分治法

分治 divide and conquer，一種解決問題的方法

他的理念是「將一個大問題切割成各種小問題後遞迴地解決最終合併」

分治 divide and conquer，一種解決問題的方法

他的理念是「將一個大問題切割成各種小問題後遞迴地解決最終合併」

因為他需要將問題分割成小問題，所以問題拆解的邊界劃分是他的困難之處

分治 divide and conquer，一種解決問題的方法

他的理念是「將一個大問題切割成各種小問題後遞迴地解決最終合併」

因為他需要將問題分割成小問題，所以問題拆解的邊界劃分是他的困難之處

我們喜歡用分治法有幾個原因：

分治 divide and conquer，一種解決問題的方法

他的理念是「將一個大問題切割成各種小問題後遞迴地解決最終合併」

因為他需要將問題分割成小問題，所以問題拆解的邊界劃分是他的困難之處

我們喜歡用分治法有幾個原因：

1. 解決困難問題

分治 divide and conquer，一種解決問題的方法

他的理念是「將一個大問題切割成各種小問題後遞迴地解決最終合併」

因為他需要將問題分割成小問題，所以問題拆解的邊界劃分是他的困難之處

我們喜歡用分治法有幾個原因：

1. 解決困難問題
2. 跟遞迴一樣簡化問題

分治 divide and conquer，一種解決問題的方法

他的理念是「將一個大問題切割成各種小問題後遞迴地解決最終合併」

因為他需要將問題分割成小問題，所以問題拆解的邊界劃分是他的困難之處

我們喜歡用分治法有幾個原因：

1. 解決困難問題
2. 跟遞迴一樣簡化問題
3. 實際的分治技巧能夠最佳化時間複雜度

分治法的理念如下：

Divide、Conquer 與 Combine

分治法的理念如下：

Divide、Conquer 與 Combine

1. Divide：拆解問題，如果能拆成一半會更好！

分治法的理念如下：

Divide、Conquer 與 Combine

1. Divide：拆解問題，如果能拆成一半會更好！
2. Conquer：遞迴解決子問題

分治法的理念如下：

Divide、Conquer 與 Combine

1. Divide：拆解問題，如果能拆成一半會更好！
2. Conquer：遞迴解決子問題
3. Combine：合併已解決的子問題

其中，合併排序法是一個經典的分治演算法

它的複雜度定義為：

$$T(n) = \begin{cases} \Theta(1) & , \text{if } n \leq c \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & , \text{otherwise} \end{cases}$$

也能記 $O(n\log n)$ 就好

其中，合併排序法是一個經典的分治演算法

它的複雜度定義為：

$$T(n) = \begin{cases} \Theta(1) & , \text{if } n \leq c \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & , \text{otherwise} \end{cases}$$

也能記 $O(n\log n)$ 就好

知道複雜度還ok之後，
來了解一下合併排序法是怎麼成為分治法的原因

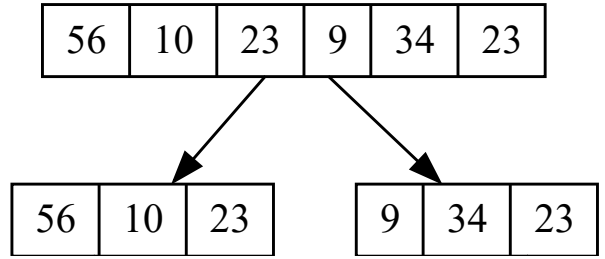
56	10	23	9	34	23

給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 divide 的部分中，合併排序法會將所有陣列分割一半

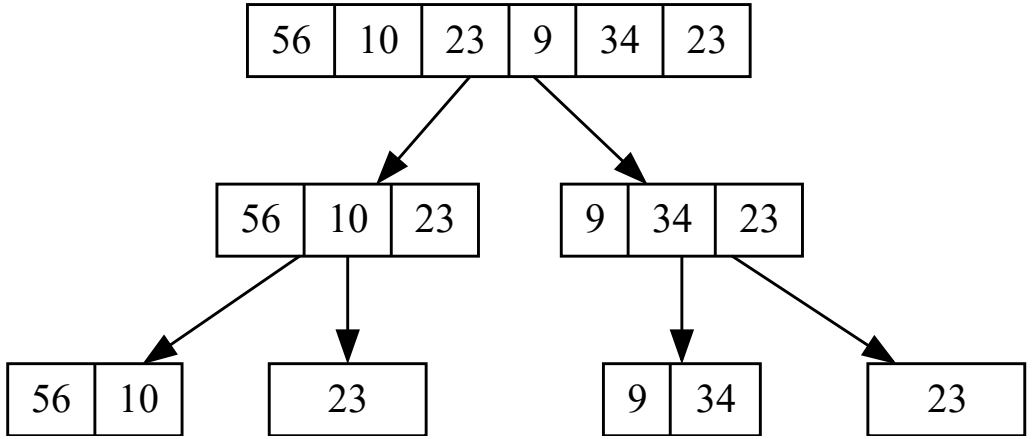
給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 divide 的部分中，合併排序法會將所有陣列分割一半



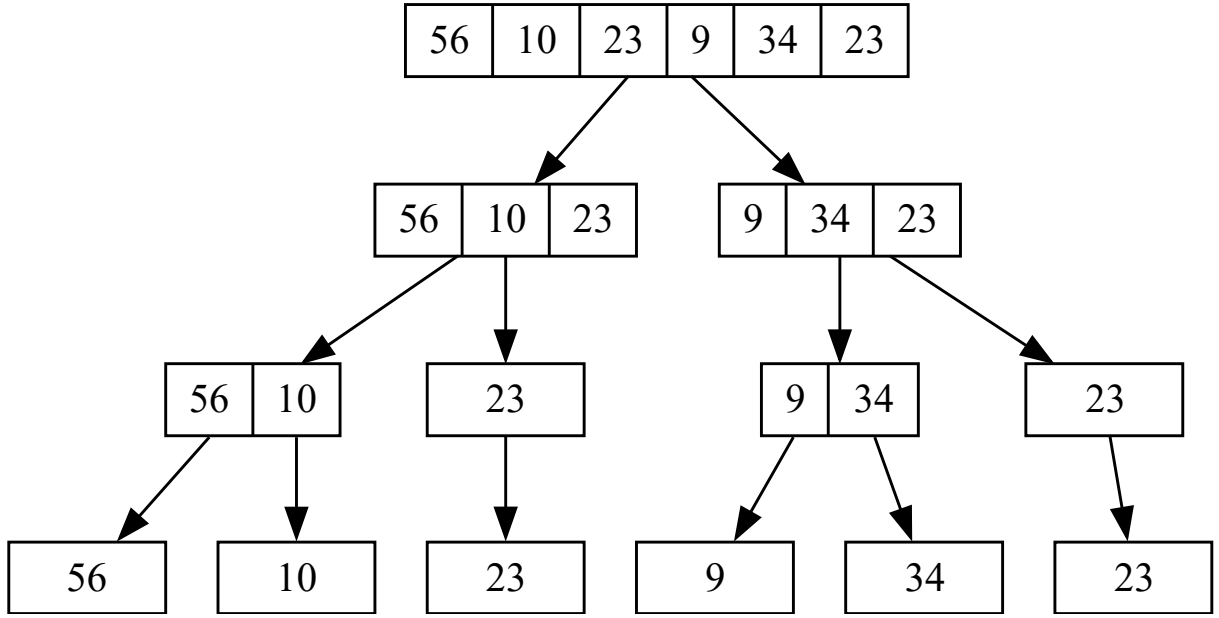
給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 divide 的部分中，合併排序法會將所有陣列分割一半



