Data Structure and Algorithms

Selection Sort

- In selection sort, the smallest value among the unsorted elements of the array is selected in every pass and inserted to its appropriate position into the array.

- the array is divided into two parts, first is sorted part, and another one is the unsorted part. Initially, the sorted part of the array is empty, and unsorted part is the given array.

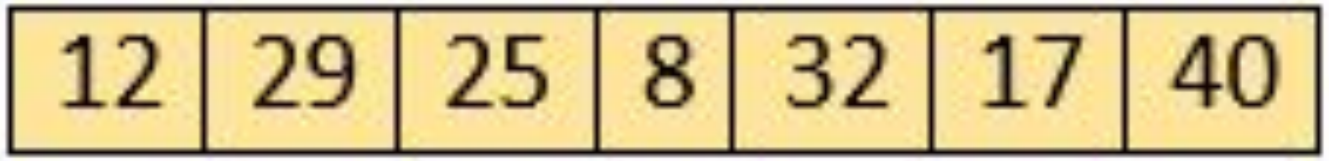
- Sorted part is placed at the left, while the unsorted part is placed at the right.

- In selection sort, the first smallest element is selected from the unsorted array and placed at the first position. After that second smallest element is selected and placed in the second position. The process continues until the array is entirely sorted.

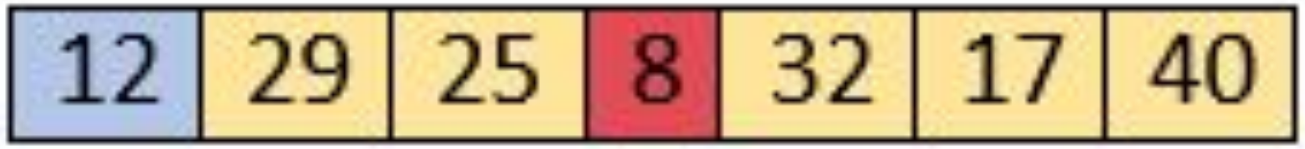
- The average and worst-case complexity of selection sort is O(n2), where n is the number of items. Due to this, it is not suitable for large data sets.

Working of Selection sort Algorithm

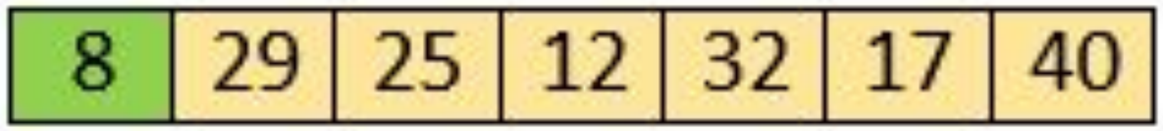
Let the elements of array are –



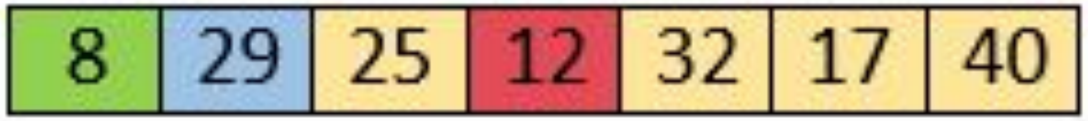
Now, for the first position in the sorted array, the entire array is to be scanned sequentially.
At present, **12** is stored at the first position, after searching the entire array, it is found that **8** is the smallest val



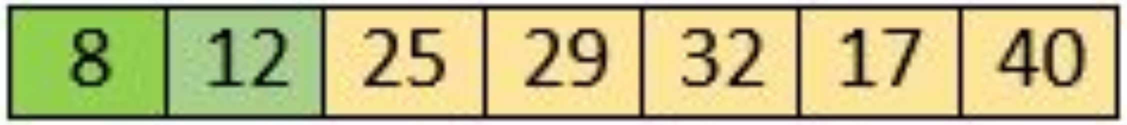
- So, swap 12 with 8. After the first iteration, 8 will appear at the first position in the sorted array.



