Bubble sort

Bubble Sort
Idea

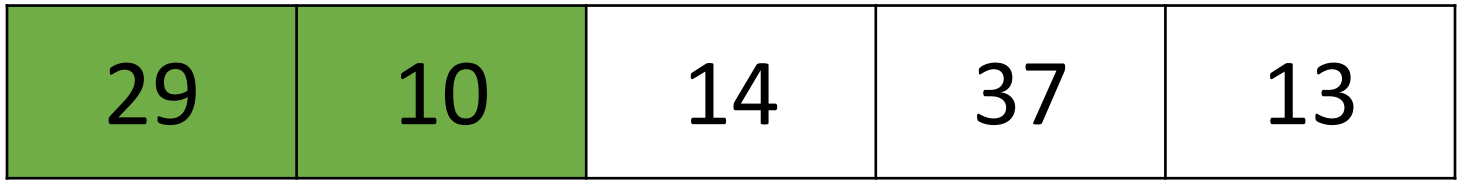
- Given an array of n items:
  - Compare pair of adjacent items.
  - Swap if the items are out of order.
  - Repeat until the end of array.
    - The largest item will be at the last position in the end of each iteration.
  - Reduce n by 1 and go to $step$ 1.

- Analogy:
  - Large item is like “bubble” that floats to the end of the array.

