Facebook | Merge two interval lists

hychin

40117 Jun 28, 2016 Jul 30, 2019

Meta Algorithm Interview

Given A and B two interval lists, A has no overlap inside A and B has no overlap inside B. Write the function to merge two interval lists, output the result with no overlap. Ask for a very efficient solution

A naive method can combine the two list, and sort and apply merge interval in the leetcode, but is not efficient enough.

For example,
A: [1,5], [10,14], [16,18]
B: [2,6], [8,10], [11,20]

output [1,6], [8, 20]

Explore

#Interview
2025 Hiring Prep Sprint – Big Te...
Meta - Got Offer | SWE | E3 US New ...

#Compensation
Help Needed || Wells Fargo
Google | L4 | SWE 3 | India| Hyderab...

#Google
Javascript for Google coding intervi...
Google | L4 | SWE 3 | India| Hyderab...

#Google Interview
Google || TSE || Technical Solutions E...
Google L3 | Feeback

Show More

Comment

Morningstar Jan 22, 2021

Simple Merge Interval Logic

```
A = [[1,5],[10,14],[16,18]]
B = [[2,6],[8,10],[11,20]]

i = 0
j = 0
res = []
while i < len(A) or j < len(B):
    if i==len(A):
        curr = B[j]
        j+=1
    elif j==len(B):
        curr = A[i]
        i+=1
    elif A[i][0] < B[j][0]:
        curr = A[i]
        i+=1
    else:
        curr = B[j]
        j+=1
    if res and res[-1][-1] >= curr[0]:
        res[-1][-1] = max(res[-1][-1],curr[-1])
    else:
        res.append(curr)

print(res)
```

102 Show 4 Replies Reply

alpharoz Jan 21, 2019

I had this exact question in a recent FB interview.

80 Reply

CodingTony Dec 21, 2016

Working Java solution: use two pointers

```
import java.util.*;

class Interval {
    int start;
    int end;
    public Interval(int start, int end) {
        this.start = start;
        this.end = end;
    }
}

class myComparator implements Comparator<Interva
    @Override
    public int compare(Interval i1, Interval i2)
        if (i1.start == i2.start) {
            return 0;
        } else {
            return i1.start < i2.start? -1: 1;
        }
    }
}

public class IntervalMerge {
    public List<Interval> mergeList(List<Interva
        if (l1 == null || l1.size()  == 0) {
            return l2;
        } else if (l2 == null || l2.size() == 0)
            return l1;
        }

        Collections.sort(l1, new myComparator())
        Collections.sort(l2, new myComparator())

        List<Interval> result = new ArrayList<>(
        int ix1 = 0;
        int ix2 = 0;
        // Get the first interval
        Interval prev = null;
```

```
        if (l1.get(0).start < l2.get(0).start) {
            prev = l1.get(0);
            ix1 ++;
        } else {
            prev = l2.get(0);
            ix2 ++;
        }
        // Move two pointers to merge lists
        while (ix1 < l1.size() || ix2 < l2.size(
            if (ix2 == l2.size() || (ix1 < l1.si
                // merge prev with ix1
                if (prev.end < l1.get(ix1).start
                    result.add(prev);
                    prev = l1.get(ix1);
                } else {
                    prev.end = Math.max(prev.enc
                }
                ix1 ++;
            } else {
                // merge prev with ix2
                if (prev.end < l2.get(ix2).start
                    result.add(prev);
                    prev = l2.get(ix2);
                } else {
                    prev.end = Math.max(prev.enc
                }
                ix2 ++;
            }
        }
        result.add(prev);
        return result;
    }

    public static void main(String[] args) {
        IntervalMerge myObj = new IntervalMerge(

        List<Interval> l1 = new ArrayList<>();
        l1.add(new Interval(1, 5));
        l1.add(new Interval(10, 14));
        l1.add(new Interval(16, 18));
```

```
        l1.add(new Interval(20, 24));
        l1.add(new Interval(30, 38));
        List<Interval> l2 = new ArrayList<>();
        l2.add(new Interval(2, 6));
        l2.add(new Interval(8, 10));
        l2.add(new Interval(11, 20));

        List<Interval> result = myObj.mergeList(
        for (Interval i1: result) {
            System.out.println(i1.start + ", " +
        }
    }

}
```

