Merging-Style Problems

Pei-yih Ting

2016/01

Contents

- Merge Sort
- LC263 Ugly Number
- LC264 Ugly Number II
- LC313 Super Ugly Number
- UVa443 Humble Number
- LC315 Count of Smaller Numbers after Self
  逆序數 (逆序對backward sequence個數)
- LC327 Count of Range Sum
- 其他應用

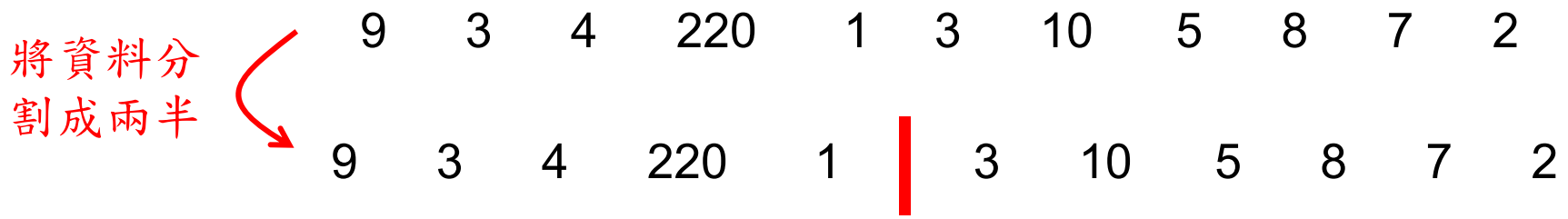
合併排序法 (Merge Sort)

- 以一個範例來說明基本的演算法

9 3 4 220 1 3 10 5 8 7 2

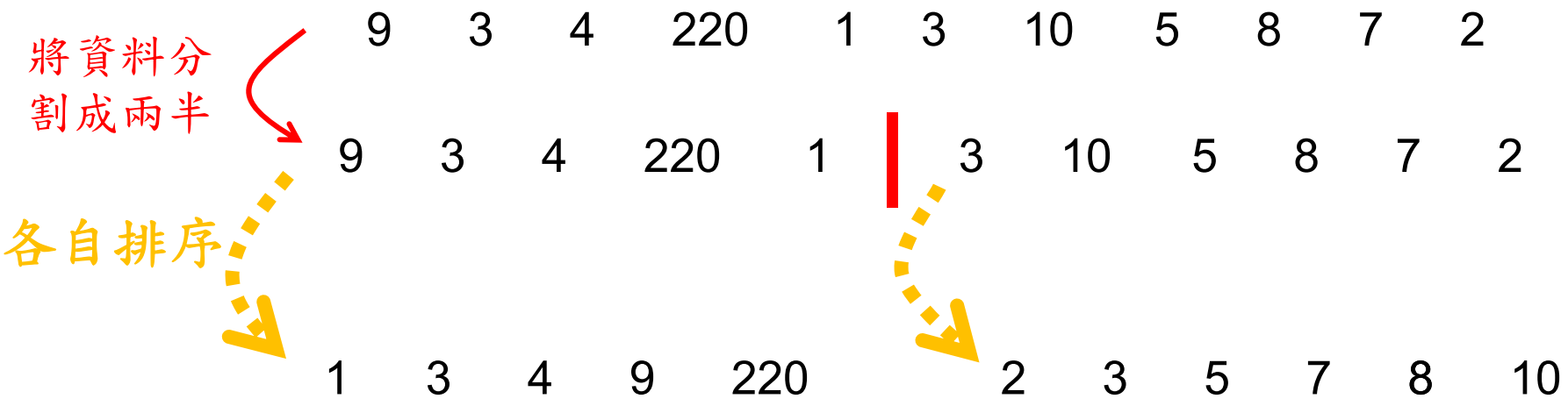
合併排序法 (Merge Sort)

- 以一個範例來說明基本的演算法



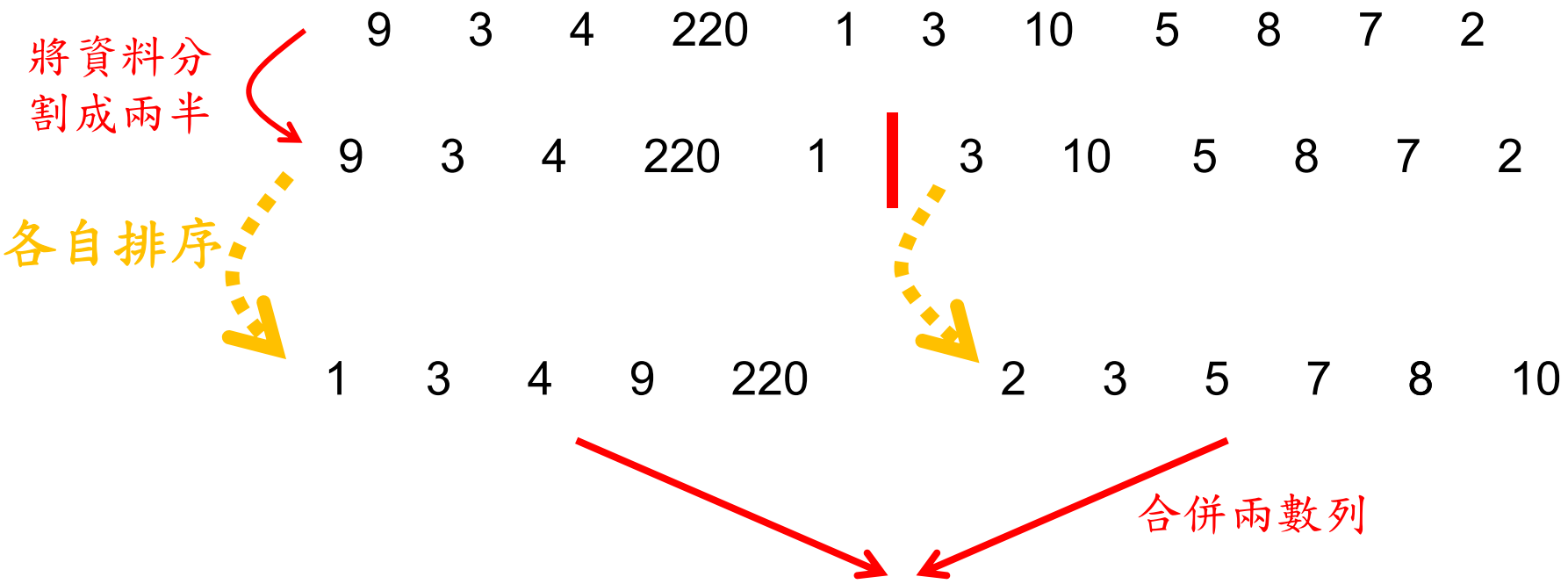
合併排序法 (Merge Sort)

- 以一個範例來說明基本的演算法



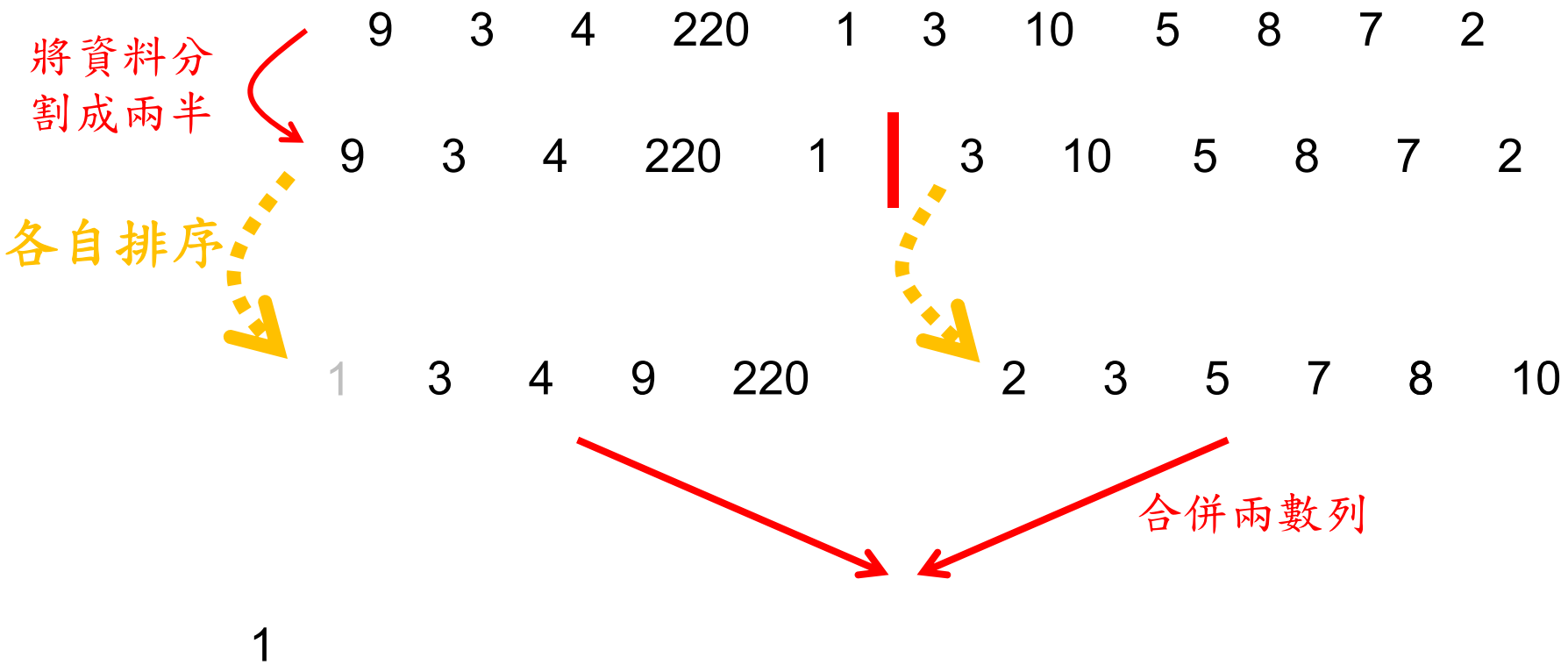
合併排序法 (Merge Sort)

- 以一個範例來說明基本的演算法



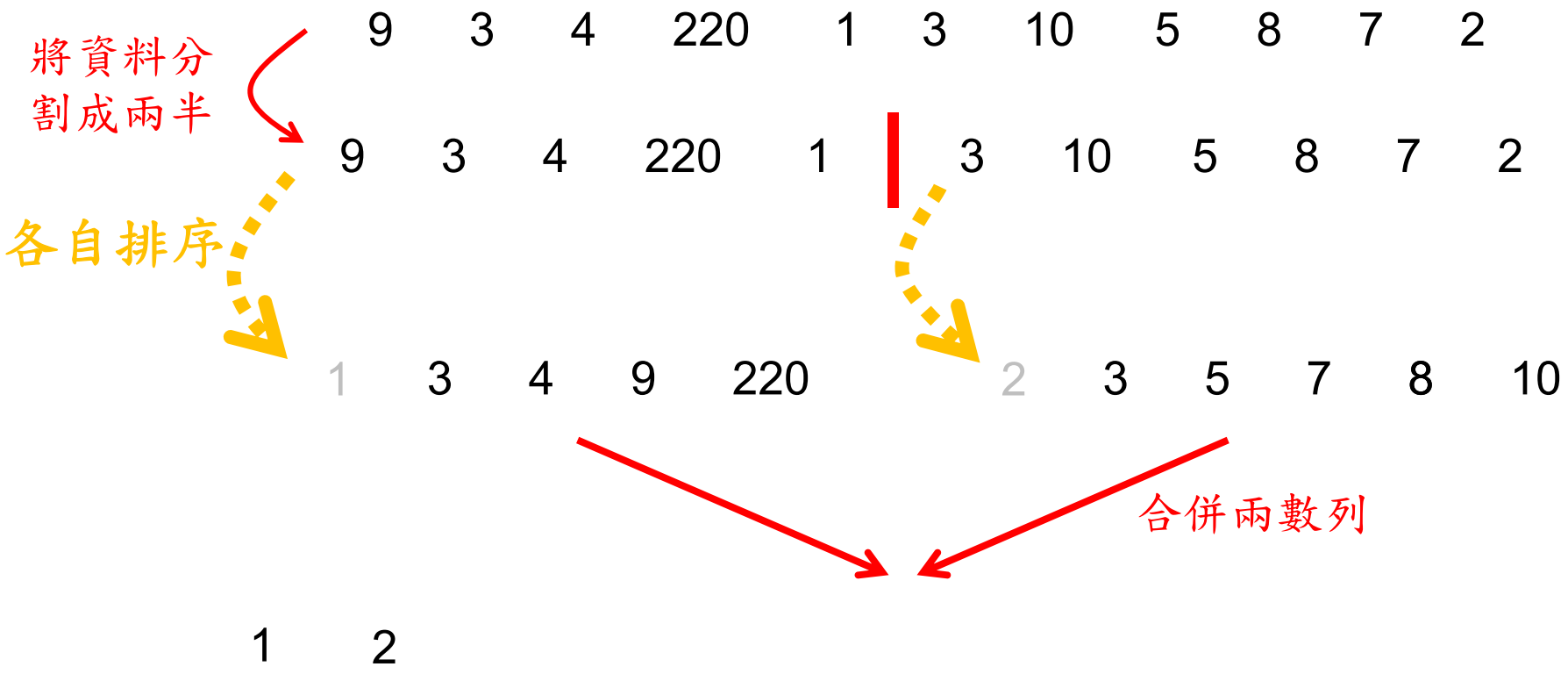
合併排序法 (Merge Sort)

- 以一個範例來說明基本的演算法



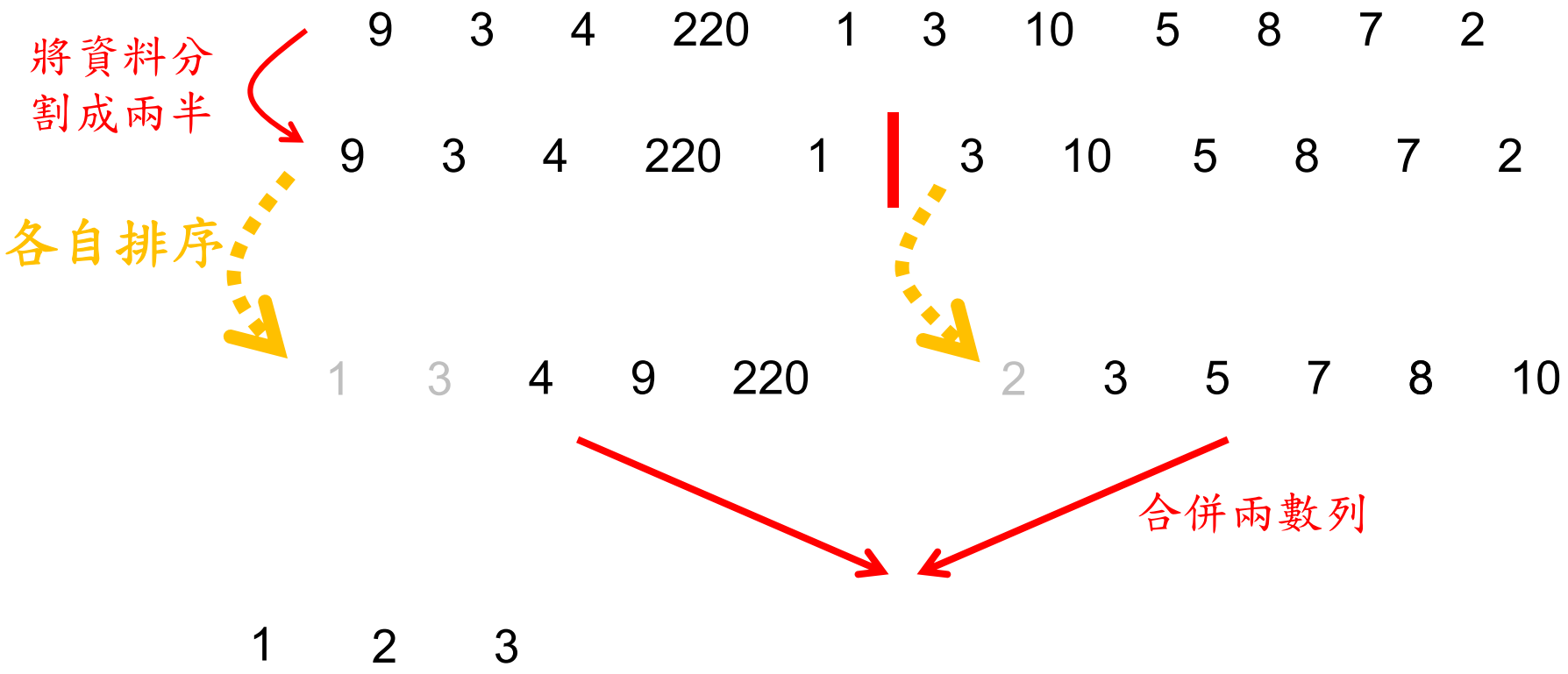
合併排序法 (Merge Sort)

- 以一個範例來說明基本的演算法



合併排序法 (Merge Sort)

- 以一個範例來說明基本的演算法



合併排序法 (Merge Sort)

- 以一個範例來說明基本的演算法

合併排序法 (Merge Sort)

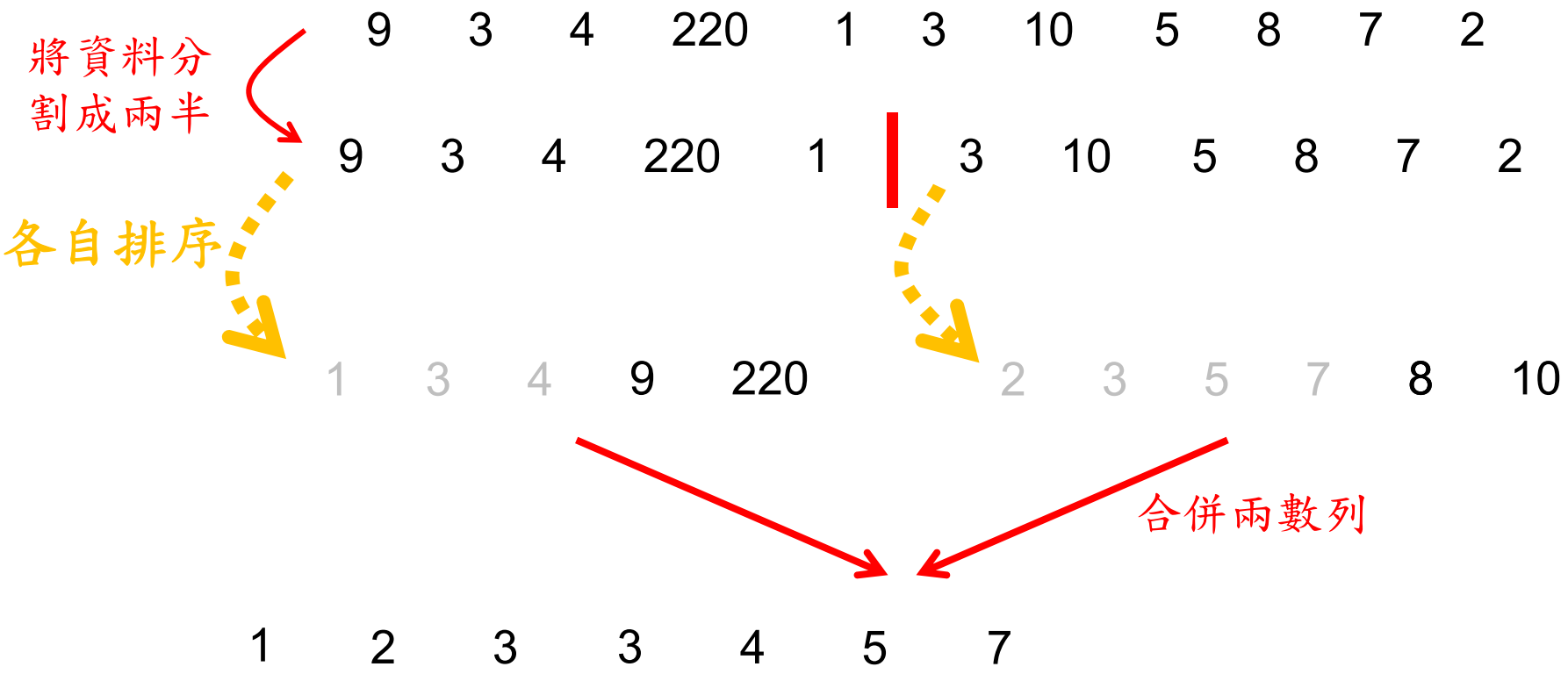
- 以一個範例來說明基本的演算法

合併排序法 (Merge Sort)

- 以一個範例來說明基本的演算法

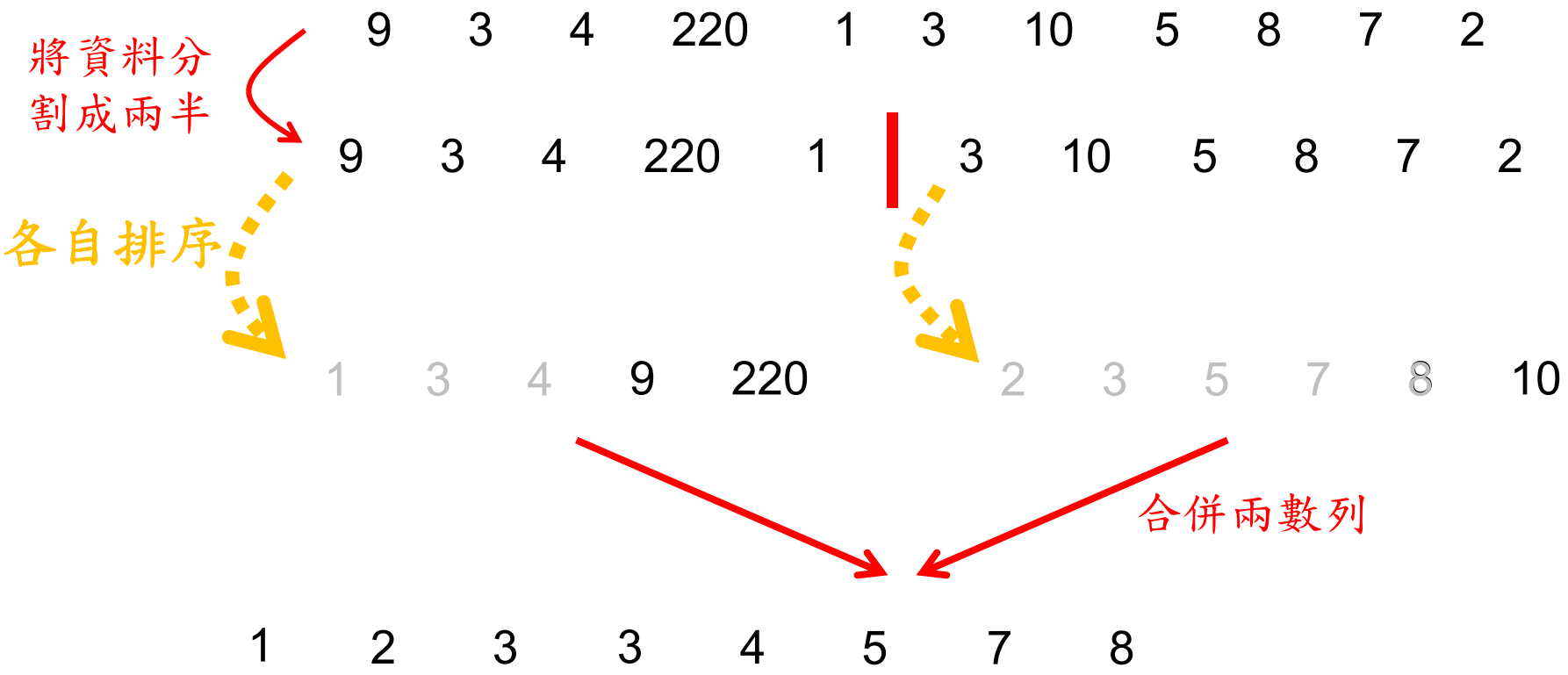
合併排序法 (Merge Sort)

- 以一個範例來說明基本的演算法



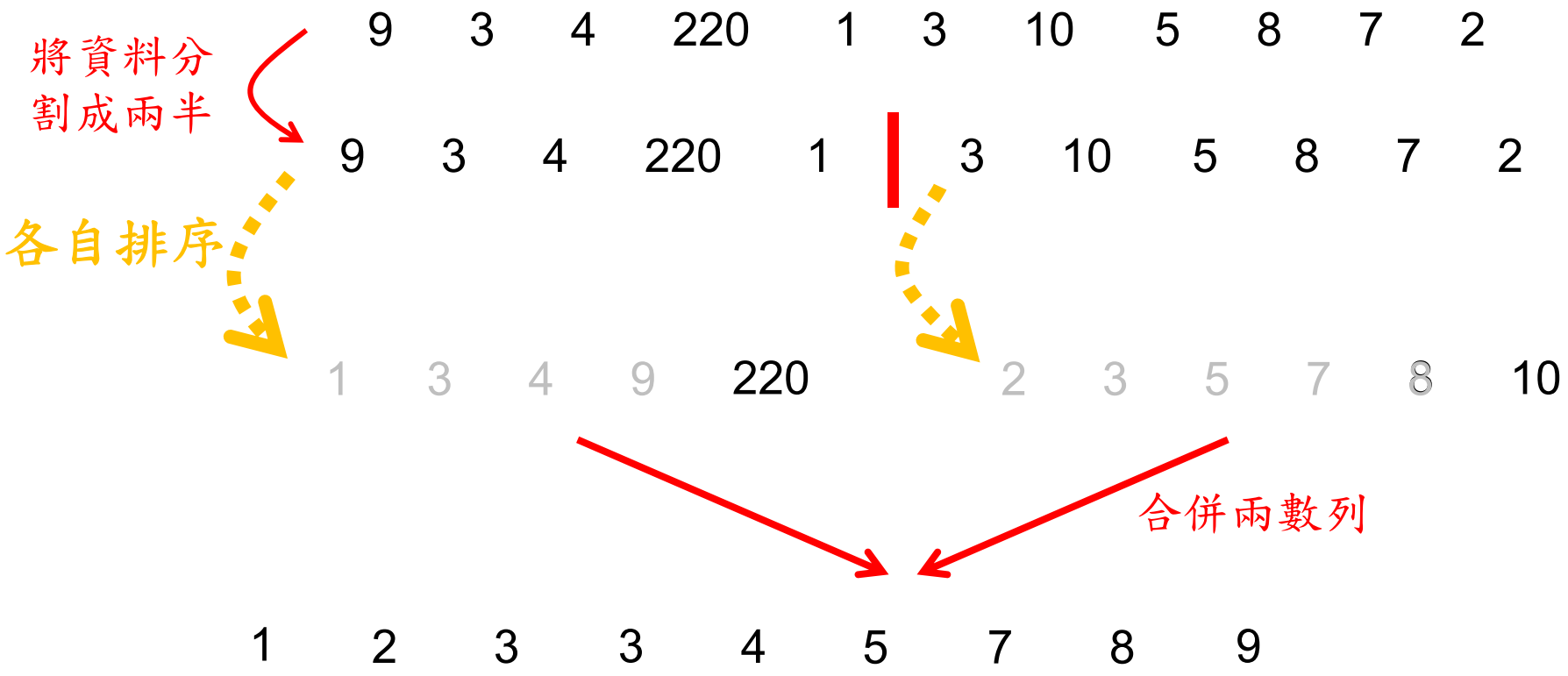
合併排序法 (Merge Sort)

