Merge Intervals (medium)

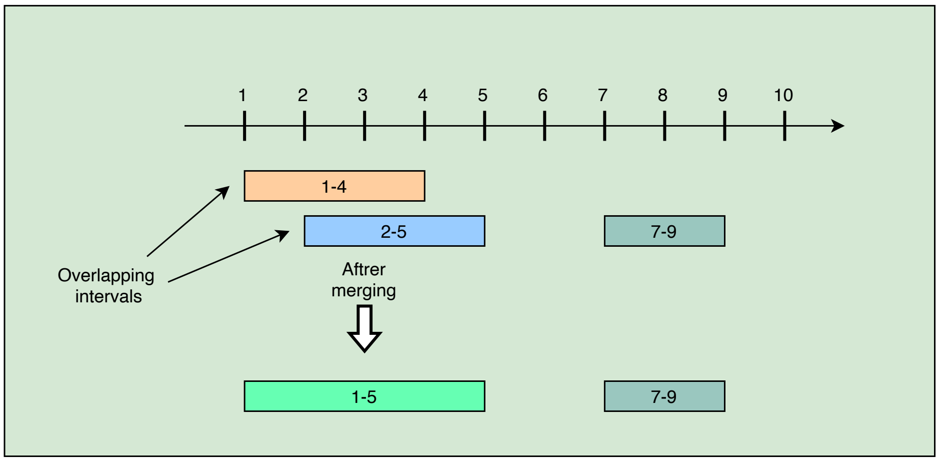
We'll cover the following

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 merged the
m into
one [1,5].
```



Example 2:

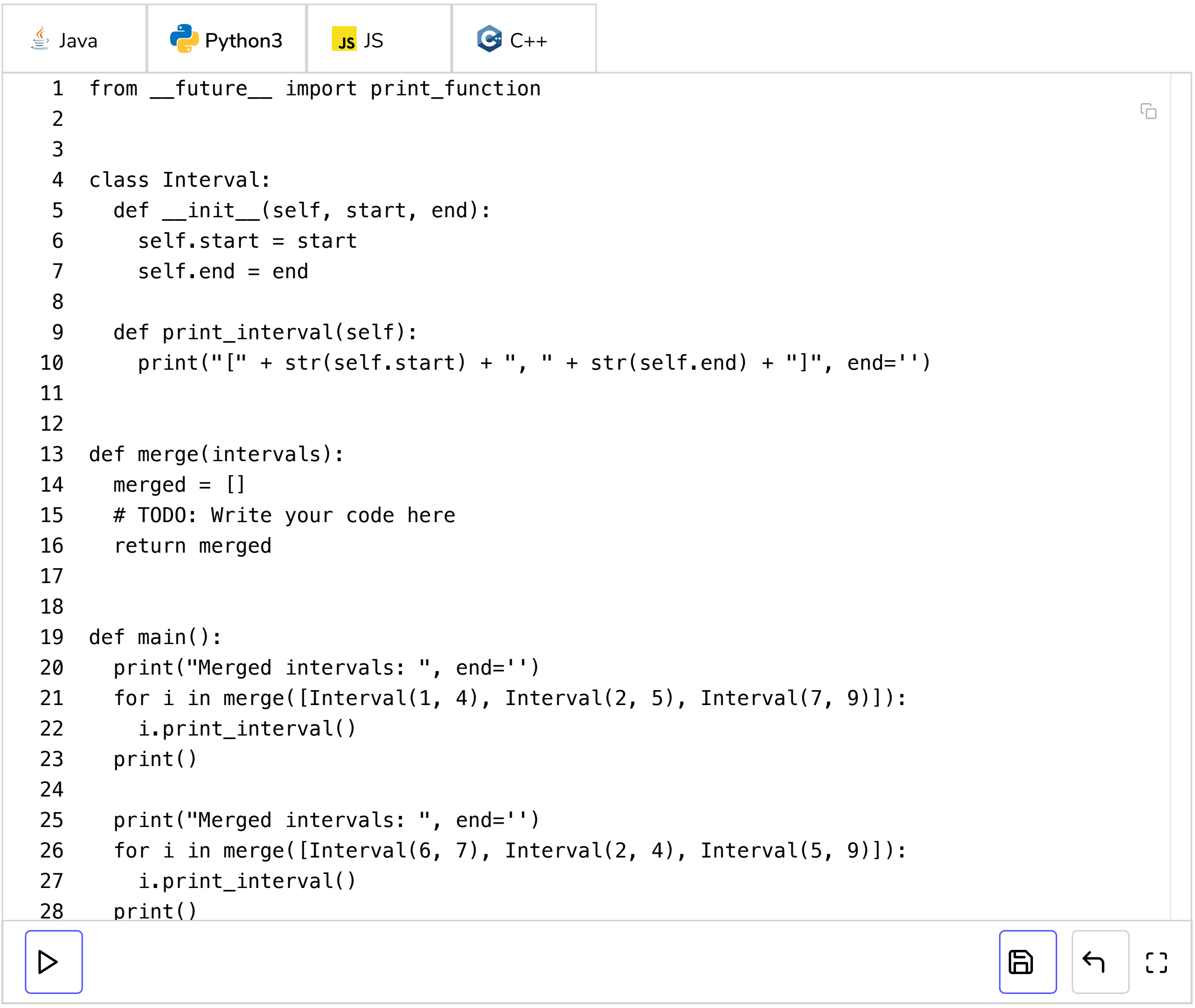
```
Intervals: [[6,7], [2,4], [5,9]]
Output: [[2,4], [5,9]]
Explanation: Since the intervals [6,7] and [5,9] overlap, we merged them into one
[5,9].
```

Example 3:

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

Try it yourself

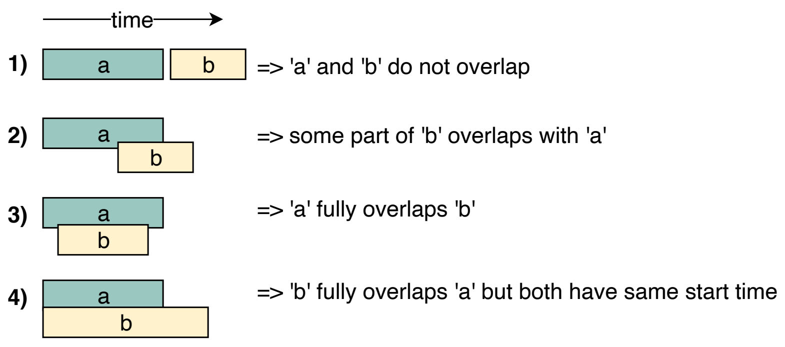
Try solving this question here:



