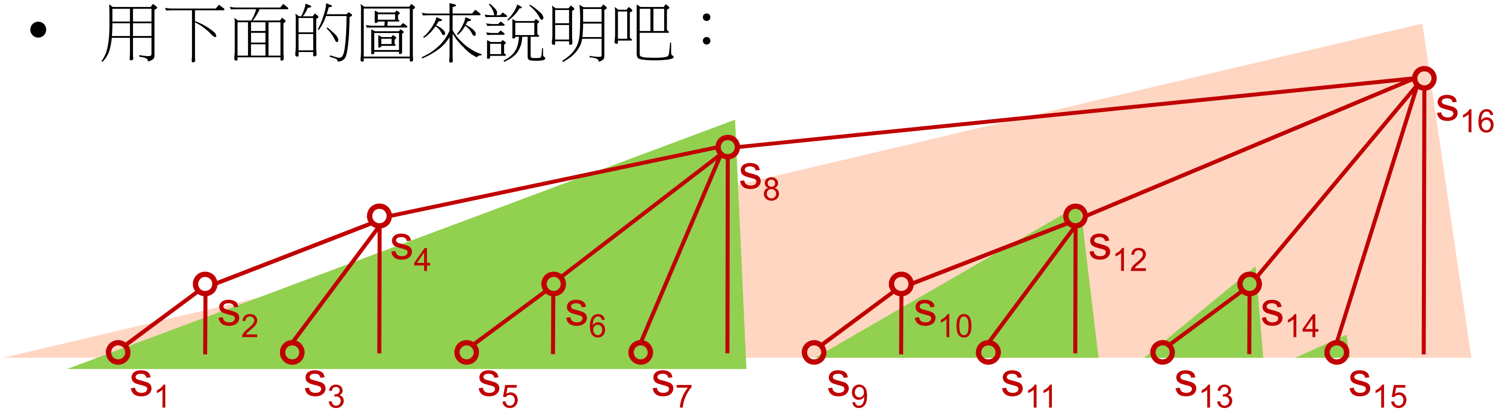
$s_{16}=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8+a_9+a_{10}+a_{11}+a_{12}+a_{13}+a_{14}+a_{15}+a_{16}$

- 所有子樹依照一種特別的遞迴架構定義出來
- **$s_i$** 是那個子樹裡所有 **$a_i$** 的和

# 樹狀數組 for RSQ (續)

$s_{16}=s_8+s_{12}+s_{14}+s_{15}+a_{16}$

- 用下面的圖來說明吧：




第二次出現某一高度時拔高一層

$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

$s_9=a_9$

$s_{10}=a_9+a_{10}$

$s_{11}=a_{11}$

$s_{12}=a_9+a_{10}+a_{11}+a_{12}$

$s_{13}=a_{13}$

$s_{14}=a_{13}+a_{14}$

$s_{15}=a_{15}$

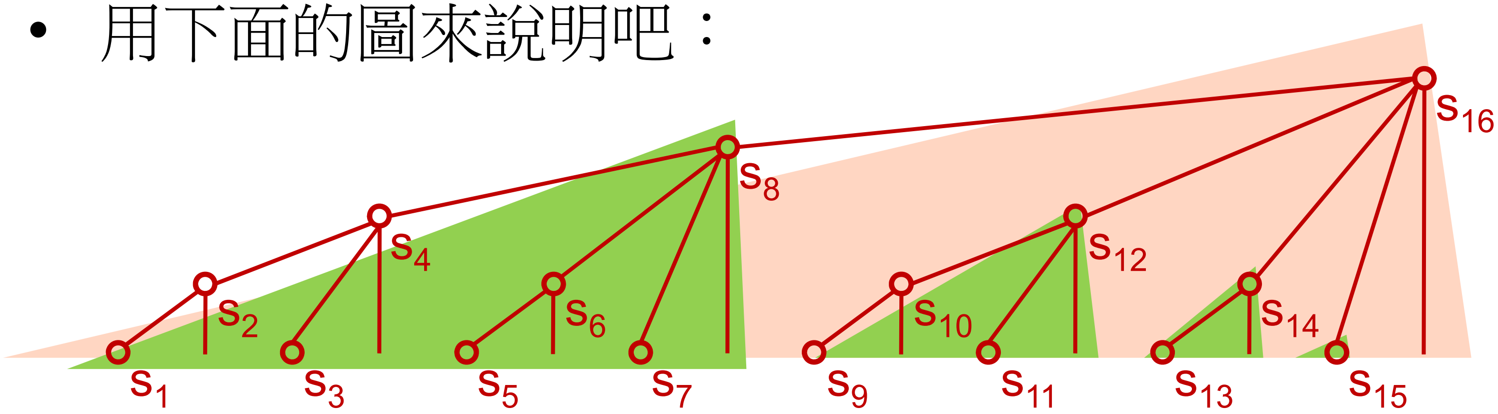
$s_{16}=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8+a_9+a_{10}+a_{11}+a_{12}+a_{13}+a_{14}+a_{15}+a_{16}$

- 所有子樹依照一種特別的遞迴架構定義出來
- $s_i$ 是那個子樹裡所有 $a_i$ 的和
- 可以由 O(log k) 個 $\{s_i\}$ 快速地計算出任意 $s_8=a_1+a_2+ \ldots +a_k$

# 樹狀數組 for RSQ (續)

$s_{16}=s_8+s_{12}+s_{14}+s_{15}+a_{16}$

- 用下面的圖來說明吧：




第二次出現某一高度時拔高一層

$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

$s_9=a_9$

$s_{10}=a_9+a_{10}$

$s_{11}=a_{11}$

$s_{12}=a_9+a_{10}+a_{11}+a_{12}$

$s_{13}=a_{13}$

$s_{14}=a_{13}+a_{14}$

$s_{15}=a_{15}$

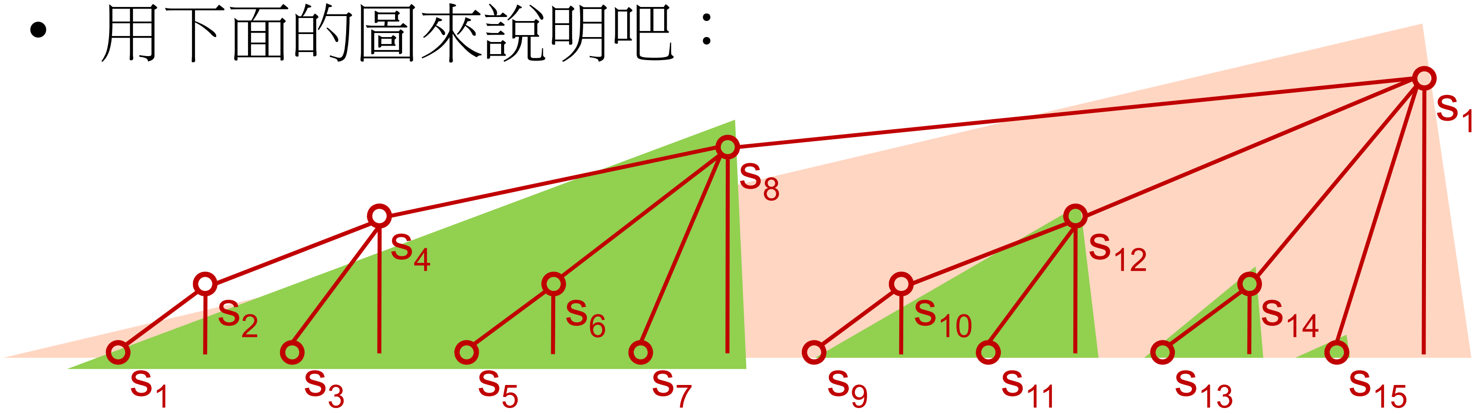
$s_{16}=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8+a_9+a_{10}+a_{11}+a_{12}+a_{13}+a_{14}+a_{15}+a_{16}$

- 所有子樹依照一種特別的遞迴架構定義出來
- **$s_i$** 是那個子樹裡所有 **$a_i$** 的和
- 可以由 O(log k) 個 $\{s_i\}$ 快速地計算出任意 $s_8=a_1+a_2+\ldots+a_k$
  例如 $a_1+\ldots+a_{10}=s_8+s_{10}$

# 樹狀數組 for RSQ (續)

$s_{16}=s_8+s_{12}+s_{14}+s_{15}+a_{16}$

- 用下面的圖來說明吧：




第二次出現某一高度時拔高一層

$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+$ …

$s_3=a_3$

$s_4=s_2+s_3+a_4=a$ …

如果 $s_k$ 在第 $\ell$ 層, 由 $s_{k-1}$ 開始往前加大於等於第 **1** 層的 $s_{k-1-lb(k-1)}$, 再往前加大於等於第 **2** 層的 $s_{k-1-lb(k-1)-lb(k-1-lb(k-1))}$, 一直到第 $\ell$-1 層, 然後到小於 $s_1$ 為止, 不會超過 log k 層, 也就是最多 **log k** 項 $\{s_i\}$

…樹依照一種特別…架構定義出來

- $s_i$ 是那個子樹裡所有 $a_i$ 的和

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

$s_{13}=a_{13}$

$s_{14}=a_{13}+a_{14}$

$s_{15}=a_{15}$

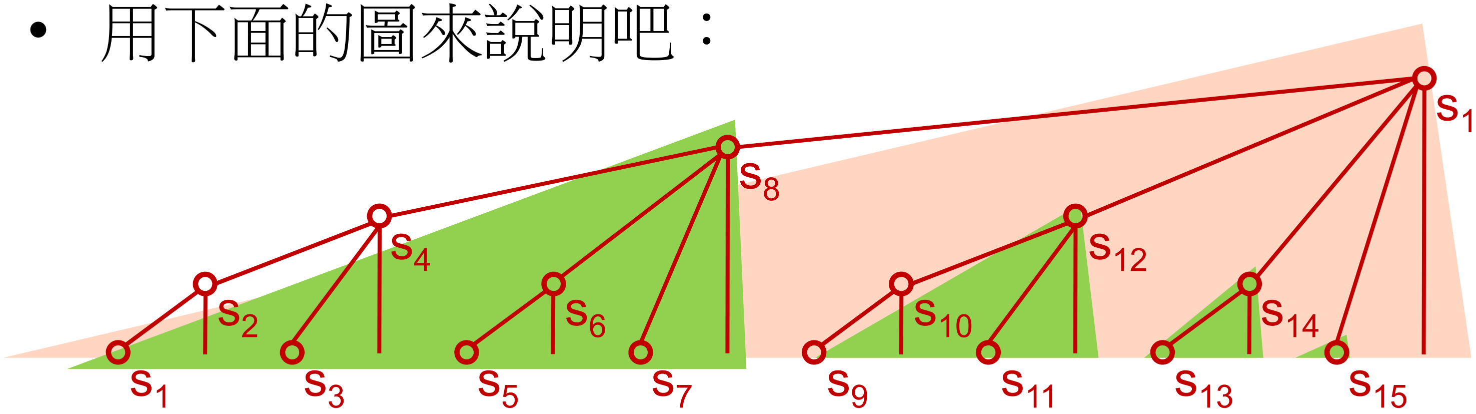
$s_{16}=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8+a_9+a_{10}+a_{11}+a_{12}+a_{13}+a_{14}+a_{15}+a_{16}$

- 可以由 O(log k) 個 $\{s_i\}$ 快速地計算出任意 $s_8=a_1+a_2+\ldots+a_k$
  例如 $a_1+\ldots+a_{10}=s_8+s_{10}$

# 樹狀數組 for RSQ (續)

- 用下面的圖來說明吧：

$s_{16}=s_8+s_{12}+s_{14}+s_{15}+a_{16}$




第二次出現某一高度時拔高一層

$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_{13}$ $a_{14}$ $a_{15}$ $a_{16}$ 原始數列

$s_1=a_1$

$s_2=s_1+a_2=a_1+a_2$

$s_3=a_3$

$s_4=s_2+s_3+a_4=a_1+a_2+a_3+a_4$

$s_5=a_5$

$s_6=s_5+a_6=a_5+a_6$

$s_7=a_7$

$s_8=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8$

$s_9=a_9$

$s_{10}=a_9+a_{10}$

$s_{11}=a_{11}$

$s_{12}=a_9+a_{10}+a_{11}+a_{12}$

$s_{13}=a_{13}$

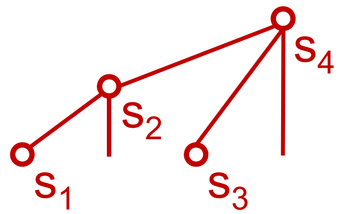
$s_{14}=a_{13}+a_{14}$

$s_{15}=a_{15}$

$s_{16}=a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8+a_9+a_{10}+a_{11}+a_{12}+a_{13}+a_{14}+a_{15}+a_{16}$

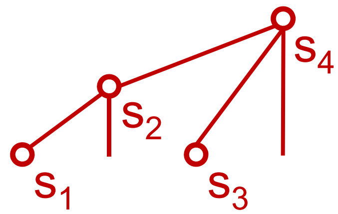
- 所有子樹依照一種特別的遞迴架構定義出來
- **$s_i$** 是那個子樹裡所有 **$a_i$** 的和
- 可以由 O(log k) 個 $\{s_i\}$ 快速地計算出任意 $s_8=a_1+a_2+\ldots+a_k$ 例如 $a_1+\ldots+a_{10}=s_8+s_{10}$
- 任一 $a_i$ 修改時只需修改由 $a_i$ 計算出來的 O(log n) 個 $\{s_i\}$

# BIT RSQ 實作



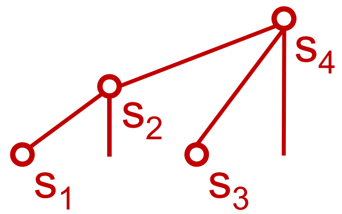
# BIT RSQ 實作

❶ $s_i$ 子樹的大小  #define **lb(i)** ((i)&(-(i)))



# BIT RSQ 實作

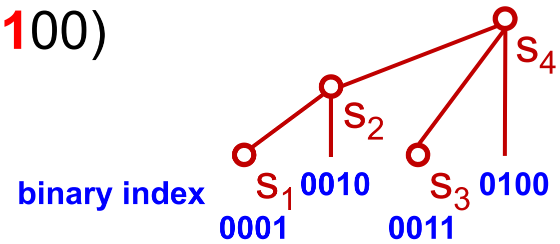
例: i=12(01**1**00), -i=-12(10100)
lb(i)=4(00**1**00)




❶ $s_i$ 子樹的大小 #define **lb(i)** ((**i**)&(-(**i**)))

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)

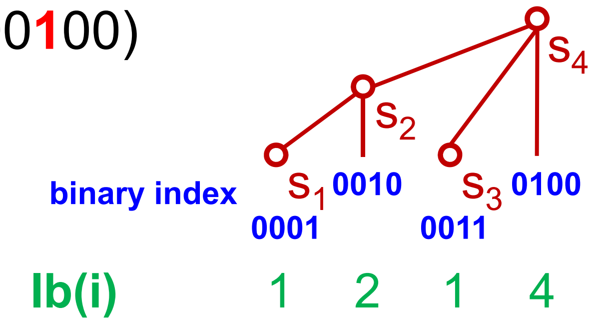



❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

# BIT RSQ 實作

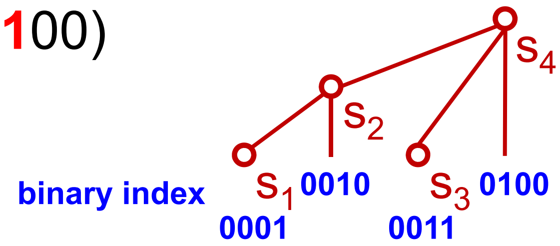
例: i=12(01**1**00), -i=-12(10100)
lb(i)=4(00**1**00)

❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))



# BIT RSQ 實作

例: i=12(01**1**00), -i=-12(10100)
lb(i)=4(00**1**00)



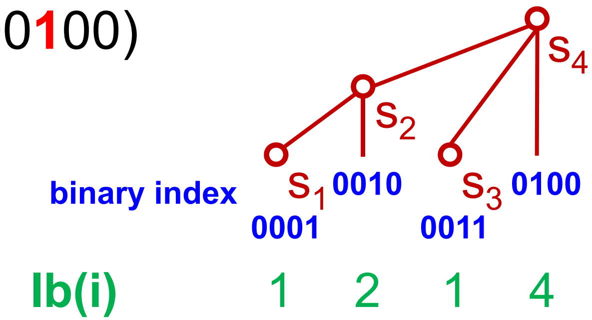

lb(i) 1 2 1 4

❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i\text{-}lb(i)}$

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)



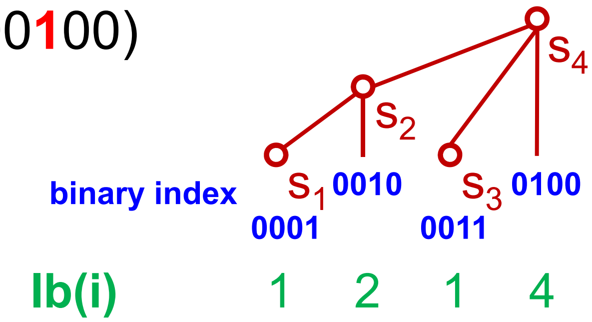

❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)




❶ $s_i$ 子樹的大小 `#define lb(i) ((i)&(-(i)))`

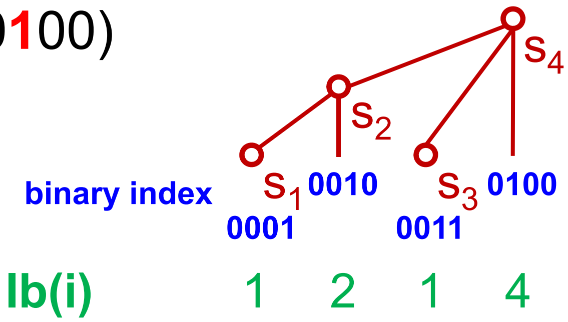
❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)




❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

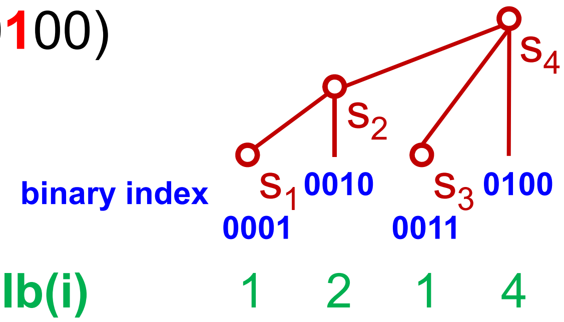
❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1}$$

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)




❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

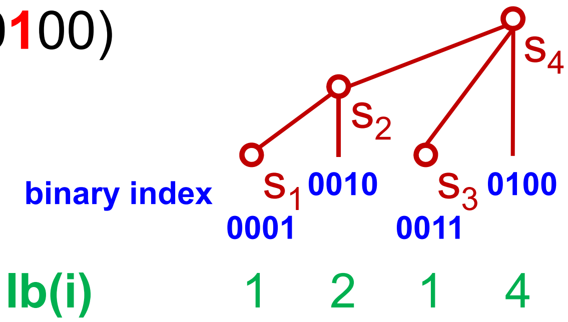
❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i + \ldots + a_{i-lb(i)+1}$$ (奇數 i, $s_i = a_i$)

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)




❶ $s_i$ 子樹的大小　#define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

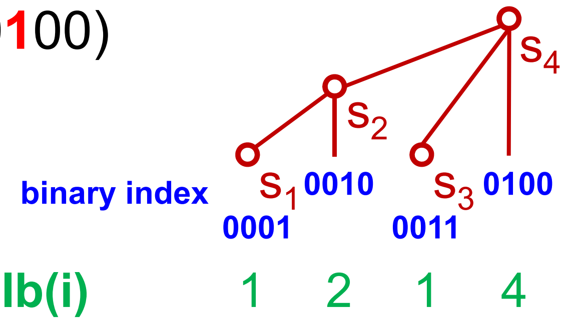
❹ 由 $\{a_i\}$ 計算 $s_i$　　子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i + \ldots + a_{i-lb(i)+1}$$　(奇數 i, $s_i = a_i$)

$a_i, \{s_{i-k}\}$

# BIT RSQ 實作

例: i=12(01**1**00), -i=-12(10100)
lb(i)=4(00**1**00)




❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

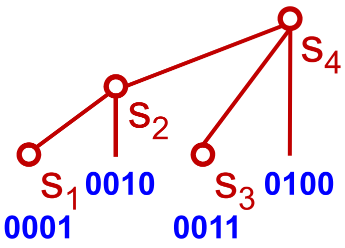
❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1}$$ (奇數 i, $s_i = a_i$)

$a_i$, $\{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$

# BIT RSQ 實作

例: i=12(01**1**00), -i=-12(10100)
lb(i)=4(00**1**00)




binary index

lb(i)　1　2　1　4

$s_b$　$s_j$　$s_i$

❶ $s_i$ 子樹的大小　#define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i\text{-}lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

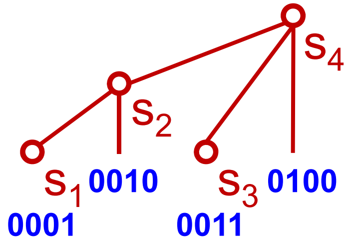
❹ 由 $\{a_i\}$ 計算 $s_i$　　子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1}$$ (奇數 i, $s_i = a_i$)

$a_i, \{s_{i-k}\}$　①　$= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$　$b=i-lb(i) == j-lb(j)$

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)




binary index

lb(i) 1 2 1 4

$s_b$ $s_j$ $s_i$

❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1} \qquad (\text{奇數 } i,\ s_i = a_i)$$

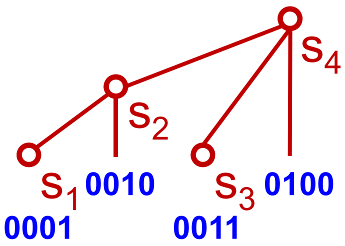
$a_i, \{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$ b=i-lb(i) == j-lb(j)

```
for (i=1; i<=n; i++)
    for (s[i]=a[i],b=i-lb(i),j=i-1; j>b; j-=lb(j))
        s[i] += s[j];
```

# BIT RSQ 實作

例: i=12(01**1**00), -i=-12(10100)
lb(i)=4(00**1**00)

binary index




lb(i) 1 2 1 4

$s_b$ $s_j$ $s_i$

❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

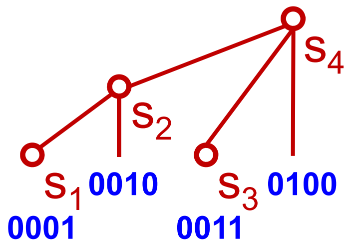
$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1}$$ (奇數 i, $s_i = a_i$)

$a_i, \{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$ b=i-lb(i) == j-lb(j)

$\{s_{i-k}\}$

# BIT RSQ 實作

例: $i=12(01\mathbf{1}00)$, $-i=-12(10100)$
$lb(i)=4(00\mathbf{1}00)$




❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 　　子樹裡 $\{a_i\}$ 的總和

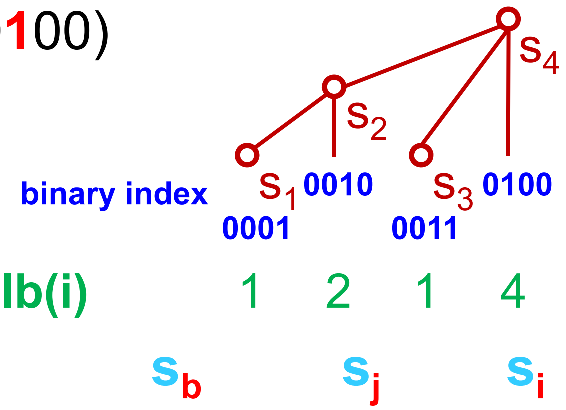
$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1}$$ (奇數 i, $s_i = a_i$)

$a_i, \{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$ 　$b=i-lb(i) == j-lb(j)$

$\{s_{i-k}\}$ ② $\{a_i\}$

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)




❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

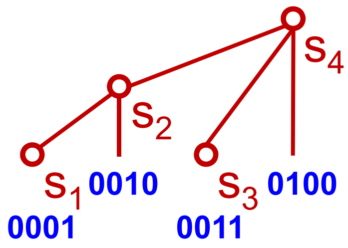
$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1}$$ (奇數 i, $s_i = a_i$)

$a_i, \{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$ b=i-lb(i) == j-lb(j)

$\{s_{i-k}\}$ ② $\{a_i\}\rightarrow\{s_i\}$

# BIT RSQ 實作

例: i=12(01**1**00), -i=-12(10100)
lb(i)=4(00**1**00)




binary index

| | | | | |
|---|---|---|---|---|
| lb(i) | 1 | 2 | 1 | 4 |
| | $s_b$ | | $s_j$ | $s_i$ |

❶ $s_i$ 子樹的大小 #define **lb(i)** ((**i**)&(-(**i**)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

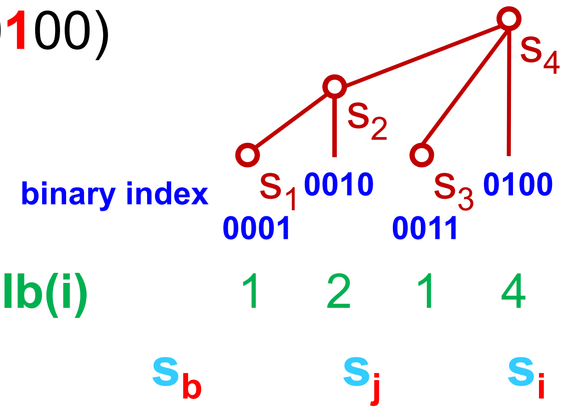
$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1}$$ (奇數 i, $s_i = a_i$)

$a_i, \{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$ $b=i-lb(i) == j-lb(j)$

$\{s_{i-k}\}$ ② $\{a_i\}\rightarrow\{s_i\}\rightarrow\{s_{i+lb(i)}\}$

# BIT RSQ 實作

例: i=12(01**1**00), -i=-12(10100)
lb(i)=4(00**1**00)




❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1}$$ (奇數 i, $s_i = a_i$)

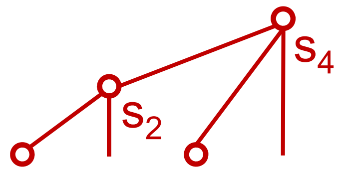
$a_i, \{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$ b=i-lb(i) == j-lb(j)

$\{s_{i-k}\}$ ② $\{a_i\}\to\{s_i\}\to\{s_{i+lb(i)}\}$

```
for (j=1; j<=n; j++) s[j] = 0;
for (i=1; i<=n; i++)
    for (j=i; j<=n; j+=lb(j))
        s[j] += a[i];
```

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)




❶ $s_i$ 子樹的大小　#define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$　　子樹裡 $\{a_i\}$ 的總和

$s_i \triangleq a_i + \ldots + a_{i-lb(i)+1}$　　(奇數 i, $s_i = a_i$)

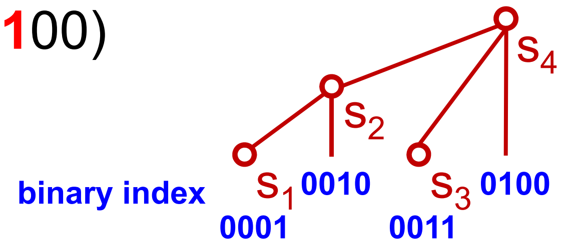
$a_i, \{s_{i-k}\}$　① $= a_i + s_{i-1} + s_{i-1-lb(i-1)} + \ldots + s_j$　$b=i-lb(i) == j-lb(j)$

$\{s_{i-k}\}$　② $\{a_i\} \rightarrow \{s_i\} \rightarrow \{s_{i+lb(i)}\}$　$\{$　$s_i=a_i,\ i=1,\ldots,n$

```
for (j=1; j<=n; j++) s[j] = 0;
for (i=1; i<=n; i++)
    for (j=i; j<=n; j+=lb(j))
        s[j] += a[i];
```

# BIT RSQ 實作

例: i=12(01**1**00), -i=-12(10100)
lb(i)=4(00**1**00)




lb(i) 1 2 1 4

$s_b$ $s_j$ $s_i$

❶ $s_i$ 子樹的大小 #define **lb(i)** ((**i**)&(-(**i**)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1} \qquad (\text{奇數 } i,\ s_i = a_i)$$

$a_i$, $\{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$ b=i-lb(i) == j-lb(j)

$\{s_{i-k}\}$ ② $\{a_i\}\rightarrow\{s_i\}\rightarrow\{s_{i+lb(i)}\}$ $\begin{cases} s_i=a_i,\ i=1,\ldots,n \\ s_{i+lb(i)} \mathrel{+}= s_i,\ i=1,\ldots,n \end{cases}$

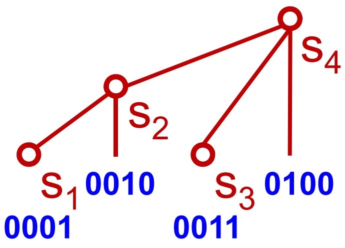
```
for (j=1; j<=n; j++) s[j] = 0;
for (i=1; i<=n; i++)
    for (j=i; j<=n; j+=lb(j))
        s[j] += a[i];
```

# BIT RSQ 實作

例: i=12(01**1**00), -i=-12(10100)
lb(i)=4(00**1**00)

❶ $s_i$ 子樹的大小  #define **lb(i)** ((**i**)&(-(**i**)))

binary index




❷ $s_i$ 的前一個子樹 $s_{i\text{-}lb(i)}$

lb(i)  1  2  1  4

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

$s_b$  $s_j$  $s_i$

❹ 由 $\{a_i\}$ 計算 $s_i$   子樹裡 $\{a_i\}$ 的總和

$s_i \triangleq a_i+\ldots+a_{i\text{-}lb(i)+1}$   (奇數 i, $s_i = a_i$)

$a_i, \{s_{i\text{-}k}\}$   ①  $= a_i+s_{i-1}+s_{i-1\text{-}lb(i-1)}+\ldots+s_j$   $b=i\text{-}lb(i) == j\text{-}lb(j)$

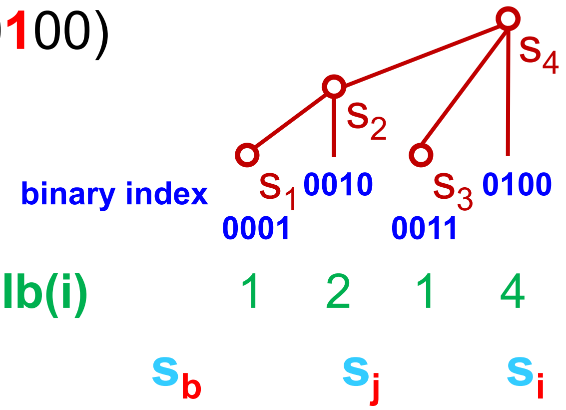
$\{s_{i\text{-}k}\}$   ②  $\{a_i\}\rightarrow\{s_i\}\rightarrow\{s_{i+lb(i)}\}$   $\begin{cases} s_i=a_i,\ i=1,\ldots,n \\ s_{i+lb(i)} \mathrel{+}= s_i,\ i=1,\ldots,n \end{cases}$

```
for (j=1; j<=n; j++) s[j] = 0;
for (i=1; i<=n; i++)
    for (j=i; j<=n; j+=lb(j))
        s[j] += a[i];
```

```
memcpy(s,a,sizeof(s));
for (i=1; i<n; i++) // if n==2^k
    s[i+lb(i)] += s[i];
```

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)




❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i + \ldots + a_{i-lb(i)+1} \quad (\text{奇數 } i,\ s_i = a_i)$$

$a_i, \{s_{i-k}\}$ ① $= a_i + s_{i-1} + s_{i-1-lb(i-1)} + \ldots + s_j$ b=i-lb(i) == j-lb(j)

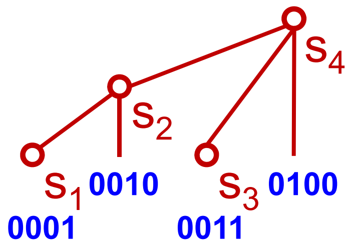
$\{s_{i-k}\}$ ② $\{a_i\} \to \{s_i\} \to \{s_{i+lb(i)}\}$

$$\begin{cases} s_i = a_i, & i=1,\ldots,n \\ s_{i+lb(i)} \mathrel{+}= s_i, & i=1,\ldots,n \end{cases}$$

❺ 修改 $a_i \to A_i$

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)




lb(i) 1 2 1 4

$s_b$ $s_j$ $s_i$

❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1} \quad (\text{奇數 } i,\ s_i = a_i)$$

$a_i, \{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$ $b=i-lb(i) == j-lb(j)$

