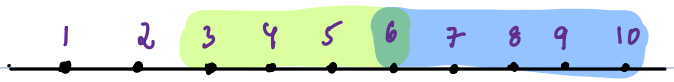
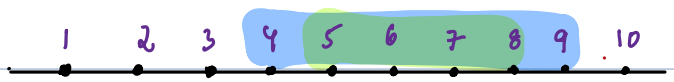
Todays Content

1. Overlapping Intervals
2. Insert in overlapping Intervals.

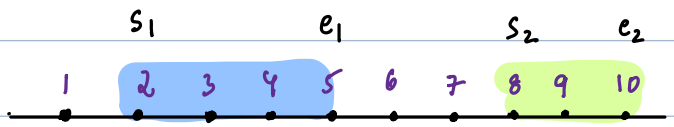
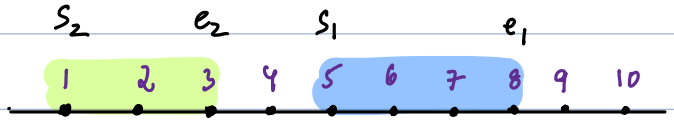
Merge Intervals: Any Interval: [s ... e]

I_1	I_2	They are overlapping	Merge Intervals
(2 6)	(3 7)		(2 7)
(2 8)	(4 6)		(2 8)
(3 7)	(4 10)		(3 10)
(3 6)	(6 10)		(3 10)
(5 8)	(4 9)		(4 9)

Overlapping Interval:

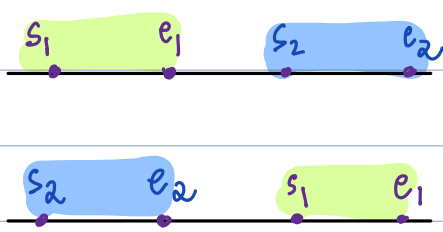
I_1	I_2	Say Overlapping	New Interval
$(s_1\ e_1)$	$(s_2\ e_2)$		$(\min(s_1, s_2)\ \max(e_1, e_2))$

Non Overlapping:

(2 5)	(8 10)	
(5 8)	(1 3)	

Non Overlapping Case:

$(s_1\ e_1)$ $(s_2\ e_2)$



#Way 1:

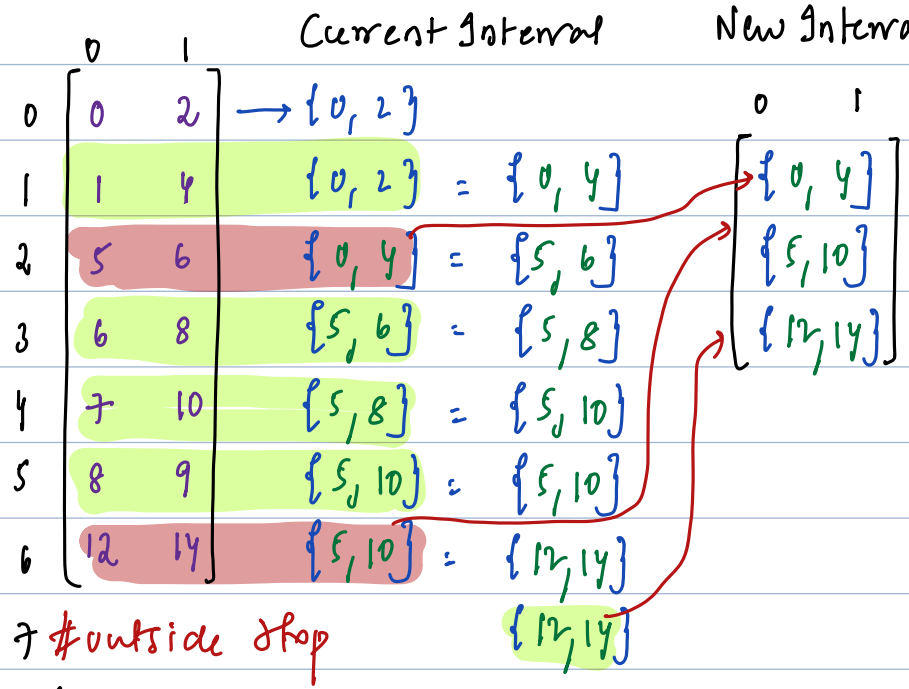
```
if( s2 > e1 || s1 > e2 ){
    Not overlapping
}
```

Q1: Given collection of intervals in a 2D array format, which are sorted based on their start time.