給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

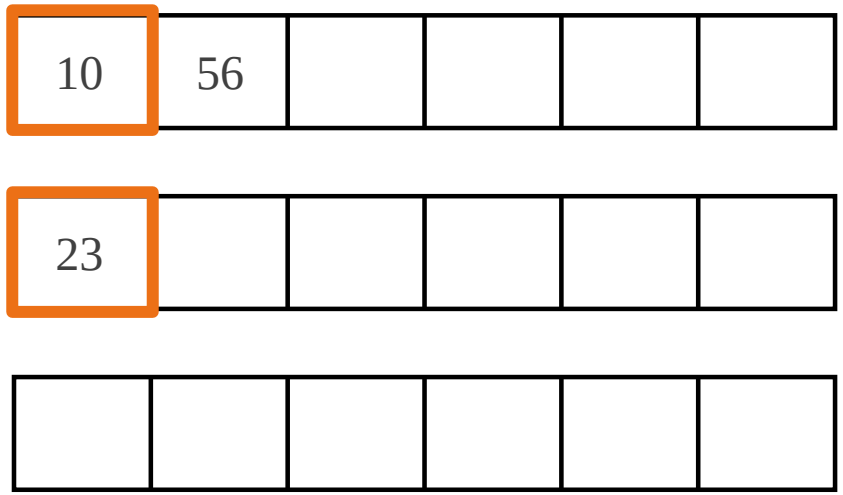
在 divide 的部分中，合併排序法會將所有陣列分割一半

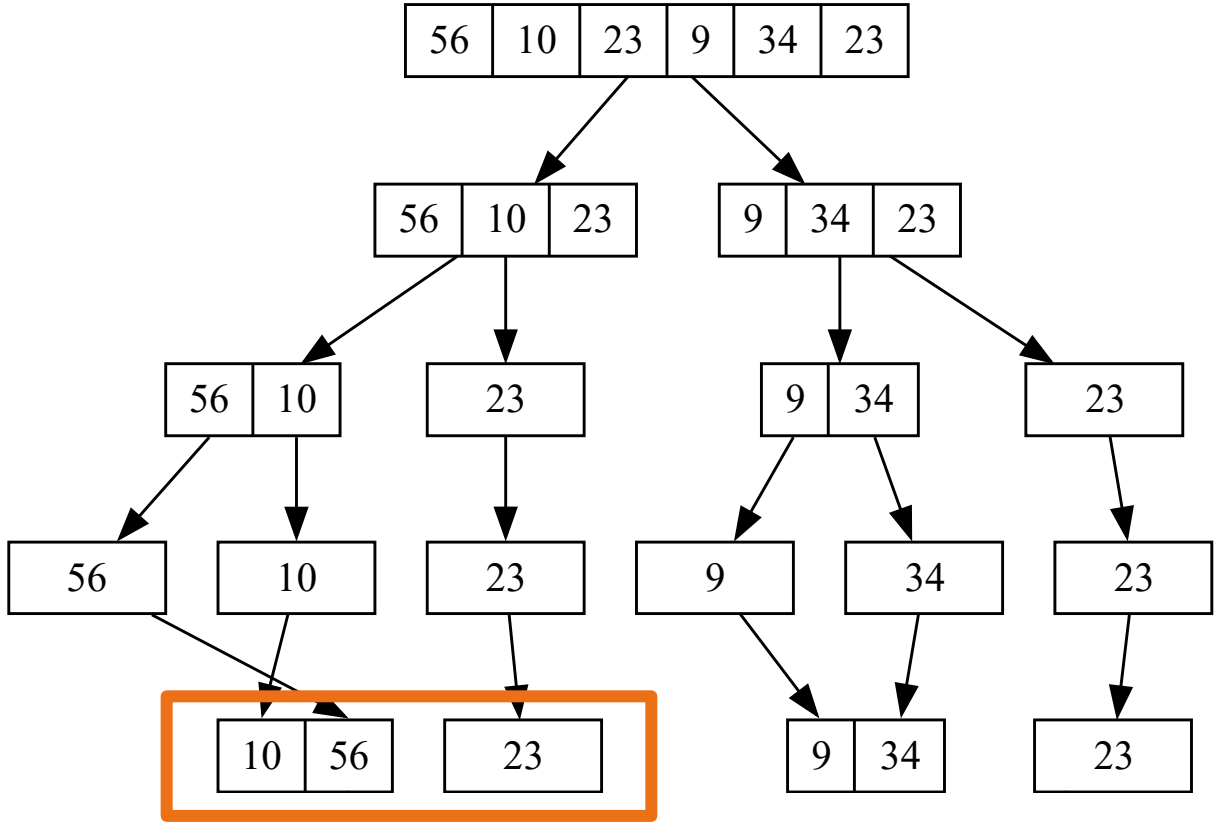


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

然後我們再把各種已解決的子問題 combine 起來

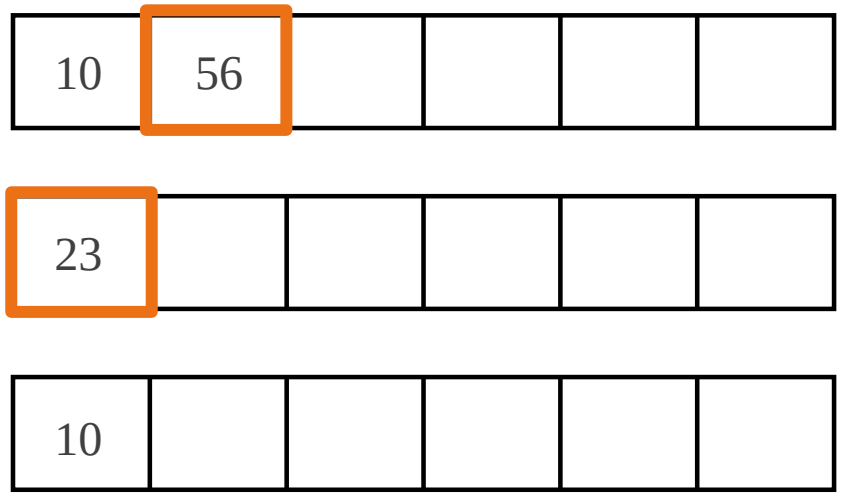


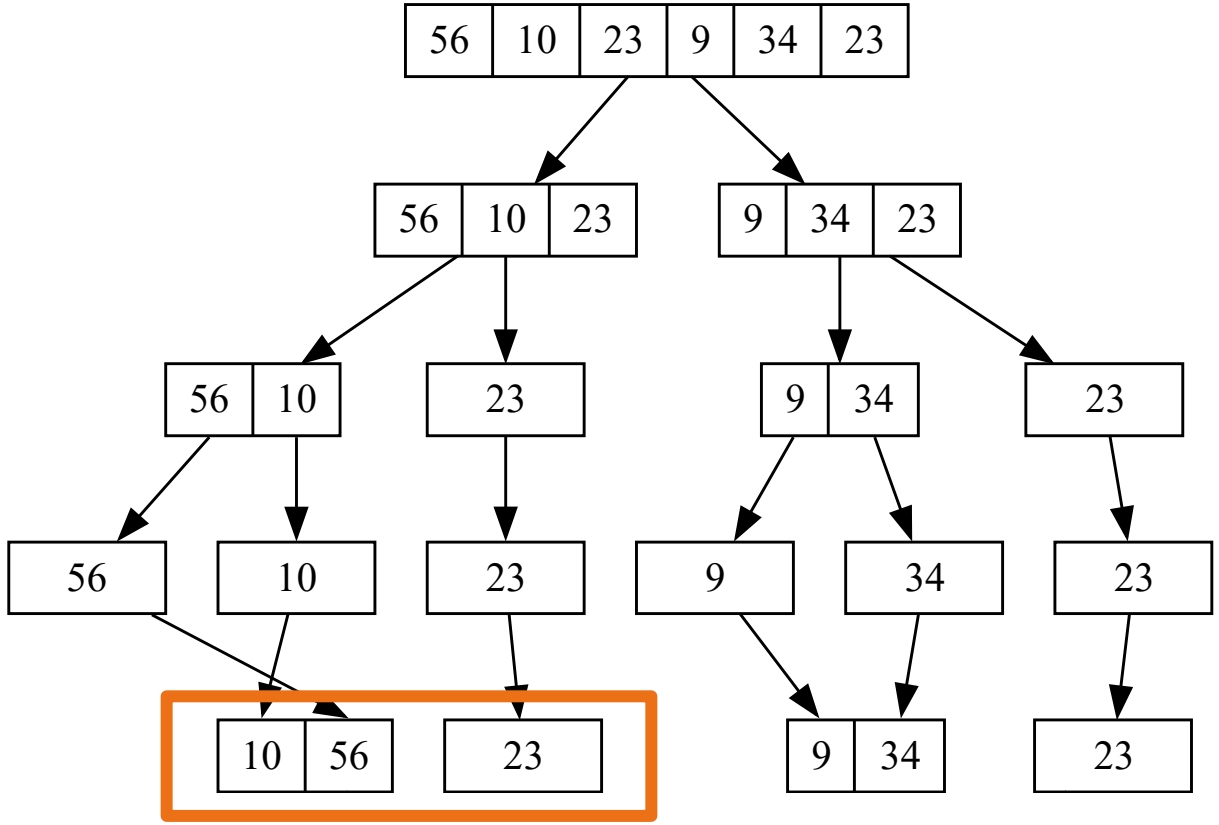


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

然後我們再把各種已解決的子問題 combine 起來

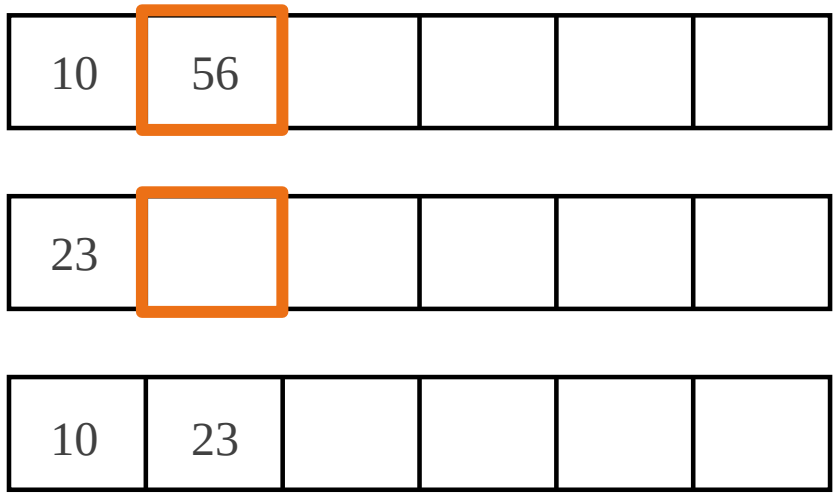


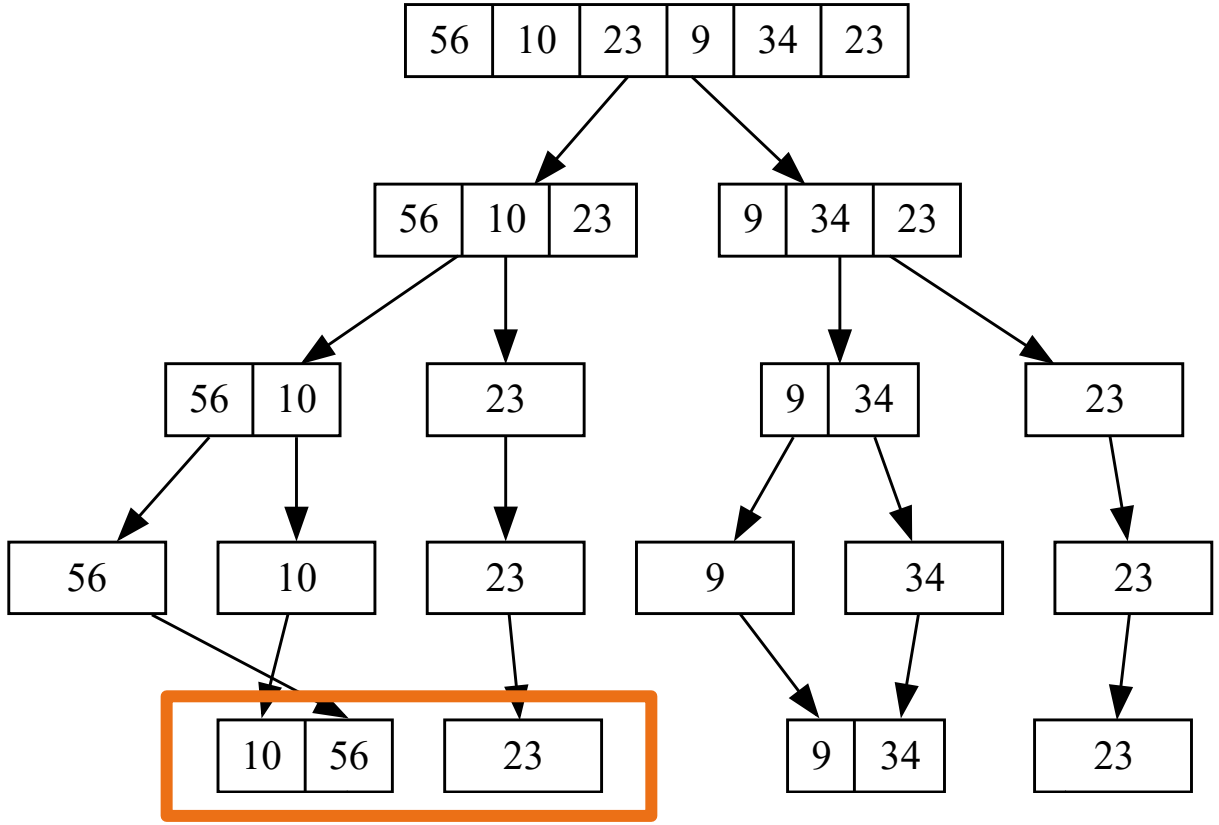


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

然後我們再把各種已解決的子問題 combine 起來