$\{s_{i-k}\}$ ② $\{a_i\}\rightarrow\{s_i\}\rightarrow\{s_{i+lb(i)}\}$ $\begin{cases} s_i=a_i, & i=1,\ldots,n \\ s_{i+lb(i)}\mathrel{+}=s_i, & i=1,\ldots,n \end{cases}$

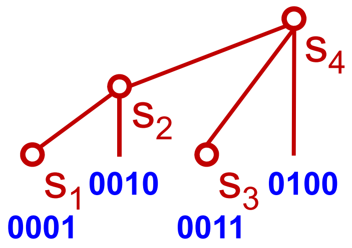
❺ 修改 $a_i\rightarrow A_i$

① $a_i$ saved

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)

binary index




lb(i) 1 2 1 4

$s_b$ $s_j$ $s_i$

❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1}$$ (奇數 i, $s_i = a_i$)

$a_i, \{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$ $b=i-lb(i) == j-lb(j)$

$\{s_{i-k}\}$ ② $\{a_i\}\rightarrow\{s_i\}\rightarrow\{s_{i+lb(i)}\}$

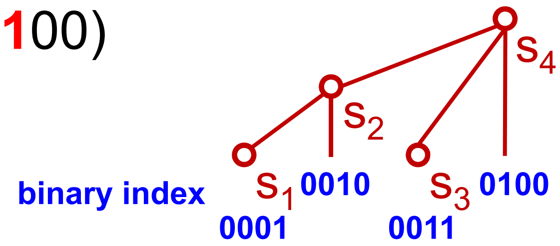
$$\begin{cases} s_i=a_i,\ i=1,\ldots,n \\ s_{i+lb(i)}\mathrel{+}=s_i,\ i=1,\ldots,n \end{cases}$$

❺ 修改 $a_i \rightarrow A_i$

① $a_i$ saved $s_{i_1}$

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)




lb(i) 1 2 1 4

$s_b$ $s_j$ $s_i$

❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1} \quad (\text{奇數 } i,\ s_i = a_i)$$

$a_i, \{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$ b=i-lb(i) == j-lb(j)

$\{s_{i-k}\}$ ② $\{a_i\}\rightarrow\{s_i\}\rightarrow\{s_{i+lb(i)}\}$ $\begin{cases} s_i=a_i,\ i=1,\ldots,n \\ s_{i+lb(i)}\mathrel{+}=s_i,\ i=1,\ldots,n \end{cases}$

❺ 修改 $a_i \rightarrow A_i$

① $a_i$ saved $s_{i_1}$

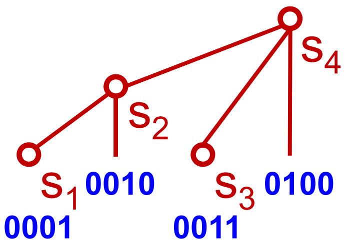
$i_1=i$

# BIT RSQ 實作

例: i=12(01**1**00), -i=-12(10100)
lb(i)=4(00**1**00)

❶ $s_i$ 子樹的大小　#define **lb(i)** ((i)&(-(i)))

binary index




❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

lb(i)　1　2　1　4

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

$s_b$　$s_j$　$s_i$

❹ 由 $\{a_i\}$ 計算 $s_i$　　子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1}$$ （奇數 i, $s_i = a_i$)

$a_i, \{s_{i-k}\}$　① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$　$b=i-lb(i) == j-lb(j)$

$\{s_{i-k}\}$　② $\{a_i\}\rightarrow\{s_i\}\rightarrow\{s_{i+lb(i)}\}$　$\begin{cases} s_i=a_i,\ i=1,\ldots,n \\ s_{i+lb(i)} \mathrel{+}= s_i,\ i=1,\ldots,n \end{cases}$

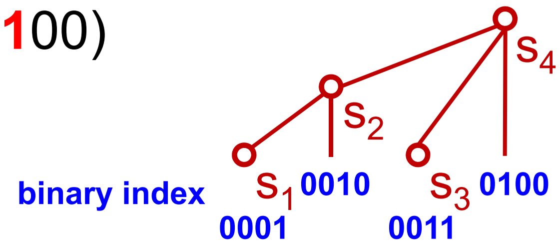
❺ 修改 $a_i \rightarrow A_i$

① $a_i$ saved　$s_{i_1} \rightarrow s_{i_2}$

$i_1=i$

# BIT RSQ 實作

例: i=12(01**1**00), -i=-12(10100)
lb(i)=4(00**1**00)




lb(i)  1  2  1  4

$s_b$  $s_j$  $s_i$

❶ $s_i$ 子樹的大小   #define **lb(i)** ((**i**)&(-(**i**)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$     子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1} \qquad (\text{奇數 } i,\ s_i = a_i)$$

$a_i, \{s_{i-k}\}$   ①  $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$   $b=i-lb(i) == j-lb(j)$

$\{s_{i-k}\}$   ②  $\{a_i\}\rightarrow\{s_i\}\rightarrow\{s_{i+lb(i)}\}$   $\begin{cases} s_i=a_i,\ i=1,\ldots,n \\ s_{i+lb(i)} \mathrel{+}= s_i,\ i=1,\ldots,n \end{cases}$

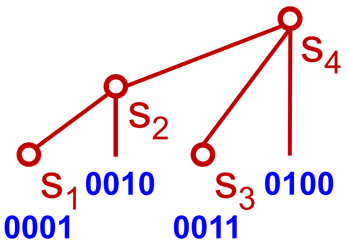
❺ 修改 $a_i \rightarrow A_i$

① $a_i$ saved   $s_{i_1} \rightarrow s_{i_2}$

$i_1=i \quad i_2=i_1+lb(i_1)$

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)




binary index

lb(i) 1 2 1 4

$s_b$ $s_j$ $s_i$

❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1} \quad (\text{奇數 } i,\ s_i = a_i)$$

$a_i, \{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$ $b=i-lb(i) == j-lb(j)$

$\{s_{i-k}\}$ ② $\{a_i\}\rightarrow\{s_i\}\rightarrow\{s_{i+lb(i)}\}$ $\begin{cases} s_i=a_i,\ i=1,\ldots,n \\ s_{i+lb(i)}\mathrel{+}=s_i,\ i=1,\ldots,n \end{cases}$

❺ 修改 $a_i\rightarrow A_i$

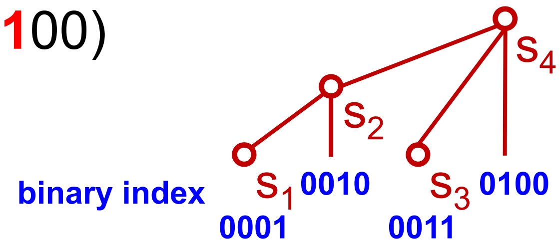
① $a_i$ saved

$s_{i_1} \rightarrow s_{i_2} \rightarrow$

$i_1=i \quad i_2=i_1+lb(i_1)$

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)




lb(i) 1 2 1 4

$s_b$ $s_j$ $s_i$

❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1} \quad (\text{奇數 } i,\ s_i = a_i)$$

$a_i, \{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$ b=i-lb(i) == j-lb(j)

$\{s_{i-k}\}$ ② $\{a_i\}\rightarrow\{s_i\}\rightarrow\{s_{i+lb(i)}\}$ $\begin{cases} s_i=a_i,\ i=1,\ldots,n \\ s_{i+lb(i)} \mathrel{+}= s_i,\ i=1,\ldots,n \end{cases}$

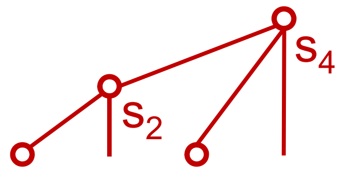
❺ 修改 $a_i\rightarrow A_i$

① $a_i$ saved $s_{i_1} \rightarrow s_{i_2} \rightarrow$

$i_1=i \quad i_2=i_1+lb(i_1) \quad i_3=i_2+lb(i_2)$

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)




binary index 0001 0010 0011 0100

lb(i) 1 2 1 4

$s_b$ $s_j$ $s_i$

❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1} \quad (\text{奇數 } i,\ s_i = a_i)$$

$a_i, \{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$ b=i-lb(i) == j-lb(j)

$\{s_{i-k}\}$ ② $\{a_i\}\rightarrow\{s_i\}\rightarrow\{s_{i+lb(i)}\}$ $\begin{cases} s_i=a_i,\ i=1,\ldots,n \\ s_{i+lb(i)} \mathrel{+}= s_i,\ i=1,\ldots,n \end{cases}$

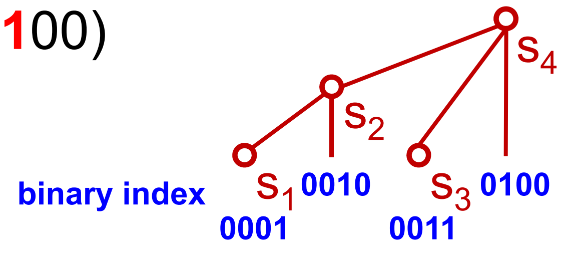
❺ 修改 $a_i \rightarrow A_i$

① $a_i$ saved $s_{i_1} \rightarrow s_{i_2} \rightarrow$ 。。。

$i_1=i$ $\quad i_2=i_1+lb(i_1)$ $\quad i_3=i_2+lb(i_2)$ 。。。

# BIT RSQ 實作

例: i=12(01**1**00), -i=-12(10100)
lb(i)=4(00**1**00)




lb(i) 1 2 1 4

$s_b$ $s_j$ $s_i$

❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1} \quad (\text{奇數 } i,\ s_i = a_i)$$

$a_i, \{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$ $b=i-lb(i) == j-lb(j)$

$\{s_{i-k}\}$ ② $\{a_i\}\rightarrow\{s_i\}\rightarrow\{s_{i+lb(i)}\}$ $\begin{cases} s_i=a_i,\ i=1,\ldots,n \\ s_{i+lb(i)}+=s_i,\ i=1,\ldots,n \end{cases}$

❺ 修改 $a_i\rightarrow A_i$

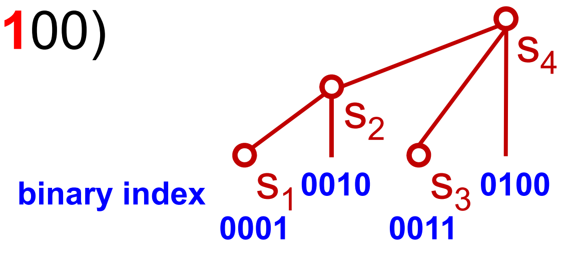
① $a_i$ saved

$$s_{i_1} \rightarrow s_{i_2} \rightarrow \ \circ\ \circ\ \circ \rightarrow s_n$$

$$i_1=i \quad i_2=i_1+lb(i_1) \quad i_3=i_2+lb(i_2) \quad \circ\ \circ\ \circ$$

# BIT RSQ 實作

例: i=12(01**1**00), -i=-12(10100)
lb(i)=4(00**1**00)




| lb(i) | 1 | 2 | 1 | 4 |
|---|---|---|---|---|
| | $s_b$ | | $s_j$ | $s_i$ |

❶ $s_i$ 子樹的大小 #define **lb(i)** ((**i**)&(-(**i**)))

❷ $s_i$ 的前一個子樹 $s_{i\text{-}lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i+\ldots+a_{i\text{-}lb(i)+1}$$ (奇數 i, $s_i = a_i$)

$a_i$, $\{s_{i\text{-}k}$ … $_{(i-1)}+\ldots+s_j$  $b=i\text{-}lb(i) == j\text{-}lb(j)$

```
for (d=Ai-a[i], j=i; j<=n; j+=lb(j))
    s[j] += d;
```

$\{s_{i\text{-}k}\}$ ② $\{a_i\}\rightarrow\{s_i\}\rightarrow\{s_{i+lb(i)}\}$

$$\begin{cases} s_i=a_i,\ i=1,\ldots,n \\ s_{i+lb(i)}+=s_i,\ i=1,\ldots,n \end{cases}$$

❺ 修改 $a_i \rightarrow A_i$

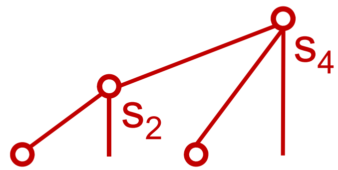
① $a_i$ saved

$$s_{i_1} \rightarrow s_{i_2} \rightarrow \circ\circ\circ \rightarrow s_n$$

$$i_1=i \quad i_2=i_1+lb(i_1) \quad i_3=i_2+lb(i_2) \quad \circ\circ\circ$$

# BIT RSQ 實作

例: i=12(01**1**00), -i=-12(10100)
lb(i)=4(00**1**00)




| binary index | 0001 | 0010 | 0011 | 0100 |
|---|---|---|---|---|
| lb(i) | 1 | 2 | 1 | 4 |
| | $s_b$ | $s_j$ | | $s_i$ |

❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1}$$ (奇數 i, $s_i = a_i$)

$a_i$, $\{s_{i-k}$ … $_{(i-1)}+\ldots+s_j$  b=i-lb(i) == j-lb(j)

```
for (d=Ai-a[i], j=i; j<=n; j+=lb(j))
    s[j] += d;
```

$\{s_{i-k}\}$ ② $\{a_i\}\to\{s_i\}\to\{s_{i+lb(i)}\}$

$$\begin{cases} s_i=a_i,\ i=1,\ldots,n \\ s_{i+lb(i)}\mathrel{+}=s_i,\ i=1,\ldots,n \end{cases}$$

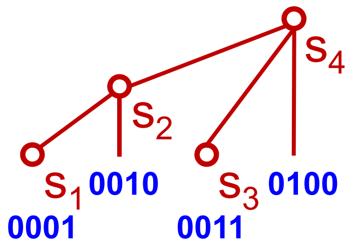
❺ 修改 $a_i \to A_i$

① $a_i$ saved  $s_{i_1} \to s_{i_2} \to$ 。。。$\to s_n$

② $a_i$ not saved  $i_1=i$  $i_2=i_1+lb(i_1)$  $i_3=i_2+lb(i_2)$ 。。。

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)




binary index

lb(i) 1 2 1 4

$s_b$ $s_j$ $s_i$

❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 s 子樹裡 $\{a_i\}$ 的總和

```
for (ai=s[i],b=i-lb(i),j=i-1; j>b; j-=lb(j))
    ai -= s[j];
```

(奇數 i, $s_i = a_i$)

$a_i$, $\{s_{i-k}$

```
for (d=Ai -ai , j=i; j<=n; j+=lb(j))
    s[j] += d;
```

$_{(i-1)}+\ldots+s_j$ b=i-lb(i) == j-lb(j)

$\{s_{i-k}\}$ ② $\{a_i\}\to\{s_i\}\to\{s_{i+lb(i)}\}$

$$\begin{cases} s_i=a_i, & i=1,\ldots,n \\ s_{i+lb(i)} += s_i, & i=1,\ldots,n \end{cases}$$

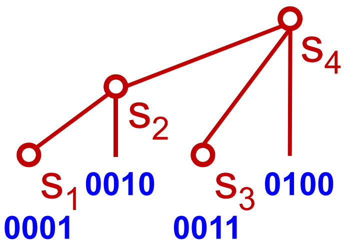
❺ 修改 $a_i \to A_i$

① $a_i$ saved $s_{i_1} \to s_{i_2} \to \ \circ\ \circ\ \circ \to s_n$

② $a_i$ not saved $i_1=i$ $i_2=i_1+lb(i_1)$ $i_3=i_2+lb(i_2)$ $\circ\ \circ\ \circ$

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)




binary index

| | | | | |
|---|---|---|---|---|
| lb(i) | 1 | 2 | 1 | 4 |
| | $s_b$ | $s_j$ | | $s_i$ |

❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1}$ (奇數 i, $s_i = a_i$)

$a_i, \{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$ $b=i-lb(i) == j-lb(j)$

$\{s_{i-k}\}$ ② $\{a_i\}\rightarrow\{s_i\}\rightarrow\{s_{i+lb(i)}\}$ $\begin{cases} s_i=a_i,\ i=1,\ldots,n \\ s_{i+lb(i)}\mathrel{+}=s_i,\ i=1,\ldots,n \end{cases}$

❺ 修改 $a_i \rightarrow A_i$

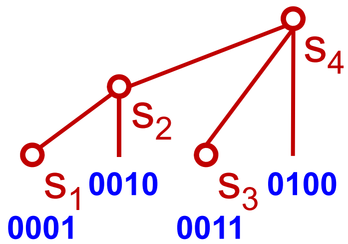
① $a_i$ saved $s_{i_1} \rightarrow s_{i_2} \rightarrow$ 。。。$\rightarrow s_n$

② $a_i$ not saved $i_1=i$ $i_2=i_1+lb(i_1)$ $i_3=i_2+lb(i_2)$ 。。。

❻ 區間和 $\sum_{k=i}^{j} a_k$

# BIT RSQ 實作

例: i=12(01**1**00), -i=-12(10100)
lb(i)=4(00**1**00)




binary index

lb(i) 1 2 1 4

$s_b$ $s_j$ $s_i$

❶ $s_i$ 子樹的大小 #define **lb(i)** ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1}$ (奇數 i, $s_i = a_i$)

$a_i, \{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$ $b=i-lb(i) == j-lb(j)$

$\{s_{i-k}\}$ ② $\{a_i\}\to\{s_i\}\to\{s_{i+lb(i)}\}$ $\begin{cases} s_i=a_i, & i=1,\ldots,n \\ s_{i+lb(i)} \mathrel{+}= s_i, & i=1,\ldots,n \end{cases}$

❺ 修改 $a_i \to A_i$

① $a_i$ saved $s_{i_1} \to s_{i_2} \to \ \circ\ \circ\ \circ \to s_n$

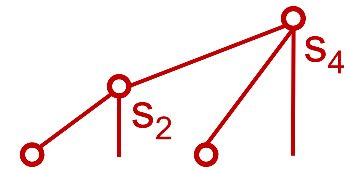
② $a_i$ not saved $i_1=i$ $i_2=i_1+lb(i_1)$ $i_3=i_2+lb(i_2)$

```
for (rsq=0,k=j; k>0; k-=lb(k))
   rsq += s[k];
```

❻ 區間和 $\sum_{k=i}^{j} a_k = \sum_{k=1}^{j} a_k$

# BIT RSQ 實作

例: i=12(01100), -i=-12(10100)
lb(i)=4(00100)




❶ $s_i$ 子樹的大小 #define lb(i) ((i)&(-(i)))

❷ $s_i$ 的前一個子樹 $s_{i-lb(i)}$

❸ $s_i$ 的父節點 $s_{i+lb(i)}$

❹ 由 $\{a_i\}$ 計算 $s_i$ 子樹裡 $\{a_i\}$ 的總和

$s_i \triangleq a_i+\ldots+a_{i-lb(i)+1}$ (奇數 i, $s_i = a_i$)

$a_i, \{s_{i-k}\}$ ① $= a_i+s_{i-1}+s_{i-1-lb(i-1)}+\ldots+s_j$ $b=i-lb(i) == j-lb(j)$

$\{s_{i-k}\}$ ② $\{a_i\}\rightarrow\{s_i\}\rightarrow\{s_{i+lb(i)}\}$ $\begin{cases} s_i=a_i,\ i=1,\ldots,n \\ s_{i+lb(i)}\mathrel{+}=s_i,\ i=1,\ldots,n \end{cases}$

❺ 修改 $a_i\rightarrow A_i$

$s_{i_1} \rightarrow s_{i_2} \rightarrow \circ\circ\circ \rightarrow s_n$

① $a_i$ saved

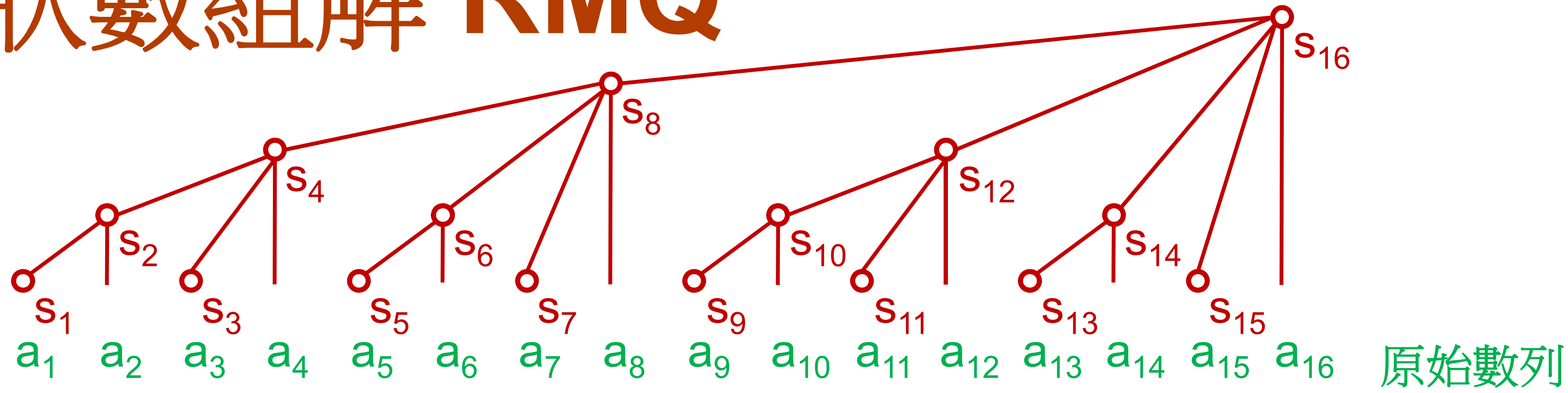
② $a_i$ not saved $i_1=i$ $i_2=i_1+lb(i_1)$ $i_3=i_2+lb(i_2)$

```
for (rsq=0,k=j; k>0; k-=lb(k))
    rsq += s[k];
for (k=i-1; k>0; k-=lb(k))
    rsq -= s[k];
```

❻ 區間和 $\sum_{k=i}^{j} a_k = \sum_{k=1}^{j} a_k - \sum_{k=1}^{i-1} a_k$

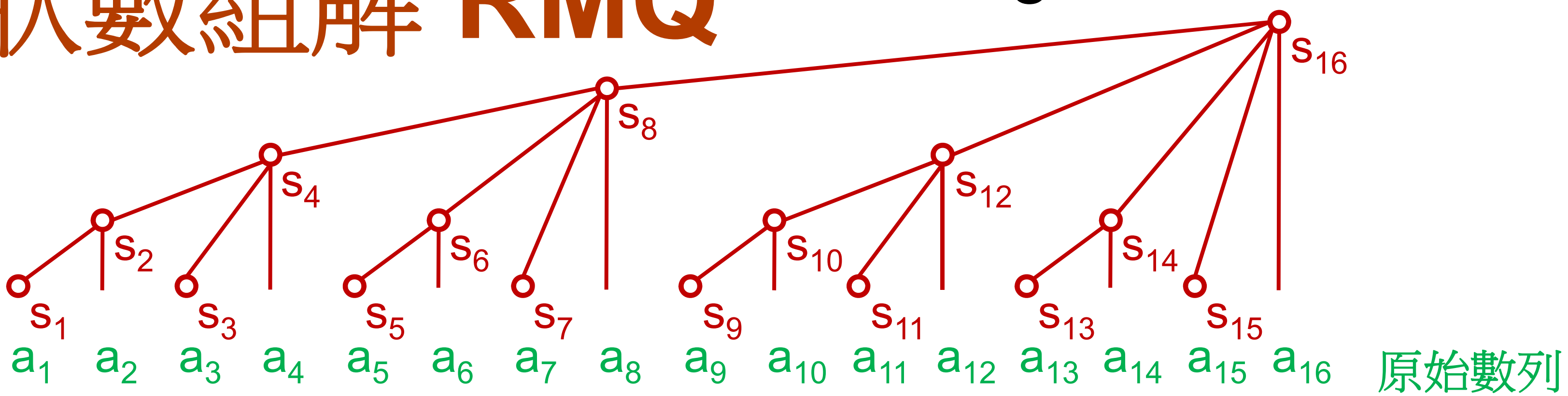
# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery



# 樹狀數組解 RMQ

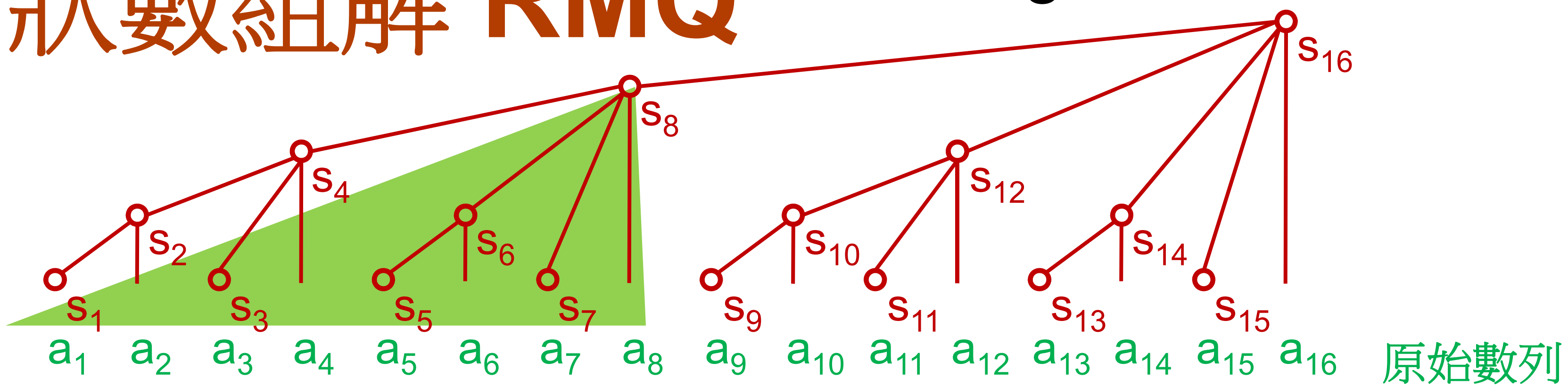
**Range Maximum Query**




- 建構 **build(), $\{a_i\} \rightarrow \{s_i\}$**

# 樹狀數組解 RMQ

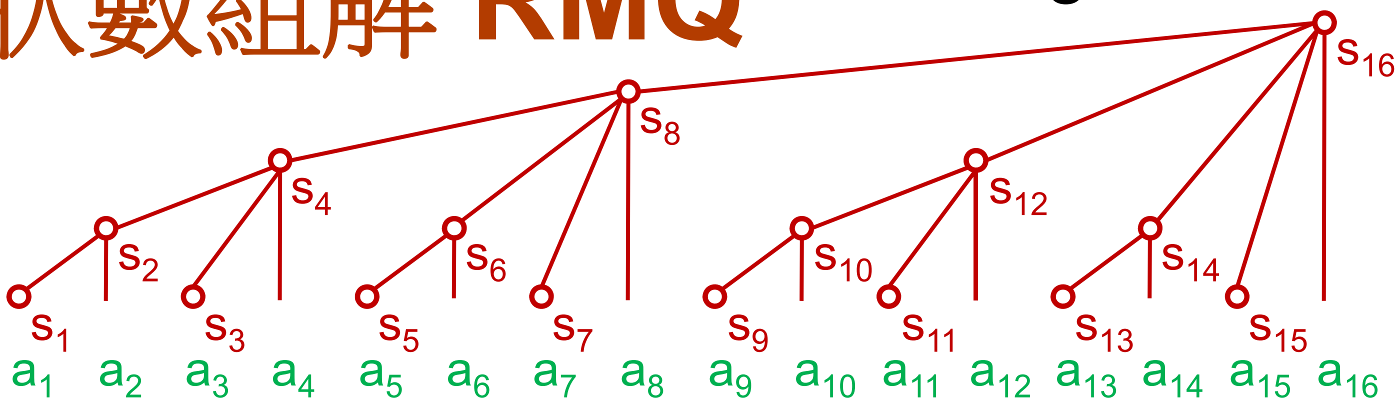
**R**ange **M**aximum **Q**uery




- 建構 **build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

# 樹狀數組解 RMQ

Range Maximum Query

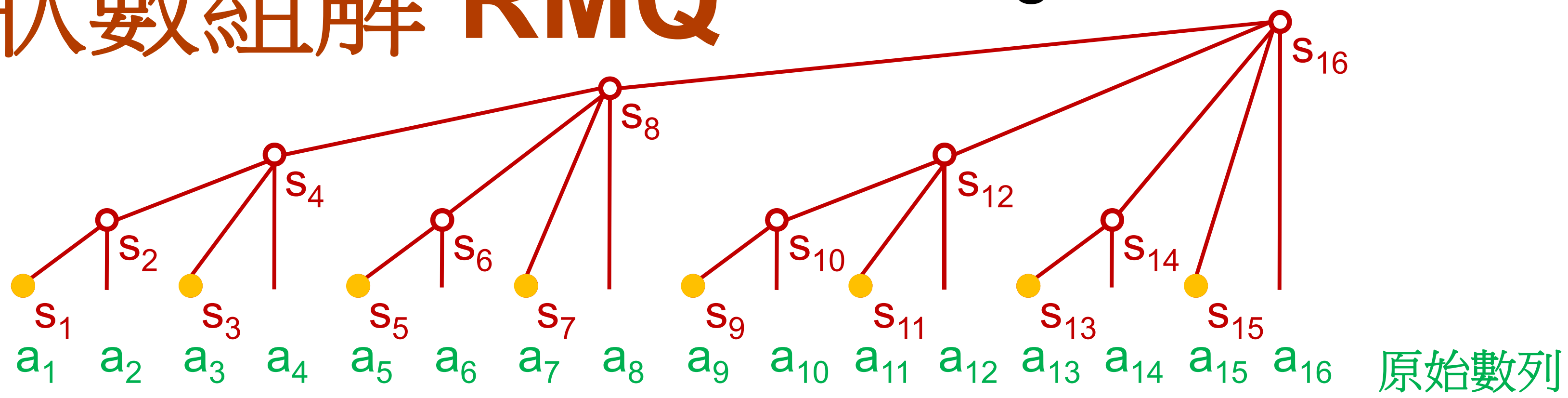



- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

  1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery



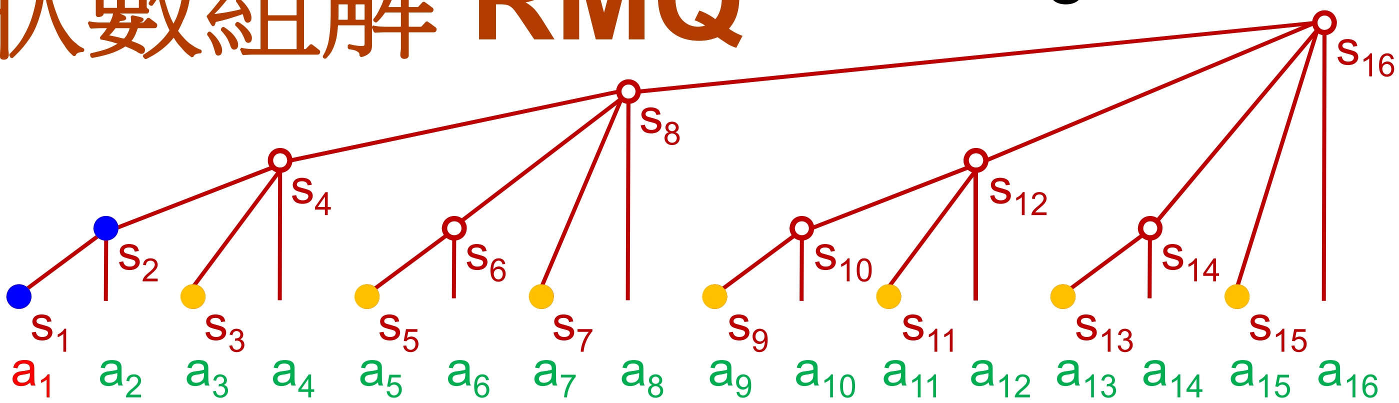

- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$

2. $s_i = a_i$, i=1,…,16

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

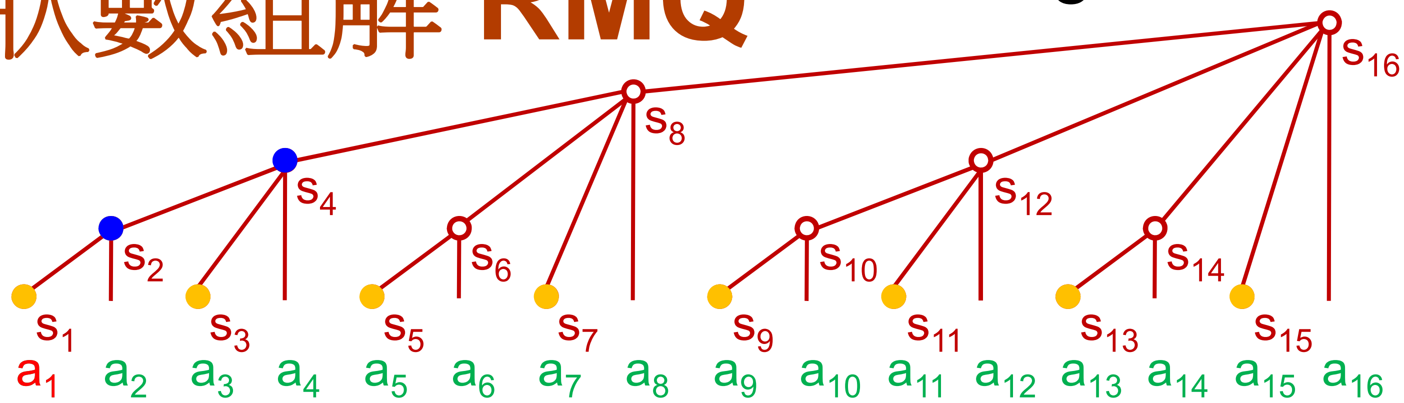
  1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$