- 以一個範例來說明基本的演算法



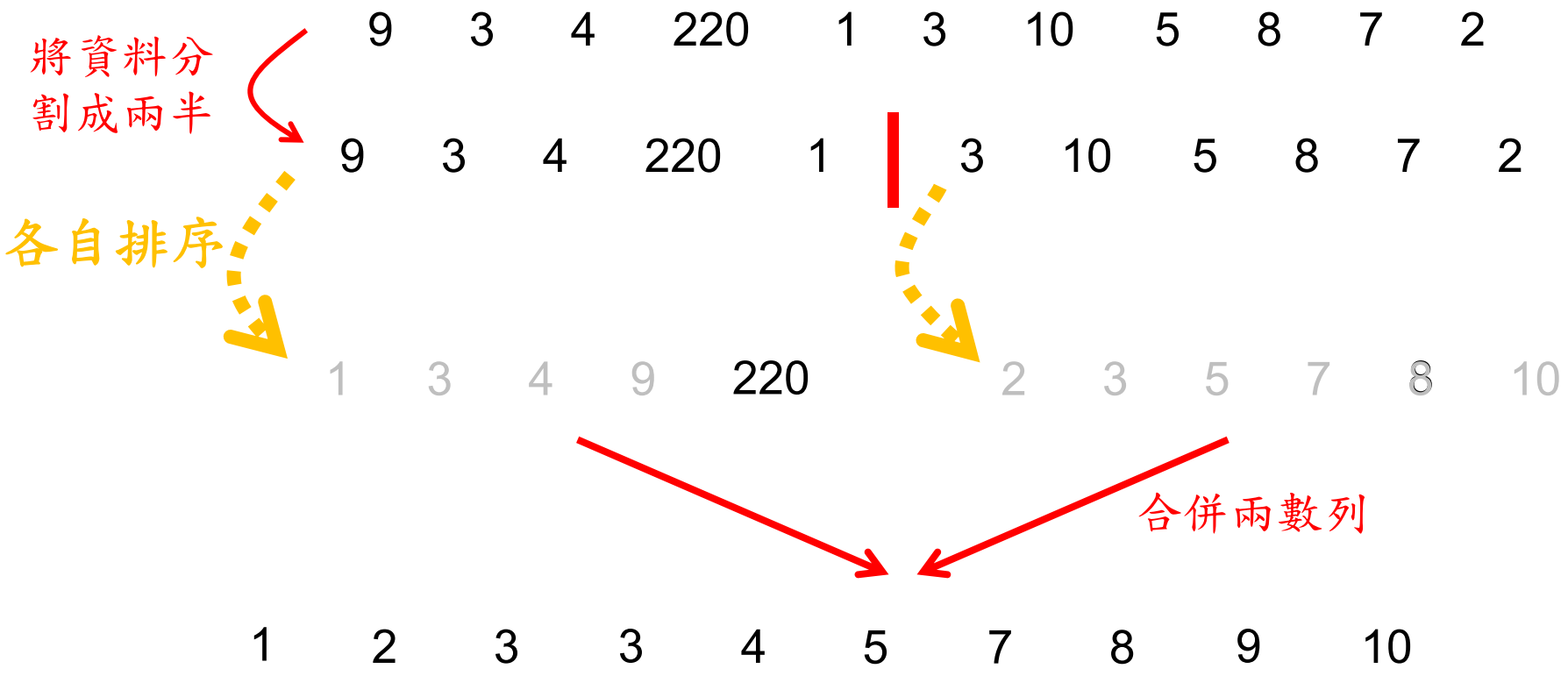
合併排序法 (Merge Sort)

- 以一個範例來說明基本的演算法



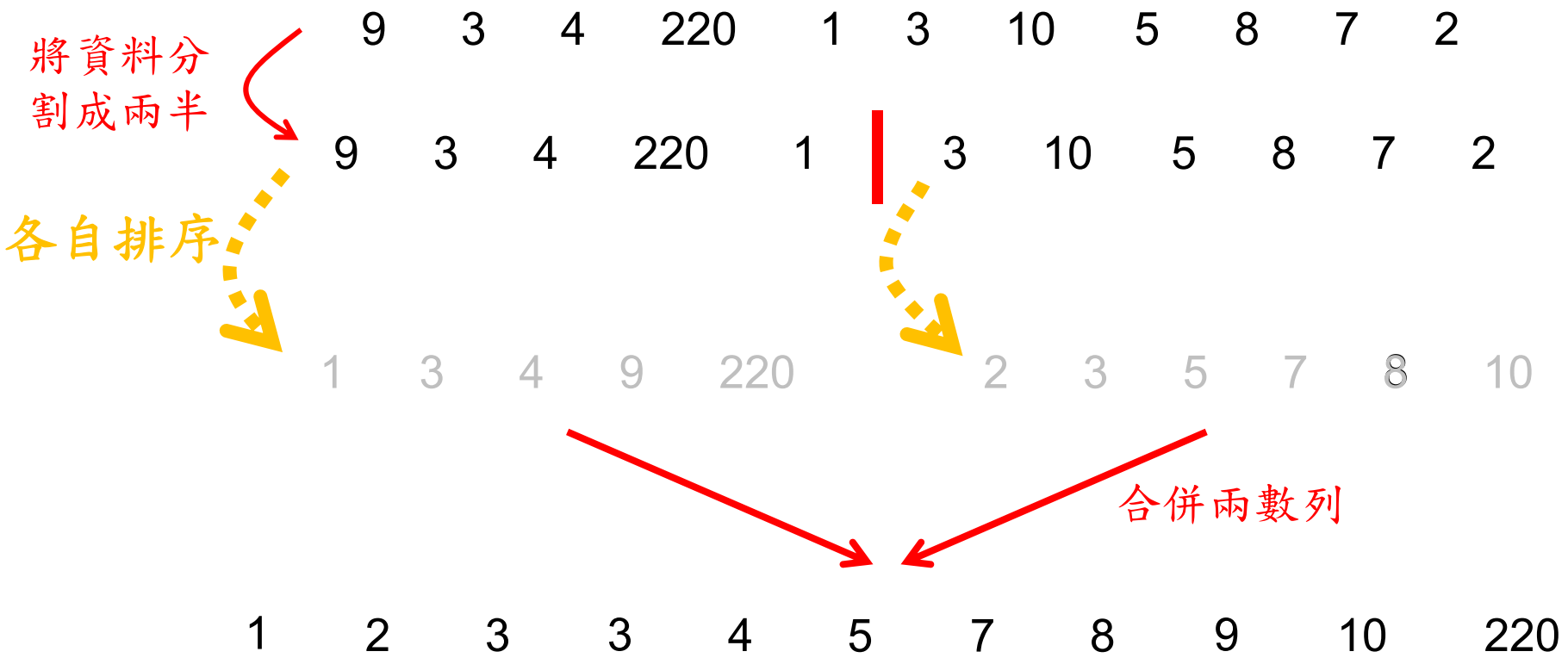
合併排序法 (Merge Sort)

- 以一個範例來說明基本的演算法



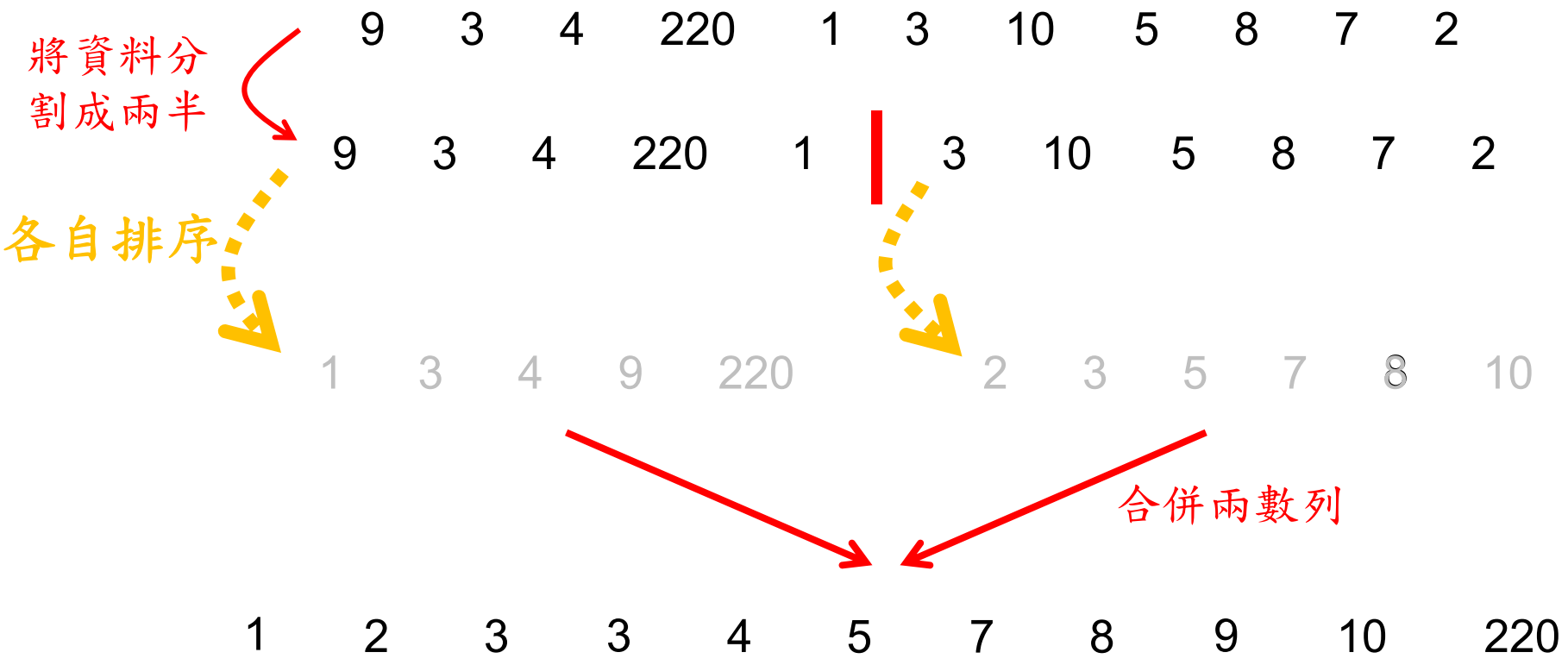
合併排序法 (Merge Sort)

- 以一個範例來說明基本的演算法



合併排序法 (Merge Sort)

- 以一個範例來說明基本的演算法



- 運算複雜度: O(n log n), 比 quicksort 的平均 O(n log n) 好
- 需要額外的 O(n) 記憶體, 比 quicksort 多

```
void mergesort(int data[], int len) {
    int len1=len/2, len2=len-len1;
    if (len <= 1) return;
    mergesort(data, len1);  mergesort(&data[len1], len2);
    merge(data, len1, &data[len1], len2);
}
```

```
void merge(int data1[], int len1, int data2[], int len2) {
    int data1_idx=0, data2_idx=0, result_idx=0;
    int *buf = (int *) malloc((len1+len2) * sizeof(int));
    while (data1_idx < len1 || data2_idx < len2) {
        if (data1_idx < len1 && data2_idx < len2)
            buf[result_idx++] = (data1[data1_idx]<=data2[data2_idx]) ?
                                        data1[data1_idx++] : data2[data2_idx++];
        else if (data1_idx < len1)
            buf[result_idx++] = data1[data1_idx++];
        else
            buf[result_idx++] = data2[data2_idx++];
    }
    for (result_idx = 0; result_idx < len1+len2; result_idx++)
        data1[result_idx] = buf[result_idx];
    free(buf);
}
```

Ugly Number

- 一個正整數如果質因數只有 2 或是 3 或是 5 的時候稱為 Ugly number;一個正整數如果質因數只有 2, 3, 5, 7 時稱為 Humble number

Ugly Number

- 一個正整數如果質因數只有 2 或是 3 或是 5 的時候稱為 Ugly number;一個正整數如果質因數只有 2, 3, 5, 7 時稱為 Humble number
- 也就是一個 Ugly number 可以寫成 $2^a3^b5^c$ (定義 a=0, b=0, c=0 也是一個 Ugly number)

Ugly Number

- 一個正整數如果質因數只有 2 或是 3 或是 5 的時候稱為 Ugly number;一個正整數如果質因數只有 2, 3, 5, 7 時稱為 Humble number
- 也就是一個 Ugly number 可以寫成 $2^a3^b5^c$ (定義 a=0, b=0, c=0 也是一個 Ugly number)
- 要測試一個正整數是不是 Ugly number 可以把所有 2, 3, 5 的因數都除掉以後看看是不是 1, 例如要看一個數字是不 2 的次方數可以用下面的迴圈把 2 的因數都除掉

Ugly Number

- 一個正整數如果質因數只有 2 或是 3 或是 5 的時候稱為 Ugly number;一個正整數如果質因數只有 2, 3, 5, 7 時稱為 Humble number
- 也就是一個 Ugly number 可以寫成 $2^a3^b5^c$ (定義 a=0, b=0, c=0 也是一個 Ugly number)
- 要測試一個正整數是不是 Ugly number 可以把所有 2, 3, 5 的因數都除掉以後看看是不是 1, 例如要看一個數字是不 2 的次方數可以用下面的迴圈把 2 的因數都除掉

```
while (n%2==0) n /= 2;
if (n==1)
    printf("n is a power of 2\n");
else
    printf("n is not a power of 2\n");
```

尋找第 n 個 Ugly Number

- Ugly number 依序排列是 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, …
  請寫一個程式尋找第 n 個 Ugly number

尋找第 n 個 Ugly Number

- Ugly number 依序排列是 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, …
  請寫一個程式尋找第 n 個 Ugly number
- 當然可以先想一下最簡單、最直覺的作法：用一個迴圈測試每一個自然數看看是不是 Ugly number

尋找第 n 個 Ugly Number

- Ugly number 依序排列是 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, …
  請寫一個程式尋找第 n 個 Ugly number
- 當然可以先想一下最簡單、最直覺的作法：用一個迴圈測試每一個自然數看看是不是 Ugly number

```
int n, count = 1, i = 2;
scanf("%d", &n);
while (count < n)
    if (isUgly(i++)) count++;
```

```
int isUgly(int n) {
    while (n%2==0)  n /= 2;
    while (n%3==0)  n /= 3;
    while (n%5==0)  n /= 5;
    return n==1;
}
```

尋找第 n 個 Ugly Number

- Ugly number 依序排列是 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, …
  請寫一個程式尋找第 n 個 Ugly number
- 當然可以先想一下最簡單、最直覺的作法：用一個迴圈測試每一個自然數看看是不是 Ugly number

```
int n, count = 1, i = 2;
scanf("%d", &n);
while (count < n)
    if (isUgly(i++)) count++;
```

```
int isUgly(int n) {
    while (n%2==0)  n /= 2;
    while (n%3==0)  n /= 3;
    while (n%5==0)  n /= 5;
    return n==1;
}
```

- 這個作法很快地會發現兩個問題

尋找第 n 個 Ugly Number

- Ugly number 依序排列是 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, …
  請寫一個程式尋找第 n 個 Ugly number
- 當然可以先想一下最簡單、最直覺的作法：用一個迴圈測試每一個自然數看看是不是 Ugly number

```
int n, count = 1, i = 2;
scanf("%d", &n);
while (count < n)
    if (isUgly(i++)) count++;
```

```
int isUgly(int n) {
    while (n%2==0)  n /= 2;
    while (n%3==0)  n /= 3;
    while (n%5==0)  n /= 5;
    return n==1;
}
```

- 這個作法很快地會發現兩個問題

1. i 的數值比 count 大很多很多, 如果預期的 n 值在 100000, 那麼 i 一定要宣告成 long long, 否則很快就出現溢位

尋找第 n 個 Ugly Number

- Ugly number 依序排列是 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, …
  請寫一個程式尋找第 n 個 Ugly number
- 當然可以先想一下最簡單、最直覺的作法：用一個迴圈測試每一個自然數看看是不是 Ugly number

```
int n, count = 1, i = 2;
scanf("%d", &n);
while (count < n)
    if (isUgly(i++)) count++;
```

```
int isUgly(int n) {
    while (n%2==0)  n /= 2;
    while (n%3==0)  n /= 3;
    while (n%5==0)  n /= 5;
    return n==1;
}
```

- 這個作法很快地會發現兩個問題
  1. i 的數值比 count 大很多很多, 如果預期的 n 值在 100000, 那麼 i 一定要宣告成 long long, 否則很快就出現溢位
  2. n 值變大的時候, 程式執行時間很久

尋找第 n 個 Ugly Number

- 能不能加快尋找的速度?

尋找第 n 個 Ugly Number

- 能不能加快尋找的速度?
  - 執行前面這個程式時看到的問題是自然數中不是 Ugly number 的數字太多了, 每個自然數都測試太慢了

尋找第 n 個 Ugly Number

- 能不能加快尋找的速度?
  - 執行前面這個程式時看到的問題是自然數中不是 Ugly number 的數字太多了, 每個自然數都測試太慢了
  - 如果可以只列舉出 Ugly number 的話, 應該就快得多

尋找第 n 個 Ugly Number

- 能不能加快尋找的速度?
  - 執行前面這個程式時看到的問題是自然數中不是 Ugly number 的數字太多了, 每個自然數都測試太慢了
  - 如果可以只列舉出 Ugly number 的話, 應該就快得多
  - 由Ugly number $x=2^a3^b5^c$ 的定義可以看到一種列出所有 Ugly number 的方法是列舉 (a,b,c) 的數值 (0,0,0), (0,0,1), (0,1,0), (1,0,0), (0,0,2), (0,1,1), (1,0,1), (1,1,0), (0,2,0), (2,0,0), … 不過列舉的順序不太好決定, 列舉出來的 Ugly number 也不見得是由小排到大的

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, ...　　注意: **1.** 順序不對 **2.** 有重複

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, …　　注意: **1.** 順序不對 **2.** 有重複

1

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, …　　注意: **1.** 順序不對 **2.** 有重複

1

2

3

5

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, …　　注意: **1.** 順序不對 **2.** 有重複

1 2

4

3 6

5 10

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, …　　注意: **1.** 順序不對 **2.** 有重複

1 2 3

4 6

6 9

5 10 15

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, …　　注意: **1.** 順序不對 **2.** 有重複

1 2 3 4

6 8

6 9 12

5 10 15 20

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, …　　注意: **1.** 順序不對 **2.** 有重複

1 2 3 4 5

6 8 10

6 9 12 15

10 15 20 25

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, …　　注意: **1.** 順序不對 **2.** 有重複

1 2 3 4 5 6

8 10 12

6 9 12 15 18

10 15 20 25 30

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, …　　注意: **1.** 順序不對 **2.** 有重複

1 2 3 4 5 6

8 10 12

9 12 15 18

10 15 20 25 30

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, …　　注意: **1.** 順序不對 **2.** 有重複

1 2 3 4 5 6 8

10 12 16

9 12 15 18 24

10 15 20 25 30 40

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, ...　　注意: **1.** 順序不對 **2.** 有重複

1 2 3 4 5 6 8 9

10 12 16 18

12 15 18 24 27

10 15 20 25 30 40 45

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, …　　注意: **1.** 順序不對 **2.** 有重複

1 2 3 4 5 6 8 9 10

12 16 18 20

12 15 18 24 27 30

10 15 20 25 30 40 45 50

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, …　　注意: **1.** 順序不對 **2.** 有重複

1 2 3 4 5 6 8 9 10

12 16 18 20

12 15 18 24 27 30

15 20 25 30 40 45 50

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, …　　注意: **1.** 順序不對 **2.** 有重複

1 2 3 4 5 6 8 9 10 12

16 18 20 24

12 15 18 24 27 30 36

15 20 25 30 40 45 50 60

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, ...　　注意: **1.** 順序不對 **2.** 有重複

1 2 3 4 5 6 8 9 10 12

16 18 20 24

15 18 24 27 30 36

15 20 25 30 40 45 50 60

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, …　　注意: **1.** 順序不對 **2.** 有重複

1 2 3 4 5 6 8 9 10 12 15

16 18 20 24 30

18 24 27 30 36 45

15 20 25 30 40 45 50 60 75

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, …　　注意: **1.** 順序不對 **2.** 有重複

1 2 3 4 5 6 8 9 10 12 15

16 18 20 24 30

18 24 27 30 36 45

20 25 30 40 45 50 60 75

尋找第 n 個 Ugly Number

- 加快尋找的速度?
  - 另一種看法是每一個 Ugly number x 都定義出三個大於它的 Ugly number: {2x, 3x, 5x}, 這三個數字雖然不見得是緊接著 x 的三個 Ugly number, 也不見得是連續的三個 Ugly number, 但是這些集合的聯集的確是所有的 Ugly number {2,3,5}, {4,6,10}, {6,9,15}, {8,12,20}, {10,15,25}, {12,18, 30}, {16,24,40}, …　　注意: **1.** 順序不對 **2.** 有重複