27 Show 2 Replies Reply

Anqi Hu Nov 16, 2017

Here is my c++ code using two pointer:

```
vector<pair<int, int>> mergeNonOverlappingInter
        int s = INT_MIN, e = INT_MIN, i = 0, j =
        vector<pair<int,int>> res;
        while (i < a.size() || j < b.size()) {
            pair<int, int> cur;
            if (i >= a.size()) cur = b[j++];
            else if (j >= b.size()) cur = a[i++]
            else cur = a[i].first < b[j].first ?
            if (cur.first > e) {
                if (e > INT_MIN)
                    res.emplace_back(s, e);
                s = cur.first;
                e = cur.second;
            }
            else {
                e = max(cur.second, e);
            }
        }
```

```
        if (e > INT_MIN) res.emplace_back(s, e);
        return res;

    }
```

18 Show 2 Replies Reply

LEI

Apr 10, 2022

I think it's similar to Problem "Interval List intersections" with the only difference it's asking for a union rather than an intersection. The idea is the same though

12 Reply

sachshah16

Mar 02, 2018

Super easy to understand Java Solution

```
private List<Interval> mergeIntervalsLists( List
        List<Interval> result = new ArrayList();
        if ( list1 == null && list2 == null ) re
        if ( list1 == null ) return list2;
        if ( list2 == null ) return list1;

        sortList(list1);
        sortList(list2);

        int i = 0, j = 0;
        Interval merge = new Interval ( list1.ge

        while ( i < list1.size() && j < list2.si
            Interval i1 = list1.get(i);
            Interval i2 = list2.get(j);
            if ( i1.start <= merge.end ) {
                mergeIntervals( merge, i1 );
                i++;
            } else if ( i2.start <= merge.end )
                mergeIntervals( merge, i2 );
                j++;
            } else {
```

```
                result.add( merge );
                merge = new Interval ( i1.start
            }
        }

        while( i < list1.size() ) {
            Interval i1 = list1.get(i);
            if ( i1.start <= merge.end ) {
                mergeIntervals( merge, i1 );
                i++;
            } else {
                result.add( merge );
                merge = new Interval( i1 );
            }
        }

        while( j < list2.size() ) {
            Interval i2 = list2.get(j);
            if ( i2.start <= merge.end ) {
                mergeIntervals( merge, i2 );
                j++;
            } else {
                result.add( merge );
                merge = new Interval( i2 );
            }
        }

        result.add( merge );
        return result;
    }

    private void mergeIntervals( Interval mergeInto,
        mergeInto.start = Math.min( mergeInto.st
        mergeInto.end = Math.max( mergeInto.end,
    }

    private void sortList( List<Interval> list ) {
        Collections.sort( list, new Comparator<I
            @Override
            public int compare( Interval i1, Int
```

```
                return ( i1.start < i2.start ) ?
            }
        } );
    }

    //Just for reference!
    class Interval {
        int start, end;
        public Interval(int start, int end) {
            this.start = start;
            this.end = end;
        }
        public Interval( Interval interval ) {
            this.start = interval.start;
            this.end = interval.end;
        }
    }
```

For list of sizes m and n where n > m
Time Complexity:
O(n log n + m) if sorting of each list is required.
If the input lists are already sorted, then the algorithm runs in O(m + n) time

Space Complexity:
No additional memory than the result. For the result O(m + n) is used in the worst case

