Merge Intervals (medium)

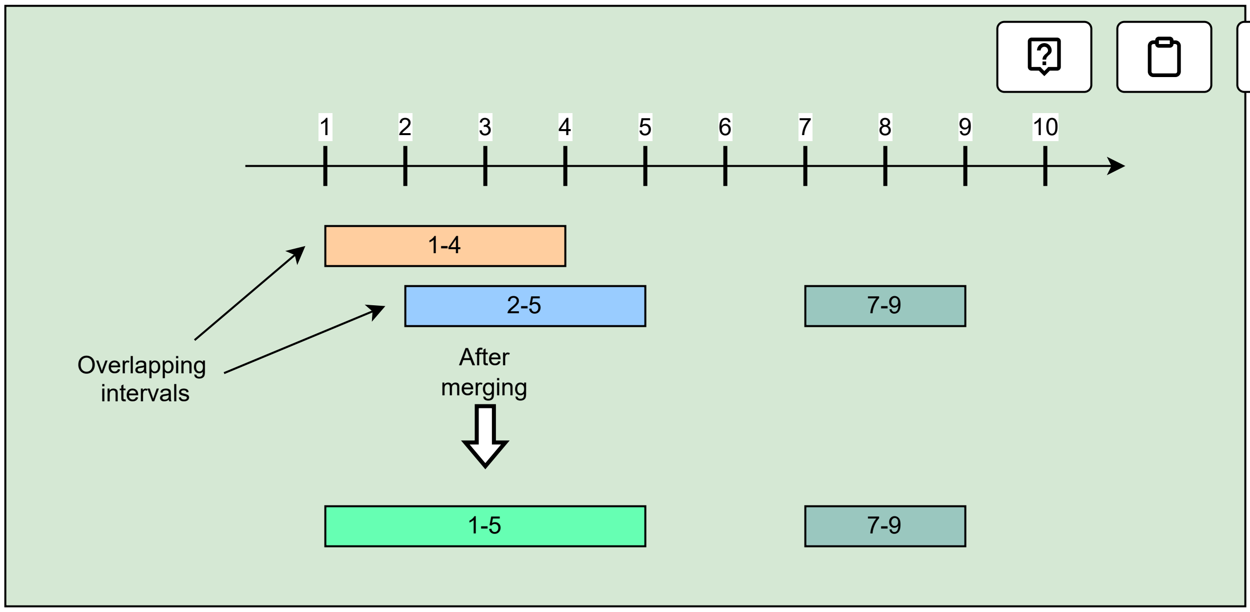
We'll cover the following

Problem Statement

Given a list of intervals, **merge all the overlapping intervals** to produce a list that has only mutually exclusive intervals.

Example 1:

```
Intervals: [[1,4], [2,5], [7,9]]
Output: [[1,5], [7,9]]
Explanation: Since the first two intervals [1,4] and [2,5] overlap, we m
erged them into
one [1,5].
```



Example 2:

```
Intervals: [[6,7], [2,4], [5,9]]
Output: [[2,4], [5,9]]
Explanation: Since the intervals [6,7] and [5,9] overlap, we merged the
m into one [5,9].
```

Example 3:

```
Intervals: [[1,4], [2,6], [3,5]]
Output: [[1,6]]
Explanation: Since all the given intervals overlap, we merged them int
o one.
```

Try it yourself

Try solving this question here:

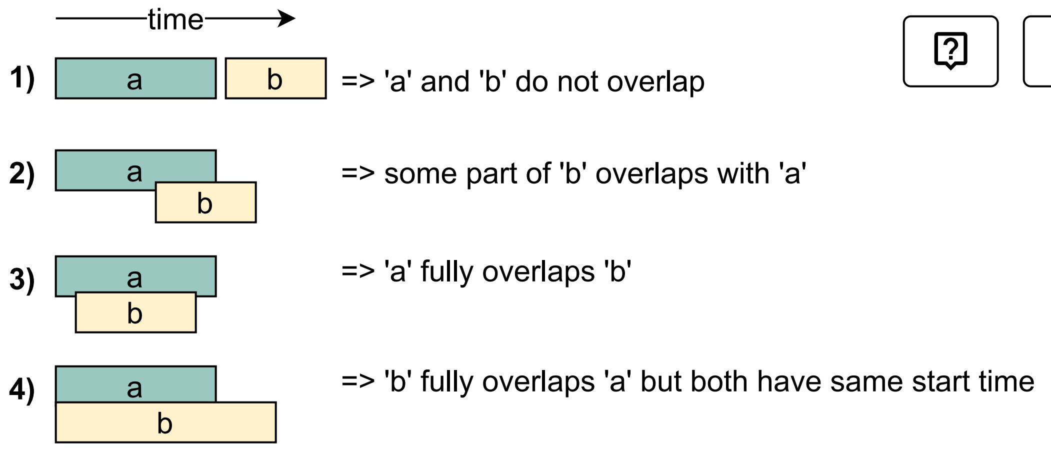
C++

```
using namespace std;

#include <algorithm>
```