Bubble Sort Example

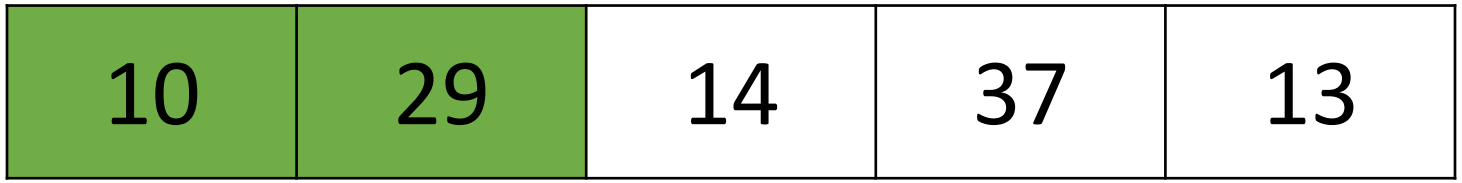
29	10	14	37	13

Bubble Sort Example



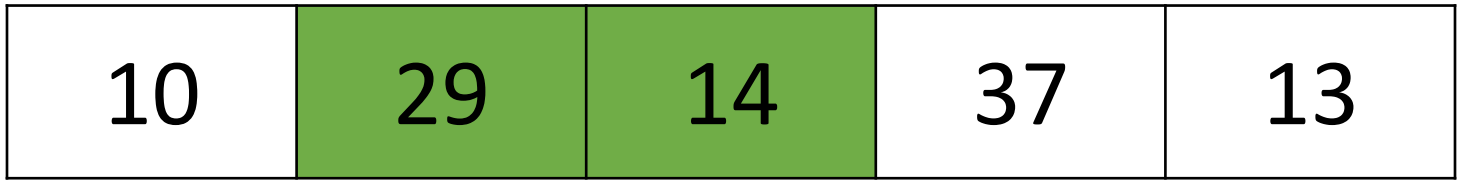
Comparing 29 and 10, and swapping them

Bubble Sort Example



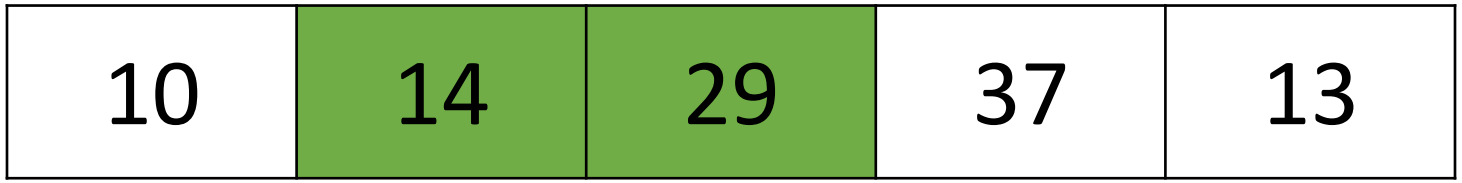
Comparing 29 and 10, and swapping them

Bubble Sort Example



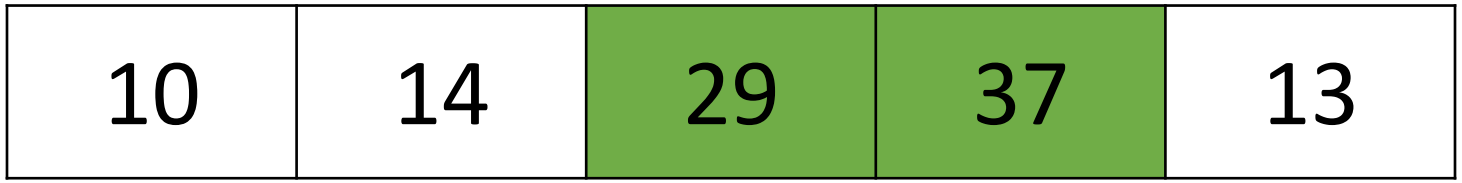
Comparing 29 and 14, and swapping them

Bubble Sort Example



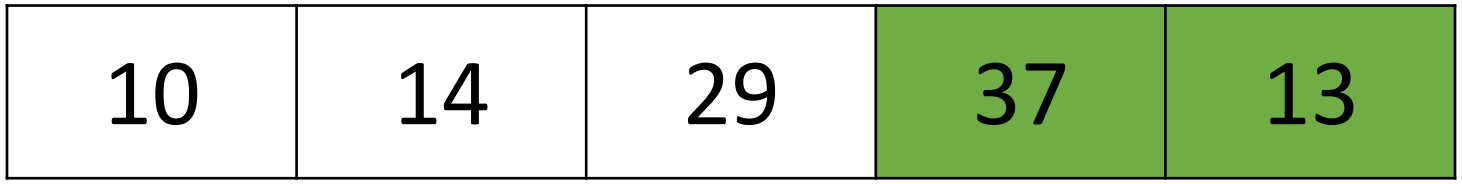
