Insertion sort

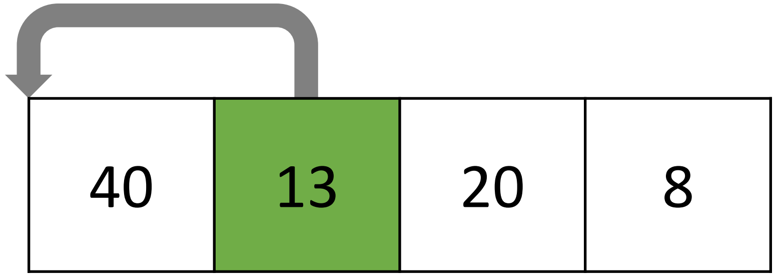
Insertion Sort

- Similar to how most people arrange a hand of cards:
  - Start with one card in your hand.
  - Pick the next card and insert it into its proper sorted order.
  - Repeat previous step for all cards.

Insertion Sort Example

40	13	20	8

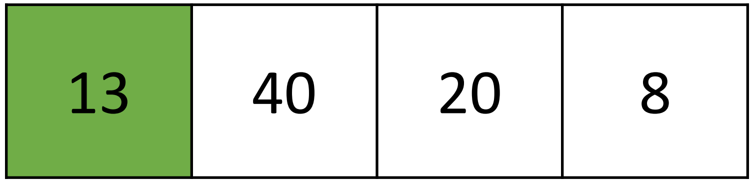
Insertion Sort Example



40	13	20	8

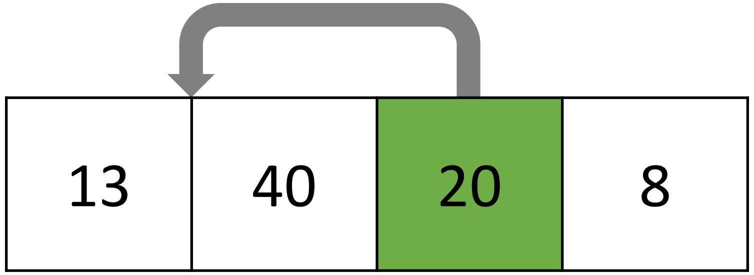
On first iteration we take second element and insert it into its proper sorted order

Insertion Sort Example



13	40	20	8

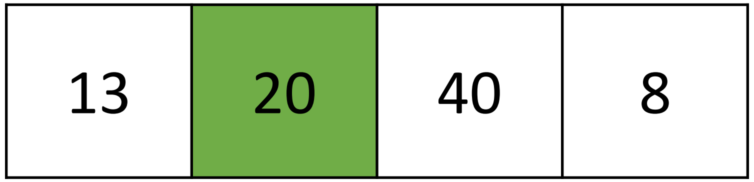
Insertion Sort Example



13	40	20	8

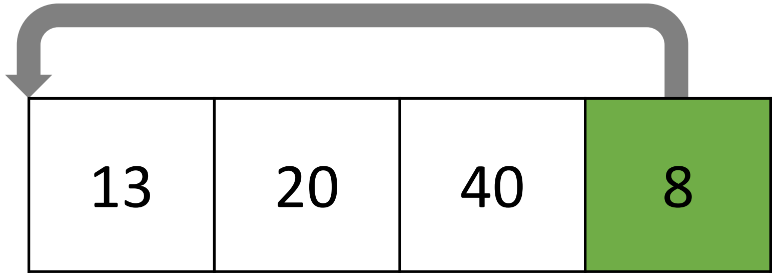
On second iteration we take third element and insert it into its proper sorted order

Insertion Sort Example



13	20	40	8

Insertion Sort Example



13	20	40	8

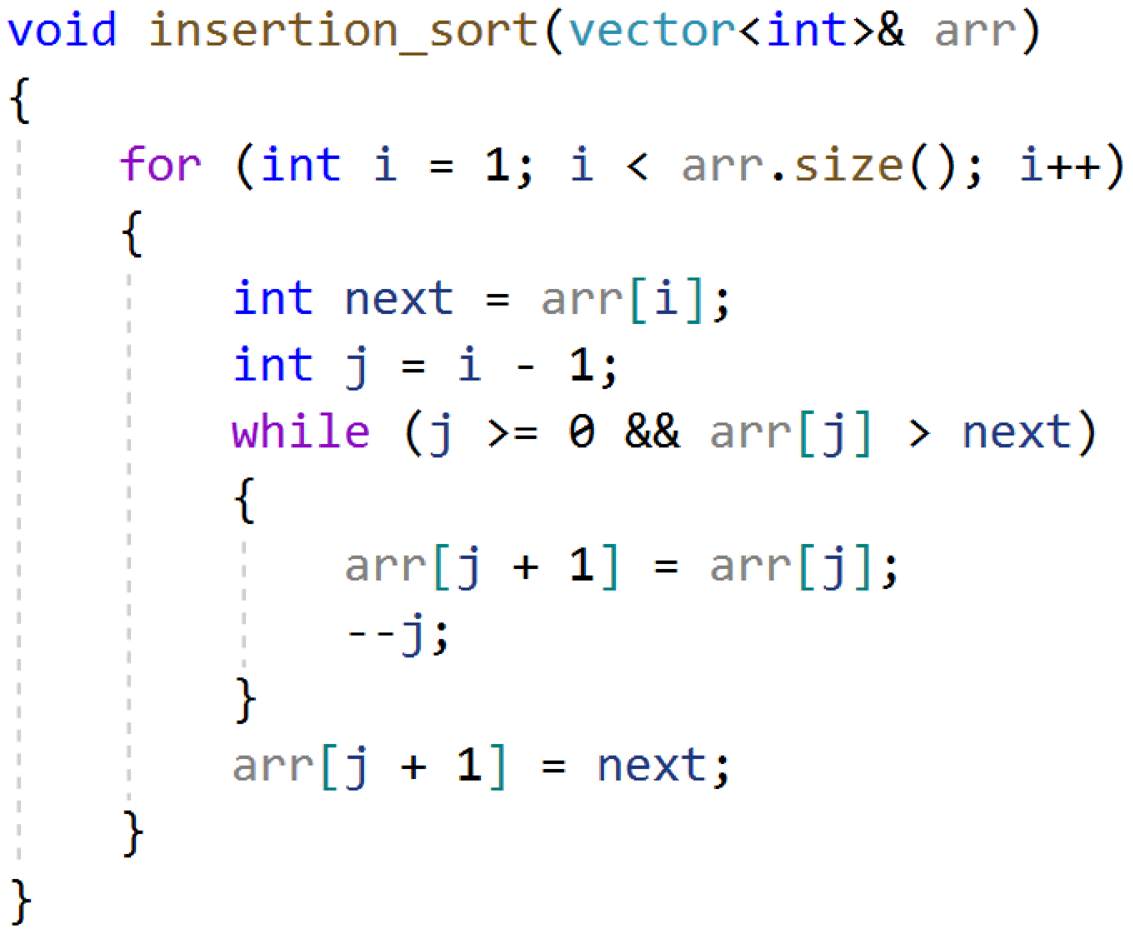
On last iteration we take the last element and insert it into its proper sorted order