1 2 3 4 5 6 8 9 10 12 15 16 …

18 20 24 30 32 …

18 24 27 30 36 45 48 …

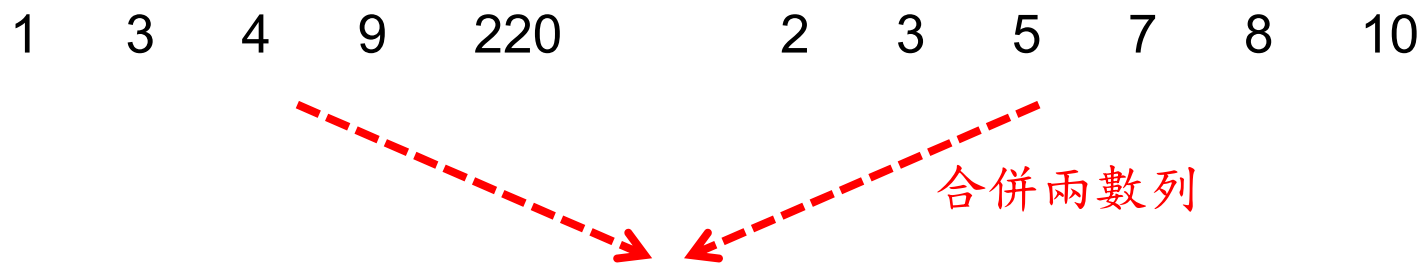
20 25 30 40 45 50 60 75 80 …

合併兩個數列

合併兩個數列

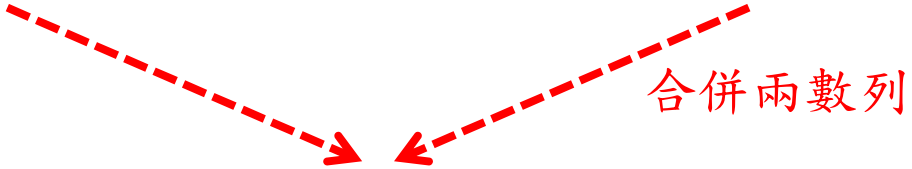
1　3　4　9　220　　2　3　5　7　8　10

合併兩個數列



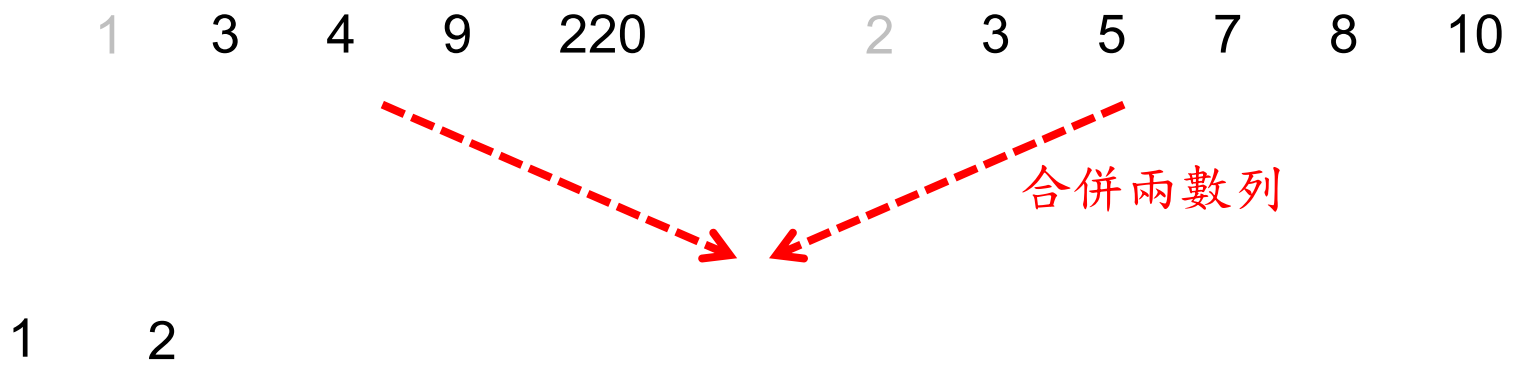
合併兩個數列

1 3 4 9 220 2 3 5 7 8 10

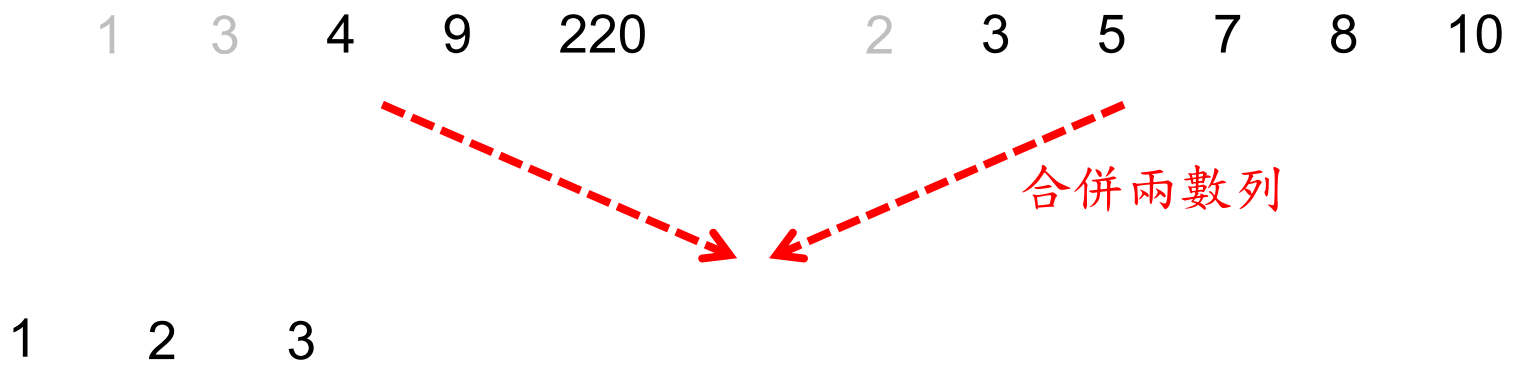


1

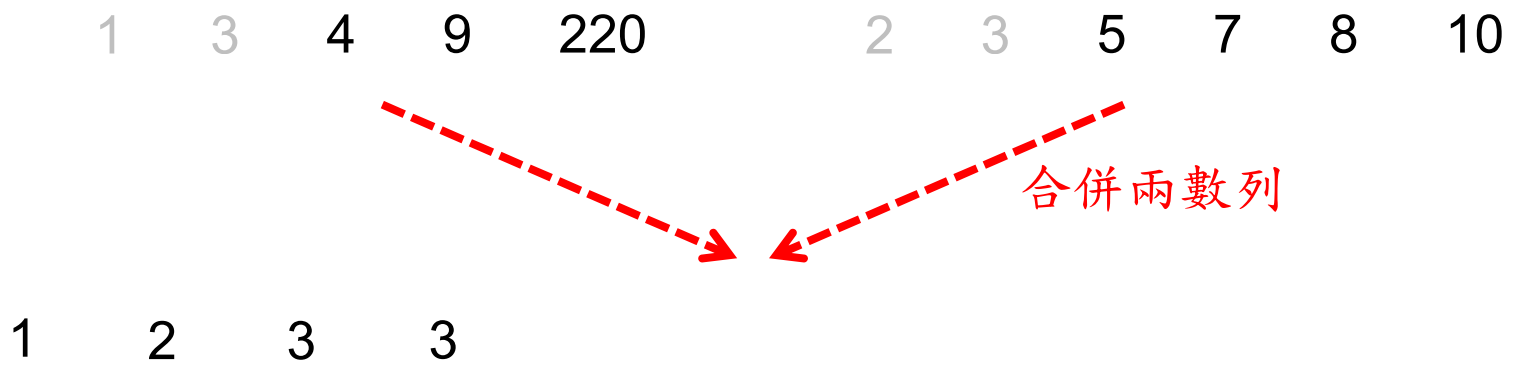
合併兩個數列



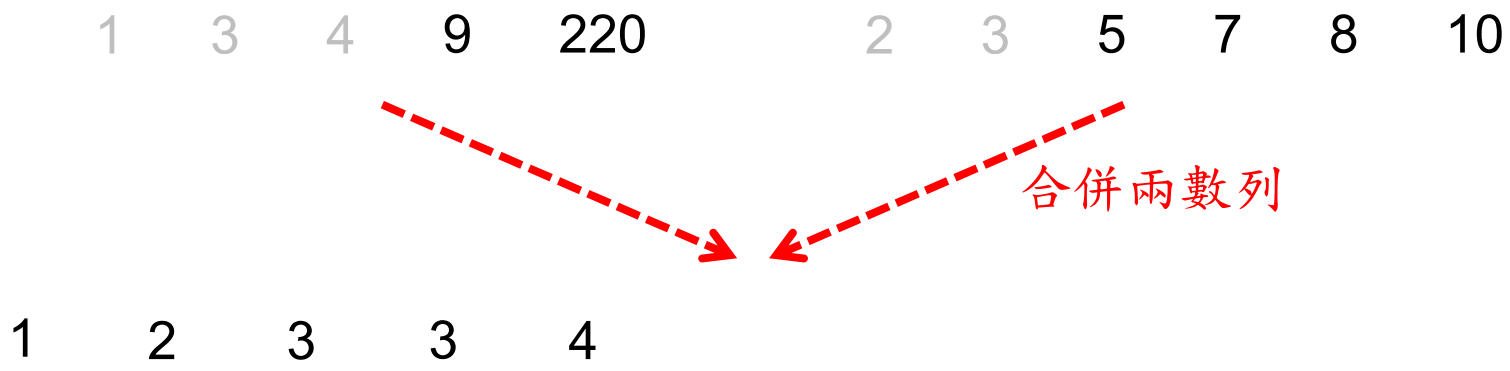
合併兩個數列



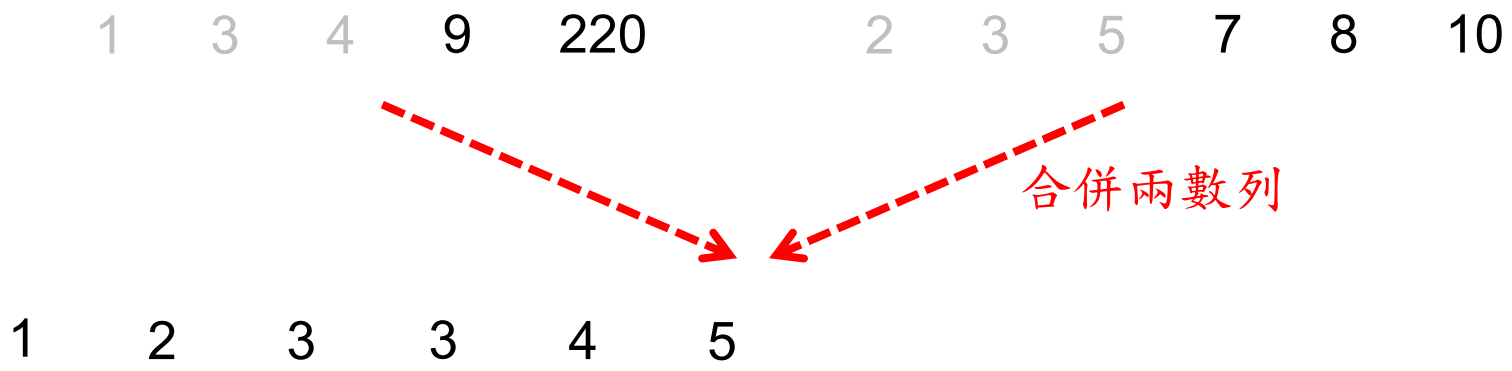
合併兩個數列



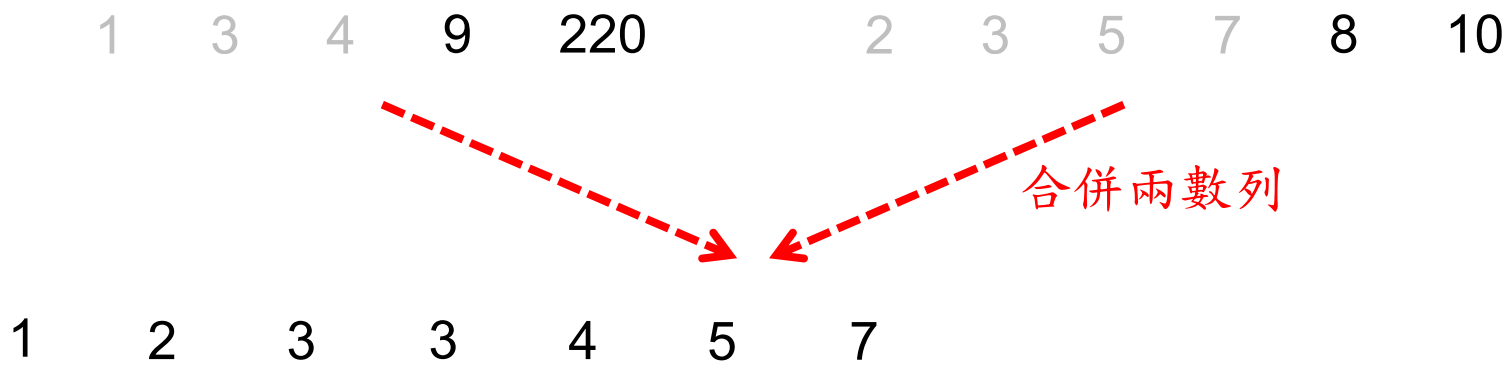
合併兩個數列



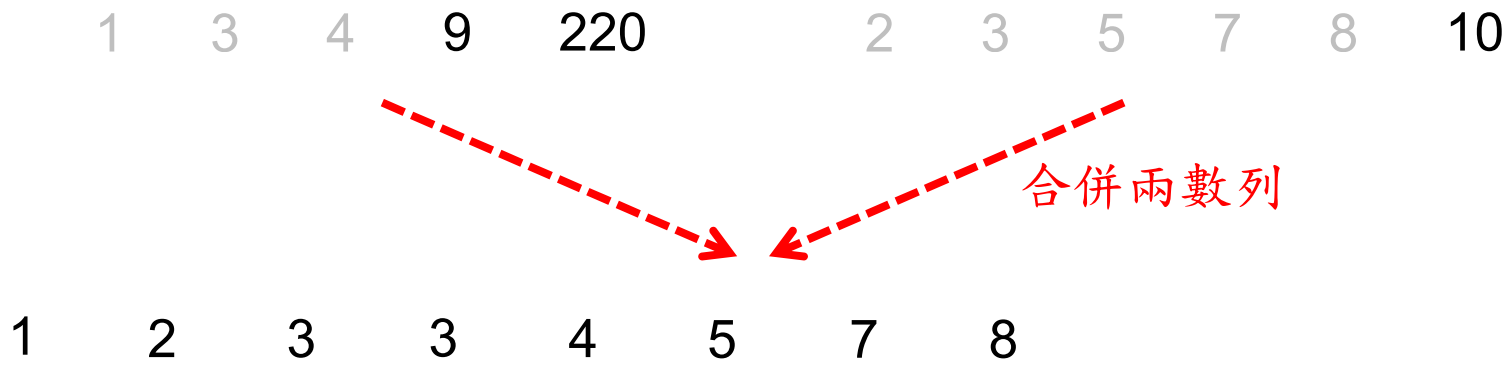
合併兩個數列



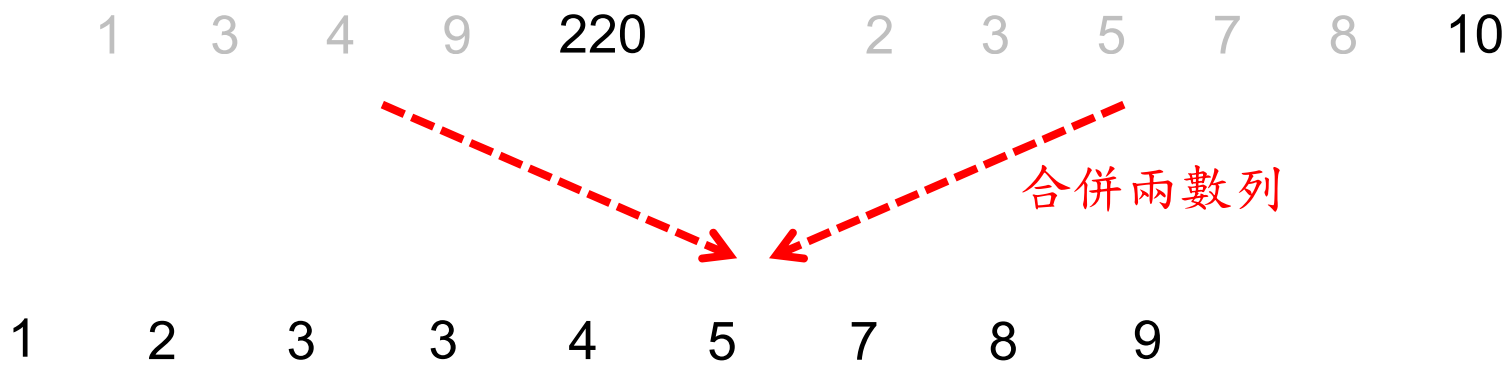
合併兩個數列



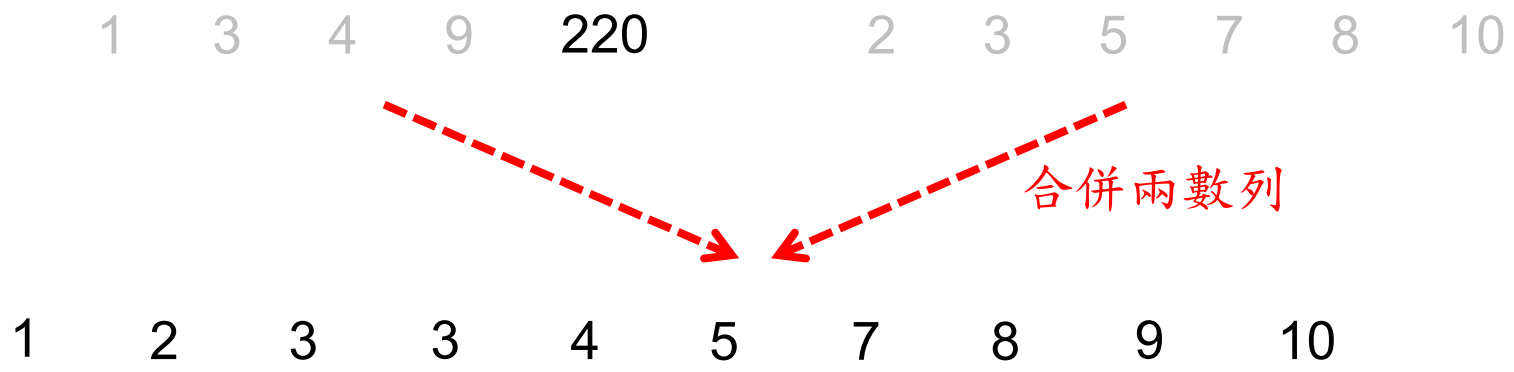
合併兩個數列



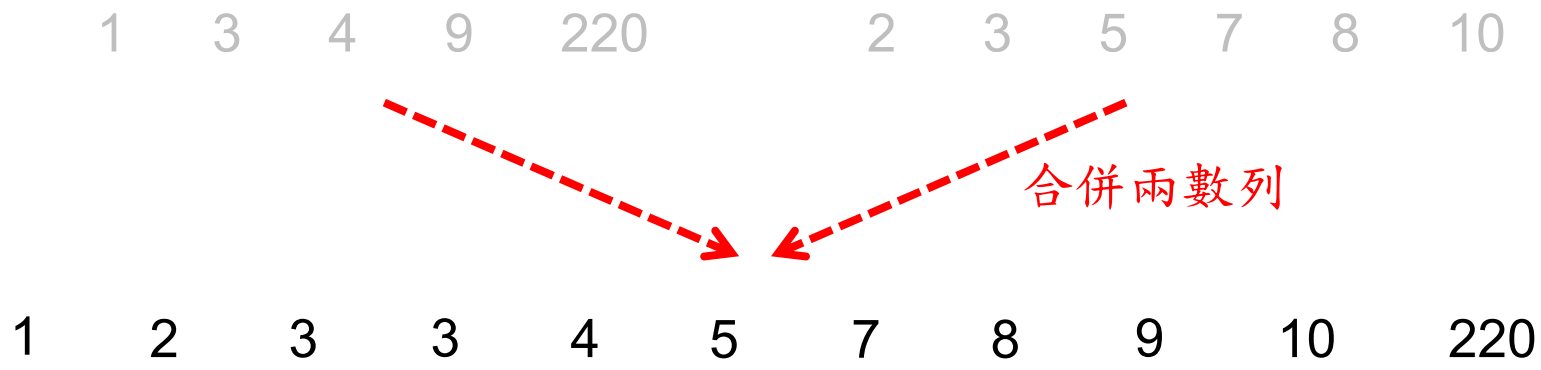
合併兩個數列



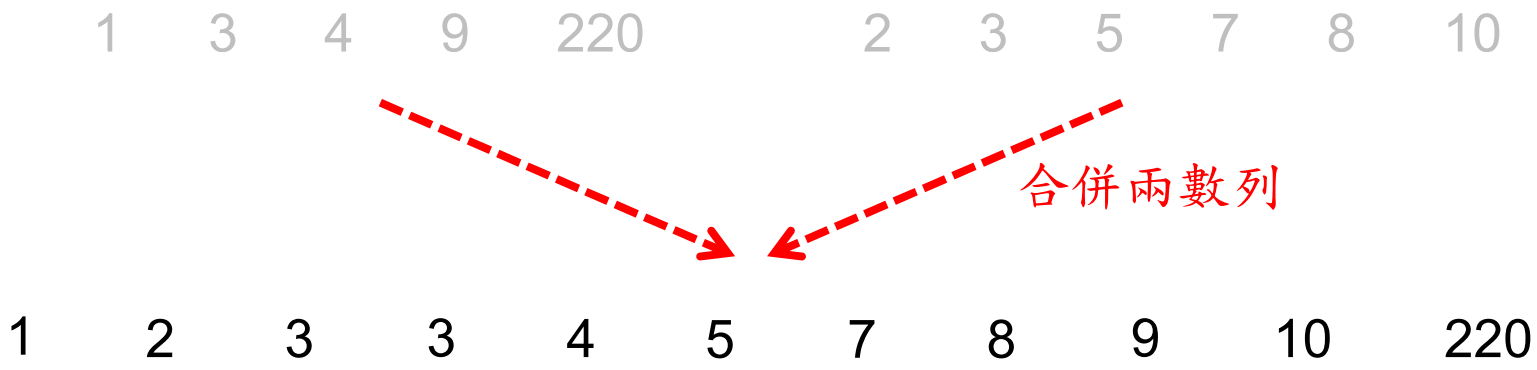
合併兩個數列



合併兩個數列



合併兩個數列

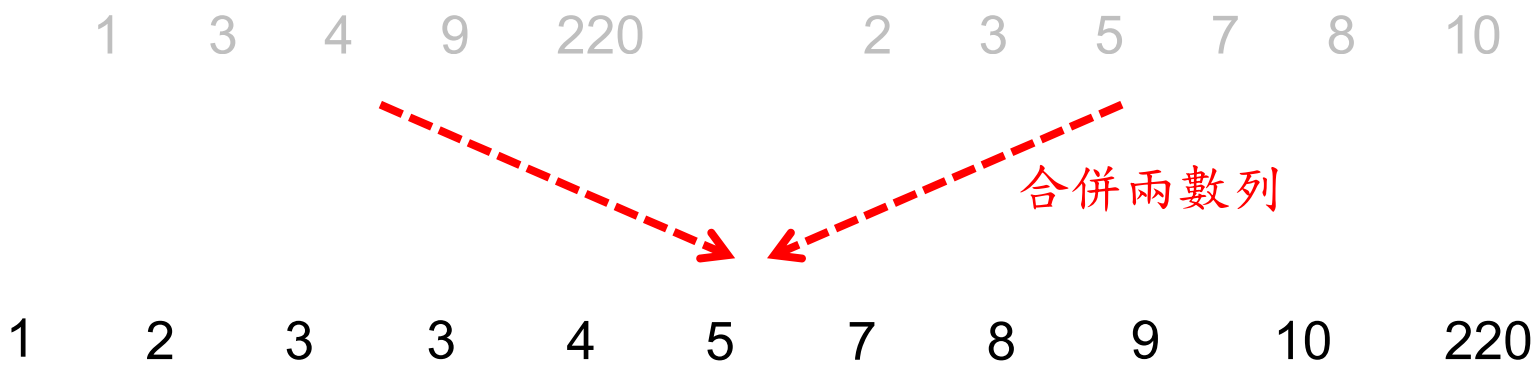



```
void merge(int data1[], int len1, int data2[], int len2, int rst[]) {
    int idx1=0, idx2=0, rst_idx=0;
    while (idx1 < len1 || idx2 < len2) {




    }
}
```

合併兩個數列

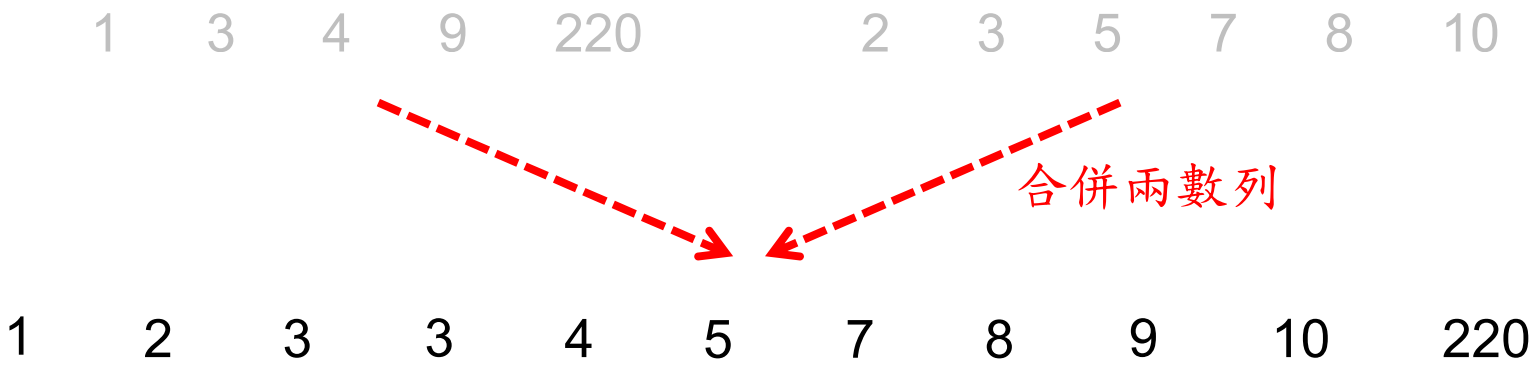



```
void merge(int data1[], int len1, int data2[], int len2, int rst[]) {
    int idx1=0, idx2=0, rst_idx=0;
    while (idx1 < len1 || idx2 < len2) {
        if (idx1 < len1 && idx2 < len2)
            rst[rst_idx++] = (data1[idx1]<=data2[idx2]) ? data1[idx1++] : data2[idx2++];




    }
}
```

合併兩個數列




```
void merge(int data1[], int len1, int data2[], int len2, int rst[]) {
    int idx1=0, idx2=0, rst_idx=0;
    while (idx1 < len1 || idx2 < len2) {
        if (idx1 < len1 && idx2 < len2)
            rst[rst_idx++] = (data1[idx1]<=data2[idx2]) ? data1[idx1++] : data2[idx2++];
        else if (idx1 < len1)
            rst[rst_idx++] = data1[idx1++];
        else
            rst[rst_idx++] = data2[idx2++];
    }
}
```

尋找第 n 個 Ugly Number

- 需要以陣列紀錄接下來有哪些可能性, 由其中找到下一個 Ugly number, 因為若 a<b 則 2a<2b, 3a<3b, 5a<5b, 所以讓這些數字排成三隊, 每次只需要考慮排在最前面的三個數字就可以了

尋找第 n 個 Ugly Number

- 需要以陣列紀錄接下來有哪些可能性, 由其中找到下一個 Ugly number, 因為若 a<b 則 2a<2b, 3a<3b, 5a<5b, 所以讓這些數字排成三隊, 每次只需要考慮排在最前面的三個數字就可以了

2 4 6 8 10 12 16 18 20 24 30 32 …

尋找第 n 個 Ugly Number

- 需要以陣列紀錄接下來有哪些可能性, 由其中找到下一個 Ugly number, 因為若 a<b 則 2a<2b, 3a<3b, 5a<5b, 所以讓這些數字排成三隊, 每次只需要考慮排在最前面的三個數字就可以了

2 4 6 8 10 12 16 18 20 24 30 32 …

3 6 9 12 15 18 24 27 30 36 45 48 …

尋找第 n 個 Ugly Number

- 需要以陣列紀錄接下來有哪些可能性, 由其中找到下一個 Ugly number, 因為若 a<b 則 2a<2b, 3a<3b, 5a<5b, 所以讓這些數字排成三隊, 每次只需要考慮排在最前面的三個數字就可以了

2 4 6 8 10 12 16 18 20 24 30 32 …

3 6 9 12 15 18 24 27 30 36 45 48 …

5 10 15 20 25 30 40 45 50 60 75 80 …

尋找第 n 個 Ugly Number

- 需要以陣列紀錄接下來有哪些可能性, 由其中找到下一個 Ugly number, 因為若 a<b 則 2a<2b, 3a<3b, 5a<5b, 所以讓這些數字排成三隊, 每次只需要考慮排在最前面的三個數字就可以了

2 4 6 8 10 12 16 18 20 24 30 32 …

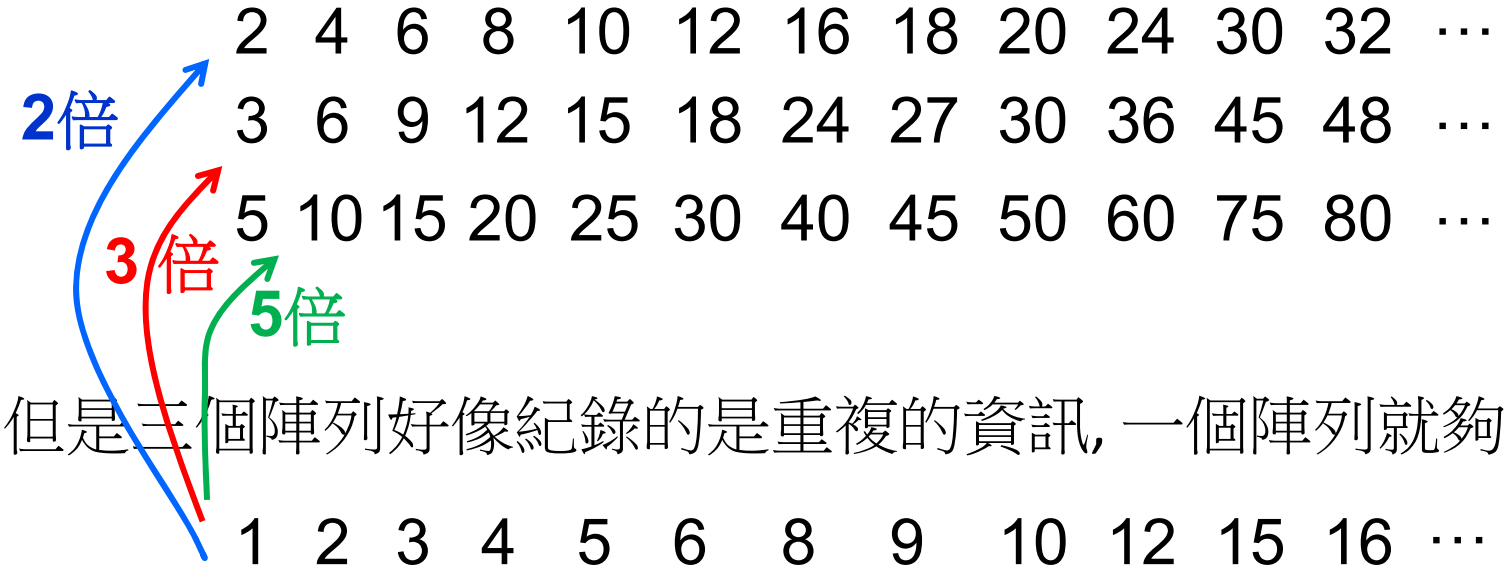
3 6 9 12 15 18 24 27 30 36 45 48 …

5 10 15 20 25 30 40 45 50 60 75 80 …

- 但是三個陣列好像紀錄的是重複的資訊, 一個陣列就夠了

尋找第 n 個 Ugly Number

- 需要以陣列紀錄接下來有哪些可能性, 由其中找到下一個 Ugly number, 因為若 a<b 則 2a<2b, 3a<3b, 5a<5b, 所以讓這些數字排成三隊, 每次只需要考慮排在最前面的三個數字就可以了