  2. $s_i = a_i$, i=1,…,16

  3. $s_2=\max(s_2,\mathbf{s_1})$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

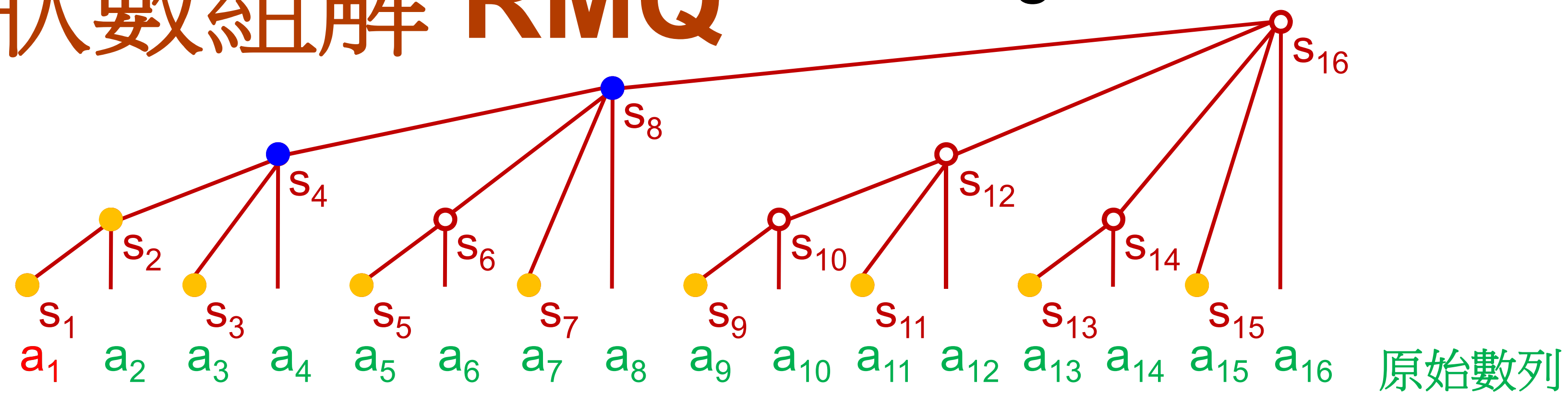
1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery

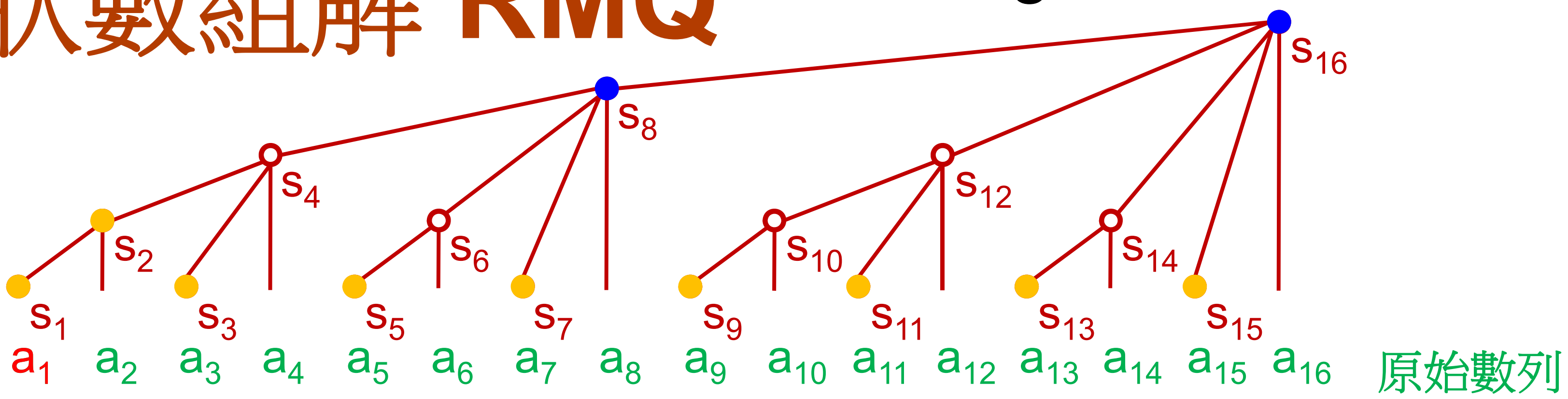



- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$
2. $s_i = a_i$, i=1,…,16
3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery

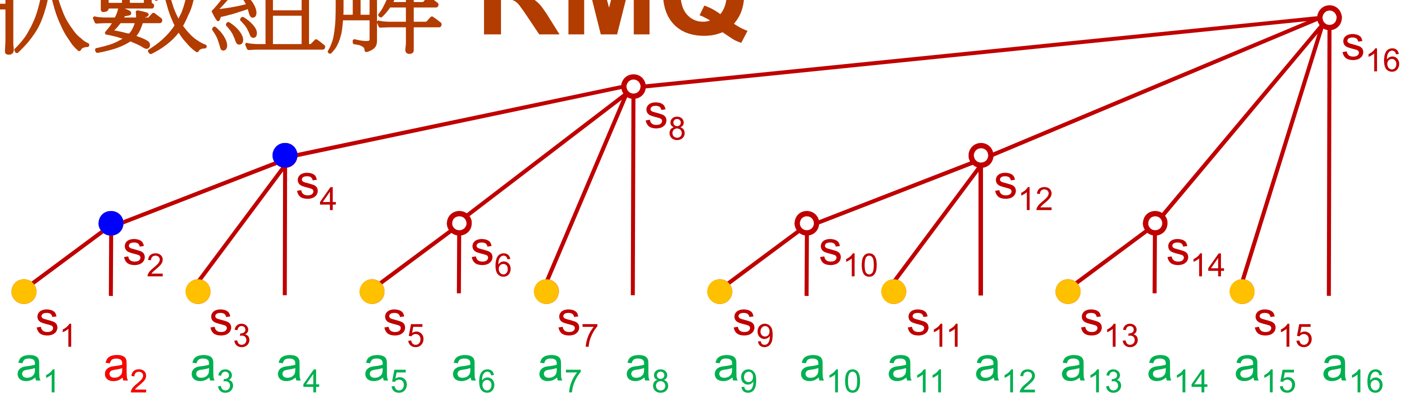



- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$
2. $s_i = a_i$, i=1,…,16
3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1, \ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$

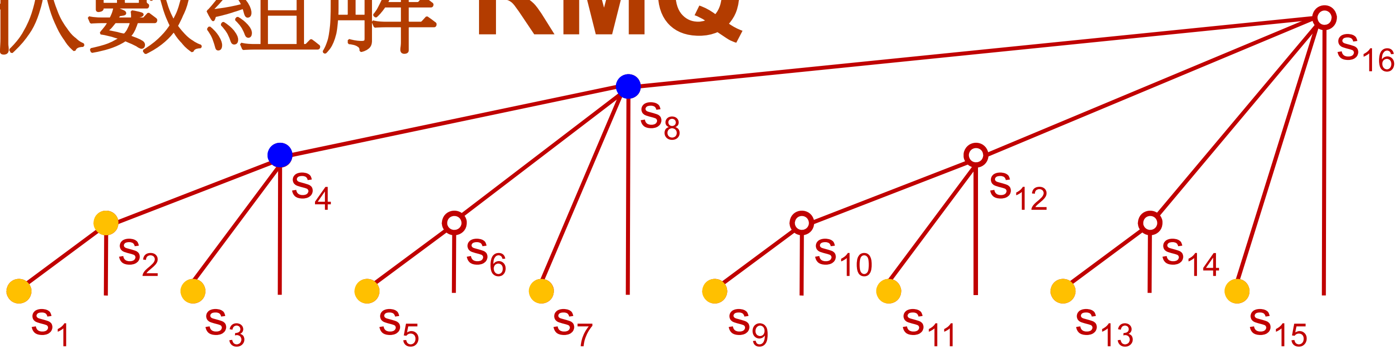
2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   $s_4=\max(s_4,\mathbf{s_2})$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$

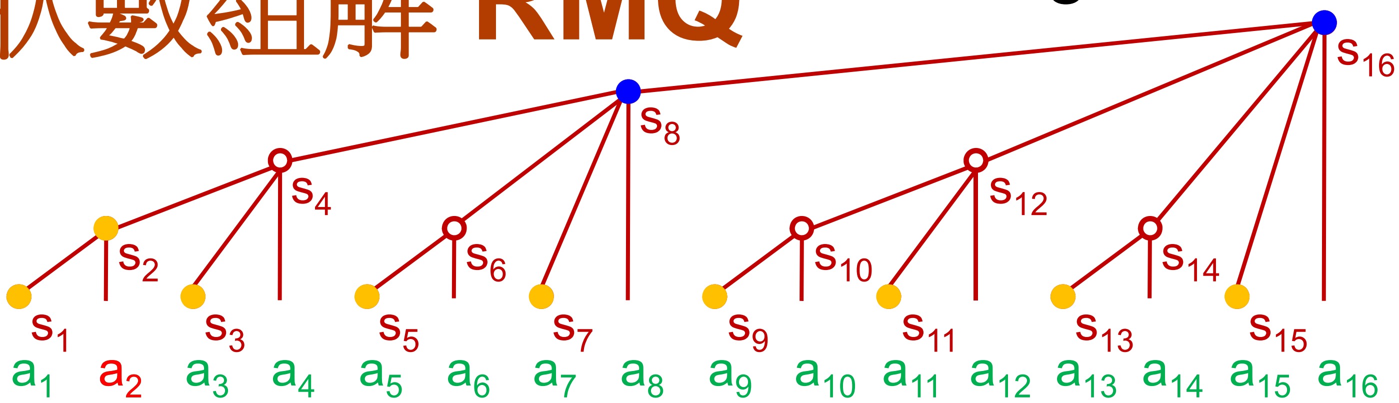
2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   $s_4=\max(s_4,\mathbf{s_2})$
   $s_8=\max(s_8,s_4)$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$

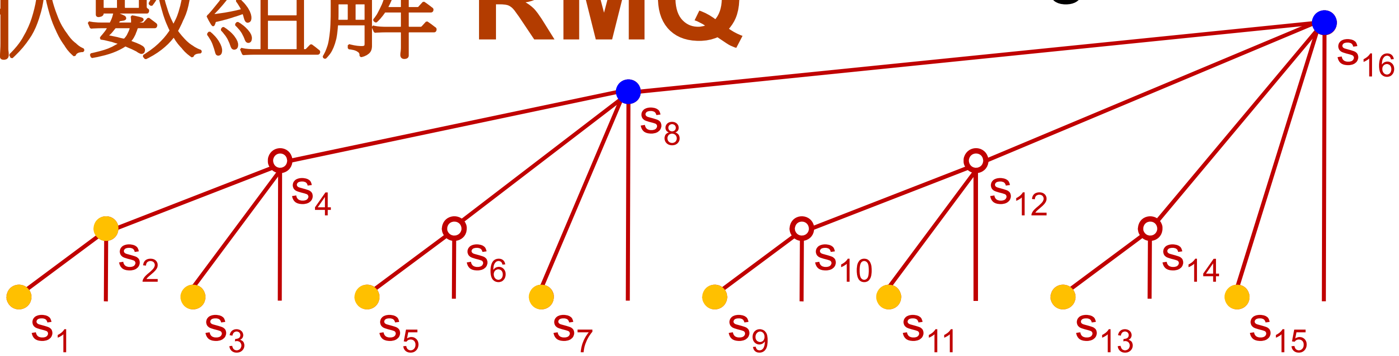
2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   $s_4=\max(s_4,\mathbf{s_2})$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$

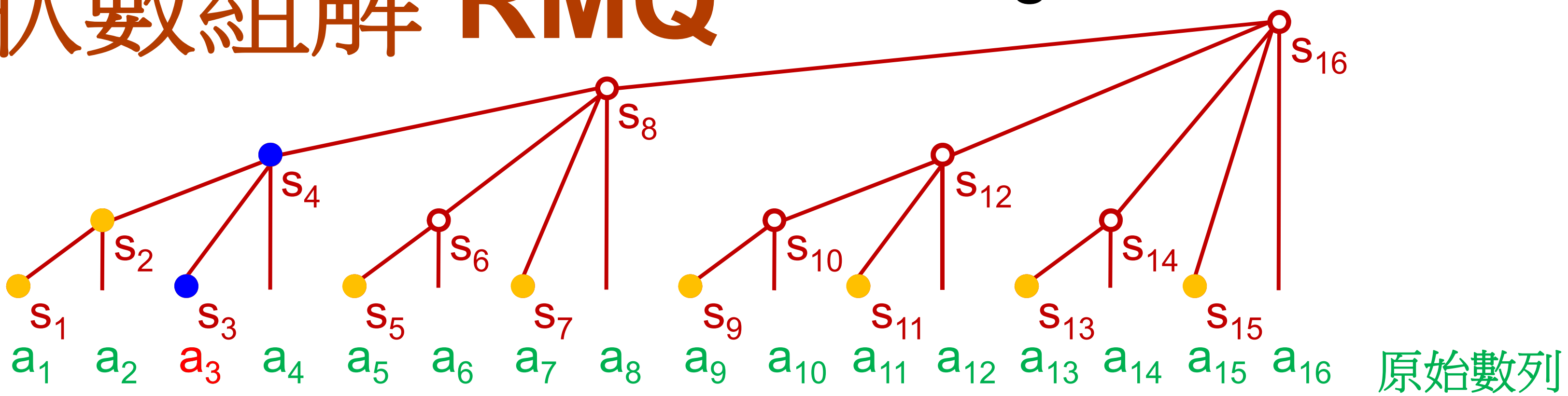
2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,s_1)$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_4=\max(s_4,s_2)$~~
   ~~$s_8=\max(s_8,s_4)$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$

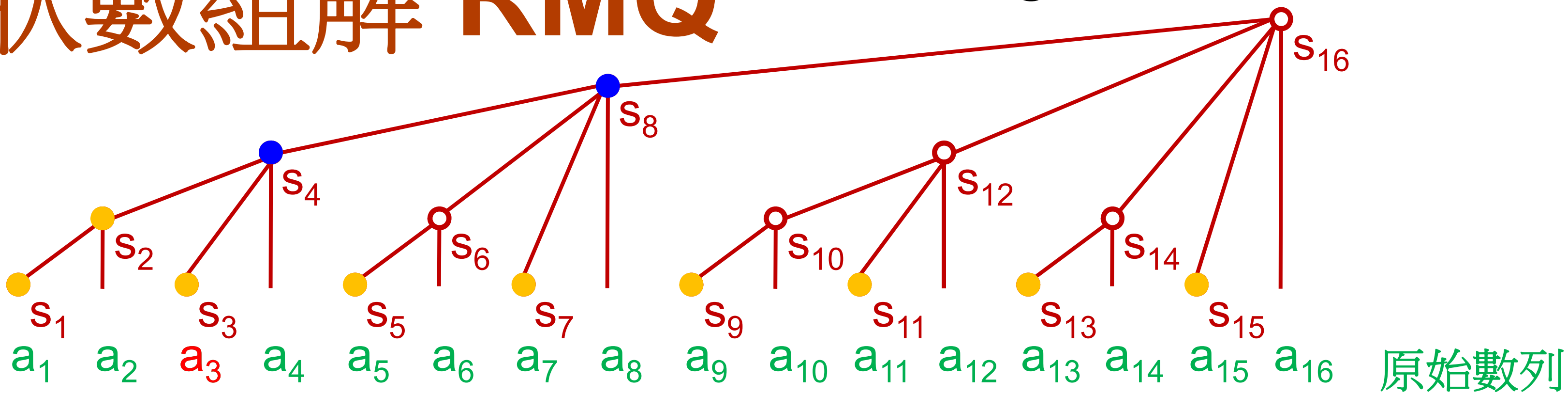
2. $s_i = a_i$, i=1,…,16 　　$s_4=\max(s_4,\mathbf{s_3})$

3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   $\cancel{s_4=\max(s_4,\mathbf{s_2})}$
   $\cancel{s_8=\max(s_8,s_4)}$
   $\cancel{s_{16}=\max(s_{16},s_8)}$

# 樹狀數組解 RMQ

**Range Maximum Query**




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ $(s_8)$ 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$

2. $s_i = a_i$, $i=1,\ldots,16$

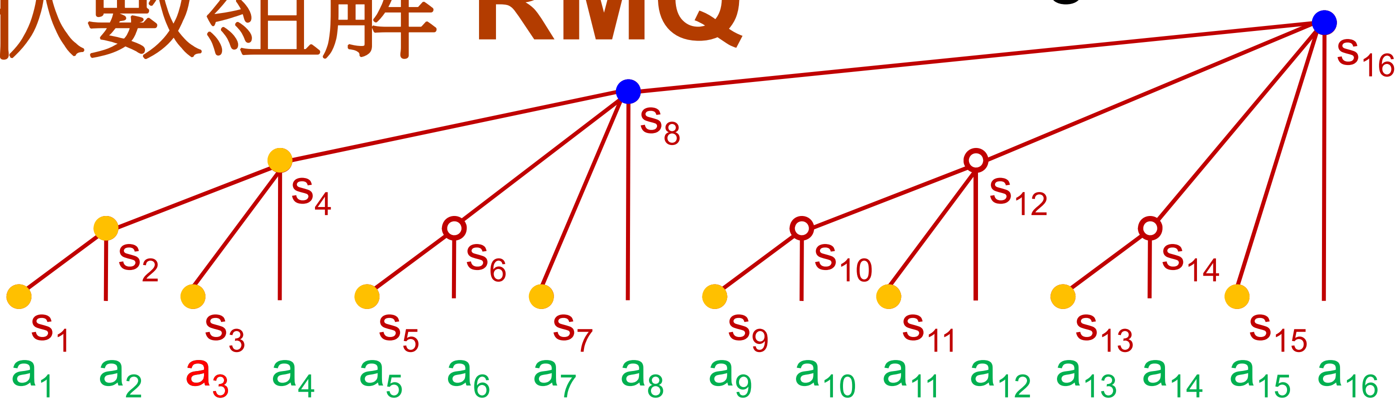
$s_4=\max(s_4,s_3)$
$s_8=\max(s_8,s_4)$

3. $s_2=\max(s_2,s_1)$
$s_4=\max(s_4,s_2)$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_4=\max(s_4,s_2)}$
$\cancel{s_8=\max(s_8,s_4)}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




原始數列

- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$

2. $s_i = a_i$, i=1,…,16

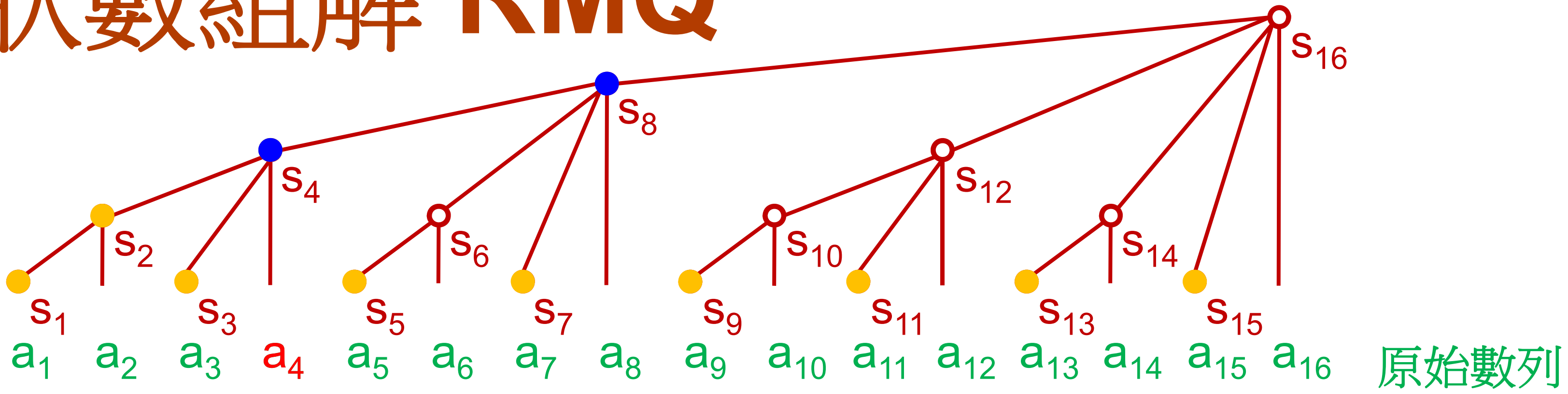
3. $s_2=\max(s_2,s_1)$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

$s_4=\max(s_4,s_3)$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_4=\max(s_4,s_2)}$
$\cancel{s_8=\max(s_8,s_4)}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- 建構 **build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

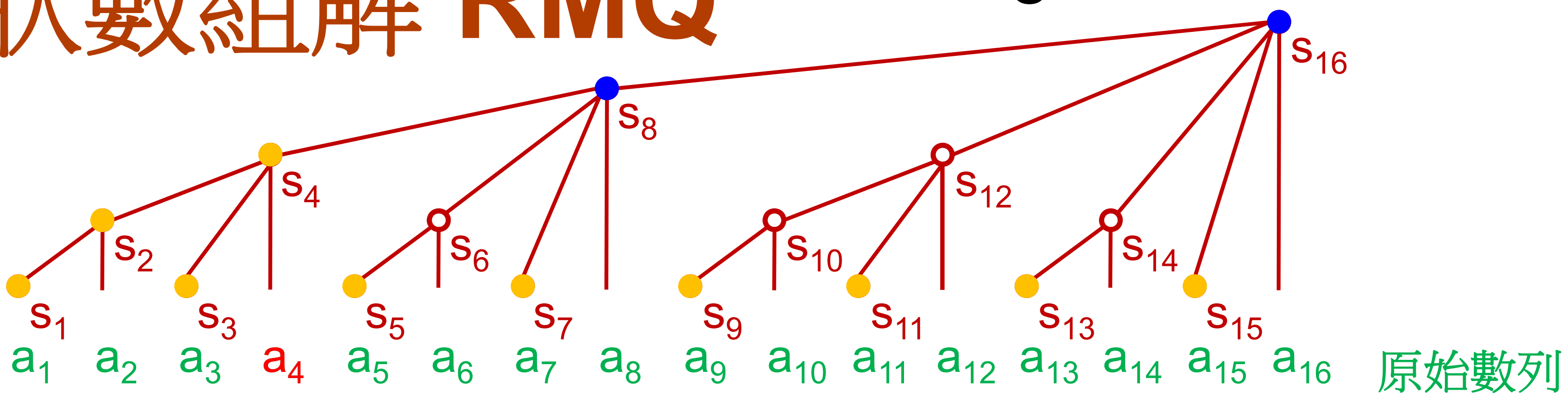
   $s_4=\max(s_4,\mathbf{s_3})$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   $s_8=\max(s_8,\mathbf{s_4})$

   ~~$s_4=\max(s_4,\mathbf{s_2})$~~
   ~~$s_8=\max(s_8,s_4)$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

# 樹狀數組解 RMQ

**Range Maximum Query**




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,s_1)$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

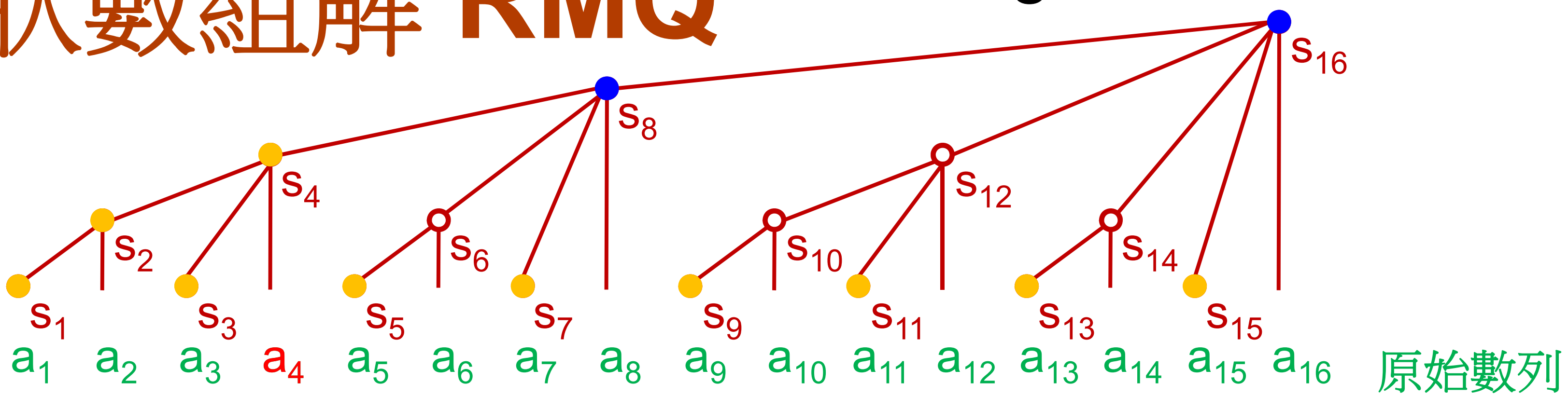
   $\cancel{s_4=\max(s_4,s_2)}$
   $\cancel{s_8=\max(s_8,s_4)}$
   $\cancel{s_{16}=\max(s_{16},s_8)}$

$s_4=\max(s_4,s_3)$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1, \ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

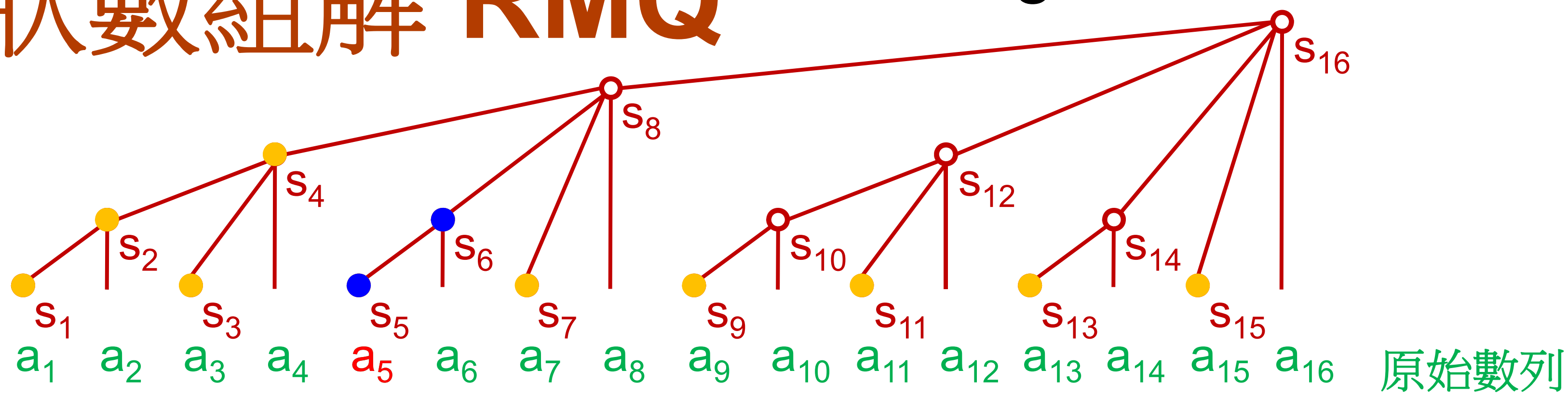
   ~~$s_4=\max(s_4,\mathbf{s_2})$~~
   ~~$s_8=\max(s_8,s_4)$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

   $s_4=\max(s_4,\mathbf{s_3})$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_8=\max(s_8,\mathbf{s_4})$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$

2. $s_i = a_i$, $i=1,\ldots,16$

3. $s_2=\max(s_2,\mathbf{s_1})$
$s_4=\max(s_4,s_2)$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_4=\max(s_4,\mathbf{s_2})}$
$\cancel{s_8=\max(s_8,s_4)}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

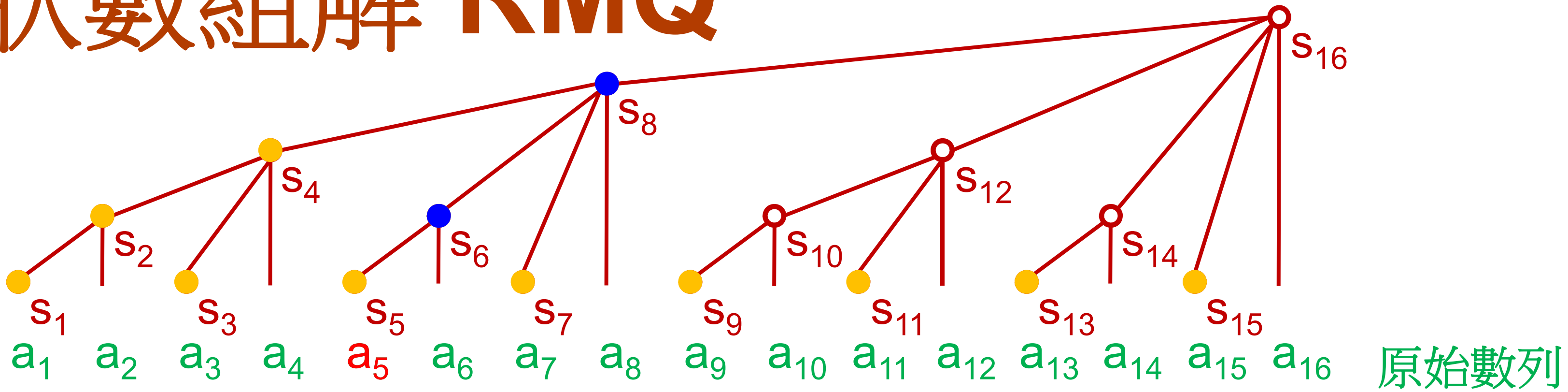
$s_4=\max(s_4,\mathbf{s_3})$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,\mathbf{s_4})}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_6=\max(s_6,\mathbf{s_5})$

# 樹狀數組解 RMQ

**Range Maximum Query**




- **建構 build(), $\{a_i\} \to \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$

2. $s_i = a_i$, $i=1,\ldots,16$

3. $s_2=\max(s_2,s_1)$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_4=\max(s_4,s_2)$~~
   ~~$s_8=\max(s_8,s_4)$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

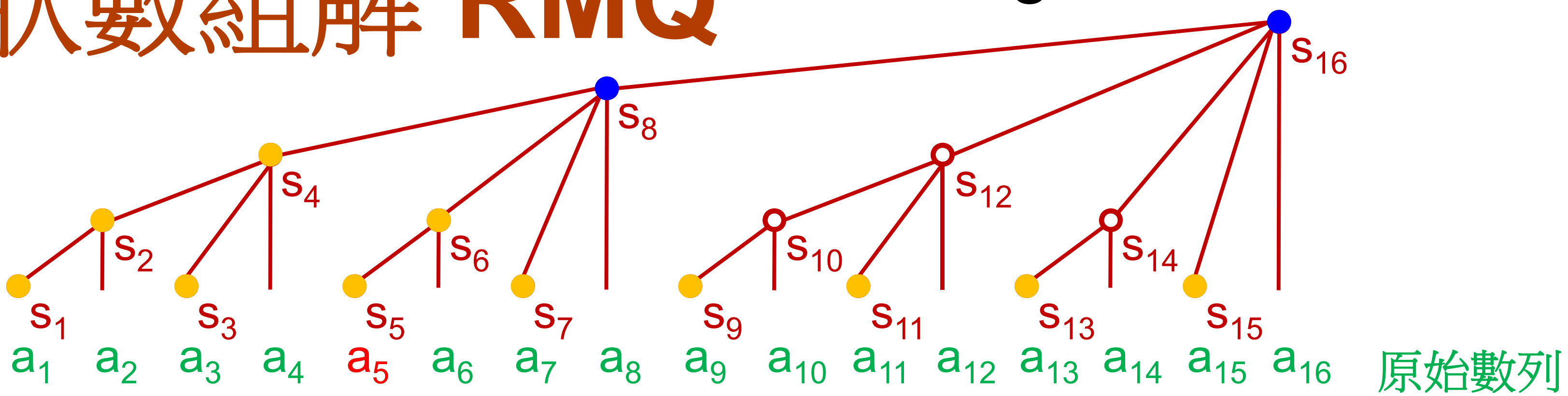
   $s_4=\max(s_4,s_3)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_8=\max(s_8,s_4)$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

