

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

Slides by **Sean Szumlanski**  
for **CS106B**, Programming Abstractions

*Spring 2025*

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

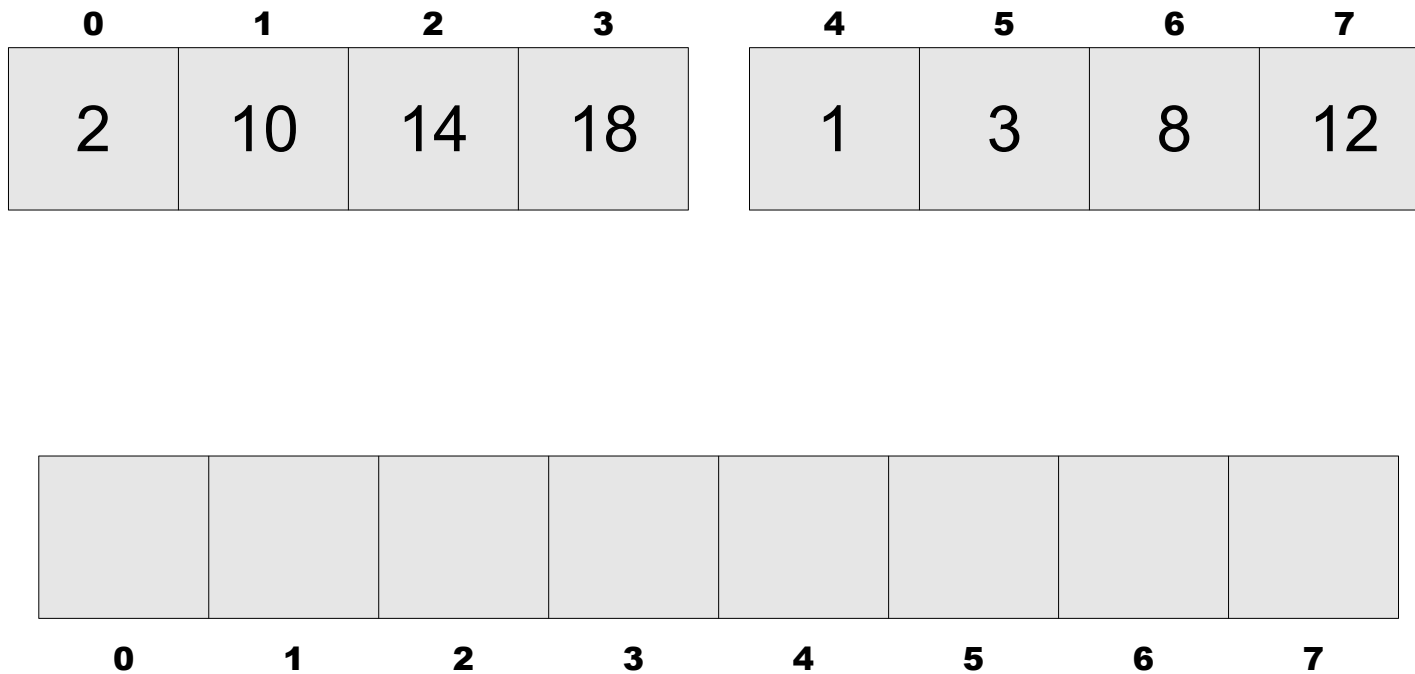
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?

0	1	2	3	4	5	6	7
2	10	14	18	1	3	8	12

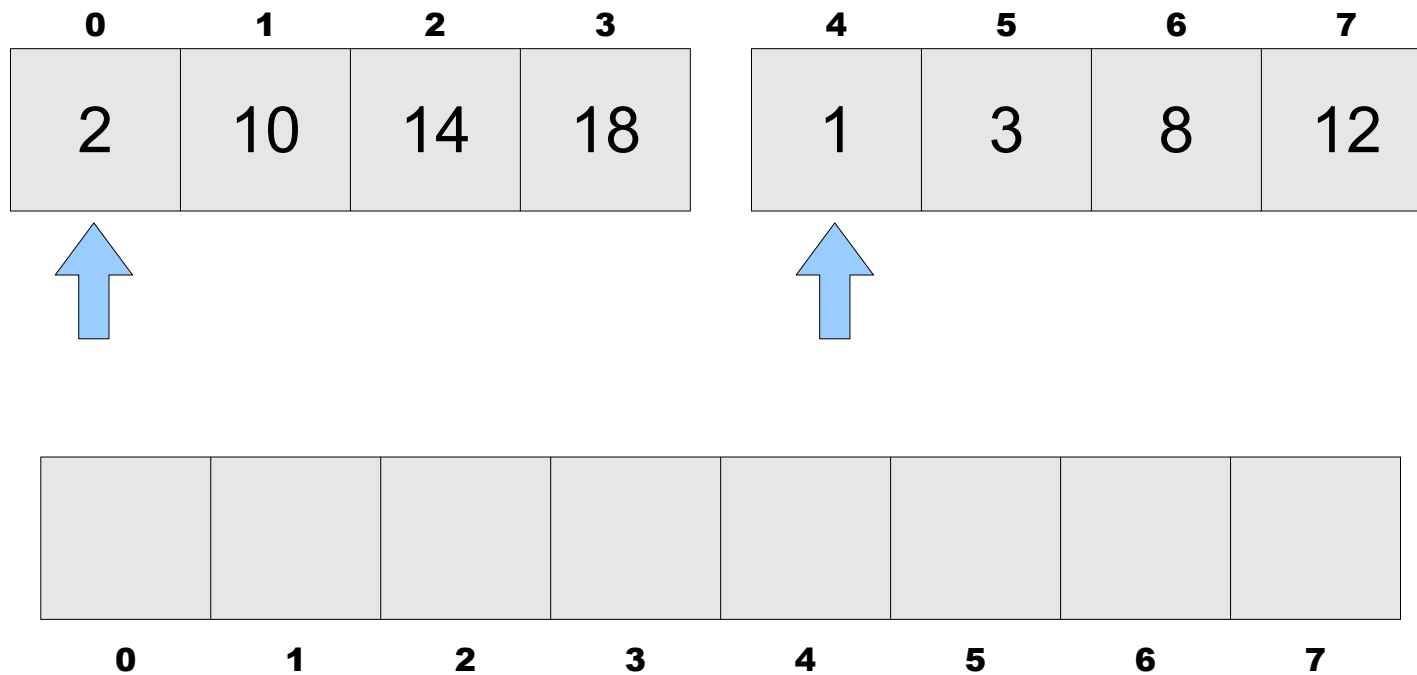
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



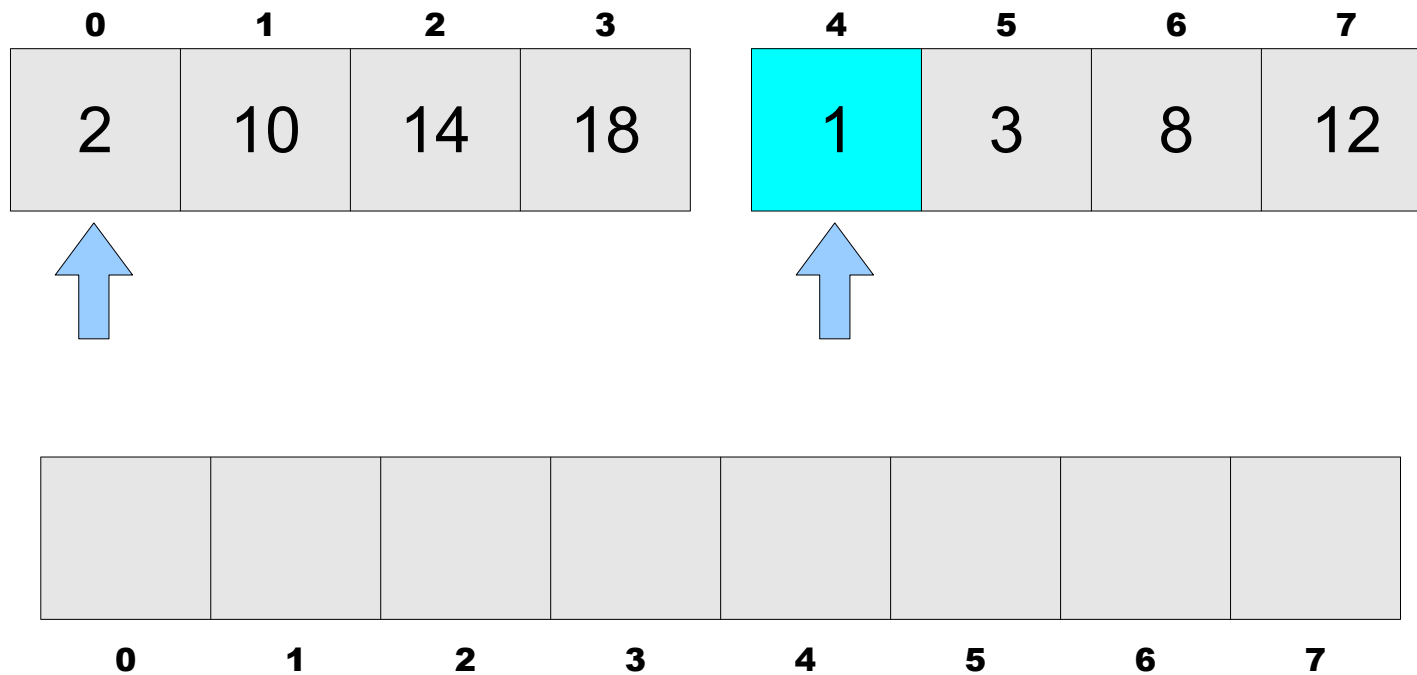
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



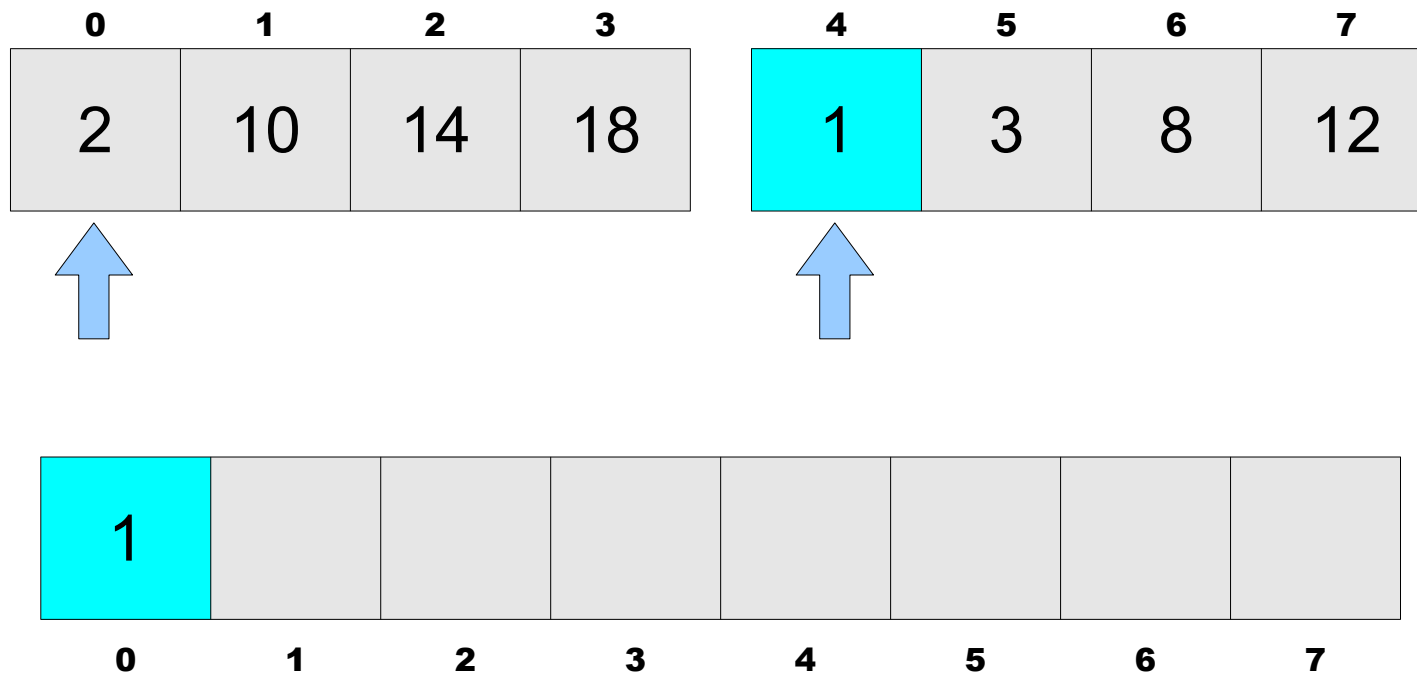
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



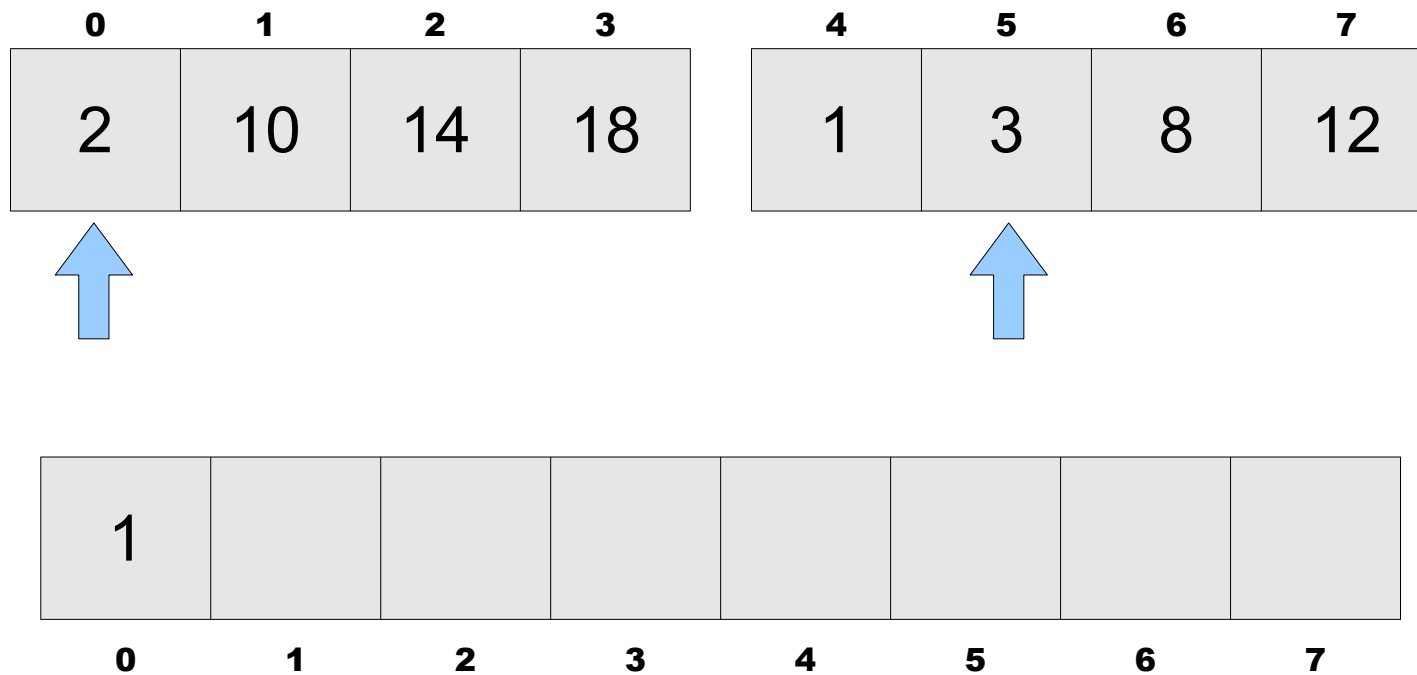
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



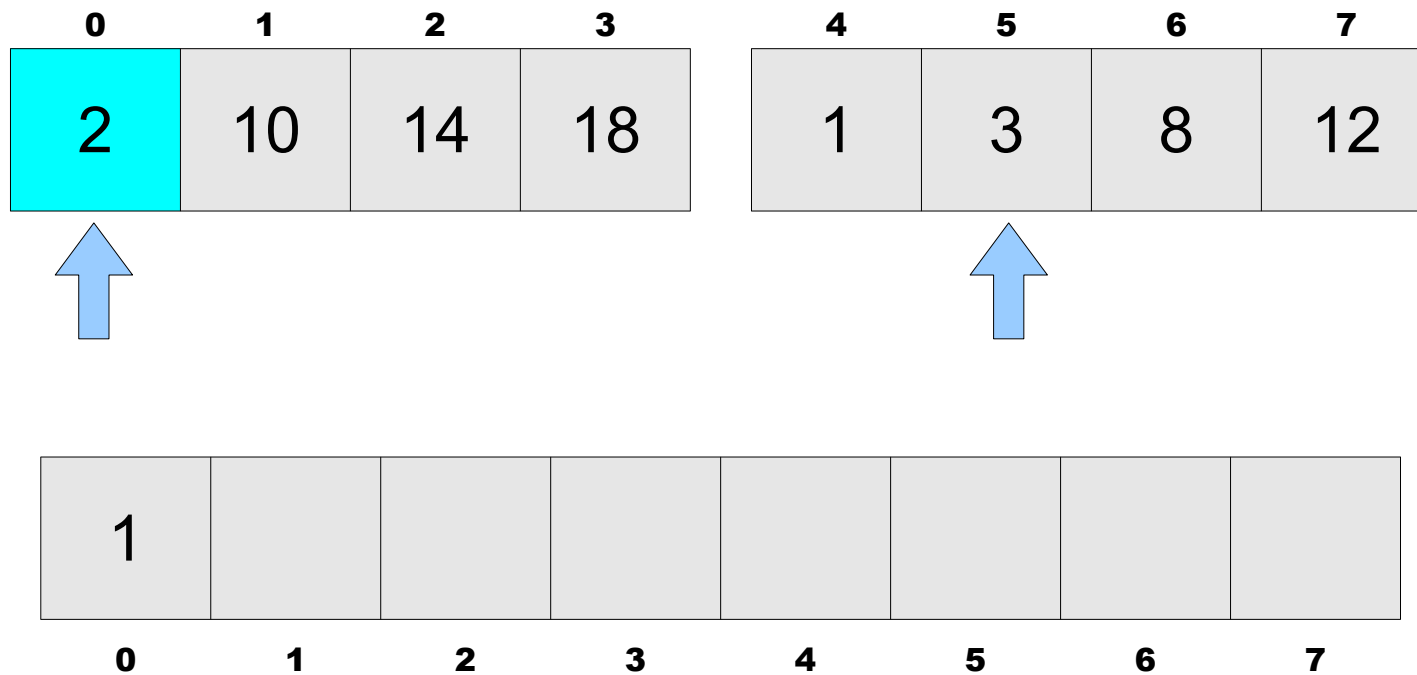
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



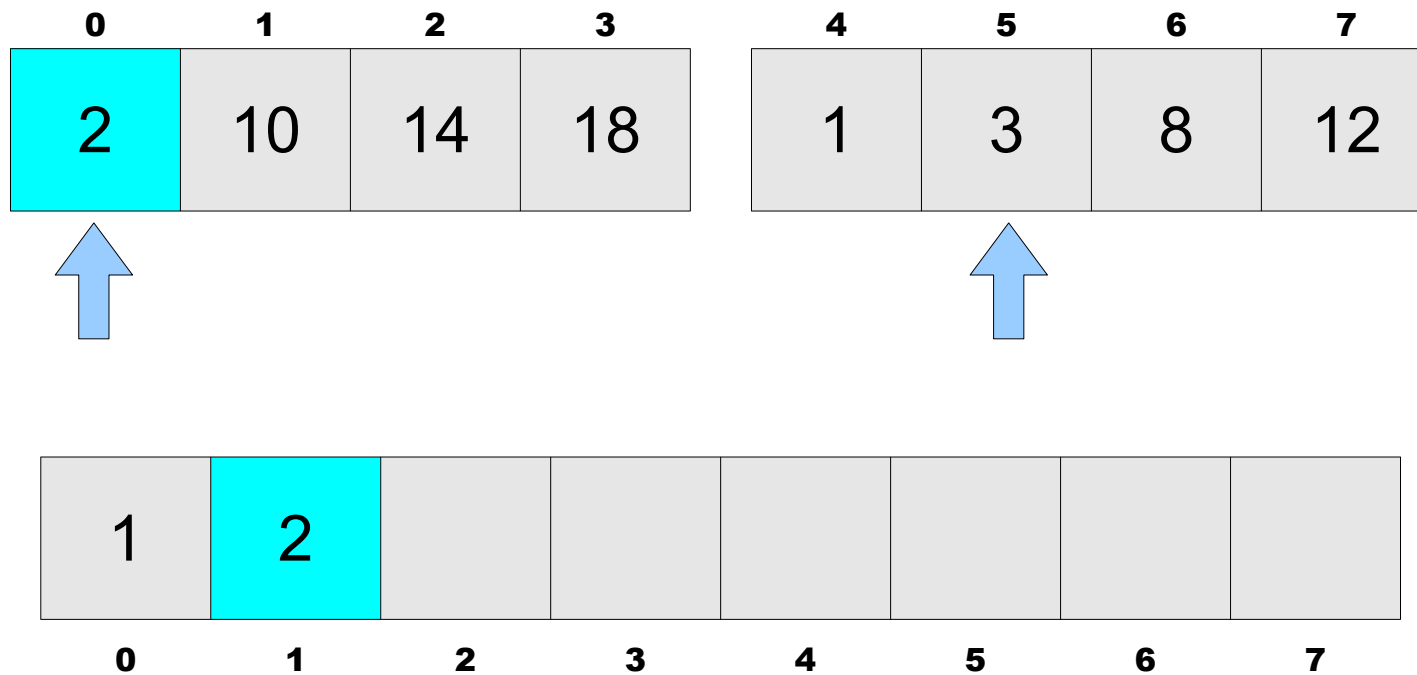
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



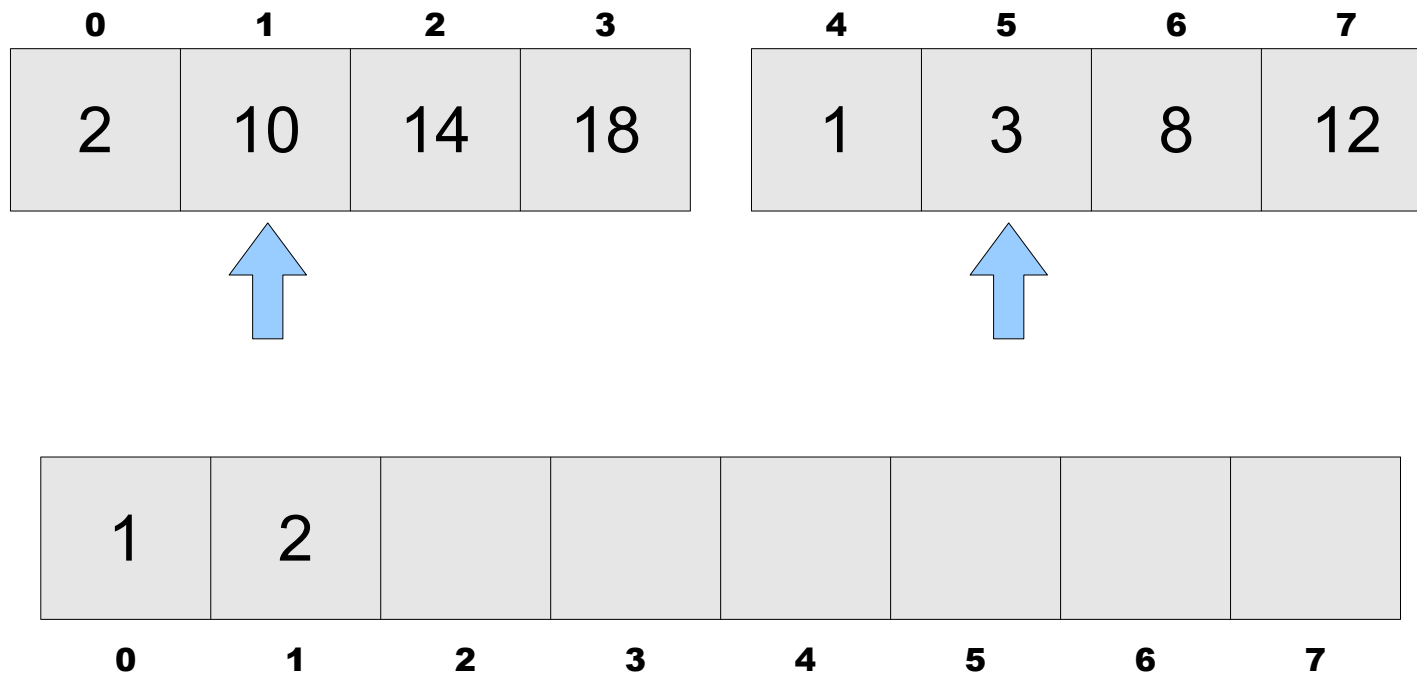
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



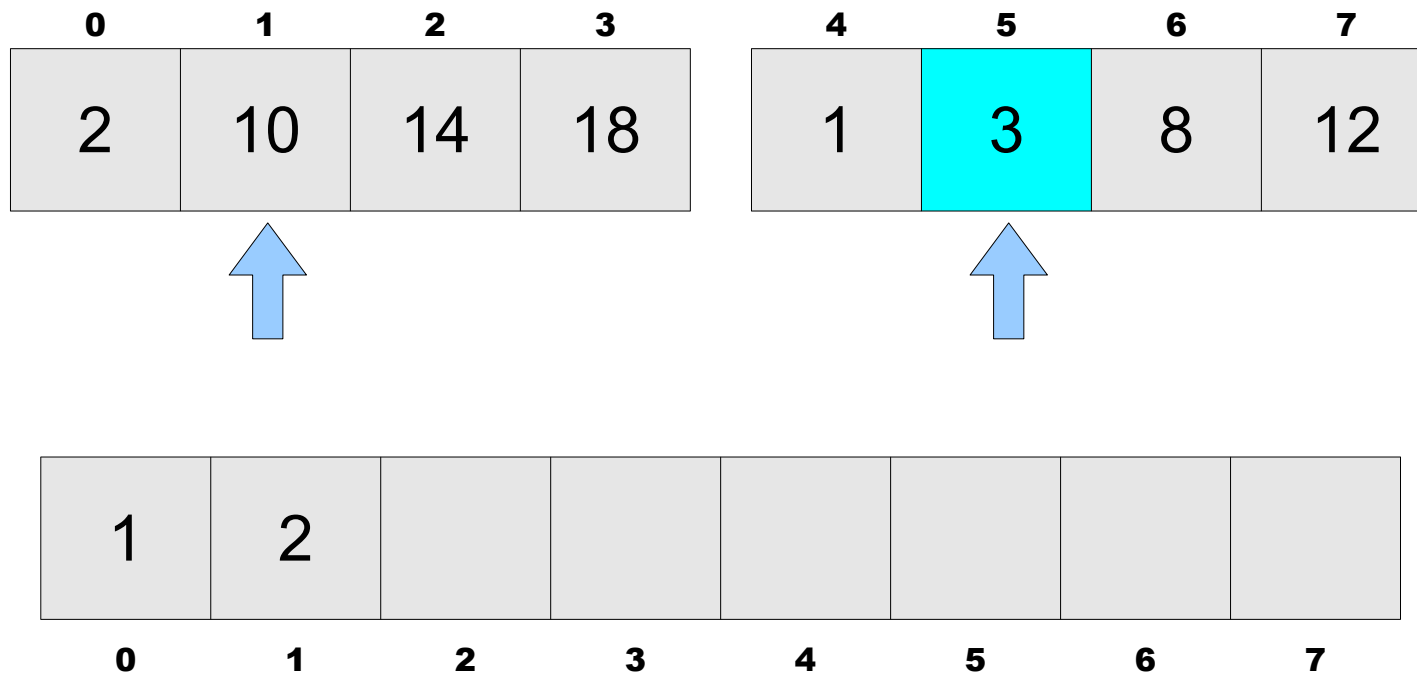
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