- 但是三個陣列好像紀錄的是重複的資訊, 一個陣列就夠了

1 2 3 4 5 6 8 9 10 12 15 16 …

就是所有小於 n 的 Ugly number 的陣列

尋找第 n 個 Ugly Number

尋找第 n 個 Ugly Number

```
int n, p2=1, p3=1, p5=1;
```

尋找第 n 個 Ugly Number

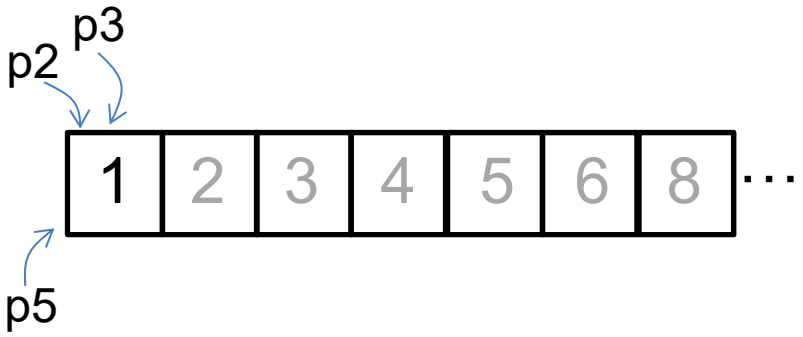
```
int n, p2=1, p3=1, p5=1;
long long int i=1, x2, x3, x5;
```

尋找第 n 個 Ugly Number

```
int n, p2=1, p3=1, p5=1;
long long int i=1, x2, x3, x5;
long long int a[10001] = {0,1};
```

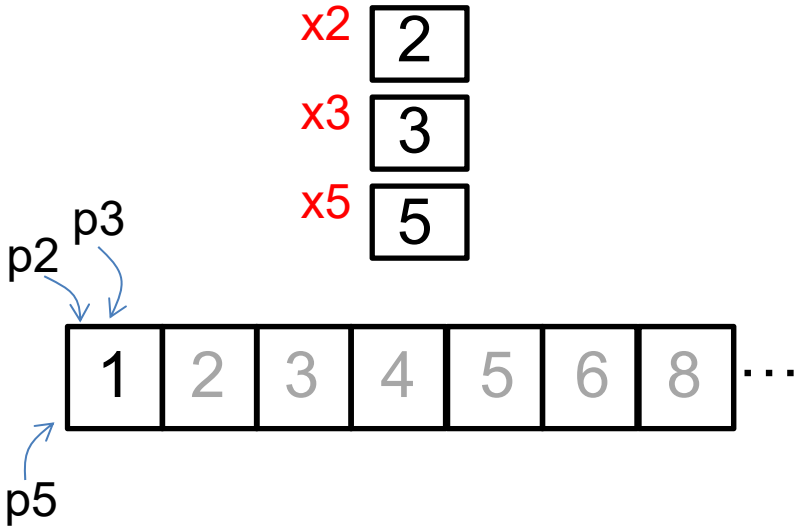
尋找第 n 個 Ugly Number

```
int n, p2=1, p3=1, p5=1;
long long int i=1, x2, x3, x5;
long long int a[10001] = {0,1};
scanf("%d", &n);
```



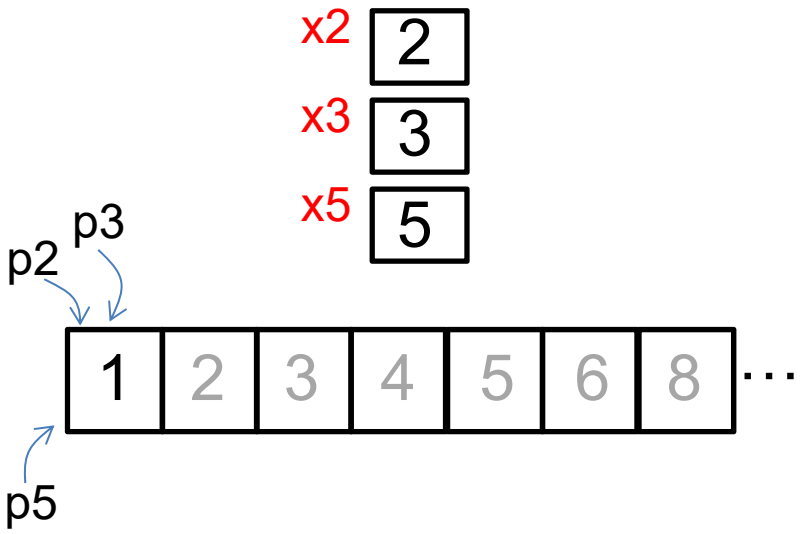
尋找第 n 個 Ugly Number

```
int n, p2=1, p3=1, p5=1;
long long int i=1, x2, x3, x5;
long long int a[10001] = {0,1};
scanf("%d", &n);
x2 = a[p2]*2;
x3 = a[p3]*3;
x5 = a[p5]*5;
```



尋找第 n 個 Ugly Number

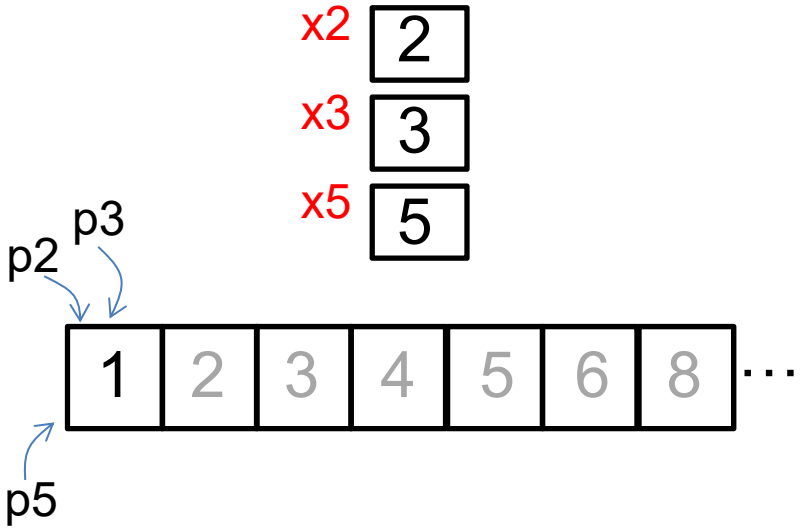
```
int n, p2=1, p3=1, p5=1;
long long int i=1, x2, x3, x5;
long long int a[10001] = {0,1};
scanf("%d", &n);
x2 = a[p2]*2;
x3 = a[p3]*3;
x5 = a[p5]*5;
```




```
while (++i<=n) {
}
printf("%lld\n", a[n]);
```

尋找第 n 個 Ugly Number

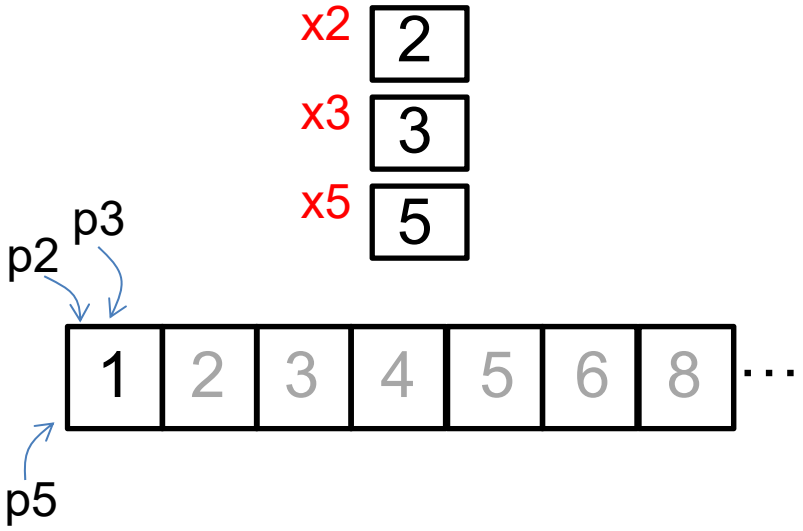
```
int n, p2=1, p3=1, p5=1;
long long int i=1, x2, x3, x5;
long long int a[10001] = {0,1};
scanf("%d", &n);
x2 = a[p2]*2;
x3 = a[p3]*3;
x5 = a[p5]*5;
```




```
while (++i<=n) {
    if (x2<x3&&x2<x5) {
        a[i] = x2; x2 = a[++p2]*2;
        while ((x2==x3) || (x2==x5))
          x2 = a[++p2]*2;
    }
}
printf("%lld\n", a[n]);
```

尋找第 n 個 Ugly Number

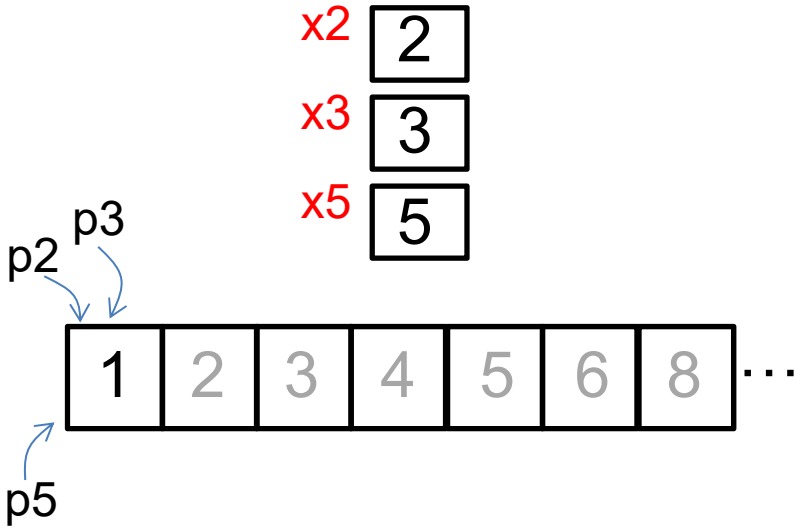
```
int n, p2=1, p3=1, p5=1;
long long int i=1, x2, x3, x5;
long long int a[10001] = {0,1};
scanf("%d", &n);
x2 = a[p2]*2;
x3 = a[p3]*3;
x5 = a[p5]*5;
```




```
while (++i<=n) {
    if (x2<x3&&x2<x5) {
        a[i] = x2; x2 = a[++p2]*2;
        while ((x2==x3) || (x2==x5))
          x2 = a[++p2]*2;
    }
    else if (x3<x2&&x3<x5) {
        a[i] = x3; x3 = a[++p3]*3;
        while ((x3==x2)||(x3==x5))
            x3 = a[++p3]*3;
    }
}
printf("%lld\n", a[n]);
```

尋找第 n 個 Ugly Number

```
int n, p2=1, p3=1, p5=1;
long long int i=1, x2, x3, x5;
long long int a[10001] = {0,1};
scanf("%d", &n);
x2 = a[p2]*2;
x3 = a[p3]*3;
x5 = a[p5]*5;
```




```
while (++i<=n) {
    if (x2<x3&&x2<x5) {
        a[i] = x2; x2 = a[++p2]*2;
        while ((x2==x3) || (x2==x5))
            x2 = a[++p2]*2;
    }
    else if (x3<x2&&x3<x5) {
        a[i] = x3; x3 = a[++p3]*3;
        while ((x3==x2)||(x3==x5))
            x3 = a[++p3]*3;
    }
    else if (x5<x2&&x5<x3) {
        a[i] = x5; x5 = a[++p5]*5;
        while ((x5==x2)||(x5==x3))
            x5 = a[++p5]*5;
    }
}
printf("%lld\n", a[n]);
```

ZJ a241

有多少個 1/x 是有限小數

- 若 x 為整數且 1<x≤n，請問有幾個 1/x 是有限小數 (finite decimal)? (實數還包括循環小數 cyclic decimal 以及無理數)

ZJ a241

有多少個 1/x 是有限小數

- 若 x 為整數且 $1<x\le n$，請問有幾個 1/x 是有限小數 (finite decimal)? (實數還包括循環小數 cyclic decimal 以及無理數)
- m 位的有限小數可以寫成 $y/10^m$，其中 y 不是 10 的整數倍

ZJ a241

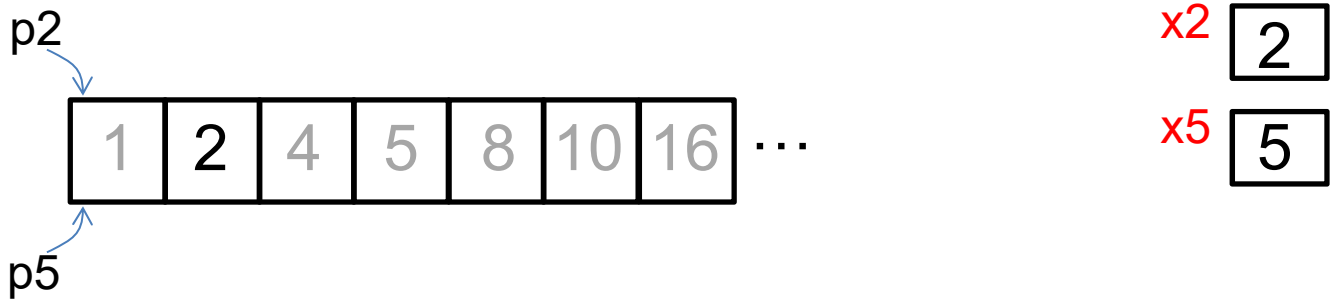
有多少個 1/x 是有限小數

- 若 x 為整數且 $1<x\le n$，請問有幾個 1/x 是有限小數 (finite decimal)? (實數還包括循環小數 cyclic decimal 以及無理數)
- m 位的有限小數可以寫成 $y/10^m$，其中 y 不是 10 的整數倍
- x, y 為整數且 $1/x = y/10^m \Rightarrow x\,y = 2^m\,5^m$，亦即 x, y 都只能有 2 和 5 的質因數，每一個 $x=2^a\,5^b$ 所對應的 $1/x=2^{m-a}\,5^{m-b}/2^m\,5^m$, $m=\max(a,b)$, $y=2^{m-a}\,5^{m-b}$ 都是有限小數

ZJ a241

有多少個 1/x 是有限小數

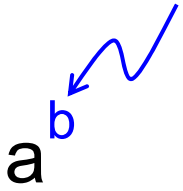
- 若 x 為整數且 $1<x\le n$，請問有幾個 1/x 是有限小數 (finite decimal)? (實數還包括循環小數 cyclic decimal 以及無理數)
- m 位的有限小數可以寫成 $y/10^m$，其中 y 不是 10 的整數倍
- x, y 為整數且 $1/x = y/10^m \Rightarrow x\,y = 2^m\,5^m$，亦即 x, y 都只能有 2 和 5 的質因數，每一個 $x=2^a\,5^b$ 所對應的 $1/x=2^{m-a}\,5^{m-b}/2^m\,5^m$, $m=\max(a,b)$, $y=2^{m-a}\,5^{m-b}$ 都是有限小數
- 尋找可能的 x 基本上和尋找 Ugly Number 的方法是一樣的，數量事實上很少， $1\sim10^8$ 之間只有 171 個



LC315 Count of Smaller Numbers after Self

O(n^2) solution is trivial, the goal is to devise an O(n logn) solution.

希望得到的結果
(小於 a 的個數)



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

逆序, x > y
(backward sequence,
inverted sequence)

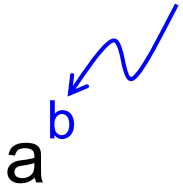
(7,3), (3,2), … 逆序對

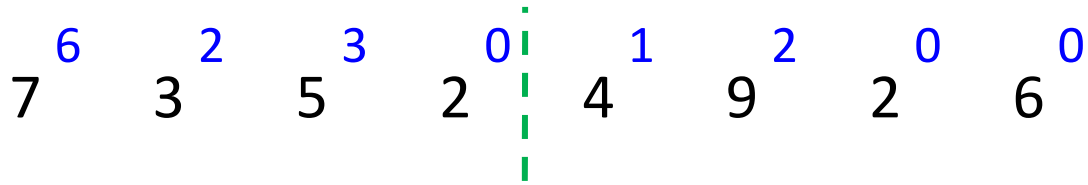
逆序數: 14
序列中逆序對的數目

LC315 Count of Smaller Numbers after Self

O(n^2) solution is trivial, the goal is to devise an O(n logn) solution.

希望得到的結果
(小於 a 的個數)





逆序, x > y
(backward sequence,
inverted sequence)

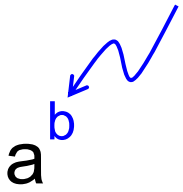
(7,3), (3,2), … 逆序對

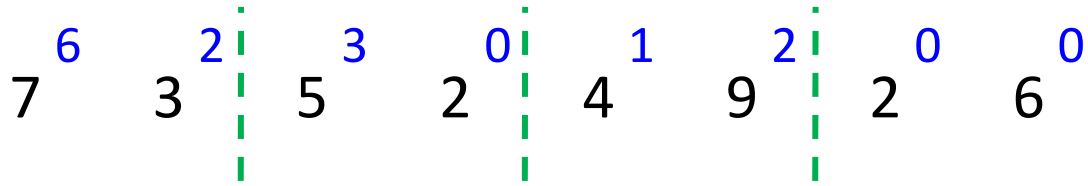
逆序數: 14
序列中逆序對的數目

LC315 Count of Smaller Numbers after Self

O(n^2) solution is trivial, the goal is to devise an O(n logn) solution.

希望得到的結果
(小於 a 的個數)





逆序, x > y
(backward sequence,
inverted sequence)

(7,3), (3,2), … 逆序對

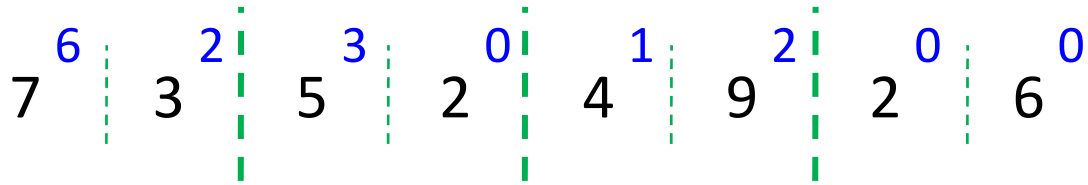
逆序數: 14
序列中逆序對的數目

LC315 Count of Smaller Numbers after Self

O(n^2) solution is trivial, the goal is to devise an O(n logn) solution.

希望得到的結果
(小於 a 的個數)

a b



逆序, x > y
(backward sequence,
inverted sequence)

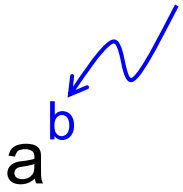
(7,3), (3,2), … 逆序對

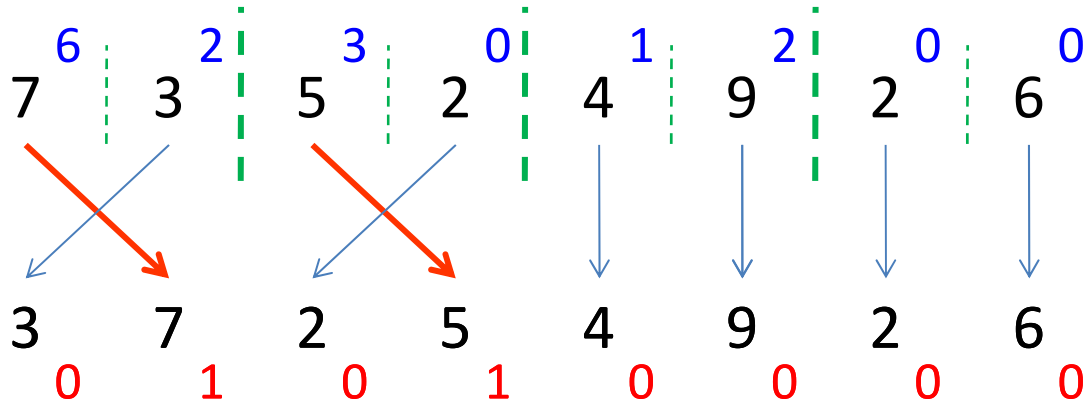
逆序數: 14
序列中逆序對的數目

LC315 Count of Smaller Numbers after Self

O(n^2) solution is trivial, the goal is to devise an O(n logn) solution.

希望得到的結果
(小於 a 的個數)





逆序, x > y
(backward sequence,
inverted sequence)

(7,3), (3,2), … 逆序對

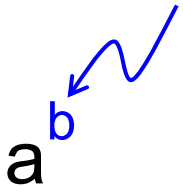
逆序數: 14
序列中逆序對的數目

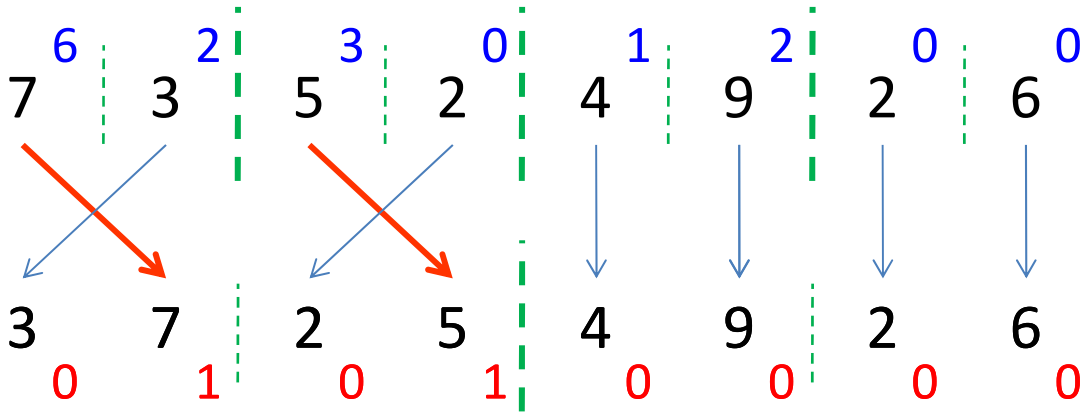
Mergesort 過程中往右移 **k** 位的意思是表示右邊有 **k** 個比較小的數字

LC315 Count of Smaller Numbers after Self

O(n^2) solution is trivial, the goal is to devise an O(n logn) solution.

希望得到的結果
(小於 a 的個數)





逆序, x > y
(backward sequence,
inverted sequence)

(7,3), (3,2), … 逆序對

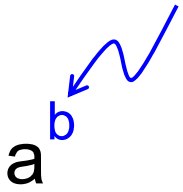
逆序數: 14
序列中逆序對的數目

Mergesort 過程中往右移 **k** 位的意思是表示右邊有 **k** 個比較小的數字

LC315 Count of Smaller Numbers after Self

$O(n^2)$ solution is trivial, the goal is to devise an O(n logn) solution.

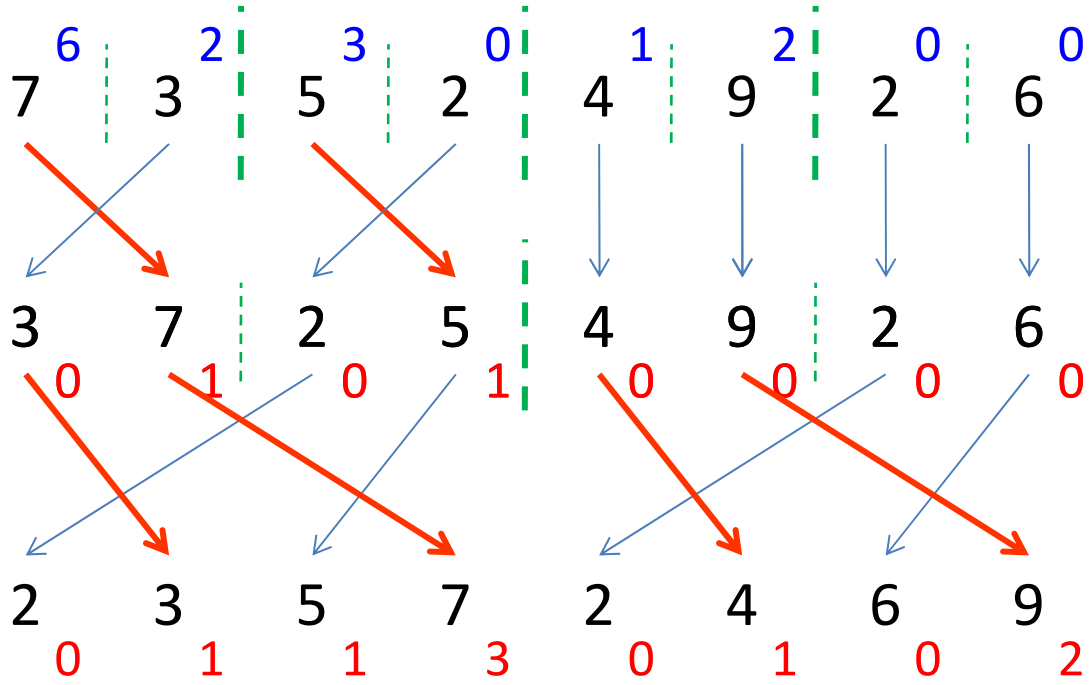
希望得到的結果
(小於 a 的個數)



逆序, x > y
(backward sequence,
inverted sequence)

(7,3), (3,2), … 逆序對

逆序數: 14
序列中逆序對的數目



Mergesort 過程中往右移 **k** 位的意思是表示右邊有 **k** 個比較小的數字

LC315 Count of Smaller Numbers after Self

$O(n^2)$ solution is trivial, the goal is to devise an O(n logn) solution.