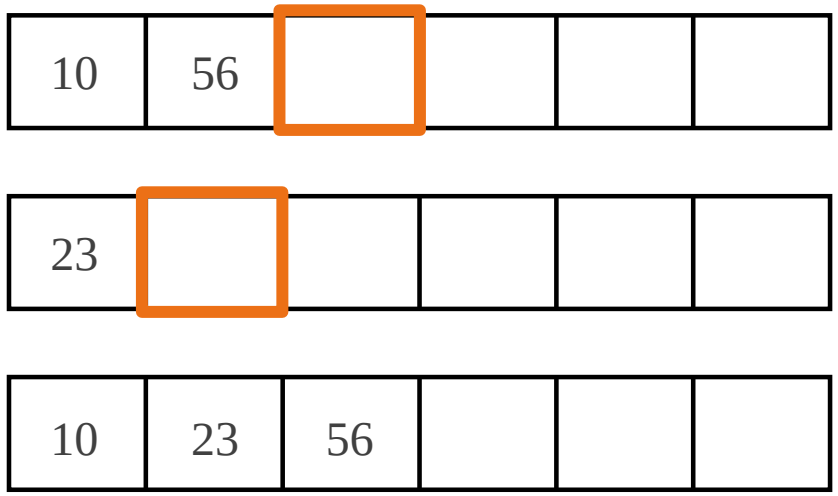


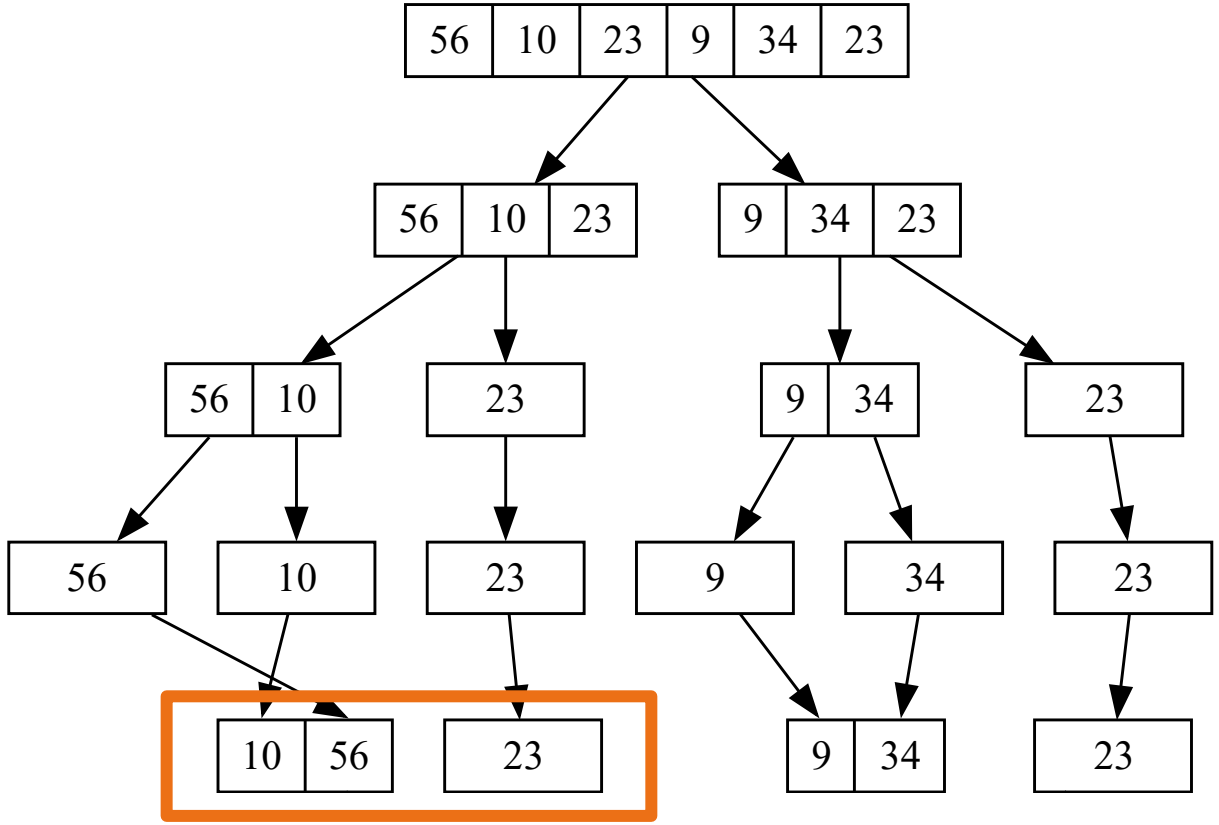


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

然後我們再把各種已解決的子問題 combine 起來

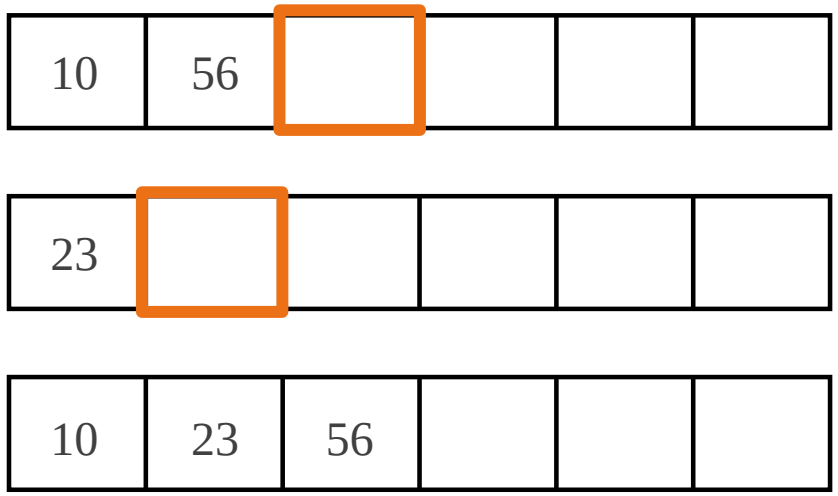


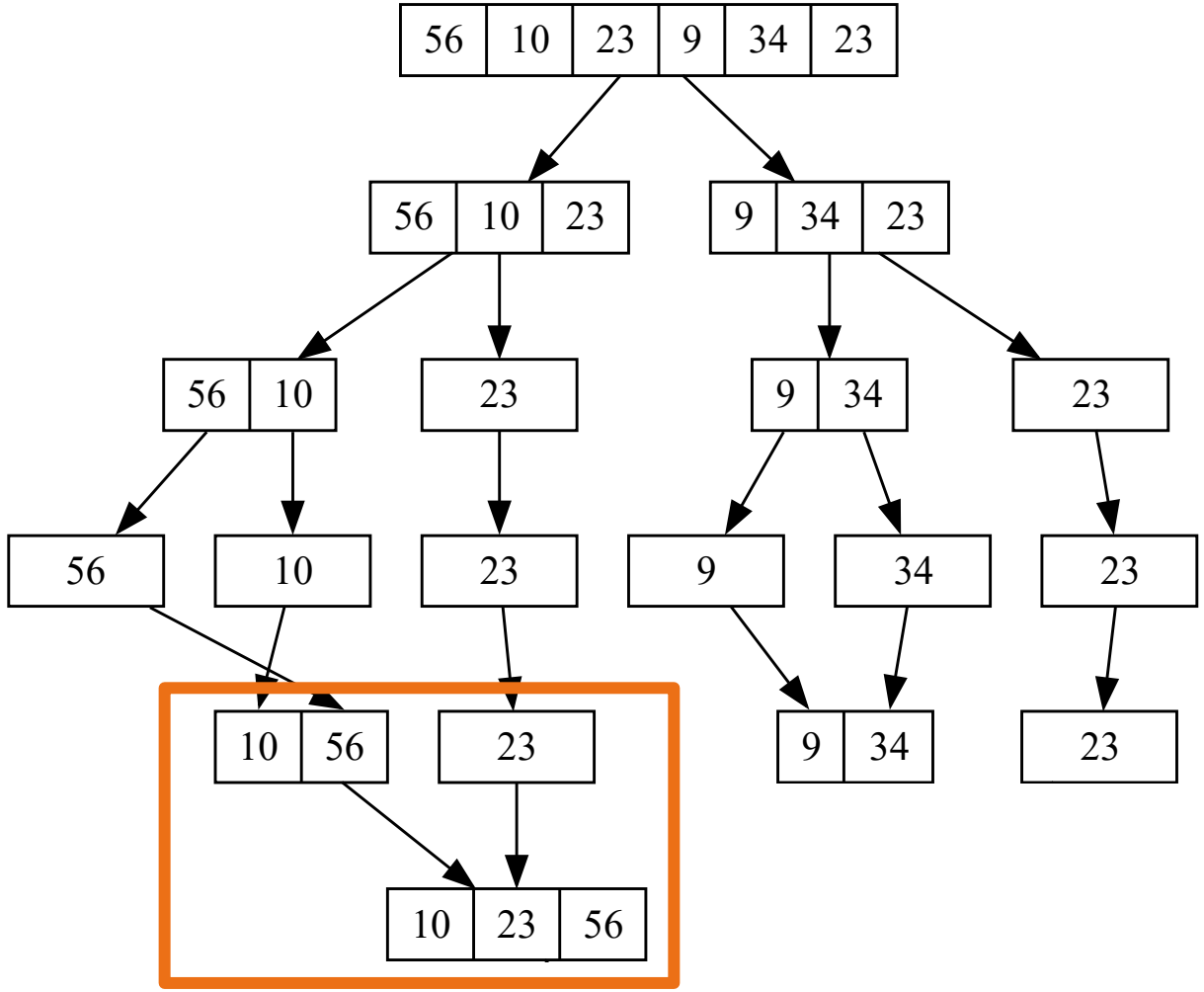


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

然後我們再把各種已解決的子問題 combine 起來

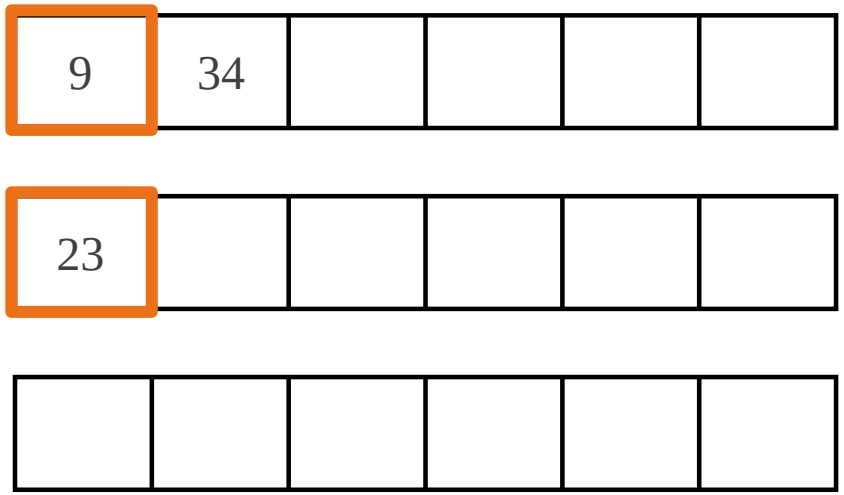


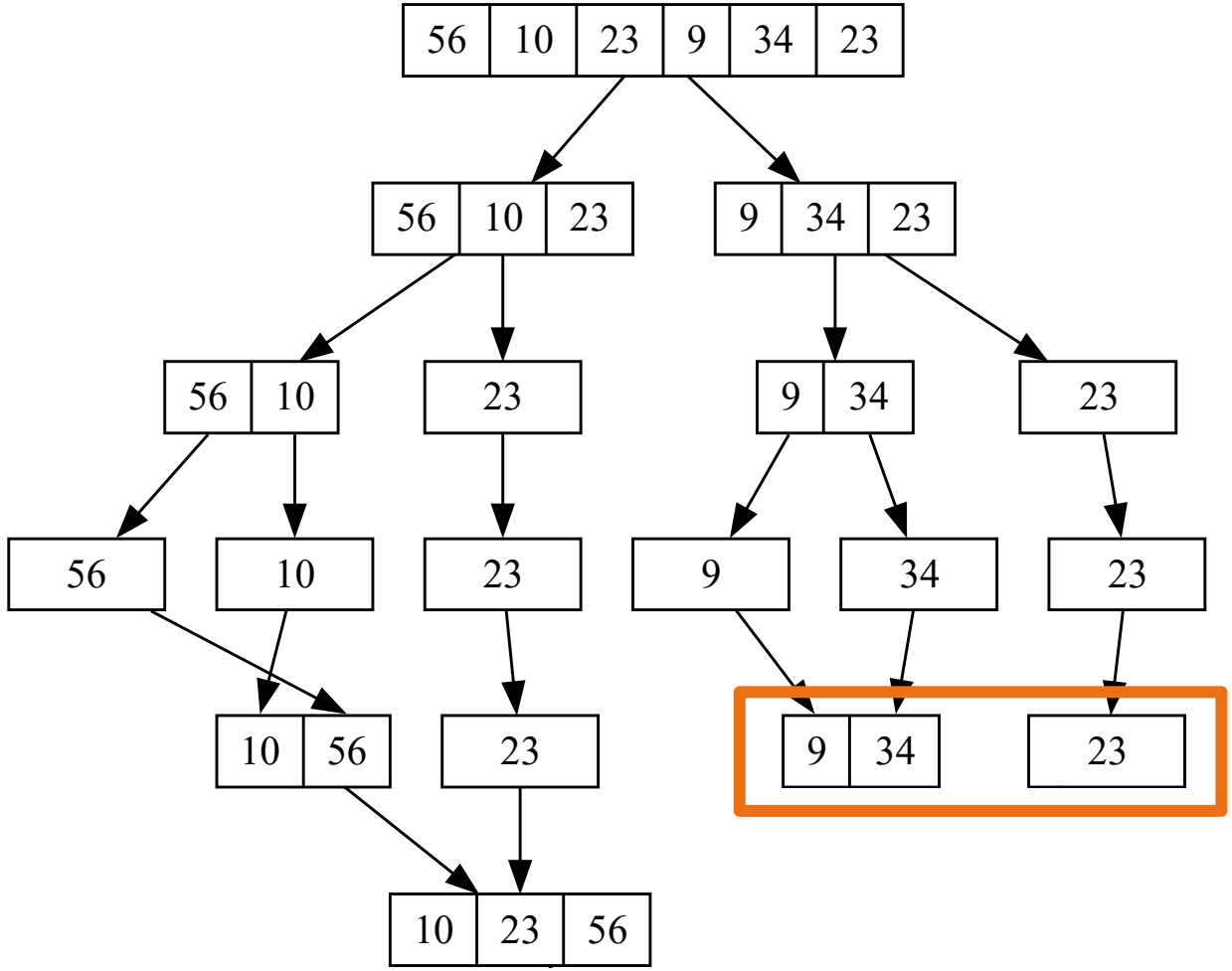


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

然後我們再把各種已解決的子問題 combine 起來

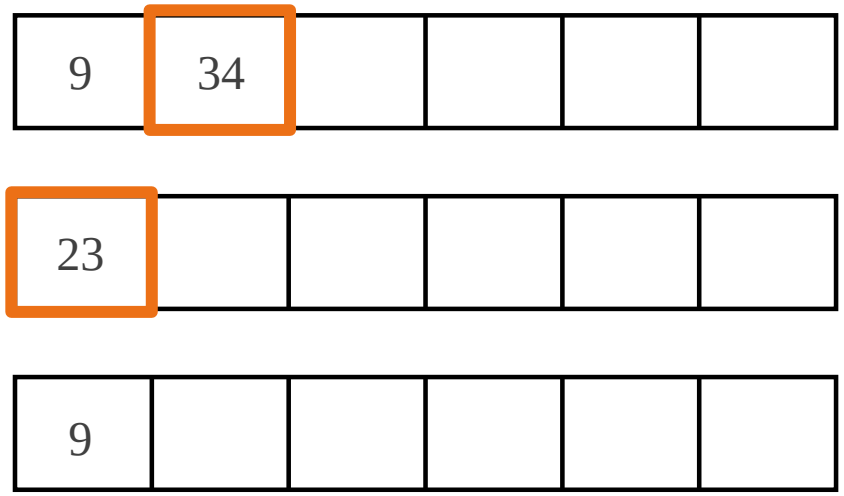