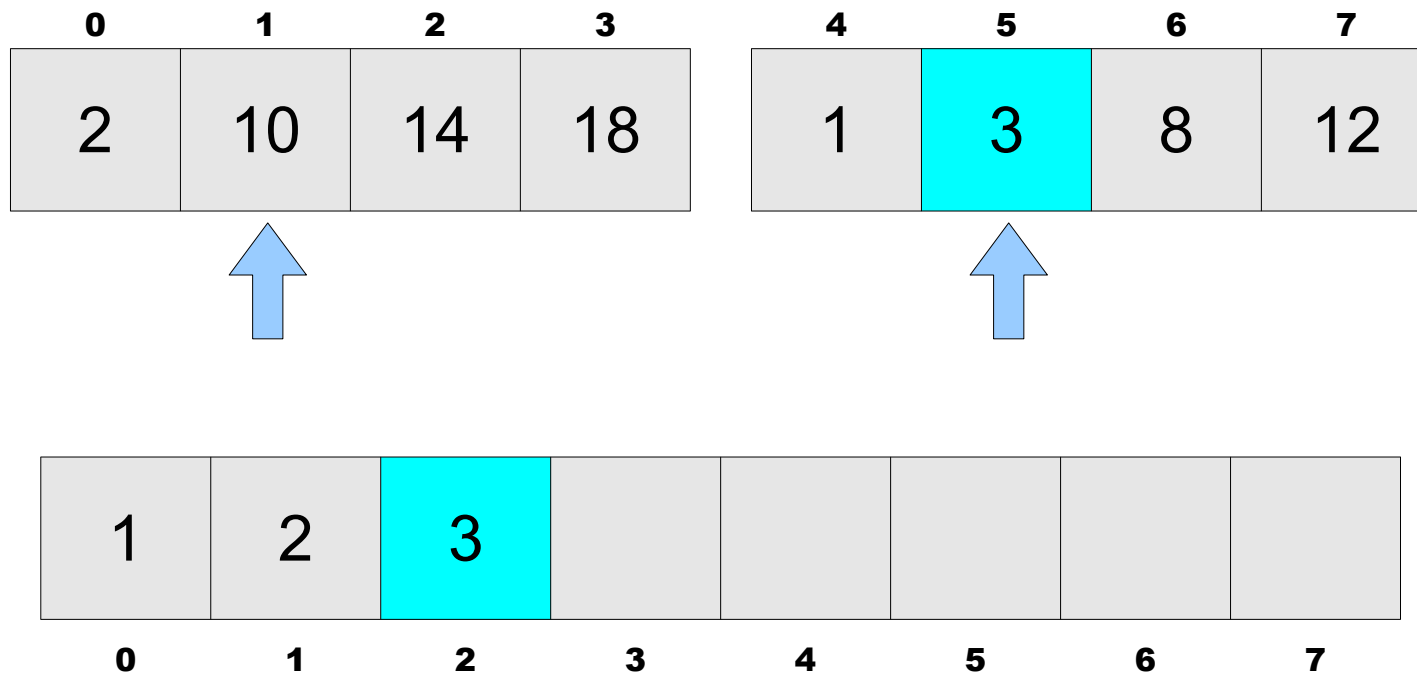
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



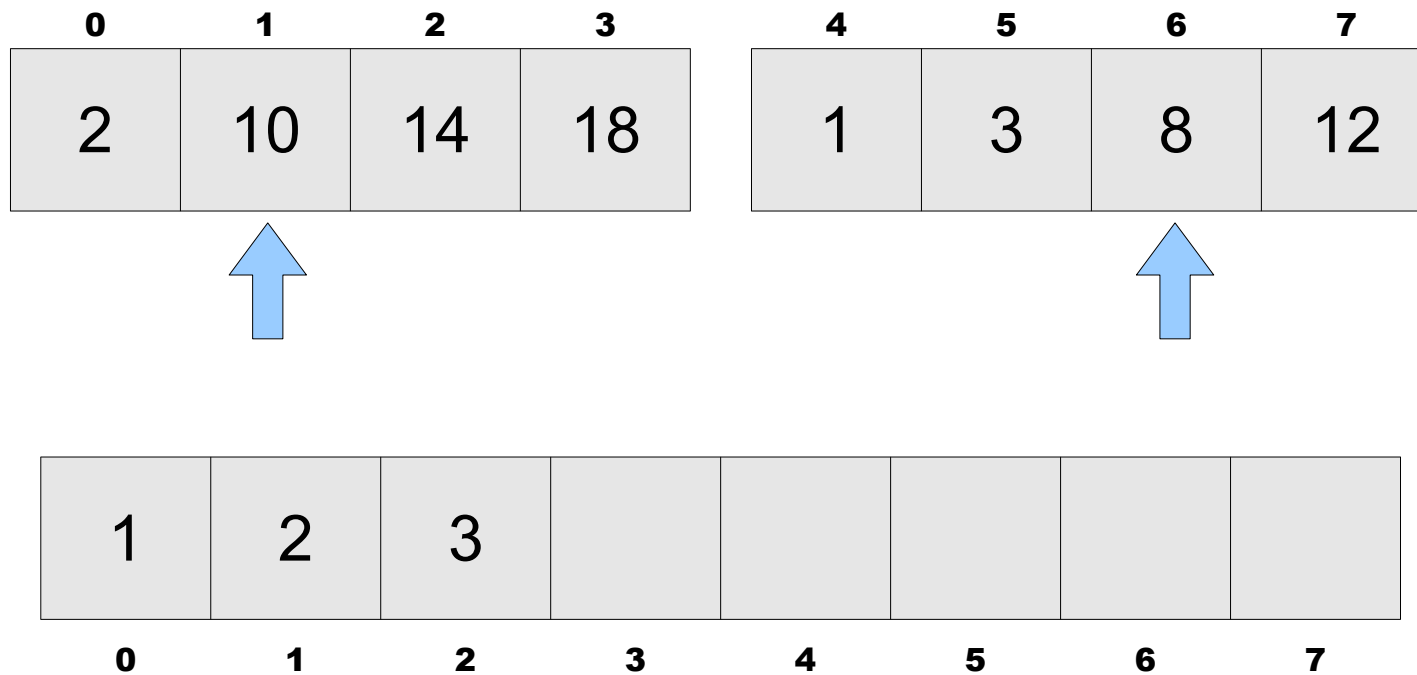
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



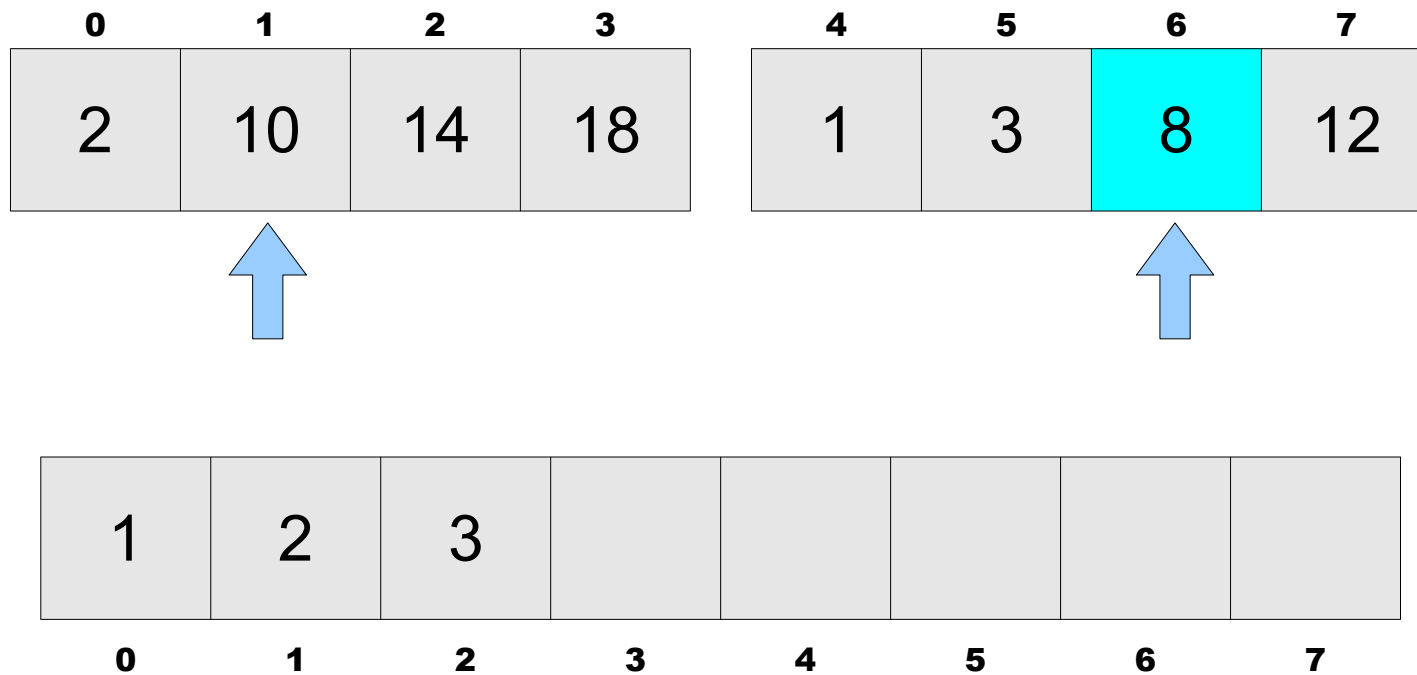
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



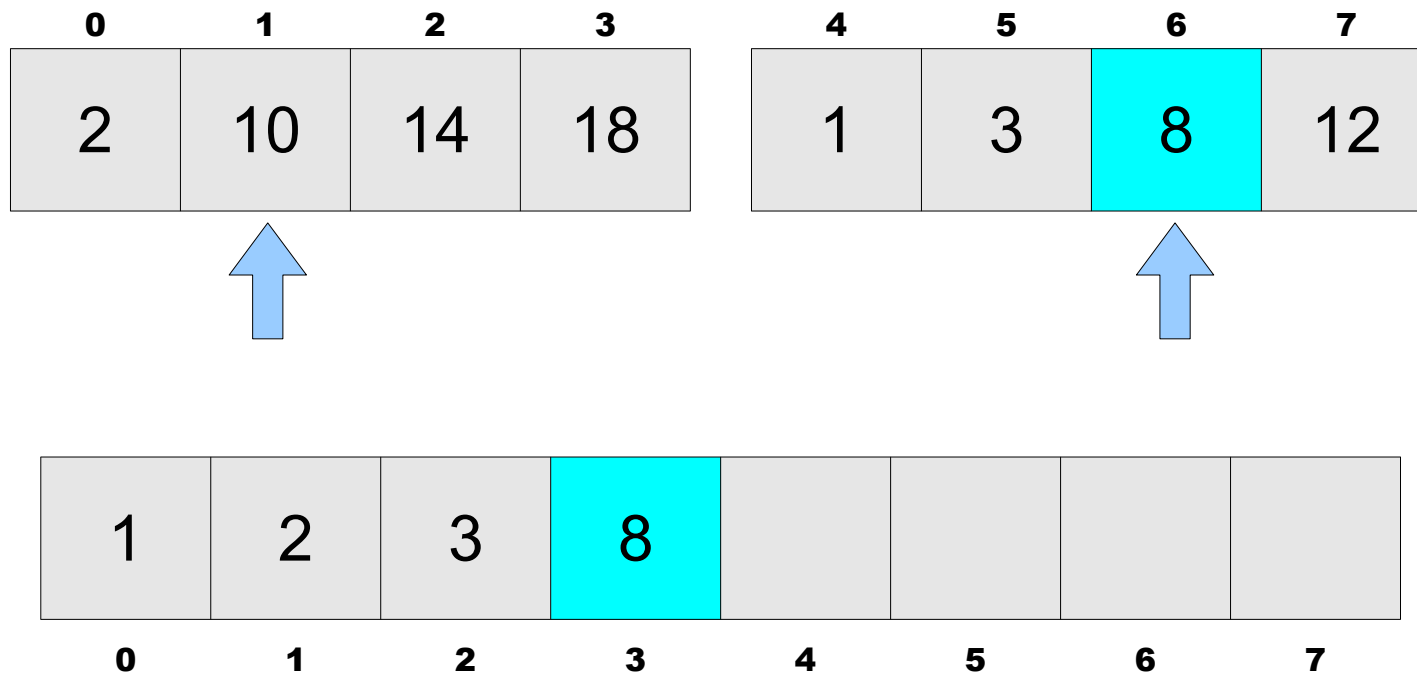
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



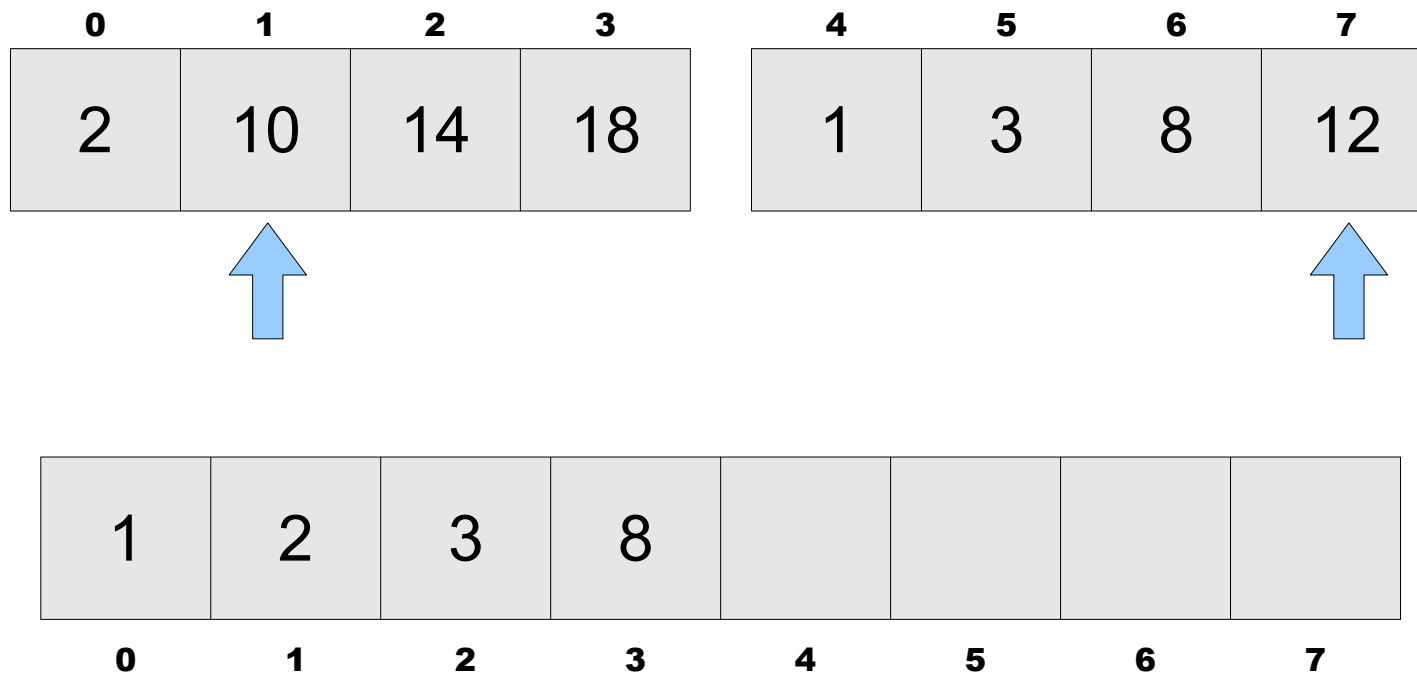
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



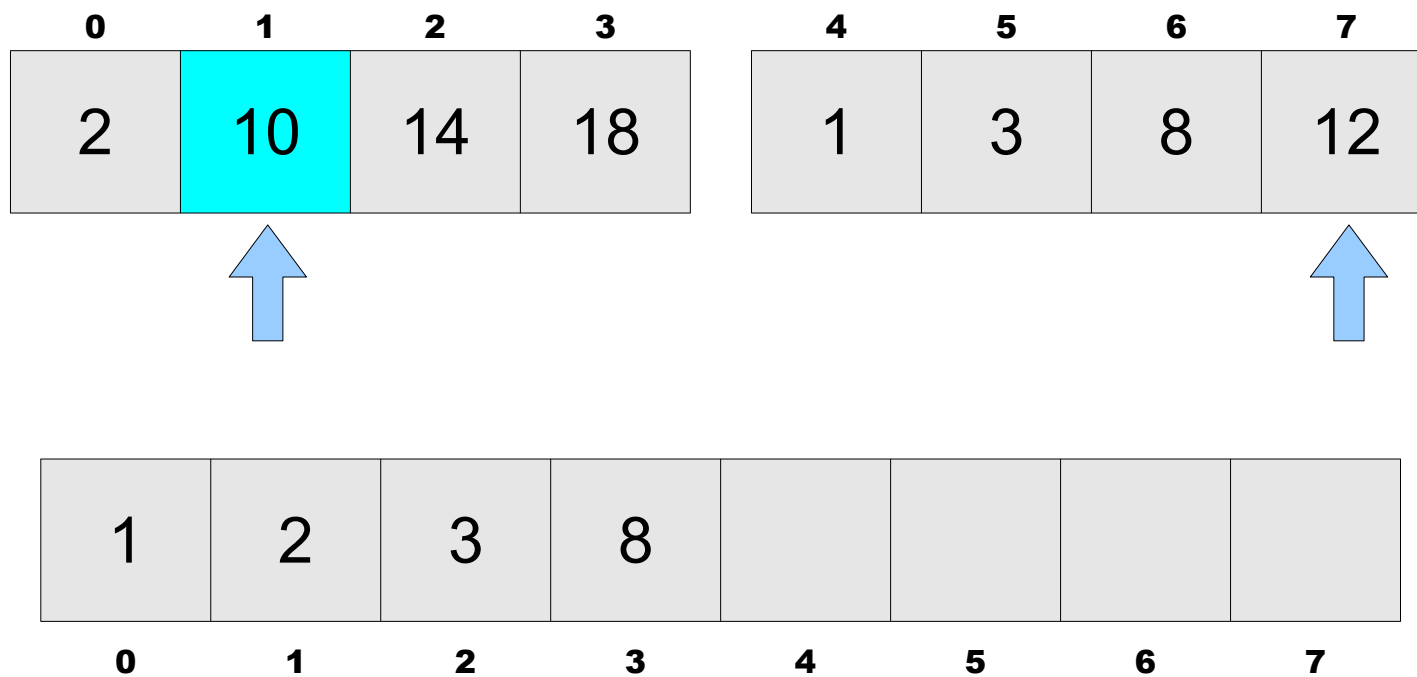
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



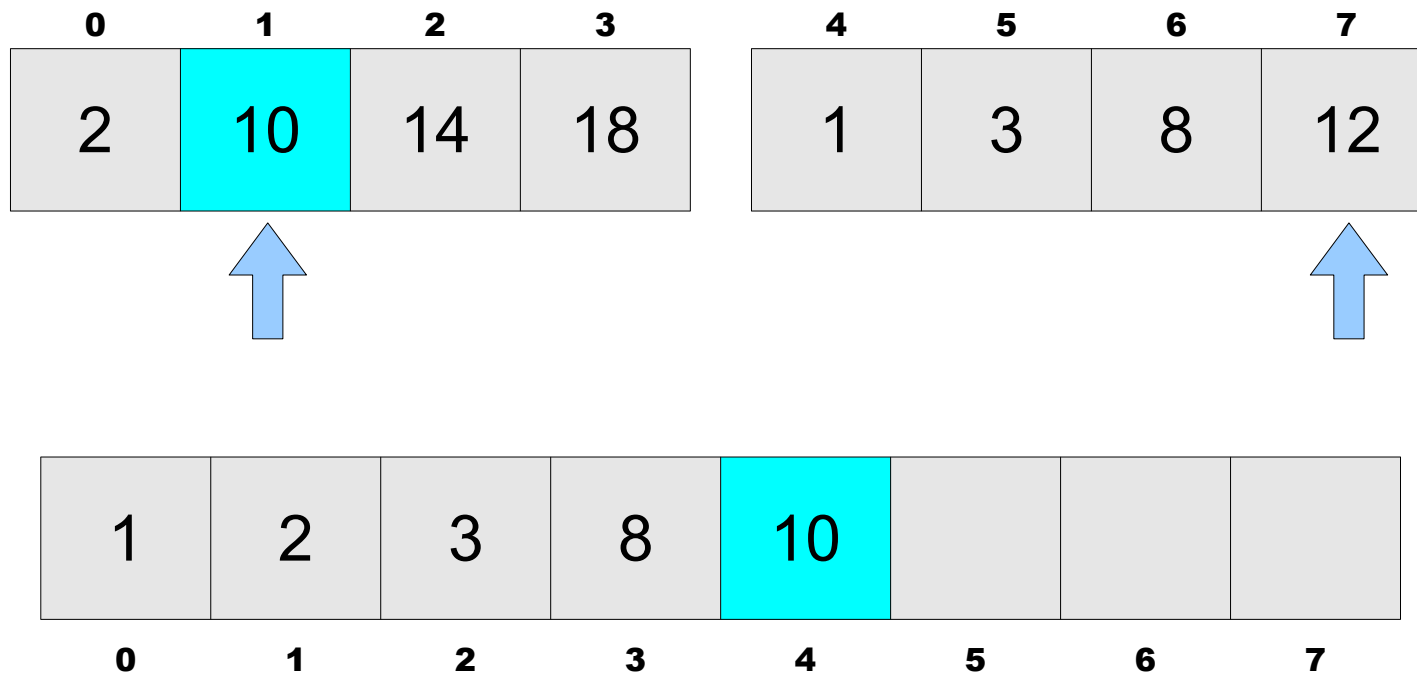
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



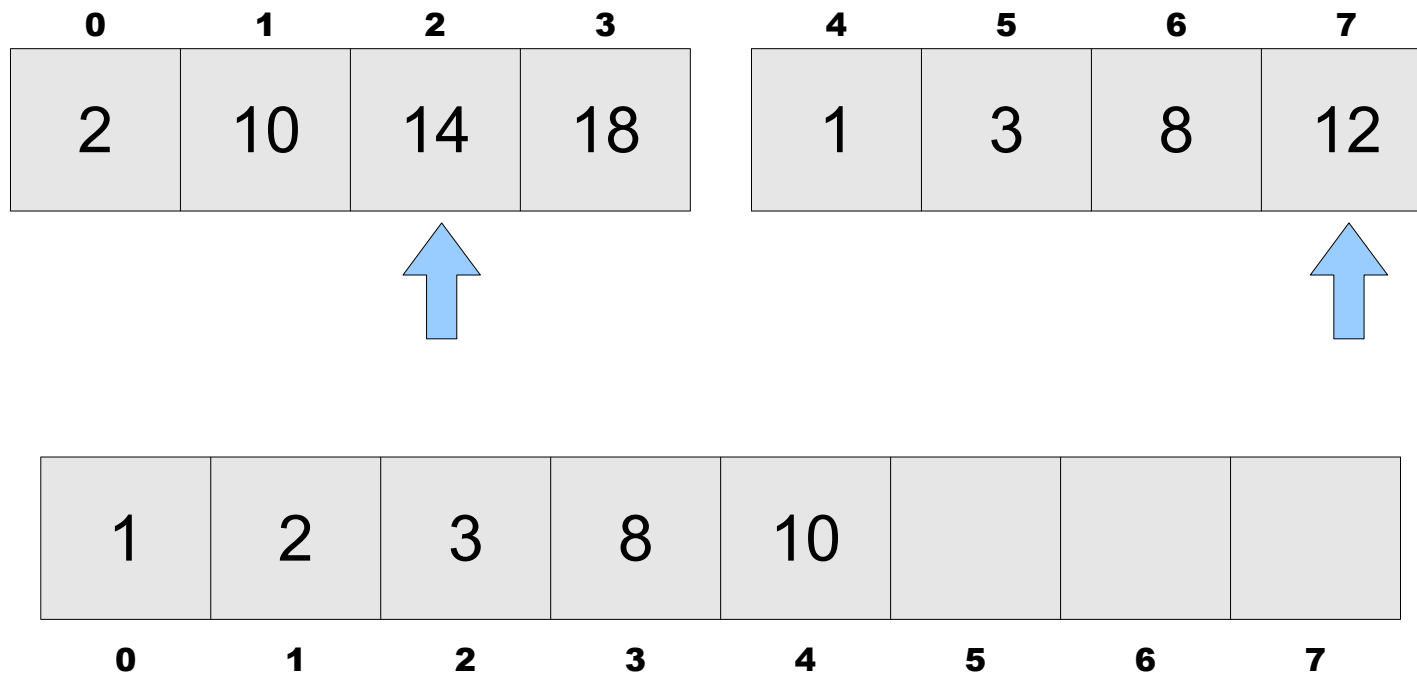
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