   $s_6=\max(s_6,s_5)$
   $s_8=\max(s_8,s_6)$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_4=\max(s_4,\mathbf{s_2})$~~
   ~~$s_8=\max(s_8,s_4)$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

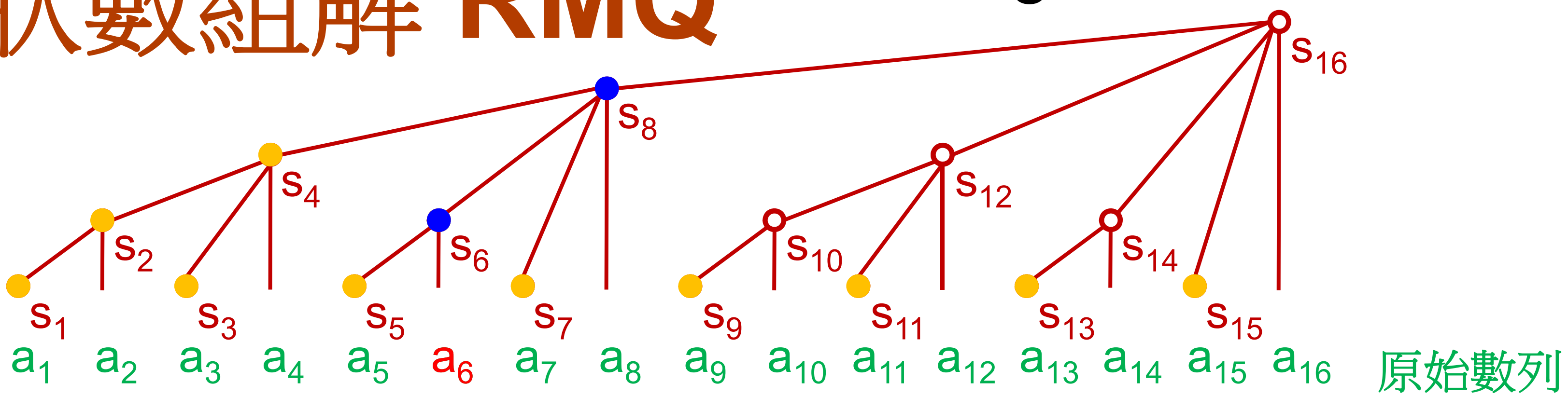
   $s_4=\max(s_4,\mathbf{s_3})$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_8=\max(s_8,\mathbf{s_4})$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

   $s_6=\max(s_6,\mathbf{s_5})$
   $s_8=\max(s_8,s_6)$
   $s_{16}=\max(s_{16},s_8)$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,s_1)$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_4=\max(s_4,s_2)$~~
   ~~$s_8=\max(s_8,s_4)$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

$s_4=\max(s_4,s_3)$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

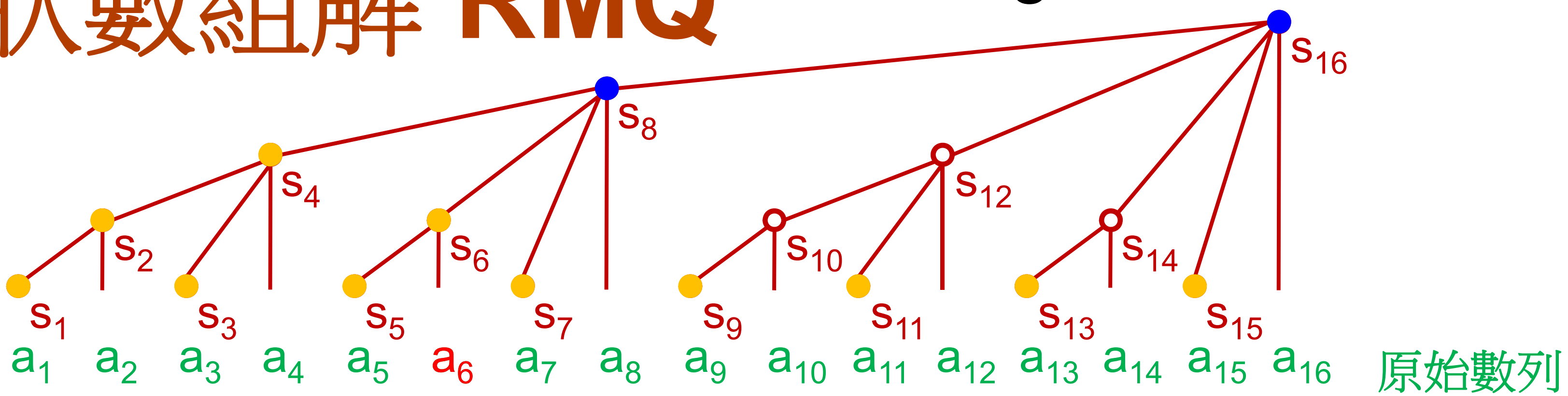
~~$s_8=\max(s_8,s_4)$~~
~~$s_{16}=\max(s_{16},s_8)$~~

$s_6=\max(s_6,s_5)$
$s_8=\max(s_8,s_6)$
$s_{16}=\max(s_{16},s_8)$

$s_8=\max(s_8,s_6)$

# 樹狀數組解 RMQ

Range Maximum Query




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$
2. $s_i = a_i$, i=1,…,16
3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   $\cancel{s_4=\max(s_4,\mathbf{s_2})}$
   $\cancel{s_8=\max(s_8,s_4)}$
   $\cancel{s_{16}=\max(s_{16},s_8)}$

   $s_4=\max(s_4,\mathbf{s_3})$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

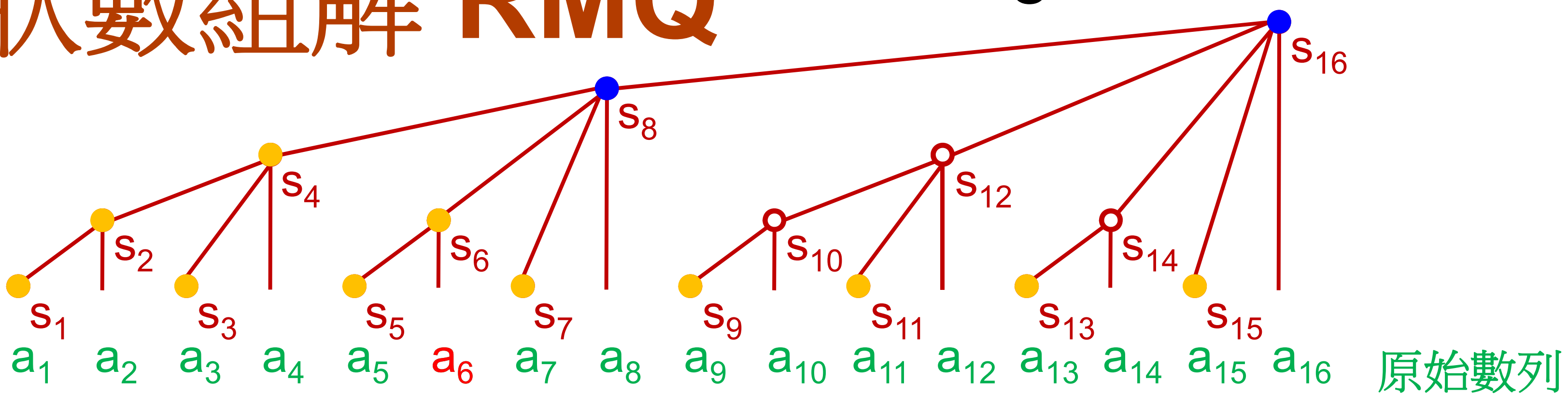
   $\cancel{s_8=\max(s_8,\mathbf{s_4})}$
   $\cancel{s_{16}=\max(s_{16},s_8)}$

   $s_6=\max(s_6,\mathbf{s_5})$
   $s_8=\max(s_8,s_6)$
   $s_{16}=\max(s_{16},s_8)$

   $s_8=\max(s_8,\mathbf{s_6})$
   $s_{16}=\max(s_{16},s_8)$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,s_1)$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

~~$s_4=\max(s_4,s_2)$~~
~~$s_8=\max(s_8,s_4)$~~
~~$s_{16}=\max(s_{16},s_8)$~~

$s_4=\max(s_4,s_3)$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

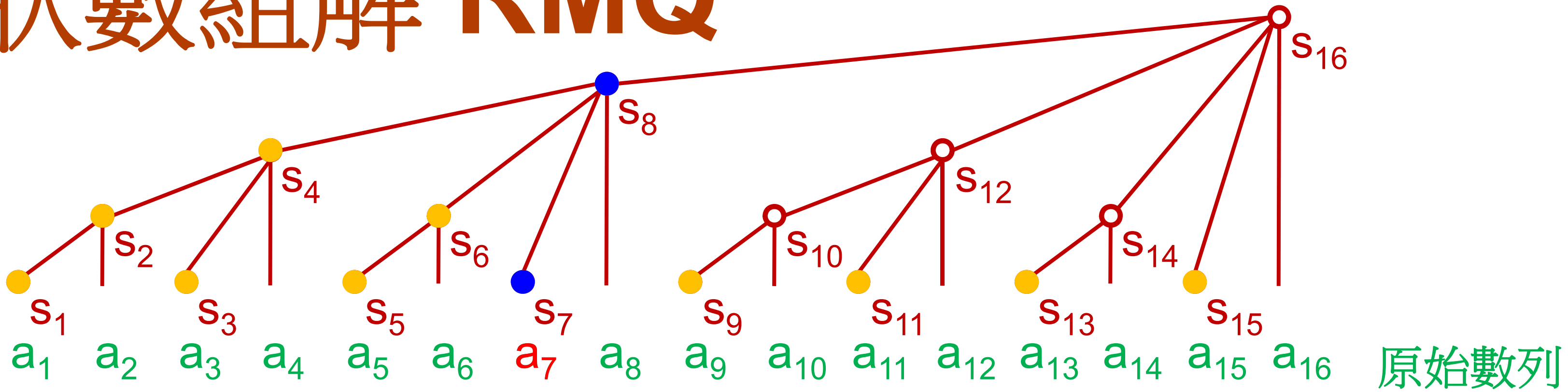
~~$s_8=\max(s_8,s_4)$~~
~~$s_{16}=\max(s_{16},s_8)$~~

$s_6=\max(s_6,s_5)$
$s_8=\max(s_8,s_6)$
$s_{16}=\max(s_{16},s_8)$

~~$s_8=\max(s_8,s_6)$~~
~~$s_{16}=\max(s_{16},s_8)$~~

# 樹狀數組解 RMQ

**Range Maximum Query**




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,\mathbf{s_1})$
$s_4=\max(s_4,s_2)$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

~~$s_4=\max(s_4,\mathbf{s_2})$~~
~~$s_8=\max(s_8,s_4)$~~
~~$s_{16}=\max(s_{16},s_8)$~~

$s_4=\max(s_4,\mathbf{s_3})$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

~~$s_8=\max(s_8,\mathbf{s_4})$~~
~~$s_{16}=\max(s_{16},s_8)$~~

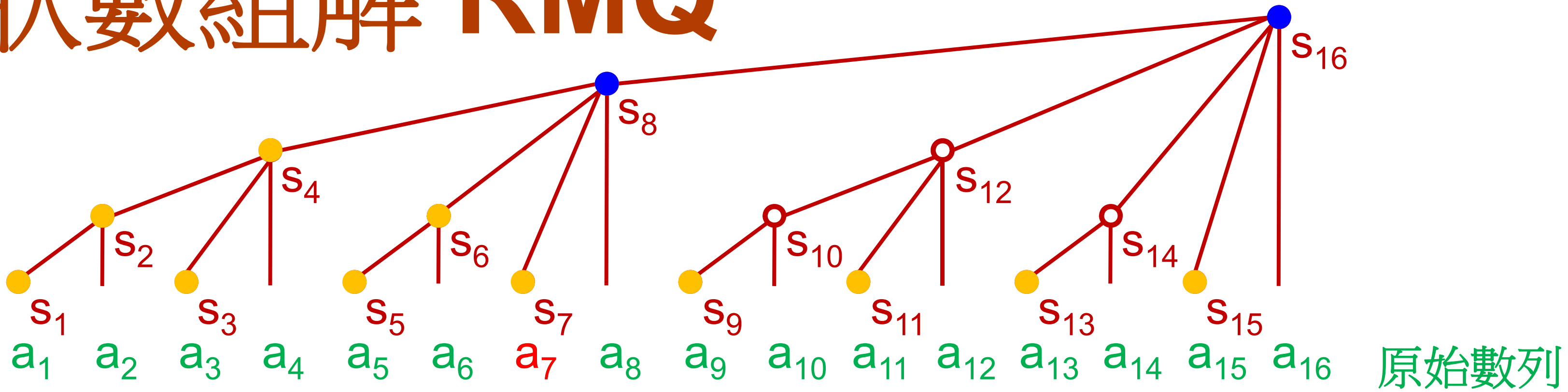
$s_6=\max(s_6,\mathbf{s_5})$
$s_8=\max(s_8,s_6)$
$s_{16}=\max(s_{16},s_8)$

~~$s_8=\max(s_8,\mathbf{s_6})$~~
~~$s_{16}=\max(s_{16},s_8)$~~

$s_8=\max(s_8,\mathbf{s_7})$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$
2. $s_i = a_i$, i=1,…,16
3. $s_2=\max(s_2,s_1)$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_4=\max(s_4,s_2)$~~
   ~~$s_8=\max(s_8,s_4)$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

$s_4=\max(s_4,s_3)$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

~~$s_8=\max(s_8,s_4)$~~
~~$s_{16}=\max(s_{16},s_8)$~~

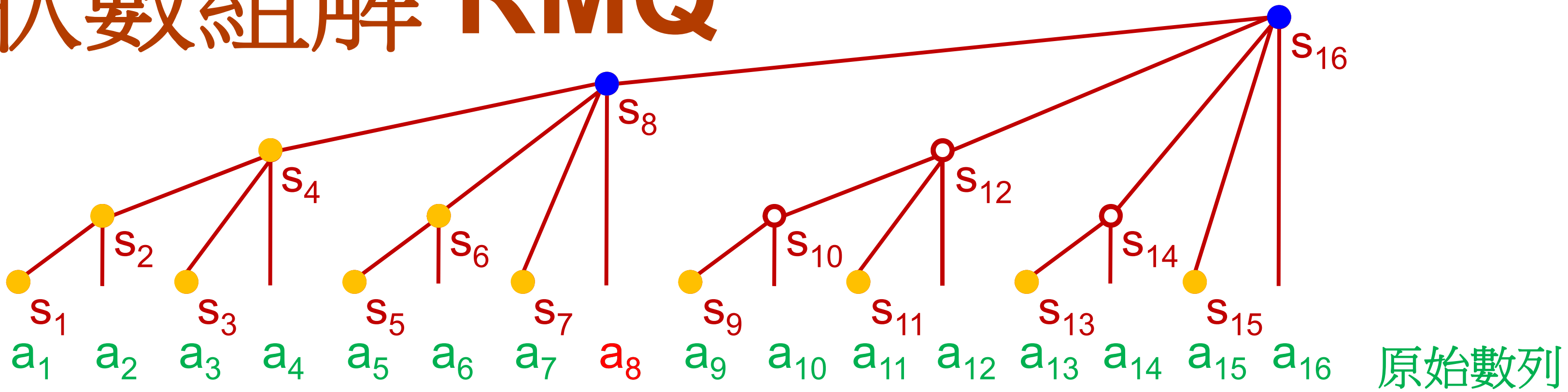
$s_6=\max(s_6,s_5)$
$s_8=\max(s_8,s_6)$
$s_{16}=\max(s_{16},s_8)$

~~$s_8=\max(s_8,s_6)$~~
~~$s_{16}=\max(s_{16},s_8)$~~

$s_8=\max(s_8,s_7)$
$s_{16}=\max(s_{16},s_8)$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$
2. $s_i = a_i$, $i=1,\ldots,16$
3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   $\cancel{s_4=\max(s_4,\mathbf{s_2})}$
   $\cancel{s_8=\max(s_8,s_4)}$
   $\cancel{s_{16}=\max(s_{16},s_8)}$

$s_4=\max(s_4,\mathbf{s_3})$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,\mathbf{s_4})}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_6=\max(s_6,\mathbf{s_5})$
$s_8=\max(s_8,s_6)$
$s_{16}=\max(s_{16},s_8)$

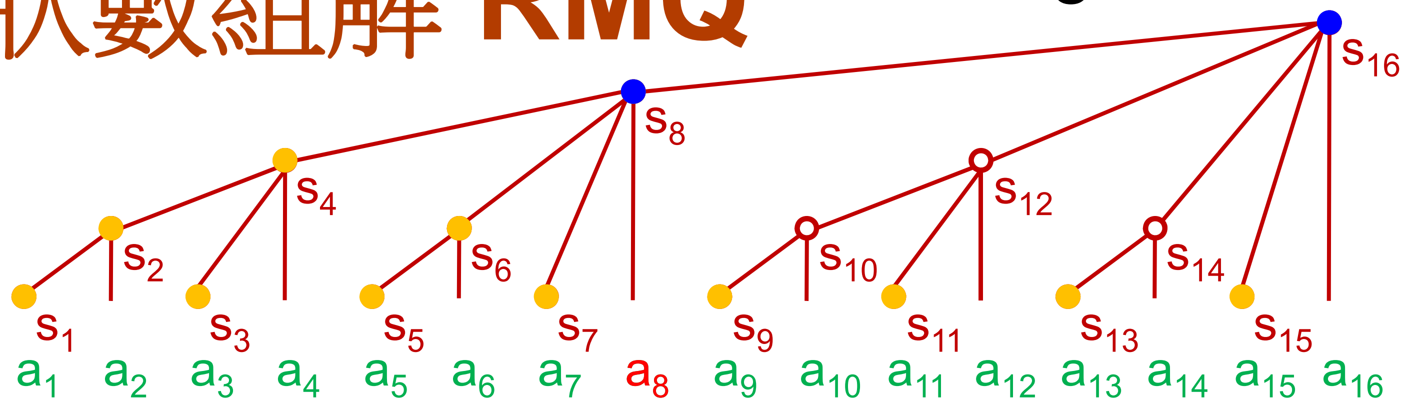
$\cancel{s_8=\max(s_8,\mathbf{s_6})}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_8=\max(s_8,\mathbf{s_7})$
$s_{16}=\max(s_{16},s_8)$

$s_{16}=\max(s_{16},\mathbf{s_8})$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,\mathbf{s_1})$
$s_4=\max(s_4,s_2)$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_4=\max(s_4,\mathbf{s_2})}$
$\cancel{s_8=\max(s_8,s_4)}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_4=\max(s_4,\mathbf{s_3})$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,\mathbf{s_4})}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_6=\max(s_6,\mathbf{s_5})$
$s_8=\max(s_8,s_6)$
$s_{16}=\max(s_{16},s_8)$

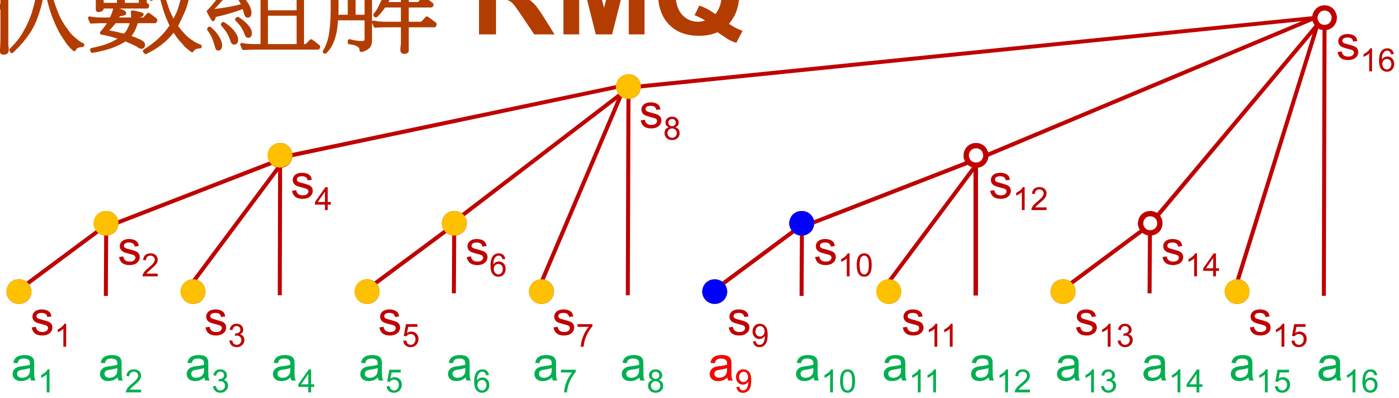
$\cancel{s_8=\max(s_8,\mathbf{s_6})}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_8=\max(s_8,\mathbf{s_7})$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_{16}=\max(s_{16},\mathbf{s_8})}$

# 樹狀數組解 RMQ

**Range Maximum Query**




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,s_1)$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_4=\max(s_4,s_2)$~~
   ~~$s_8=\max(s_8,s_4)$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

   $s_4=\max(s_4,s_3)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_8=\max(s_8,s_4)$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

   $s_6=\max(s_6,s_5)$
   $s_8=\max(s_8,s_6)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_8=\max(s_8,s_6)$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

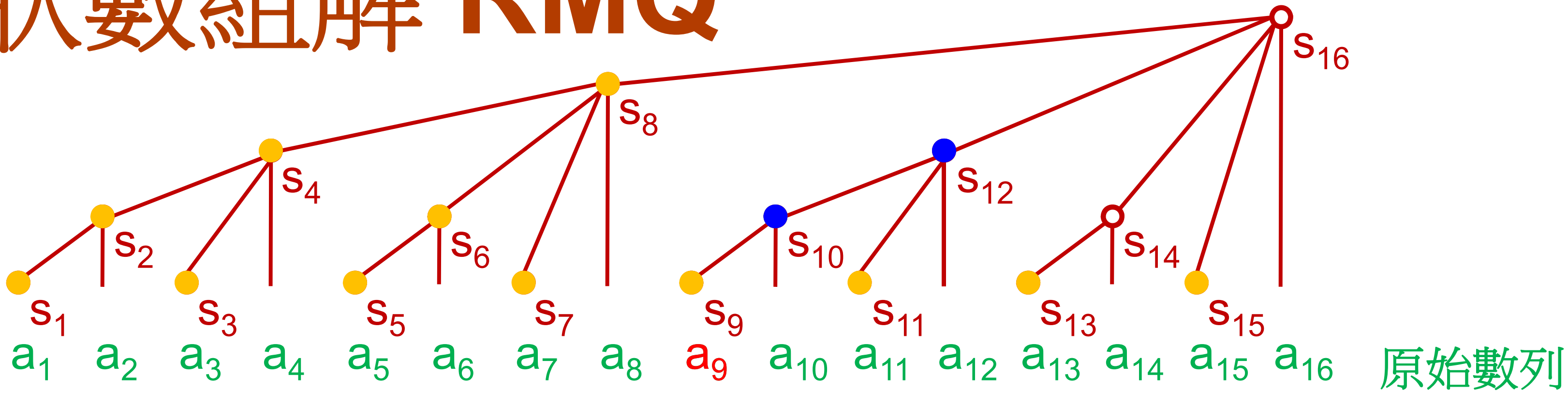
   $s_8=\max(s_8,s_7)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_{16}=\max(s_{16},s_8)$~~

   $s_{10}=\max(s_{10},s_9)$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \to \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   $\cancel{s_4=\max(s_4,\mathbf{s_2})}$
   $\cancel{s_8=\max(s_8,s_4)}$
   $\cancel{s_{16}=\max(s_{16},s_8)}$

   $s_4=\max(s_4,\mathbf{s_3})$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   $\cancel{s_8=\max(s_8,\mathbf{s_4})}$
   $\cancel{s_{16}=\max(s_{16},s_8)}$

   $s_6=\max(s_6,\mathbf{s_5})$
   $s_8=\max(s_8,s_6)$
   $s_{16}=\max(s_{16},s_8)$

   $\cancel{s_8=\max(s_8,\mathbf{s_6})}$
   $\cancel{s_{16}=\max(s_{16},s_8)}$

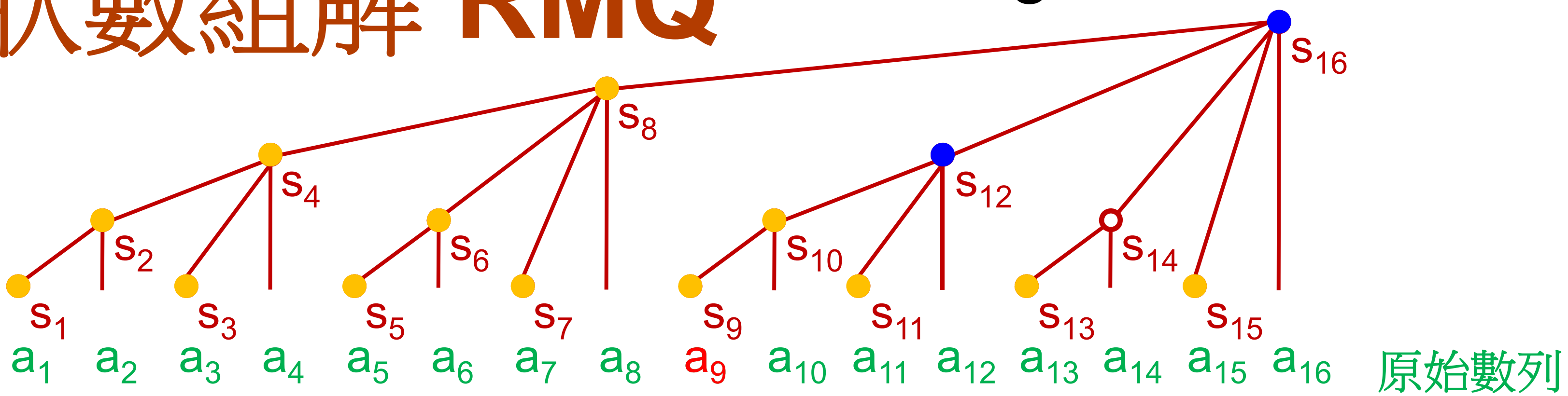
   $s_8=\max(s_8,\mathbf{s_7})$
   $s_{16}=\max(s_{16},s_8)$

   $\cancel{s_{16}=\max(s_{16},\mathbf{s_8})}$

   $s_{10}=\max(s_{10},\mathbf{s_9})$
   $s_{12}=\max(s_{12},s_{10})$

# 樹狀數組解 RMQ

**Range Maximum Query**




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_4=\max(s_4,\mathbf{s_2})$~~
   ~~$s_8=\max(s_8,s_4)$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

   $s_4=\max(s_4,\mathbf{s_3})$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_8=\max(s_8,\mathbf{s_4})$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

   $s_6=\max(s_6,\mathbf{s_5})$
   $s_8=\max(s_8,s_6)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_8=\max(s_8,\mathbf{s_6})$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

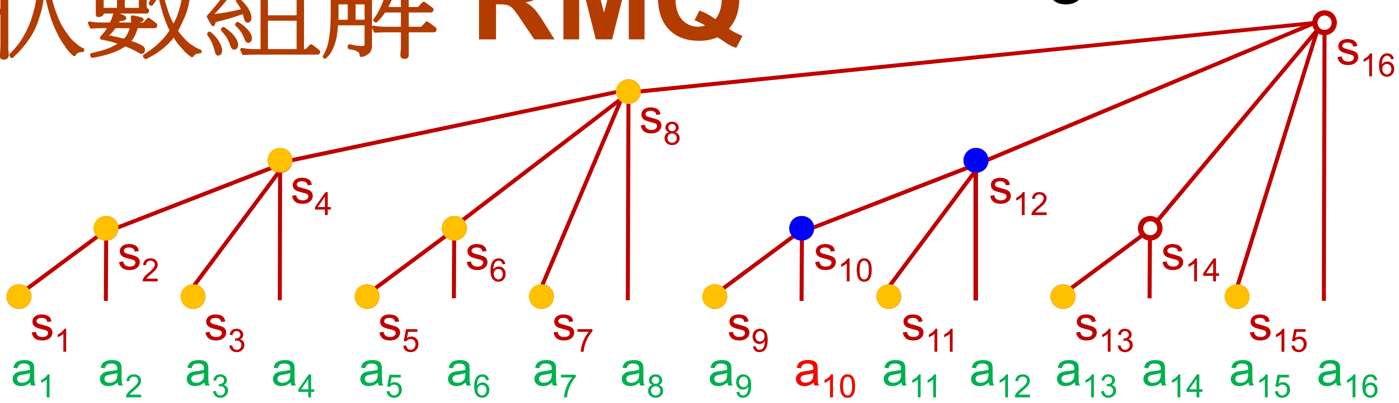
   $s_8=\max(s_8,\mathbf{s_7})$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_{16}=\max(s_{16},\mathbf{s_8})$~~

   $s_{10}=\max(s_{10},\mathbf{s_9})$
   $s_{12}=\max(s_{12},s_{10})$
   $s_{16}=\max(s_{16},s_{12})$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- 建構 **build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$

$s_{12}=\max(s_{12},\mathbf{s_{10}})$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,\mathbf{s_1})$
$s_4=\max(s_4,s_2)$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

~~$s_4=\max(s_4,\mathbf{s_2})$~~
~~$s_8=\max(s_8,s_4)$~~
~~$s_{16}=\max(s_{16},s_8)$~~

$s_4=\max(s_4,\mathbf{s_3})$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

~~$s_8=\max(s_8,\mathbf{s_4})$~~
~~$s_{16}=\max(s_{16},s_8)$~~

$s_6=\max(s_6,\mathbf{s_5})$
$s_8=\max(s_8,s_6)$
$s_{16}=\max(s_{16},s_8)$

~~$s_8=\max(s_8,\mathbf{s_6})$~~
~~$s_{16}=\max(s_{16},s_8)$~~

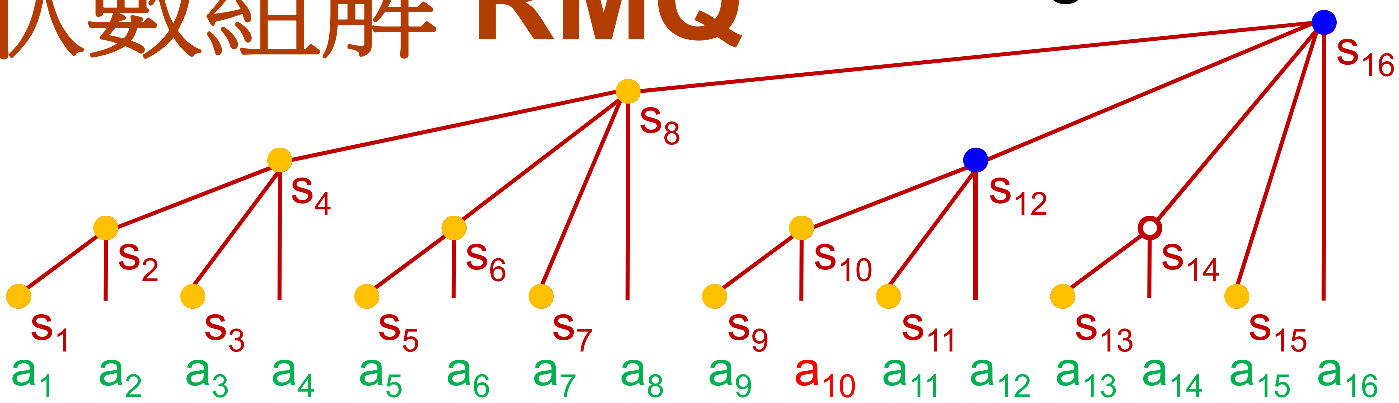
$s_8=\max(s_8,\mathbf{s_7})$
$s_{16}=\max(s_{16},s_8)$

~~$s_{16}=\max(s_{16},\mathbf{s_8})$~~

$s_{10}=\max(s_{10},\mathbf{s_9})$
$s_{12}=\max(s_{12},s_{10})$
$s_{16}=\max(s_{16},s_{12})$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




原始數列

- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$
2. $s_i = a_i$, i=1,…,16
3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   $\cancel{s_4=\max(s_4,\mathbf{s_2})}$
   $\cancel{s_8=\max(s_8,s_4)}$
   $\cancel{s_{16}=\max(s_{16},s_8)}$