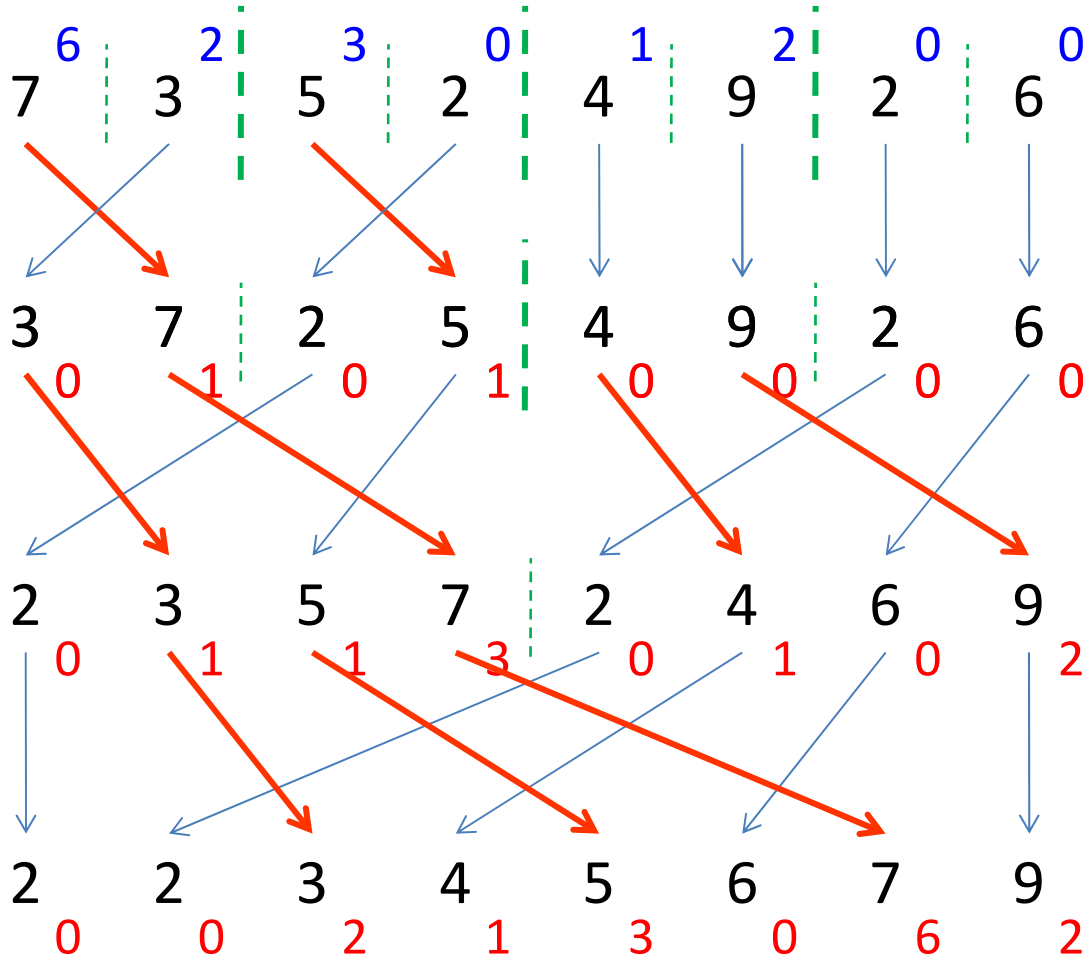
希望得到的結果
(小於 a 的個數)

a^b

逆序, x > y
(backward sequence,
inverted sequence)

(7,3), (3,2), … 逆序對

逆序數: 14
序列中逆序對的數目



Mergesort 過程中往右移 **k** 位的意思是表示右邊有 **k** 個比較小的數字

LC315 Count of Smaller Numbers after Self

$O(n^2)$ solution is trivial, the goal is to devise an O(n logn) solution.

希望得到的結果
(小於 a 的個數)

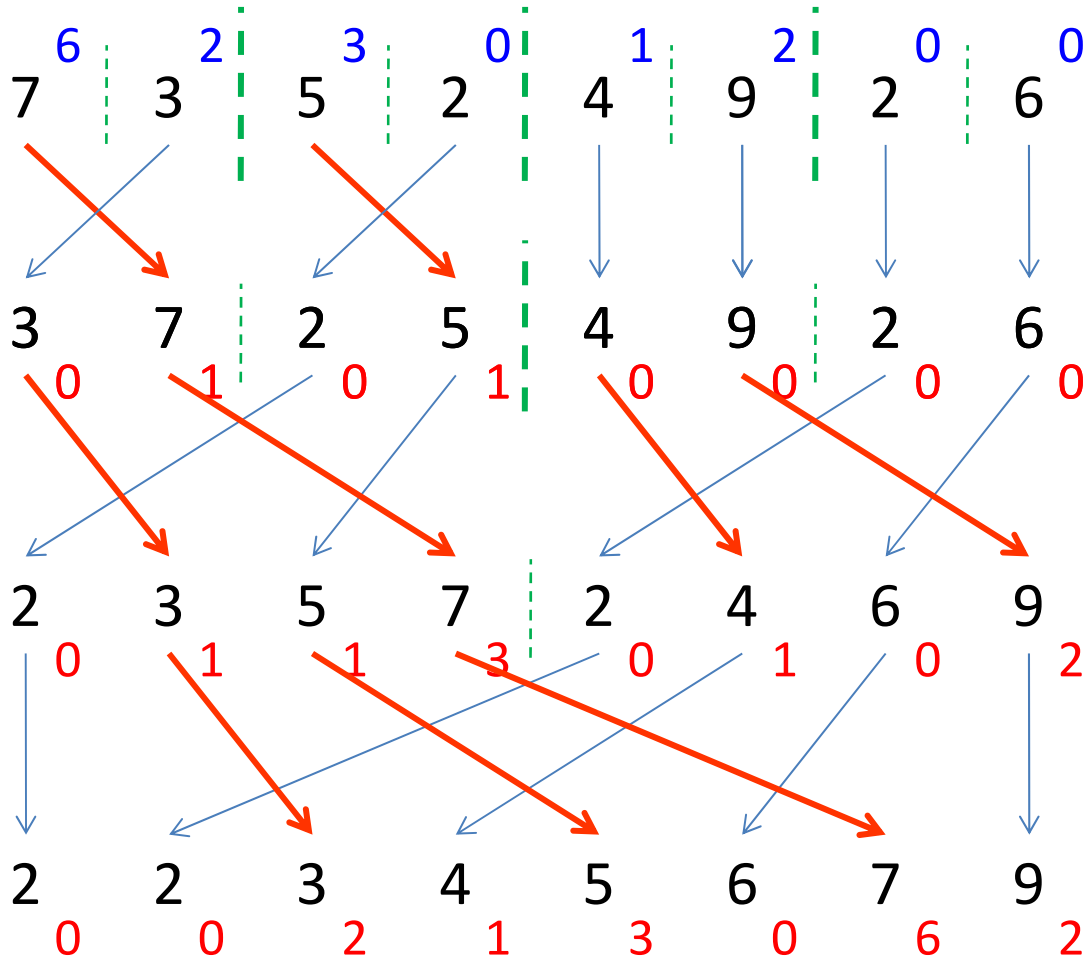
a^b

逆序, x > y
(backward sequence,
inverted sequence)

(7,3), (3,2), … 逆序對

逆序數: 14
序列中逆序對的數目

UVa10810, ZJ b684



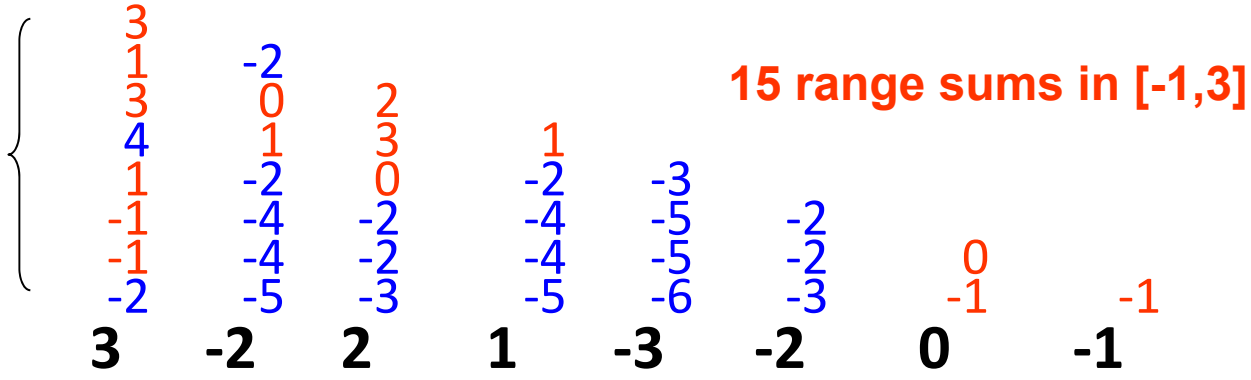
Mergesort 過程中往右移 **k** 位的意思是表示右邊有 **k** 個比較小的數字

LC327 Count of Range Sum

3 -2 2 1 -3 -2 0 -1

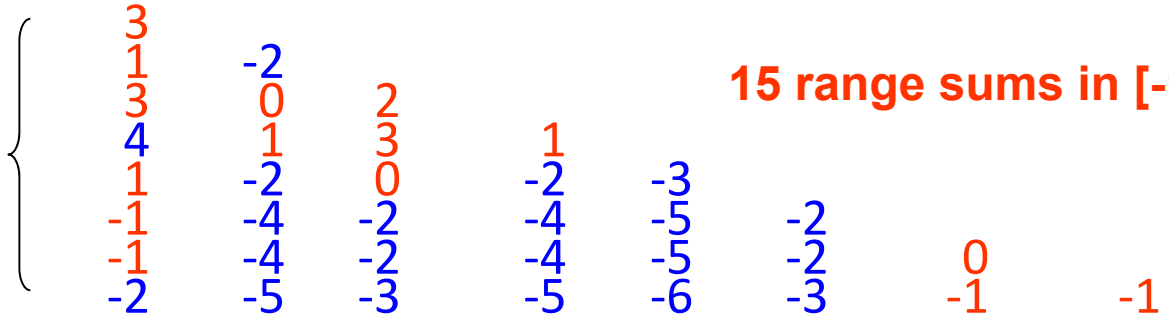
LC327 Count of Range Sum

$O(n^2)$ solution is trivial, the goal is to devise an O(n logn) solution.



LC327 Count of Range Sum

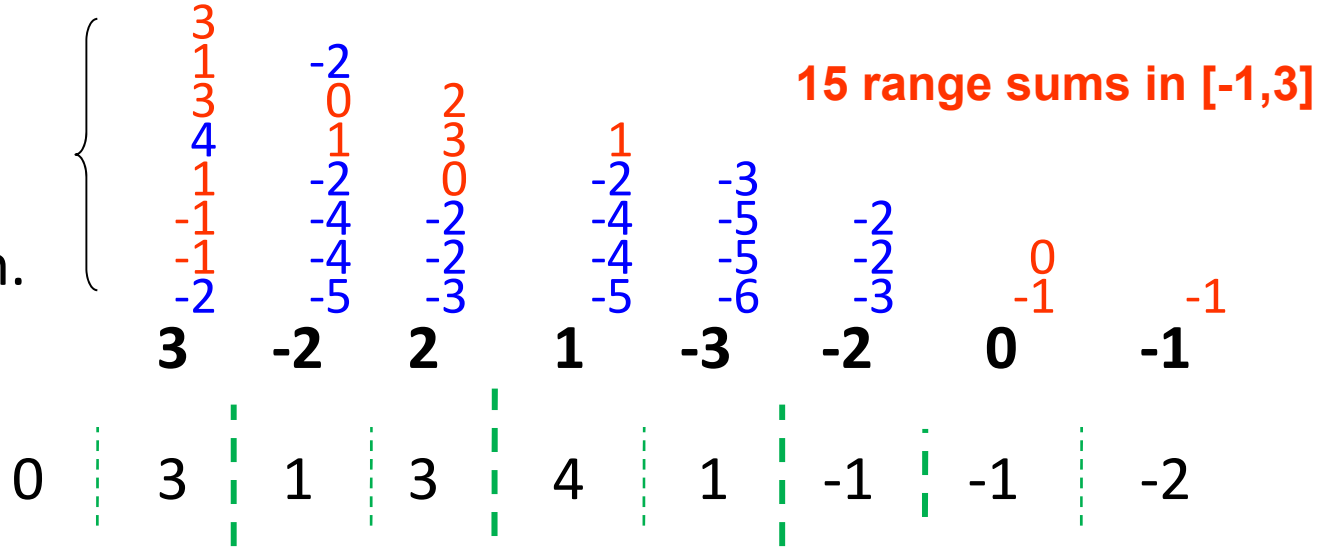
$O(n^2)$ solution is trivial, the goal is to devise an $O(n \log n)$ solution.



	3	**-2**	**2**	**1**	**-3**	**-2**	**0**	**-1**
0	3	1	3	4	1	-1	-1	-2

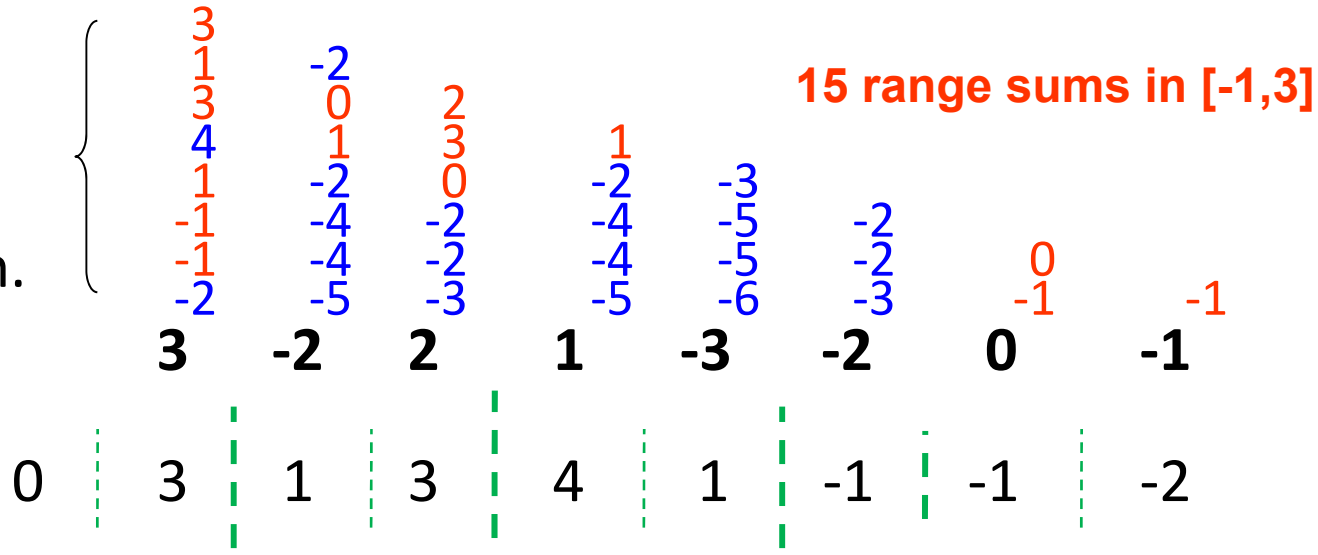
LC327 Count of Range Sum

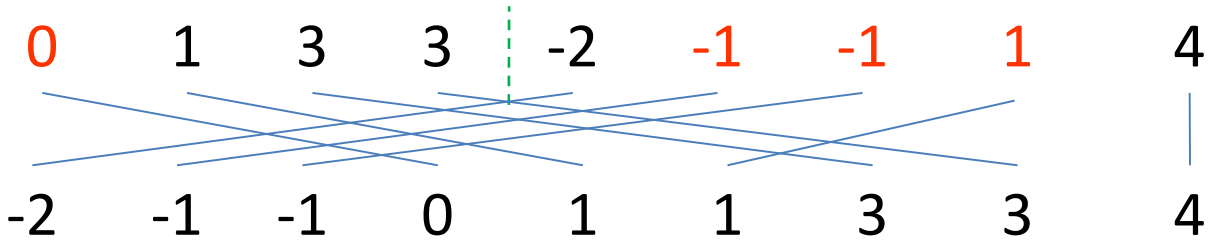
$O(n^2)$ solution is trivial, the goal is to devise an $O(n \log n)$ solution.



LC327 Count of Range Sum

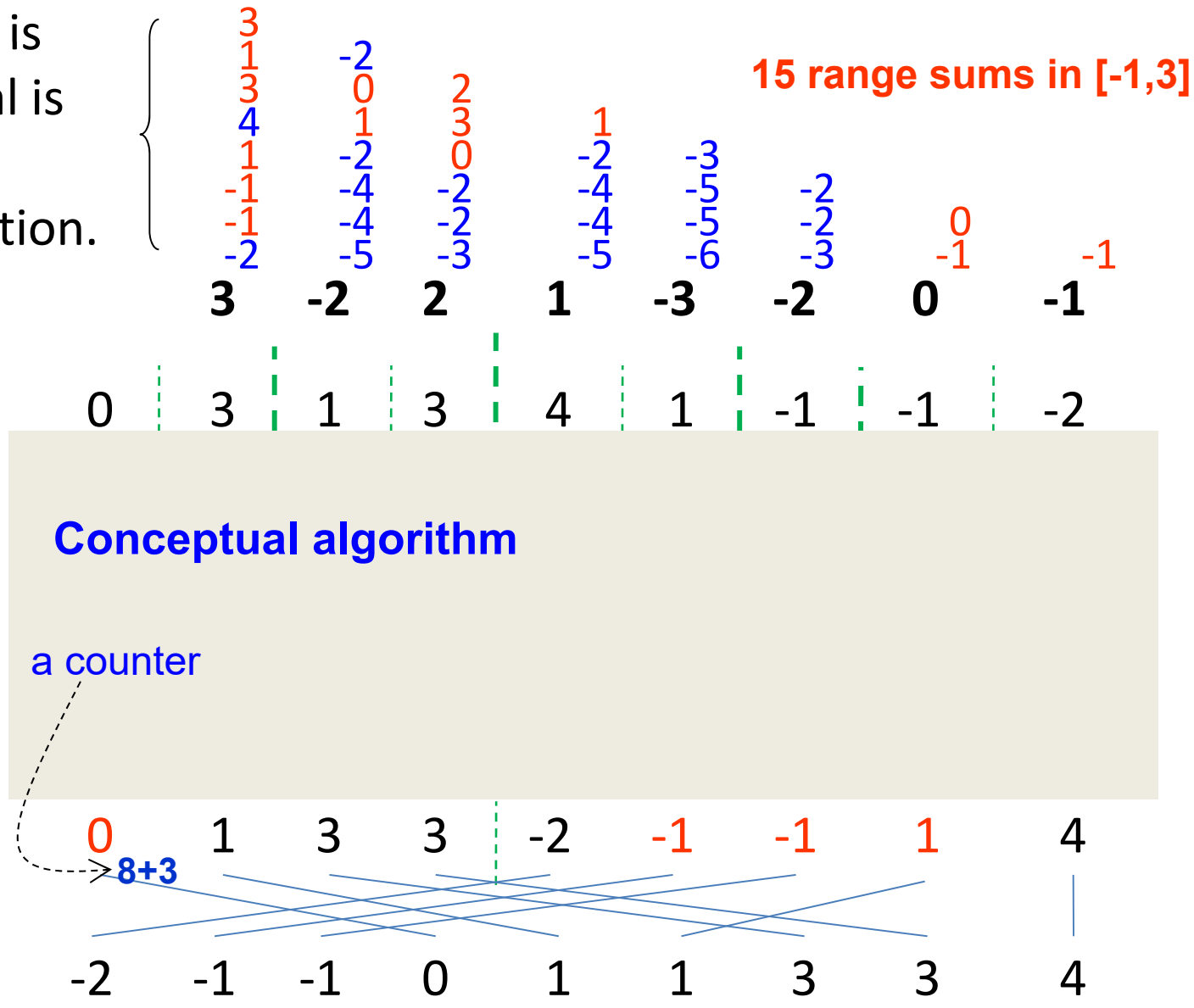
$O(n^2)$ solution is trivial, the goal is to devise an $O(n \log n)$ solution.





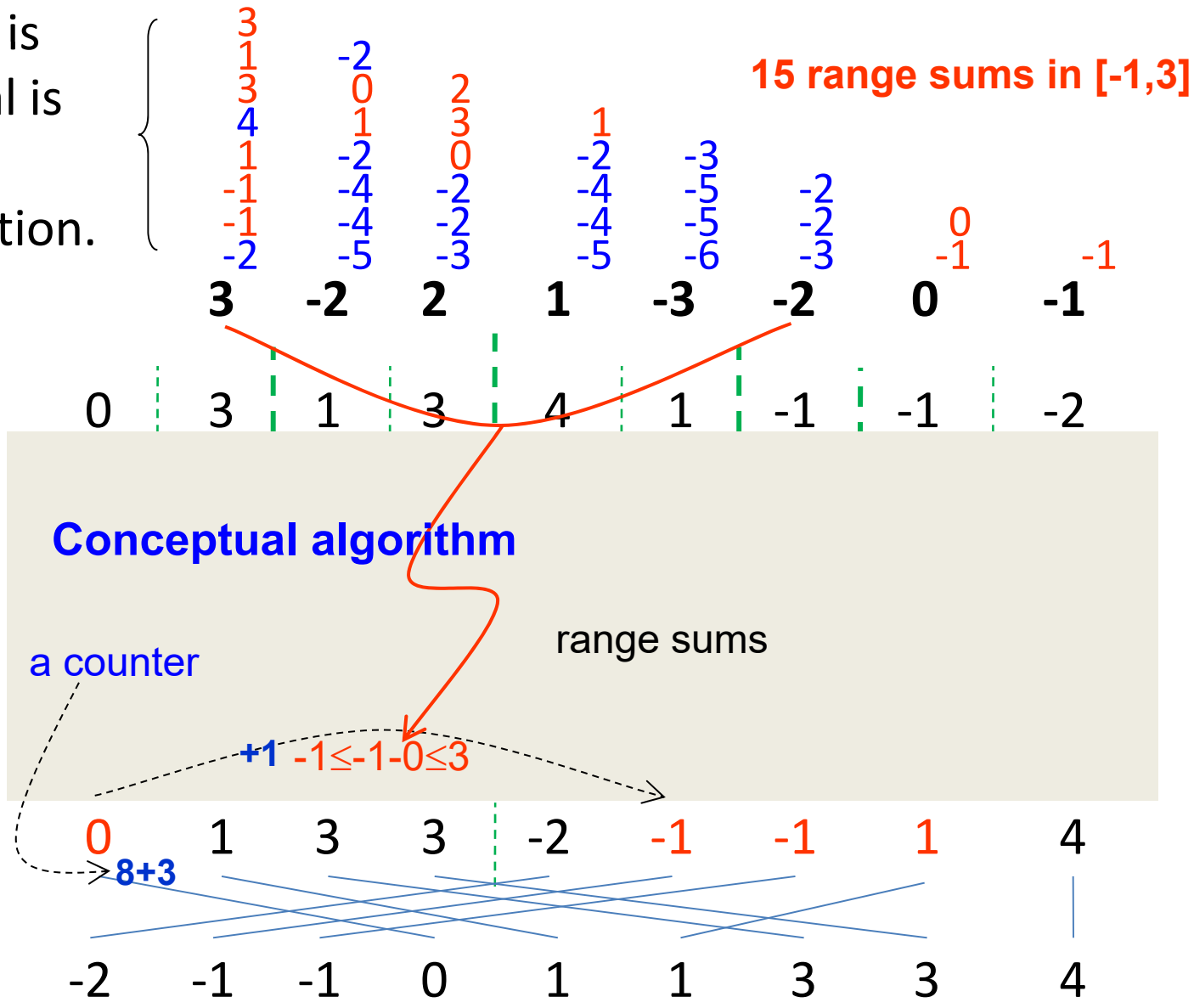
LC327 Count of Range Sum

$O(n^2)$ solution is trivial, the goal is to devise an $O(n \log n)$ solution.



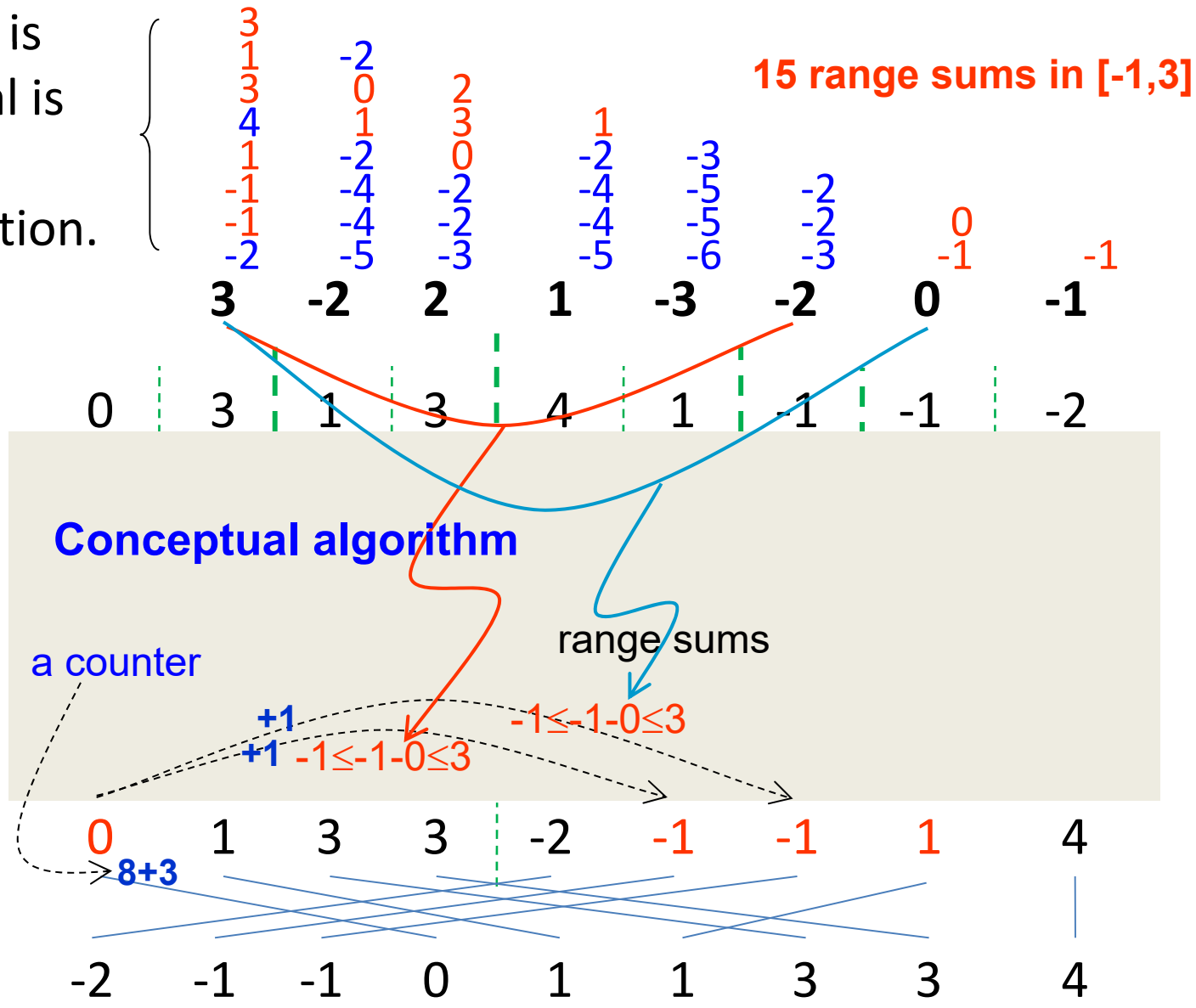
LC327 Count of Range Sum

$O(n^2)$ solution is trivial, the goal is to devise an $O(n \log n)$ solution.