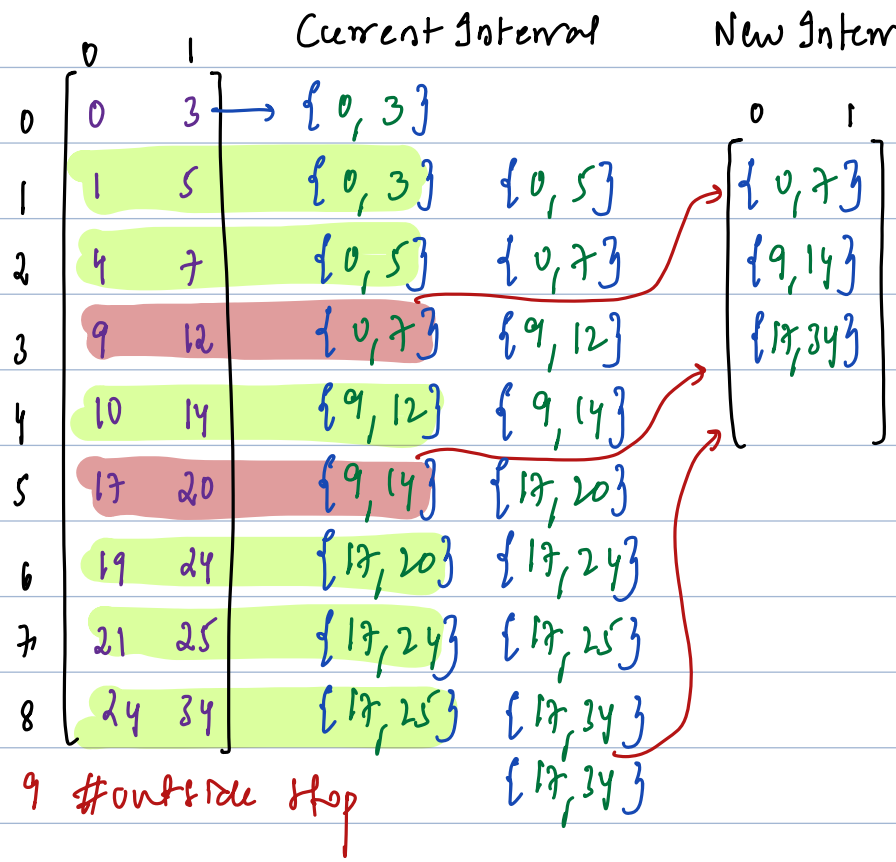
Merge all overlapping intervals & return set of non-overlapping intervals

Note: If intervals are not already sorted, sort them.

Ex1: Intervals:



Ex2: Intervals:



TC: O(N) SC: O(1)

```
vector<pair<int, int>> MergeAll(vector<pair<int, int>>& arr){
    sort(arr.begin(), arr.end()); #Sorting based on start time

    vector<pair<int, int>> newInterval;
    pair<int, int> cI = arr[0]; #cI current Interval

    for(int i=1; i<arr.size(); i++){
        // current Interval cI    arr[i].s arr[i].e    cI.s   cI.e
        // latest Interval arr[i]
        if(cI.e < arr[i].s || arr[i].e < cI.s){ # Not overlapping
            newInterval.push_back(cI); # Inserting
            cI = arr[i]; # updating          [#arr[i].s  arr[i].e]
        }else{ #overlapping merge            [#cI.s  cI.e]
            cI.s = min(arr[i].s, cI.s);
            cI.e = max(arr[i].e, cI.e);
        }
    }

    newInterval.push_back(cI);
    return newInterval;
}
```

