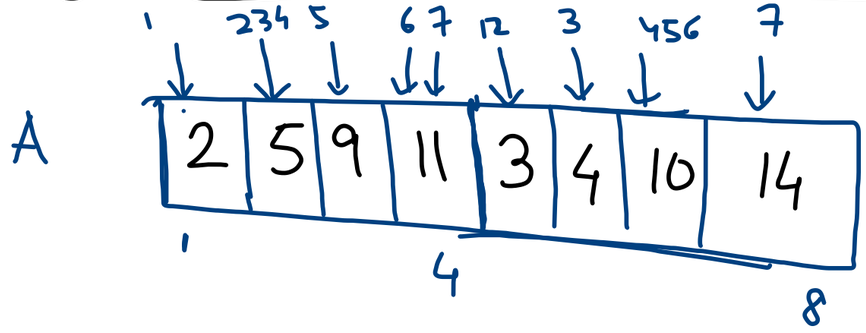
Merging Two Sorted Arrays

① Using O(n) Space



B

2	3	4	5	9	10	11	14

Copy (B, A)

② Using O(1) Space

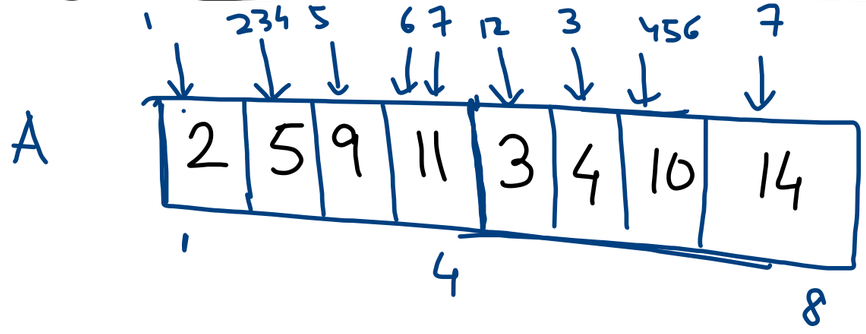
- Use insertion sort
- $O(n^2)$ ✓

Complexity of Merge Sort = $O(n^2 \log n)$

Can we do better?

Merging Two Sorted Arrays

① Using O(n) Space



B

2	3	4	5	9	10	11	14

Copy (B, A)