Comparing 29 and 14, and swapping them

Bubble Sort Example



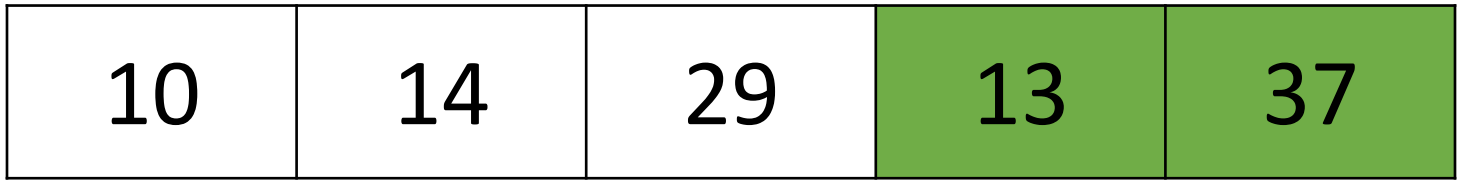
Comparing 29 and 37, and do nothing

Bubble Sort Example



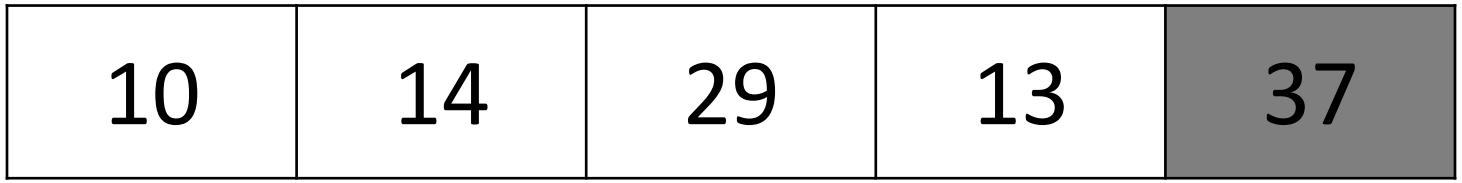
Comparing 37 and 13, and swapping them

Bubble Sort Example



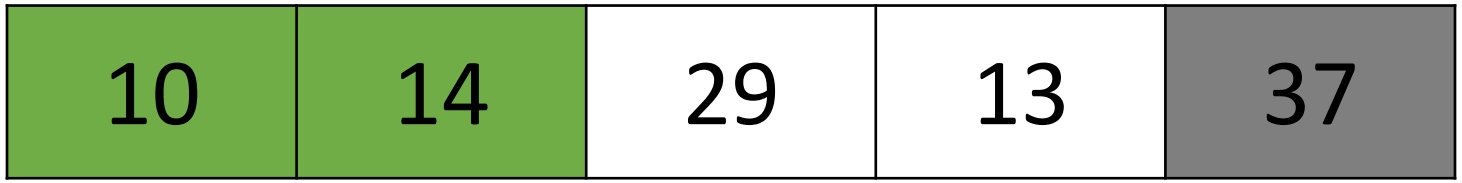
Comparing 37 and 13, and swapping them

Bubble Sort Example



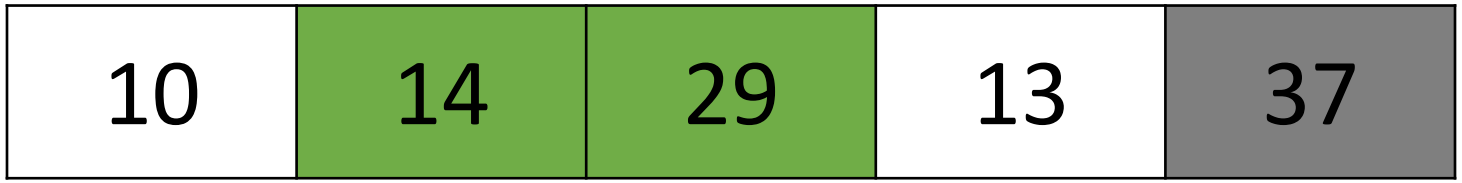
Largest element now in the end, performing again