$s_4=\max(s_4,\mathbf{s_3})$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,\mathbf{s_4})}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_6=\max(s_6,\mathbf{s_5})$
$s_8=\max(s_8,s_6)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,\mathbf{s_6})}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_8=\max(s_8,\mathbf{s_7})$
$s_{16}=\max(s_{16},s_8)$

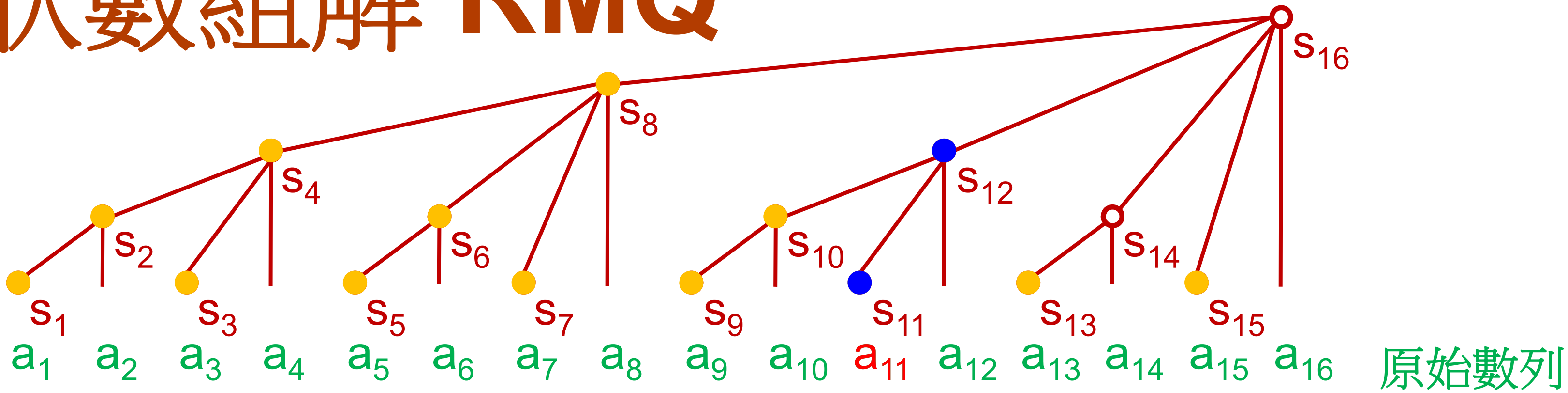
$\cancel{s_{16}=\max(s_{16},\mathbf{s_8})}$

$s_{10}=\max(s_{10},\mathbf{s_9})$
$s_{12}=\max(s_{12},s_{10})$
$s_{16}=\max(s_{16},s_{12})$

$\cancel{s_{12}=\max(s_{12},\mathbf{s_{10}})}$
$\cancel{s_{16}=\max(s_{16},s_{12})}$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- 建構 **build(), $\{a_i\} \to \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$
2. $s_i = a_i$, i=1,…,16
3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   $\cancel{s_4=\max(s_4,\mathbf{s_2})}$
   $\cancel{s_8=\max(s_8,s_4)}$
   $\cancel{s_{16}=\max(s_{16},s_8)}$

$s_4=\max(s_4,\mathbf{s_3})$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,\mathbf{s_4})}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_6=\max(s_6,\mathbf{s_5})$
$s_8=\max(s_8,s_6)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,\mathbf{s_6})}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_8=\max(s_8,\mathbf{s_7})$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_{16}=\max(s_{16},\mathbf{s_8})}$

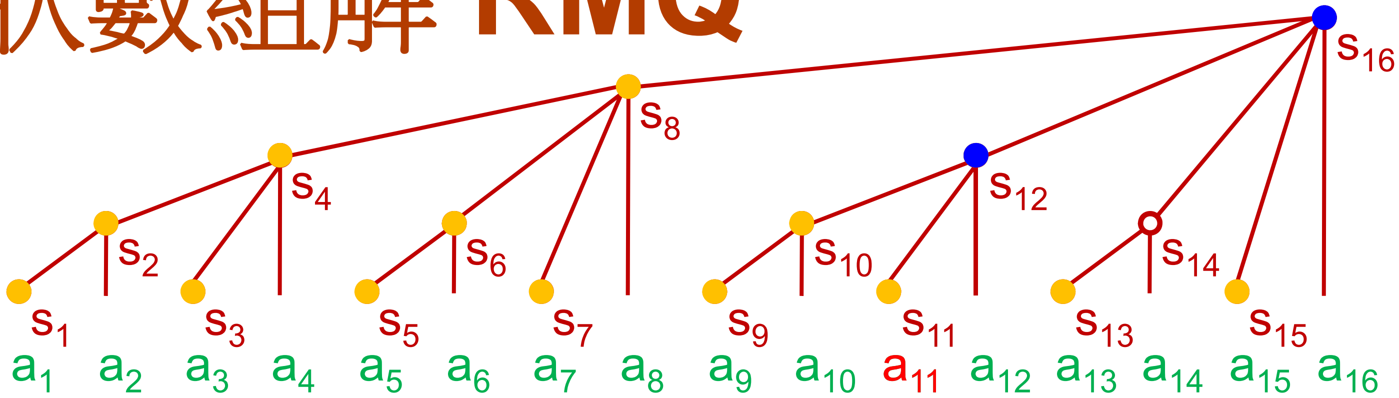
$s_{10}=\max(s_{10},\mathbf{s_9})$
$s_{12}=\max(s_{12},s_{10})$
$s_{16}=\max(s_{16},s_{12})$

$\cancel{s_{12}=\max(s_{12},\mathbf{s_{10}})}$
$\cancel{s_{16}=\max(s_{16},s_{12})}$

$s_{12}=\max(s_{12},\mathbf{s_{11}})$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




原始數列

- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,s_1)$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

$\cancel{s_4=\max(s_4,s_2)}$
$\cancel{s_8=\max(s_8,s_4)}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_4=\max(s_4,s_3)$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,s_4)}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_6=\max(s_6,s_5)$
$s_8=\max(s_8,s_6)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,s_6)}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_8=\max(s_8,s_7)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_{16}=\max(s_{16},s_8)}$

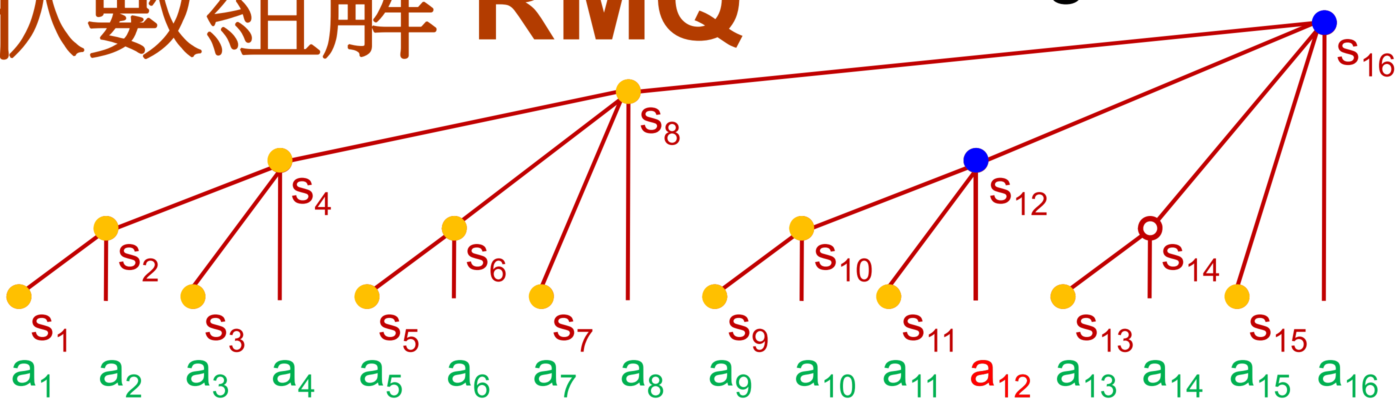
$s_{10}=\max(s_{10},s_9)$
$s_{12}=\max(s_{12},s_{10})$
$s_{16}=\max(s_{16},s_{12})$

$\cancel{s_{12}=\max(s_{12},s_{10})}$
$\cancel{s_{16}=\max(s_{16},s_{12})}$

$s_{12}=\max(s_{12},s_{11})$
$s_{16}=\max(s_{16},s_{12})$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_4=\max(s_4,\mathbf{s_2})$~~
   ~~$s_8=\max(s_8,s_4)$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

   $s_4=\max(s_4,\mathbf{s_3})$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_8=\max(s_8,\mathbf{s_4})$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

   $s_6=\max(s_6,\mathbf{s_5})$
   $s_8=\max(s_8,s_6)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_8=\max(s_8,\mathbf{s_6})$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

   $s_8=\max(s_8,\mathbf{s_7})$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_{16}=\max(s_{16},\mathbf{s_8})$~~

   $s_{10}=\max(s_{10},\mathbf{s_9})$
   $s_{12}=\max(s_{12},s_{10})$
   $s_{16}=\max(s_{16},s_{12})$

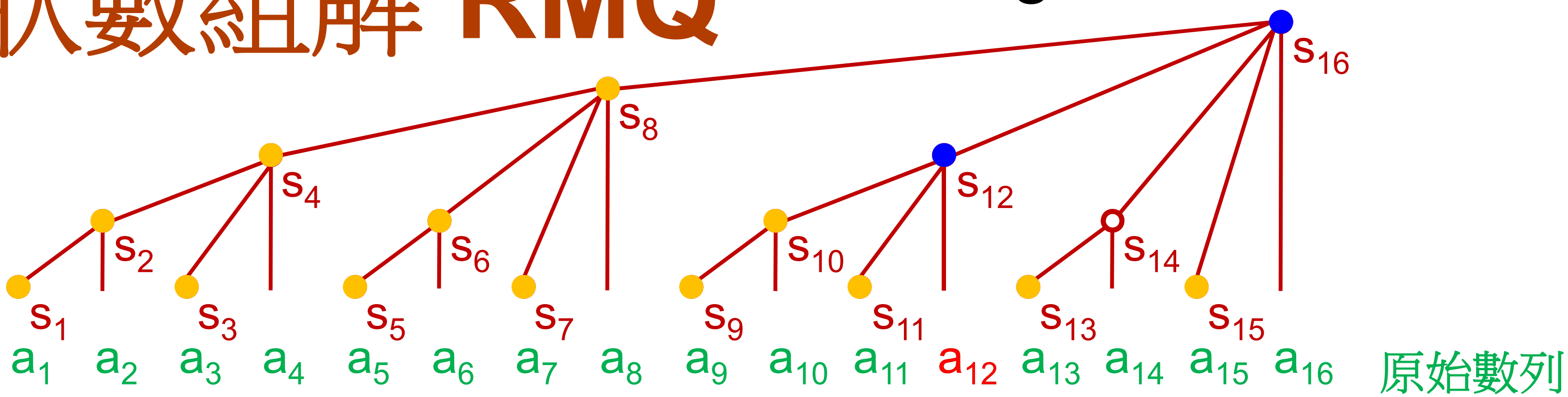
   ~~$s_{12}=\max(s_{12},\mathbf{s_{10}})$~~
   ~~$s_{16}=\max(s_{16},s_{12})$~~

   $s_{12}=\max(s_{12},\mathbf{s_{11}})$
   $s_{16}=\max(s_{16},s_{12})$

   $s_{16}=\max(s_{16},\mathbf{s_{12}})$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,s_1)$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   $\cancel{s_4=\max(s_4,s_2)}$
   $\cancel{s_8=\max(s_8,s_4)}$
   $\cancel{s_{16}=\max(s_{16},s_8)}$

$s_4=\max(s_4,s_3)$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,s_4)}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_6=\max(s_6,s_5)$
$s_8=\max(s_8,s_6)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,s_6)}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_8=\max(s_8,s_7)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_{10}=\max(s_{10},s_9)$
$s_{12}=\max(s_{12},s_{10})$
$s_{16}=\max(s_{16},s_{12})$

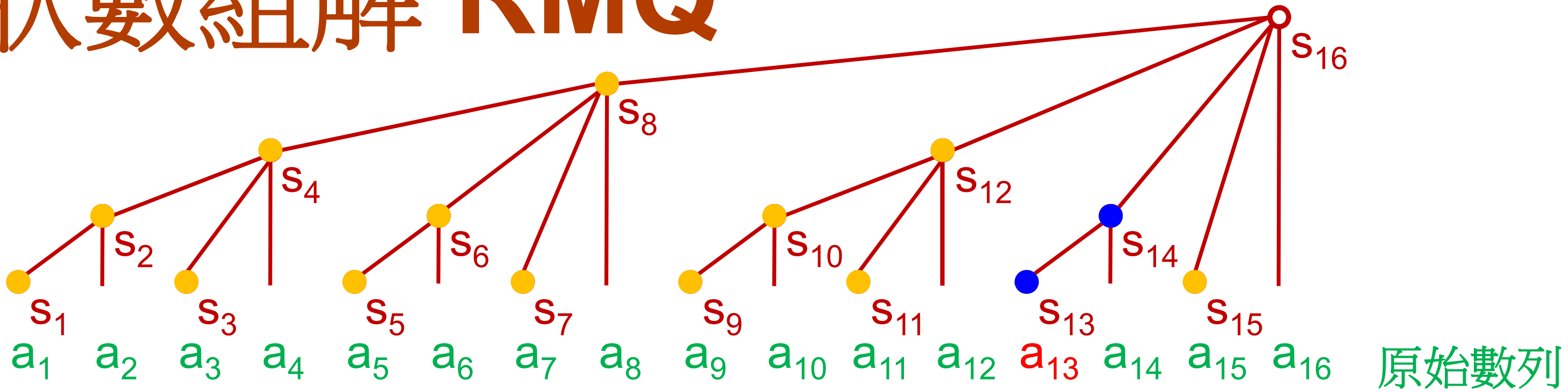
$\cancel{s_{12}=\max(s_{12},s_{10})}$
$\cancel{s_{16}=\max(s_{16},s_{12})}$

$s_{12}=\max(s_{12},s_{11})$
$s_{16}=\max(s_{16},s_{12})$

$\cancel{s_{16}=\max(s_{16},s_{12})}$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$
2. $s_i = a_i$, i=1,…,16
3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   $\cancel{s_4=\max(s_4,\mathbf{s_2})}$
   $\cancel{s_8=\max(s_8,s_4)}$
   $\cancel{s_{16}=\max(s_{16},s_8)}$

$s_4=\max(s_4,\mathbf{s_3})$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,\mathbf{s_4})}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_6=\max(s_6,\mathbf{s_5})$
$s_8=\max(s_8,s_6)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,\mathbf{s_6})}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_8=\max(s_8,\mathbf{s_7})$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_{16}=\max(s_{16},\mathbf{s_8})}$

$s_{10}=\max(s_{10},\mathbf{s_9})$
$s_{12}=\max(s_{12},s_{10})$
$s_{16}=\max(s_{16},s_{12})$

$\cancel{s_{12}=\max(s_{12},\mathbf{s_{10}})}$
$\cancel{s_{16}=\max(s_{16},s_{12})}$

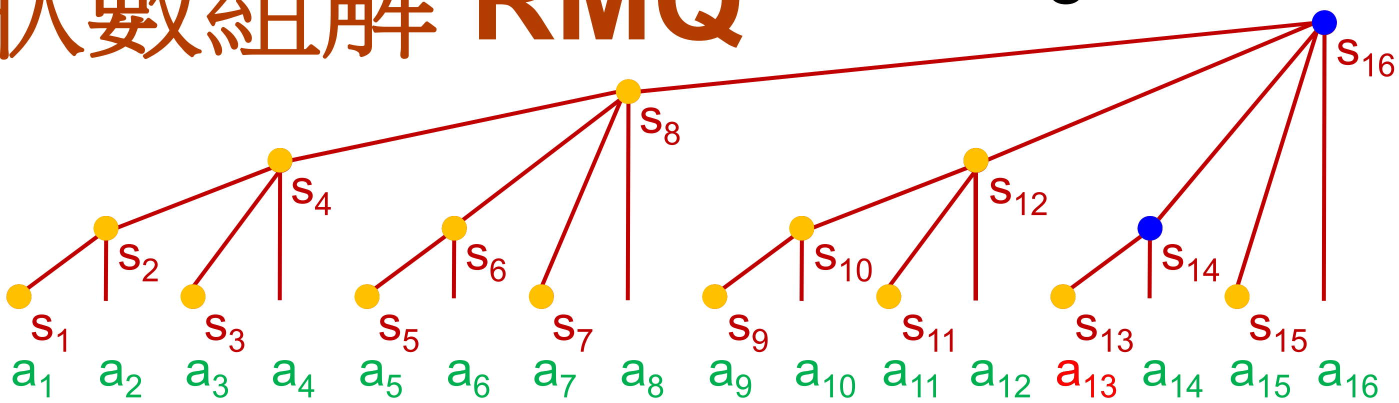
$s_{12}=\max(s_{12},\mathbf{s_{11}})$
$s_{16}=\max(s_{16},s_{12})$

$\cancel{s_{16}=\max(s_{16},\mathbf{s_{12}})}$

$s_{14}=\max(s_{14},\mathbf{s_{13}})$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




原始數列

- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_4=\max(s_4,\mathbf{s_2})$~~
   ~~$s_8=\max(s_8,s_4)$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

   $s_4=\max(s_4,\mathbf{s_3})$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_8=\max(s_8,\mathbf{s_4})$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

   $s_6=\max(s_6,\mathbf{s_5})$
   $s_8=\max(s_8,s_6)$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_8=\max(s_8,\mathbf{s_6})$~~
   ~~$s_{16}=\max(s_{16},s_8)$~~

   $s_8=\max(s_8,\mathbf{s_7})$
   $s_{16}=\max(s_{16},s_8)$

   ~~$s_{16}=\max(s_{16},\mathbf{s_8})$~~

   $s_{10}=\max(s_{10},\mathbf{s_9})$
   $s_{12}=\max(s_{12},s_{10})$
   $s_{16}=\max(s_{16},s_{12})$

   ~~$s_{12}=\max(s_{12},\mathbf{s_{10}})$~~
   ~~$s_{16}=\max(s_{16},s_{12})$~~

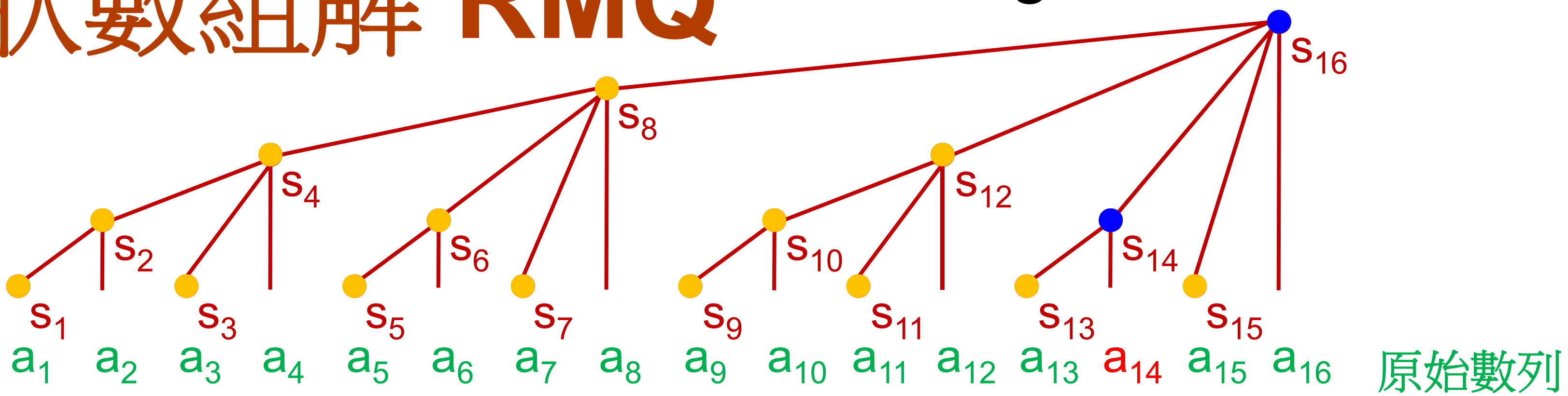
   $s_{12}=\max(s_{12},\mathbf{s_{11}})$
   $s_{16}=\max(s_{16},s_{12})$

   ~~$s_{16}=\max(s_{16},\mathbf{s_{12}})$~~

   $s_{14}=\max(s_{14},\mathbf{s_{13}})$
   $s_{16}=\max(s_{16},s_{14})$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$

2. $s_i = a_i$, i=1,…,16

3. $s_2=\max(s_2,s_1)$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   $\cancel{s_4=\max(s_4,s_2)}$
   $\cancel{s_8=\max(s_8,s_4)}$
   $\cancel{s_{16}=\max(s_{16},s_8)}$

$s_4=\max(s_4,s_3)$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,s_4)}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_6=\max(s_6,s_5)$
$s_8=\max(s_8,s_6)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,s_6)}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_8=\max(s_8,s_7)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_{10}=\max(s_{10},s_9)$
$s_{12}=\max(s_{12},s_{10})$
$s_{16}=\max(s_{16},s_{12})$

$\cancel{s_{12}=\max(s_{12},s_{10})}$
$\cancel{s_{16}=\max(s_{16},s_{12})}$

$s_{12}=\max(s_{12},s_{11})$
$s_{16}=\max(s_{16},s_{12})$

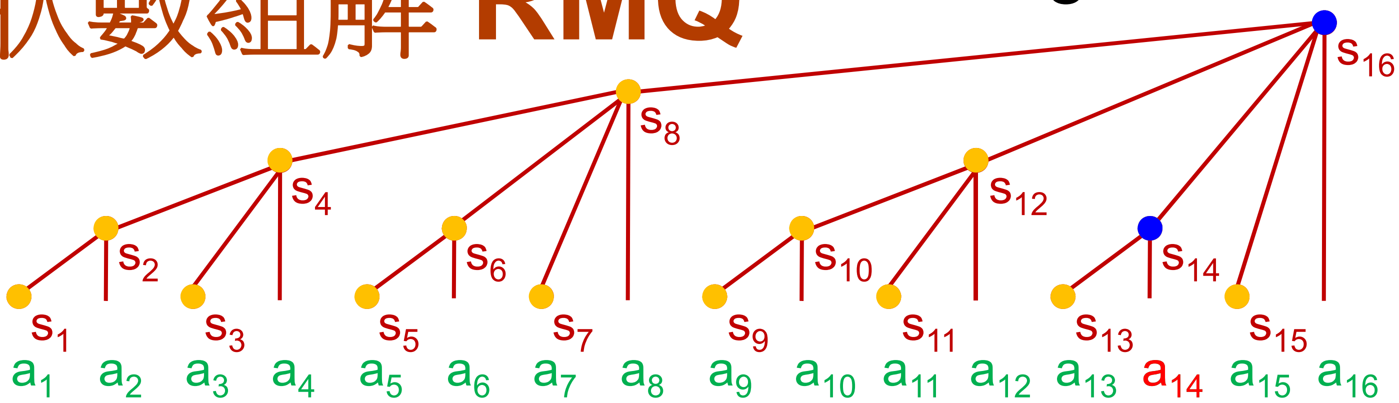
$\cancel{s_{16}=\max(s_{16},s_{12})}$

$s_{14}=\max(s_{14},s_{13})$
$s_{16}=\max(s_{16},s_{14})$

$s_{16}=\max(s_{16},s_{14})$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{a_{2k}\}$
2. $s_i = a_i$, i=1,…,16
3. $s_2=\max(s_2,s_1)$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   $\cancel{s_4=\max(s_4,s_2)}$
   $\cancel{s_8=\max(s_8,s_4)}$
   $\cancel{s_{16}=\max(s_{16},s_8)}$

$s_4=\max(s_4,s_3)$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,s_4)}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_6=\max(s_6,s_5)$
$s_8=\max(s_8,s_6)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,s_6)}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_8=\max(s_8,s_7)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_{10}=\max(s_{10},s_9)$
$s_{12}=\max(s_{12},s_{10})$
$s_{16}=\max(s_{16},s_{12})$

$\cancel{s_{12}=\max(s_{12},s_{10})}$
$\cancel{s_{16}=\max(s_{16},s_{12})}$

$s_{12}=\max(s_{12},s_{11})$
$s_{16}=\max(s_{16},s_{12})$

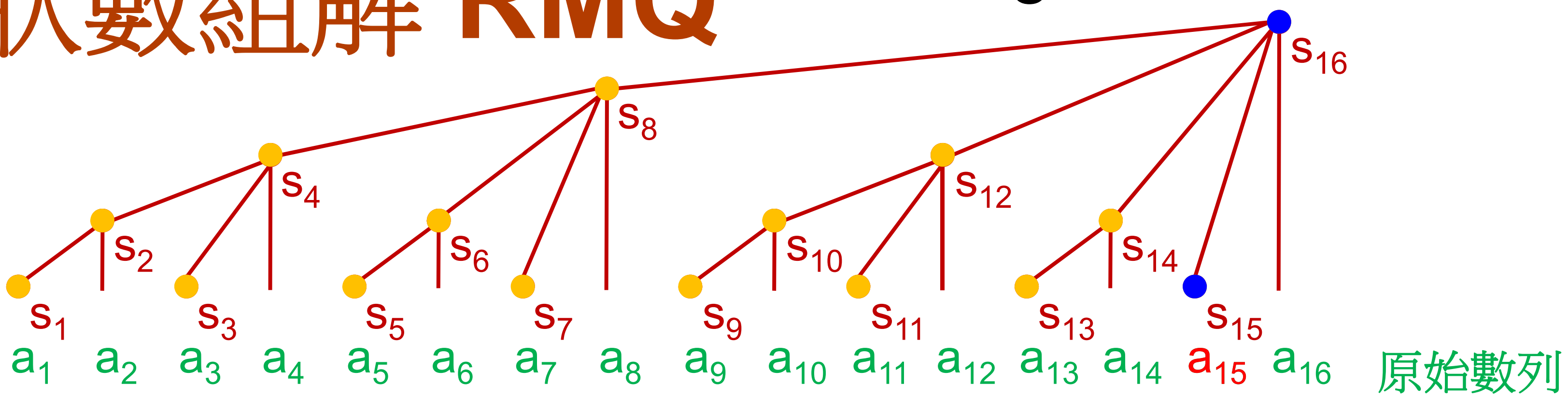
$\cancel{s_{16}=\max(s_{16},s_{12})}$

$s_{14}=\max(s_{14},s_{13})$
$s_{16}=\max(s_{16},s_{14})$

$\cancel{s_{16}=\max(s_{16},s_{14})}$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- 建構 **build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$
2. $s_i = a_i$, i=1,…,16
3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   $\cancel{s_4=\max(s_4,\mathbf{s_2})}$
   $\cancel{s_8=\max(s_8,s_4)}$
   $\cancel{s_{16}=\max(s_{16},s_8)}$

$s_4=\max(s_4,\mathbf{s_3})$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,\mathbf{s_4})}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_6=\max(s_6,\mathbf{s_5})$
$s_8=\max(s_8,s_6)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,\mathbf{s_6})}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_8=\max(s_8,\mathbf{s_7})$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_{16}=\max(s_{16},\mathbf{s_8})}$

$s_{10}=\max(s_{10},\mathbf{s_9})$
$s_{12}=\max(s_{12},s_{10})$
$s_{16}=\max(s_{16},s_{12})$

$\cancel{s_{12}=\max(s_{12},\mathbf{s_{10}})}$
$\cancel{s_{16}=\max(s_{16},s_{12})}$

$s_{12}=\max(s_{12},\mathbf{s_{11}})$
$s_{16}=\max(s_{16},s_{12})$

$\cancel{s_{16}=\max(s_{16},\mathbf{s_{12}})}$

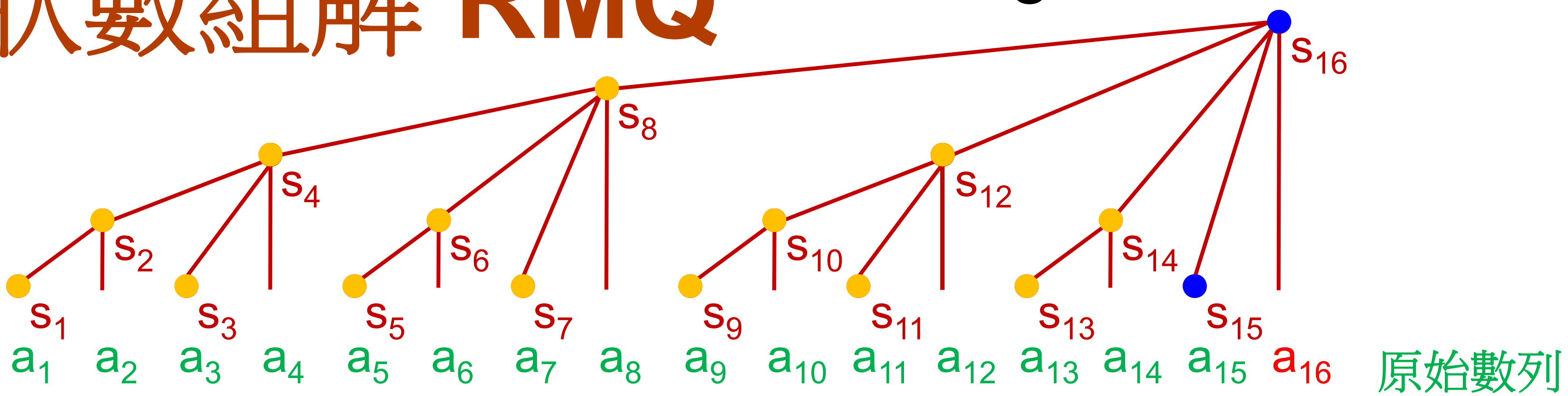
$s_{14}=\max(s_{14},\mathbf{s_{13}})$
$s_{16}=\max(s_{16},s_{14})$

$\cancel{s_{16}=\max(s_{16},\mathbf{s_{14}})}$

$s_{16}=\max(s_{16},\mathbf{s_{15}})$

# 樹狀數組解 RMQ

**R**ange **M**aximum **Q**uery




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\dots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$
2. $s_i = a_i$, i=1,…,16
3. $s_2=\max(s_2,\mathbf{s_1})$
   $s_4=\max(s_4,s_2)$
   $s_8=\max(s_8,s_4)$
   $s_{16}=\max(s_{16},s_8)$

   $\cancel{s_4=\max(s_4,\mathbf{s_2})}$
   $\cancel{s_8=\max(s_8,s_4)}$
   $\cancel{s_{16}=\max(s_{16},s_8)}$

$s_4=\max(s_4,\mathbf{s_3})$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,\mathbf{s_4})}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_6=\max(s_6,\mathbf{s_5})$
$s_8=\max(s_8,s_6)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,\mathbf{s_6})}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_8=\max(s_8,\mathbf{s_7})$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_{16}=\max(s_{16},\mathbf{s_8})}$

$s_{10}=\max(s_{10},\mathbf{s_9})$
$s_{12}=\max(s_{12},s_{10})$
$s_{16}=\max(s_{16},s_{12})$

$\cancel{s_{12}=\max(s_{12},\mathbf{s_{10}})}$
$\cancel{s_{16}=\max(s_{16},s_{12})}$

$s_{12}=\max(s_{12},\mathbf{s_{11}})$
$s_{16}=\max(s_{16},s_{12})$

$\cancel{s_{16}=\max(s_{16},\mathbf{s_{12}})}$

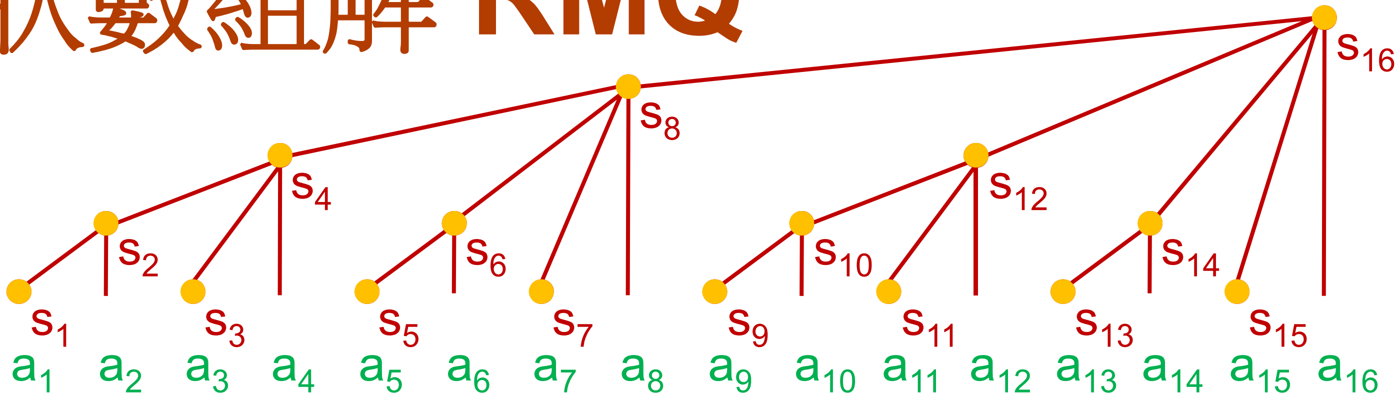
$s_{14}=\max(s_{14},\mathbf{s_{13}})$
$s_{16}=\max(s_{16},s_{14})$