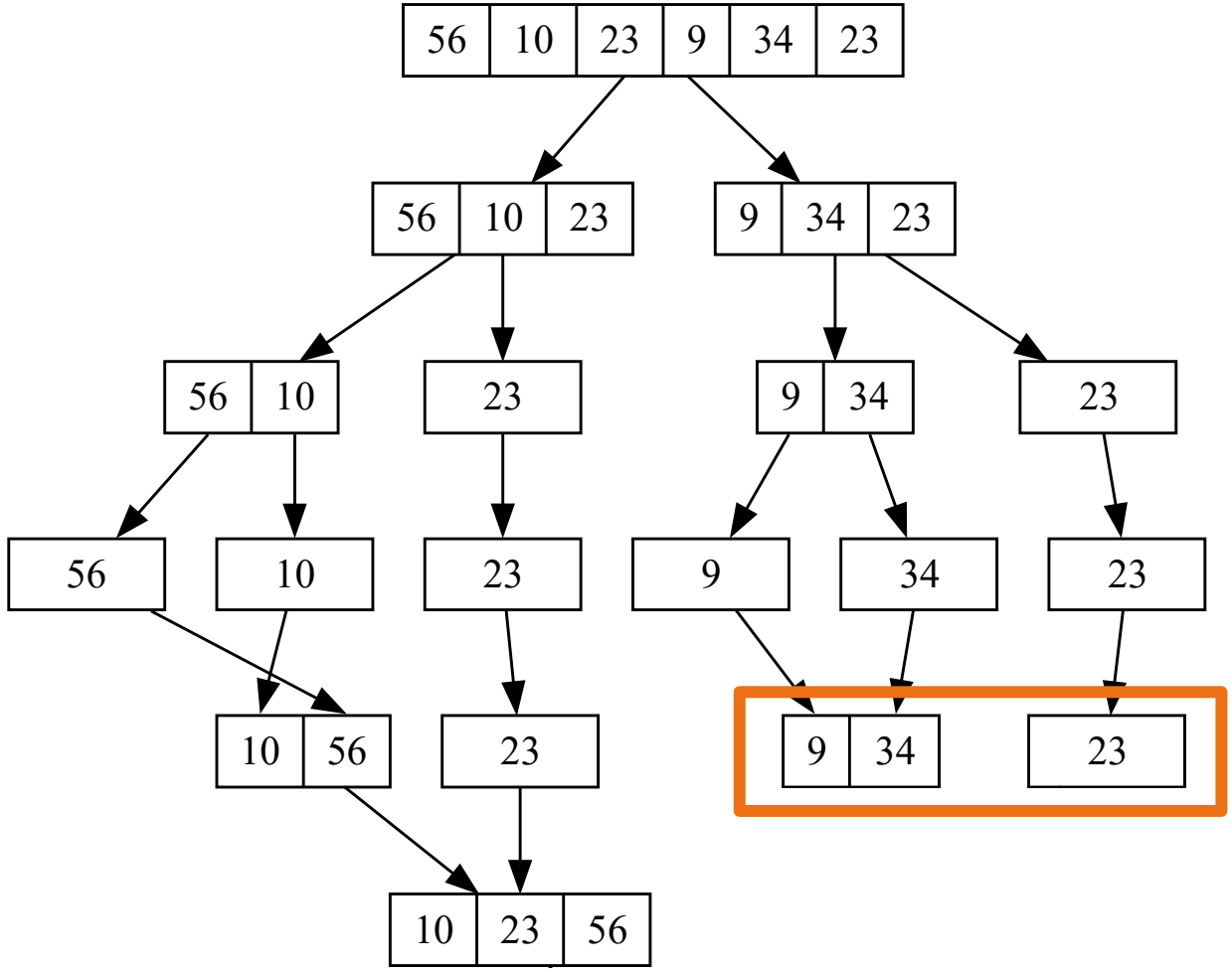


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

然後我們再把各種已解決的子問題 combine 起來

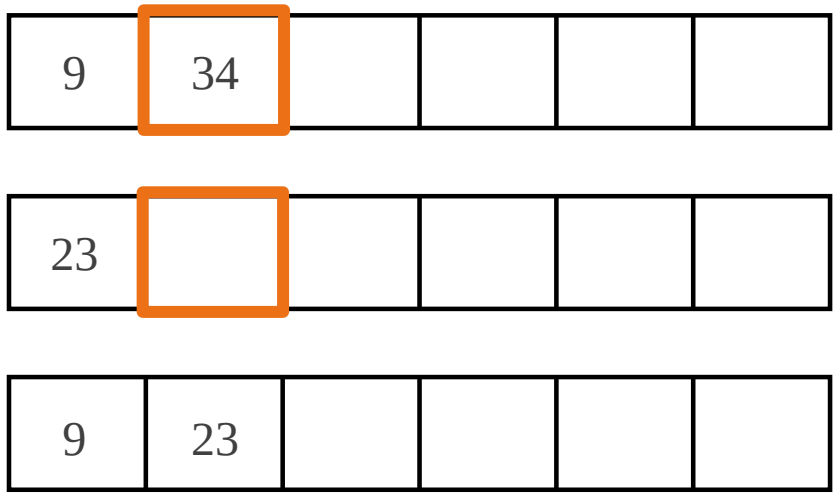


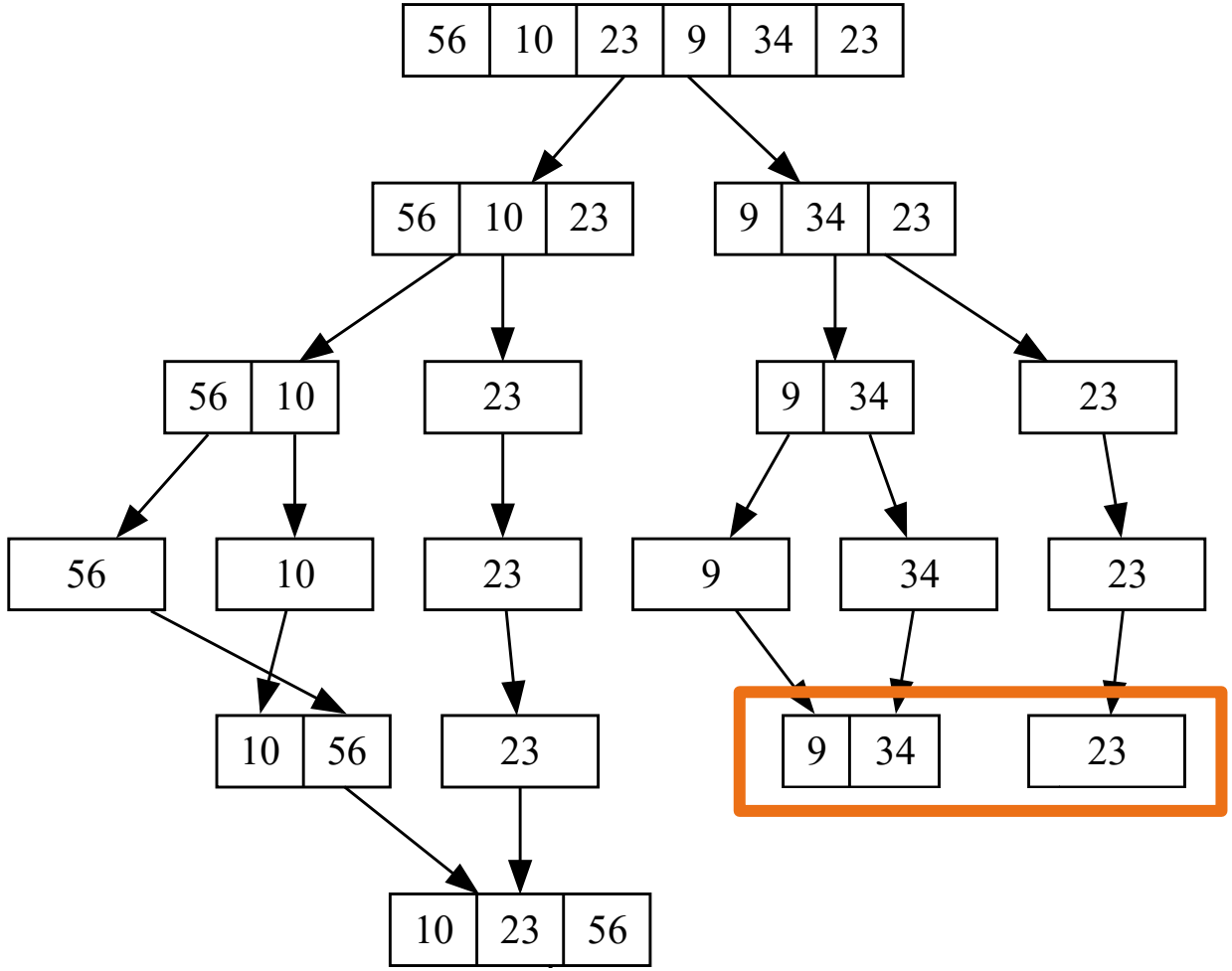


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

然後我們再把各種已解決的子問題 combine 起來

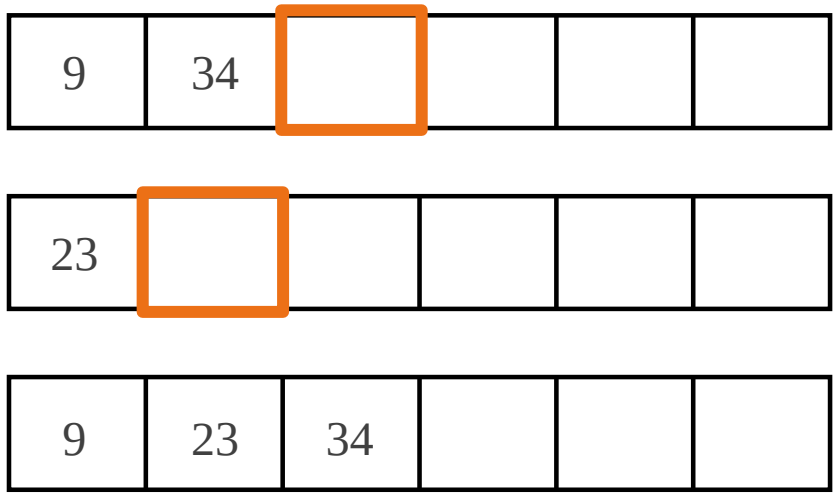


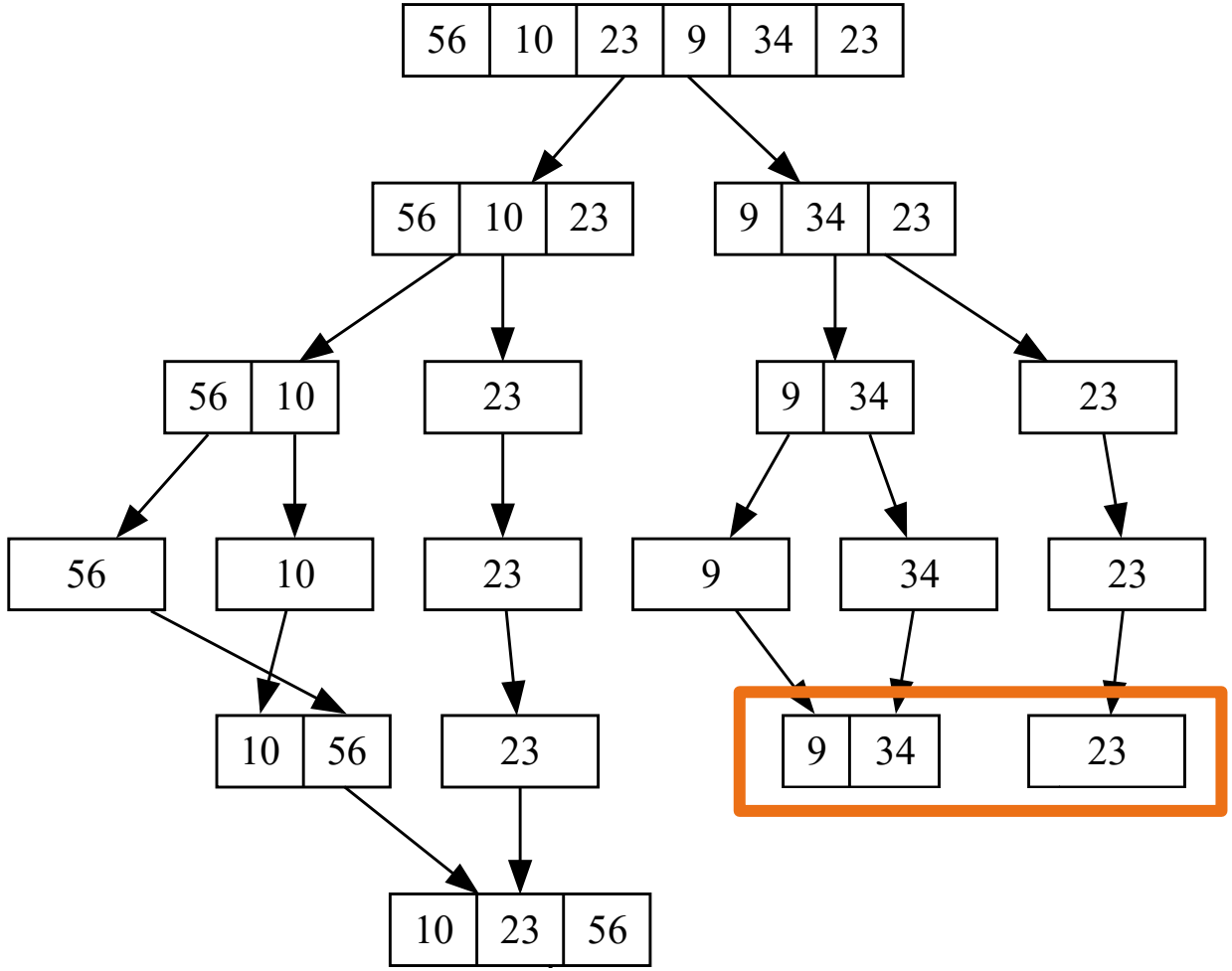


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

然後我們再把各種已解決的子問題 combine 起來

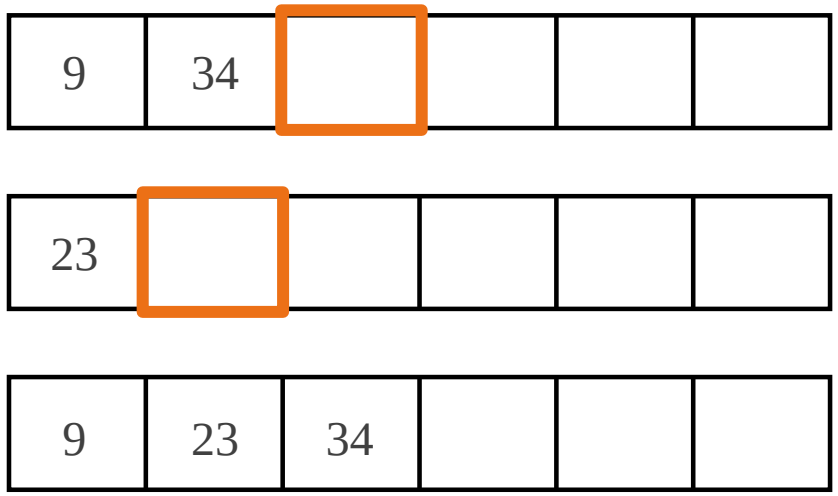


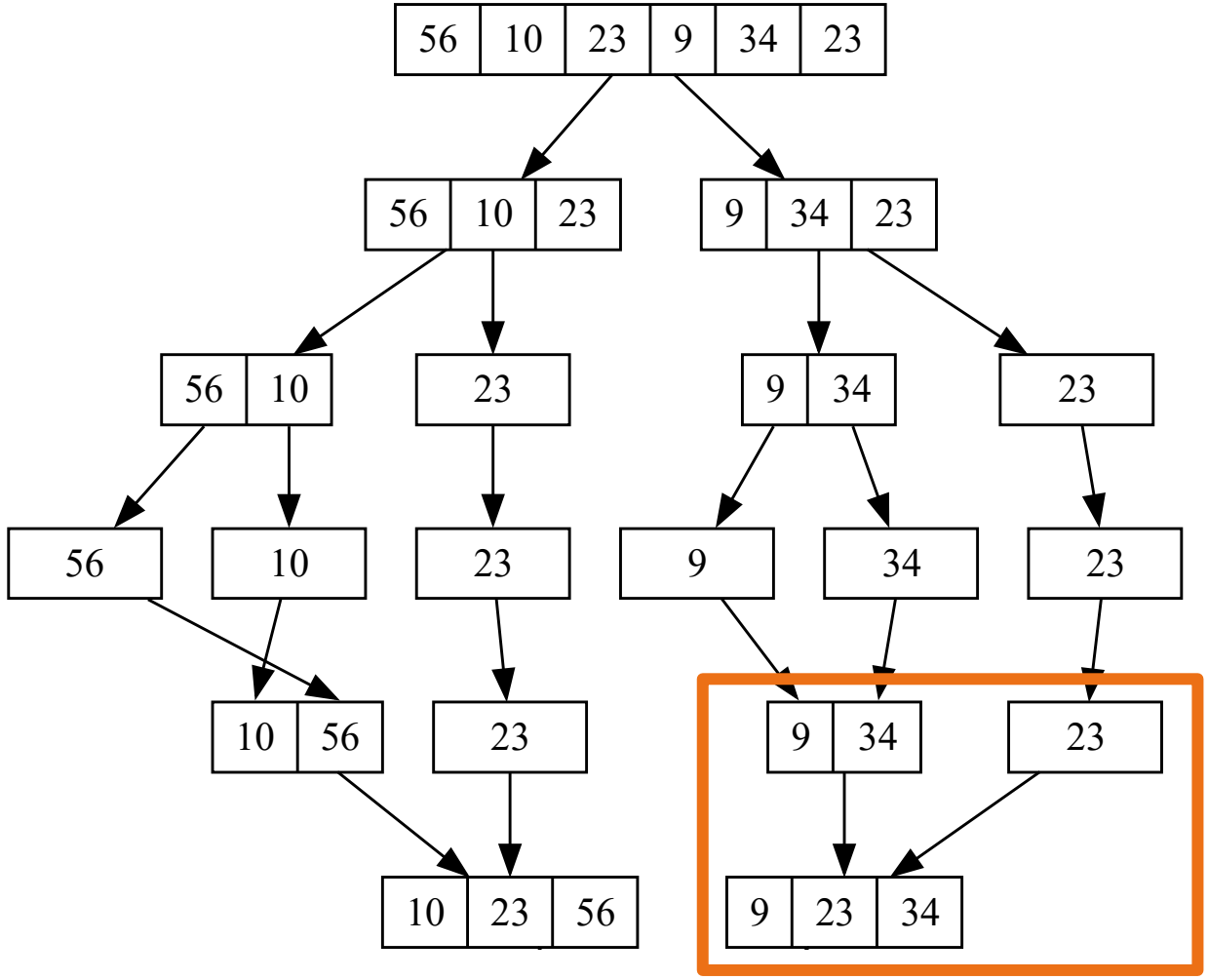


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