Bubble Sort Example



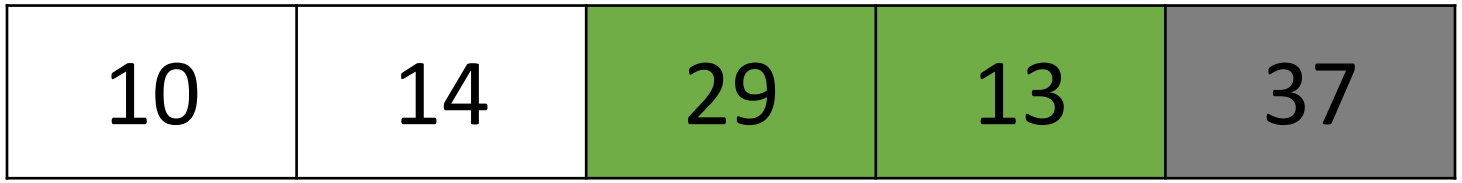
Comparing 10 and 14, and do nothing

Bubble Sort Example



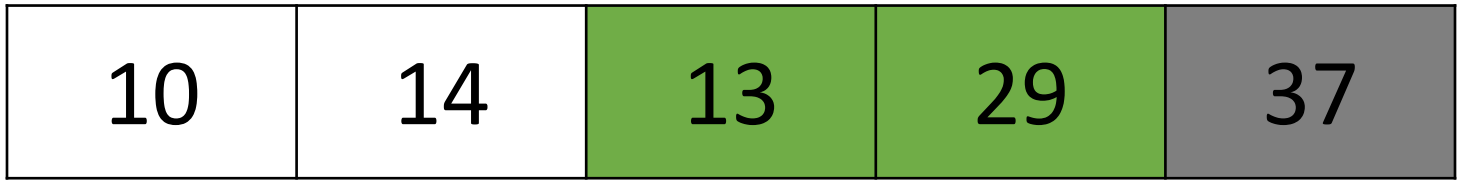
Comparing 14 and 29, and do nothing

Bubble Sort Example



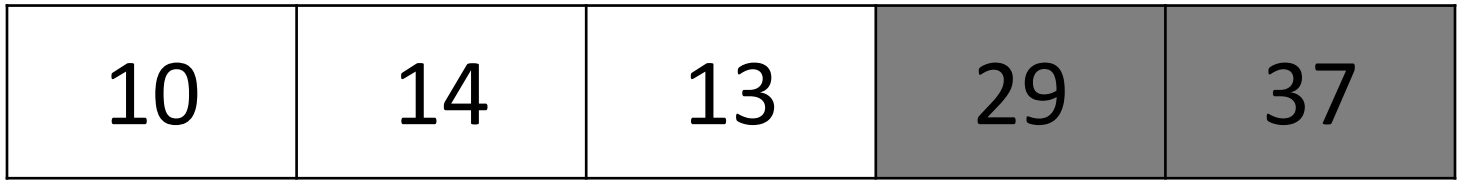
Comparing 29 and 13, and swapping them

Bubble Sort Example

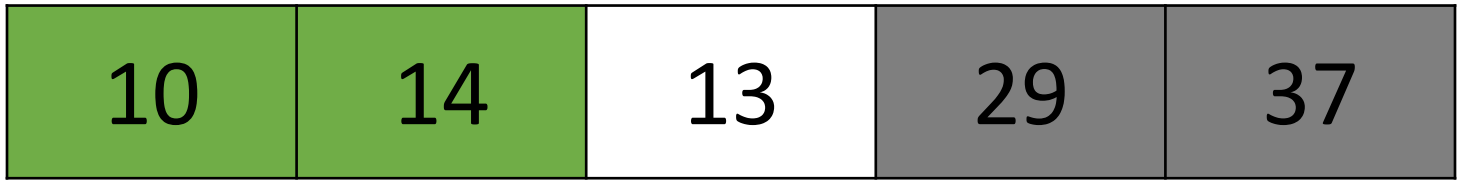


Comparing 29 and 13, and swapping them

Bubble Sort Example

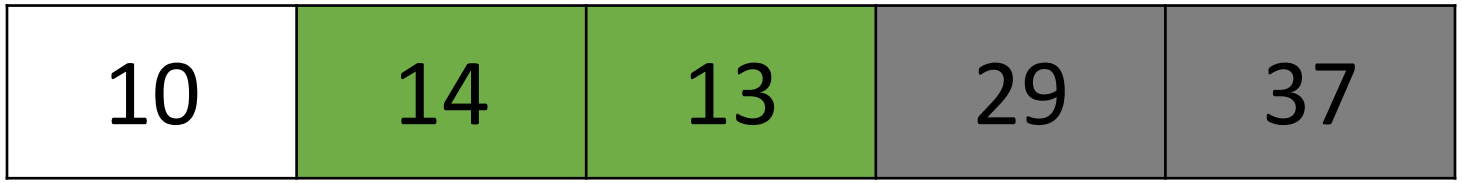


Bubble Sort Example



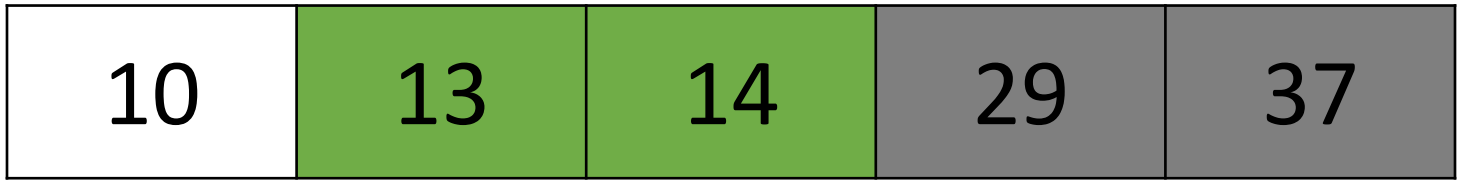
Comparing 10 and 14, and do nothing

Bubble Sort Example



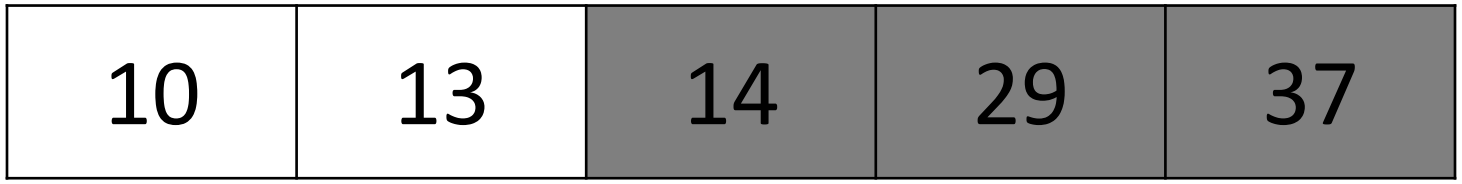
Comparing 14 and 13, and swapping them

Bubble Sort Example



Comparing 14 and 13, and swapping them

Bubble Sort Example



Bubble Sort Example

Comparing 10 and 13, and do nothing

Bubble Sort Example

Comparing 10 and 13, and do nothing

Bubble Sort Implementation

```
void bubble_sort(vector<int>& arr)
{
    for (int i = arr.size() - 1; i >= 1; i--)
        for (int j = 1; j <= i; j++)
            if (arr[j - 1] > arr[j])
                swap(arr[j], arr[j - 1]);
}
```

Bubble Sort Analysis

- Bubble sort is performing $n - i - 1$ operations for each.

- So time complexity of this sort is $(n - 1) + (n - 2) + \ldots + 1 = O(n^2)$.

Bubble Sort Optimization

- If we go through the inner loop with no swapping, then the array is already sorted and we can stop early.

```
void bubble_sort(vector<int>& arr)
{
    for (int i = arr.size() - 1; i >= 1; i--)
    {
        bool is_sorted = true;
        for (int j = 1; j <= i; j++)
            if (arr[j - 1] > arr[j])
            {
                swap(arr[j], arr[j - 1]);
                is_sorted = false;
            }
        if (is_sorted) return;
    }
}
```

Bubble Sort Optimization

- Worst-case:
  - Input is in descending order.
  - Running time remains the same: $O(n^2)$.

- Best-case:
  - Input is already in ascending order.
  - The algorithm returns after a single outer iteration.
  - Running time: $O(n)$.