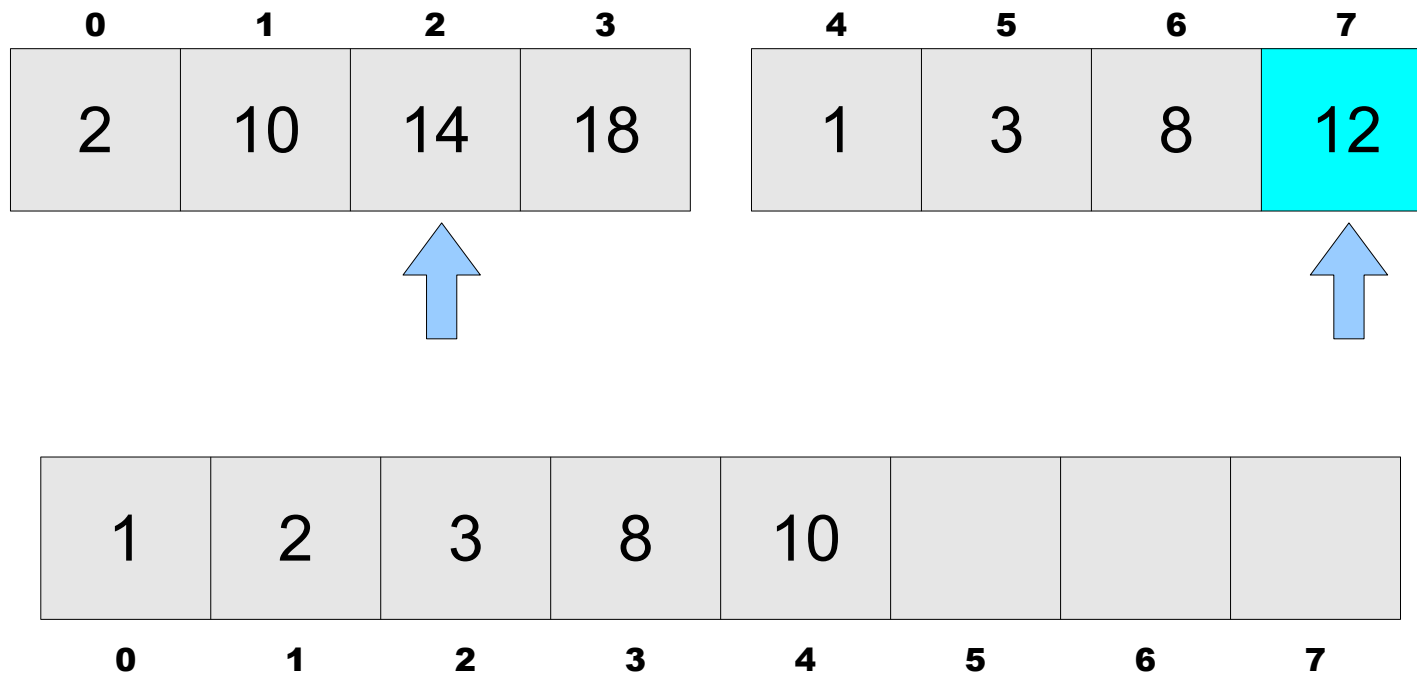
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



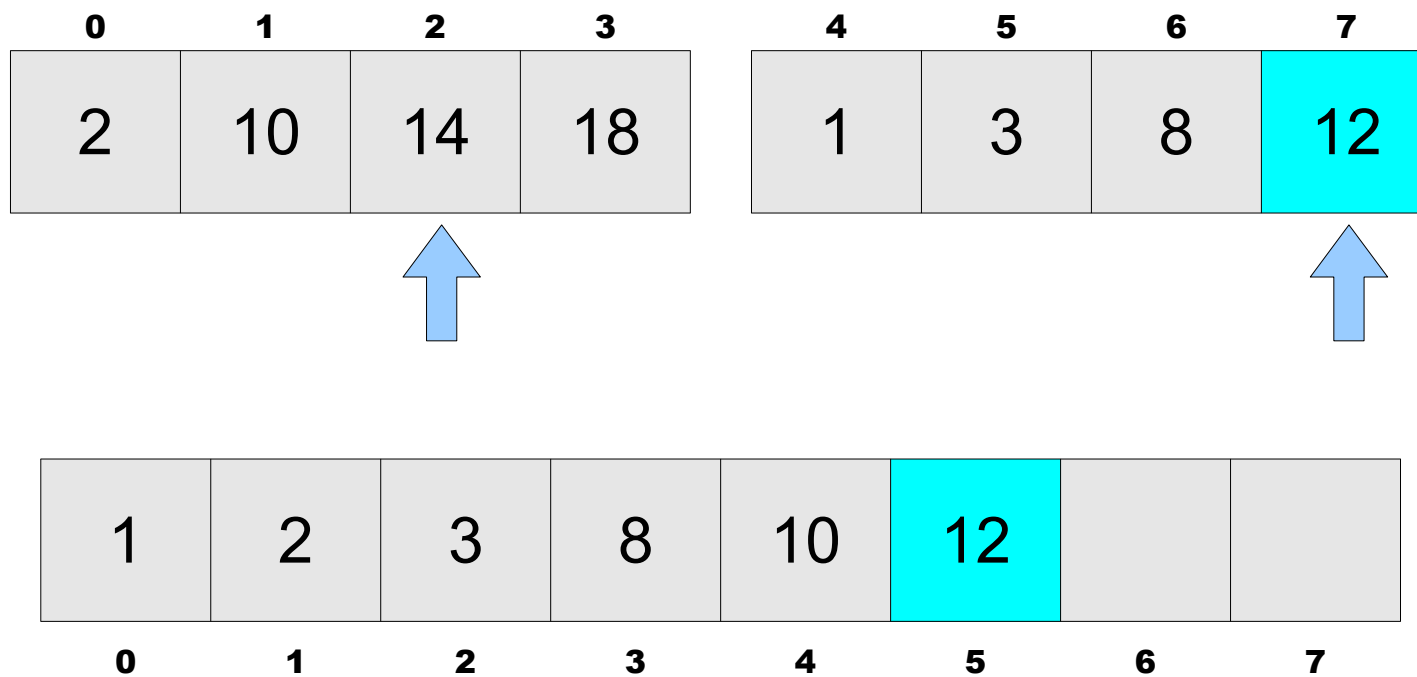
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



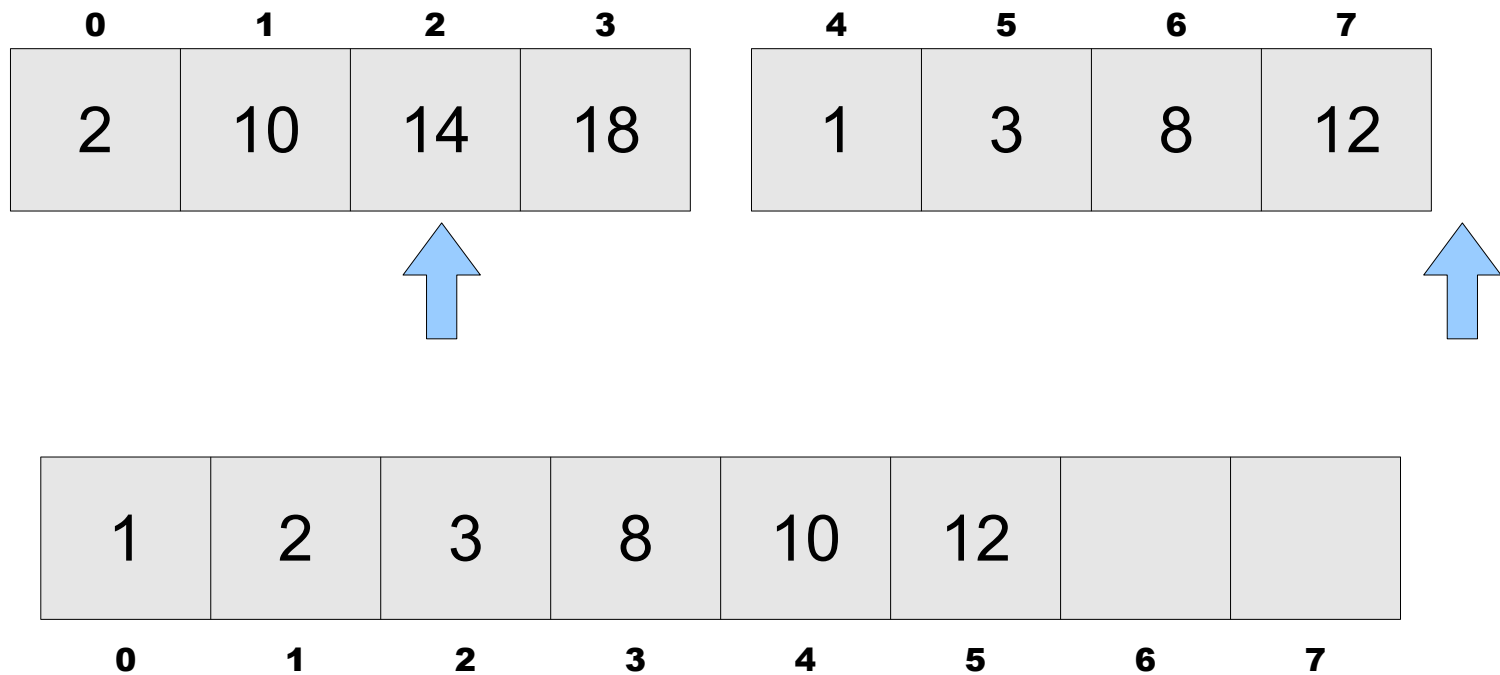
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



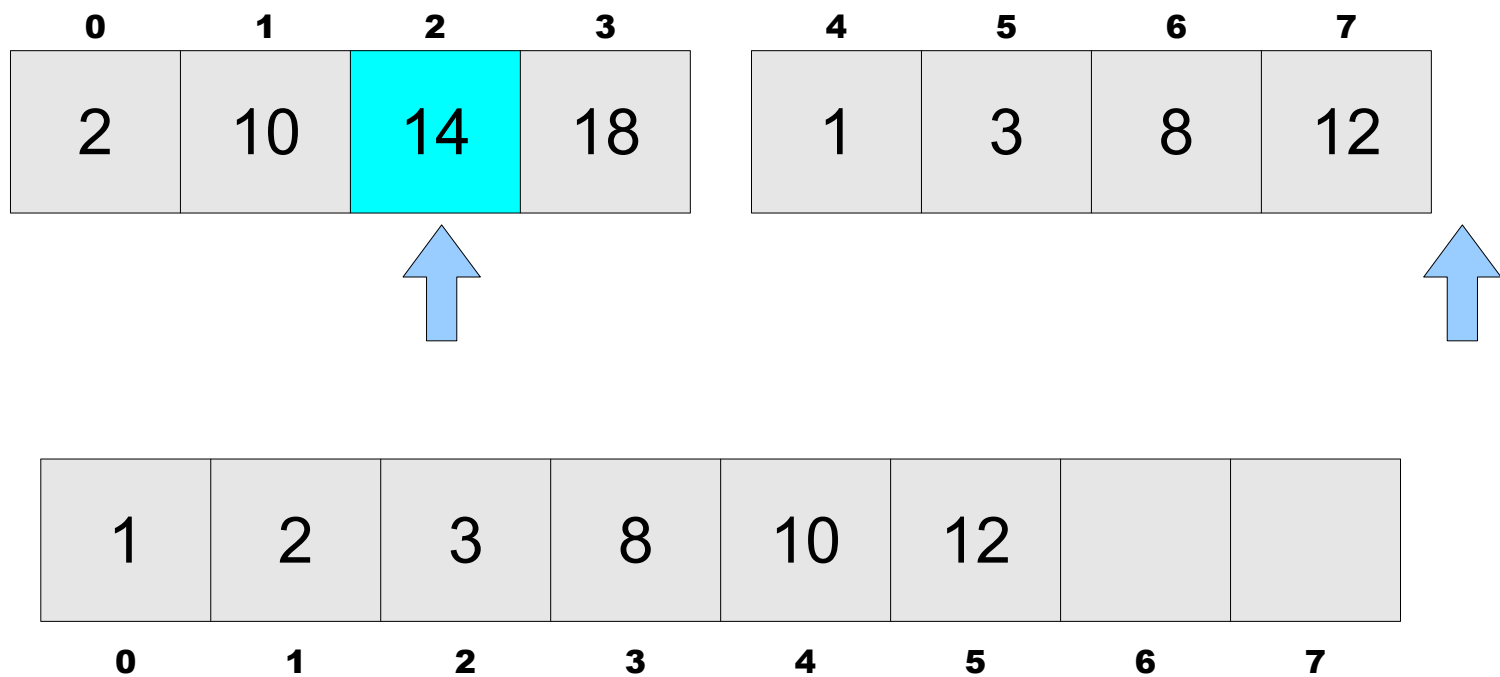
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



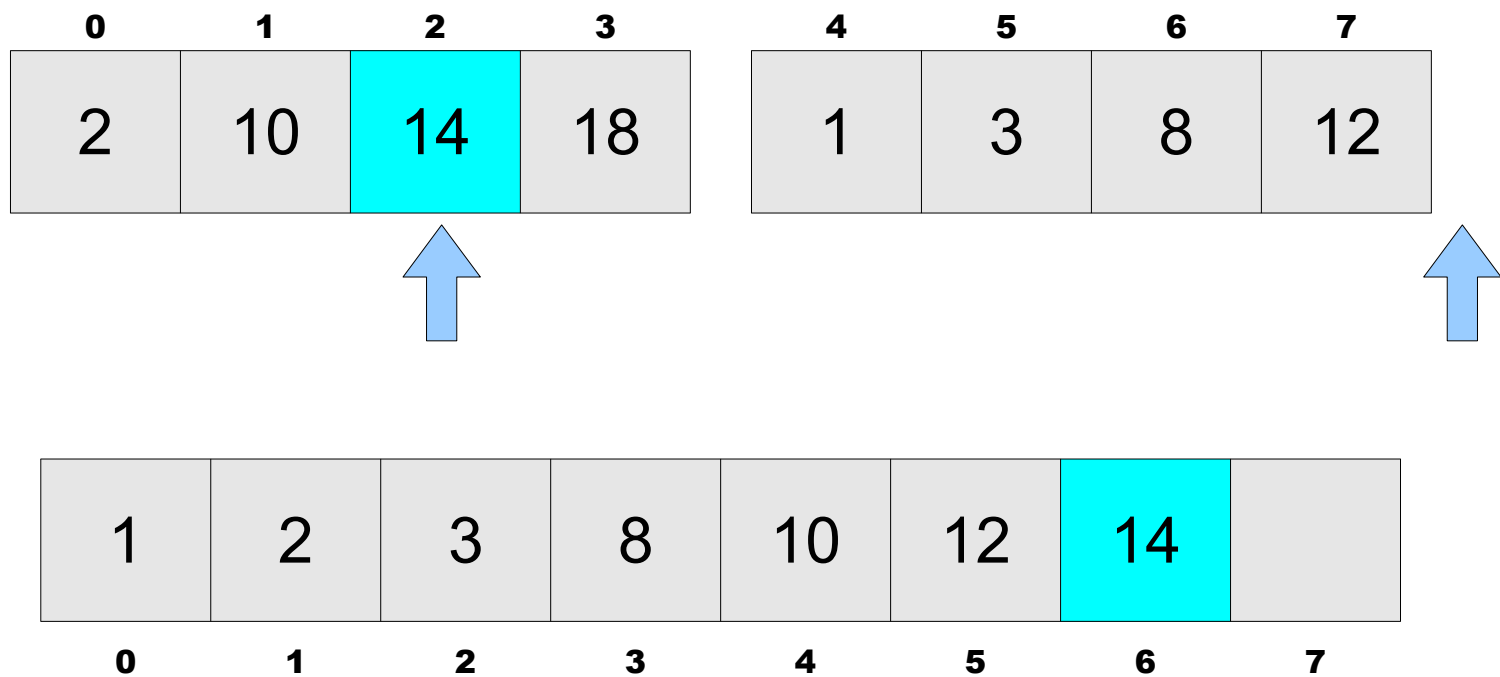
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



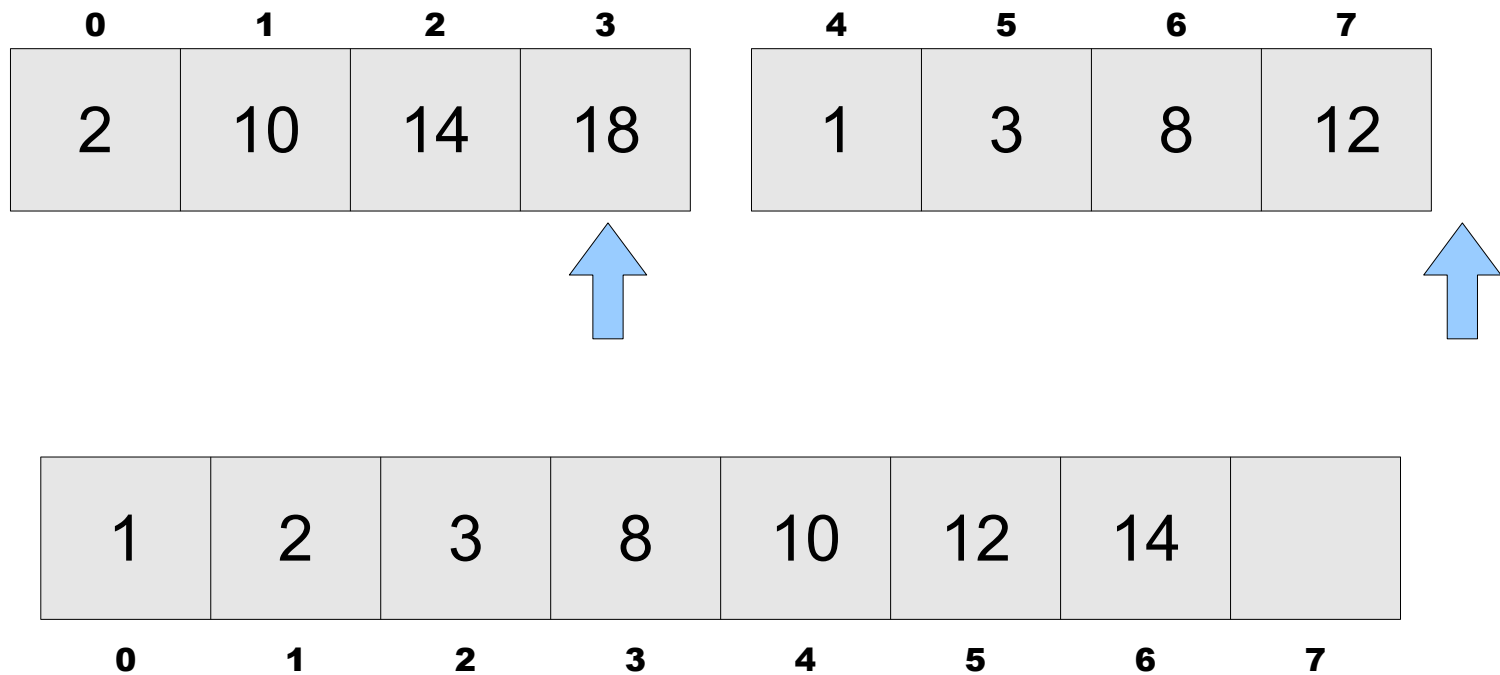
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



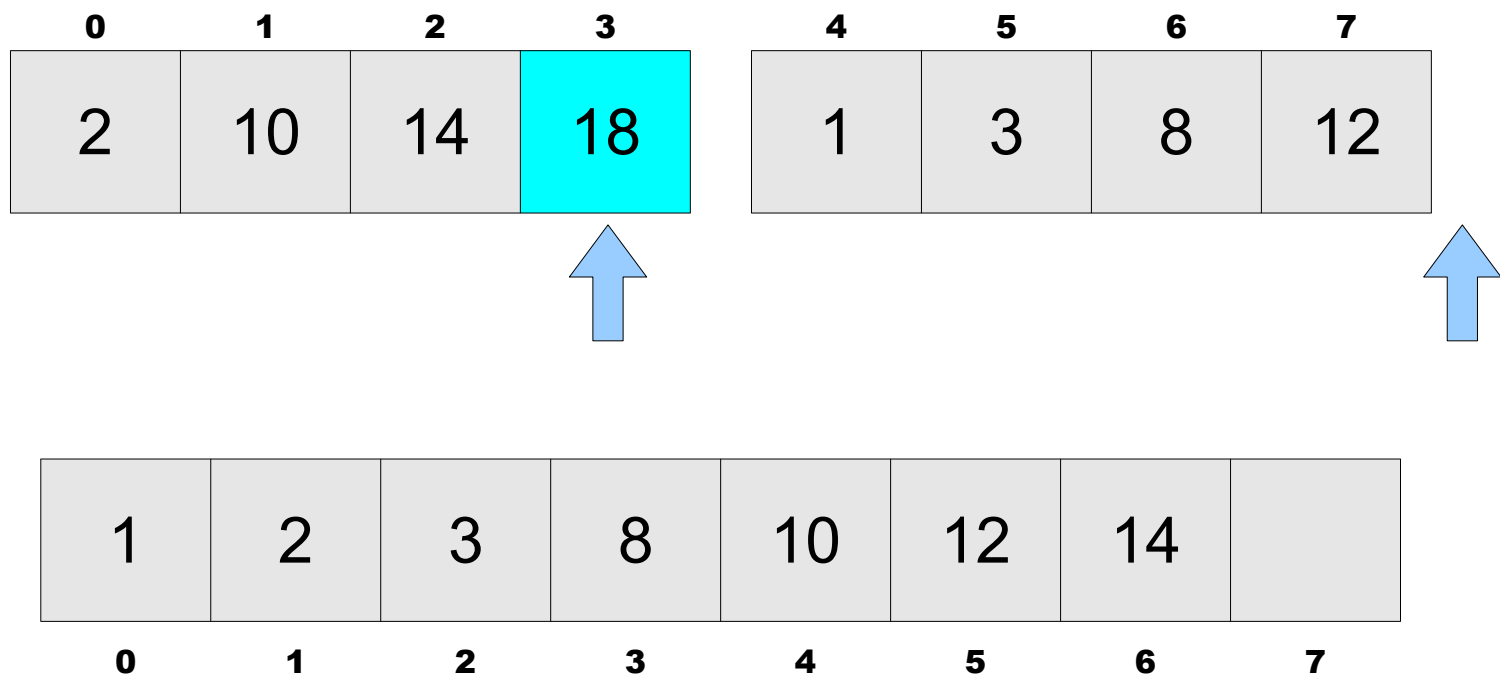
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



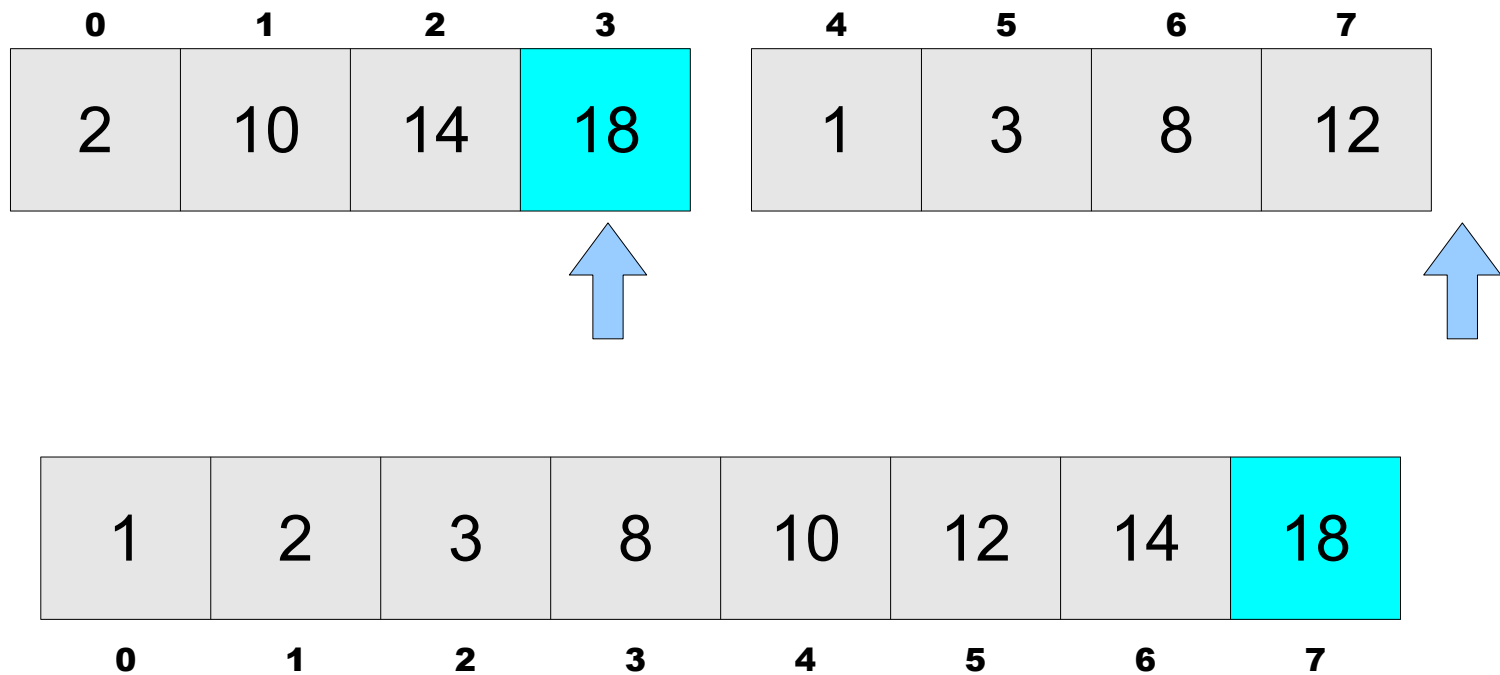
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



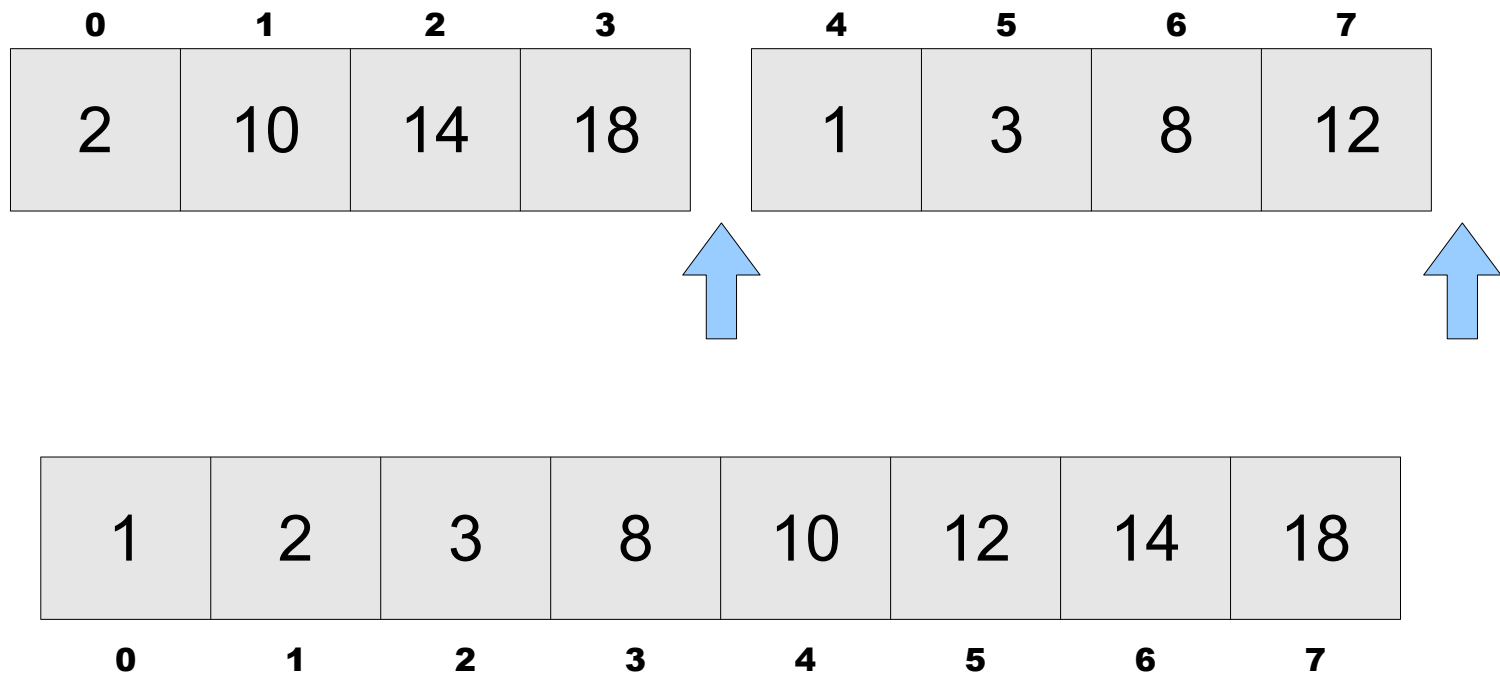
# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?