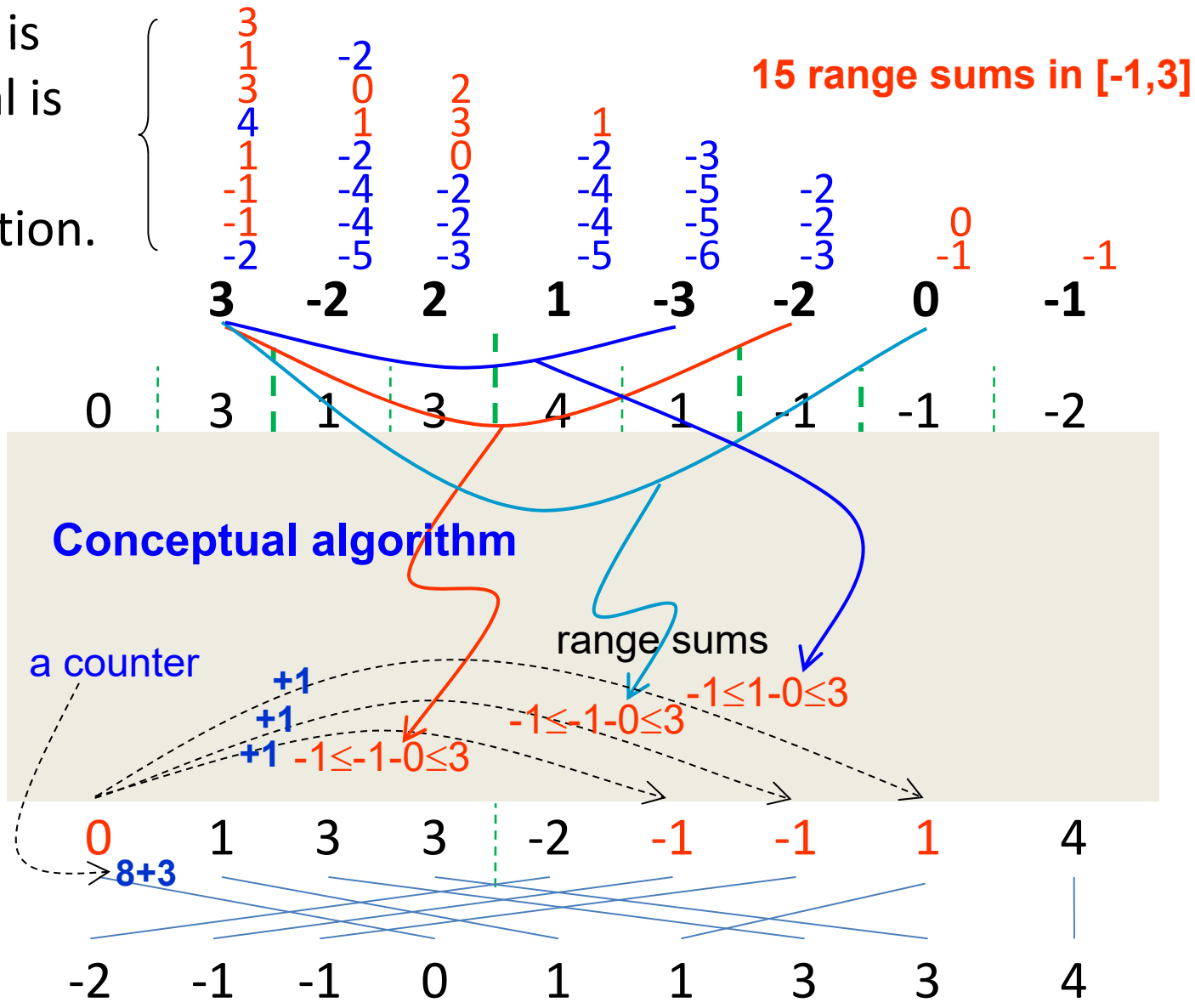
LC327 Count of Range Sum

$O(n^2)$ solution is trivial, the goal is to devise an O(n logn) solution.



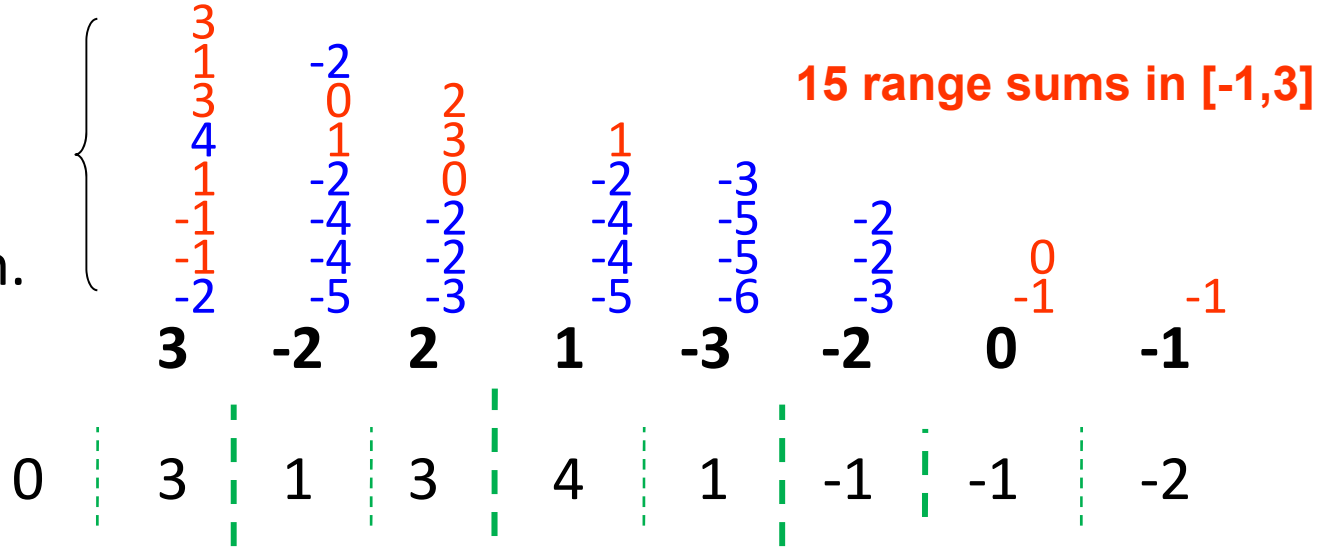
LC327 Count of Range Sum

O(n^2) solution is trivial, the goal is to devise an O(n logn) solution.



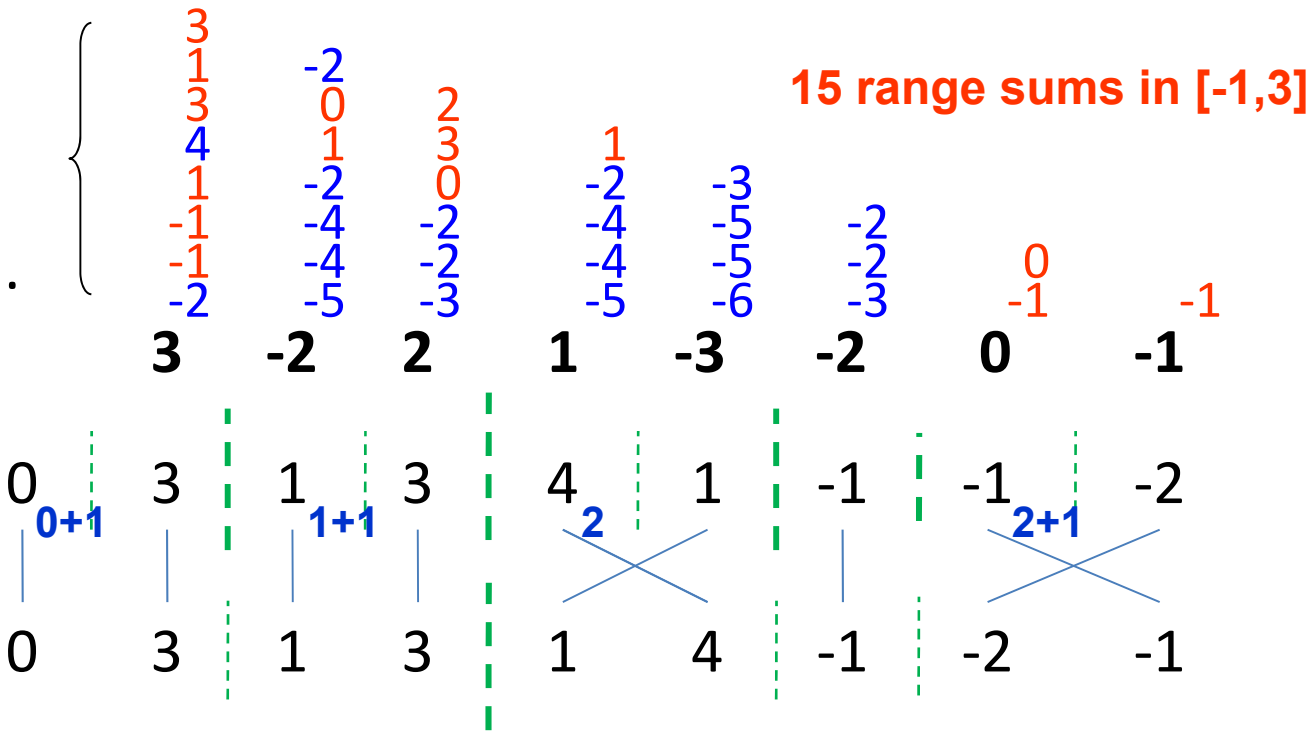
LC327 Count of Range Sum

$O(n^2)$ solution is trivial, the goal is to devise an $O(n \log n)$ solution.



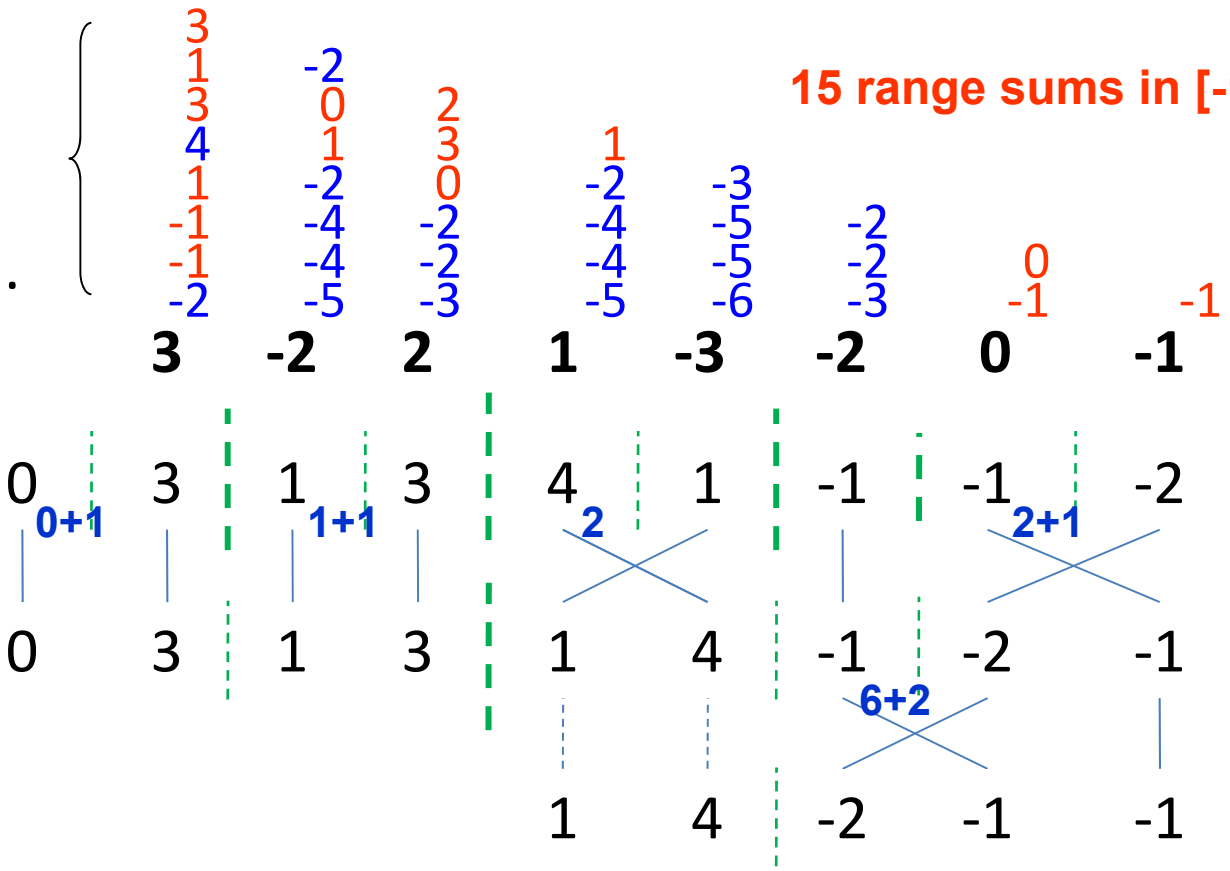
LC327 Count of Range Sum

$O(n^2)$ solution is trivial, the goal is to devise an $O(n \log n)$ solution.



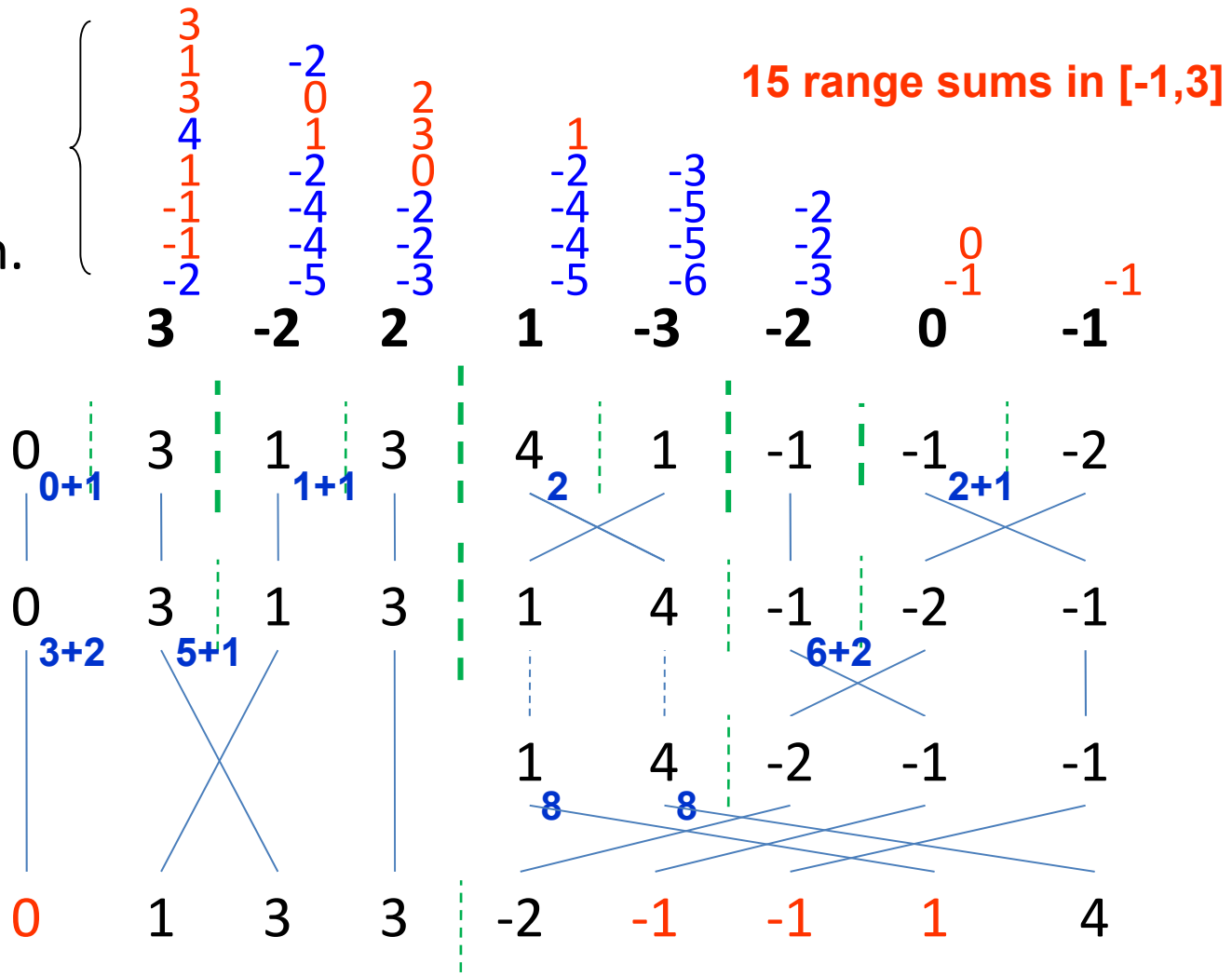
LC327 Count of Range Sum

$O(n^2)$ solution is trivial, the goal is to devise an $O(n \log n)$ solution.



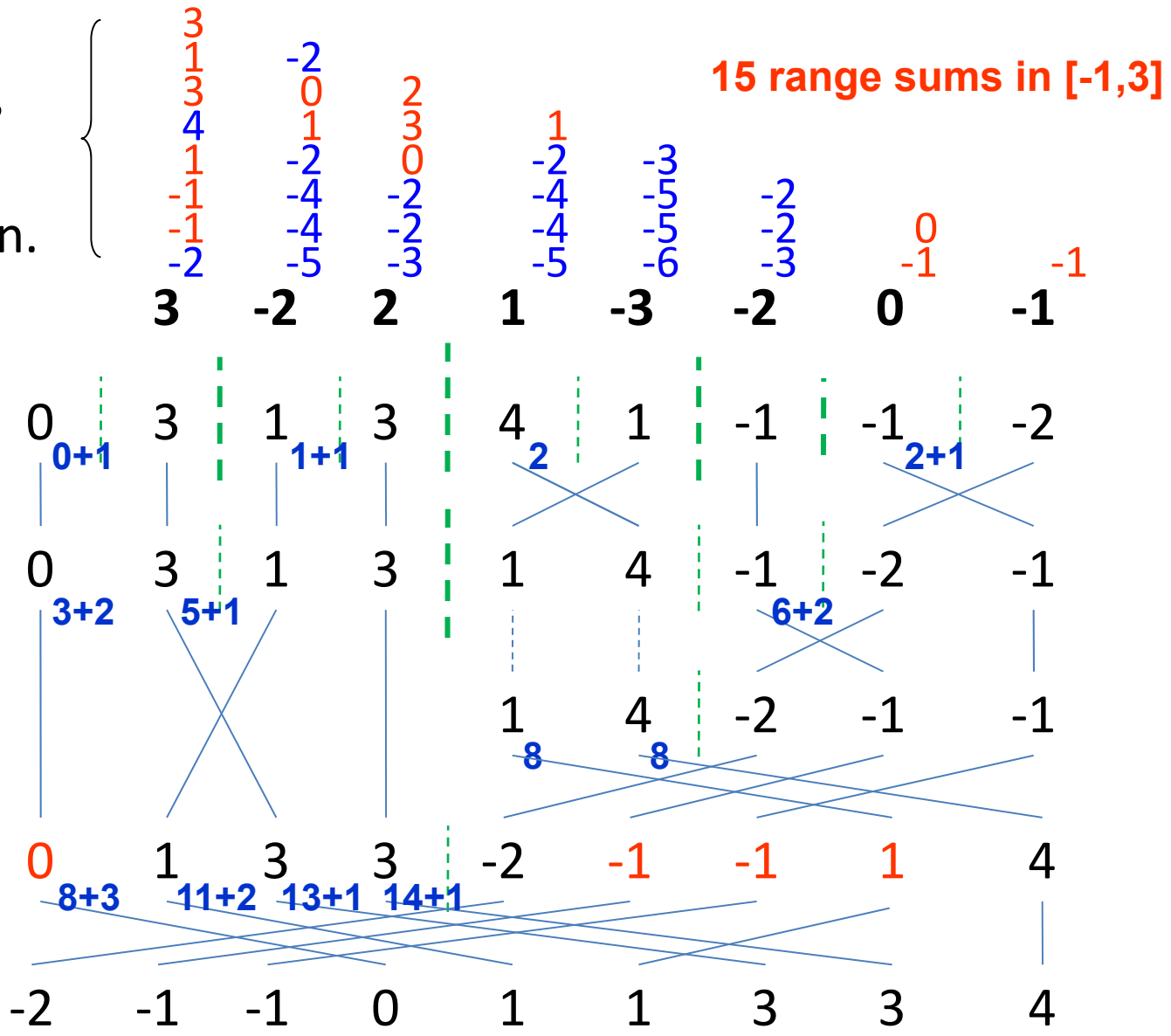
LC327 Count of Range Sum

$O(n^2)$ solution is trivial, the goal is to devise an $O(n \log n)$ solution.



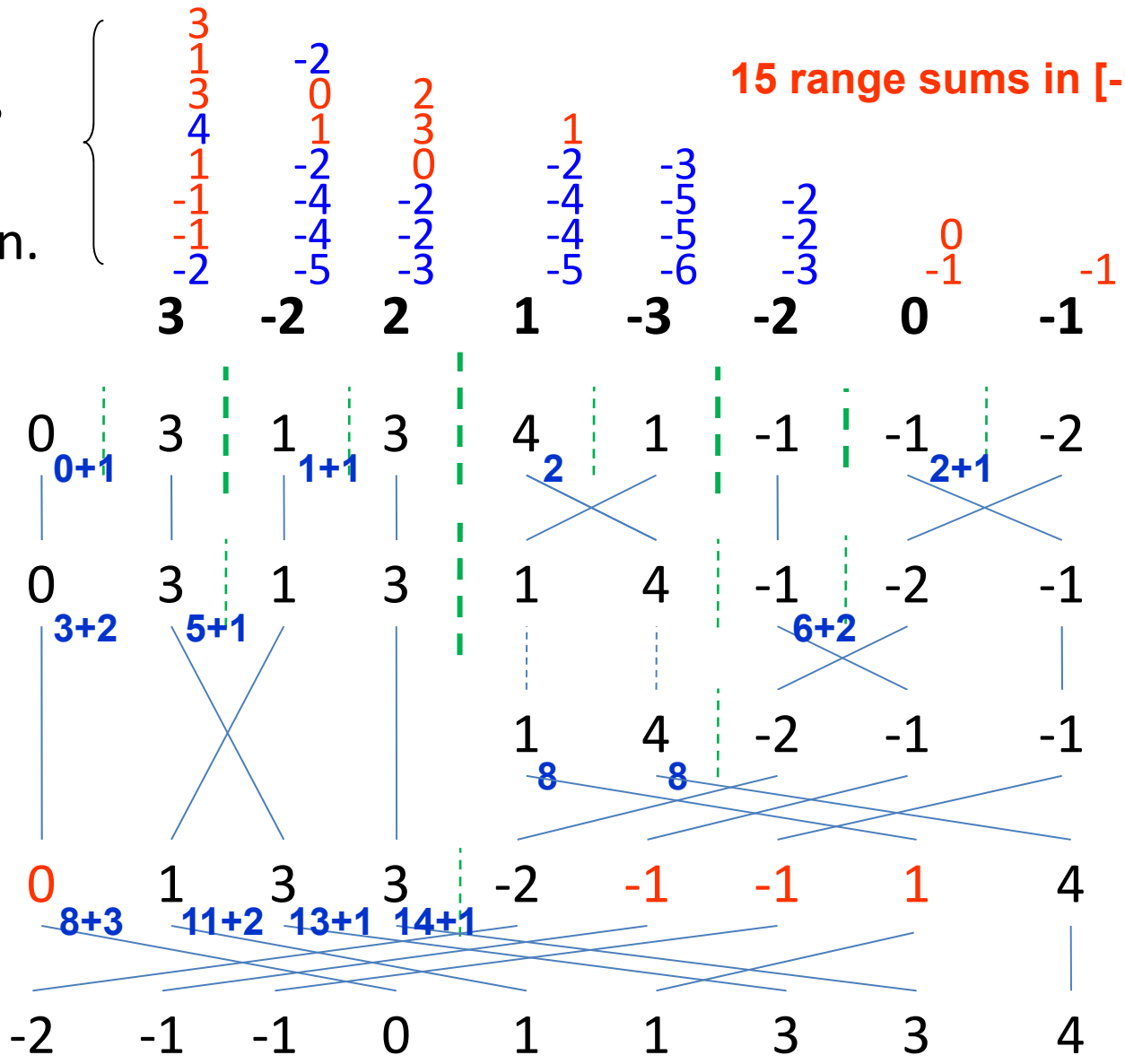
LC327 Count of Range Sum

$O(n^2)$ solution is trivial, the goal is to devise an $O(n \log n)$ solution.