然後我們再把各種已解決的子問題 combine 起來

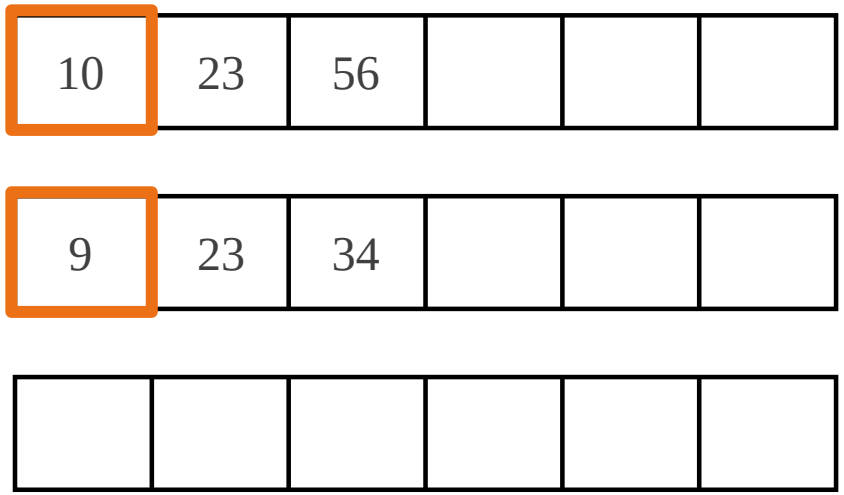


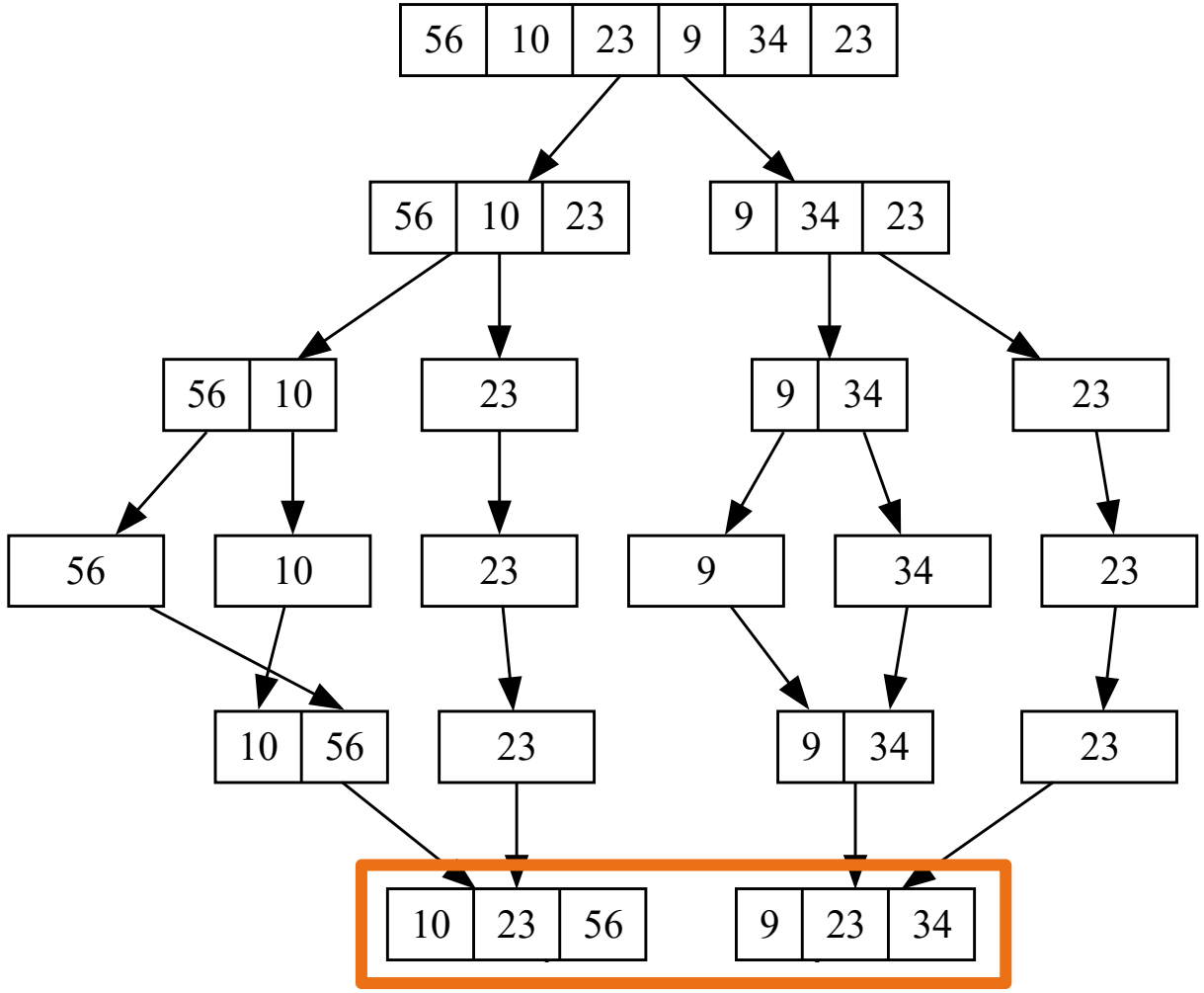


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

然後我們再把各種已解決的子問題 combine 起來

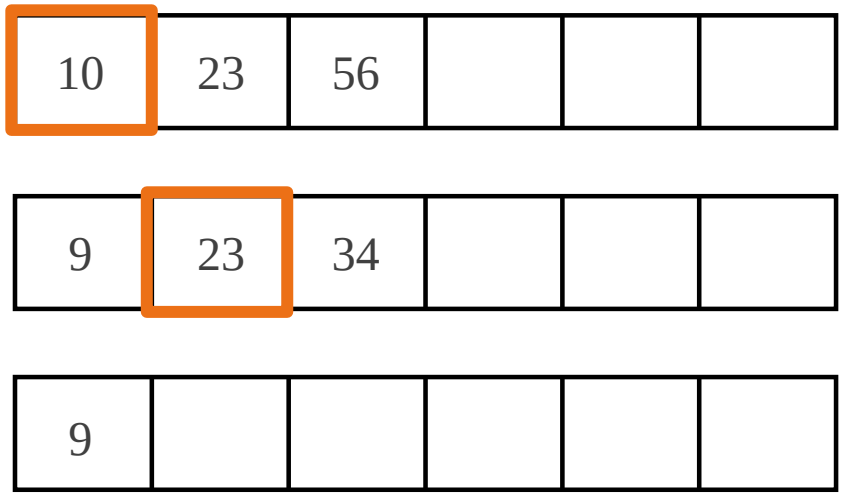


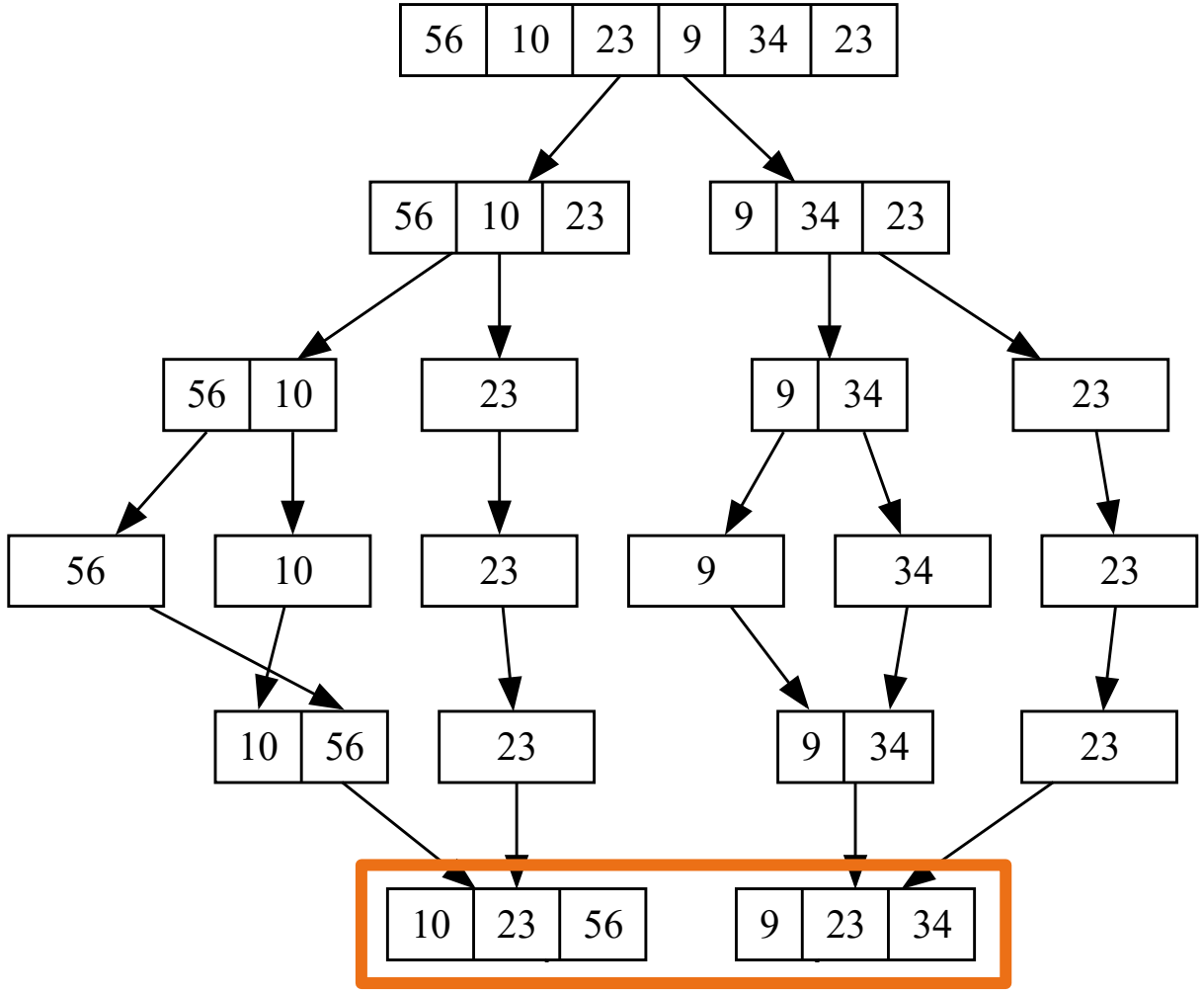


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

然後我們再把各種已解決的子問題 combine 起來

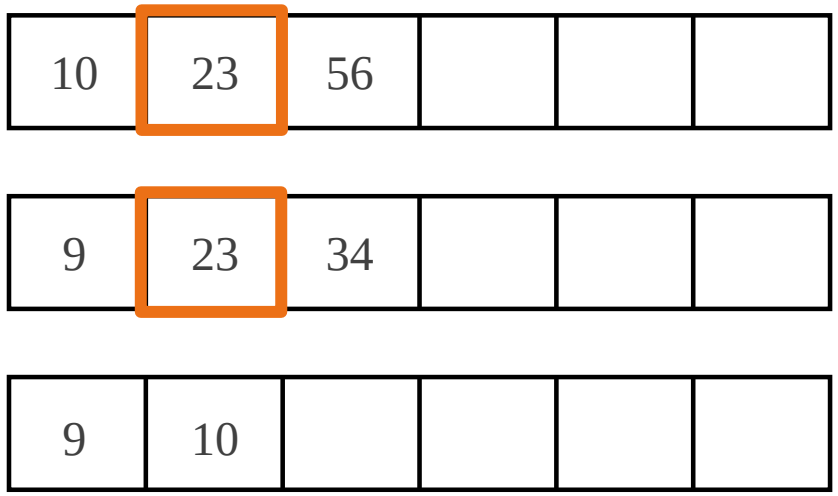


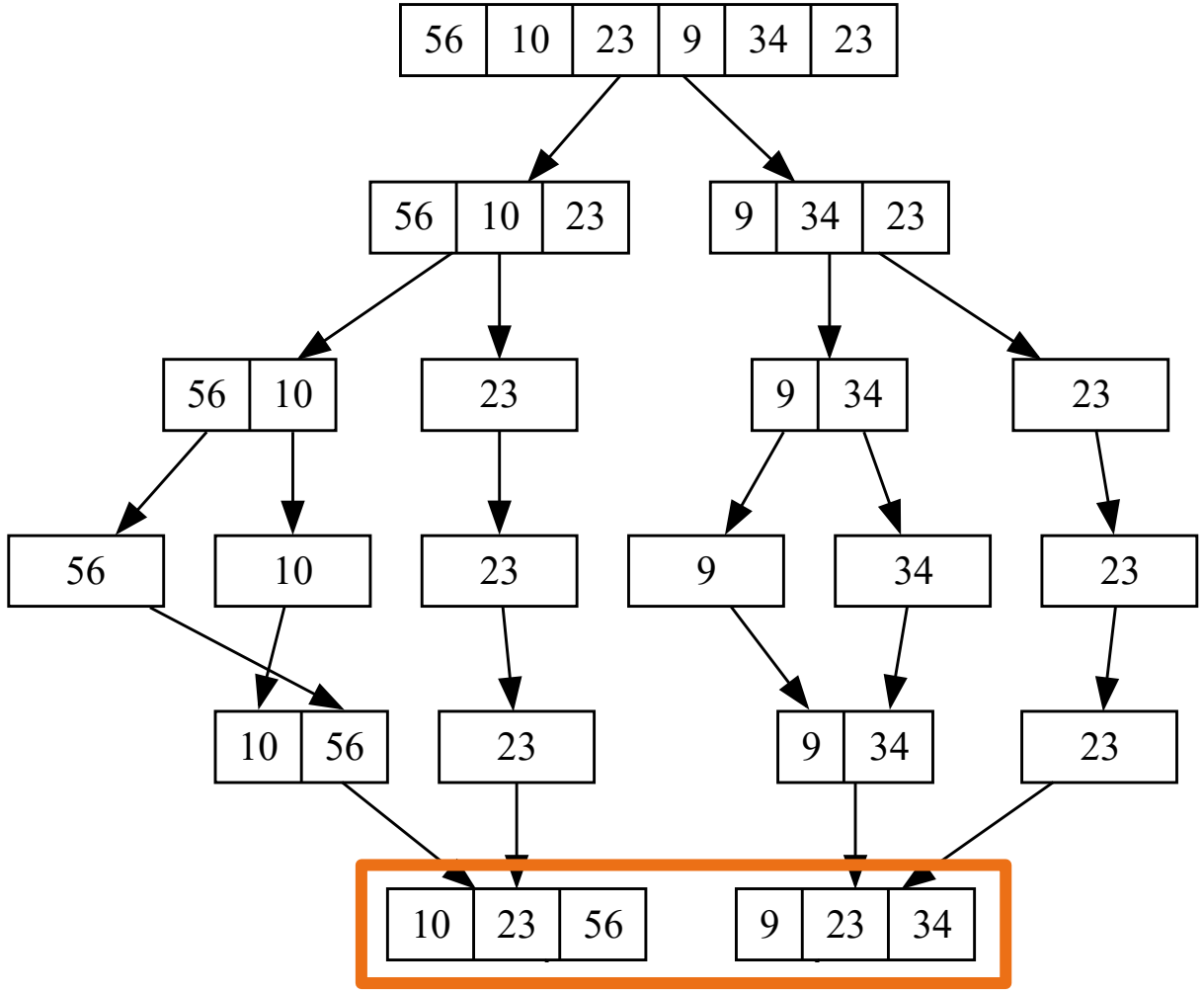


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

然後我們再把各種已解決的子問題 combine 起來

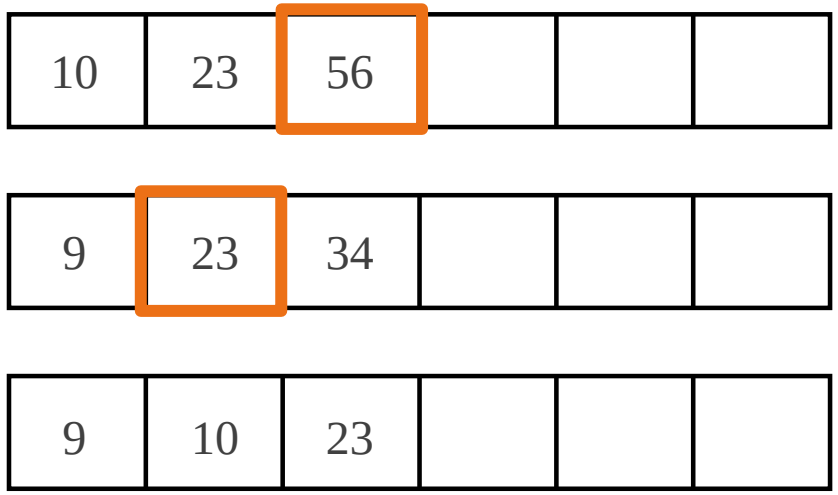