# But First of All...

Do you agree we can merge two **sorted** vectors into **one** sorted vector in **linear** time?

0	1	2	3	4	5	6	7
2	10	14	18	1	3	8	12

**\*TADA\***

1	2	3	8	10	12	14	18
0	1	2	3	4	5	6	7

# Merge Sort

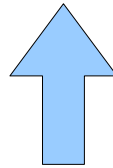
(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

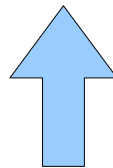


**MID**

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

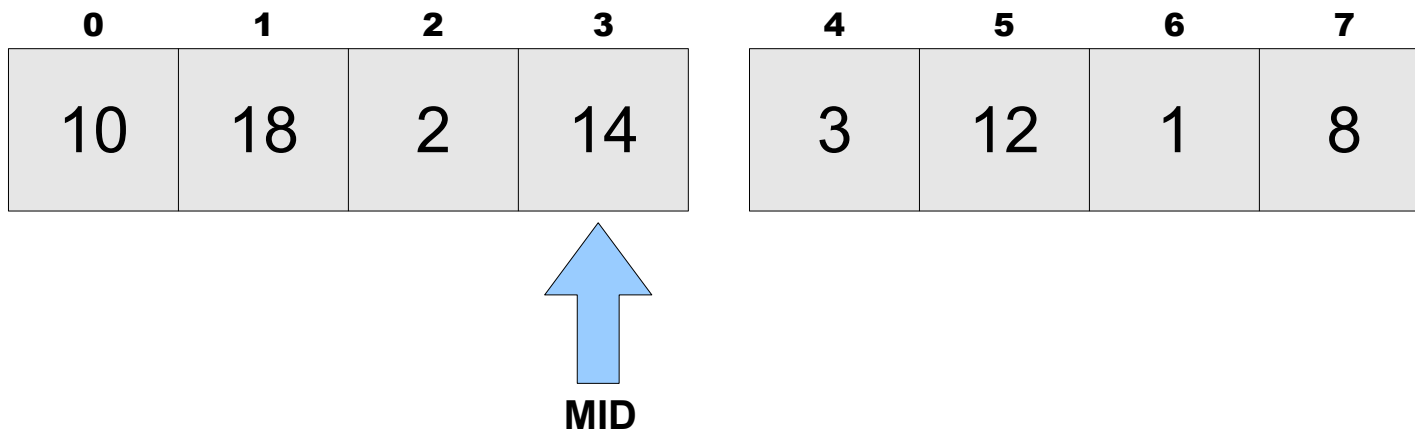


**MID**

**Formula:**  $\text{mid} = \text{lo} + (\text{hi} - \text{lo}) / 2;$

# Merge Sort

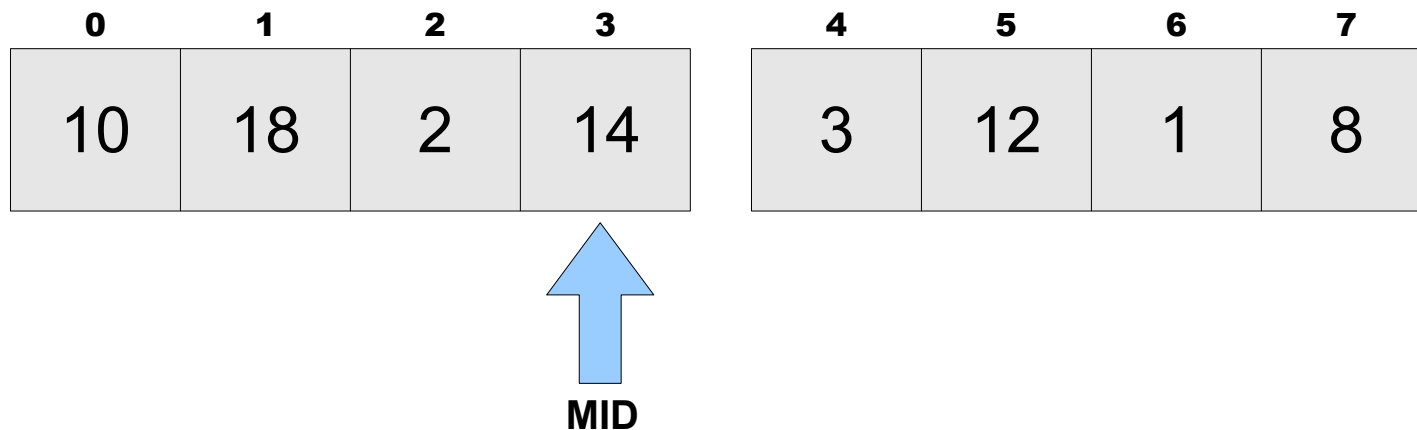
(break down the vector, then merge the pieces back together)



**Formula:**  $\text{mid} = \text{lo} + (\text{hi} - \text{lo}) / 2;$

# Merge Sort

(break down the vector, then merge the pieces back together)

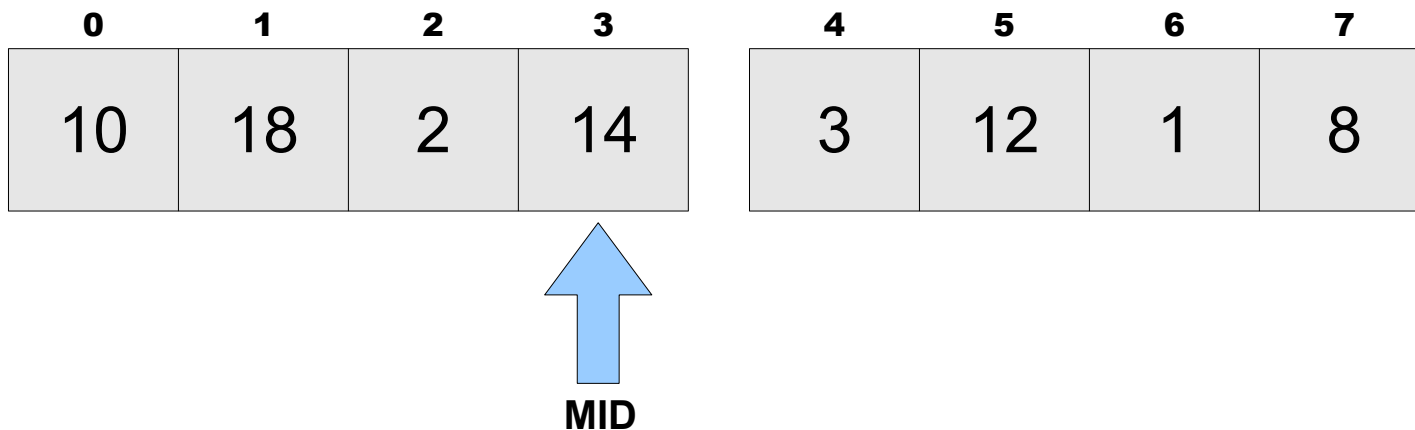


**Formula:**  $\text{mid} = \text{lo} + (\text{hi} - \text{lo}) / 2;$

**mergeSort**(v, lo, hi) →

# Merge Sort

(break down the vector, then merge the pieces back together)

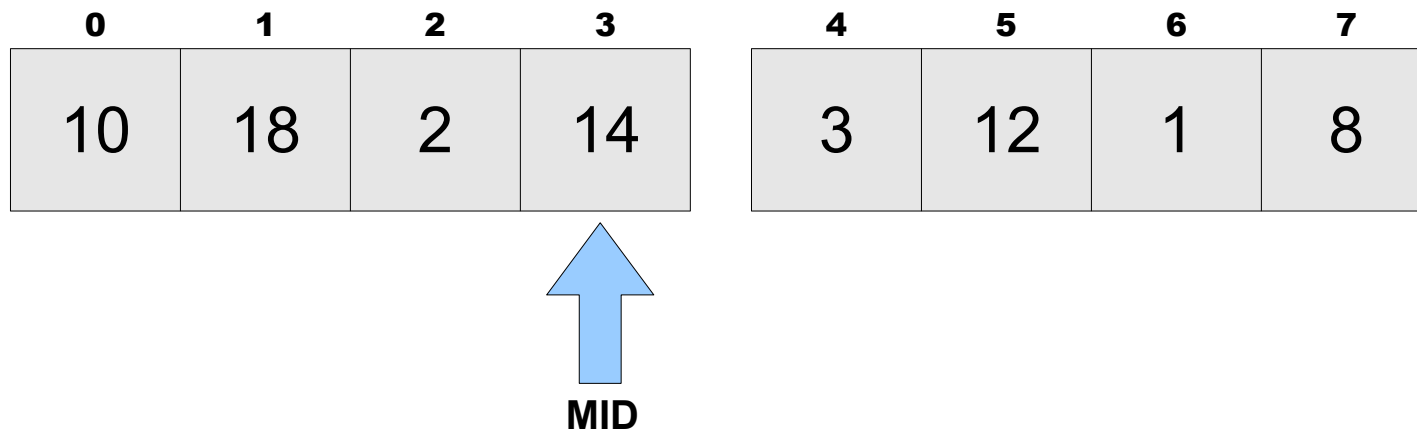


**Formula:**  $\text{mid} = \text{lo} + (\text{hi} - \text{lo}) / 2;$

**mergeSort**(v, lo, hi) → **mergeSort**(v, lo, mid)

# Merge Sort

(break down the vector, then merge the pieces back together)



**Formula:**  $\text{mid} = \text{lo} + (\text{hi} - \text{lo}) / 2;$

**mergeSort(v, lo, hi) →**

- mergeSort(v, lo, mid)**
- mergeSort(v, mid + 1, hi)**

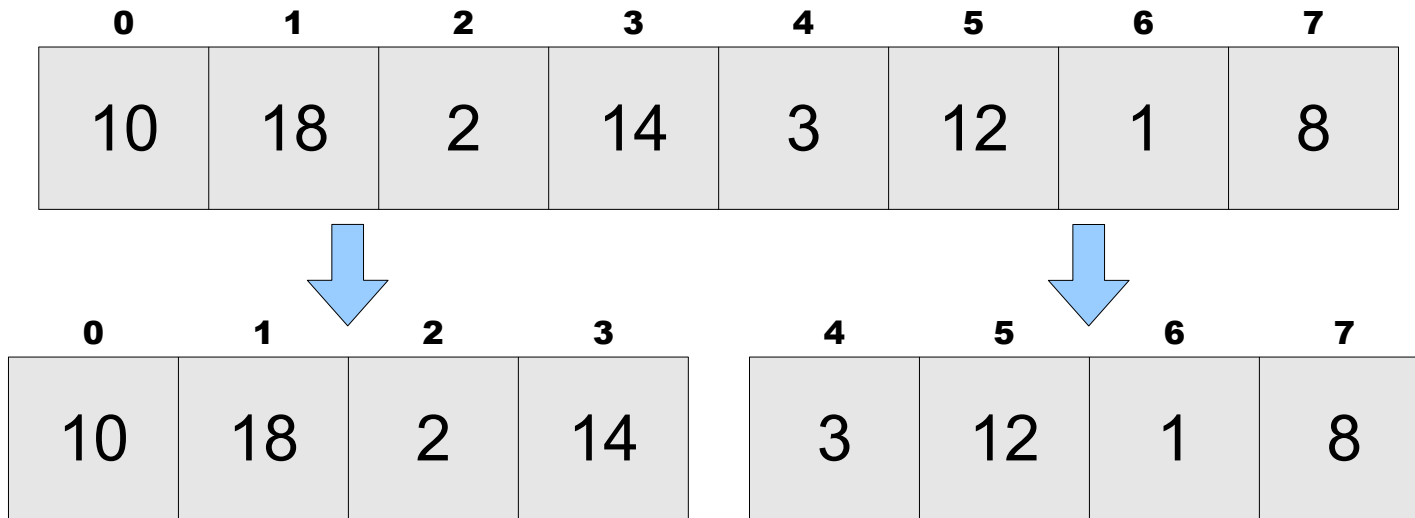
# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

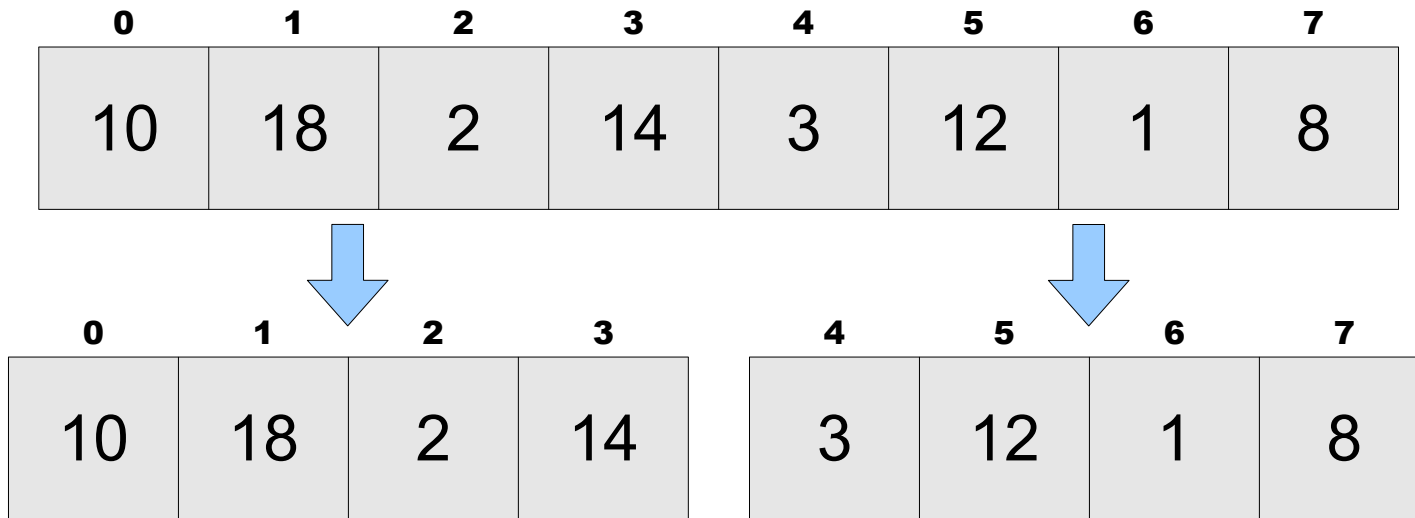
# Merge Sort

(break down the vector, then merge the pieces back together)



# Merge Sort

(break down the vector, then merge the pieces back together)

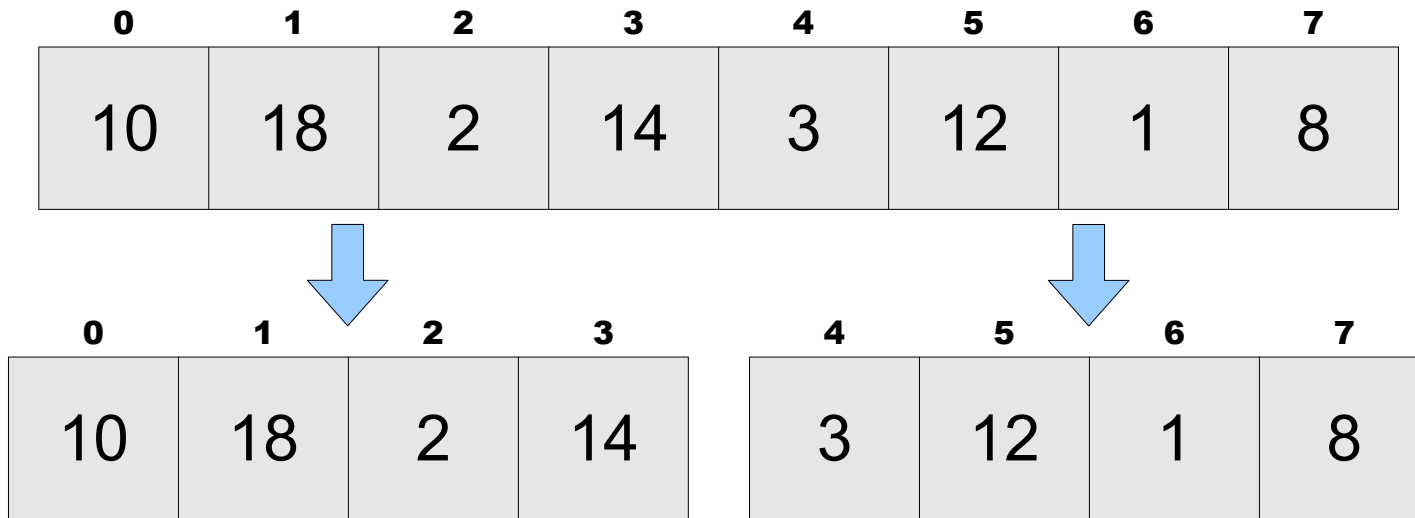


**mergeSort(v, 0, 7) →**

- mergeSort(v, 0, 3)**
- mergeSort(v, 4, 7)**

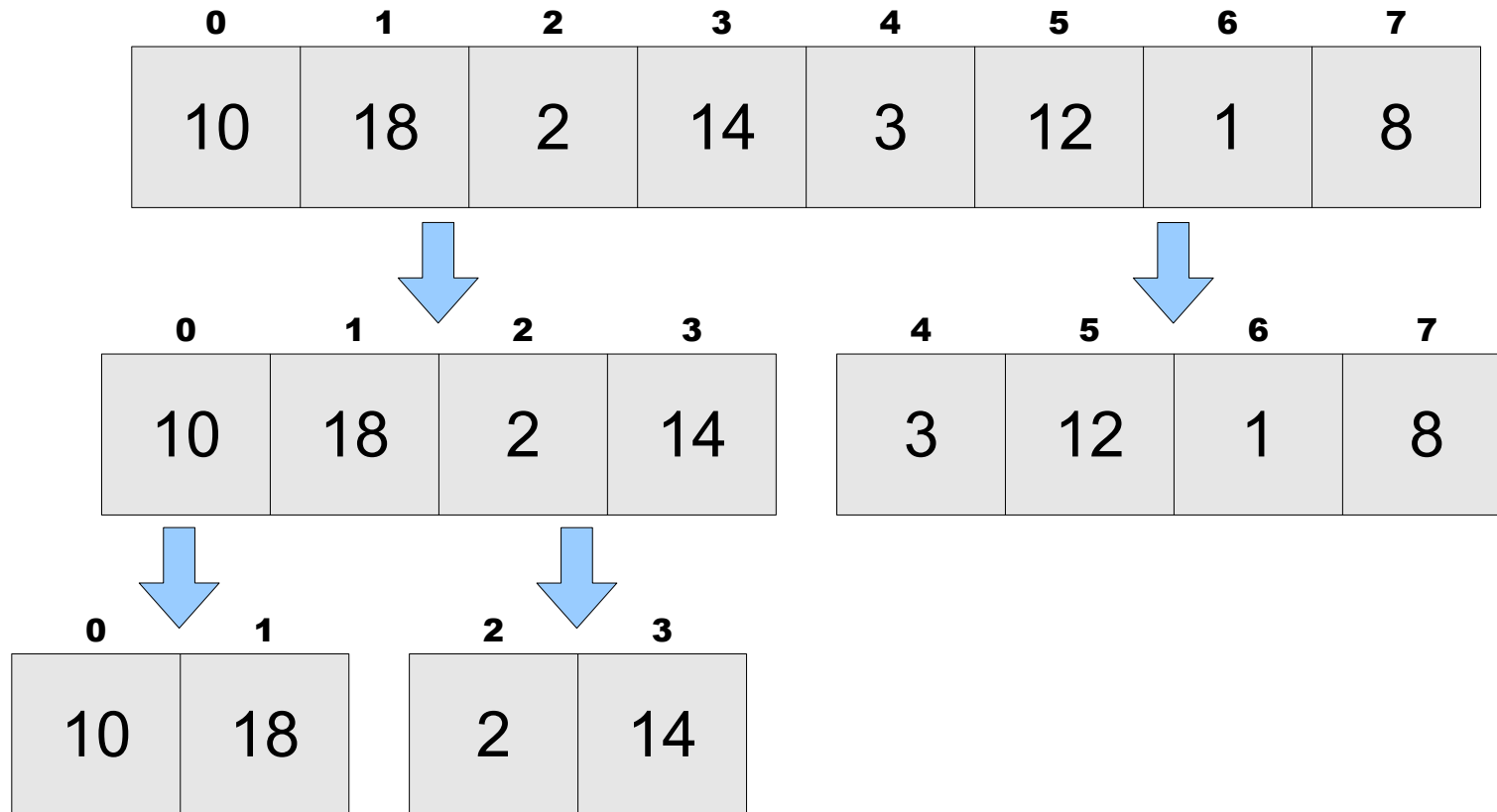
# Merge Sort

(break down the vector, then merge the pieces back together)



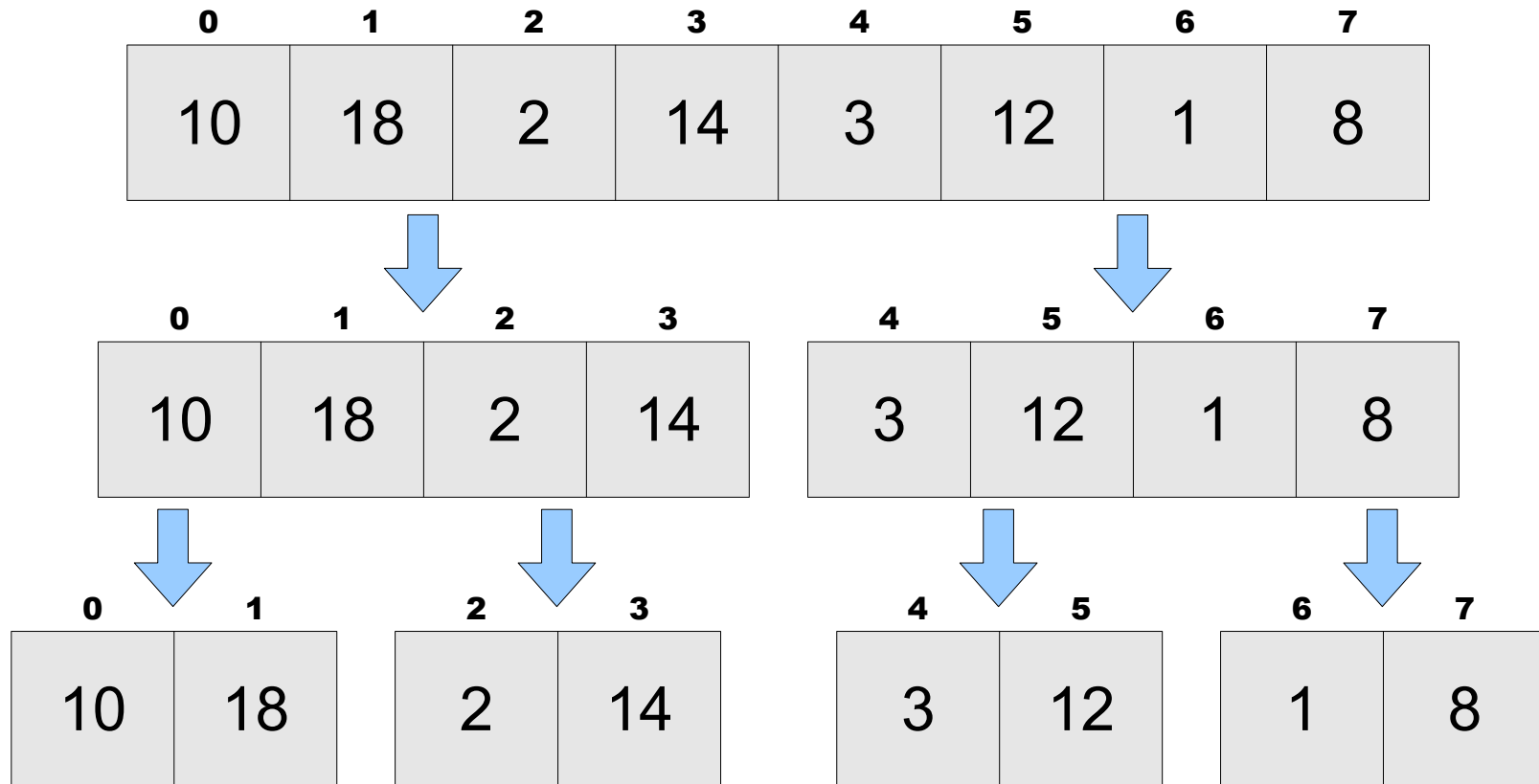
# Merge Sort

(break down the vector, then merge the pieces back together)