LC327 Count of Range Sum

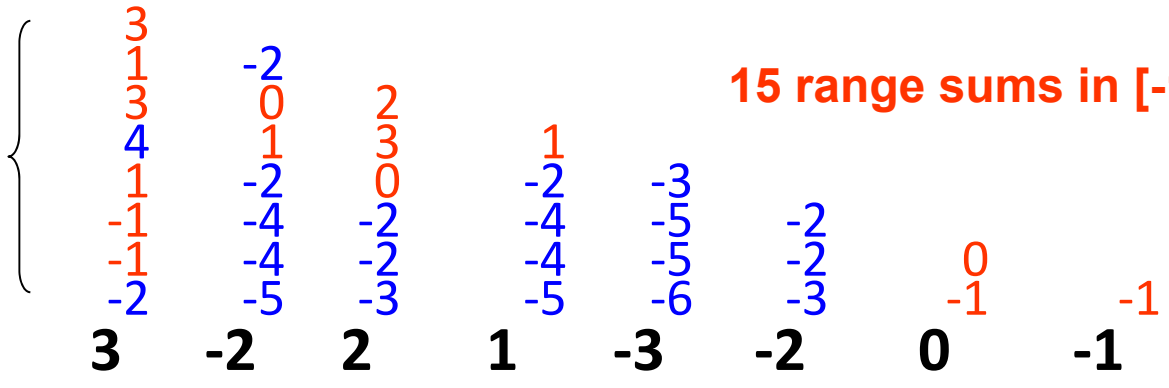
$O(n^2)$ solution is trivial, the goal is to devise an O(n logn) solution.



Why does it run in time O(n logn)?

LC327 Count of Range Sum

$O(n^2)$ solution is trivial, the goal is to devise an O(n logn) solution.

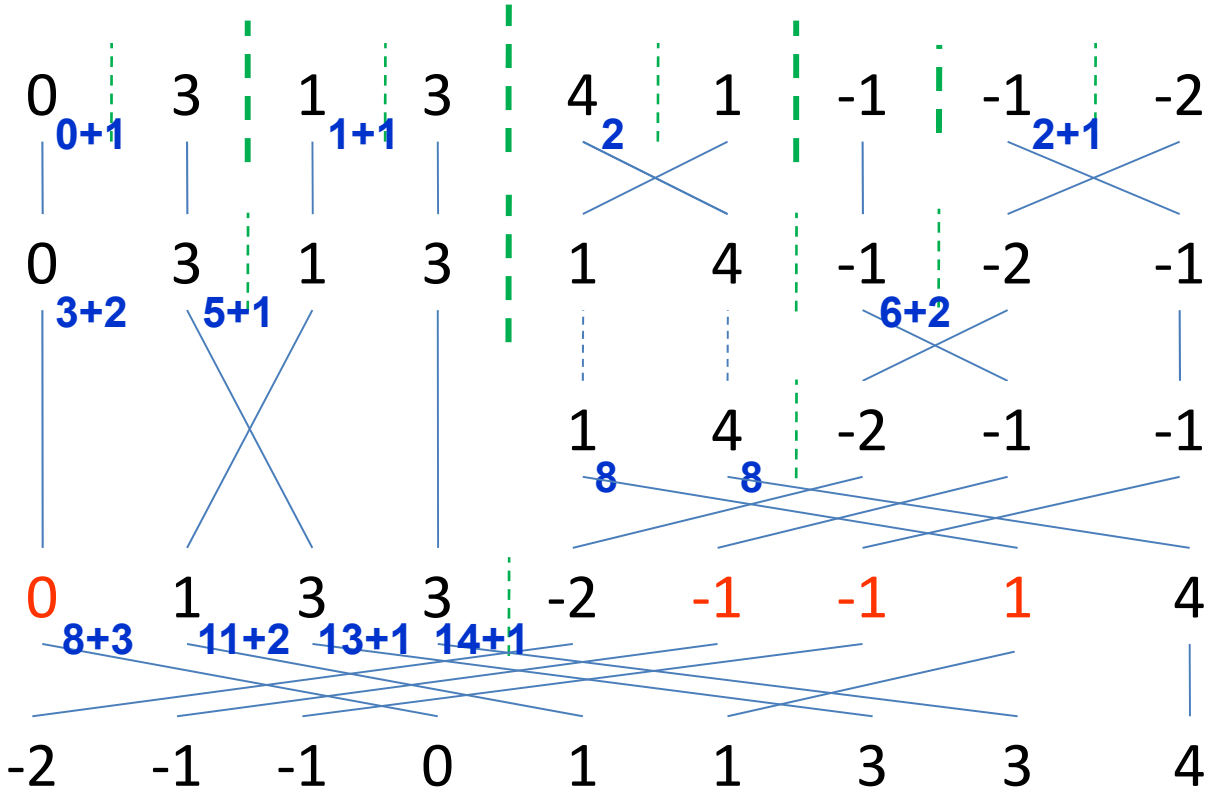


Why does it run in time O(n logn)?

Ans.

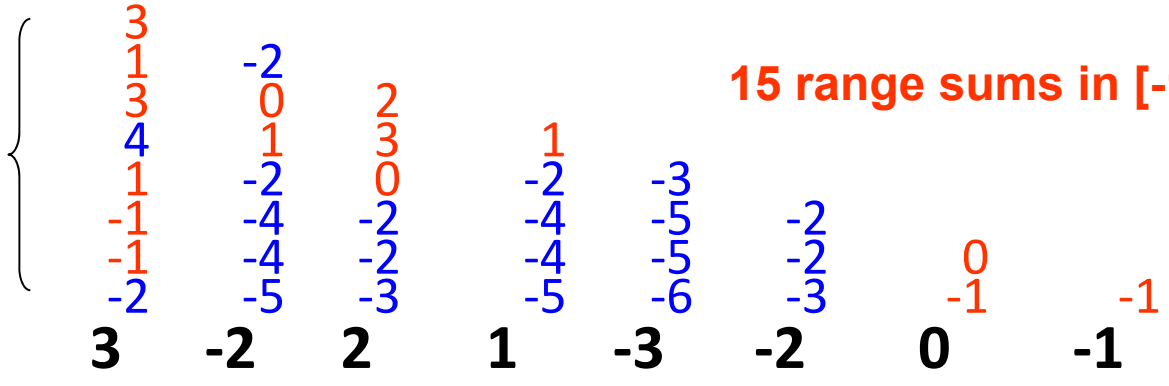
Some comparisons are skipped because they are ordered.

e.g. 0 ↔ -2
0 ↔ -1
0 ↔ -1
0 ↔ 1
0 ↔ 4
0<1 1 ↔ -2
1 ↔ -1
1 ↔ -1
1 ↔ 1
1 ↔ 4



LC327 Count of Range Sum

$O(n^2)$ solution is trivial, the goal is to devise an O(n logn) solution.

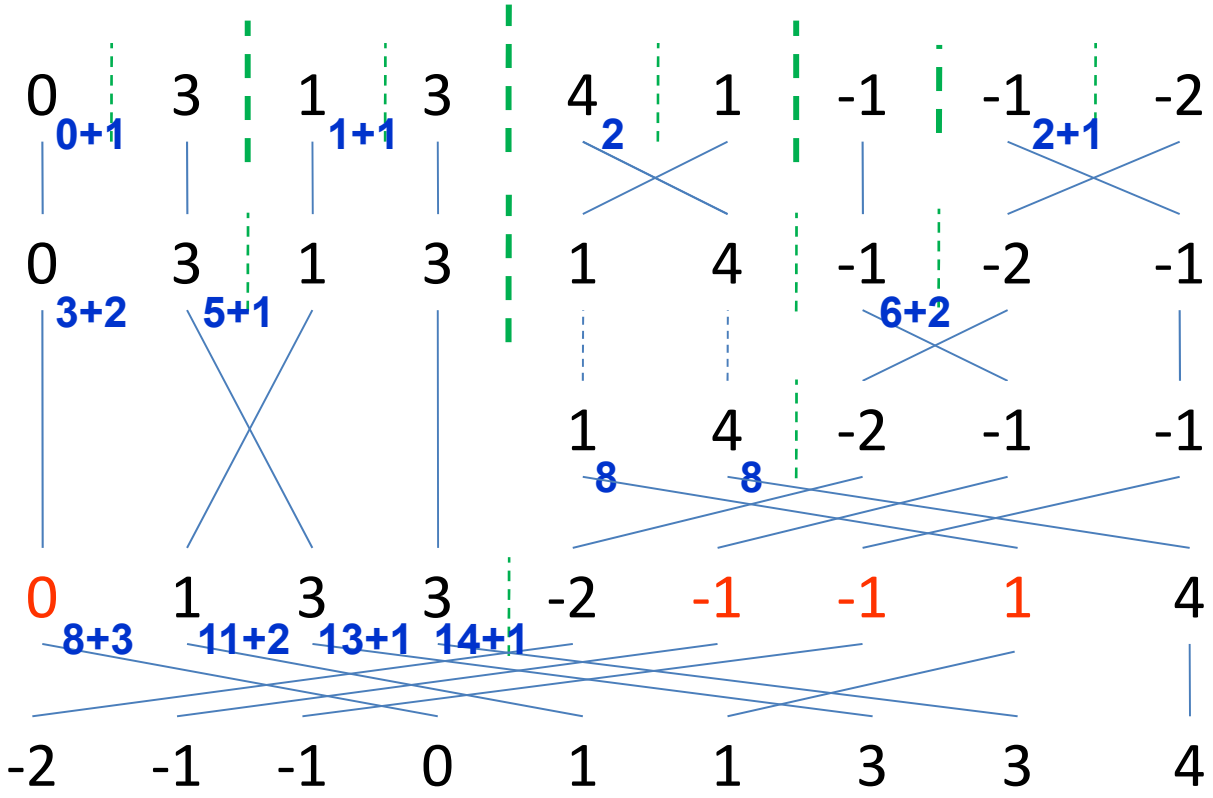


Why does it run in time O(n logn)?

Ans.

Some comparisons are skipped because they are ordered.

e.g. 0 ↔ -2
0 ↔ -1
0 ↔ -1
0 ↔ 1
0 ↔ 4
0<1 1 ↔ -2
1 ↔ -1
1 ↔ -1
1 ↔ 1
1 ↔ 4



Not tied to the merge sorting algorithm.

LC327 Count of Range Sum

15 range sums in [-1,3]

3 -2 2 1 -3 -2 0 -1

min=-1
max=3

LC327 Count of Range Sum

15 range sums in [-1,3]

		3	**-2**	**2**	**1**	**-3**	**-2**	**0**	**-1**
min=-1									
max=3	0	3	1	3	4	1	-1	-1	-2

LC327 Count of Range Sum

15 range sums in [-1,3]

min=-1
max=3

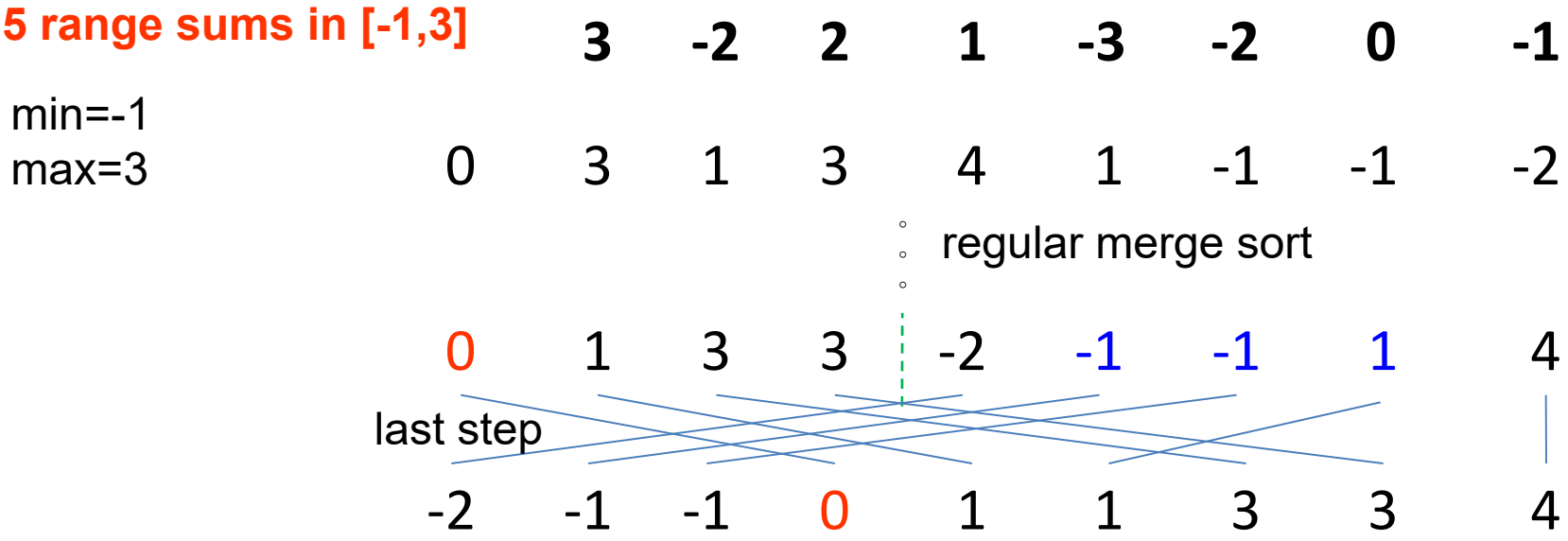
	3	**-2**	**2**	**1**	**-3**	**-2**	**0**	**-1**
0	3	1	3	4	1	-1	-1	-2

regular merge sort

0	1	3	3	-2	-1	-1	1	4

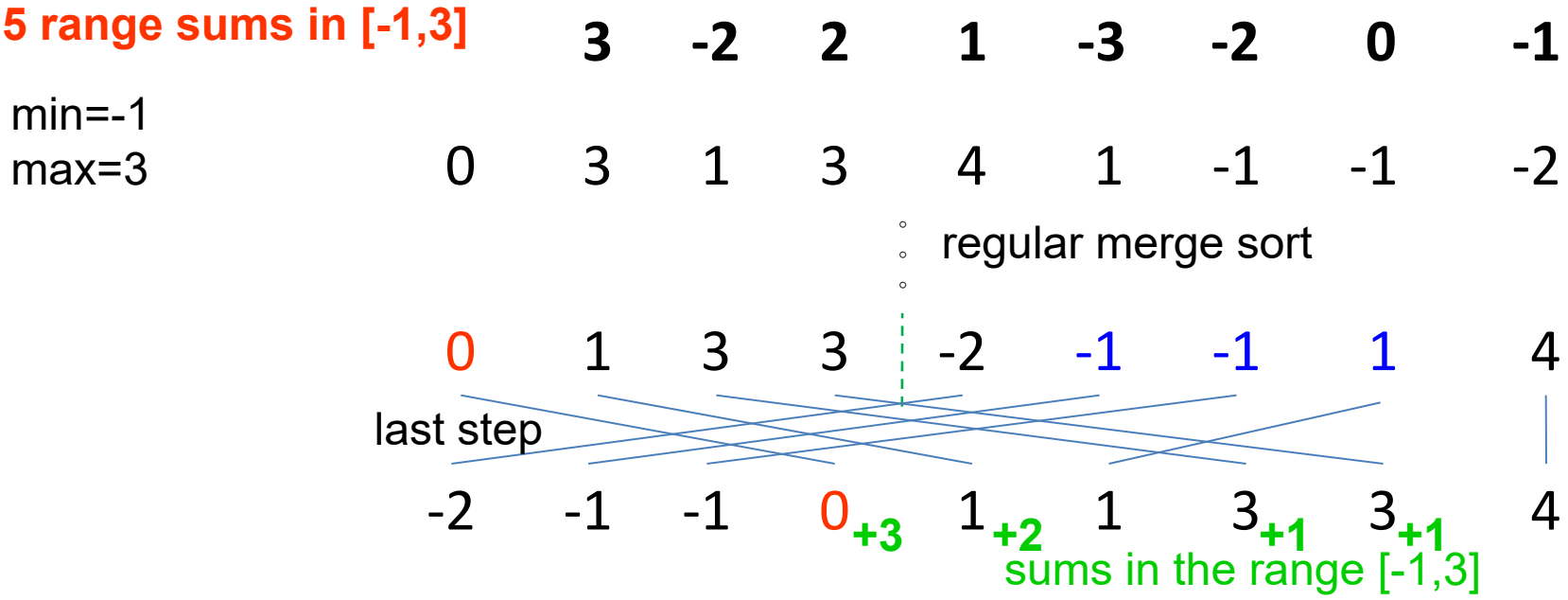
LC327 Count of Range Sum

15 range sums in [-1,3]

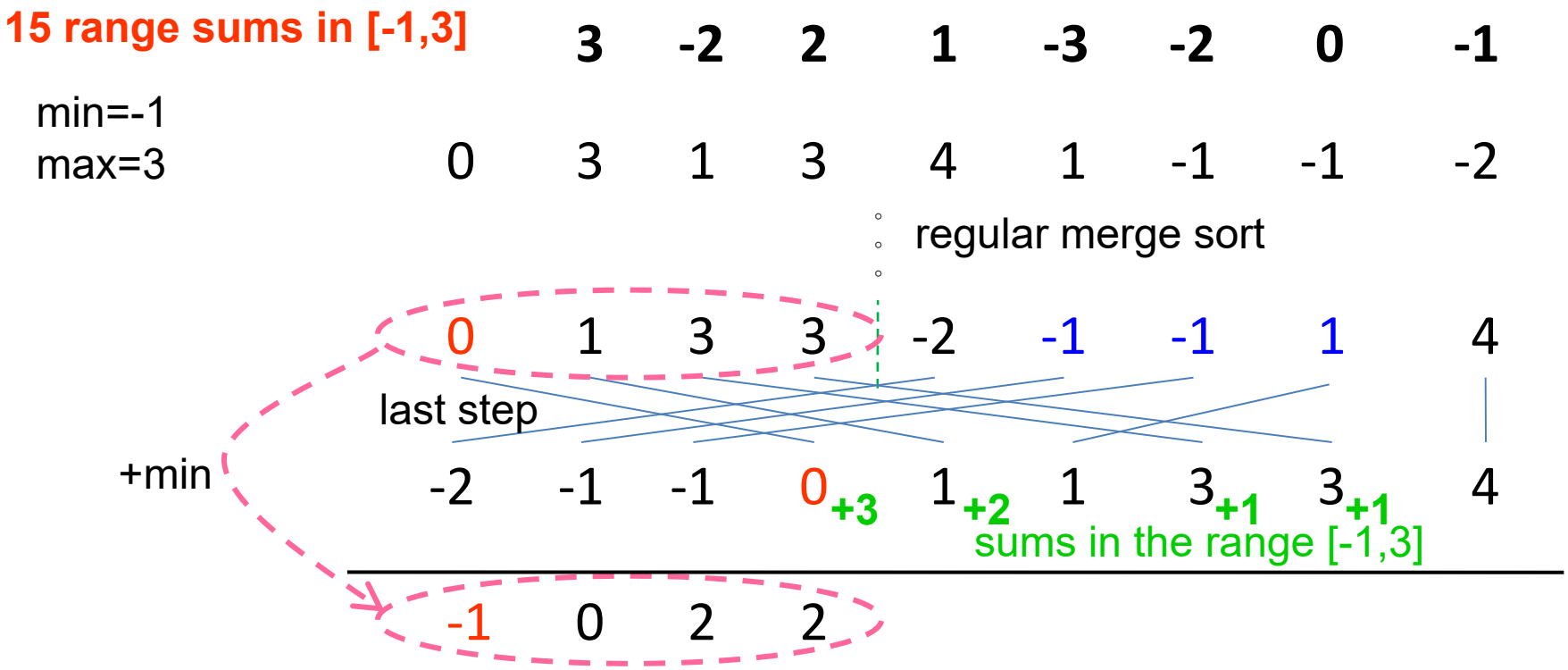


LC327 Count of Range Sum

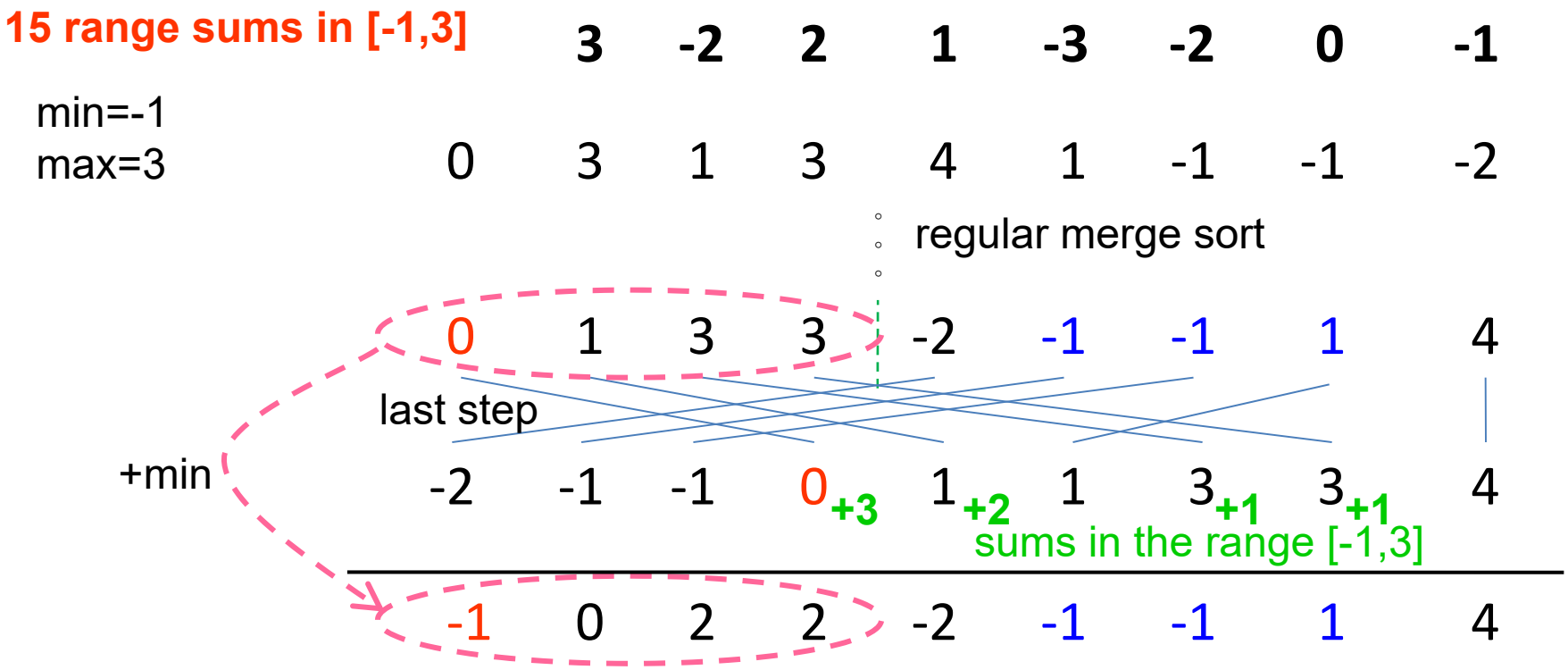
15 range sums in [-1,3]