$\cancel{s_{16}=\max(s_{16},\mathbf{s_{14}})}$

$s_{16}=\max(s_{16},\mathbf{s_{15}})$

# 樹狀數組解 RMQ

**Range Maximum Query**




- **建構 build(), $\{a_i\} \rightarrow \{s_i\}$**, $s_i$ ($s_8$) 是子樹涵蓋數列 $\{a_1,\ldots, a_8\}$ 的最大值

1. 原始資料陣列需要保留起來, 查詢及修改時需要 $\{\mathbf{a_{2k}}\}$
2. $s_i = a_i$, i=1,…,16
3. $s_2=\max(s_2,\mathbf{s_1})$
$s_4=\max(s_4,s_2)$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_4=\max(s_4,\mathbf{s_2})}$
$\cancel{s_8=\max(s_8,s_4)}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_4=\max(s_4,\mathbf{s_3})$
$s_8=\max(s_8,s_4)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,\mathbf{s_4})}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_6=\max(s_6,\mathbf{s_5})$
$s_8=\max(s_8,s_6)$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_8=\max(s_8,\mathbf{s_6})}$
$\cancel{s_{16}=\max(s_{16},s_8)}$

$s_8=\max(s_8,\mathbf{s_7})$
$s_{16}=\max(s_{16},s_8)$

$\cancel{s_{16}=\max(s_{16},\mathbf{s_8})}$

$s_{10}=\max(s_{10},\mathbf{s_9})$
$s_{12}=\max(s_{12},s_{10})$
$s_{16}=\max(s_{16},s_{12})$

$\cancel{s_{12}=\max(s_{12},\mathbf{s_{10}})}$
$\cancel{s_{16}=\max(s_{16},s_{12})}$

$s_{12}=\max(s_{12},\mathbf{s_{11}})$
$s_{16}=\max(s_{16},s_{12})$

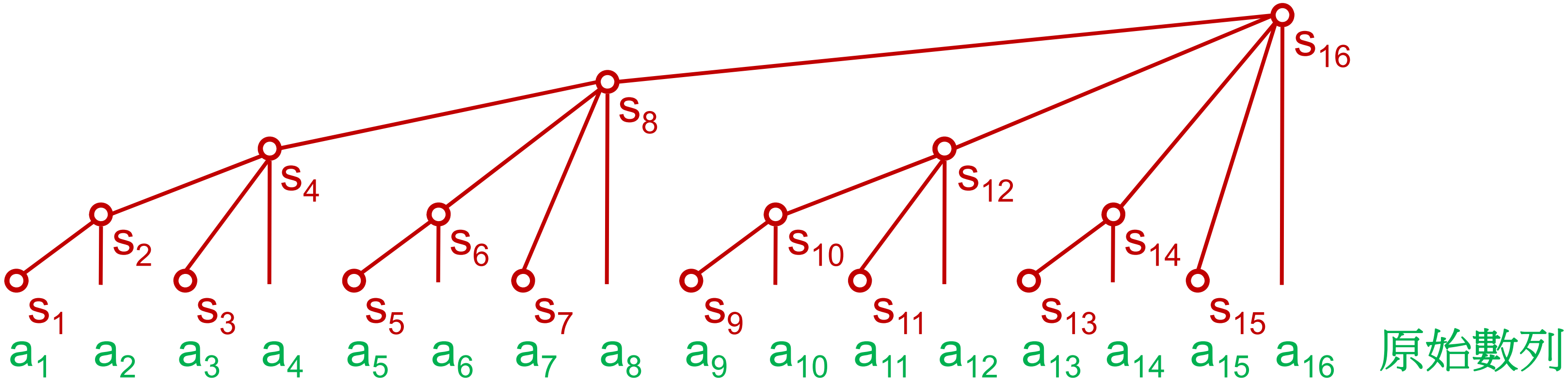
$\cancel{s_{16}=\max(s_{16},\mathbf{s_{12}})}$

$s_{14}=\max(s_{14},\mathbf{s_{13}})$
$s_{16}=\max(s_{16},s_{14})$

$\cancel{s_{16}=\max(s_{16},\mathbf{s_{14}})}$

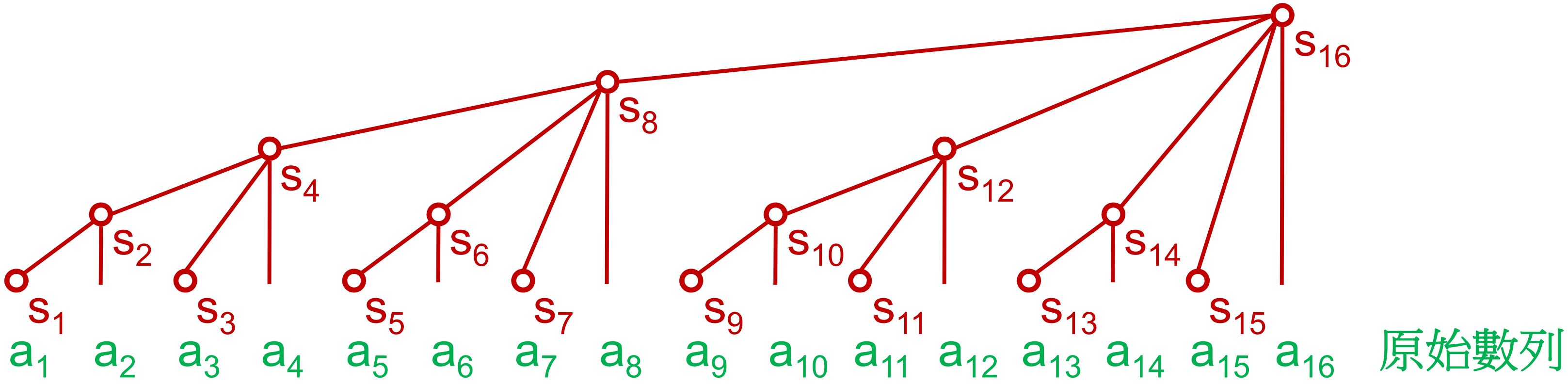
$s_{16}=\max(s_{16},\mathbf{s_{15}})$

# 樹狀數組解 RMQ (續)



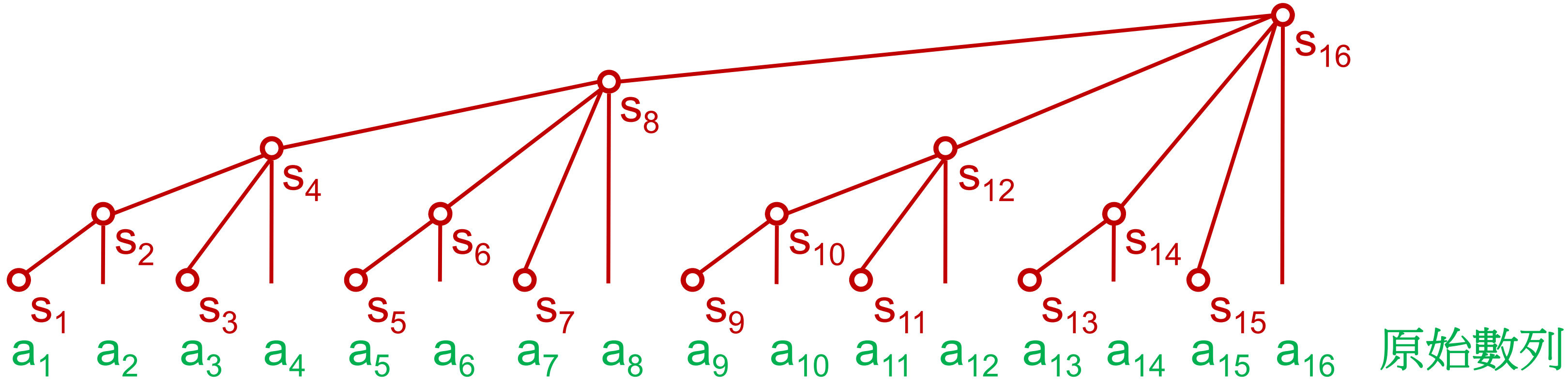

- 修改 **update(), $a_i \rightarrow A_i$**

# 樹狀數組解 RMQ (續)



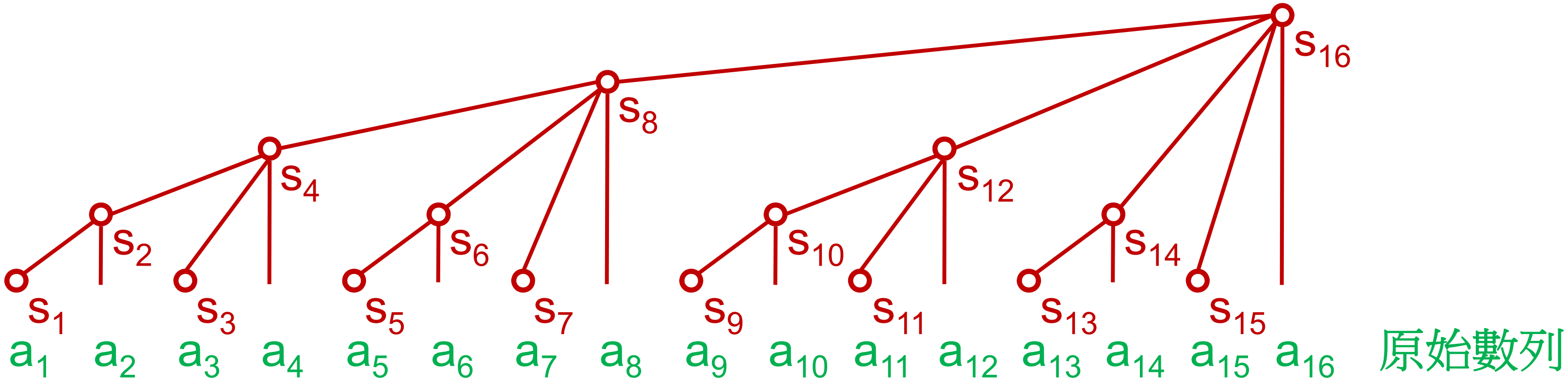

- 修改 **update(), $a_i \rightarrow A_i$**
  - 只有 $s_i$ 及 $s_i$ 的父節點 $\{s_j\}$ 需要修改

# 樹狀數組解 RMQ (續)



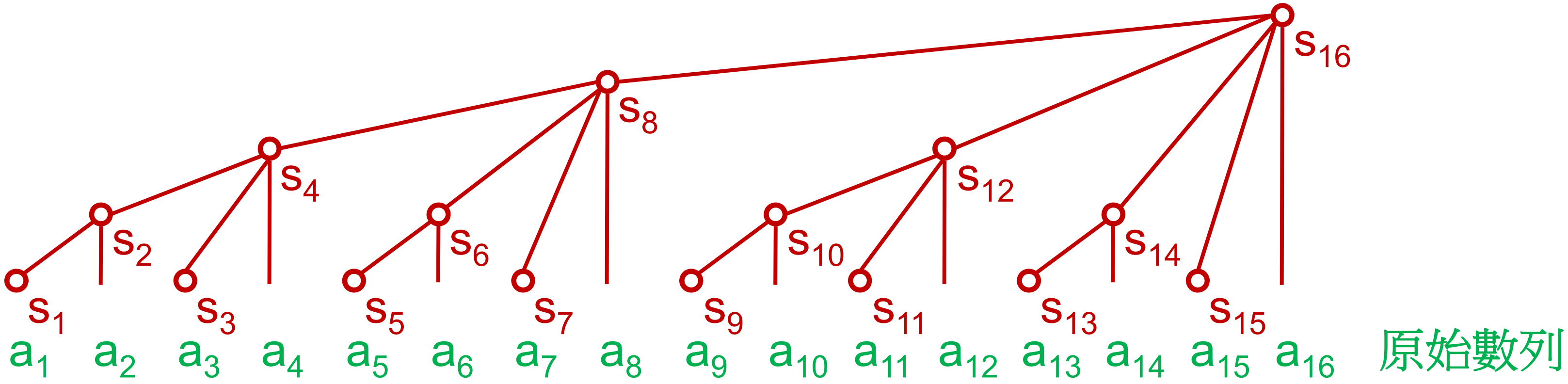

- 修改 **update(), $a_i \rightarrow A_i$**
  - 只有 $s_i$ 及 $s_i$ 的父節點 $\{s_j\}$ 需要修改
  - 變大 $A_i > a_i$

# 樹狀數組解 RMQ (續)



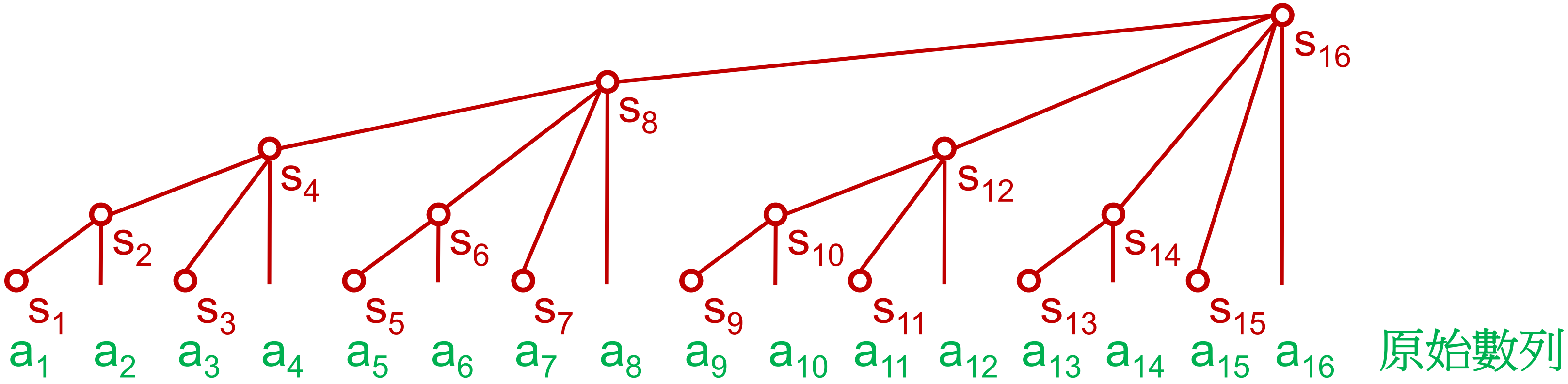

- 修改 update(), $a_i \rightarrow A_i$
  - 只有 $s_i$ 及 $s_i$ 的父節點 $\{s_j\}$ 需要修改
  - 變大 $A_i > a_i$
  - 變小 $A_i < a_i$

# 樹狀數組解 RMQ (續)



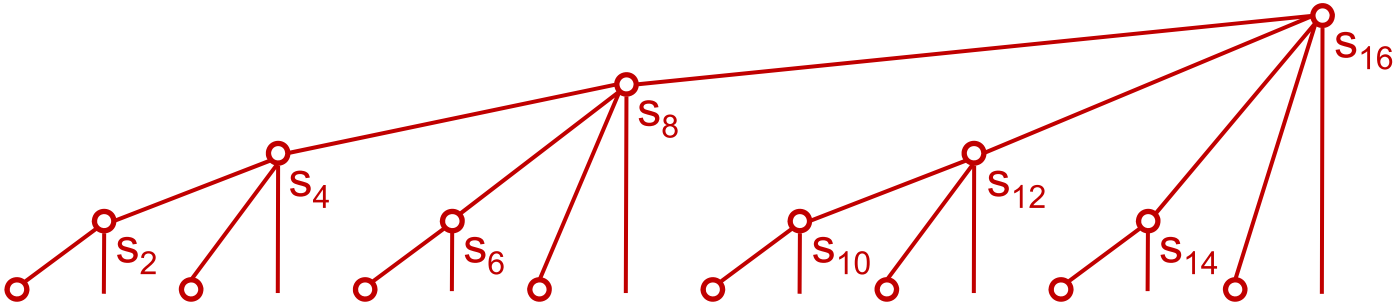

- 修改 **update(), $a_i \rightarrow A_i$**
  - 只有 $s_i$ 及 $s_i$ 的父節點 $\{s_j\}$ 需要修改
  - 變大 $A_i > a_i$
    - ① $A_i >= s_j$ : $s_j = A_i$
  - 變小 $A_i < a_i$

# 樹狀數組解 RMQ (續)



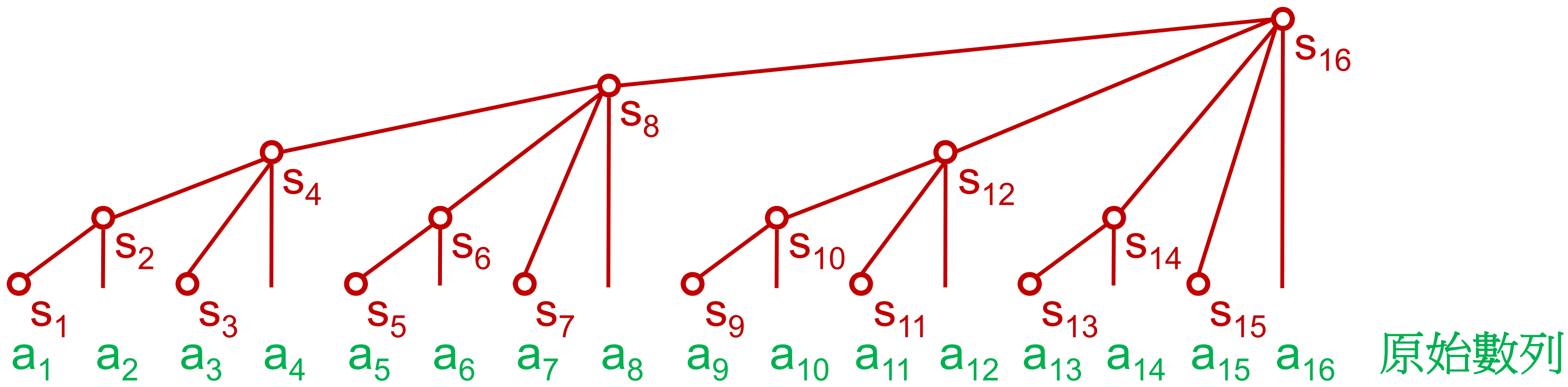

- 修改 **update(), $a_i \rightarrow A_i$**
  - 只有 $s_i$ 及 $s_i$ 的父節點 $\{s_j\}$ 需要修改
  - 變大 $A_i > a_i$
    - ① $A_i >= s_j$ : $s_j = A_i$
    - ② $A_i < s_j$ : 不影響 $s_j$, 也不影響更上層的父節點 $\{s_{j'}\}$
  - 變小 $A_i < a_i$

# 樹狀數組解 RMQ (續)




- 修改 **update(), $a_i \rightarrow A_i$**
  - 只有 $s_i$ 及 $s_i$ 的父節點 $\{s_j\}$ 需要修改
  - 變大 $A_i > a_i$
    - ① $A_i >= s_j$ : $s_j = A_i$
    - ② $A_i < s_j$ : 不影響 $s_j$, 也不影響更上層的父節點 $\{s_{j'}\}$
  - 變小 $A_i < a_i$
    - ③ $a_i < s_j$ : 不影響 $s_j$, 也不影響更上層的父節點 $\{s_{j'}\}$

# 樹狀數組解 RMQ (續)




- 修改 **update(), $a_i \rightarrow A_i$**
  - 只有 $s_i$ 及 $s_i$ 的父節點 $\{s_j\}$ 需要修改
  - 變大 $A_i > a_i$
    - ① $A_i >= s_j$ : $s_j = A_i$
    - ② $A_i < s_j$ : 不影響 $s_j$, 也不影響更上層的父節點 $\{s_{j'}\}$
  - 變小 $A_i < a_i$
    - ③ $a_i < s_j$ : 不影響 $s_j$, 也不影響更上層的父節點 $\{s_{j'}\}$
    - ④ $a_i == s_j$ : 重新搜尋整個子樹

# 樹狀數組解 RMQ (續)

$a_i \to A_i$, 修改所有 $a_i$ 的父節點 $s_j$

```
for (oldA=a[i],a[i]=Ai,j=i; j<=N; j+=lb(j)) {

}
```

# 樹狀數組解 RMQ (續)

$a_i \rightarrow A_i$, 修改所有 $a_i$ 的父節點 $s_j$

```
for (oldA=a[i],a[i]=Ai,j=i; j<=N; j+=lb(j)) {
    if (s[j]>oldA&&s[j]>Ai) break;
```

② $A_i > a_i$ $A_i < s_j$

③ $A_i < a_i$ $a_i < s_j$

```
}
```

# 樹狀數組解 RMQ (續)

$a_i \to A_i$, 修改所有 $a_i$ 的父節點 $s_j$

```
for (oldA=a[i],a[i]=Ai,j=i; j<=N; j+=lb(j)) {
    if (s[j]>oldA&&s[j]>Ai) break;
    if (Ai>s[j]&&Ai>oldA) s[j]=Ai;
}
```

① $A_i > a_i$  $A_i >= s_j$

② $A_i > a_i$  $A_i < s_j$

③ $A_i < a_i$  $a_i < s_j$

# 樹狀數組解 RMQ (續)

$a_i \to A_i$, 修改所有 $a_i$ 的父節點 $s_j$

```
for (oldA=a[i],a[i]=Ai,j=i; j<=N; j+=lb(j)) {
    if (s[j]>oldA&&s[j]>Ai) break;
    if (Ai>s[j]&&Ai>oldA) s[j]=Ai;
    else {
        b = j-lb(j);
        for (max=a[j],k=j-1; k>b; k-=lb(k))
            if (s[k]>max) max=s[k];
        s[j]=max;
    }
}
```

① $A_i > a_i$ $A_i >= s_j$

② $A_i > a_i$ $A_i < s_j$

③ $A_i < a_i$ $a_i < s_j$

④ $A_i < a_i$ $a_i == s_j$

重新搜尋整個子樹

# 樹狀數組解 RMQ (續)

$a_i \rightarrow A_i$, 修改所有 $a_i$ 的父節點 $s_j$

```
for (oldA=a[i],a[i]=Ai,j=i; j<=N; j+=lb(j)) {
    if (s[j]>oldA&&s[j]>Ai) break;
    if (Ai>s[j]&&Ai>oldA) s[j]=Ai;
    else {
        b = j-lb(j);
        for (max=a[j],k=j-1; k>b; k-=lb(k))
            if (s[k]>max) max=s[k];
        s[j]=max;
    }
}
```

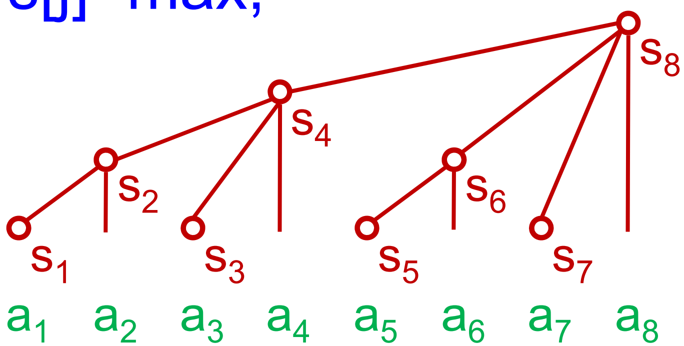
① $A_i > a_i \quad A_i >= s_j$

② $A_i > a_i \quad A_i < s_j$

③ $A_i < a_i \quad a_i < s_j$

④ $A_i < a_i \quad a_i == s_j$

重新搜尋整個子樹



# 樹狀數組解 RMQ (續)

$a_i \rightarrow A_i$, 修改所有 $a_i$ 的父節點 $s_j$

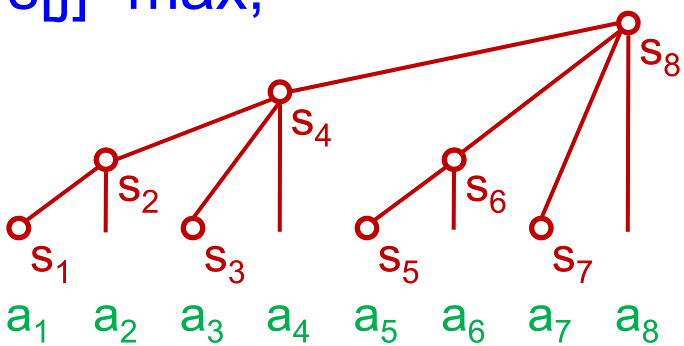
```
for (oldA=a[i],a[i]=Ai,j=i; j<=N; j+=lb(j)) {
    if (s[j]>oldA&&s[j]>Ai) break;
    if (Ai>s[j]&&Ai>oldA) s[j]=Ai;
    else {
        b = j-lb(j);
        for (max=a[j],k=j-1; k>b; k-=lb(k))
            if (s[k]>max) max=s[k];
        s[j]=max;
    }
}
```

① $A_i > a_i$ $A_i >= s_j$

② $A_i > a_i$ $A_i < s_j$

③ $A_i < a_i$ $a_i < s_j$

④ $A_i < a_i$ $a_i == s_j$




重新搜尋整個子樹

子節點 $s_1, s_2, \ldots, s_7$ 必然已經更新, 計算 $s_8$

# 樹狀數組解 RMQ (續)

$a_i \rightarrow A_i$, 修改所有 $a_i$ 的父節點 $s_j$

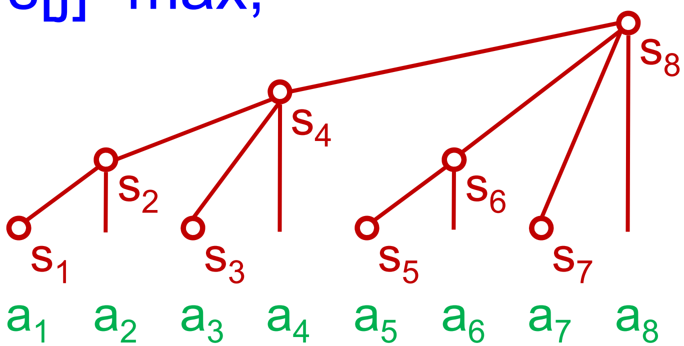
```
for (oldA=a[i],a[i]=Ai,j=i; j<=N; j+=lb(j)) {
    if (s[j]>oldA&&s[j]>Ai) break;
    if (Ai>s[j]&&Ai>oldA) s[j]=Ai;
    else {
        b = j-lb(j);
        for (max=a[j],k=j-1; k>b; k-=lb(k))
            if (s[k]>max) max=s[k];
        s[j]=max;
    }
}
```

① $A_i > a_i \quad A_i >= s_j$

② $A_i > a_i \quad A_i < s_j$

③ $A_i < a_i \quad a_i < s_j$

④ $A_i < a_i \quad a_i == s_j$




重新搜尋整個子樹

子節點 $s_1, s_2, \ldots, s_7$ 必然已經更新, 計算 $s_8$

$s_8 = \max(a_8, s_7, s_6, s_4)$

# 樹狀數組解 RMQ (續)

$a_i \rightarrow A_i$, 修改所有 $a_i$ 的父節點 $s_j$

```
for (oldA=a[i],a[i]=Ai,j=i; j<=N; j+=lb(j)) {
    if (s[j]>oldA&&s[j]>Ai) break;
    if (Ai>s[j]&&Ai>oldA) s[j]=Ai;
    else {
        b = j-lb(j);
        for (max=a[j],k=j-1; k>b; k-=lb(k))
            if (s[k]>max) max=s[k];
        s[j]=max;
    }
}
```

① $A_i > a_i \quad A_i >= s_j$

② $A_i > a_i \quad A_i < s_j$

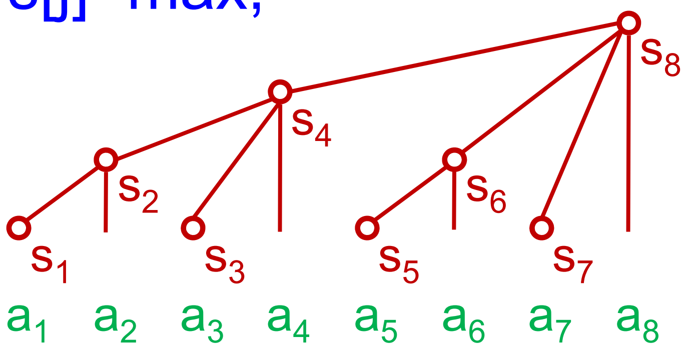
③ $A_i < a_i \quad a_i < s_j$

④ $A_i < a_i \quad a_i == s_j$

重新搜尋整個子樹

子節點 $s_1, s_2, \ldots, s_7$ 必然已經更新, 計算 $s_8$

$s_8 = \max(a_8, s_7, s_6, s_4)$




e.g. $a_2 \rightarrow \mathbf{A_2}$, $\mathbf{A_2} < a_2$, $s_8 == s_4 == s_2 == a_2$

# 樹狀數組解 RMQ (續)

$a_i \rightarrow A_i$, 修改所有 $a_i$ 的父節點 $s_j$

```
for (oldA=a[i],a[i]=Ai,j=i; j<=N; j+=lb(j)) {
    if (s[j]>oldA&&s[j]>Ai) break;
    if (Ai>s[j]&&Ai>oldA) s[j]=Ai;
    else {
        b = j-lb(j);
        for (max=a[j],k=j-1; k>b; k-=lb(k))
            if (s[k]>max) max=s[k];
        s[j]=max;
    }
}
```

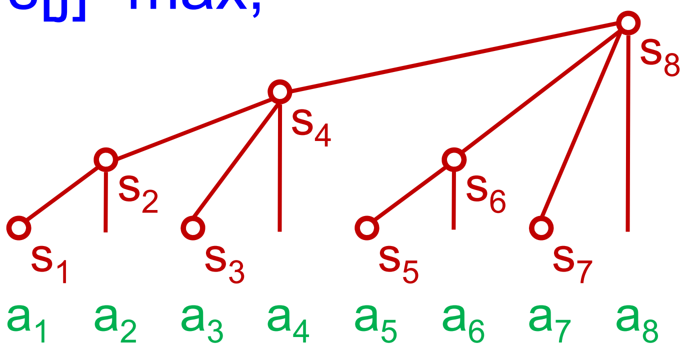
① $A_i > a_i \quad A_i >= s_j$

② $A_i > a_i \quad A_i < s_j$

③ $A_i < a_i \quad a_i < s_j$

④ $A_i < a_i \quad a_i == s_j$

重新搜尋整個子樹




子節點 $s_1, s_2, \ldots, s_7$ 必然已經更新, 計算 $s_8$

$s_8 = \max(a_8, s_7, s_6, s_4)$

e.g. $a_2 \rightarrow \mathbf{A_2}$, $\mathbf{A_2} < a_2$, $s_8 == s_4 == s_2 == a_2$ ① $a_2 = A_2$,

# 樹狀數組解 RMQ (續)

$a_i \rightarrow A_i$, 修改所有 $a_i$ 的父節點 $s_j$

```
for (oldA=a[i],a[i]=Ai,j=i; j<=N; j+=lb(j)) {
    if (s[j]>oldA&&s[j]>Ai) break;
    if (Ai>s[j]&&Ai>oldA) s[j]=Ai;
    else {
        b = j-lb(j);
        for (max=a[j],k=j-1; k>b; k-=lb(k))
            if (s[k]>max) max=s[k];
        s[j]=max;
    }
}
```

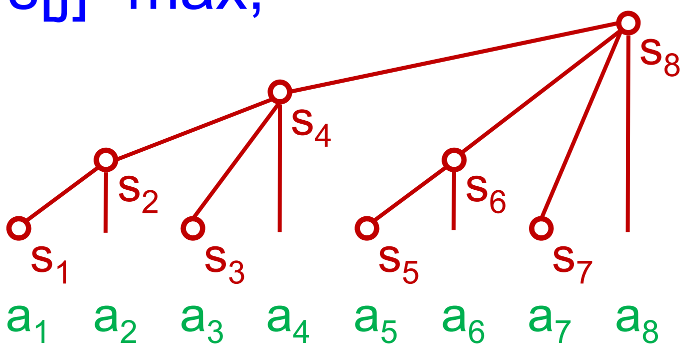
① $A_i > a_i \quad A_i >= s_j$

② $A_i > a_i \quad A_i < s_j$

③ $A_i < a_i \quad a_i < s_j$

④ $A_i < a_i \quad a_i == s_j$

重新搜尋整個子樹




子節點 $s_1, s_2, \ldots, s_7$ 必然已經更新, 計算 $s_8$

$s_8 = \max(a_8, s_7, s_6, s_4)$

e.g. $a_2 \rightarrow \mathbf{A_2}$, $\mathbf{A_2} < a_2$, $s_8 == s_4 == s_2 == a_2$ ① $a_2 = A_2$, ② **compare** $a_2, s_1$ **and update** $s_2$,