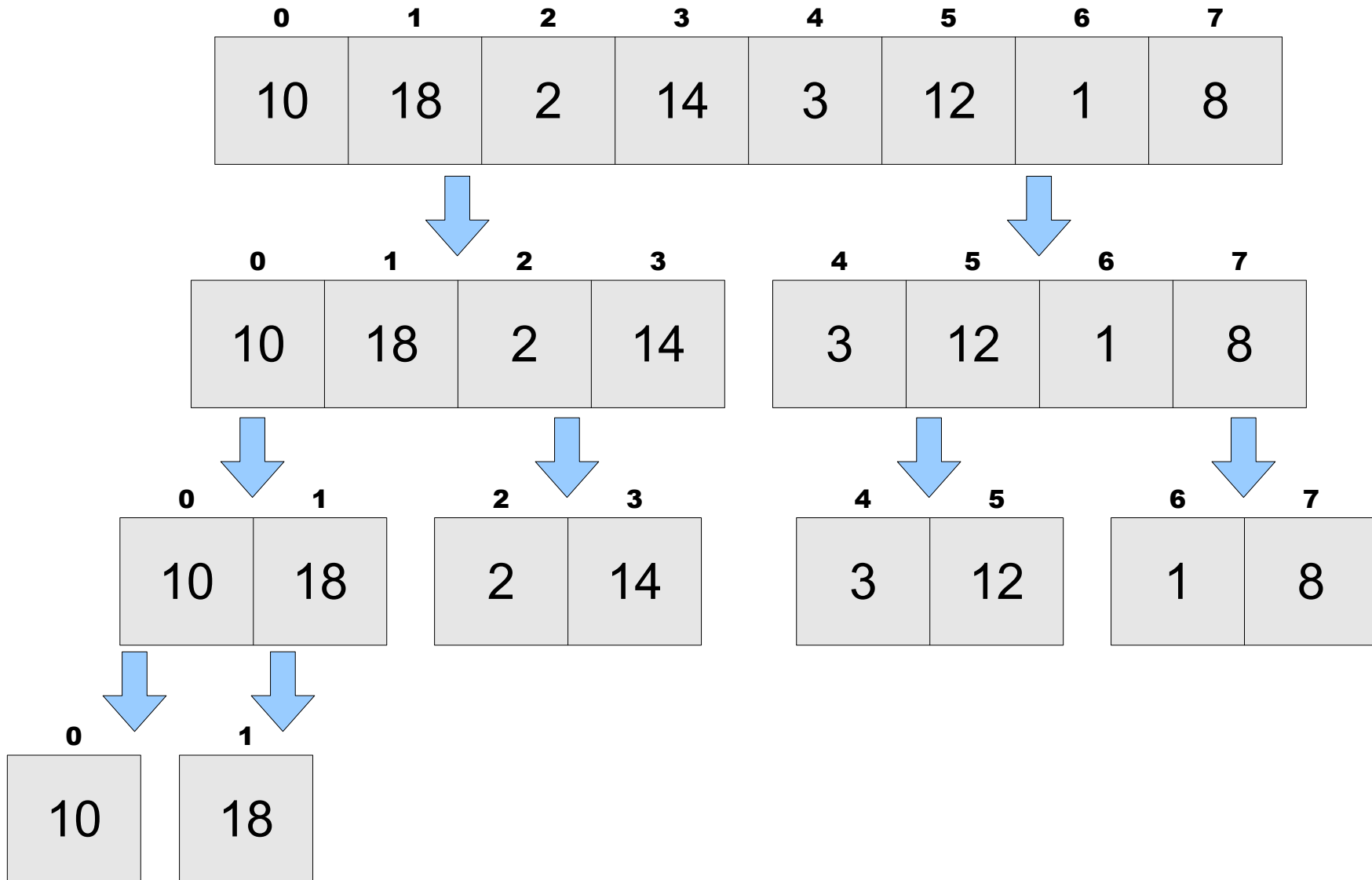
# Merge Sort

(break down the vector, then merge the pieces back together)



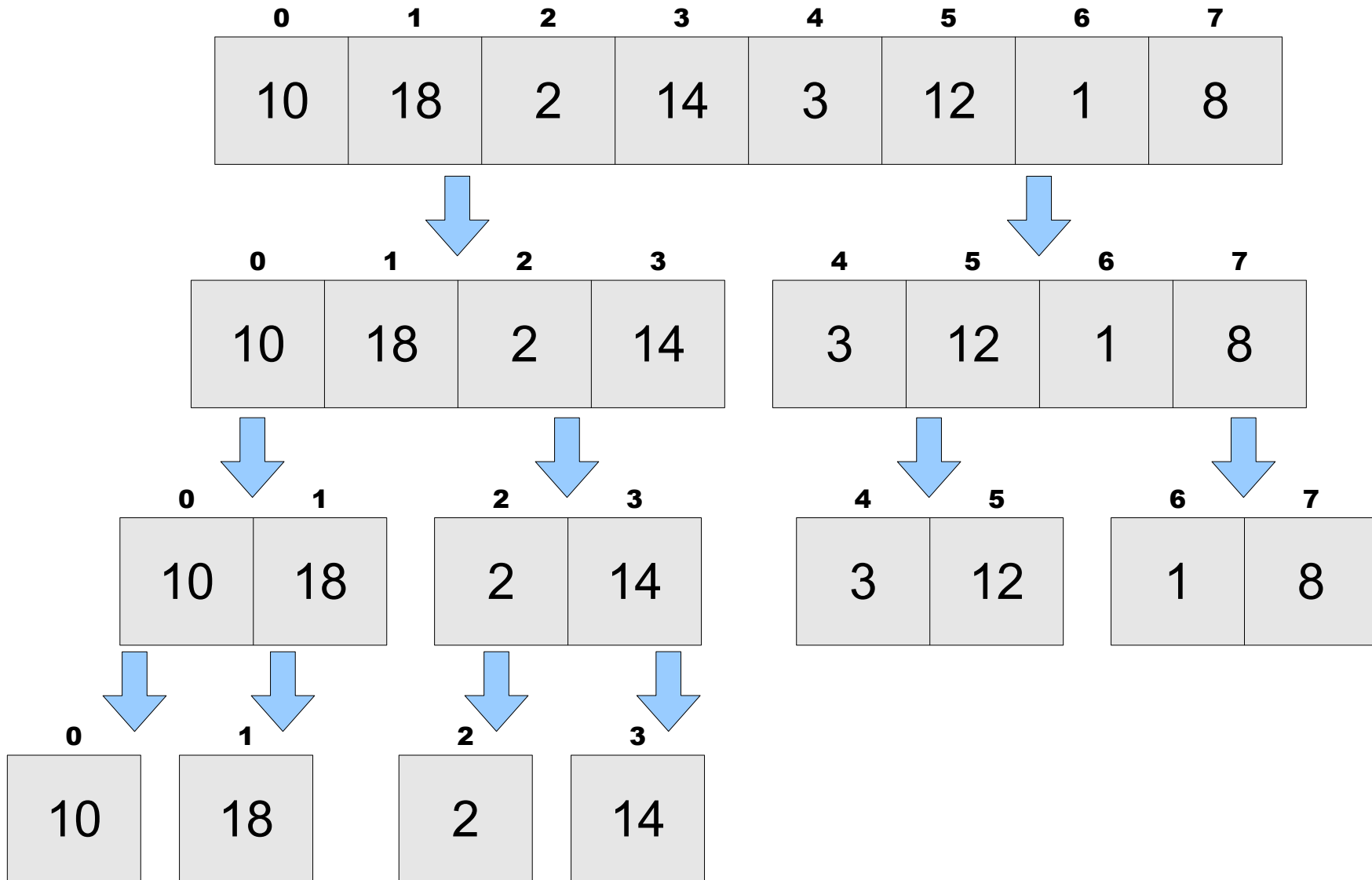
# Merge Sort

(break down the vector, then merge the pieces back together)



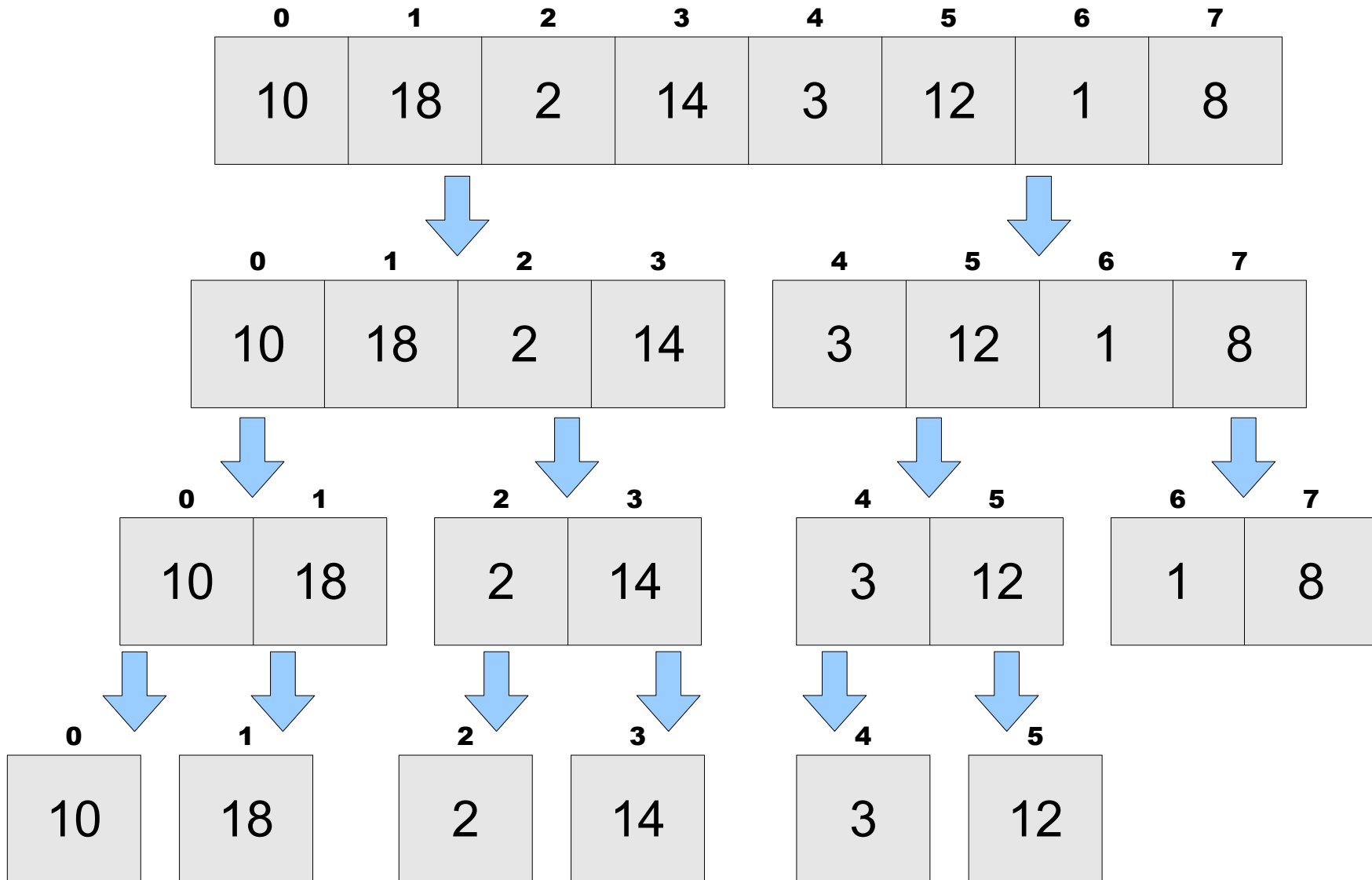
# Merge Sort

(break down the vector, then merge the pieces back together)



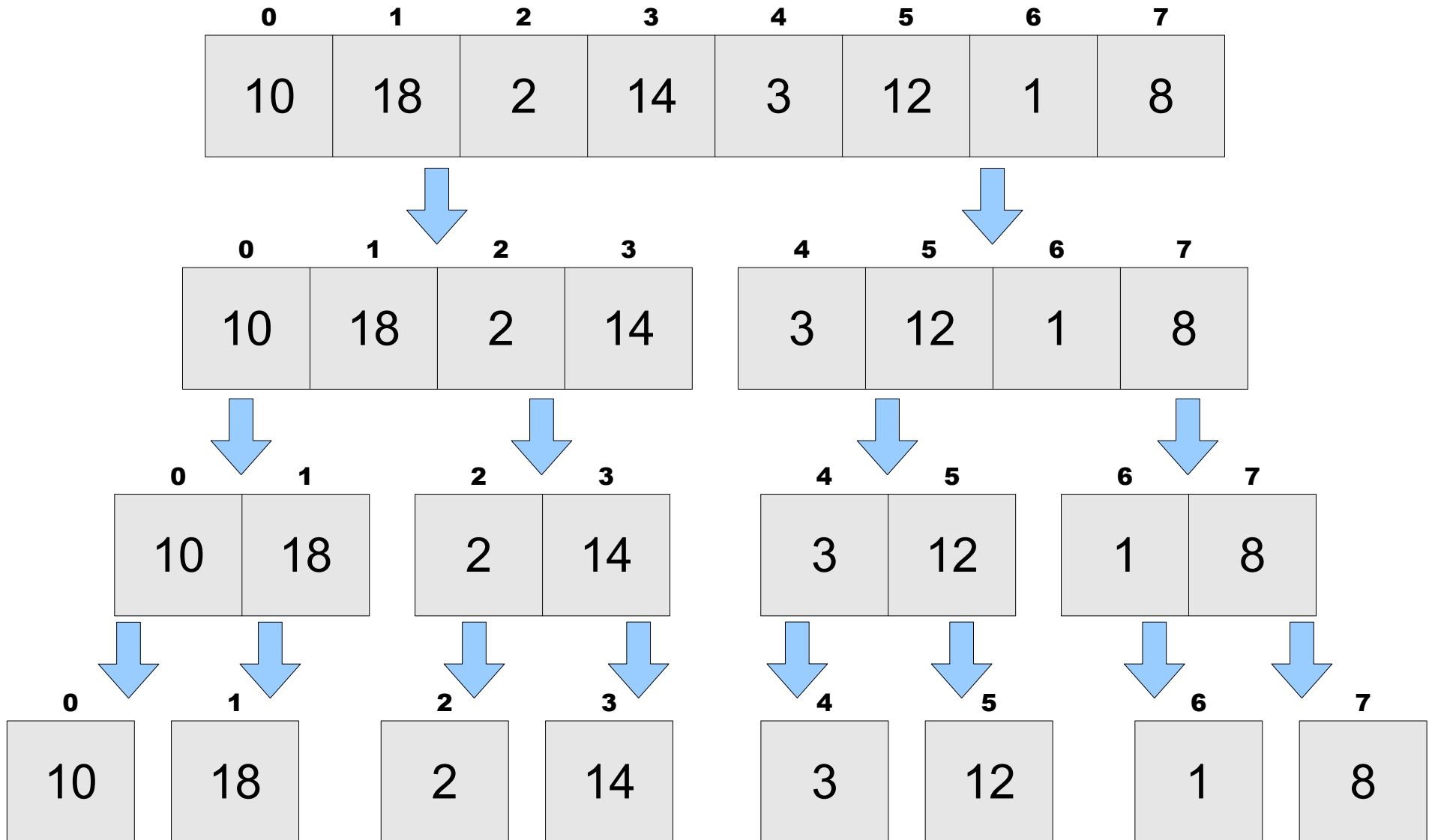
# Merge Sort

(break down the vector, then merge the pieces back together)



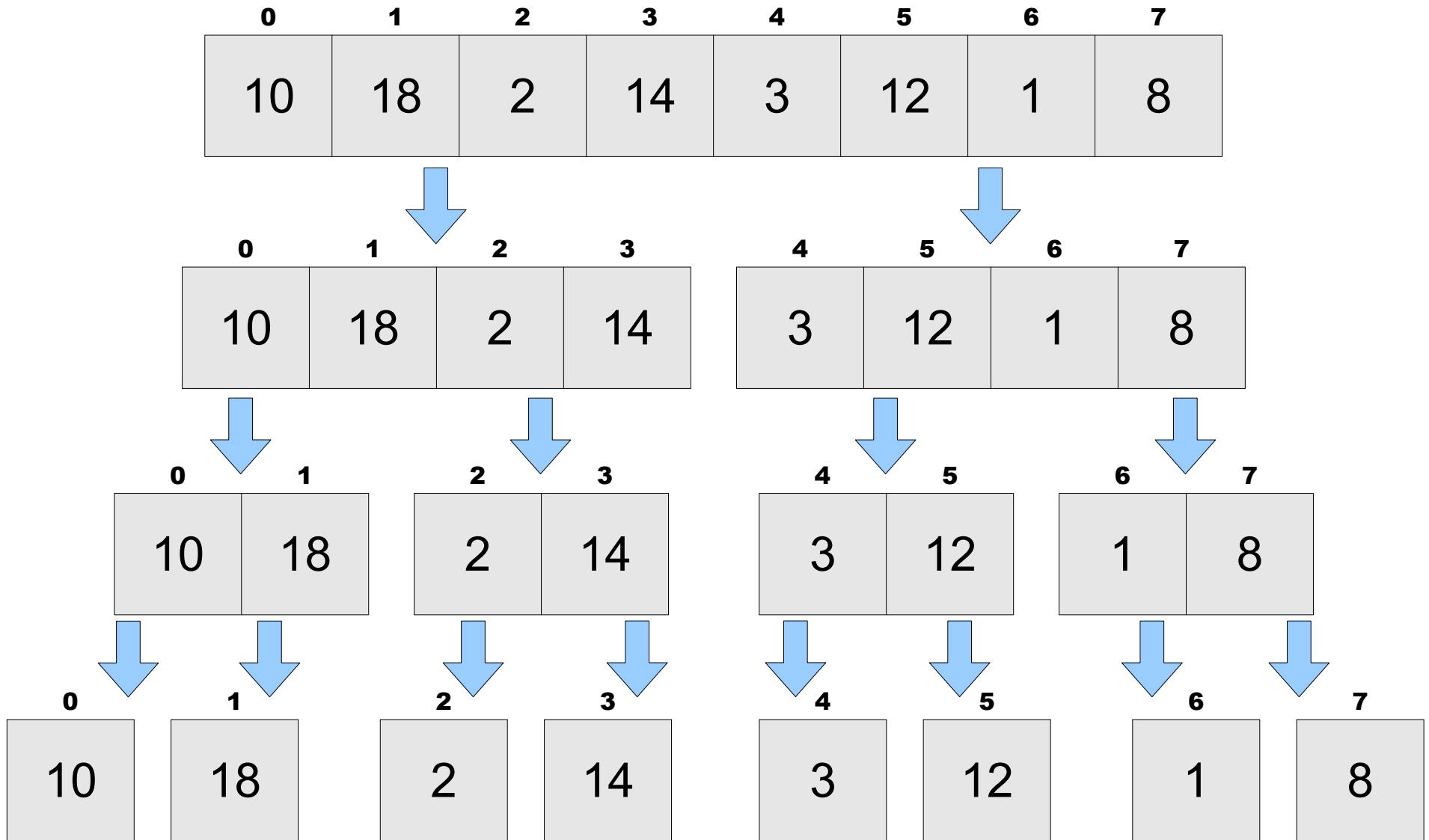
# Merge Sort

(break down the vector, then merge the pieces back together)



# Merge Sort

(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

We've reached our **base cases**. A vector with one element is **sorted**.

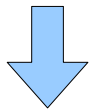
# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8



0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

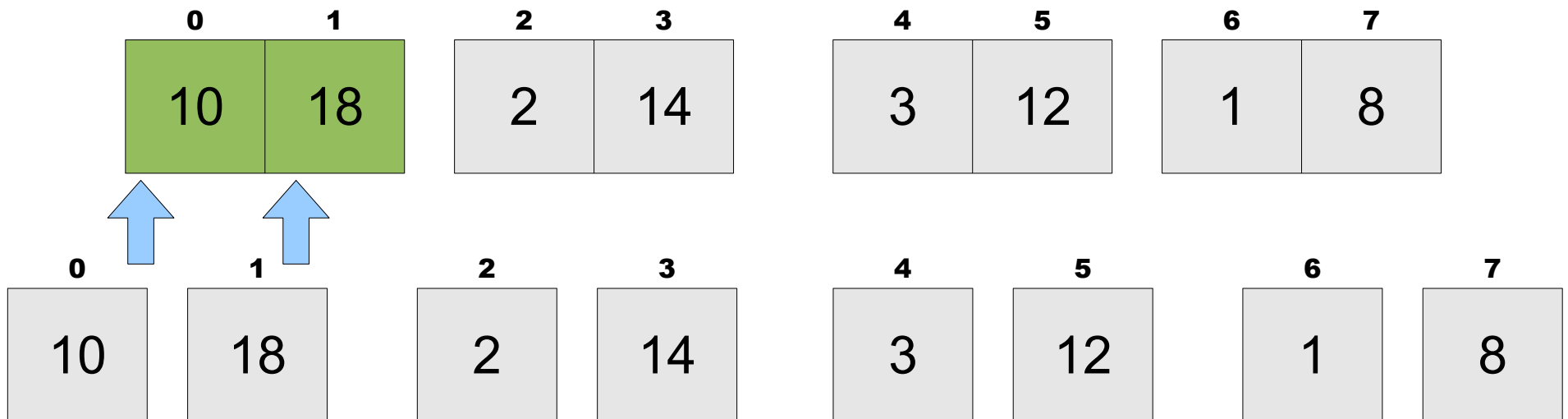
We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1
10	18

2	3
2	14

4	5
3	12

6	7
1	8

0	1
10	18

2	3
2	14

4	5
3	12

6
1

7
8

We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1
10	18

2	3
2	14

4	5
3	12

6	7
1	8

0	1
10	18

2	3
2	14

4	5
3	12

6
1

7
8

We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1
10	18

2	3
2	14

4	5
3	12

6	7
1	8

0	1
10	18

2	3
2	14

4
3

5
12

6
1

7
8

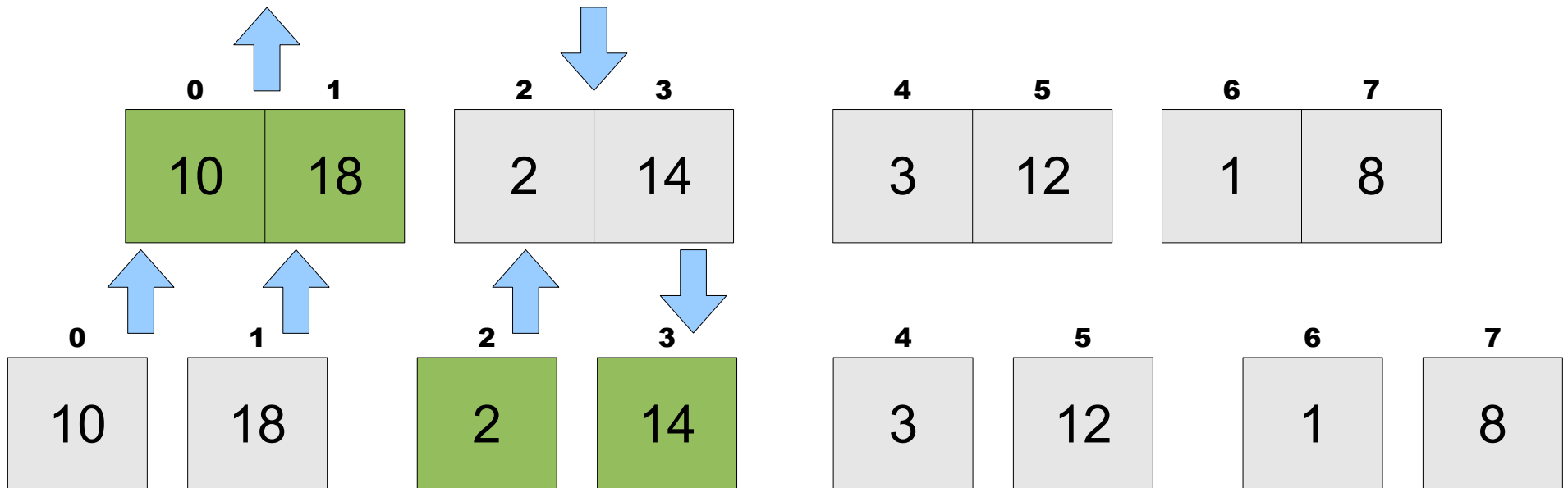
We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3
10	18	2	14

4	5	6	7
3	12	1	8

0	1
10	18

2	3
2	14

4	5
3	12

6	7
1	8

0	1
10	18

2	3
2	14

4
3

5
12

6
1

7
8

We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3
10	18	2	14

4	5	6	7
3	12	1	8

0	1
10	18

2	3
2	14

4	5
3	12

6	7
1	8

0	1
10	18

2	3
2	14

4
3

5
12

6
1

7
8

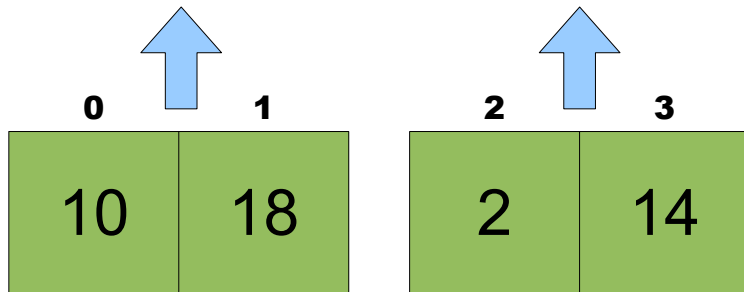
We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

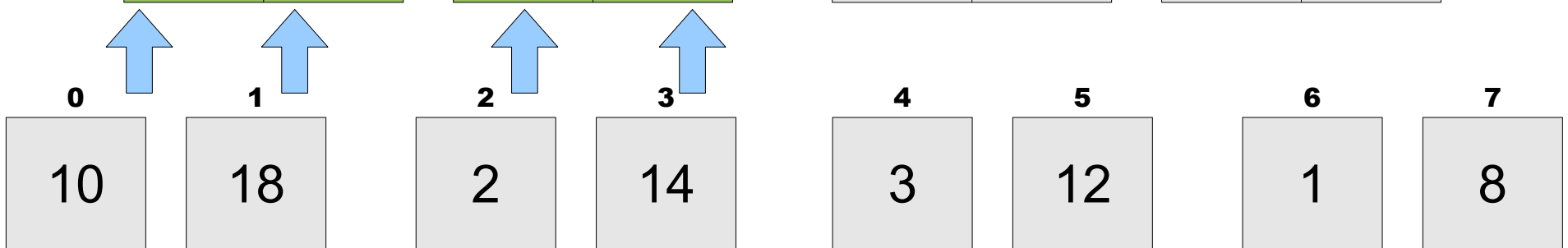
(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8