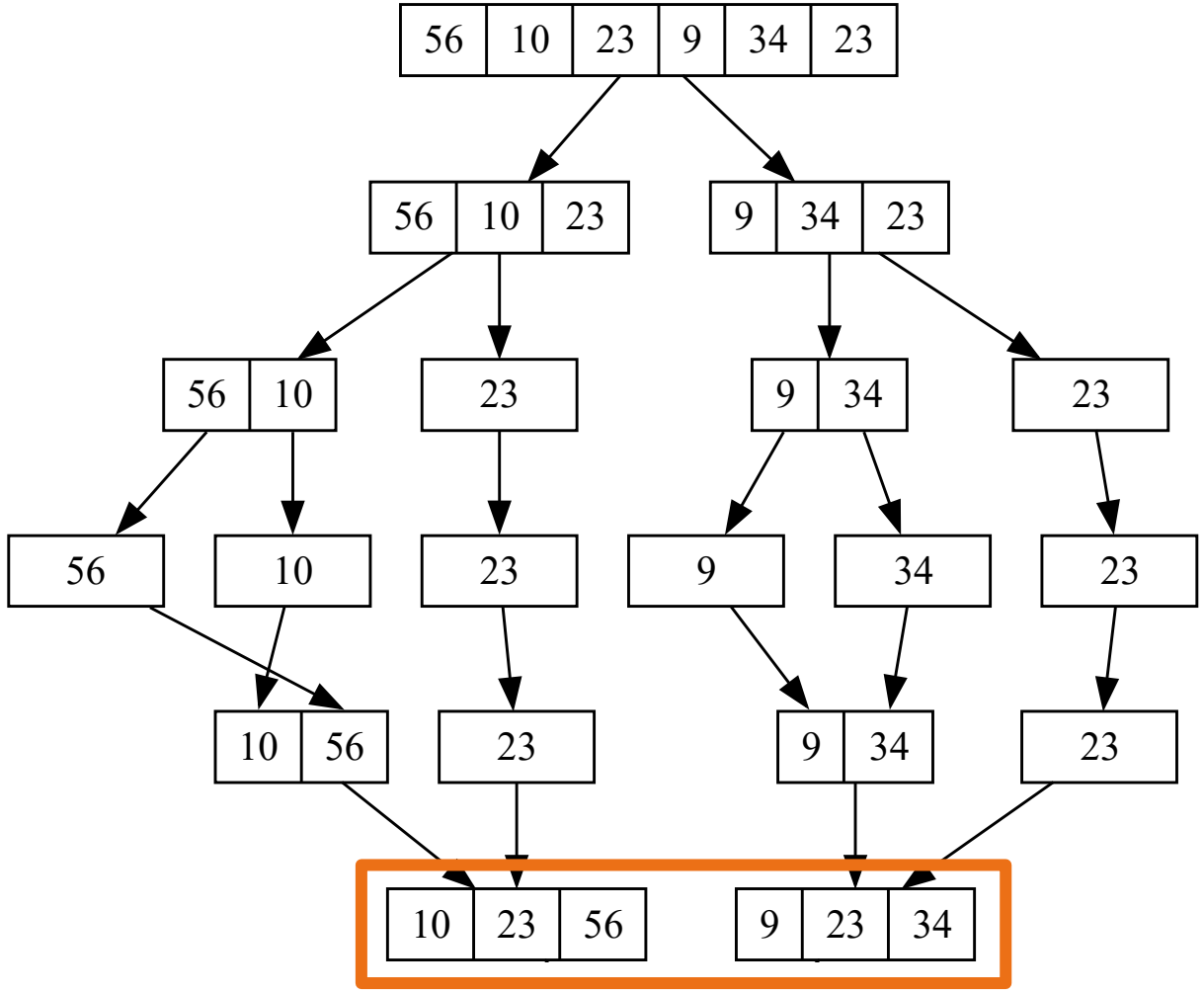


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

然後我們再把各種已解決的子問題 combine 起來

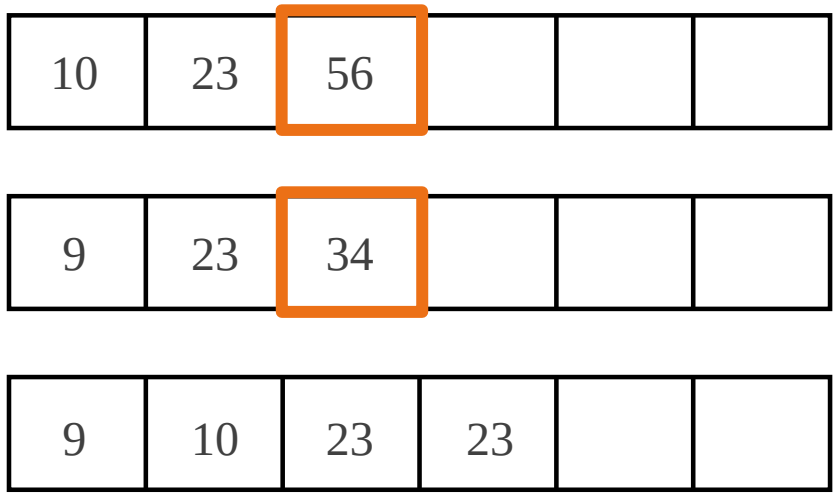


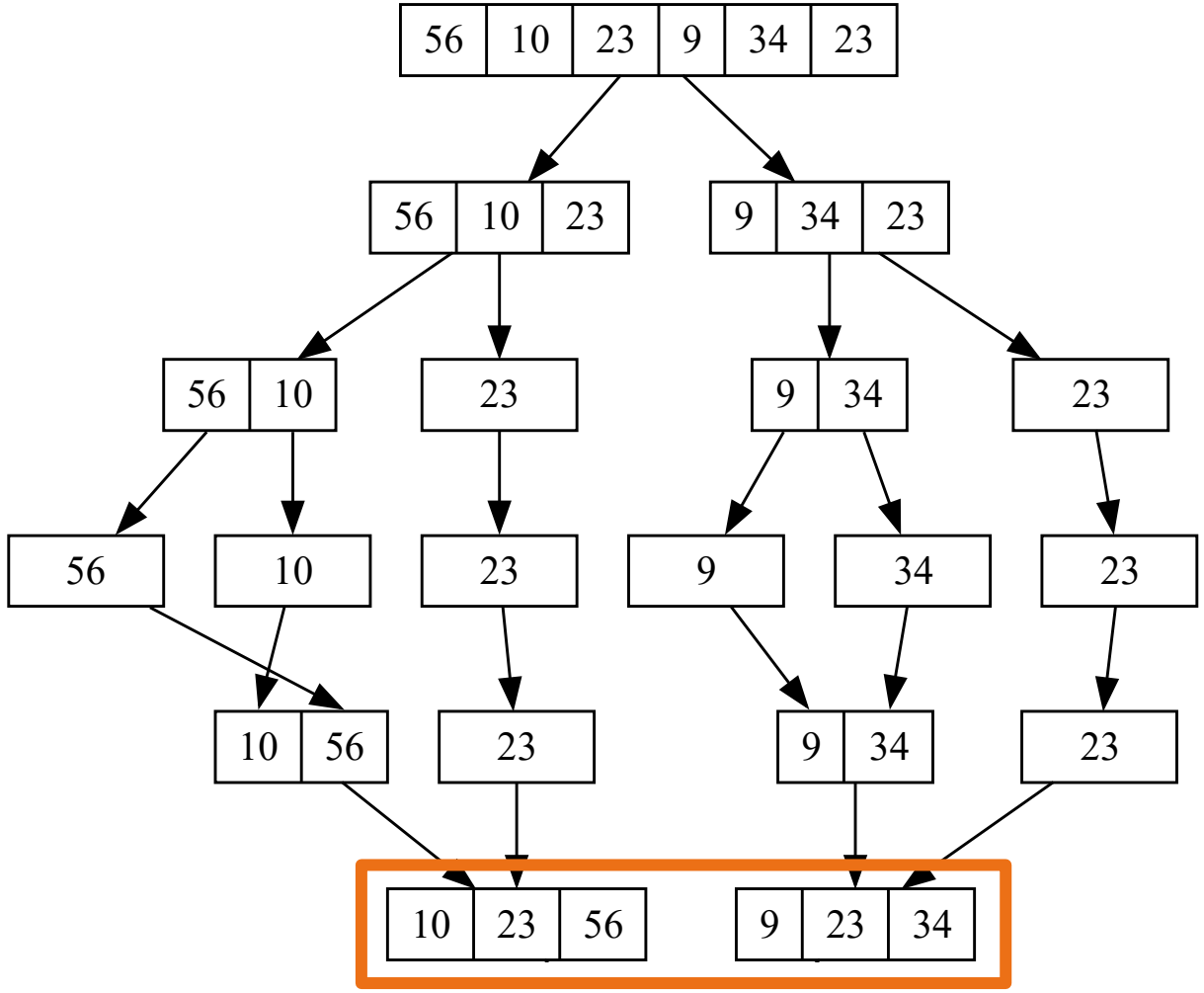


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

然後我們再把各種已解決的子問題 combine 起來

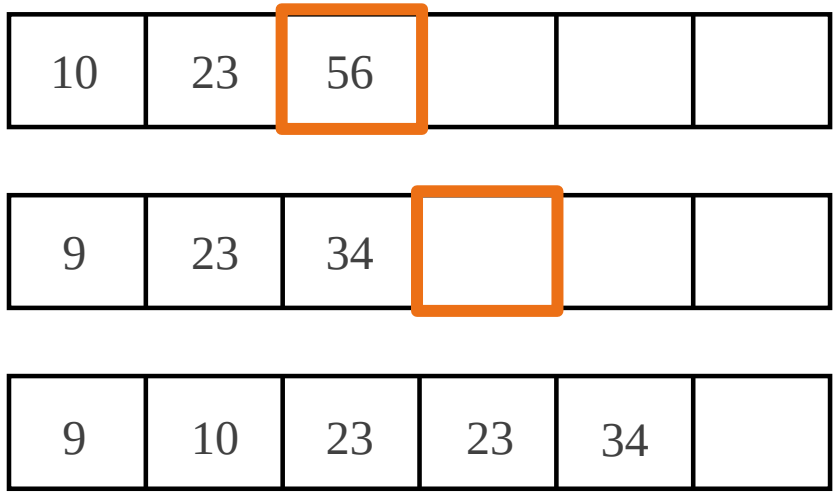


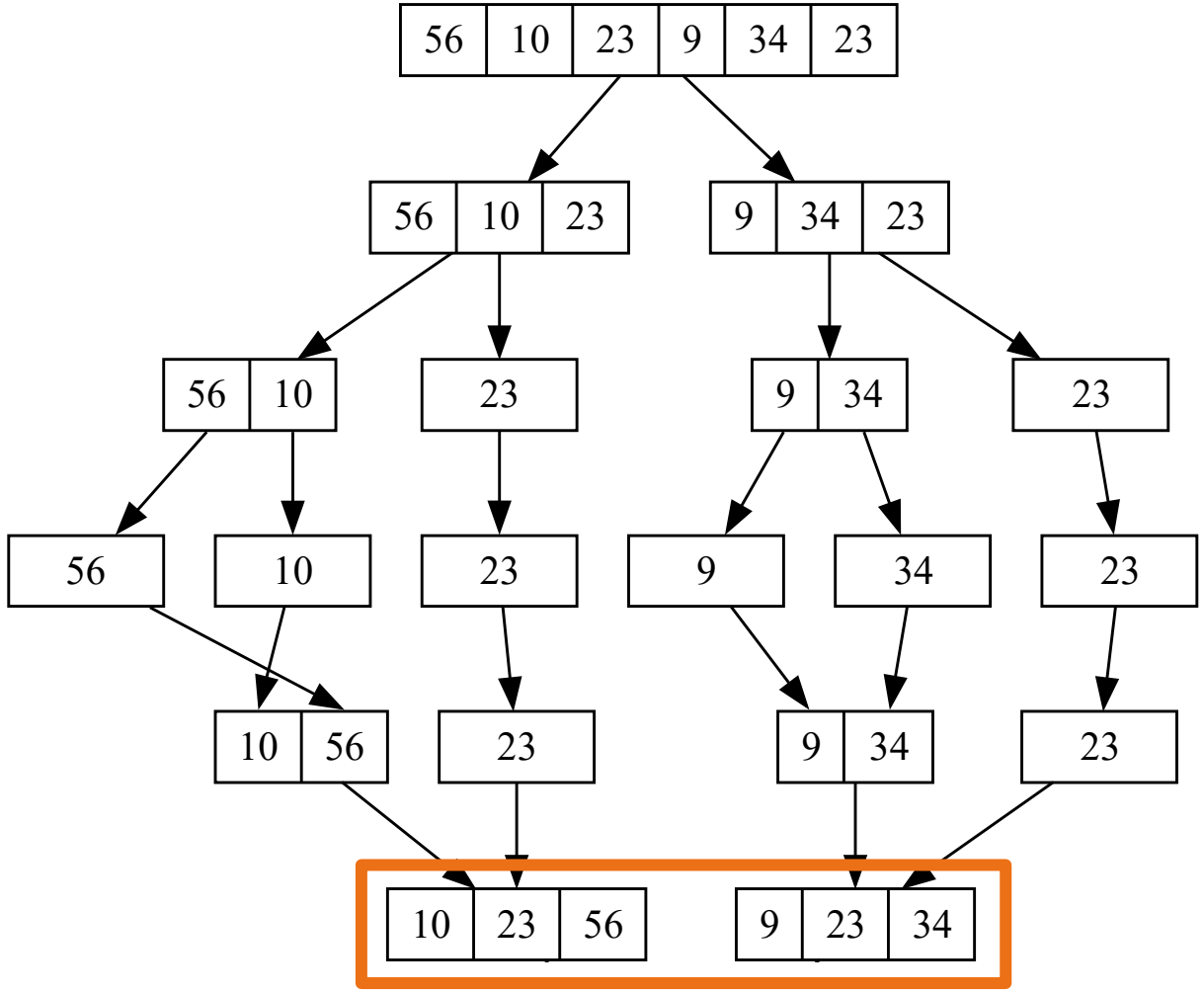


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

然後我們再把各種已解決的子問題 combine 起來

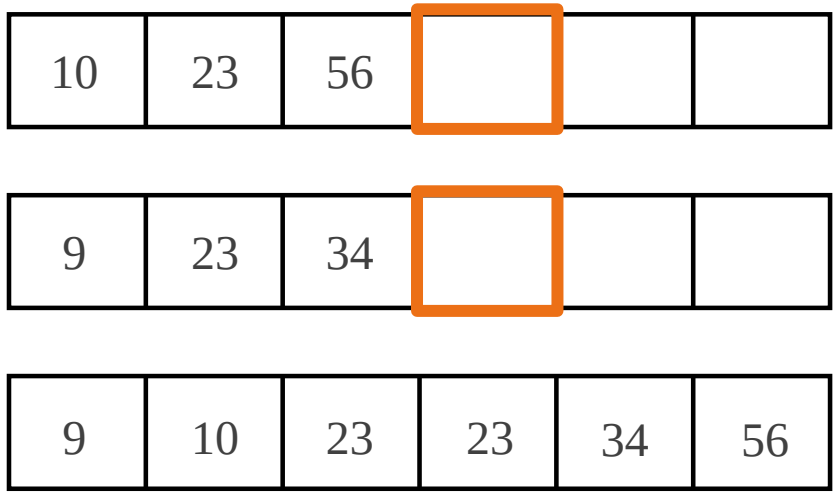


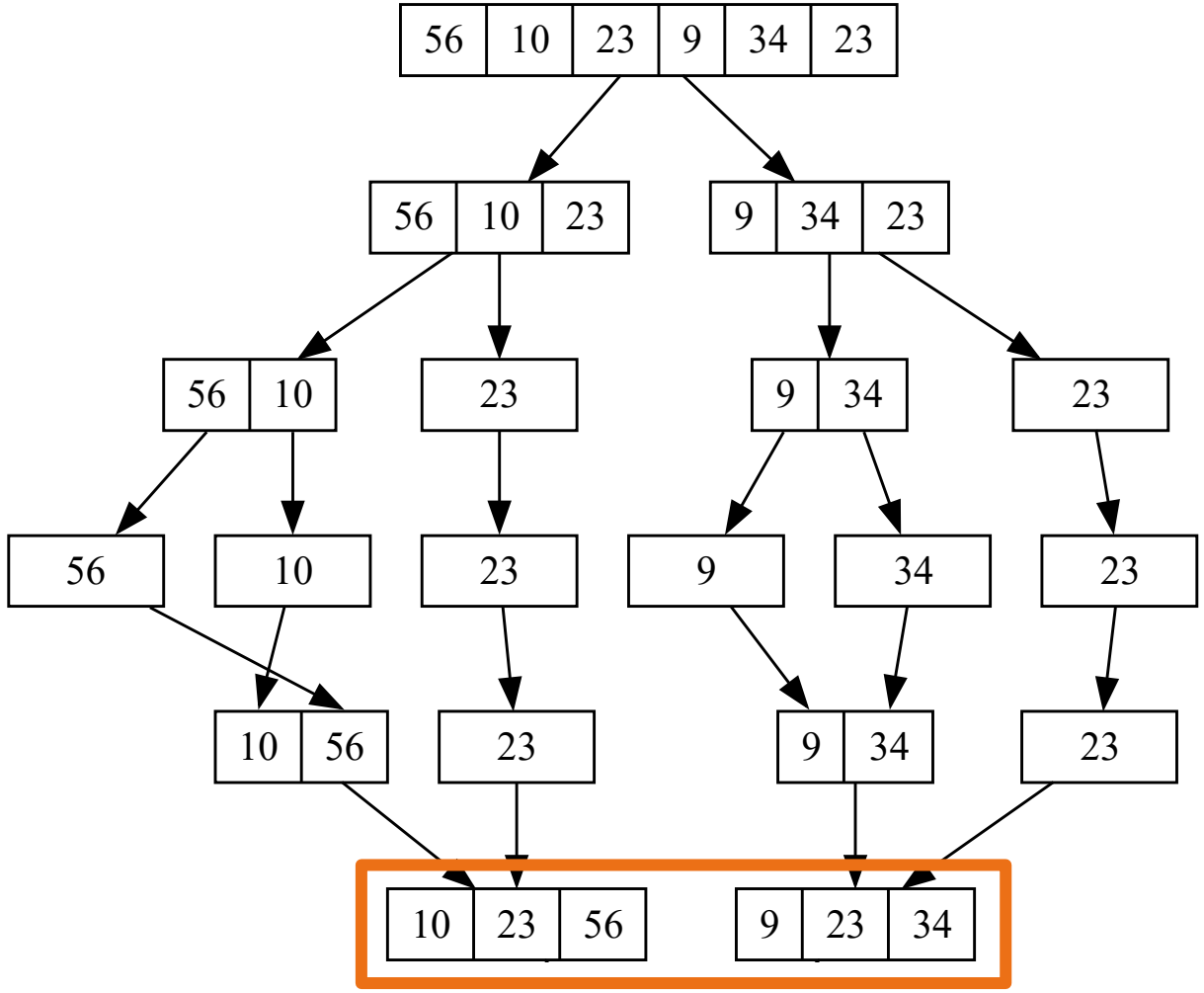