# 樹狀數組解 RMQ (續)

$a_i \rightarrow A_i$, 修改所有 $a_i$ 的父節點 $s_j$

```
for (oldA=a[i],a[i]=Ai,j=i; j<=N; j+=lb(j)) {
    if (s[j]>oldA&&s[j]>Ai) break;
    if (Ai>s[j]&&Ai>oldA) s[j]=Ai;
    else {
        b = j-lb(j);
        for (max=a[j],k=j-1; k>b; k-=lb(k))
            if (s[k]>max) max=s[k];
        s[j]=max;
    }
}
```

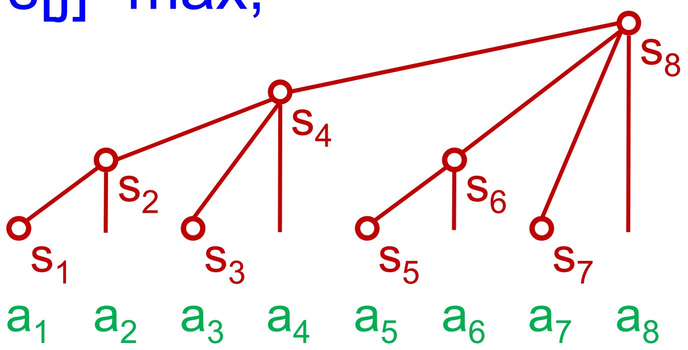
① $A_i > a_i$ $A_i >= s_j$

② $A_i > a_i$ $A_i < s_j$

③ $A_i < a_i$ $a_i < s_j$

④ $A_i < a_i$ $a_i == s_j$

重新搜尋整個子樹




子節點 $s_1, s_2, \ldots, s_7$ 必然已經更新, 計算 $s_8$

$s_8 = \max(a_8, s_7, s_6, s_4)$

e.g. $a_2 \rightarrow \mathbf{A_2}$, $\mathbf{A_2} < a_2$, $s_8 == s_4 == s_2 == a_2$ ① $a_2 = A_2$, ② **compare** $a_2, s_1$ **and update** $s_2$, ③ **compare** $a_4, s_3, s_2$ **and update** $s_4$,

# 樹狀數組解 RMQ (續)

$a_i \rightarrow A_i$, 修改所有 $a_i$ 的父節點 $s_j$

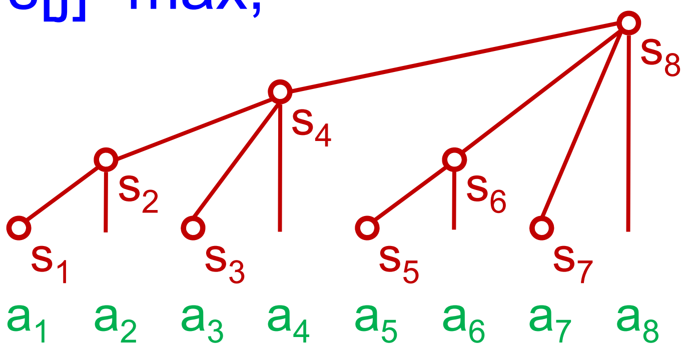
```
for (oldA=a[i],a[i]=Ai,j=i; j<=N; j+=lb(j)) {
    if (s[j]>oldA&&s[j]>Ai) break;
    if (Ai>s[j]&&Ai>oldA) s[j]=Ai;
    else {
        b = j-lb(j);
        for (max=a[j],k=j-1; k>b; k-=lb(k))
            if (s[k]>max) max=s[k];
        s[j]=max;
    }
}
```

① $A_i > a_i \quad A_i >= s_j$

② $A_i > a_i \quad A_i < s_j$

③ $A_i < a_i \quad a_i < s_j$

④ $A_i < a_i \quad a_i == s_j$




重新搜尋整個子樹

子節點 $s_1, s_2, \ldots, s_7$ 必然已經更新, 計算 $s_8$

$s_8 = \max(a_8, s_7, s_6, s_4)$

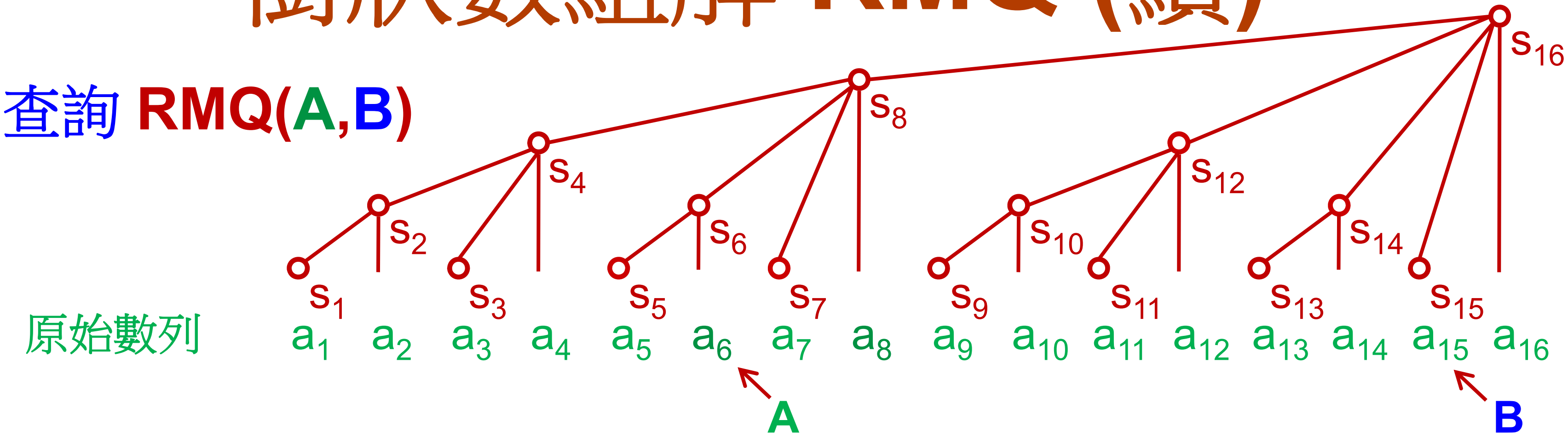
e.g. $a_2 \rightarrow \mathbf{A_2}$, $\mathbf{A_2} < a_2$, $s_8 == s_4 == s_2 == a_2$ ① $a_2 = A_2$, ② **compare** $a_2, s_1$ **and update** $s_2$, ③ **compare** $a_4, s_3, s_2$ **and update** $s_4$, ④ **compare** $a_8, s_7, s_6, s_4$ **and update** $s_8$

# 樹狀數組解 RMQ (續)

- 查詢 RMQ(A,B)

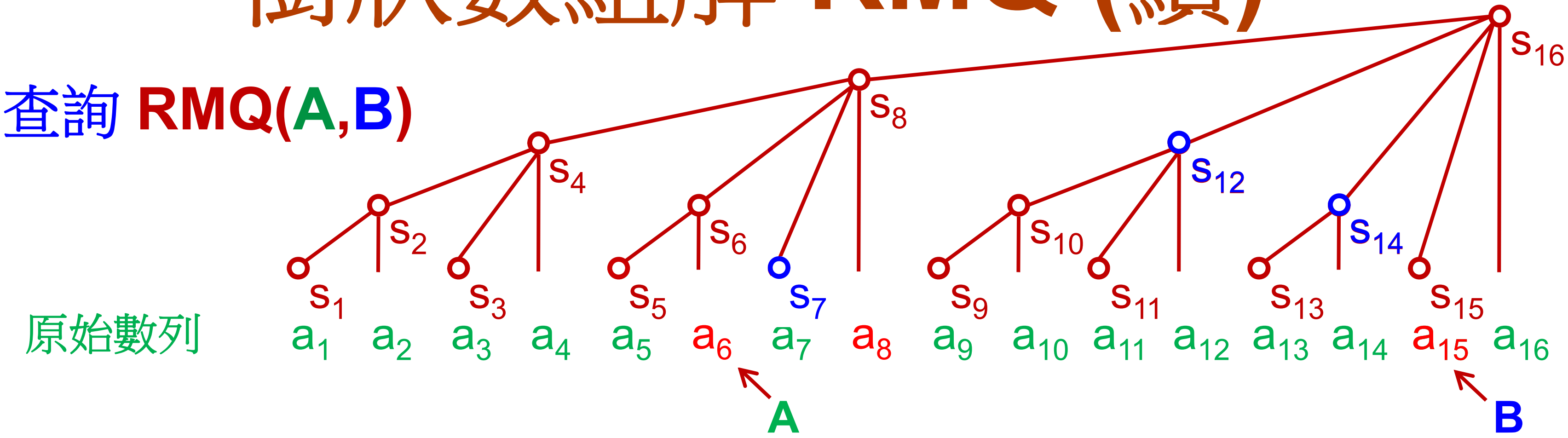
# 樹狀數組解 RMQ (續)

- 查詢 RMQ(A,B)



# 樹狀數組解 RMQ (續)

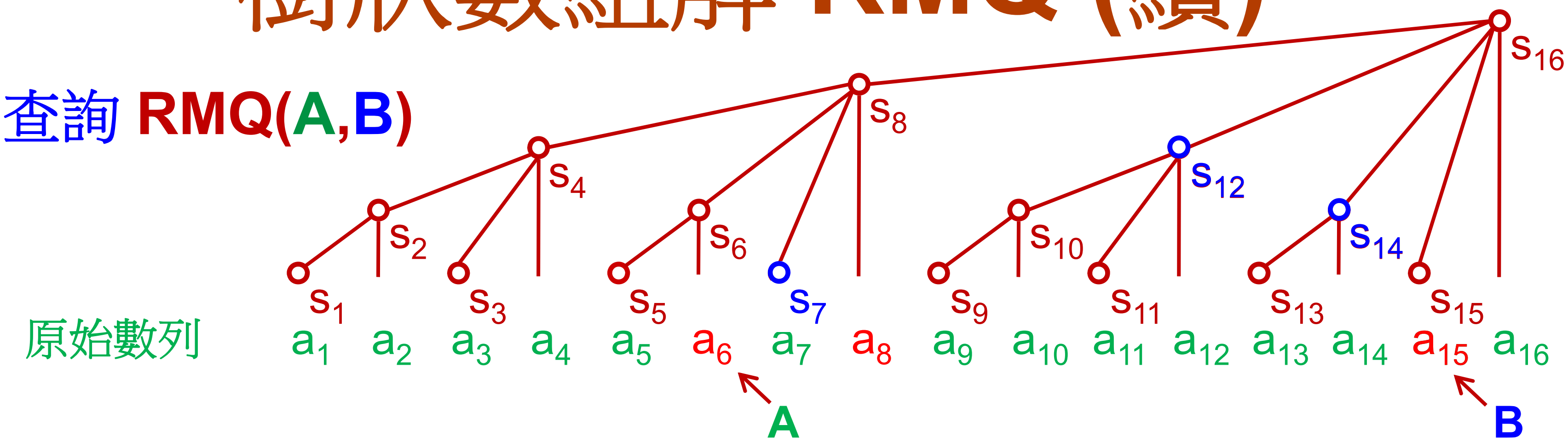
- 查詢 RMQ(A,B)




**RMQ(A,B) = max{$a_{15}$, $s_{14}$, $s_{12}$, $a_8$, $s_7$, $a_6$}**

# 樹狀數組解 RMQ (續)

- 查詢 **RMQ(A,B)**

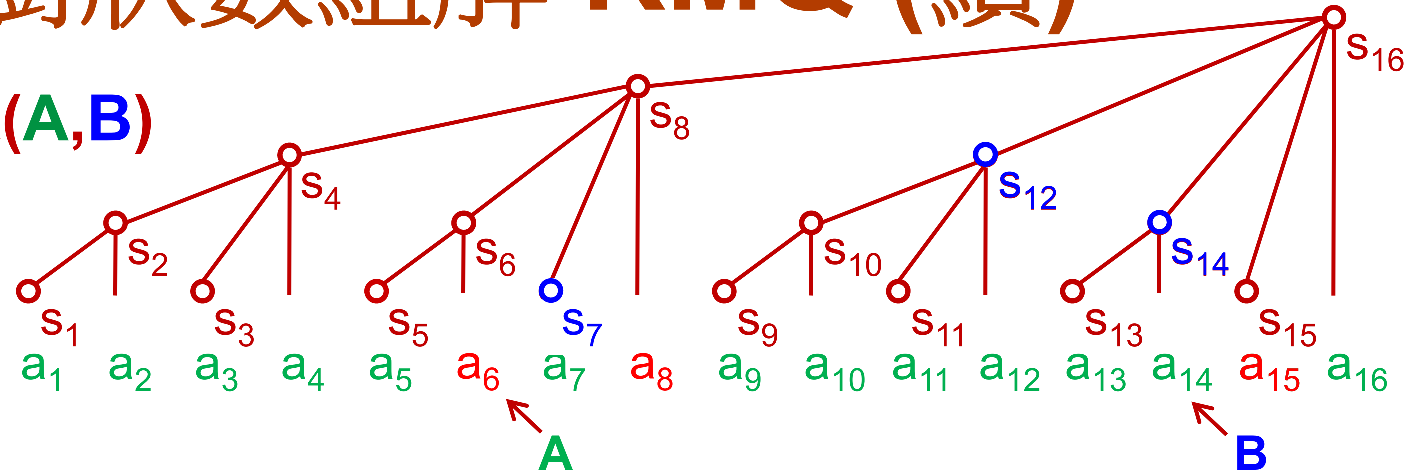



```
int RMQ(int A, int B) {

}
```

**RMQ(A,B) = max{$a_{15}$, $s_{14}$, $s_{12}$, $a_8$, $s_7$, $a_6$}**

# 樹狀數組解 RMQ (續)

- 查詢 **RMQ(A,B)**



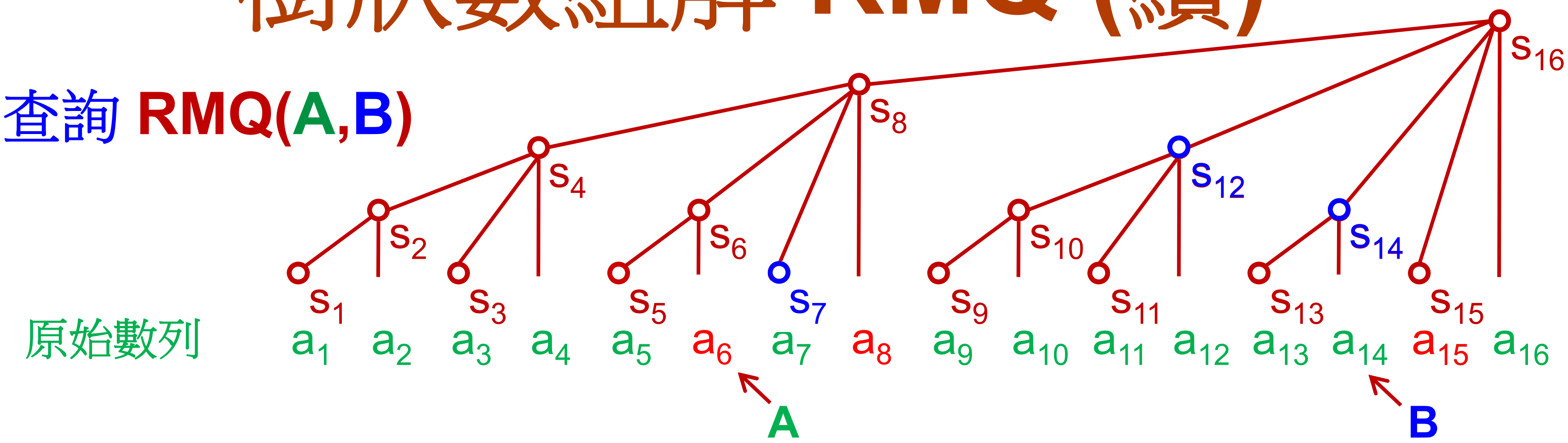

原始數列

```
int RMQ(int A, int B) {          a15
   int Bn, max = B%2 ? a[B--] : 0x80000000;



}
```

**RMQ(A,B) = max{$a_{15}$, $s_{14}$, $s_{12}$, $a_8$, $s_7$, $a_6$}**

# 樹狀數組解 RMQ (續)

- 查詢 RMQ(A,B)

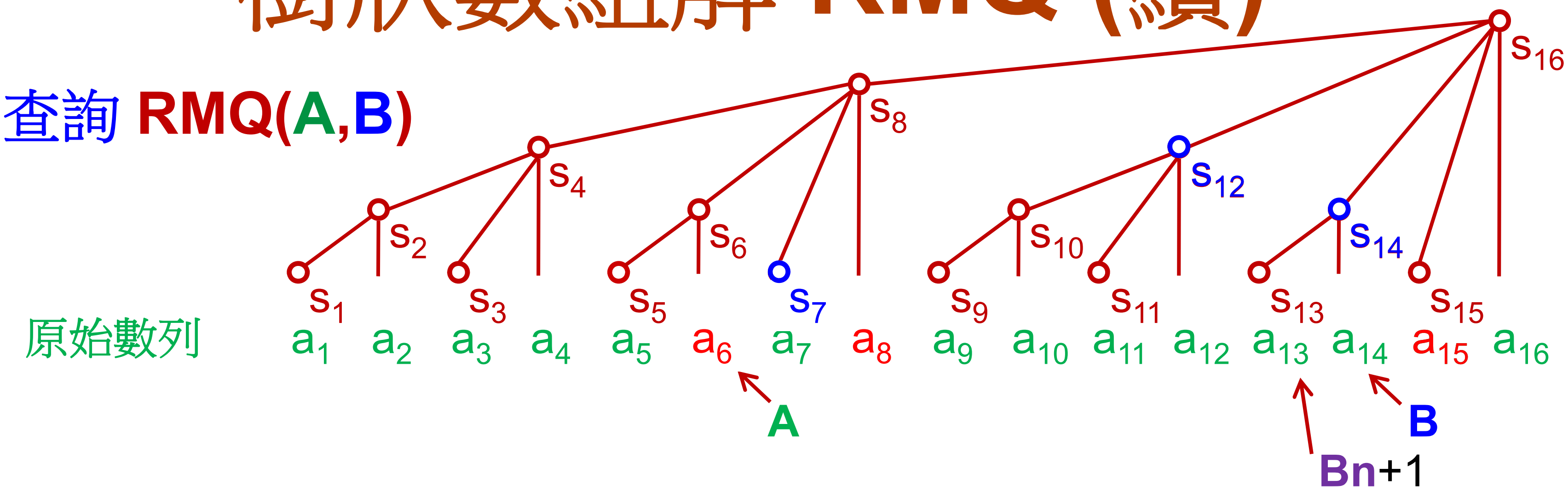



```
int RMQ(int A, int B) {
    int Bn, max = B%2 ? a[B--] : 0x80000000;
    while (A<=B) {



    }
}
```

**RMQ(A,B) = max{$a_{15}$, $s_{14}$, $s_{12}$, $a_8$, $s_7$, $a_6$}**

# 樹狀數組解 RMQ (續)

- 查詢 RMQ(A,B)

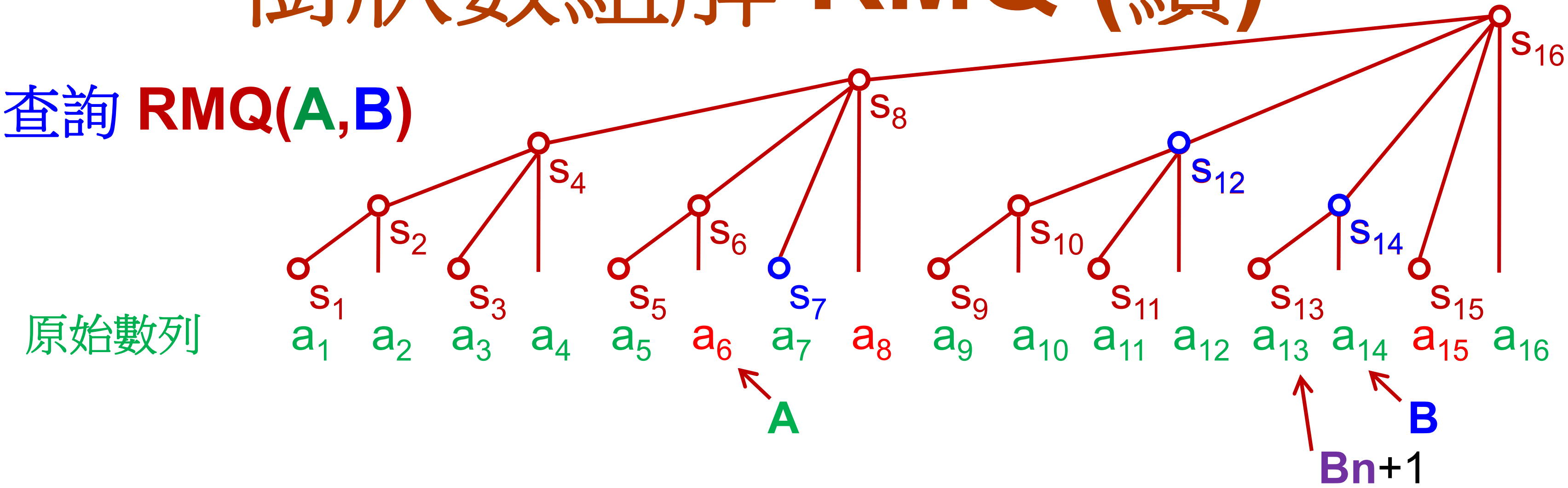



```
int RMQ(int A, int B) {
   int Bn, max = B%2 ? a[B--] : 0x80000000;
   while (A<=B) {
      for (Bn=B-lb(B); B>=A&&(Bn+1)>=A; B=Bn,Bn=B-lb(B))
         if (s[B]>max) max=s[B];



   }
}
```

$RMQ(A,B) = \max\{a_{15}, s_{14}, s_{12}, a_8, s_7, a_6\}$

# 樹狀數組解 RMQ (續)

- 查詢 RMQ(A,B)




```
int RMQ(int A, int B) {
   int Bn, max = B%2 ? a[B--] : 0x80000000;
   while (A<=B) {
      for (Bn=B-lb(B); B>=A&&(Bn+1)>=A; B=Bn,Bn=B-lb(B))
         if (s[B]>max) max=s[B];
```

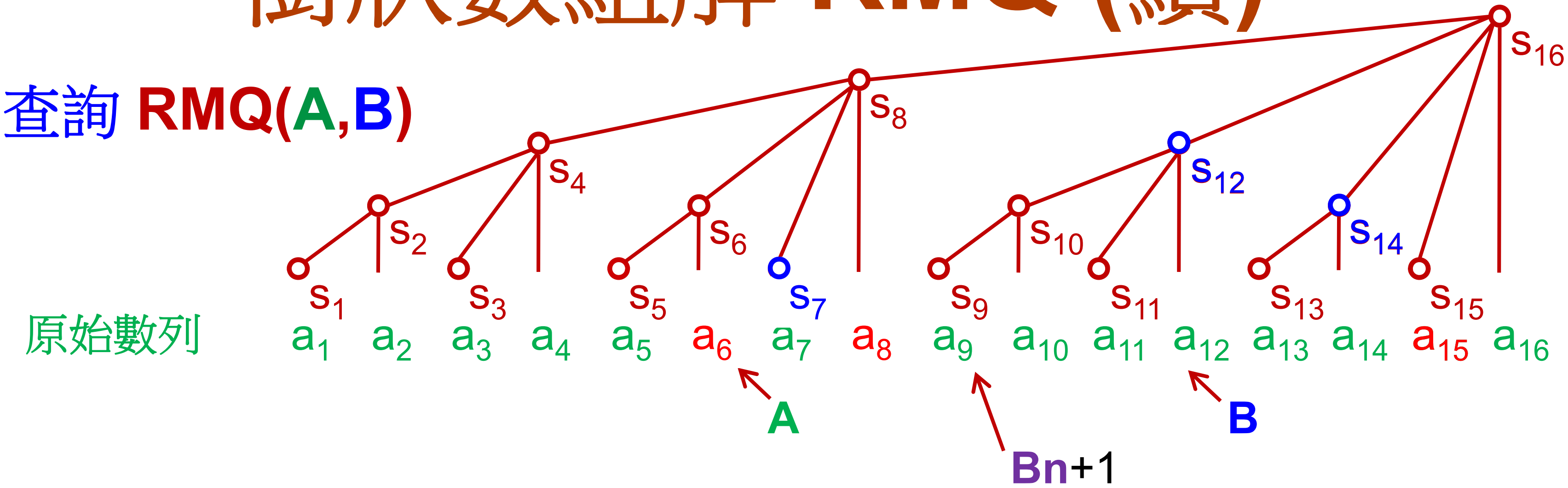
$s_{14}$

```
   }
}
```

RMQ(A,B) = max{$a_{15}$, $s_{14}$, $s_{12}$, $a_8$, $s_7$, $a_6$}

# 樹狀數組解 RMQ (續)

- 查詢 RMQ(A,B)




整個子樹都 ≥A

```
int RMQ(int A, int B) {
    int Bn, max = B%2 ? a[B--] : 0x80000000;
    while (A<=B) {
        for (Bn=B-lb(B); B>=A&&(Bn+1)>=A; B=Bn,Bn=B-lb(B))
            if (s[B]>max) max=s[B];
```

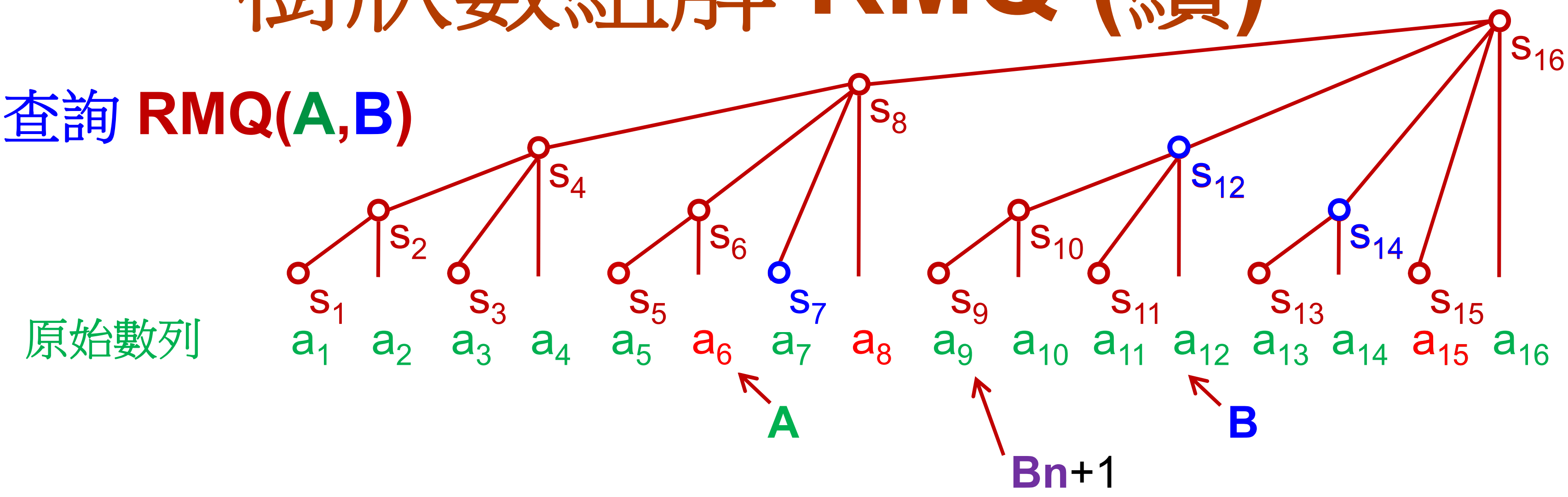
$s_{14}$

```
    }
}
```

RMQ(A,B) = max{$a_{15}$, $s_{14}$, $s_{12}$, $a_8$, $s_7$, $a_6$}

# 樹狀數組解 RMQ (續)

- 查詢 RMQ(A,B)




```
int RMQ(int A, int B) {
   int Bn, max = B%2 ? a[B--] : 0x80000000;
   while (A<=B) {
      for (Bn=B-lb(B); B>=A&&(Bn+1)>=A; B=Bn,Bn=B-lb(B))
         if (s[B]>max) max=s[B];
```

$s_{12}$

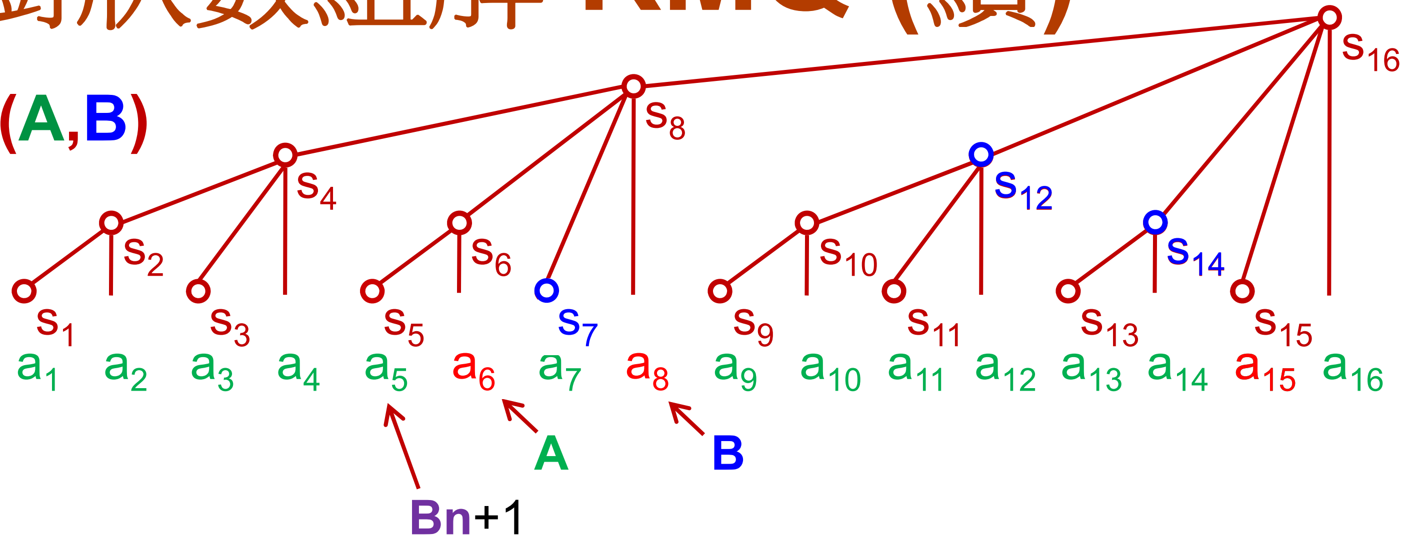
```
   }
}
```

**RMQ(A,B) = max{$a_{15}$, $s_{14}$, $s_{12}$, $a_8$, $s_7$, $a_6$}**

# 樹狀數組解 RMQ (續)

- 查詢 RMQ(A,B)

原始數列

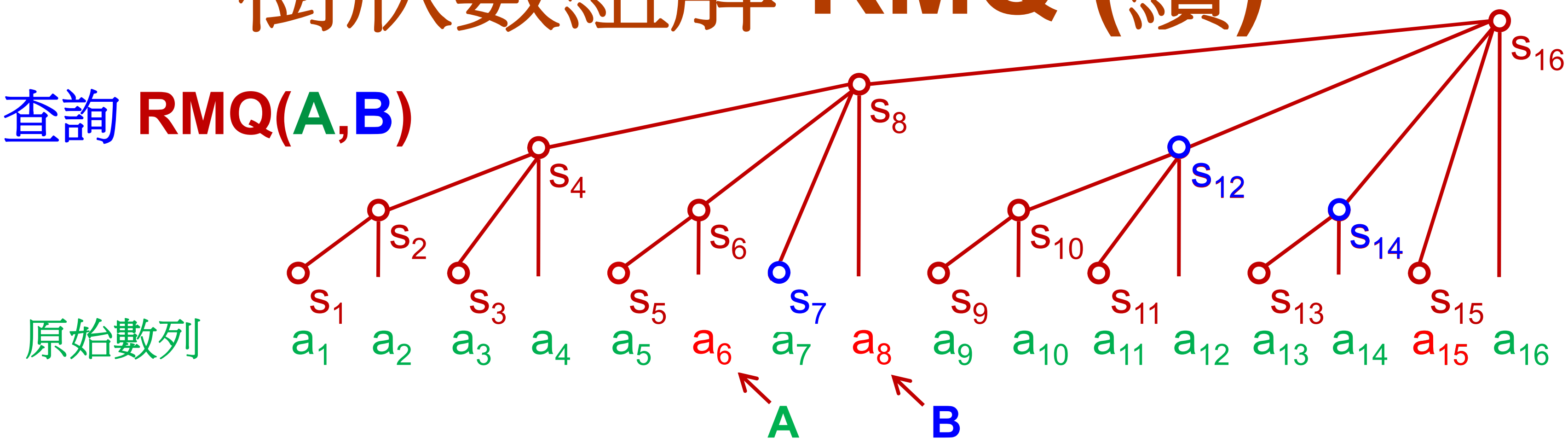



```
int RMQ(int A, int B) {
   int Bn, max = B%2 ? a[B--] : 0x80000000;
   while (A<=B) {
      for (Bn=B-lb(B); B>=A&&(Bn+1)>=A; B=Bn,Bn=B-lb(B))
         if (s[B]>max) max=s[B];


   }
}
```