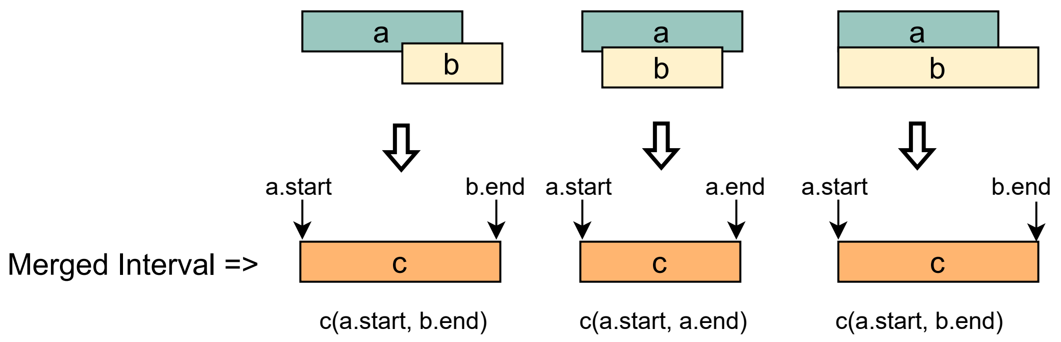
```
#include <iostream>
#include <vector>

class Interval {
 public:
  int start = 0;
  int end = 0;

  Interval(int start, int end) {
    this->start = start;
    this->end = end;
  }
};

class MergeIntervals {
 public:
  static vector<Interval> merge(vector<Interval> &intervals) {
    vector<Interval> mergedIntervals;
    // TODO: Write your code here
    return mergedIntervals;
  }
};

int main(int argc, char *argv[]) {
  vector<Interval> input = {{1, 3}, {2, 5}, {7, 9}};
  cout << "Merged intervals: ";
  for (auto interval : MergeIntervals::merge(input)) {
    cout << "[" << interval.start << "," << interval.end << "] ";
```

Solution

Let’s take the example of two intervals (‘a’ and ‘b’) such that `a.start <= b.start`. There are four possible scenarios:


time
1)
a
b
=> 'a' and 'b' do not overlap
2)
a
b
=> some part of 'b' overlaps with 'a'
3)
a
b
=> 'a' fully overlaps 'b'
4)
a
b
=> 'b' fully overlaps 'a' but both have same start time

Our goal is to merge the intervals whenever they overlap. For the above-mentioned three overlapping scenarios (2, 3, and 4), this is how we will merge them:



The diagram above clearly shows a merging approach. Our algorithm will look like this:

1. Sort the intervals on the start time to ensure `a.start <= b.start`
2. If ‘a’ overlaps ‘b’ (i.e. `b.start <= a.end`), we need to merge them into a new interval ‘c’ such that:

```
c.start = a.start
c.end = max(a.end, b.end)
```

3. We will keep repeating the above two steps to merge ‘c’ with the next interval if it overlaps with ‘c’.

Code

Here is what our algorithm will look like:

C++

```
using namespace std;

#include <algorithm>
#include <iostream>
#include <vector>

class Interval {
 public:
  int start = 0;
  int end = 0;

  Interval(int start, int end) {
    this->start = start;
    this->end = end;
  }
};

class MergeIntervals {
 public:
  static vector<Interval> merge(vector<Interval> &intervals) {
    if (intervals.size() < 2) {
      return intervals;
    }

    // sort the intervals by start time
    sort(intervals.begin(), intervals.end(),
         [](const Interval &x, const Interval &y) { return x.start < y.start; });

    vector<Interval> mergedIntervals;

    vector<Interval>::const_iterator intervalItr = intervals.begin();
```

Time complexity#

The time complexity of the above algorithm is $O(N * logN)$, where 'N' is the total number of intervals. We are iterating the intervals only once which will take $O(N)$, in the beginning though, since we need to sc e i a our algorithm will take $O(N * logN)$.

Space complexity#

The space complexity of the above algorithm will be $O(N)$ as we need to return a list containing all the merged intervals. We will also need $O(N)$ space for sorting. For Java, depending on its version, `Collections.sort()` either uses Merge sort or Timsort, and both these algorithms need $O(N)$ space. Overall, our algorithm has a space complexity of $O(N)$.

Similar Problems#

Problem 1: Given a set of intervals, find out if any two intervals overlap.

Example:

```
Intervals: [[1,4], [2,5], [7,9]]
Output: true
Explanation: Intervals [1,4] and [2,5] overlap
```

Solution: We can follow the same approach as discussed above to find if any two intervals overlap.

Back

Introduction

Next

Insert Interval (medium)

Completed

Report an Iss