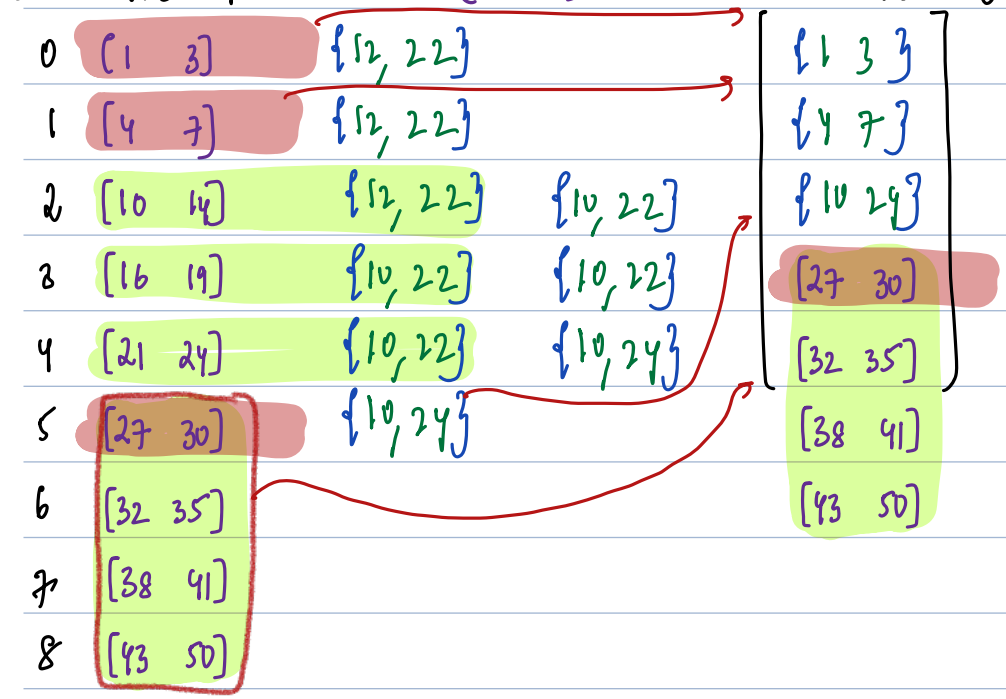
Note: In above code

cI.s = cI.first, cI.e = cI.second

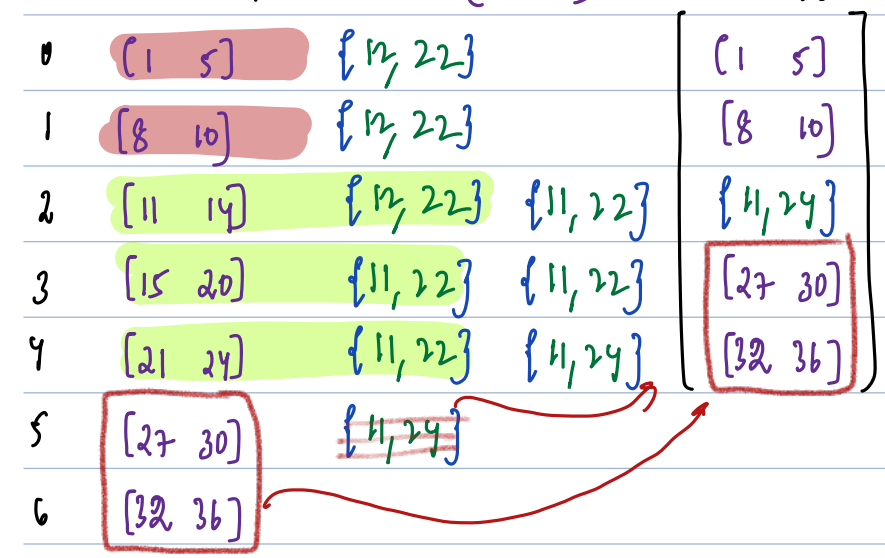
arr[i].s = arr[i].first, arr[i].e = arr[i].second.

28. Given N sorted non overlapping Interval, Insersert a new Interval
In them, {Merge If Necessary}
Return set of non-overlapping Intervals

Ex1: Interval Insert [12 22] ans Interval



Ex2: Interval Insert [12 22] ans Interval



#Idea: TC: O(N) SC: O(1)

Insert new Interval [ns, ne]

Iterate on current Interval: {cs, ce}

```
if( ce < ns or ne < cs){ #Not overalpping
    if( ce < ns){ #current Interval comes first {cs, ce} [ns ne}
        Insert curret Interval in ans ✓✓
    }
    else{ [ns ne} {cs, ce}
        Insert new Interval in ans;
        Iterate & insert all curret Intervals in ans
        retrn ans;
    }
}
else{ #overlapping. ✓✓✓
    ns = min(cs, ns);
    ne = max(ce, ne);
}
}
```

Insert new Interval in ans; #Edge Base, if we come outside loop.

retrn ans;

Ex3: Interval Insert [12 22] ans Interval

Interval	Insert	ans Interval
[1 5]	[12 22]	[1 5]
[8 10]	[12 22]	[8 10]
[11 14]	[12 22] {11, 22}	{11, 24}
[15 20]	{11, 22} {11, 22}	
[21 24]	{11, 22}	
#outside	{11, 24}	

vector<pair<int,int>> insert(vector<pair<int,int>> &ar, pair<int,int> p){

}