**RMQ(A,B) = max{$a_{15}$, $s_{14}$, $s_{12}$, $a_8$, $s_7$, $a_6$}**

# 樹狀數組解 RMQ (續)

- 查詢 RMQ(A,B)

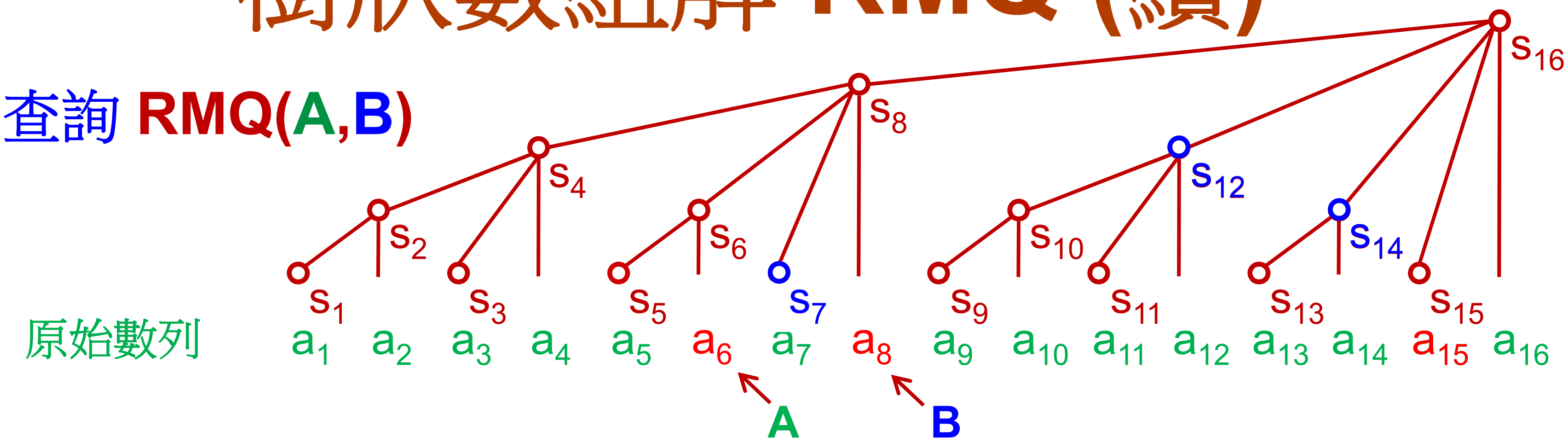



```
int RMQ(int A, int B) {
    int Bn, max = B%2 ? a[B--] : 0x80000000;
    while (A<=B) {
        for (Bn=B-lb(B); B>=A&&(Bn+1)>=A; B=Bn,Bn=B-lb(B))
            if (s[B]>max) max=s[B];
        if (B>=A) {
            if (a[B]>max) max=a[B];
            B--;
        }
    }
}
```

**RMQ(A,B) = max{$a_{15}$, $s_{14}$, $s_{12}$, $a_8$, $s_7$, $a_6$}**

# 樹狀數組解 RMQ (續)

- 查詢 **RMQ(A,B)**



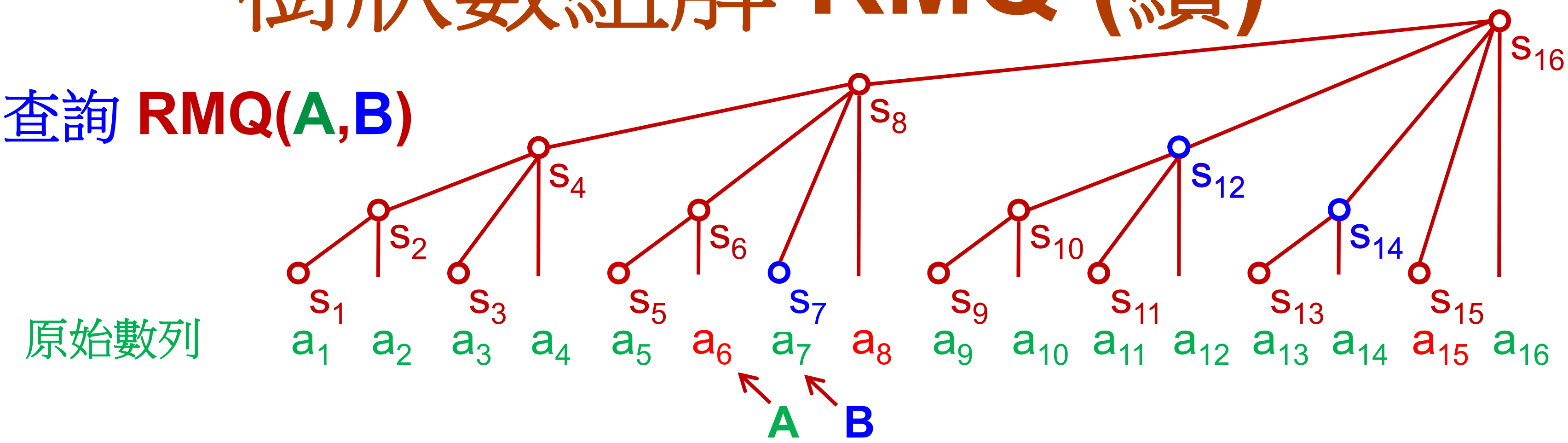

```
int RMQ(int A, int B) {
    int Bn, max = B%2 ? a[B--] : 0x80000000;
    while (A<=B) {
        for (Bn=B-lb(B); B>=A&&(Bn+1)>=A; B=Bn,Bn=B-lb(B))
            if (s[B]>max) max=s[B];
        if (B>=A) {
            if (a[B]>max) max=a[B];
            B--;
        }
    }
}
```

**$a_8$**

**RMQ(A,B) = max{$a_{15}$, $s_{14}$, $s_{12}$, $a_8$, $s_7$, $a_6$}**

# 樹狀數組解 RMQ (續)

- 查詢 RMQ(A,B)



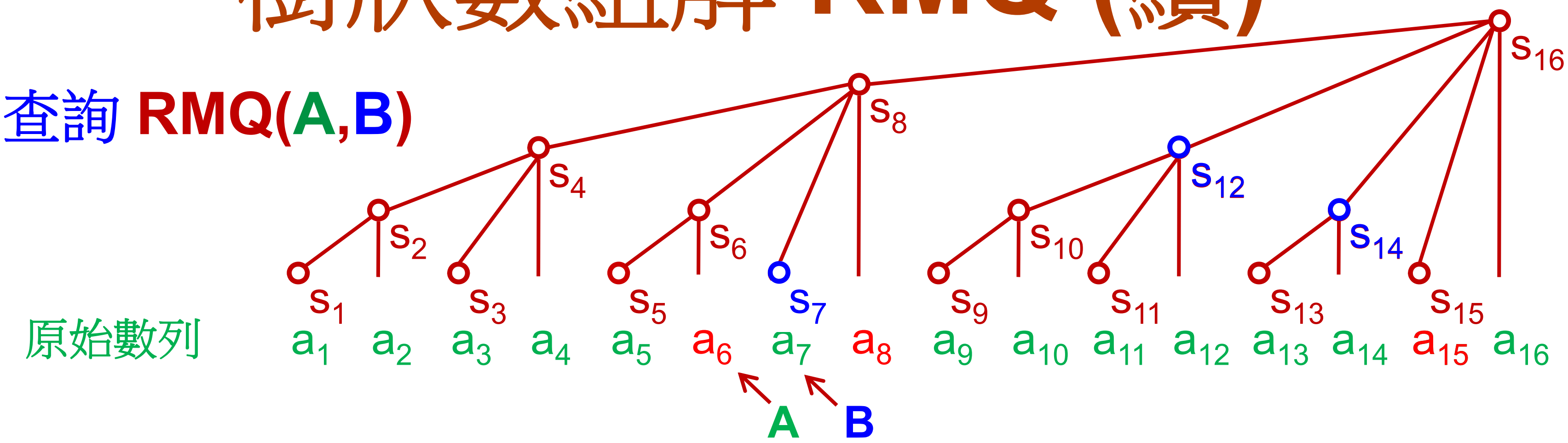

```
int RMQ(int A, int B) {
   int Bn, max = B%2 ? a[B--] : 0x80000000;
   while (A<=B) {
      for (Bn=B-lb(B); B>=A&&(Bn+1)>=A; B=Bn,Bn=B-lb(B))
         if (s[B]>max) max=s[B];
      if (B>=A) {
         if (a[B]>max) max=a[B];
         B--;
      }
   }
}
```

$a_8$

$\mathbf{RMQ(A,B) = max\{a_{15}, s_{14}, s_{12}, a_8, s_7, a_6\}}$

# 樹狀數組解 RMQ (續)

- 查詢 RMQ(A,B)

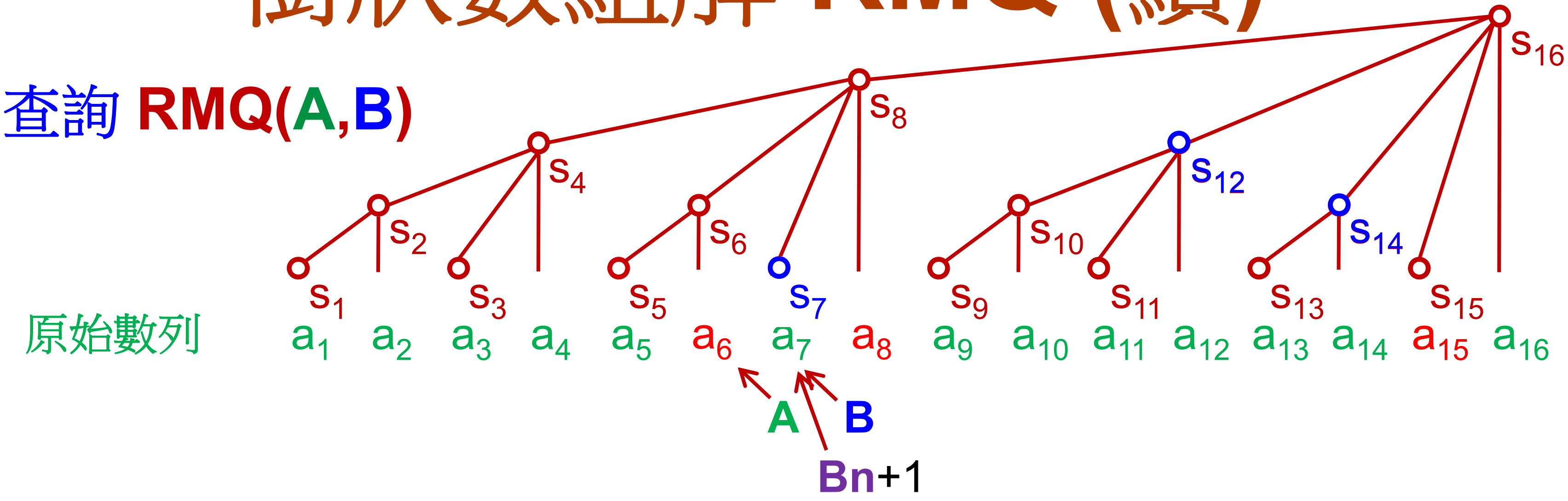



```
int RMQ(int A, int B) {
   int Bn, max = B%2 ? a[B--] : 0x80000000;
   while (A<=B) {



   }
}
```

$RMQ(A,B) = \max\{a_{15}, s_{14}, s_{12}, a_8, s_7, a_6\}$

# 樹狀數組解 RMQ (續)

- 查詢 **RMQ(A,B)**



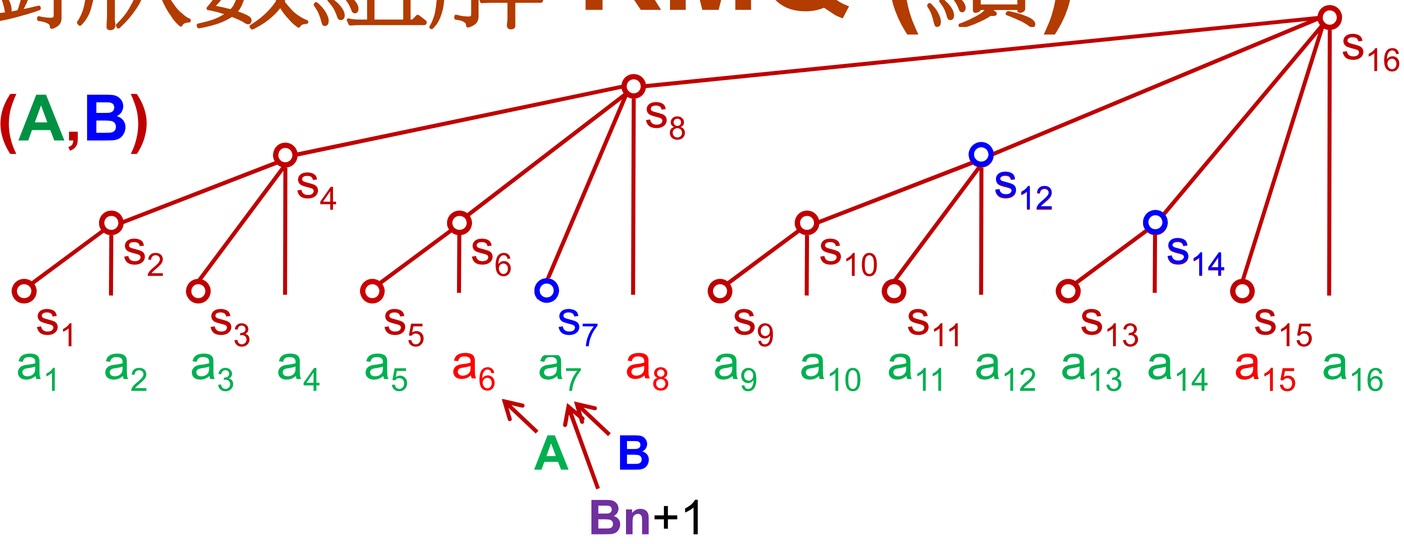

```
int RMQ(int A, int B) {
   int Bn, max = B%2 ? a[B--] : 0x80000000;
   while (A<=B) {
      for (Bn=B-lb(B); B>=A&&(Bn+1)>=A; B=Bn,Bn=B-lb(B))
         if (s[B]>max) max=s[B];



   }
}
```

**RMQ(A,B) = max{$a_{15}$, $s_{14}$, $s_{12}$, $a_8$, $s_7$, $a_6$}**

# 樹狀數組解 RMQ (續)

- 查詢 RMQ(A,B)

原始數列




```
int RMQ(int A, int B) {
    int Bn, max = B%2 ? a[B--] : 0x80000000;
    while (A<=B) {
        for (Bn=B-lb(B); B>=A&&(Bn+1)>=A; B=Bn,Bn=B-lb(B))
            if (s[B]>max) max=s[B];
```

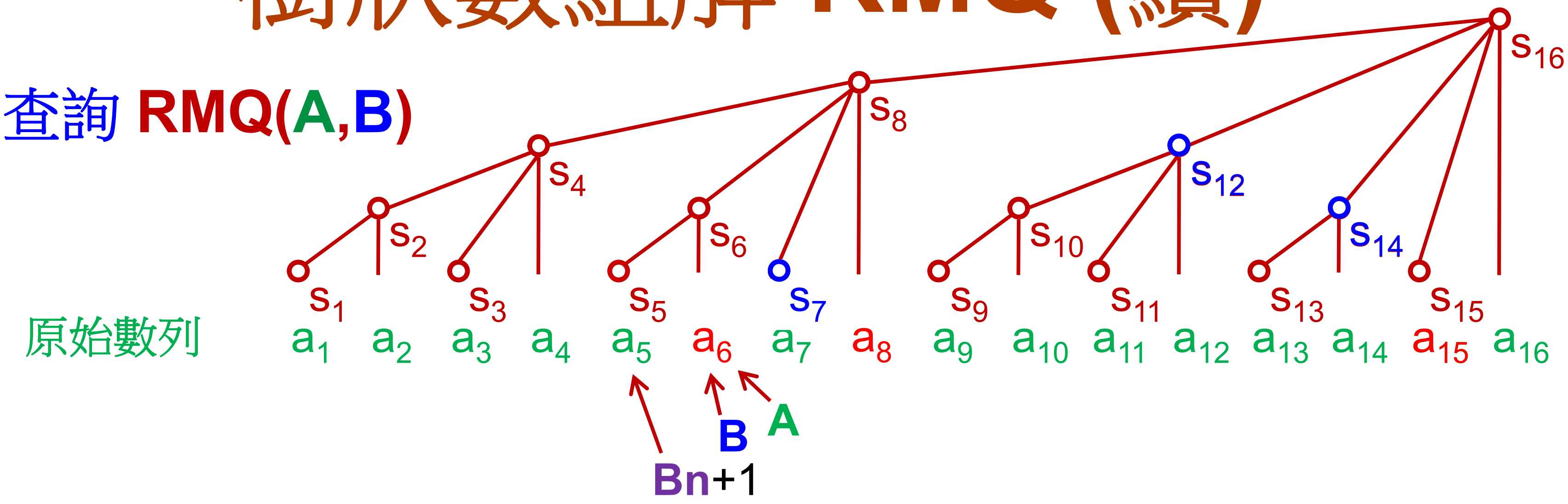
$s_7$

```
    }
}
```

$RMQ(A,B) = \max\{a_{15}, s_{14}, s_{12}, a_8, s_7, a_6\}$

# 樹狀數組解 RMQ (續)

- 查詢 RMQ(A,B)




```
int RMQ(int A, int B) {
    int Bn, max = B%2 ? a[B--] : 0x80000000;
    while (A<=B) {
        for (Bn=B-lb(B); B>=A&&(Bn+1)>=A; B=Bn,Bn=B-lb(B))
            if (s[B]>max) max=s[B];
```

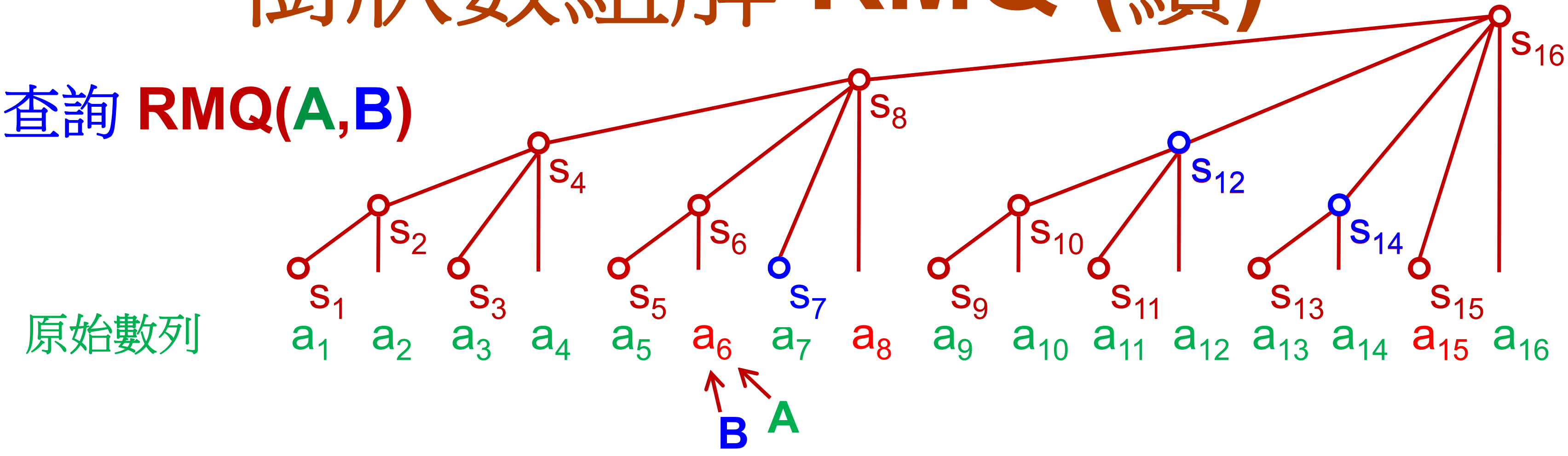
$s_7$

```
    }
}
```

**RMQ(A,B) = max{$a_{15}$, $s_{14}$, $s_{12}$, $a_8$, $s_7$, $a_6$}**

# 樹狀數組解 RMQ (續)

- 查詢 **RMQ(A,B)**

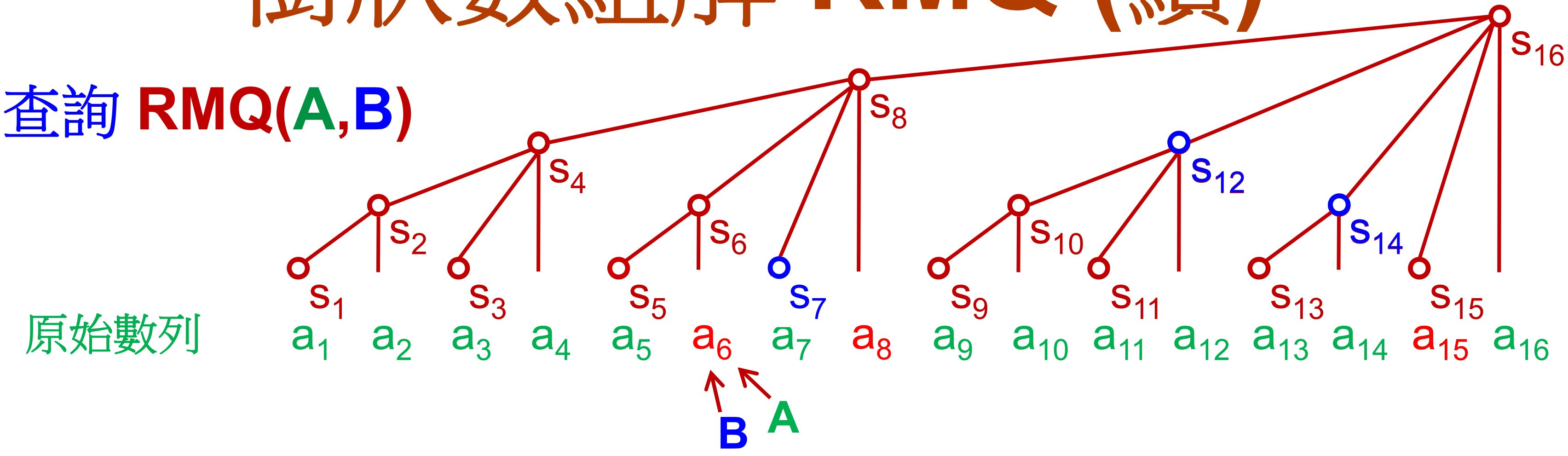



```
int RMQ(int A, int B) {
    int Bn, max = B%2 ? a[B--] : 0x80000000;
    while (A<=B) {
        for (Bn=B-lb(B); B>=A&&(Bn+1)>=A; B=Bn,Bn=B-lb(B))
            if (s[B]>max) max=s[B];
        if (B>=A) {
            if (a[B]>max) max=a[B];
            B--;
        }
    }
}
```

$$\mathrm{RMQ}(A,B) = \max\{a_{15}, s_{14}, s_{12}, a_8, s_7, a_6\}$$

# 樹狀數組解 RMQ (續)

- 查詢 **RMQ(A,B)**



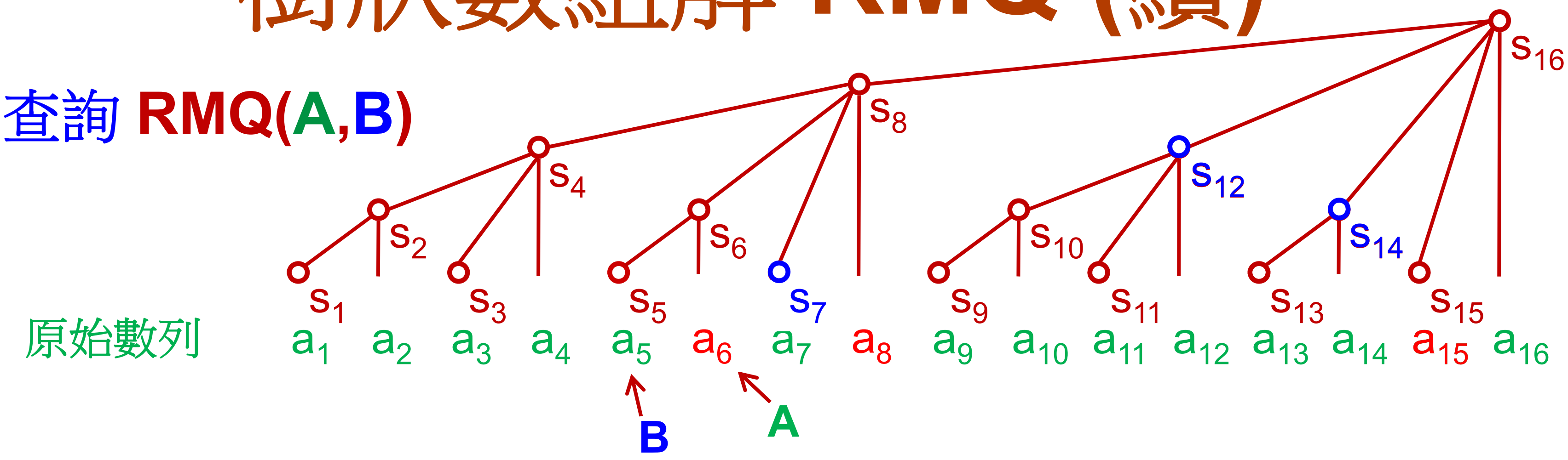

```
int RMQ(int A, int B) {
   int Bn, max = B%2 ? a[B--] : 0x80000000;
   while (A<=B) {
      for (Bn=B-lb(B); B>=A&&(Bn+1)>=A; B=Bn,Bn=B-lb(B))
         if (s[B]>max) max=s[B];
      if (B>=A) {
         if (a[B]>max) max=a[B];
         B--;
      }
   }
}
```

$a_6$

**RMQ(A,B) = max{$a_{15}$, $s_{14}$, $s_{12}$, $a_8$, $s_7$, $a_6$}**

# 樹狀數組解 RMQ (續)

- 查詢 RMQ(A,B)

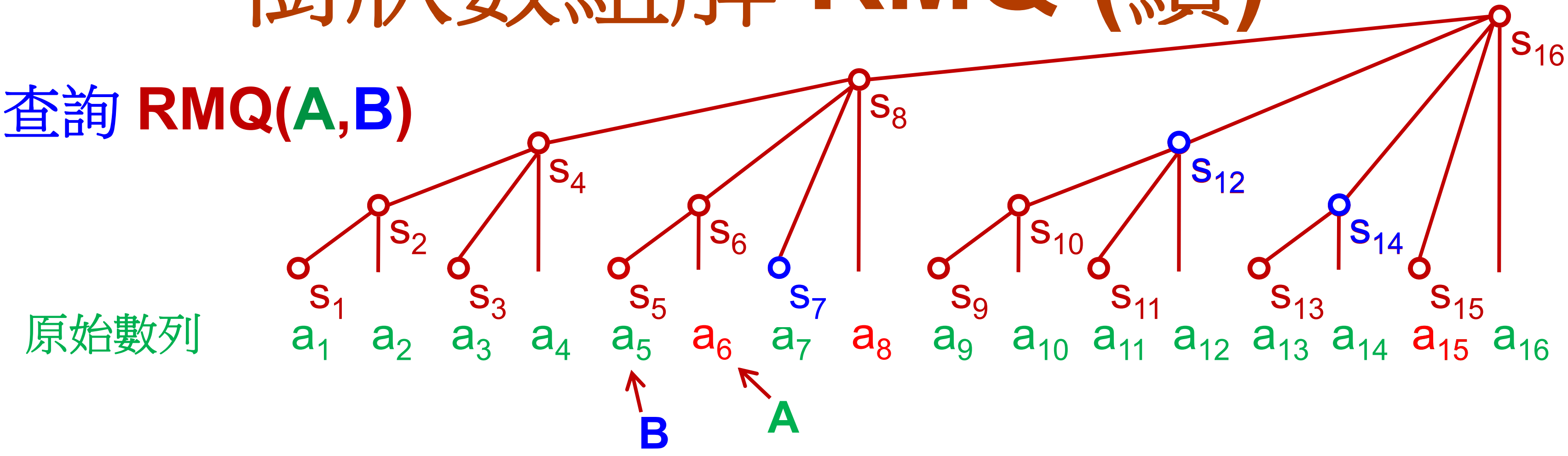



```
int RMQ(int A, int B) {
   int Bn, max = B%2 ? a[B--] : 0x80000000;
   while (A<=B) {
      for (Bn=B-lb(B); B>=A&&(Bn+1)>=A; B=Bn,Bn=B-lb(B))
         if (s[B]>max) max=s[B];
      if (B>=A) {
         if (a[B]>max) max=a[B];
         B--;
      }
   }
}
```

$RMQ(A,B) = \max\{a_{15}, s_{14}, s_{12}, a_8, s_7, a_6\}$

# 樹狀數組解 RMQ (續)

- 查詢 RMQ(A,B)



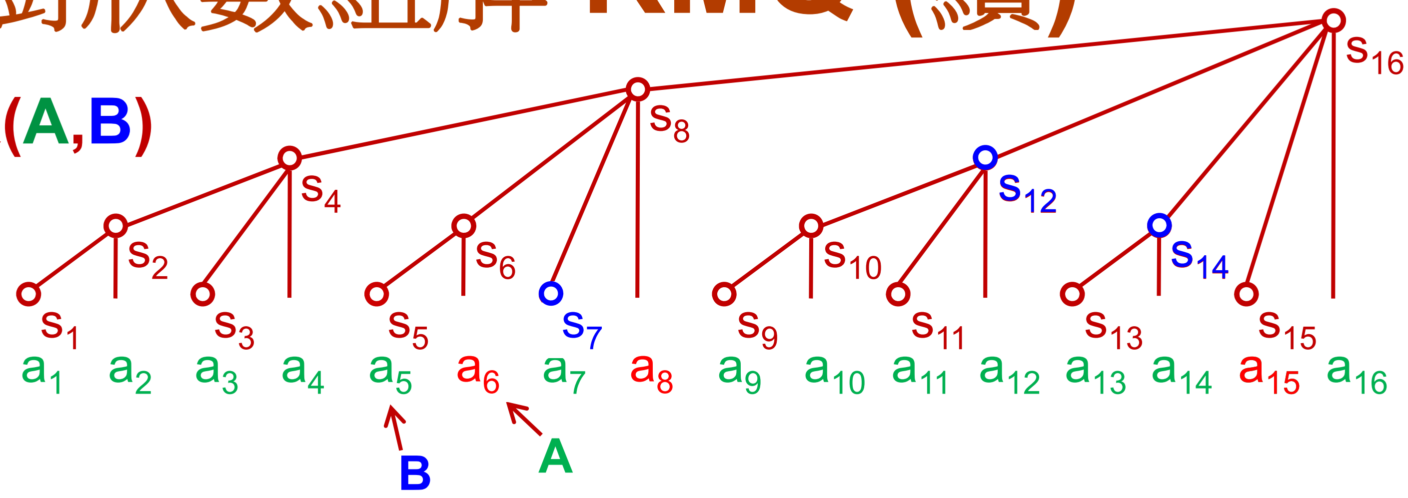

```
int RMQ(int A, int B) {
   int Bn, max = B%2 ? a[B--] : 0x80000000;
   while (A<=B) {




   }
}
```

RMQ(A,B) = max{$a_{15}$, $s_{14}$, $s_{12}$, $a_8$, $s_7$, $a_6$}

# 樹狀數組解 RMQ (續)

- 查詢 **RMQ(A,B)**

原始數列




```
int RMQ(int A, int B) {
    int Bn, max = B%2 ? a[B--] : 0x80000000;
    while (A<=B) {



    }
    return max;
}
```

**RMQ(A,B) = max{** $a_{15}$, $s_{14}$, $s_{12}$, $a_8$, $s_7$, $a_6$ **}**