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

然後我們再把各種已解決的子問題 combine 起來

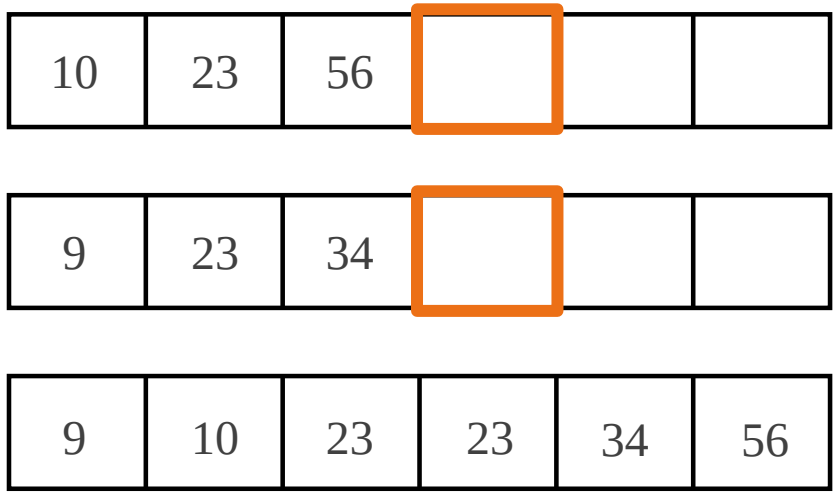


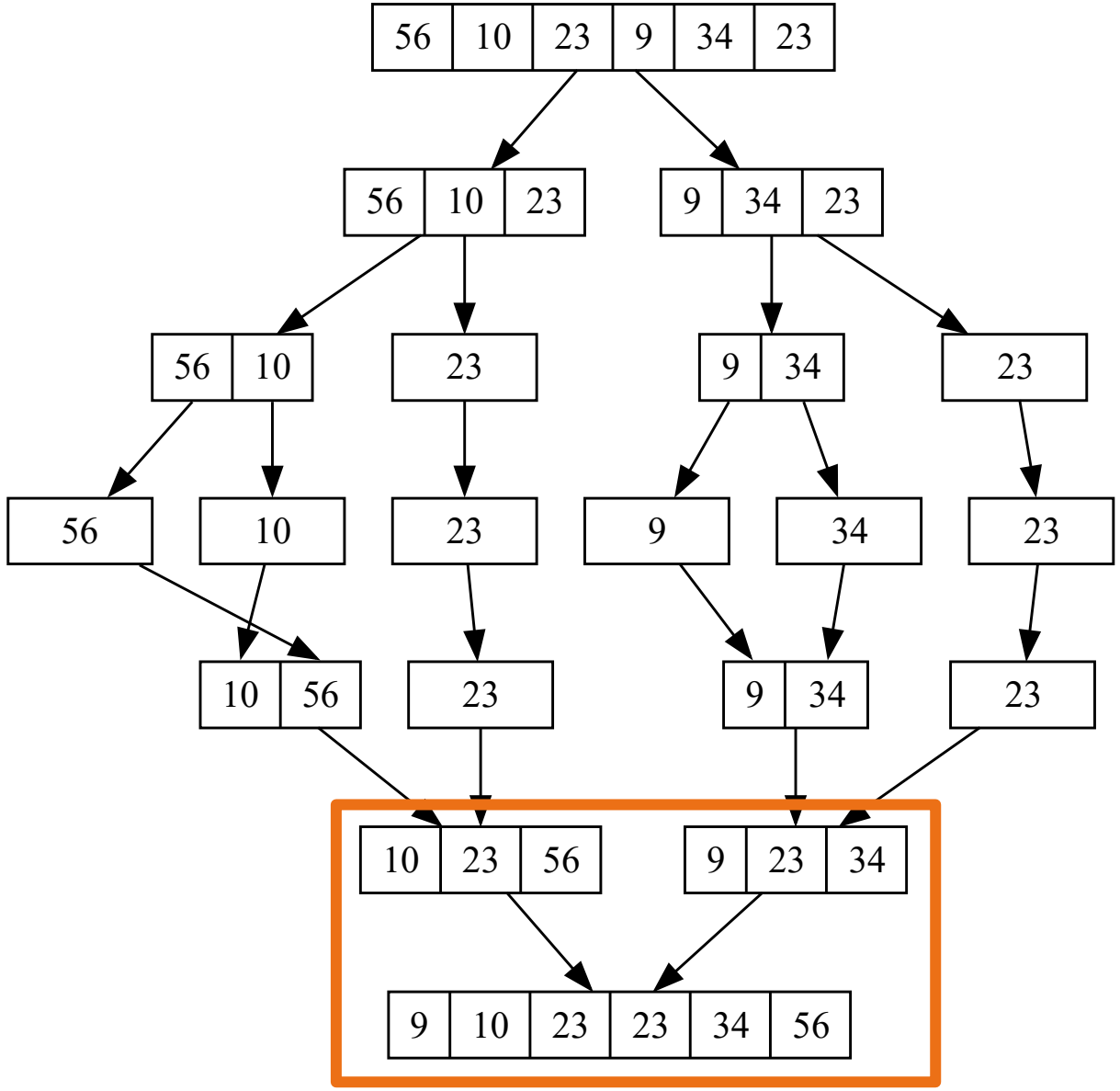


給定一組陣列 $\{56, 10, 23, 9, 34, 23\}$ 使用合併排序法排序

在 conquer 的部分中，合併排序雙指標走訪兩個目標陣列並將其排序成一個排序完成的陣列

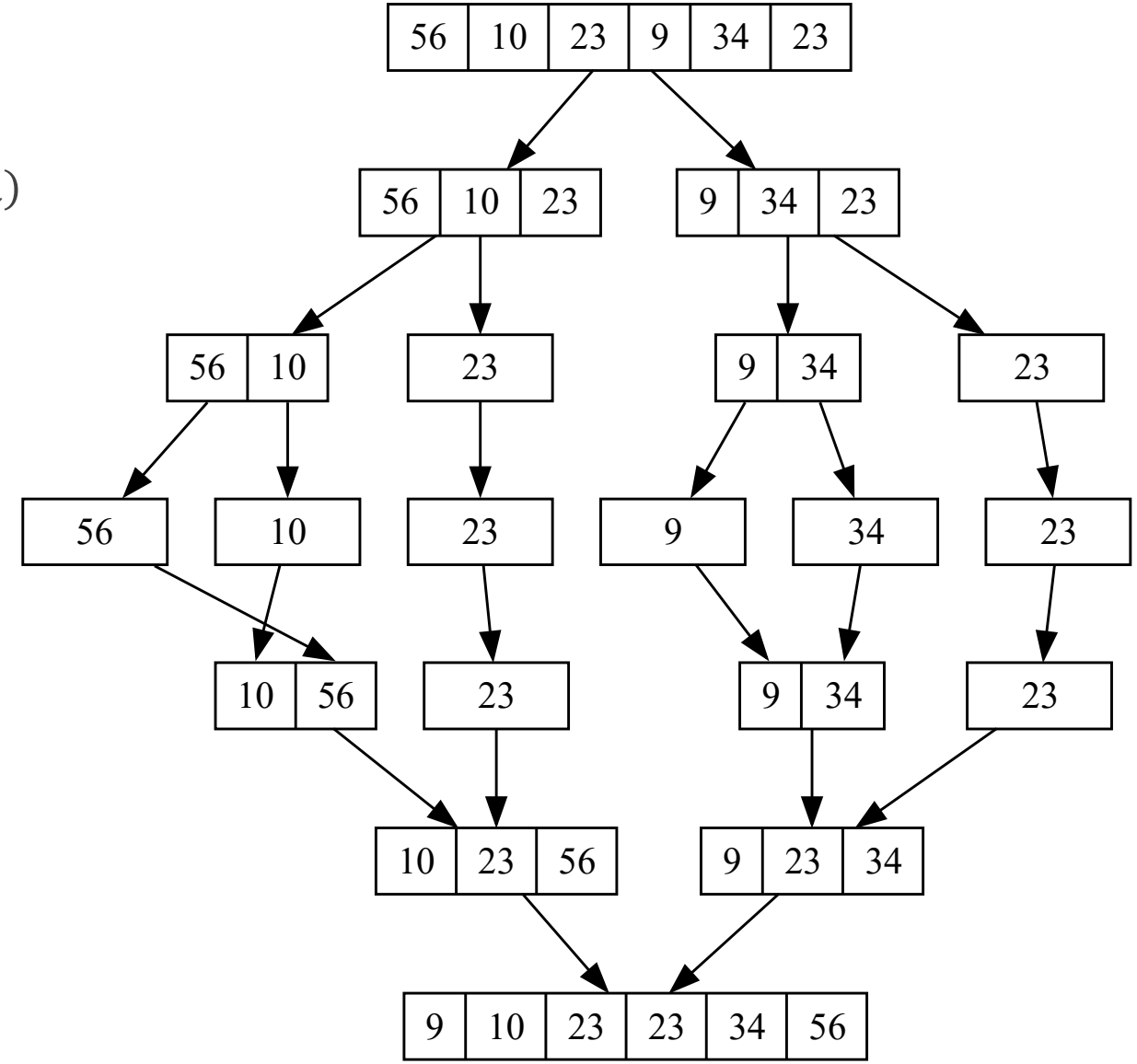
然後我們再把各種已解決的子問題 combine 起來





以結果論來看，我們透過分治法讓排序的過程降低成 $O(n)$

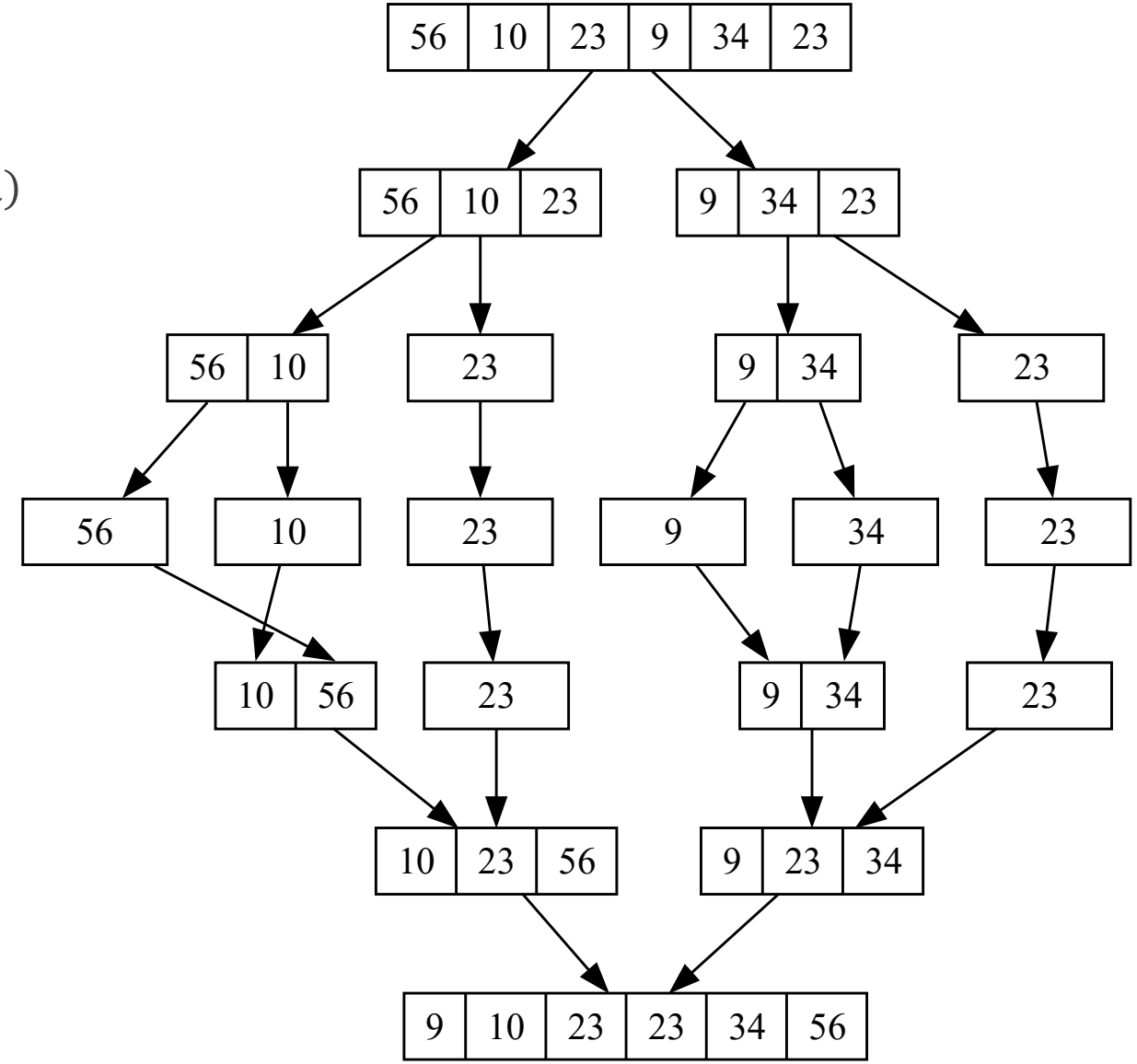
原因是 division 讓每個陣列元素只會出現一次！



以結果論來看，我們透過分治法讓排序的過程降低成 $O(n)$

原因是 division 讓每個陣列元素只會出現一次！

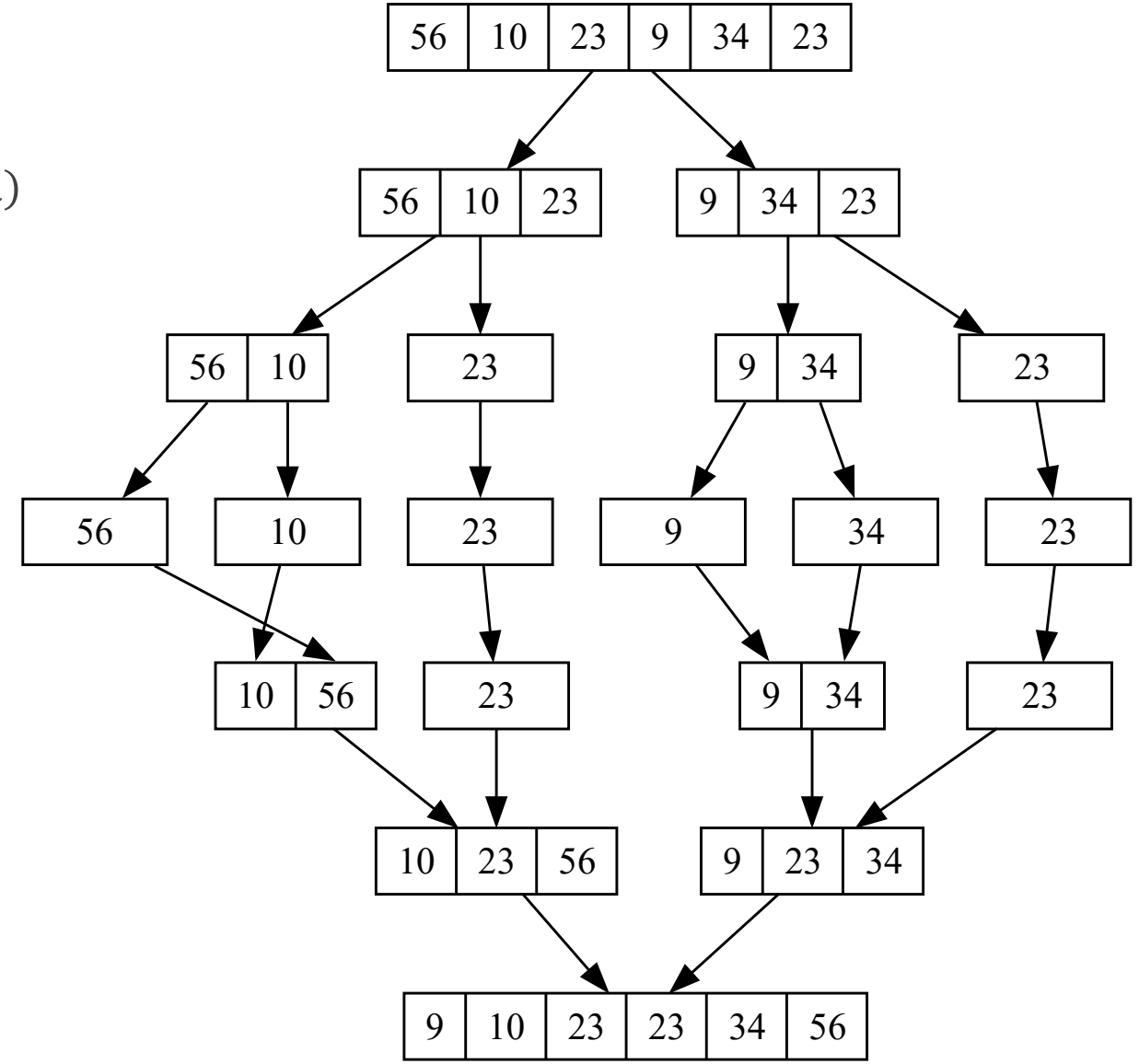
而因為我們 division 建構的好（切割一半），
所以我們在接下來的 conquer 中的工作次數也跟著減少