Iteration 2

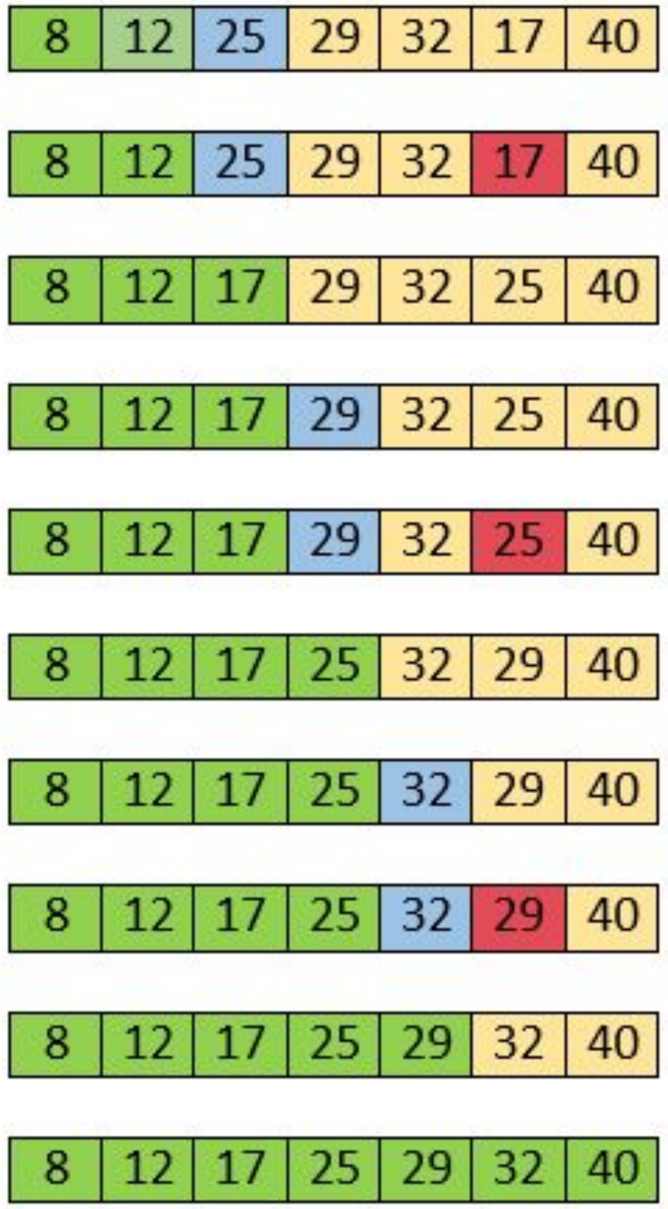


8	29	25	12	32	17	40



8	12	25	29	32	17	40

- The same process is applied to the rest of the array elements. Now, we are showing a pictorial representation of the entire sorting process.


8 12 25 29 32 17 40
8 12 25 29 32 17 40
8 12 17 29 32 25 40
8 12 17 29 32 25 40
8 12 17 29 32 25 40
8 12 17 25 32 29 40
8 12 17 25 32 29 40
8 12 17 25 32 29 40
8 12 17 25 29 32 40
8 12 17 25 29 32 40