Insertion Sort Example

8	13	20	40

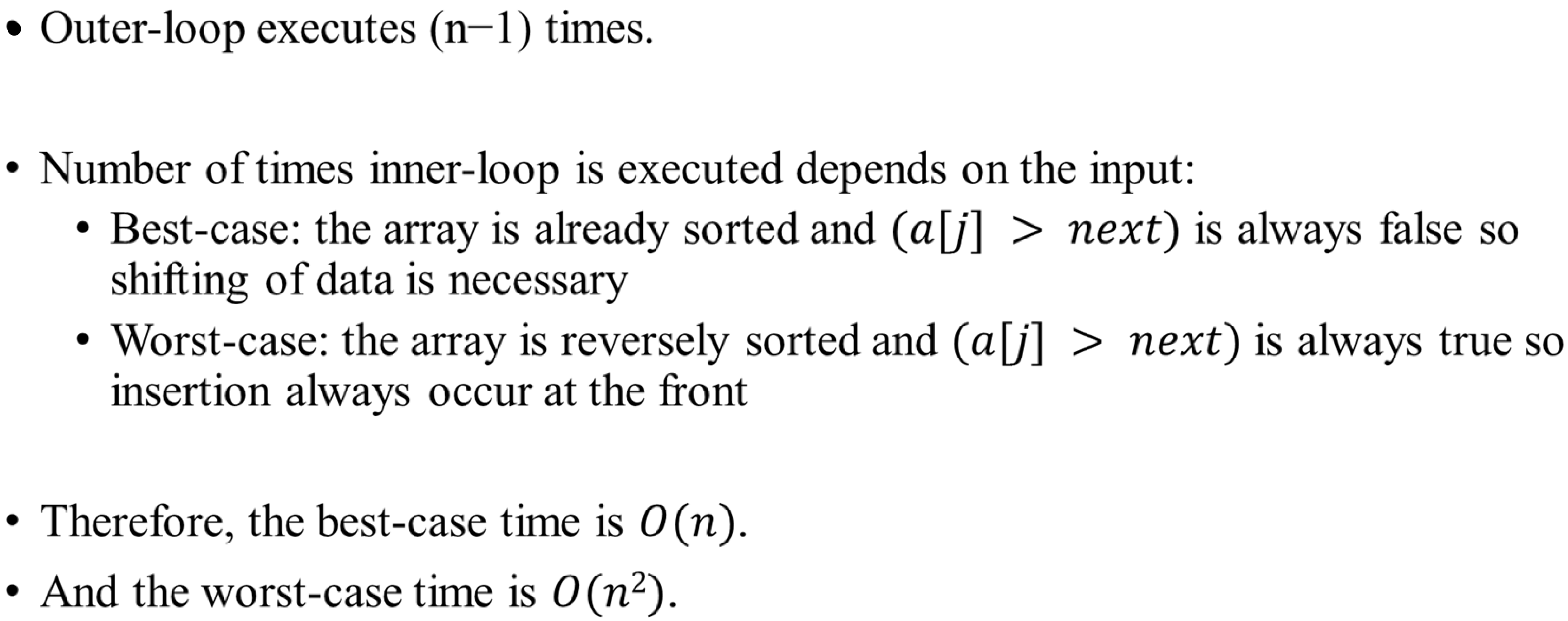
Array is sorted

Insertion Sort Implementation



```
void insertion_sort(vector<int>& arr)
{
    for (int i = 1; i < arr.size(); i++)
    {
        int next = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > next)
        {
            arr[j + 1] = arr[j];
            --j;
        }
        arr[j + 1] = next;
    }
}
```

Insertion Sort Analysis



- Outer-loop executes (n−1) times.

- Number of times inner-loop is executed depends on the input:
  - Best-case: the array is already sorted and $(a[j] > next)$ is always false so shifting of data is necessary
  - Worst-case: the array is reversely sorted and $(a[j] > next)$ is always true so insertion always occur at the front

- Therefore, the best-case time is $O(n)$.
- And the worst-case time is $O(n^2)$.