# Merge Overlapping Intervals

**Merge overlapping intervals**

## Merge Overlapping Intervals Space Optimized Approach

> *The above solution requires O(n) extra space for the stack. We can avoid the use of extra space by doing merge operations in place. Below are detailed steps.*

Follow the steps mentioned below to implement the approach:

- Sort all intervals in increasing order of start time.
- Traverse sorted intervals starting from the first interval,
- Do the following for every interval.
    - If the current interval is not the first interval and it overlaps with the previous interval,
      then merge it with the previous interval. Keep doing it while the interval overlaps with the previous one.

    - Otherwise, Add the current interval to the output list of intervals.

Below is the implementation of the above approach:

C++

```cpp
// C++ program to merge overlapping Intervals in
// O(n Log n) time and O(1) extra space.
#include <bits/stdc++.h>
using namespace std;

// An Interval
struct Interval {
    int s, e;
};

// Function used in sort
bool mycomp(Interval a, Interval b) { return a.s < b.s; }

void mergeIntervals(Interval arr[], int n)
{
    // Sort Intervals in increasing order of
    // start time
    sort(arr, arr + n, mycomp);

    int index = 0; // Stores index of last element
    // in output array (modified arr[])

    // Traverse all input Intervals
    for (int i = 1; i < n; i++) {
        // If this is not first Interval and overlaps
        // with the previous one
        if (arr[index].e >= arr[i].s) {
            // Merge previous and current Intervals
```

```
            arr[index].e = max(arr[index].e, arr[i].e);
        }
        else {
            arr[index] = arr[i];
        }
    }

    // Now arr[0..index-1] stores the merged Intervals
    cout << "\n The Merged Intervals are: ";
    for (int i = 0; i <= index; i++)
        cout << "[" << arr[i].s << ", " << arr[i].e << "] ";
}

// Driver program
int main()
{
    Interval arr[]
        = { { 6, 8 }, { 1, 9 }, { 2, 4 }, { 4, 7 } };
    int n = sizeof(arr) / sizeof(arr[0]);
    mergeIntervals(arr, n);
    return 0;
}

// This code is contributed by Aditya Kumar (adityakumar129)
```

## Output

```
The Merged Intervals are: [1, 9]
```

**Time Complexity:** O(N*log(N))
**Auxiliary Space Complexity:** O(1)

Dash
All
Articles
Videos
Problems
Quiz

Mark as Read

Report An Issue

If you are facing any issue on this page. Please let us know.