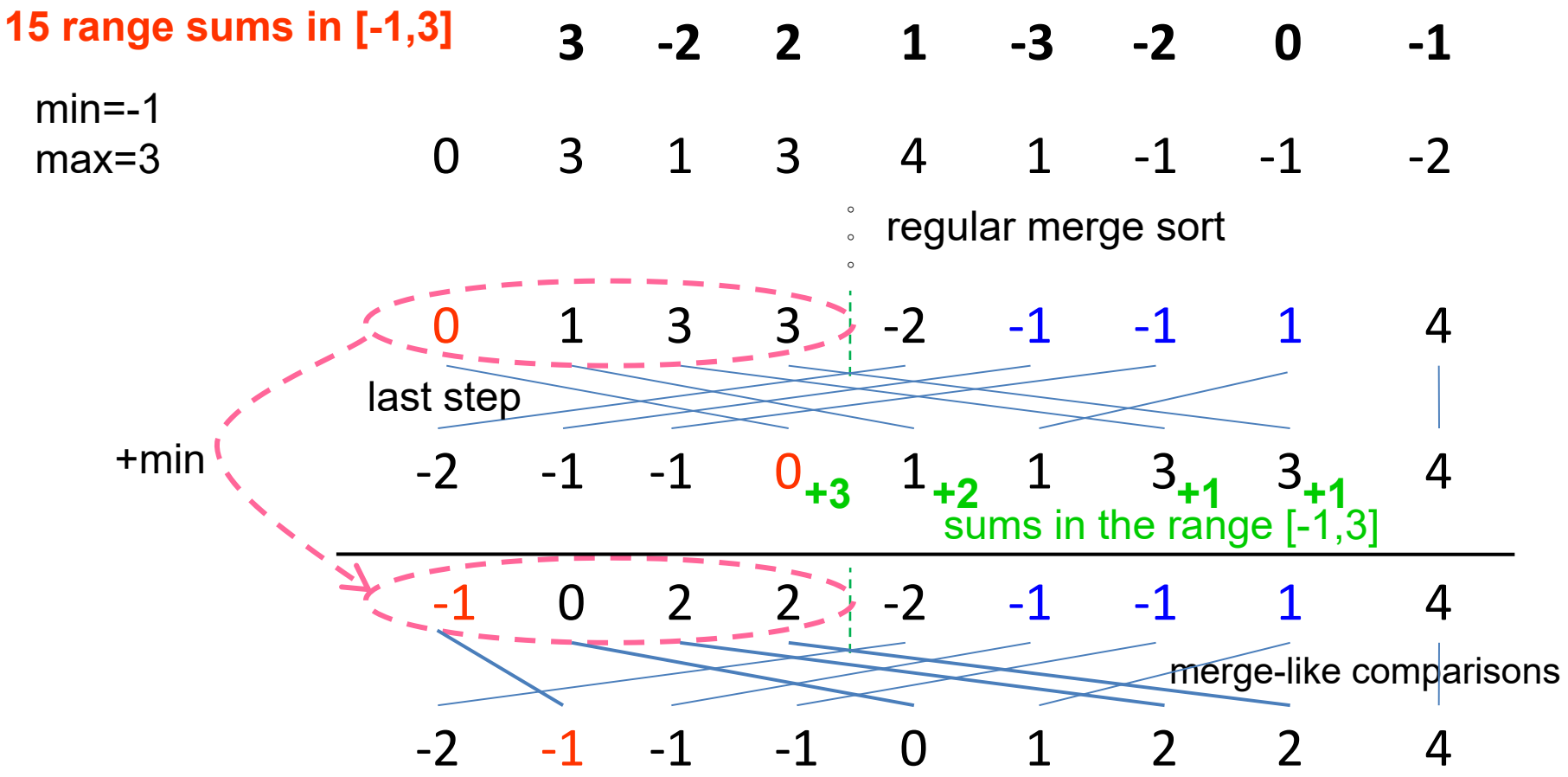
LC327 Count of Range Sum



LC327 Count of Range Sum



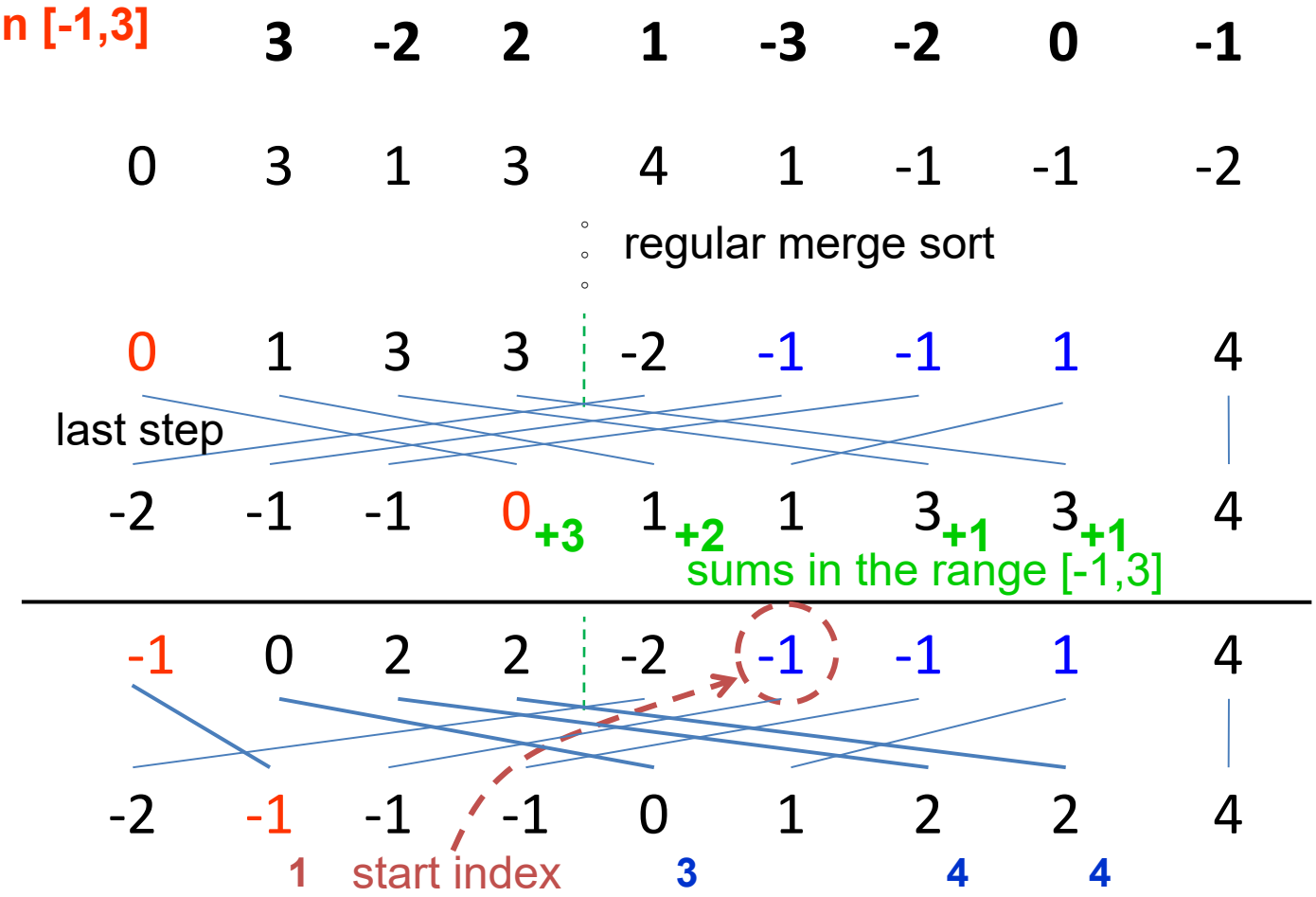
LC327 Count of Range Sum



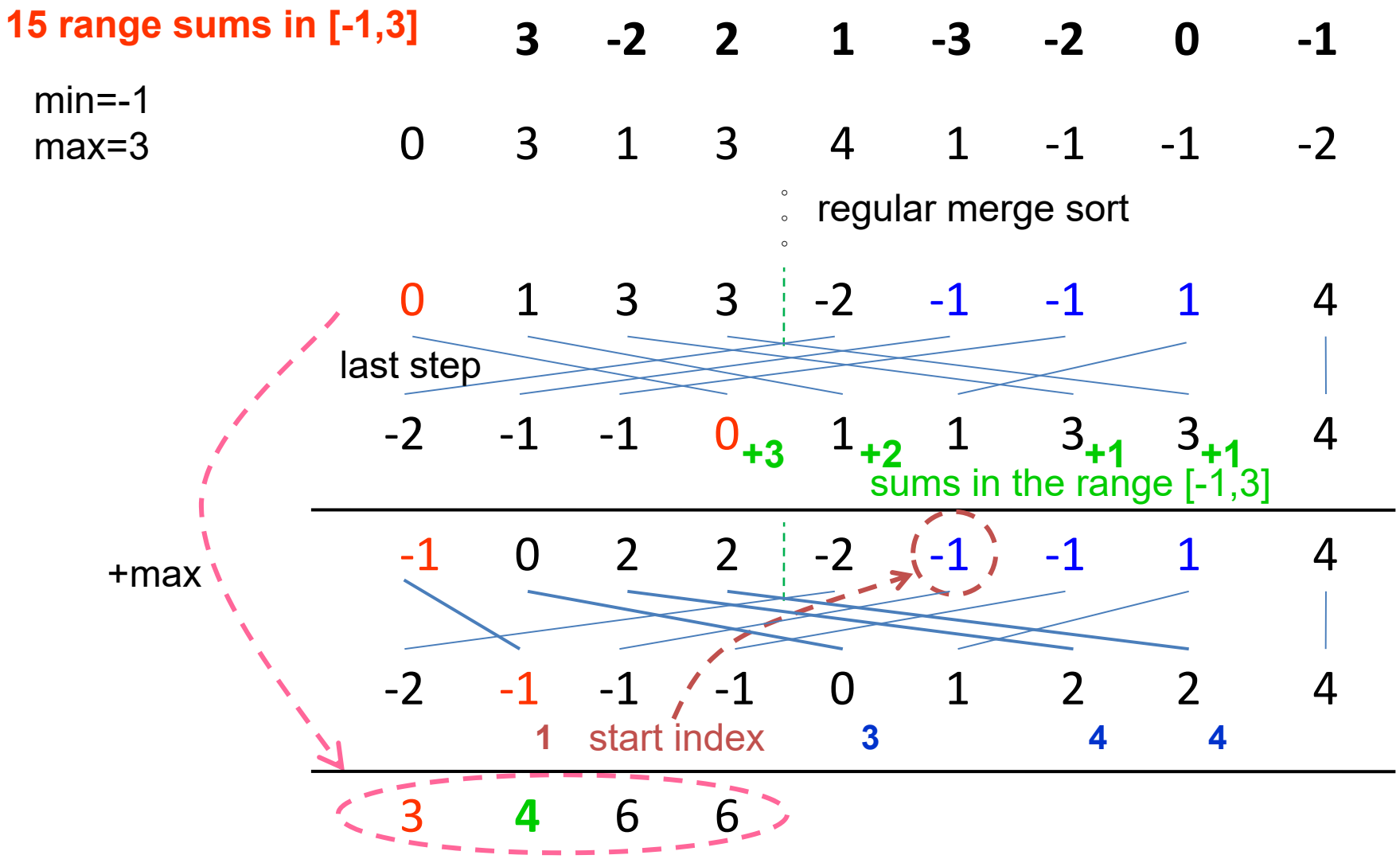
LC327 Count of Range Sum

15 range sums in [-1,3]

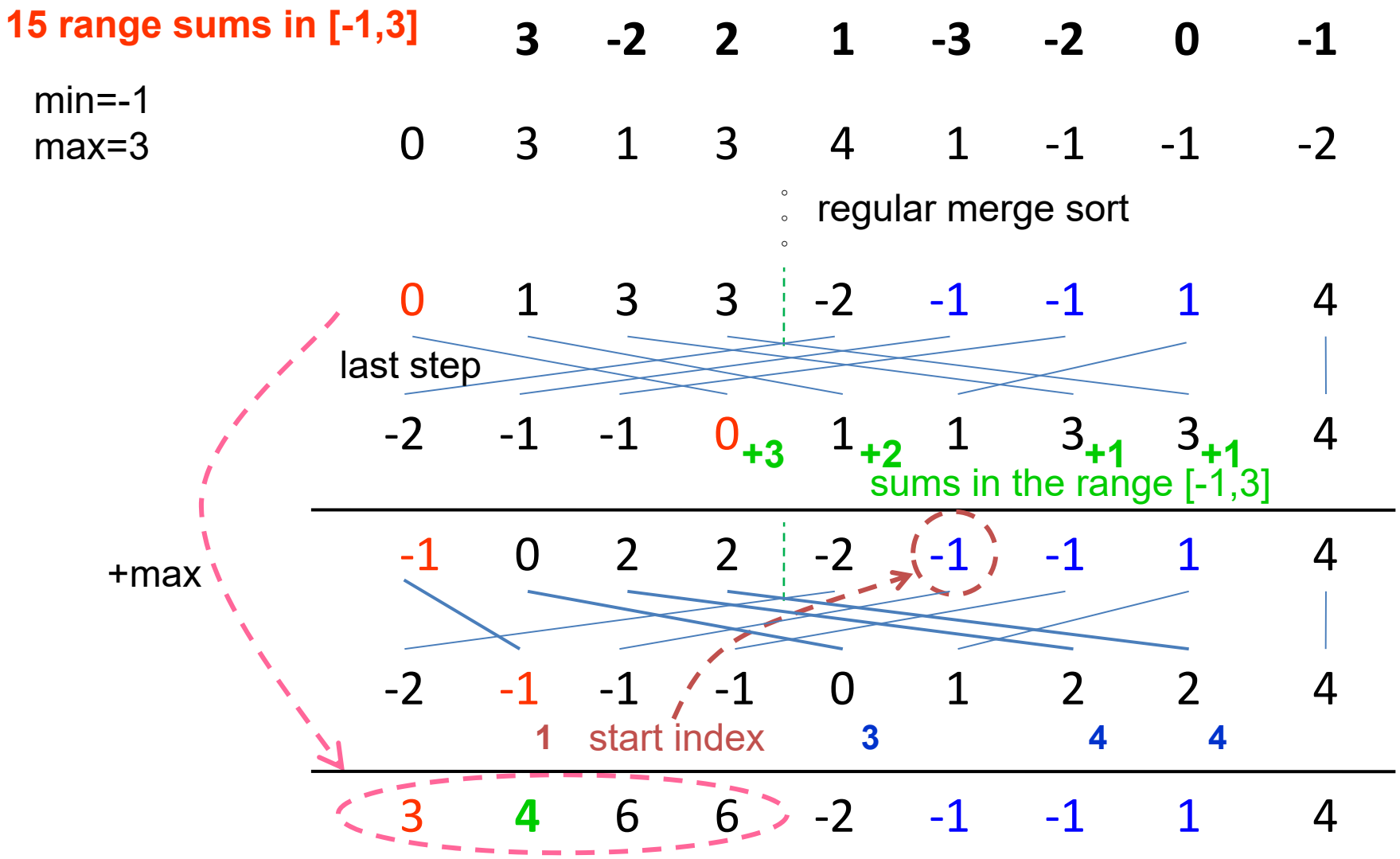
min=-1
max=3



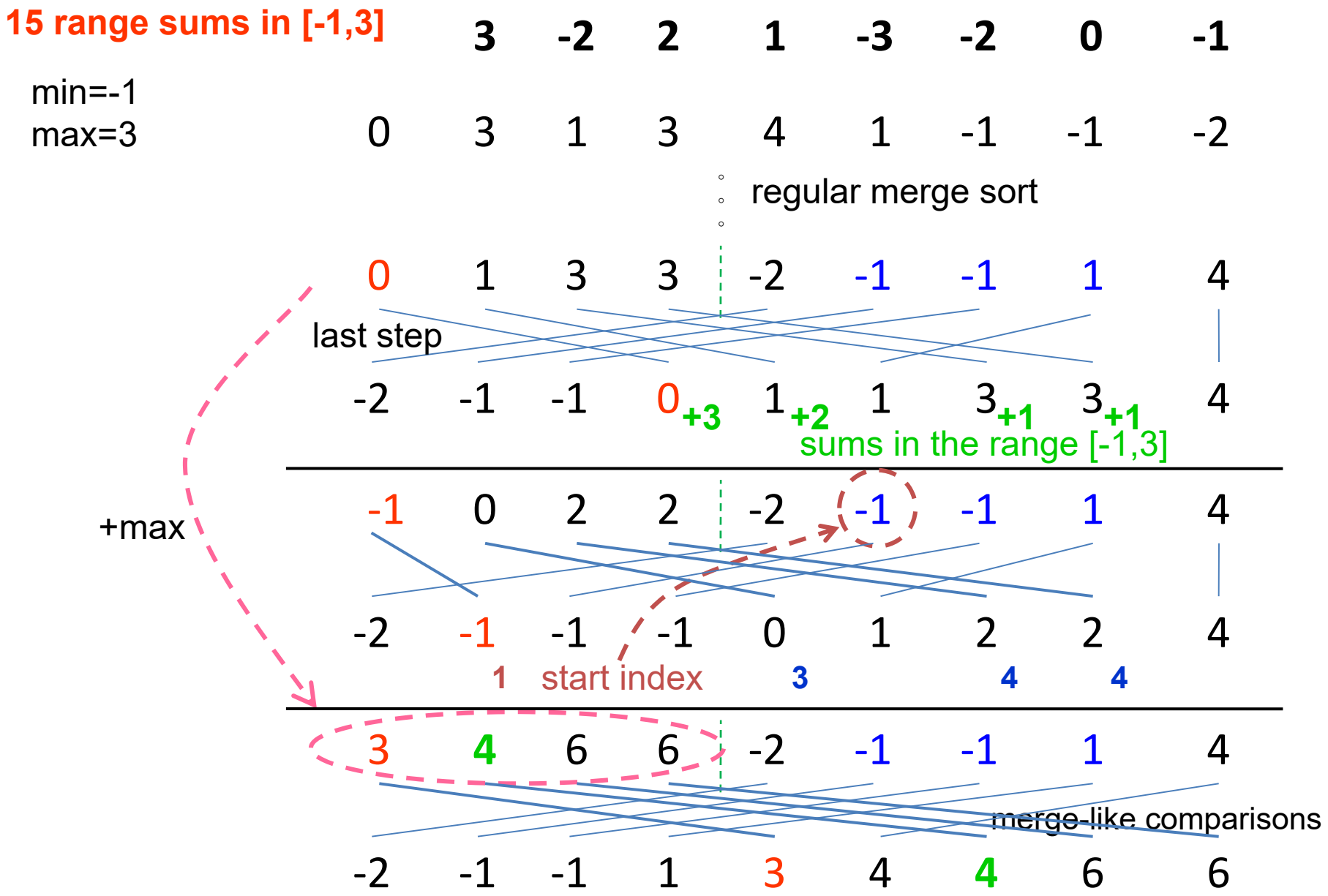
LC327 Count of Range Sum



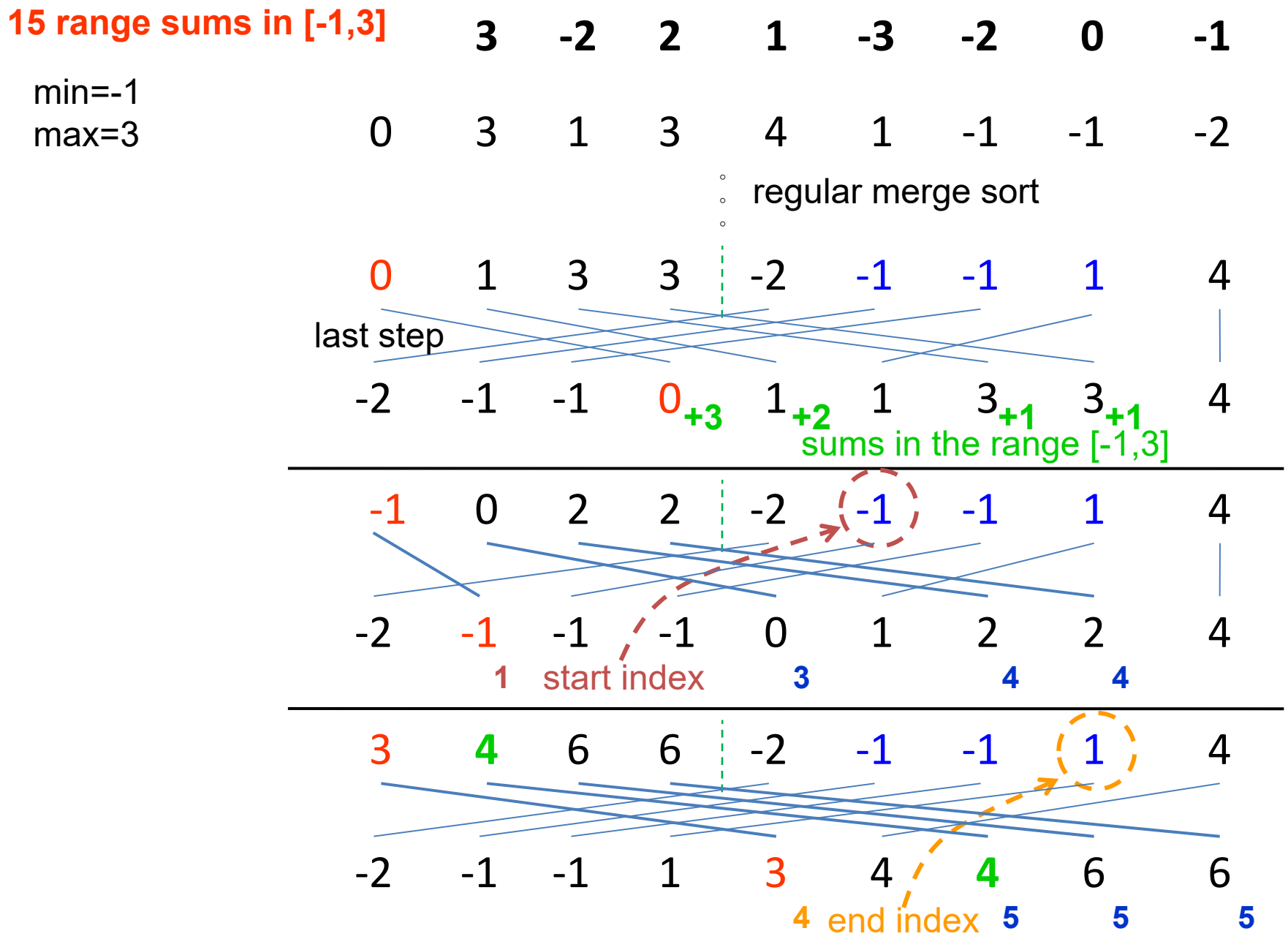
LC327 Count of Range Sum



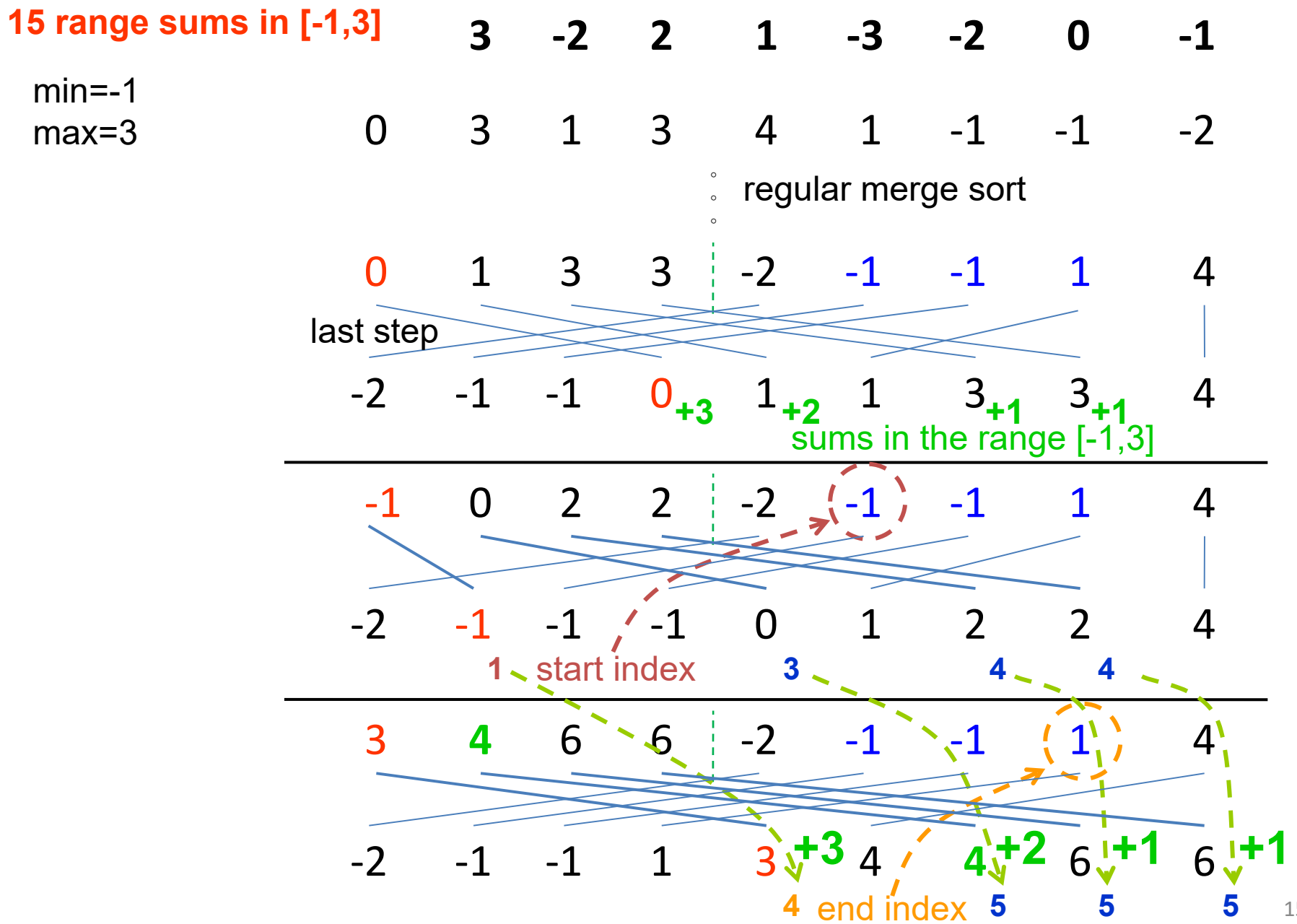
LC327 Count of Range Sum



LC327 Count of Range Sum

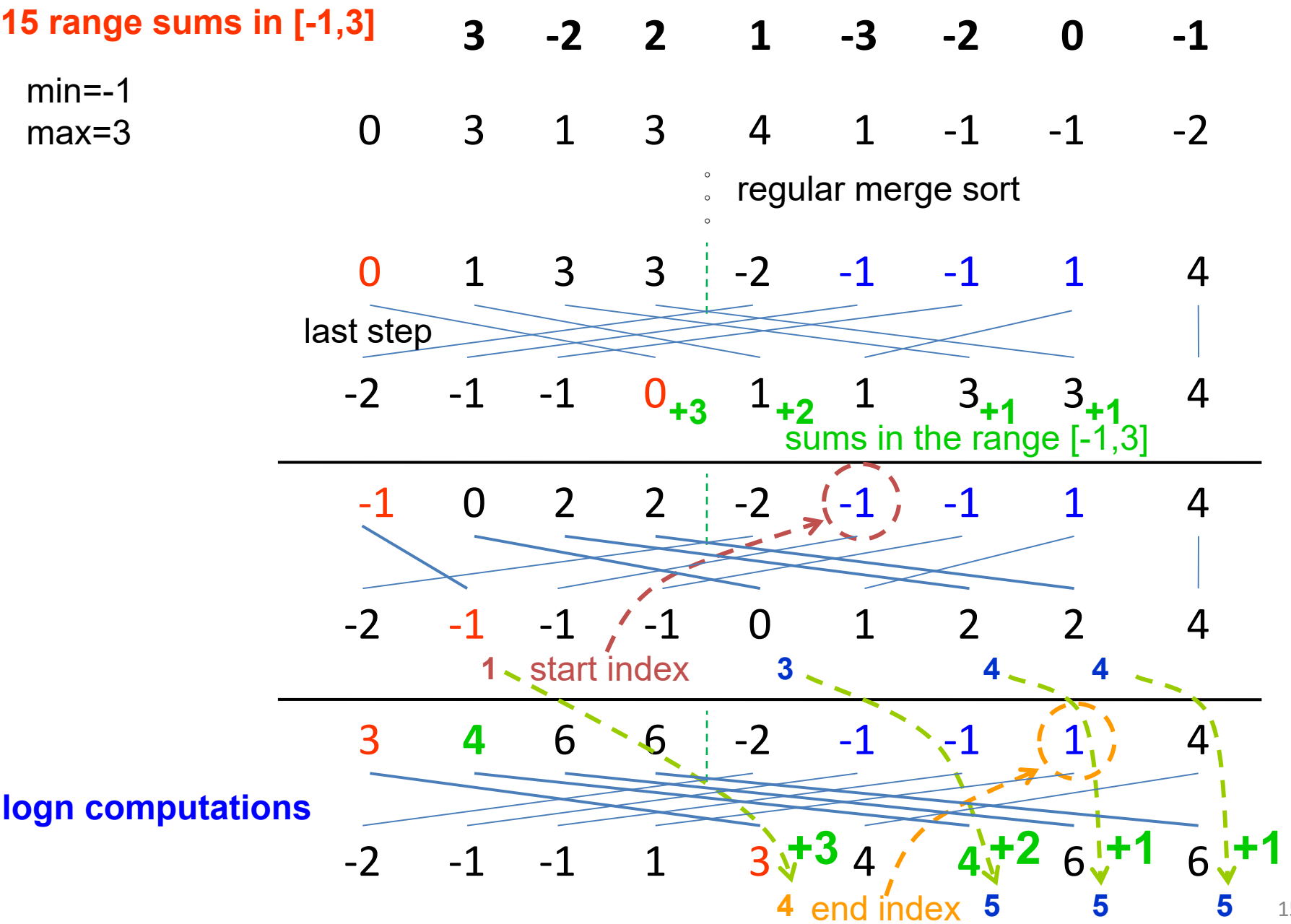


LC327 Count of Range Sum



LC327 Count of Range Sum

15 range sums in [-1,3]



3 n logn computations

其他應用

- ZJ d478. 共同的數 - 簡易版
- ZJ b512. 高維度稀疏向量