以結果論來看，我們透過分治法讓排序的過程降低成 $O(n)$

原因是 division 讓每個陣列元素只會出現一次！

而因為我們 division 建構的好（切割一半），
所以我們在接下來的 conquer 中的工作次數也跟著減少



照著這個脈絡下來，在實作合併排序法的時候就能根據

「遞迴分割」以及「$O(n)$ 排序」實作整個合併排序法

現在我們討論一下合併排序法的演算法細節

conquer

```
MERGE(A, left, mid, right):
    // conquer與combine
    temp = ARRAY OF SIZE (right - left + 1)
    i = left           // 左子陣列的起始索引
    j = mid + 1        // 右子陣列的起始索引
    k = 0              // 排序陣列的索引

    // 比較左、右子陣列的元素，將較小者加入臨時陣列
    WHILE i <= mid AND j <= right:
        IF A[i] <= A[j]:
            temp[k] = A[i]
            i = i + 1
        ELSE:
            temp[k] = A[j]
            j = j + 1
        k = k + 1

    // 將左子陣列剩餘的元素加入排序陣列
    WHILE i <= mid:
        temp[k] = A[i]
        i = i + 1
        k = k + 1

    // 將右子數列剩餘的元素加入排序陣列
    WHILE j <= right:
        temp[k] = A[j]
        j = j + 1
        k = k + 1

    // 將合併後的元素覆蓋回原陣列 A
    FOR p FROM 0 TO (k - 1):
        A[left + p] = temp[p]
```

divide, combine

```
MERGE_SORT(A, left, right):
    IF left >= right:
        RETURN  // 基本情況：子陣列長度為 1 或 0
    mid = left + (right - left) / 2
    MERGE_SORT(A, left, mid)        // 遞歸排序左半部分
    MERGE_SORT(A, mid + 1, right)   // 遞歸排序右半部分
    MERGE(A, left, mid, right)      // 合併已解決問題的子陣列

```

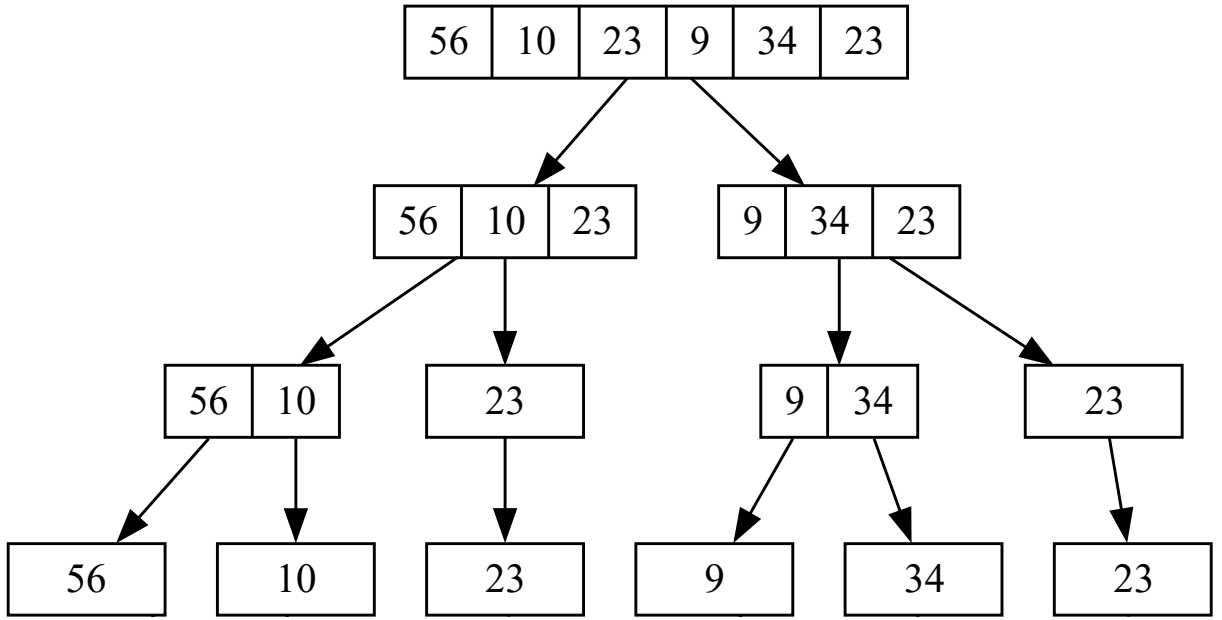
現在我們討論一下合併排序法的演算法細節

conquer

```
MERGE(A, left, mid, right):
    // conquer與combine
    temp = ARRAY OF SIZE (right - left + 1)
    i = left           // 左子陣列的起始索引
    j = mid + 1        // 右子陣列的起始索引
    k = 0              // 排序陣列的索引

    // 比較左、右子陣列的元素，將較小者加入臨時陣列
    WHILE i <= mid AND j <= right:
        IF A[i] <= A[j]:
            temp[k] = A[i]
            i = i + 1
        ELSE:
            temp[k] = A[j]
            j = j + 1
        k = k + 1

    // 將左子陣列剩餘的元素加入排序陣列
    WHILE i <= mid:
        temp[k] = A[i]
        i = i + 1
        k = k + 1

    // 將右子數列剩餘的元素加入排序陣列
    WHILE j <= right:
        temp[k] = A[j]
        j = j + 1
        k = k + 1

    // 將合併後的元素覆蓋回原陣列 A
    FOR p FROM 0 TO (k - 1):
        A[left + p] = temp[p]
```

divide, combine

```
MERGE_SORT(A, left, right):
    IF left >= right:
        RETURN  // 基本情況：子陣列長度為 1 或 0
    mid = left + (right - left) / 2
    MERGE_SORT(A, left, mid)        // 遞歸排序左半部分
    MERGE_SORT(A, mid + 1, right)   // 遞歸排序右半部分
    MERGE(A, left, mid, right)      // 合併已解決問題的子陣列

```



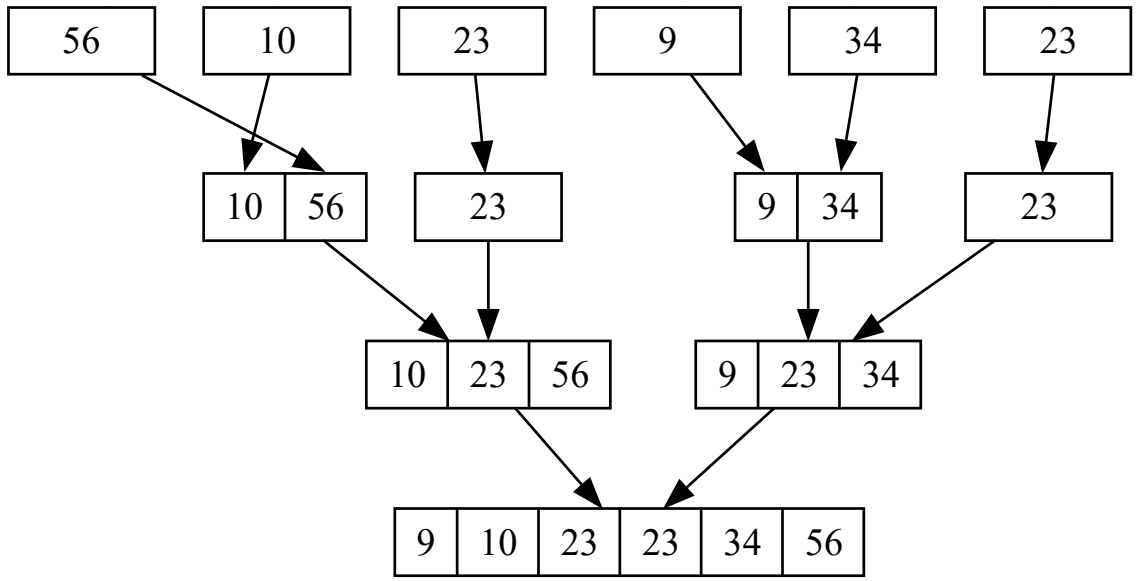
現在我們討論一下合併排序法的演算法細節

conquer

```
MERGE(A, left, mid, right):
    // conquer與combine
    temp = ARRAY OF SIZE (right - left + 1)
    i = left            // 左子陣列的起始索引
    j = mid + 1         // 右子陣列的起始索引
    k = 0               // 排序陣列的索引

    // 比較左、右子陣列的元素，將較小者加入臨時陣列
    WHILE i <= mid AND j <= right:
        IF A[i] <= A[j]:
            temp[k] = A[i]
            i = i + 1
        ELSE:
            temp[k] = A[j]
            j = j + 1
        k = k + 1

    // 將左子陣列剩餘的元素加入排序陣列
    WHILE i <= mid:
        temp[k] = A[i]
        i = i + 1
        k = k + 1

    // 將右子數列剩餘的元素加入排序陣列
    WHILE j <= right:
        temp[k] = A[j]
        j = j + 1
        k = k + 1

    // 將合併後的元素覆蓋回原陣列 A
    FOR p FROM 0 TO (k - 1):
        A[left + p] = temp[p]
```

divide, combine

```
MERGE_SORT(A, left, right):
    IF left >= right:
        RETURN  // 基本情況：子陣列長度為 1 或 0
    mid = left + (right - left) / 2
    MERGE_SORT(A, left, mid)        // 遞歸排序左半部分
    MERGE_SORT(A, mid + 1, right)   // 遞歸排序右半部分
    MERGE(A, left, mid, right)      // 合併已解決問題的子陣列

```



現在我們知道遞迴搭上分治，除了易於表達外也有不錯的效率

在競程中除了能夠思考遞迴方法解決問題，也可以試試往分治法的方向思考

現在我們知道遞迴搭上分治，除了易於表達外也有不錯的效率

在競程中除了能夠思考遞迴方法解決問題，也可以試試往分治法的方向思考

最後，分治法最重要的部份、
區分貪心法和動態規劃的核心部分：

現在我們知道遞迴搭上分治，除了易於表達外也有不錯的效率

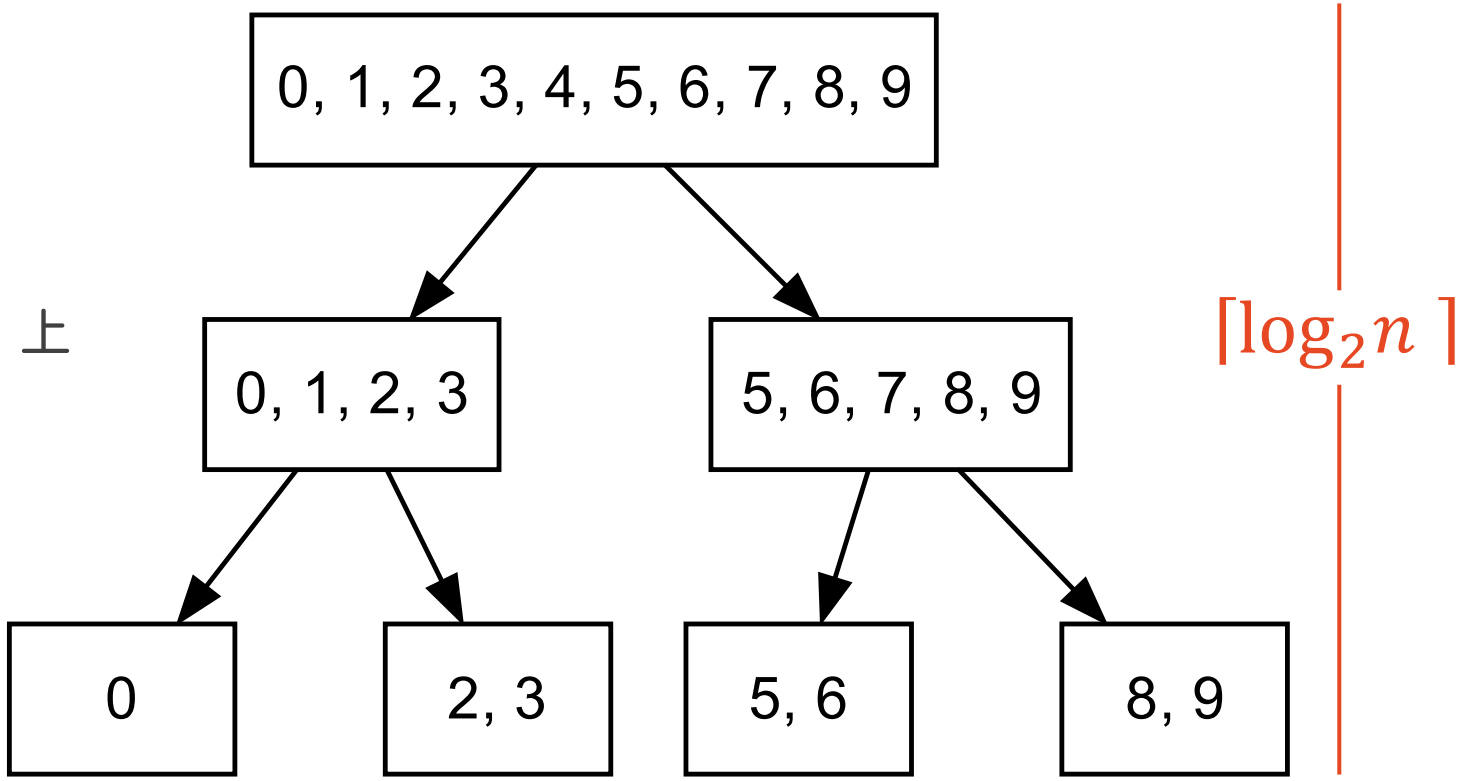
在競程中除了能夠思考遞迴方法解決問題，也可以試試往分治法的方向思考

最後，分治法最重要的部份、
區分貪心法和動態規劃的核心部分：

「關鍵是子問題之間相互獨立，彼此結構相同但又不重疊」

「持續平分陣列，直到指針碰到搜尋目標」

1. 給定一個陣列從 0 到 9
2. 找出每個節點的指針
3. 如果中間元素是目標就結束
4. 否則判斷大小，移動指針到 Mid 上



複雜度：$O(\log_2 n)$

> Divide 將整個大問題分割，但必須確保問題的題型、架構是一樣的，
> 完成後再將每個小部分都以遞迴的方式處理，即為 conquer，
> 最後把每個小部分都結合起來，即是 combine 也是最終答案。

問題：

1. 甚麼是 in place 特性？
2. bubble sort 屬於分治法嗎？
3. 該怎麼區分「分治」、「貪婪法」跟「動態規劃」呢？

謝謝聆聽

LeetCode 912. Sort an Array