Selection sort complexity

- Best Case

  $O(n^2)$

- Average Case

  $O(n^2)$

- Worst Case

  $O(n^2)$

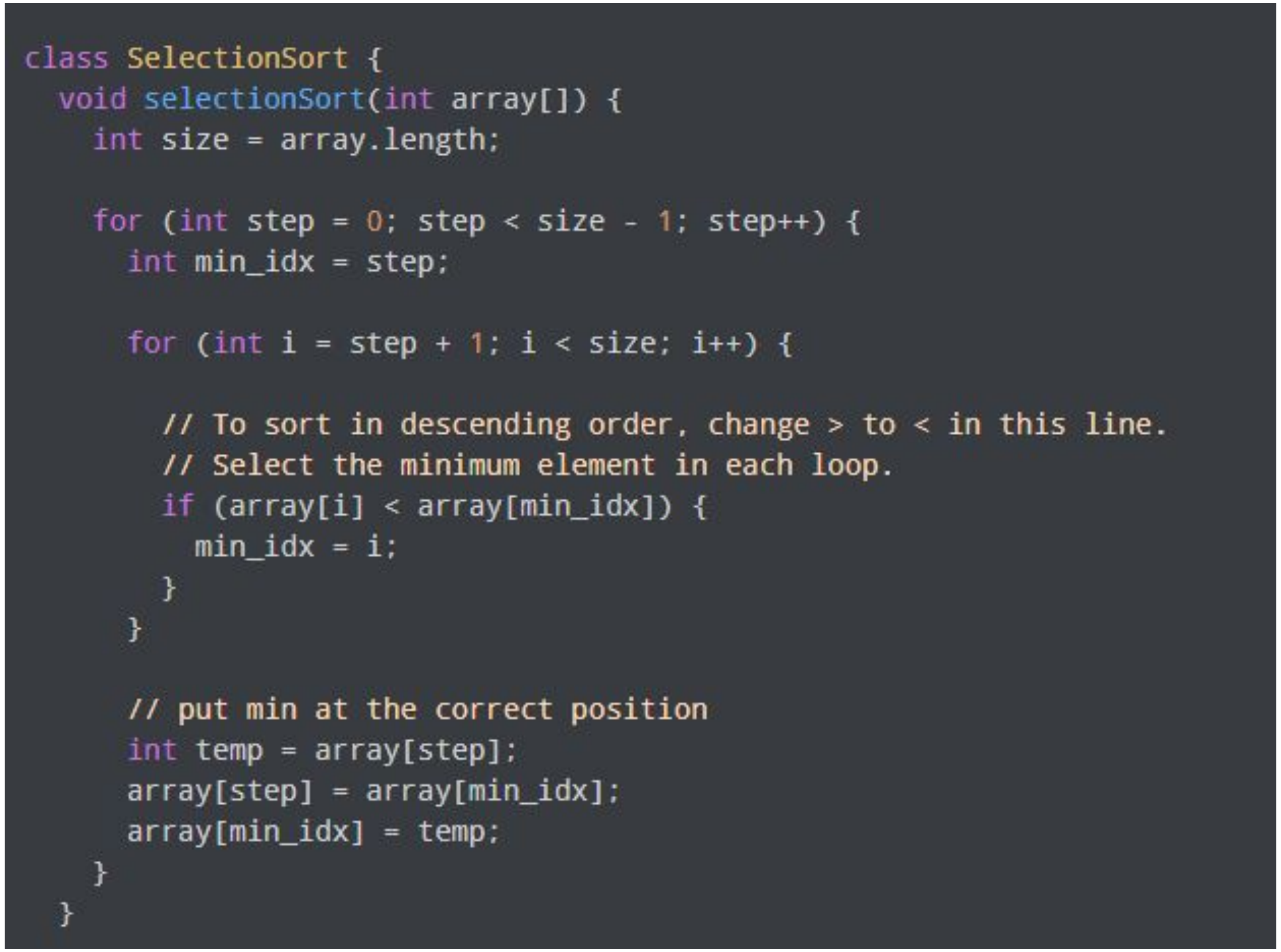
- **Best Case Complexity -** It occurs when there is no sorting required, i.e. the array is already sorted. The best-case time complexity of selection sort is **O(n^2)**.

- **Average Case Complexity -** It occurs when the array elements are in jumbled order that is not properly ascending and not properly descending. The average case time complexity of selection sort is **$O(n^2)$**.

- **Worst Case Complexity -** It occurs when the array elements are required to be sorted in reverse order. That means suppose you have to sort the array elements in ascending order, but its elements are in descending order. The worst-case time complexity of selection sort is **$O(n^2)$**.

Algorithm

```
selectionSort(array, size)
  repeat (size - 1) times
  set the first unsorted element as the minimum
  for each of the unsorted elements
    if element < currentMinimum
      set element as new minimum
  swap minimum with first unsorted position
end selectionSort
```



```
class SelectionSort {
  void selectionSort(int array[]) {
    int size = array.length;

    for (int step = 0; step < size - 1; step++) {
      int min_idx = step;

      for (int i = step + 1; i < size; i++) {

        // To sort in descending order, change > to < in this line.
        // Select the minimum element in each loop.
        if (array[i] < array[min_idx]) {
          min_idx = i;
        }
      }

      // put min at the correct position
      int temp = array[step];
      array[step] = array[min_idx];
      array[min_idx] = temp;
    }
  }
```

Selection sort Applications

The selection sort is used when

- a small list is to be sorted
- cost of swapping does not matter
- checking of all the elements is compulsory
- cost of writing to a memory matters like in flash memory (number of writes/

is **$O(n)$**

as compared to **$O(n^2)$** of bubble sort)