4	5	6	7
3	12	1	8



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

0	1	2	3	4	5	6	7
2	10	14	18	3	12	1	8

0	1	2	3
10	18	2	14

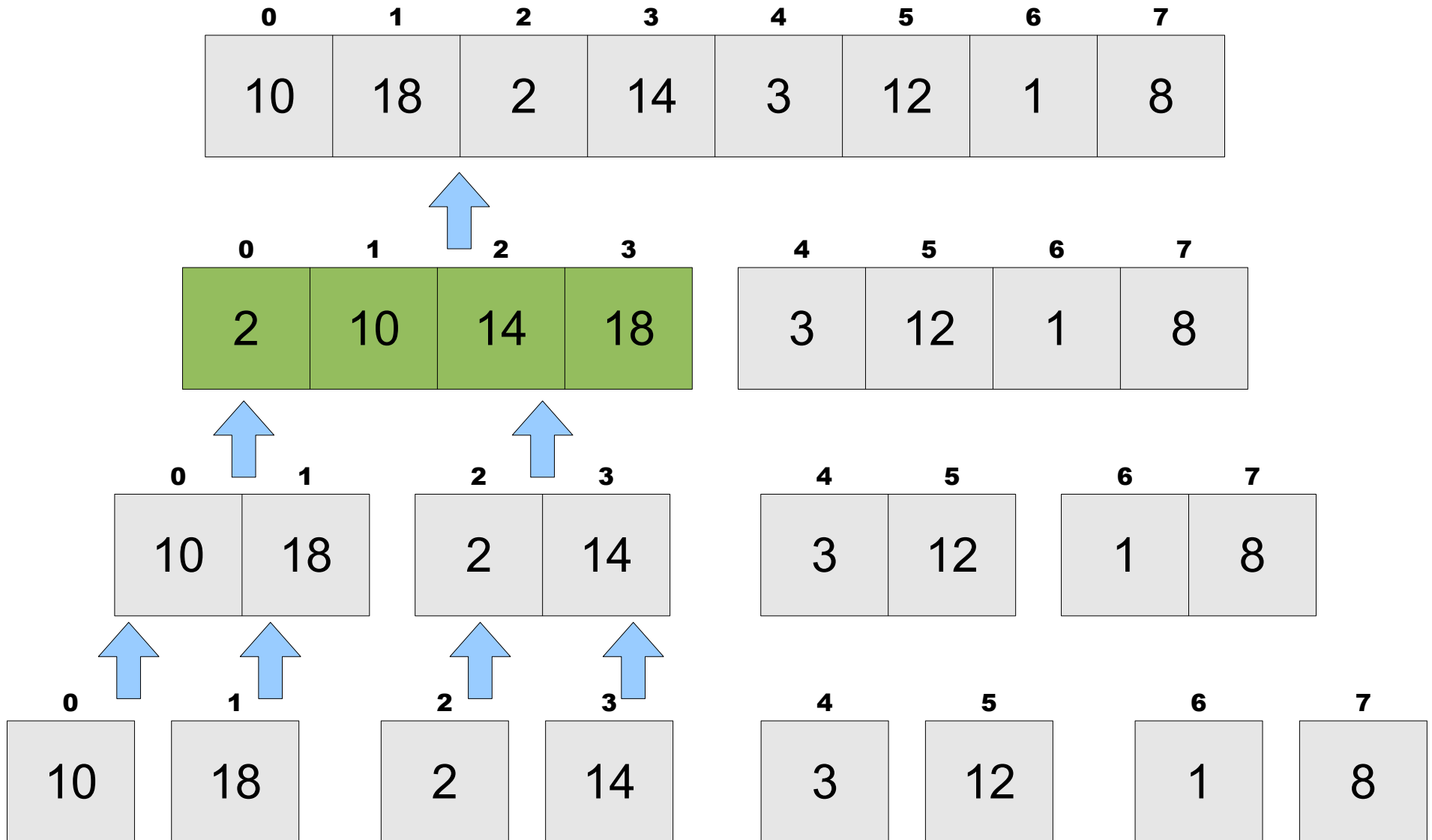
4	5	6	7
3	12	1	8

0	1	2	3	4	5	6	7
10	18	2	14	3	12	1	8

We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

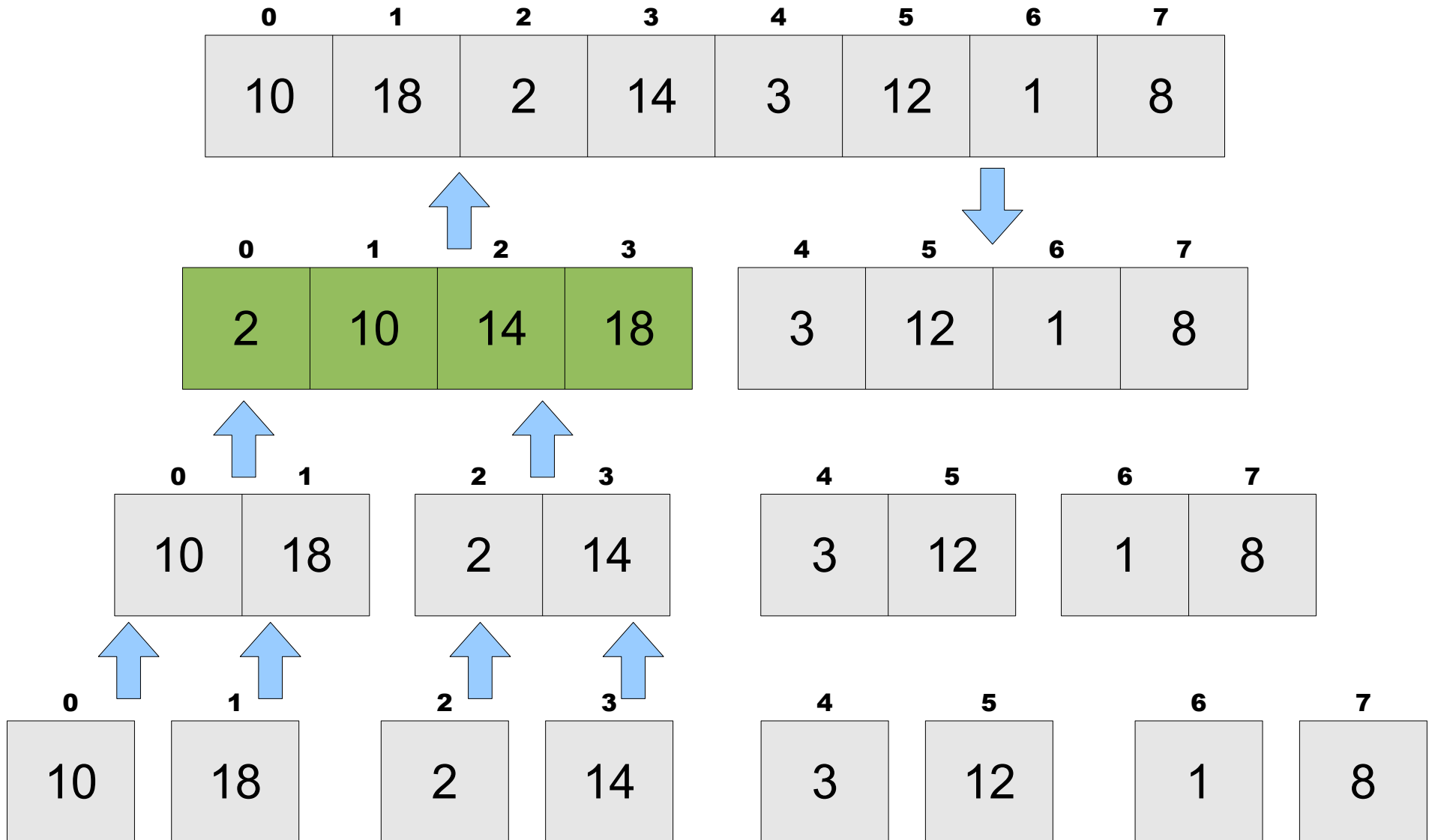
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

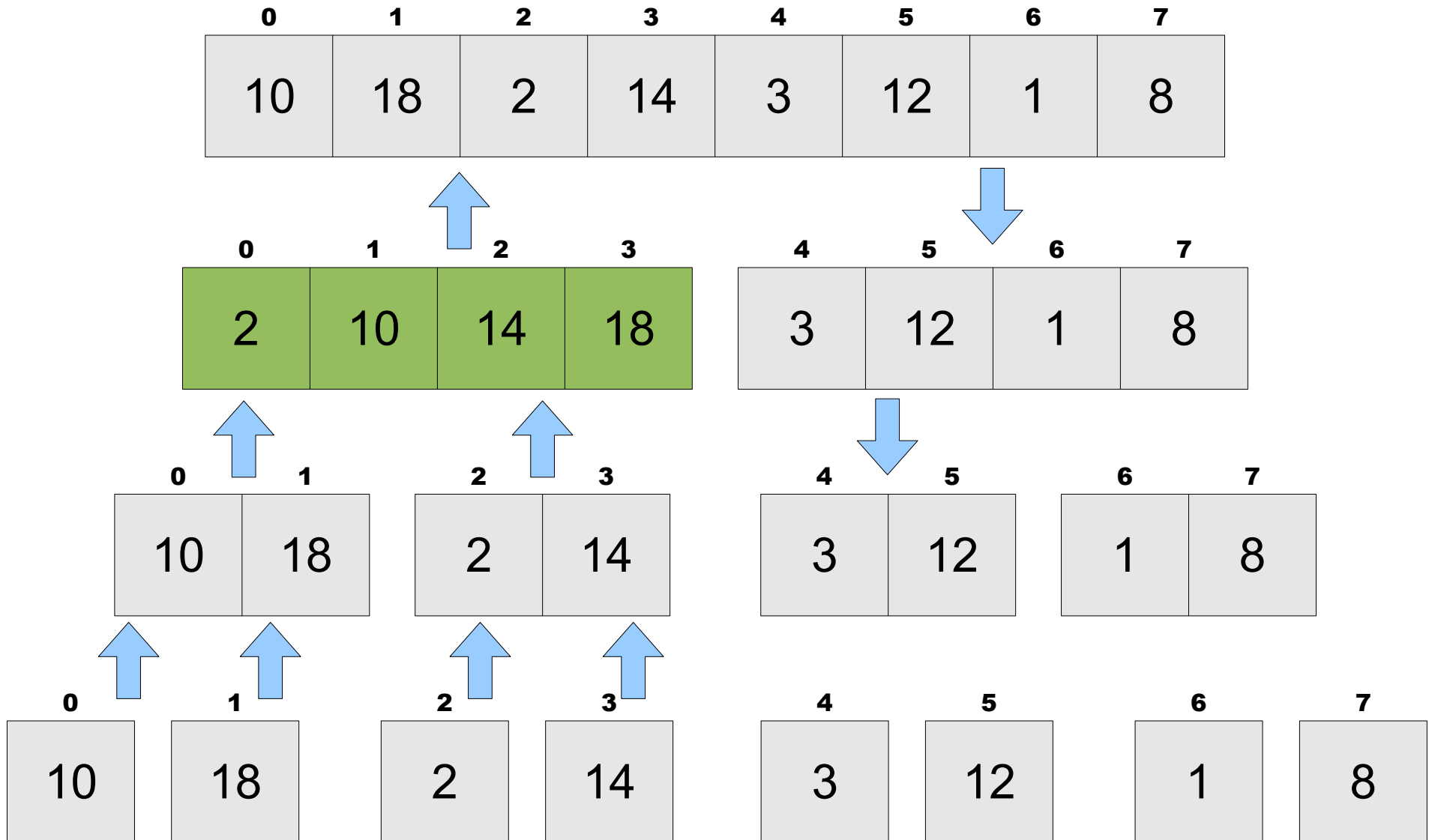
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

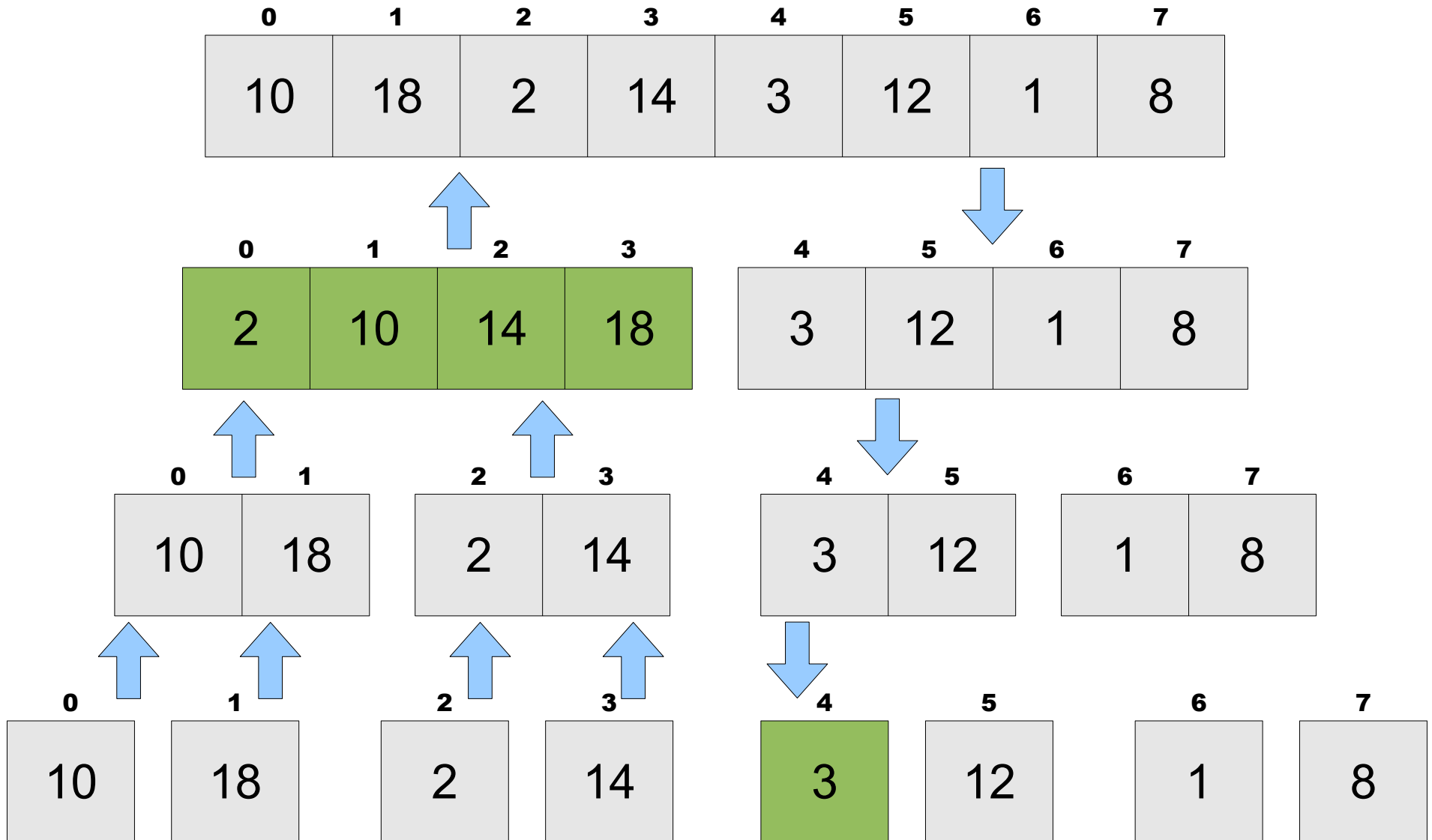
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

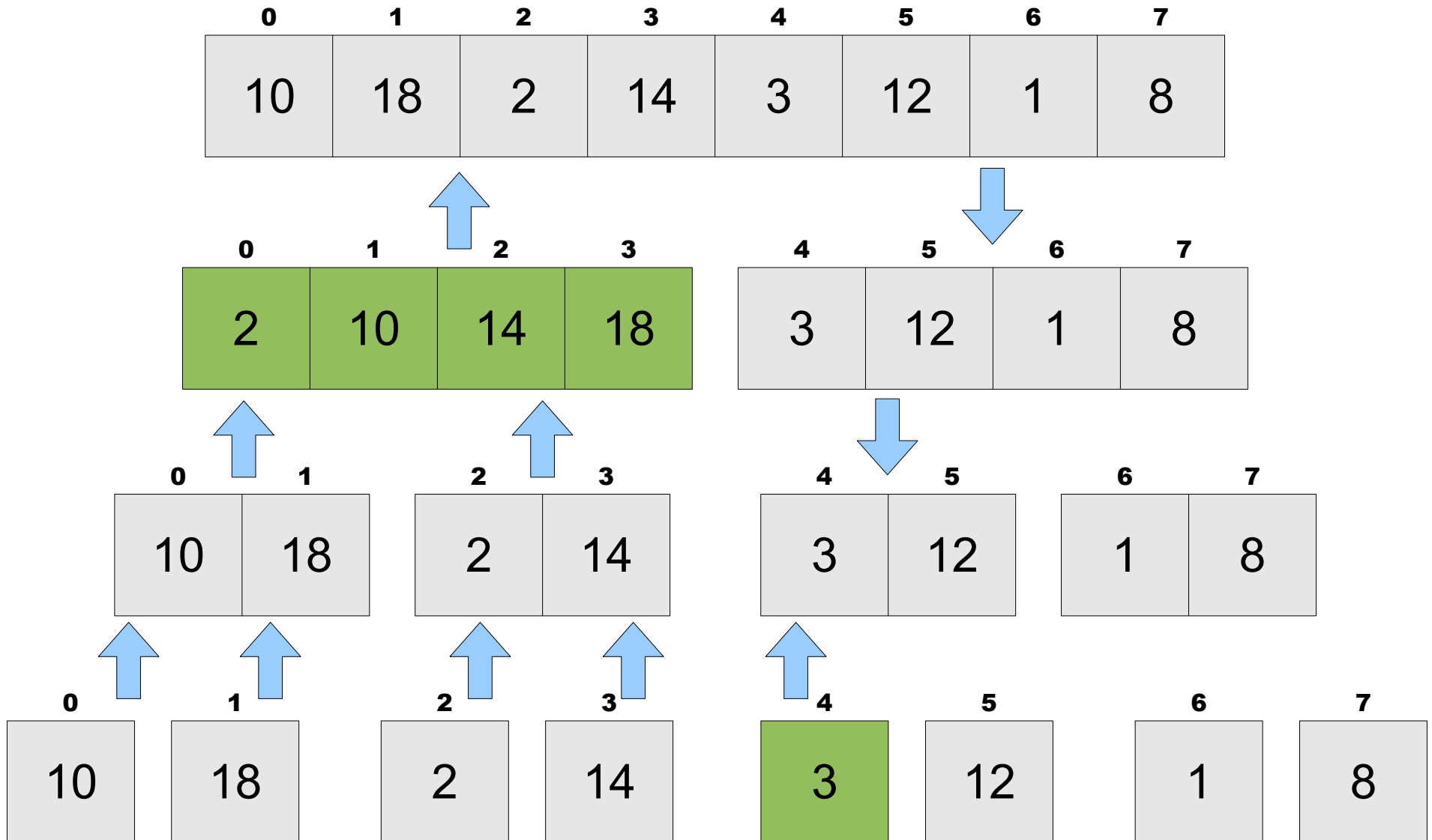
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

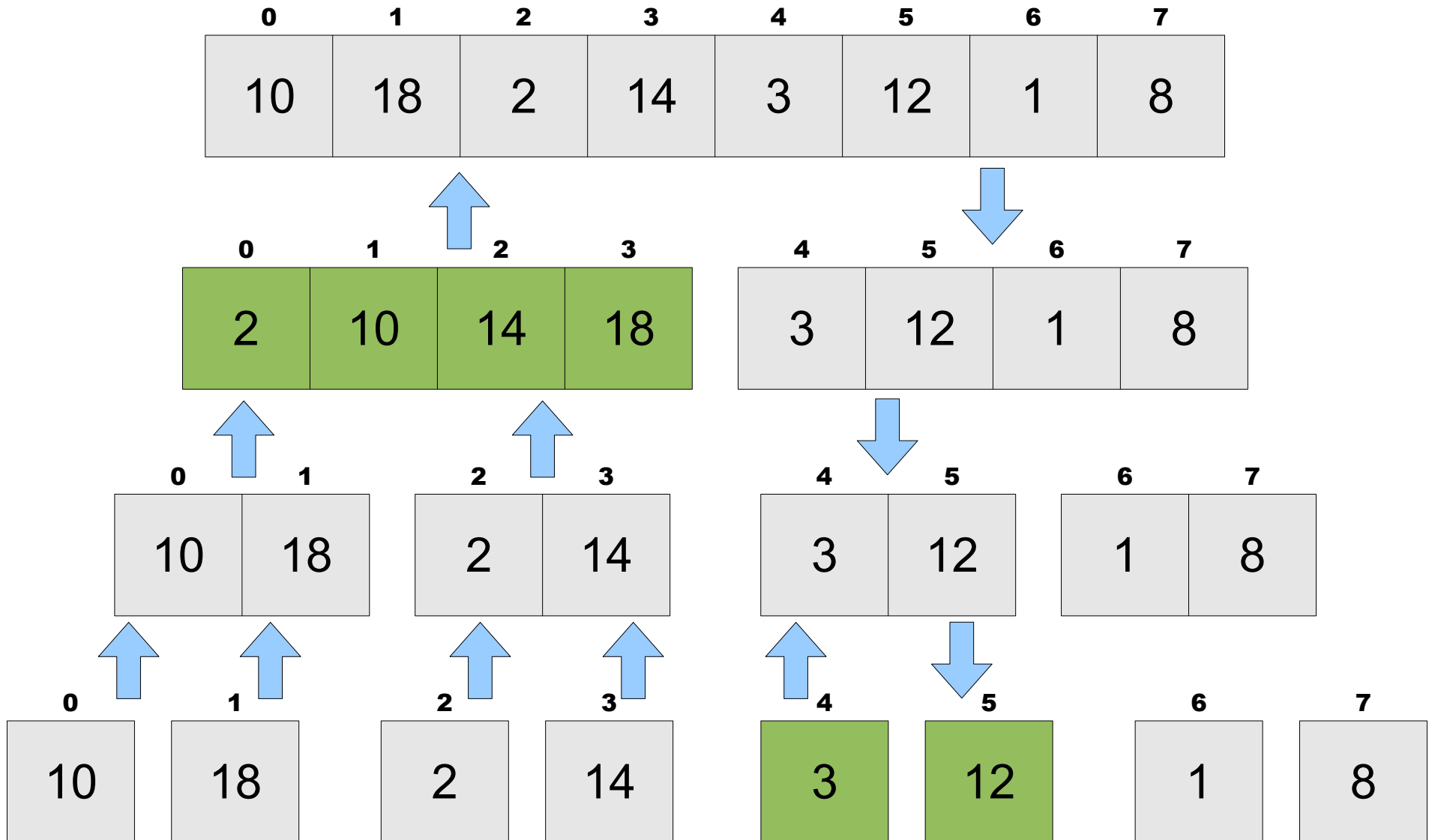
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

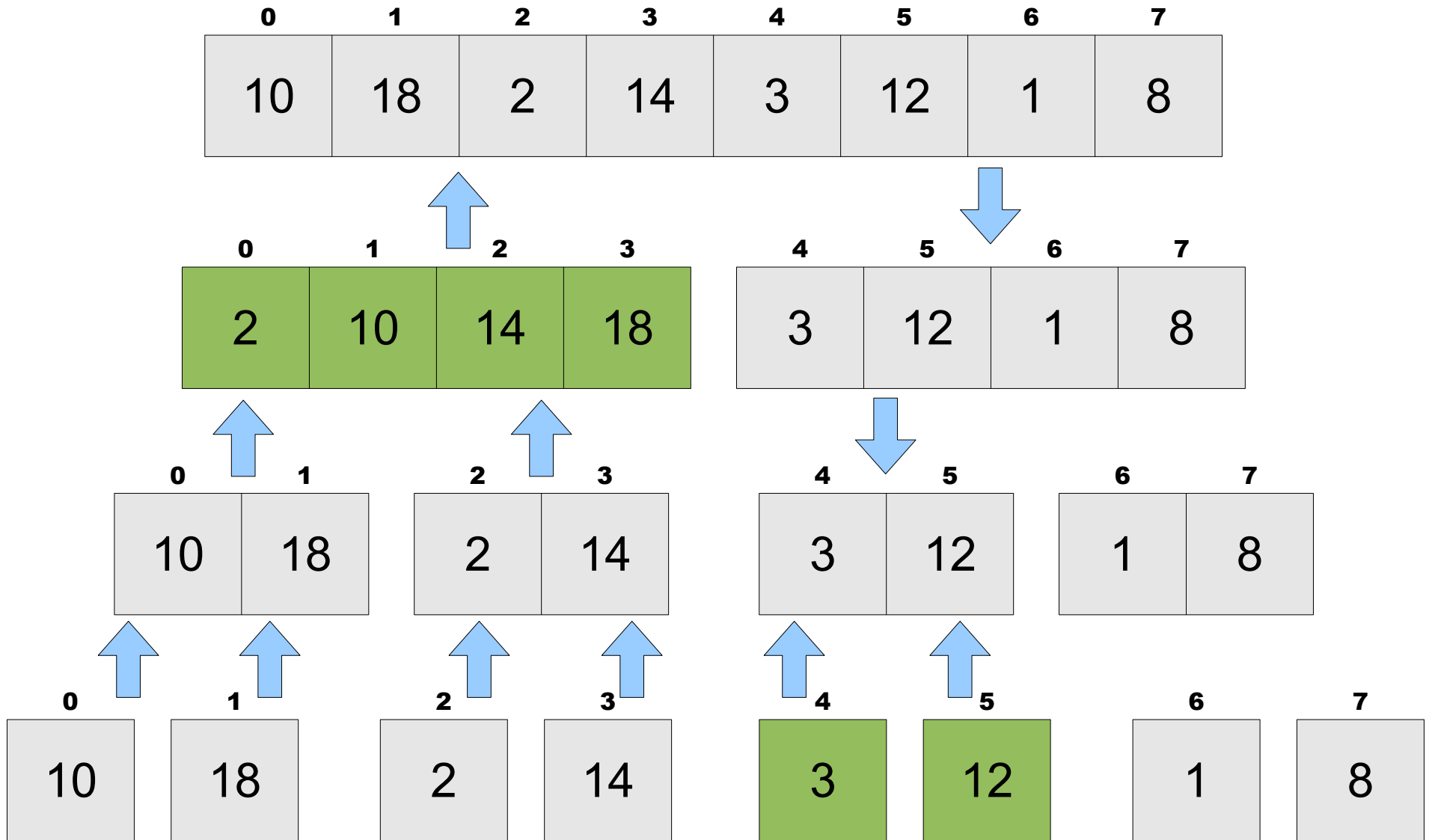
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