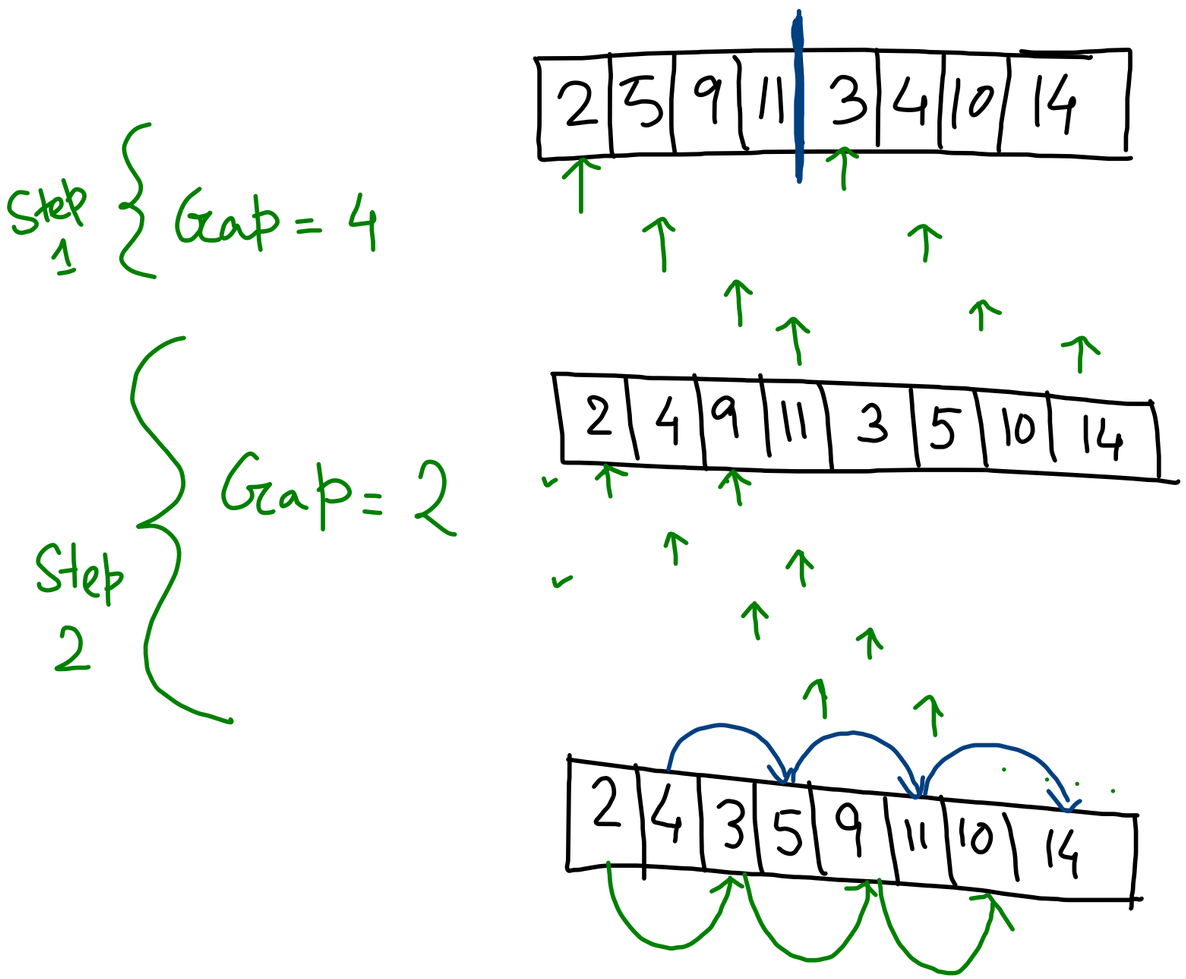
② Using O(1) Space

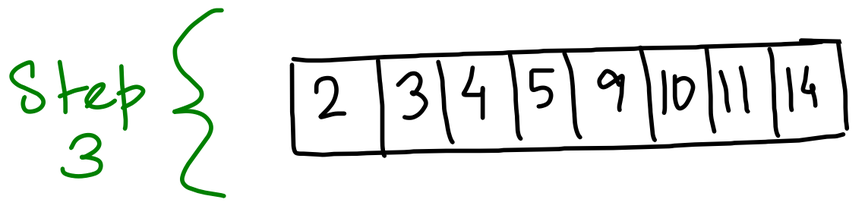
- Use insertion sort
- $O(n^2)$ ✓

Complexity of Merge Sort = $O(n^2 \log n)$

Can we do better?