```python
from __future__ import print_function


class Interval:
  def __init__(self, start, end):
    self.start = start
    self.end = end

  def print_interval(self):
    print("[" + str(self.start) + ", " + str(self.end) + "]", end='')


def merge(intervals):
  merged = []
  # TODO: Write your code here
  return merged


def main():
  print("Merged intervals: ", end='')
  for i in merge([Interval(1, 4), Interval(2, 5), Interval(7, 9)]):
    i.print_interval()
  print()

  print("Merged intervals: ", end='')
  for i in merge([Interval(6, 7), Interval(2, 4), Interval(5, 9)]):
    i.print_interval()
  print()
```

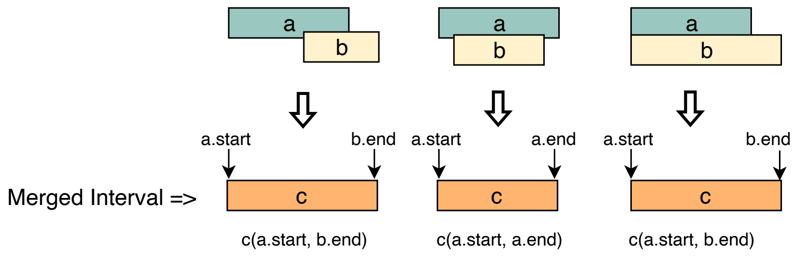
Solution

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:

Java | Python3 | C++ | JS

```python
from __future__ import print_function


class Interval:
  def __init__(self, start, end):
    self.start = start
    self.end = end

  def print_interval(self):
    print("[" + str(self.start) + ", " + str(self.end) + "]", end='')


def merge(intervals):
  if len(intervals) < 2:
    return intervals

  # sort the intervals on the start time
  intervals.sort(key=lambda x: x.start)

  mergedIntervals = []
  start = intervals[0].start
  end = intervals[0].end
  for i in range(1, len(intervals)):
    interval = intervals[i]
    if interval.start <= end:  # overlapping intervals, adjust the 'end'
      end = max(interval.end, end)
    else:  # non-overlapping interval, add the previous internval and reset
      mergedIntervals.append(Interval(start, end))
```

educative

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 sort the intervals, 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, `Collection.sort()` either uses Merge sort (https://en.wikipedia.org/wiki/Merge_sort) or Timsort (https://en.wikipedia.org/wiki/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)

Mark as Completed

Report an Issue

Ask a Question
(https://discuss.educative.io/tag/merge-intervals-medium__pattern-merge-intervals__grokking-the-coding-interview-patterns-for-coding-questions)