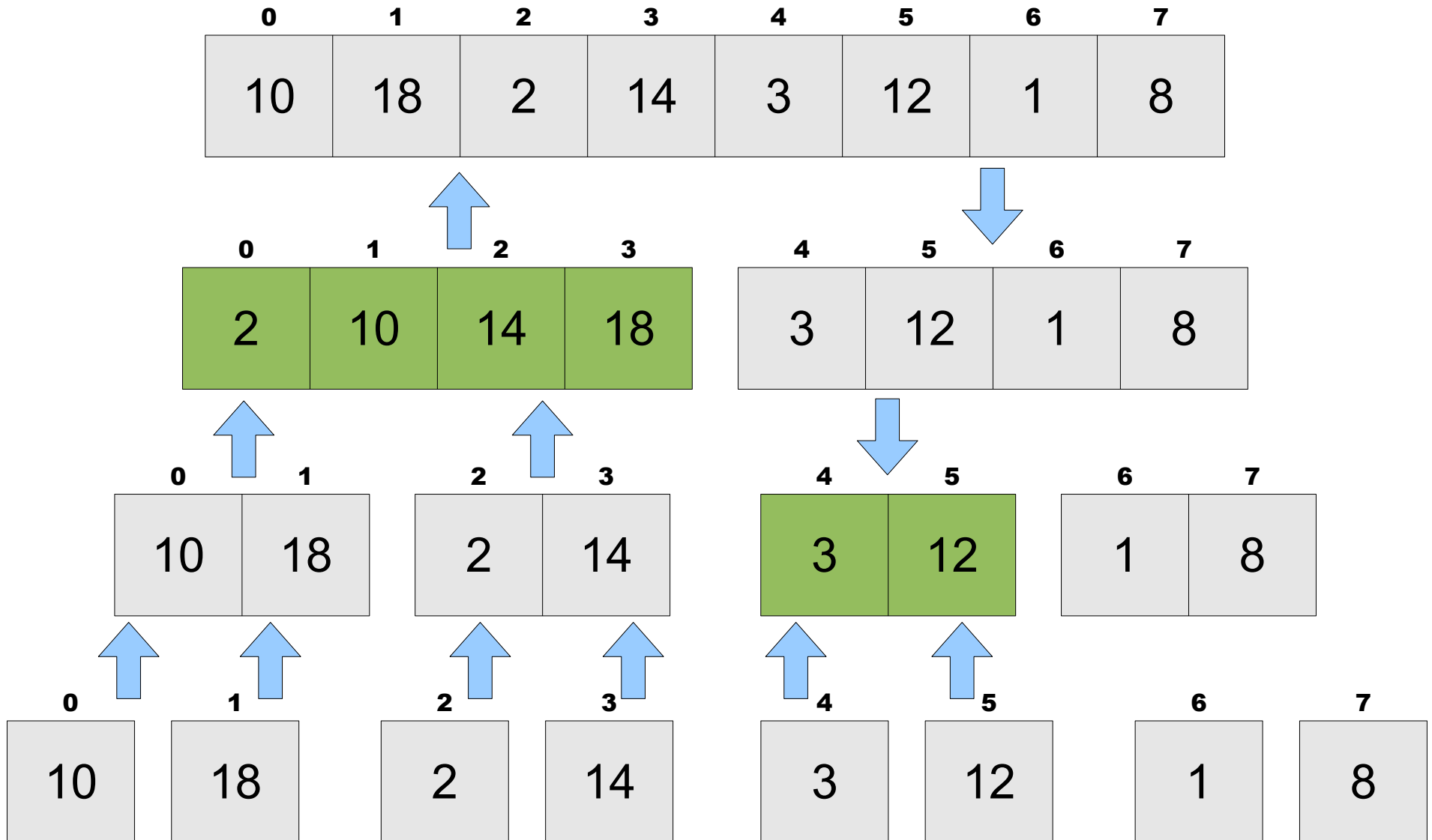
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

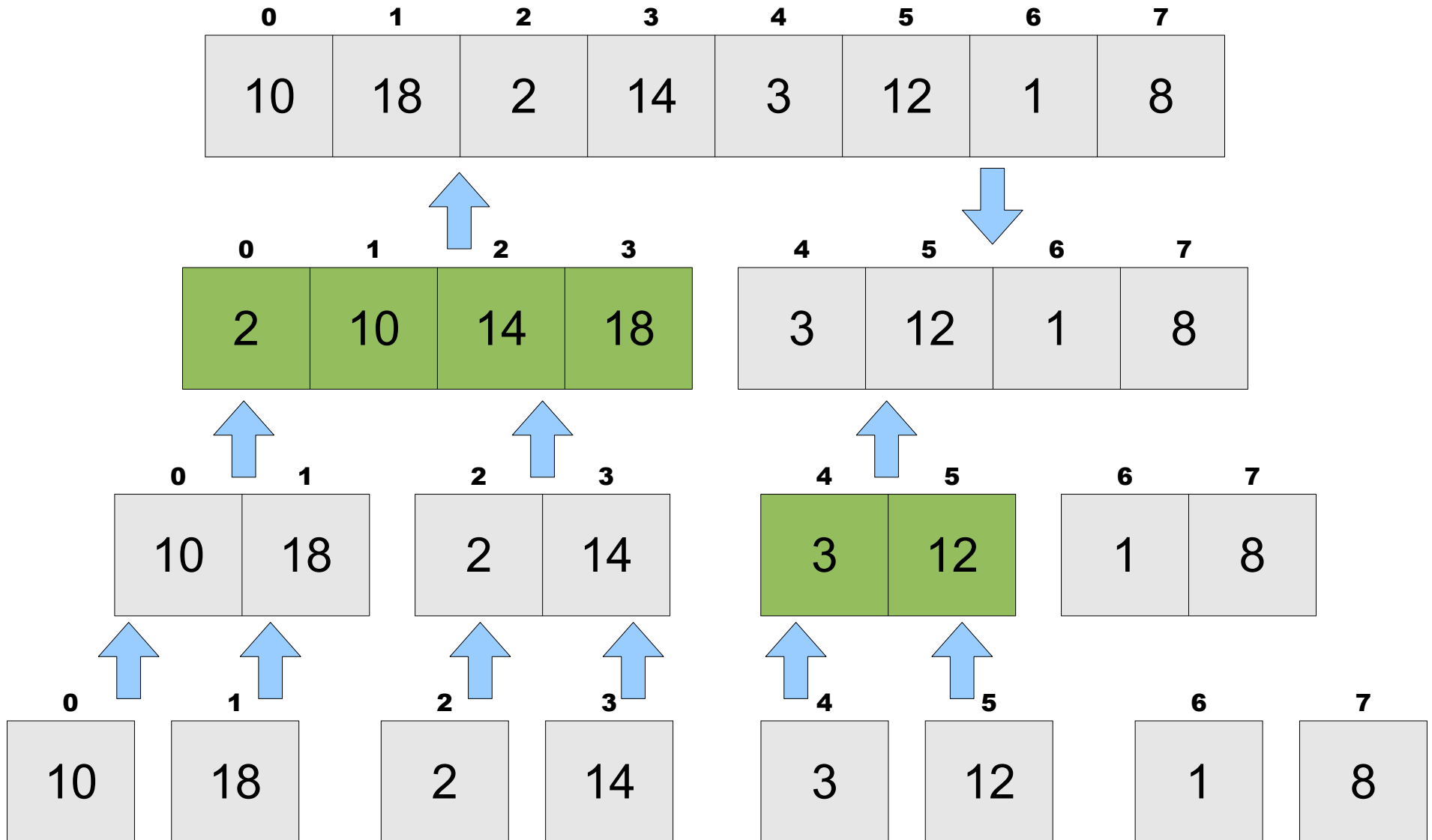
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

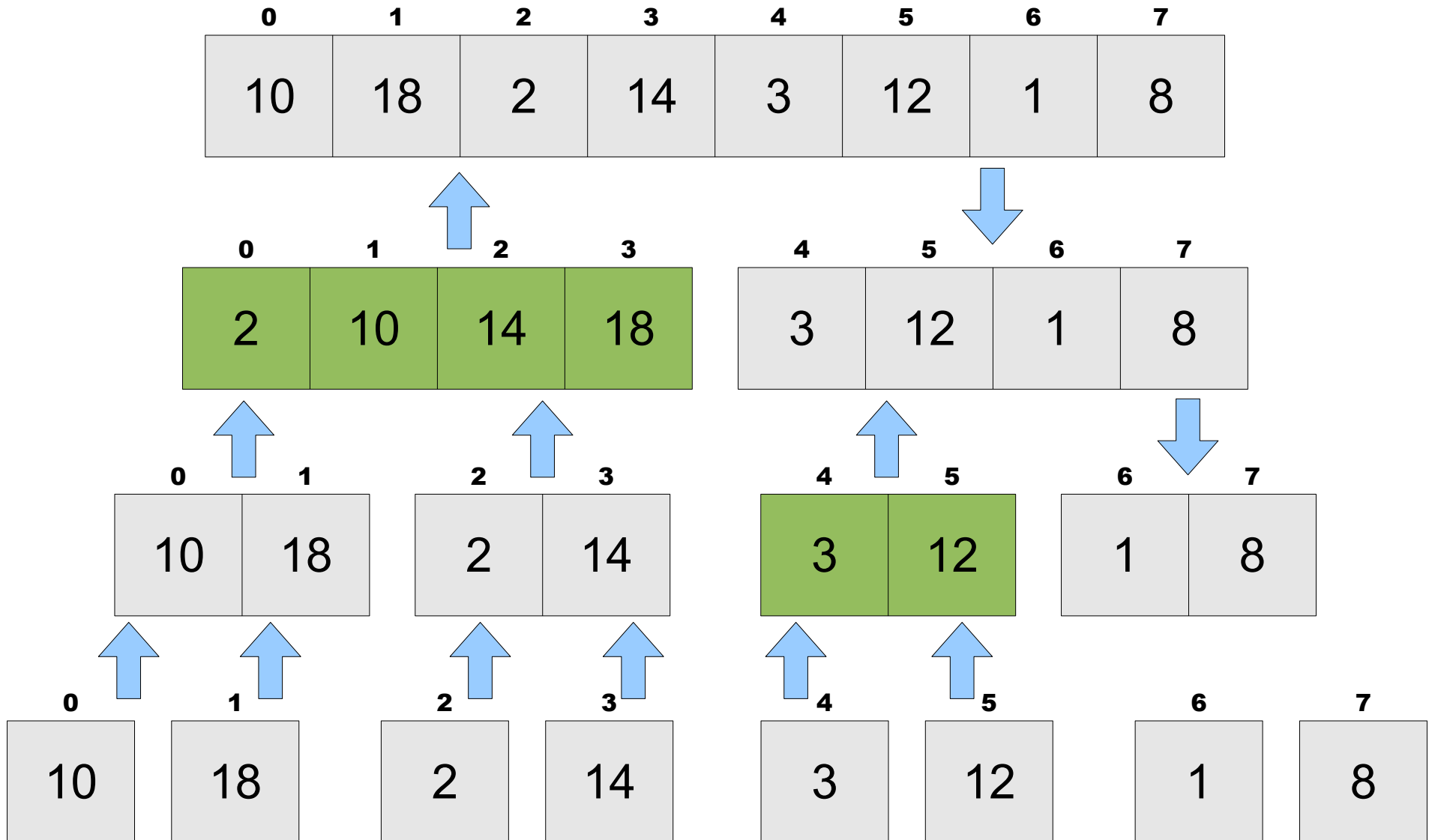
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

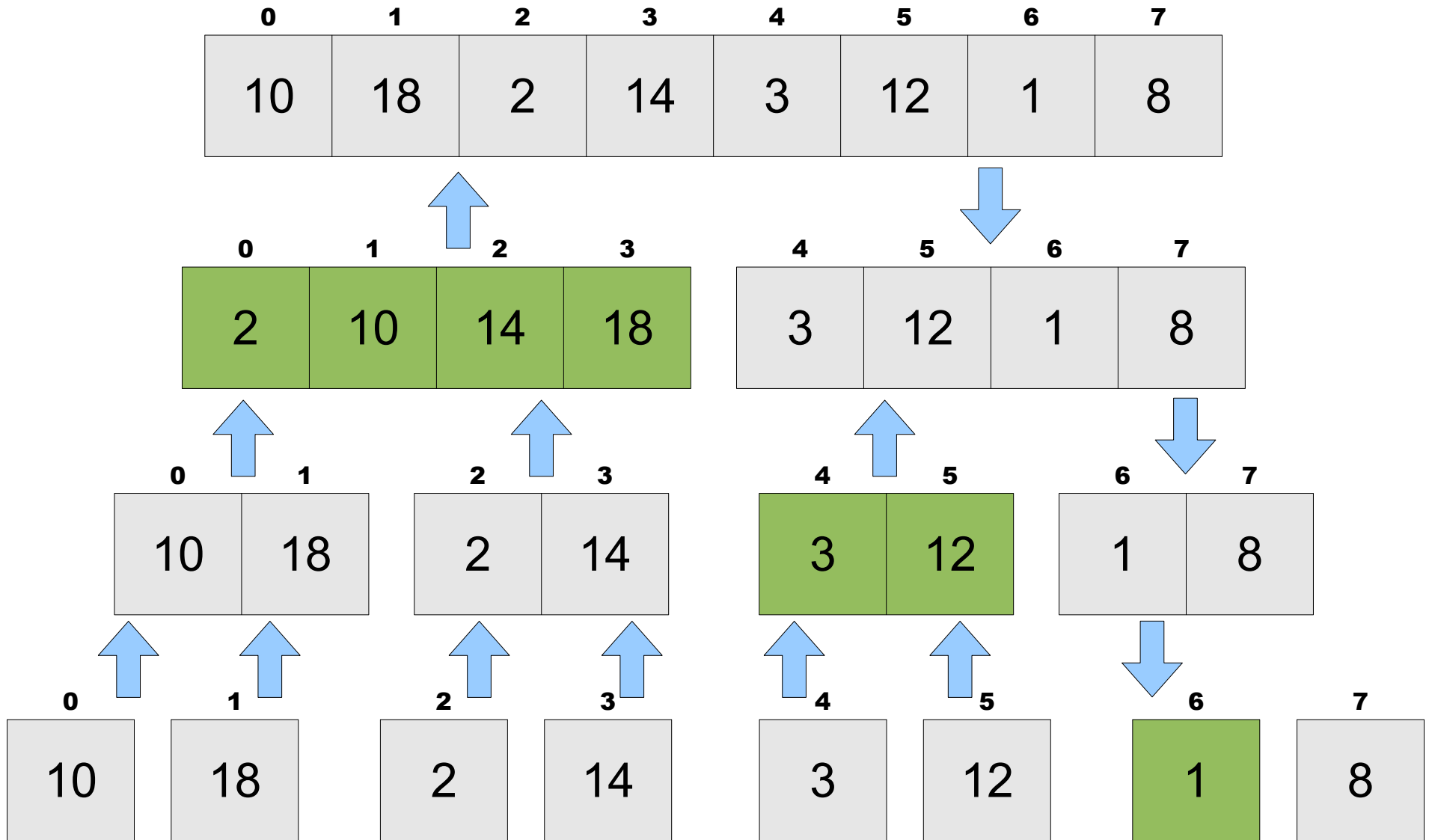
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

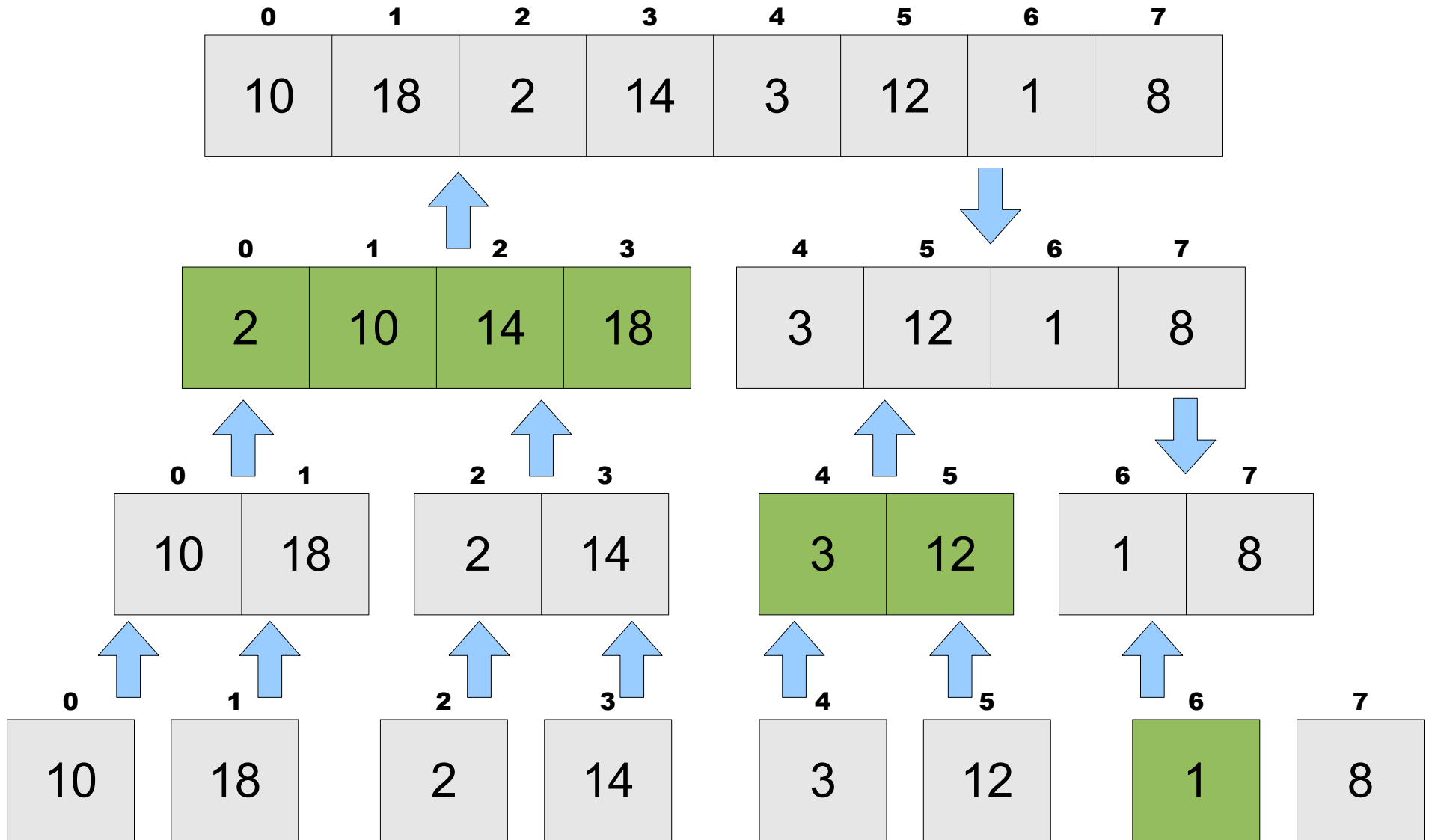
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

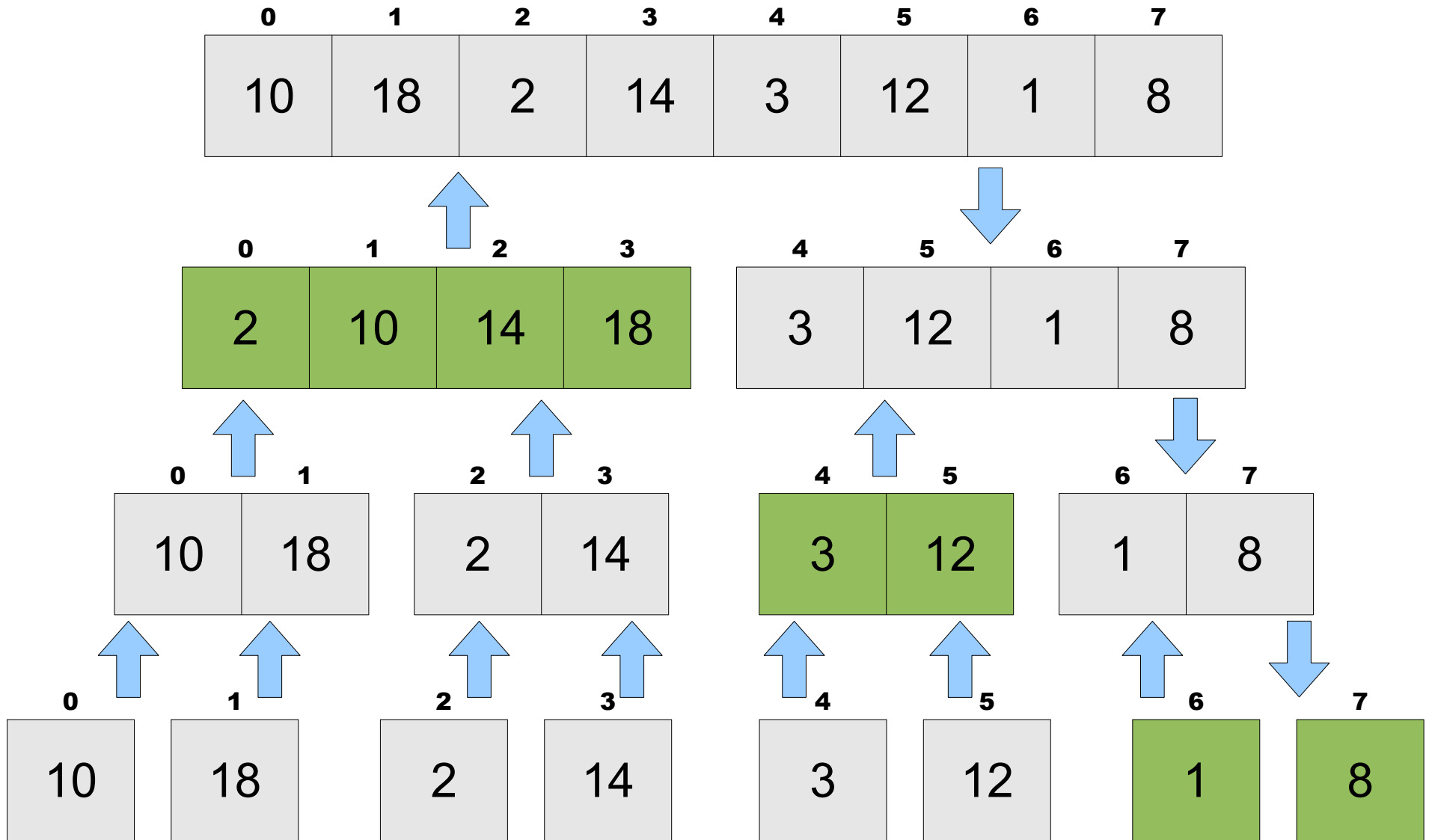
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

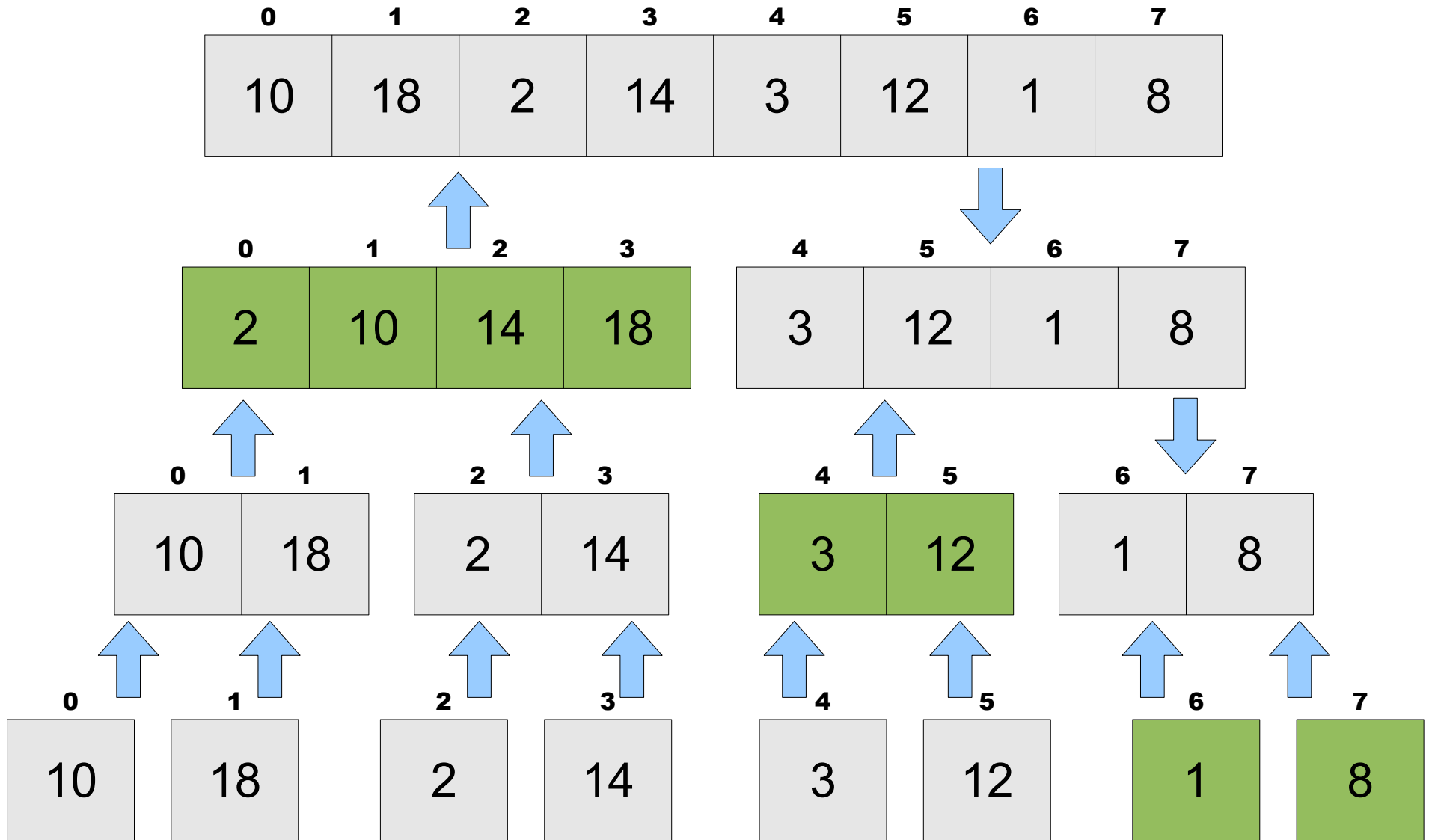
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