12 Show 1 Replies Reply

Chao Shen Jun 12, 2019

A java solution.
2 pointers.

```
public class IntervalListUnion {
    public List<int[]> getUnion(int[][] interval
        int i = 0;
        int j = 0;
        List<int[]> result = new ArrayList<>();
        int[] pre = null;
```

```
        while(i<interval1.length || j <interval2
            int[] cur = null;
            if(i>=interval1.length) {
                cur = interval2[j];
                j++;
            }else if(j>=interval2.length) {
                cur = interval1[i];
                i++;
            }else if(interval1[i][0]<interval2[j
                cur = interval1[i];
                i++;
            }else {
                cur = interval2[j];
                j++;
            }

            if(pre==null) {
                pre = cur;
            }else {
                if(pre[1]<cur[0]) {
                    result.add(pre);
                    pre = cur;
                }else {
                    pre[1] = Math.max(pre[1], cu
                }
            }
        }

        if(pre!=null) {
            result.add(pre);
        }

        return result;
    }

    public static void main(String[] args) {
        IntervalListUnion union = new IntervalLi
        List<int[]> result = union.getUnion(new
        System.out.println("Test1");
        for(int[] interval : result) {
```

```
            System.out.print("[" + interval[0] +
        }
        System.out.println("");

        System.out.println("Test2");
        result = union.getUnion(new int[][] {},
        for(int[] interval : result) {
            System.out.print("[" + interval[0] +
        }
        System.out.println("");

        System.out.println("Test3");
        result = union.getUnion(new int[][] {{1,
        for(int[] interval : result) {
            System.out.print("[" + interval[0] +
        }
        System.out.println("");

        System.out.println("Test4");
        result = union.getUnion(new int[][] {},
        for(int[] interval : result) {
            System.out.print("[" + interval[0] +
        }
        System.out.println("");
    }
}
```

9 Reply

derpt34 Apr 02, 2024

just got asked this as part of E5 onsite

8 Show 3 Replies Reply

Jayan Mar 17, 2022

Here is my python3 solution -
https://leetcode.com/playground/2gRif9c4

```
# A: [1,5], [10,14], [16,18]
# B: [2,6], [8,10], [11,20]
# output: [[1, 6], [8, 20]]
# https://leetcode.com/discuss/interview-questic
def mergeTwoIntervalLists(list1, list2):
    result = []
    i, j = 0, 0
    while i < len(list1) or j < len(list2):

        if i == len(list1): # Case A - list1 has
            temp = list2[j]
            j += 1
        elif j == len(list2): # Case B - list2 h
            temp = list1[i]
            i += 1
        elif list1[i][0] < list2[j][0]: # Case C
            temp = list1[i]
            i += 1
        else: # Case D - list2's start is smalle
            temp = list2[j]
            j += 1

        if not result or result[-1][1] < temp[0]
            result.append(temp)
        else: # overlap found
            result[-1][1] = max(result[-1][1], t

    return result

print (mergeTwoIntervalLists([[1,5], [10,14], [1
print (mergeTwoIntervalLists([[1,15], [10,14], |
print (mergeTwoIntervalLists([[1,15], [20,24], |
print (mergeTwoIntervalLists([[1,15], [20,24], |
```

8 Reply

wadhwala Oct 10, 2016

Similar to merging two sorted lists

```
def mergeIntervals(int1, int2):
    if not int1 or not int2:
        return int1 or int2
    ret = []
    i = 0
    j = 0
    if int1[0][0] < int2[0][0]:
        curr = int1[0]
        i = 1
    else:
        curr = int2[0]
        j = 1

    while i < len(int1) or j < len(int2):
        if j == len(int2) or int1[i][0] < int2[j
            nxt = int1[i]
            i += 1
        else:
            nxt = int2[j]
            j += 1
        if curr[1] < nxt[0]:
            ret.append(curr)
            curr = nxt
        else:
            curr[1] = max(curr[1], nxt[1])
    ret.append(curr)

    return ret
```

15 Show 3 Replies Reply