Merging Two Sorted Arrays in O(1) Space Efficiently

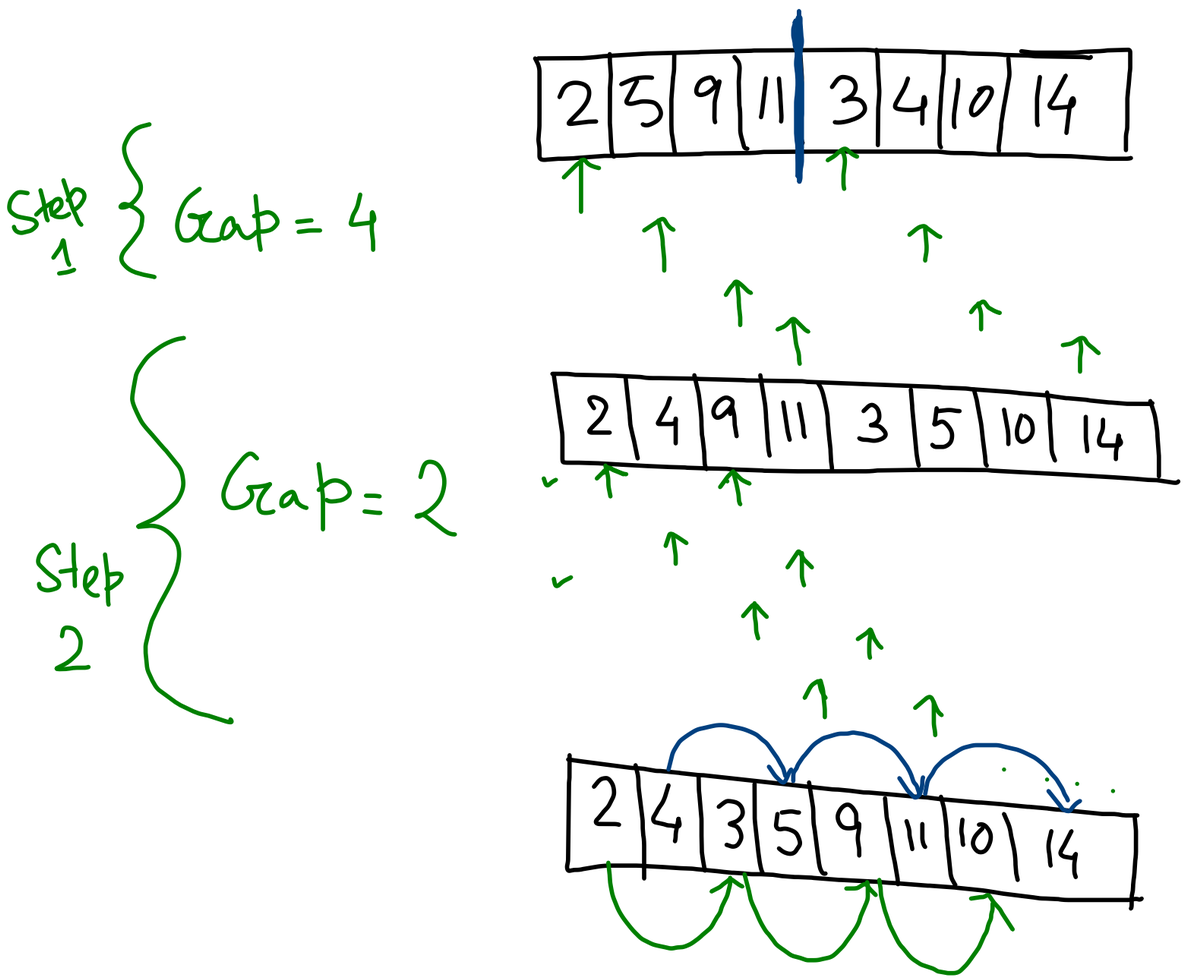




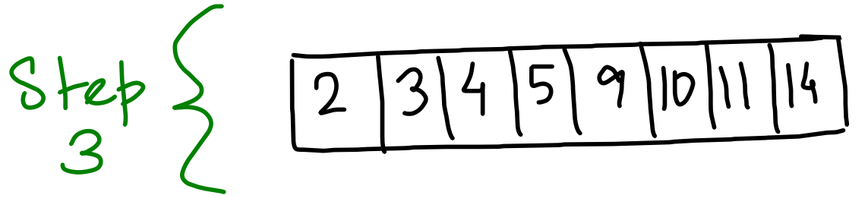
→ $O(\log n \cdot n)$

A → n
step1 Gap → n/2
step2 Gap → n/4
⋮
Gap → 1

Merging Two Sorted Arrays in O(1) Space Efficiently



⇐ Correctness



→ $O(\log n \cdot n)$

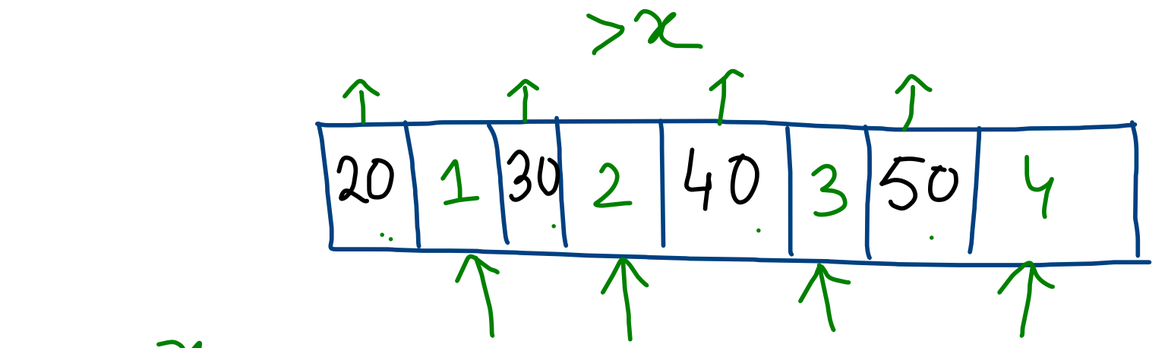
$A \to n$ step1 $Gap \to n/2$

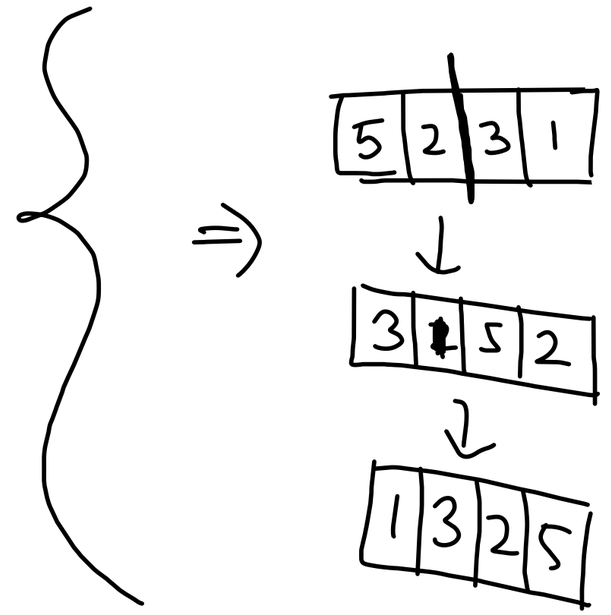
step2 $Gap \to n/4$

⋮

$Gap \to 1$

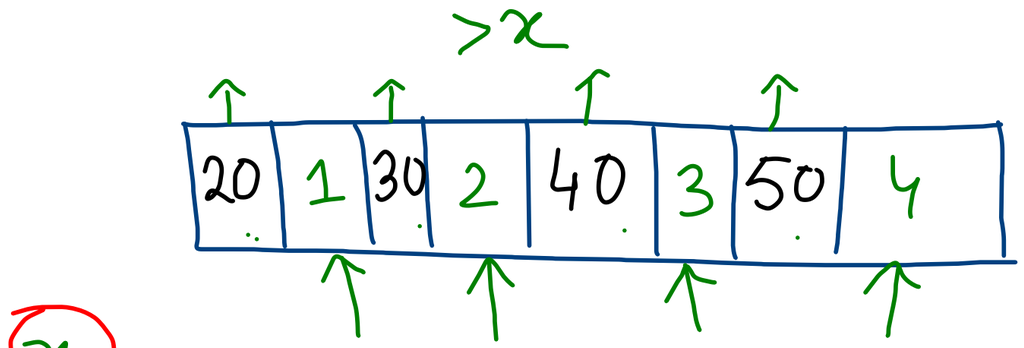
What Happens if the subarrays are not sorted?





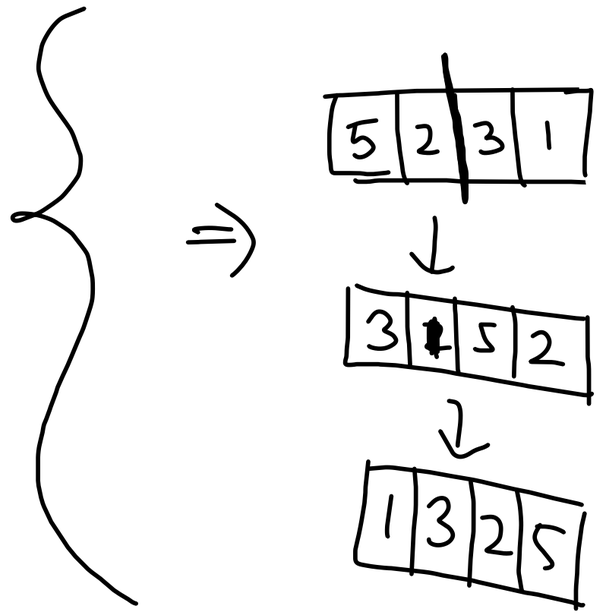
- The algorithm does not work.
- Is there any other cases where the algorithm does not work?

What Happens if the subarrays are not sorted?



- The algorithm doesnot work.

Why?



Merge Sort with O(1) space

$$T(n) = T(n/2) + T(n/2) + n \cdot \log n$$

$$T(2^k) = 2T(2^{k-1}) + k \cdot 2^k$$

$$f(k) = 2f(k-1) + k \cdot 2^k$$

$$= 2\left[2f(k-2) + (k-1)\cdot 2^{k-1}\right] + k \cdot 2^k$$

$$= \cdots = 2^k f(1) + 2^k\left[1+2+\cdots+k\right]$$

$$= 2^k f(1) + 2^k \cdot \frac{k(k+1)}{2}$$

$$T(n) = n(k) + n \cdot (\log n)^2$$

$$= O\left(n(\log n)^2\right)$$

- Insertion
- Selection
- Merge
- Quick

Ex: Which of these algorithms are Stable?

- In-place → using $O(1)$ additional space.
- Stable → Identical elements before & after sorting remains in same order.