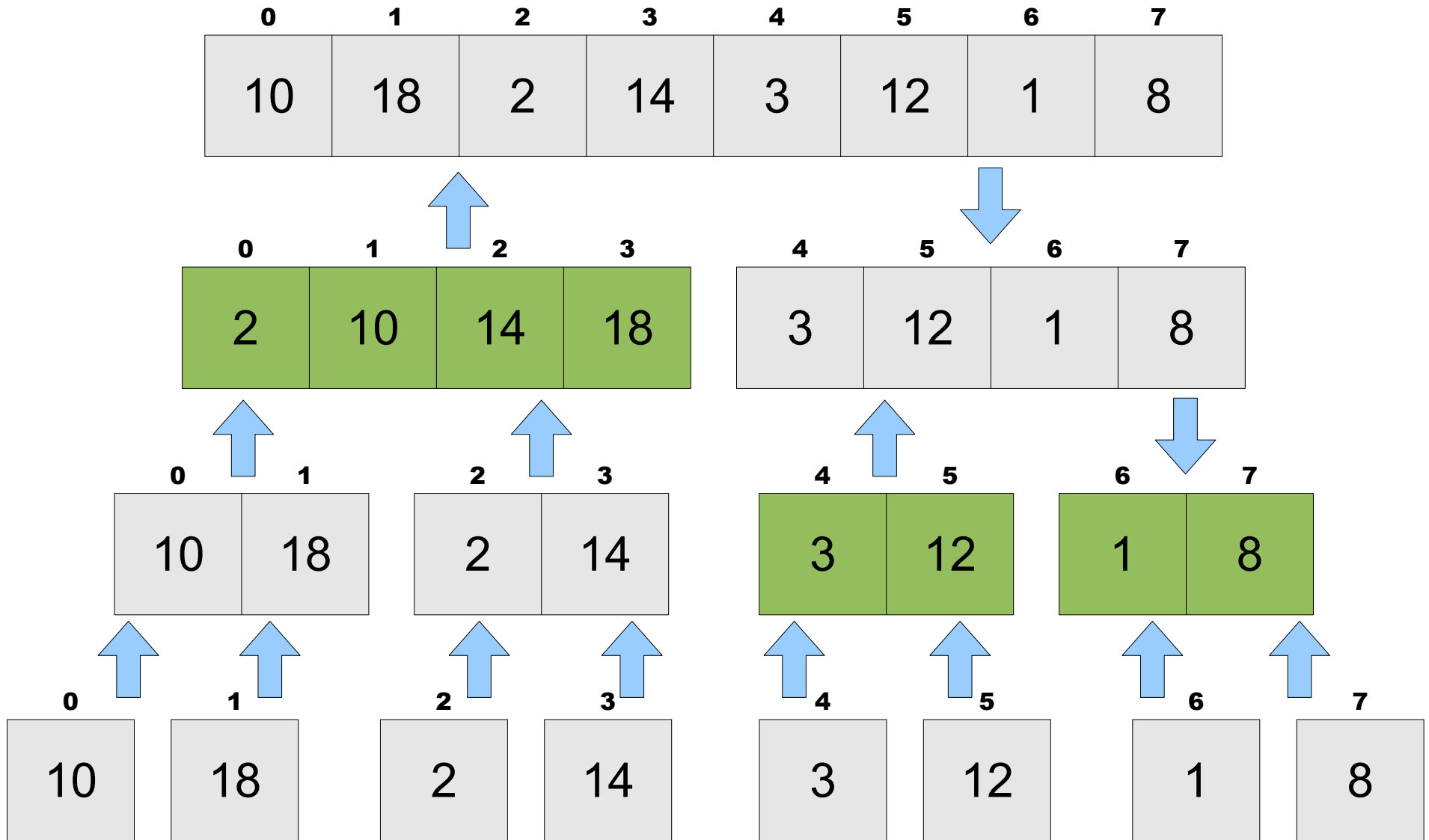
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

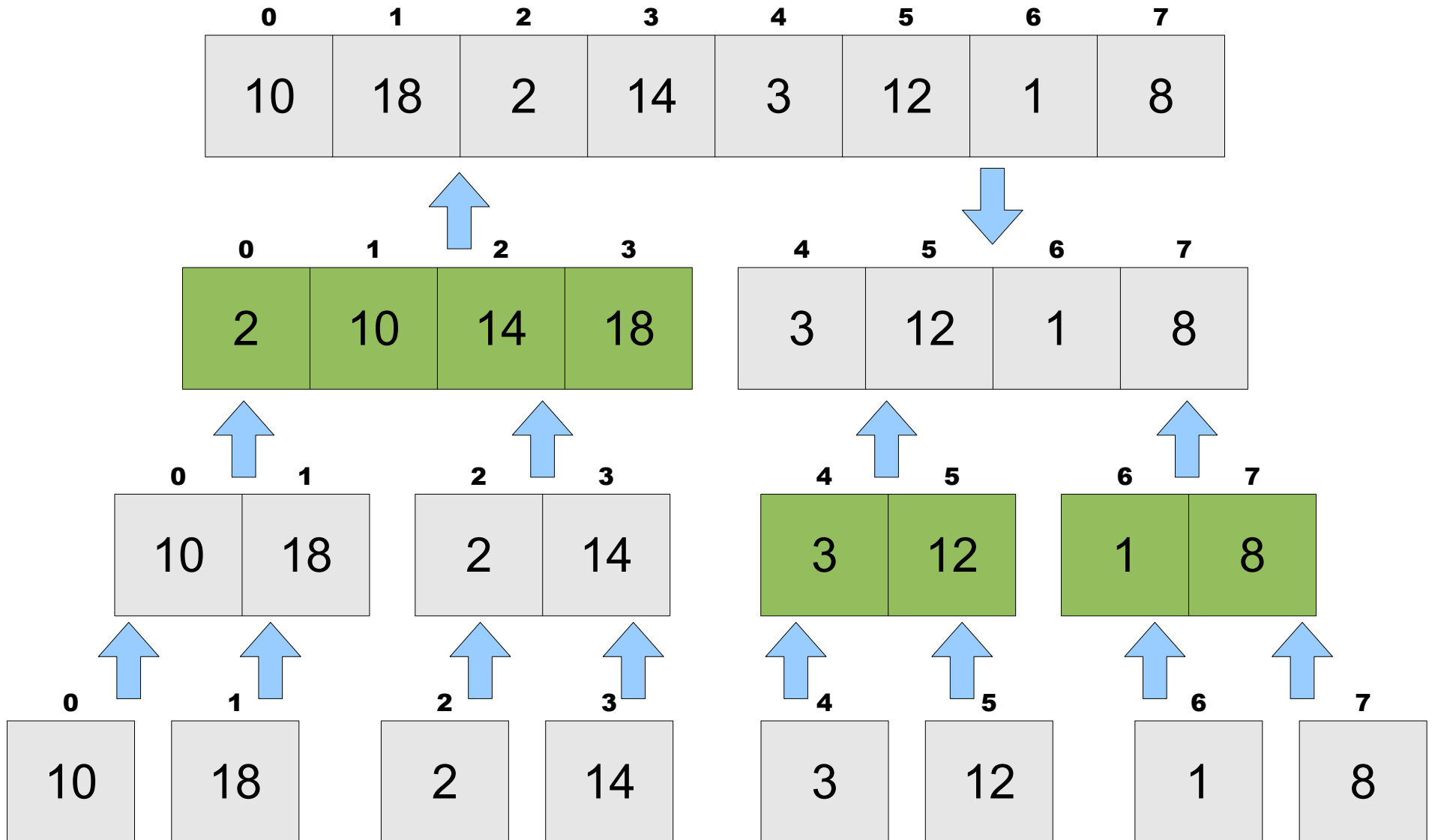
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

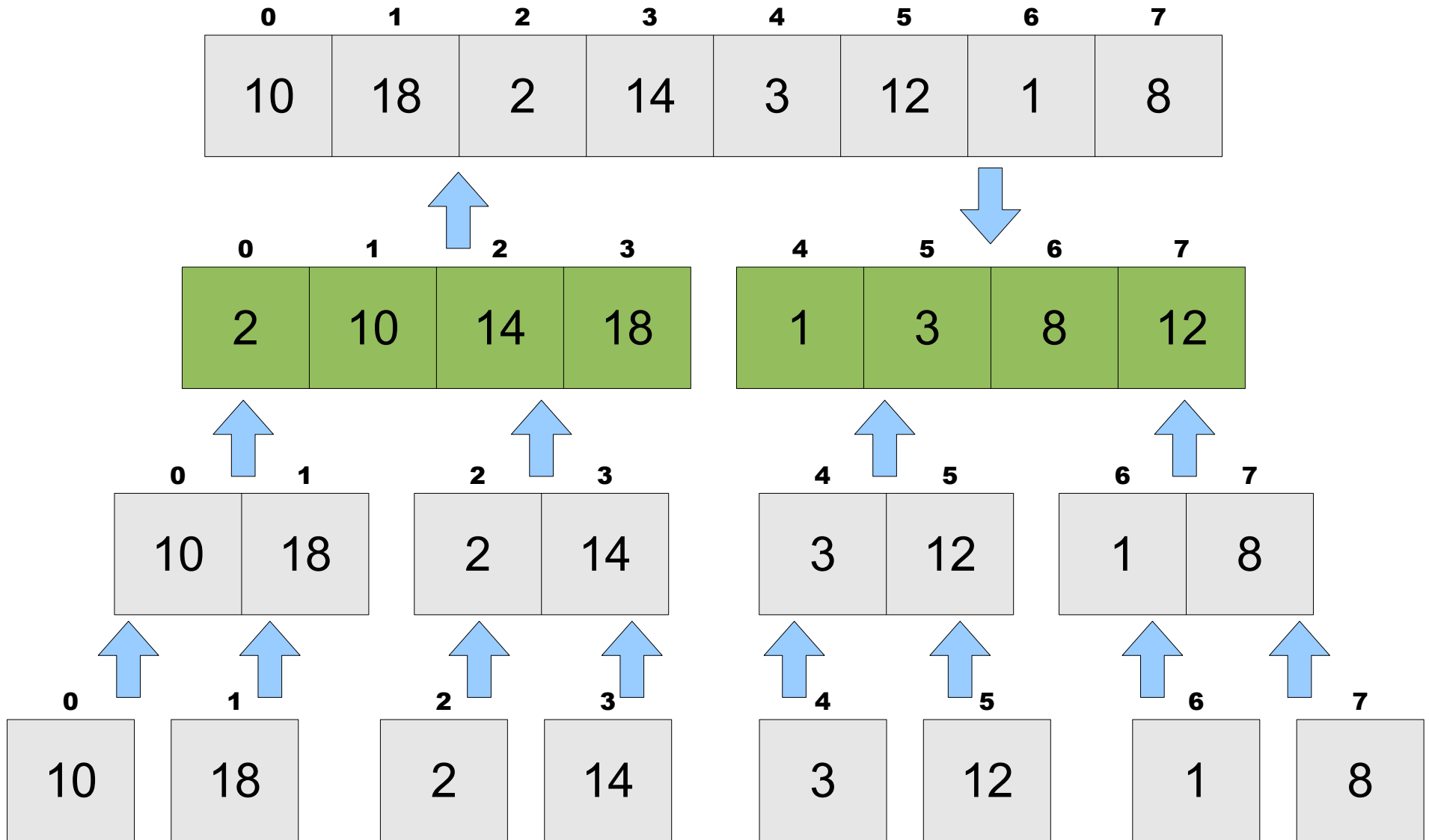
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

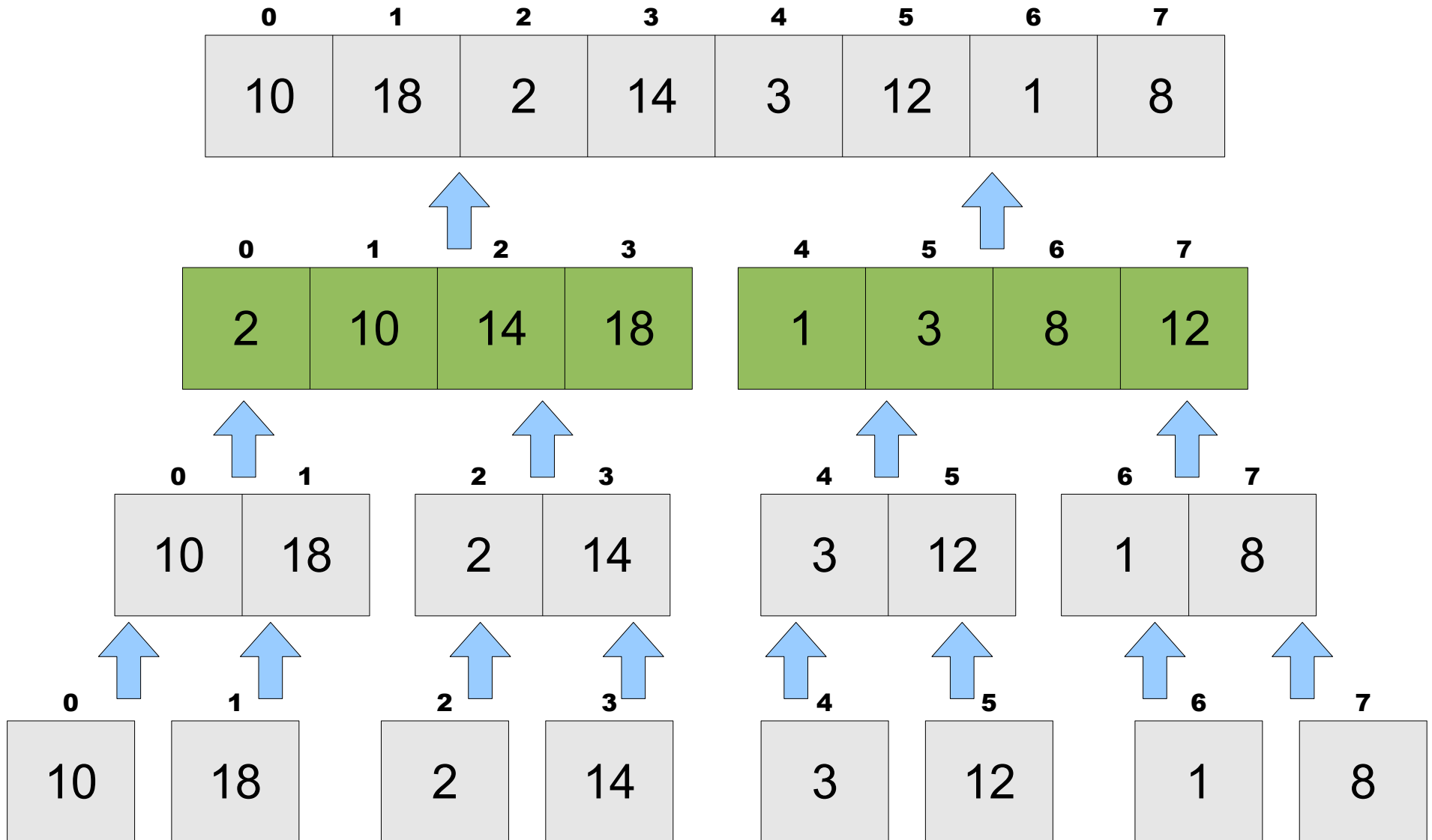
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

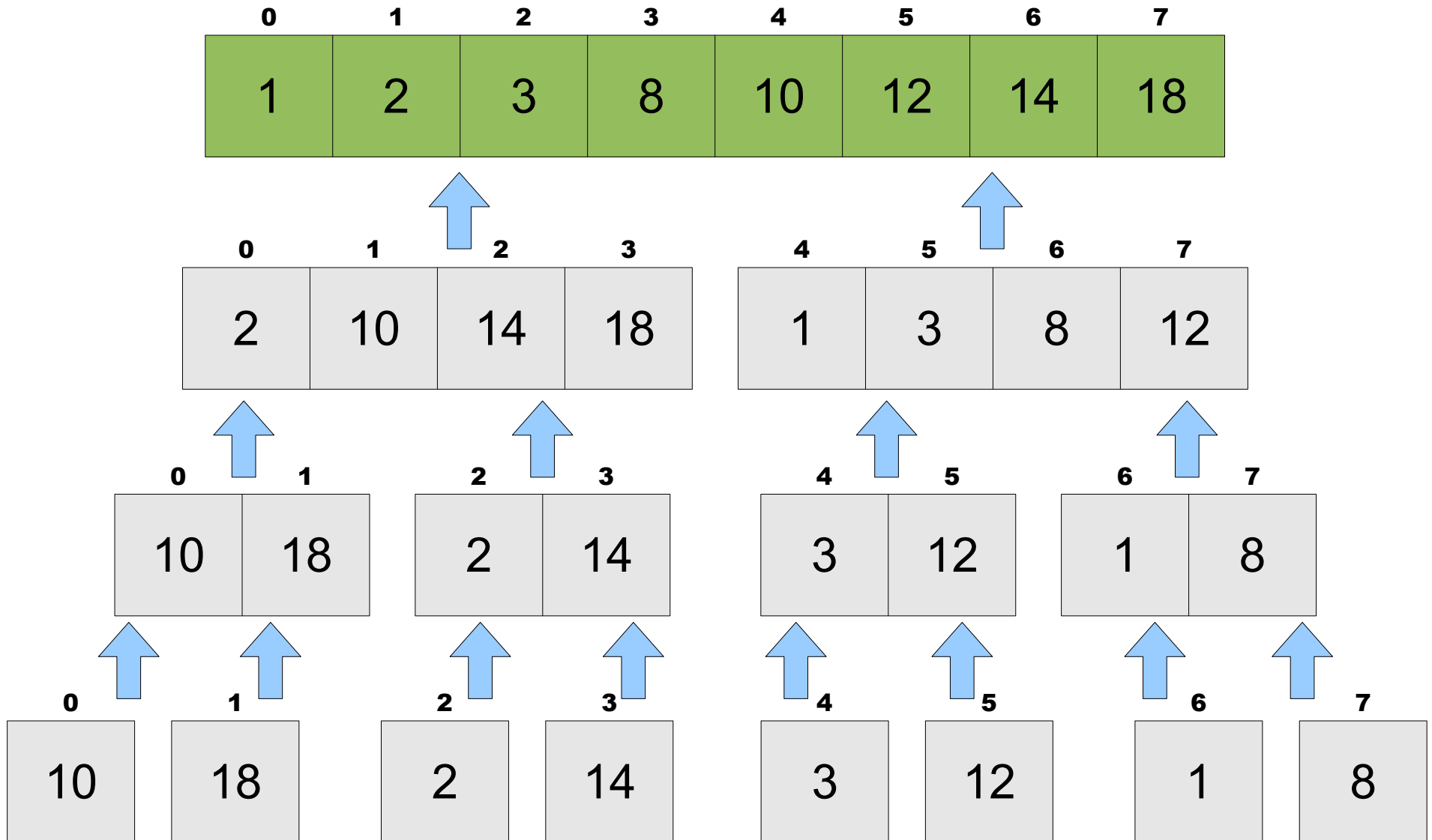
(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
1	2	3	8	10	12	14	18

**\* TADA! \***

# Merge Sort

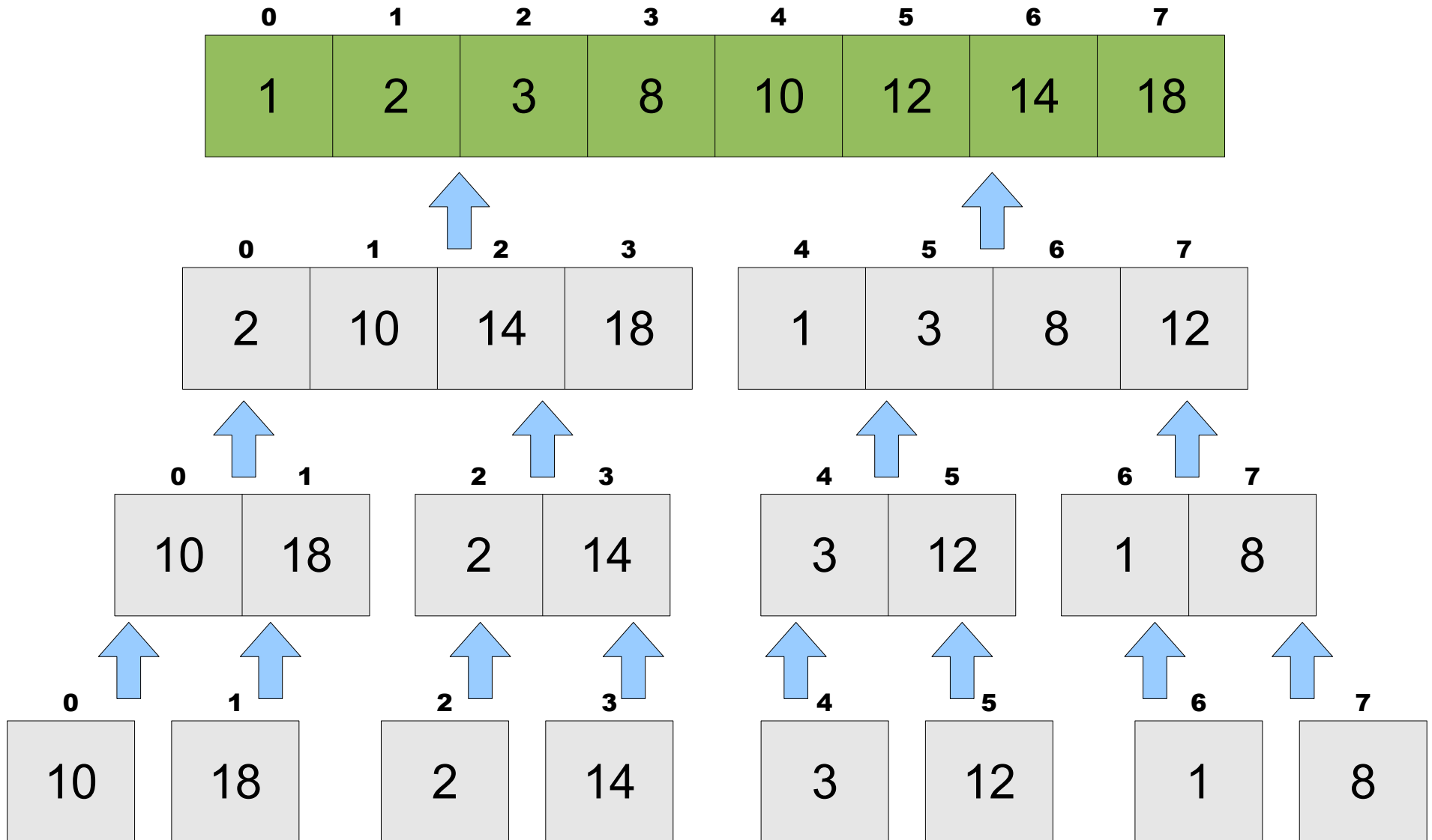
(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
1	2	3	8	10	12	14	18

1. What's the **worst-case** Big-Oh runtime?
2. What's the **best-case** Big-Oh runtime?
3. Trace the results of **each recursive sub-call** to Merge Sort.

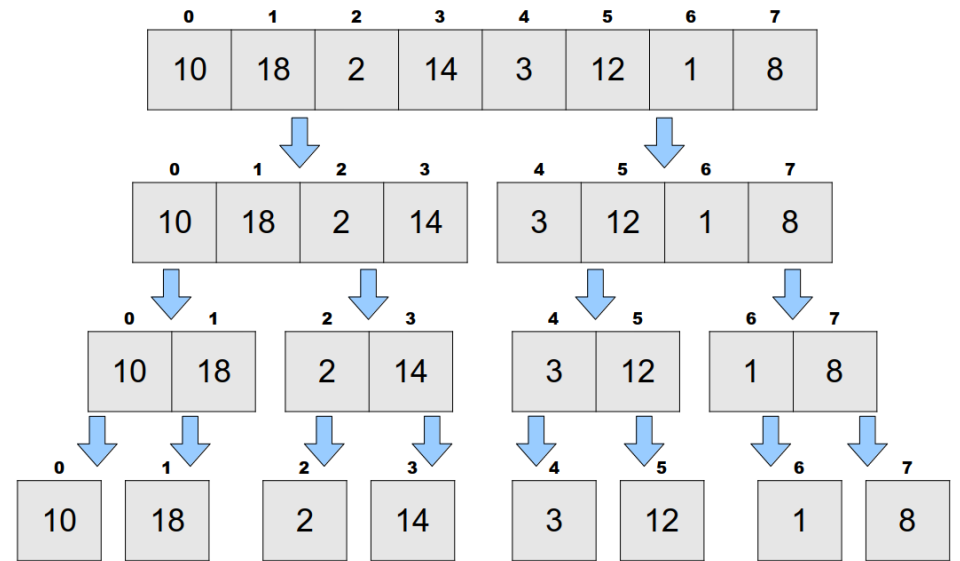
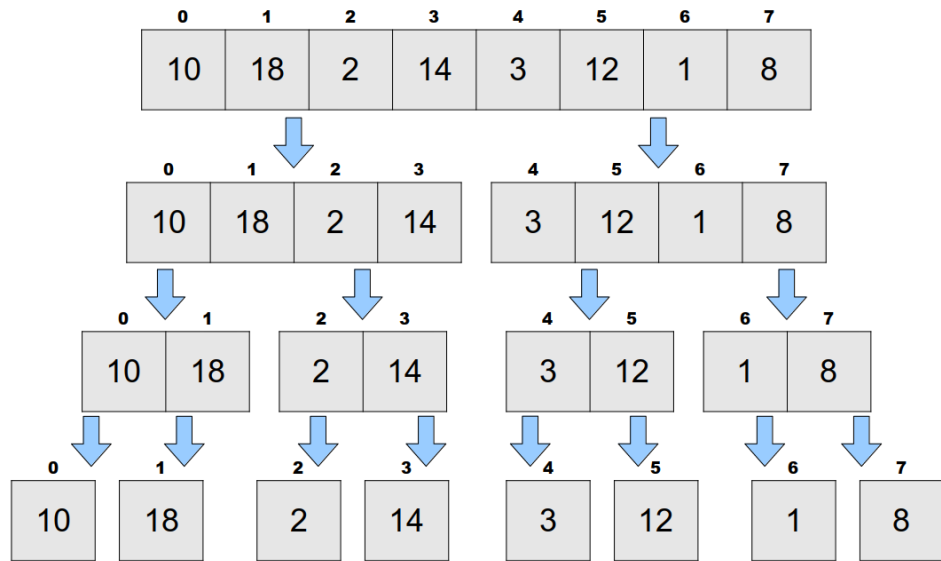
# Merge Sort

(break down the vector, then merge the pieces back together)



We've reached our **base cases**. A vector with one element is **sorted**.

10	18	2	14	3	12	1	8	10	18	2	14	3	12	1	8
----	----	---	----	---	----	---	---	----	----	---	----	---	----	---	---



# Merge Sort

(break down the vector, then merge the pieces back together)

0	1	2	3	4	5	6	7
1	2	3	8	10	12	14	18

Let's look at some code!