## INTERVAL PROBLEM

* Given N intervals, each having start and end time.
  + Two separate arrays int start[], end[]
  + Single array of custom data type interval a[];

class interval {

int start;

int end;

};

## How to find overlapping intervals?



### Code:

bool isOverlapping(interval i1, interval i2) {

return !(i1.start >= i2.end || i2.start >= i1.end);

}

interval findIntersection(interval i1, interval i2) {

if (isOverlapping(i1, i2))

return interval{ *max*(i1.start, i2.start), *min*(i1.end, i2.end) };

else

return interval{ -1, -1 };

}

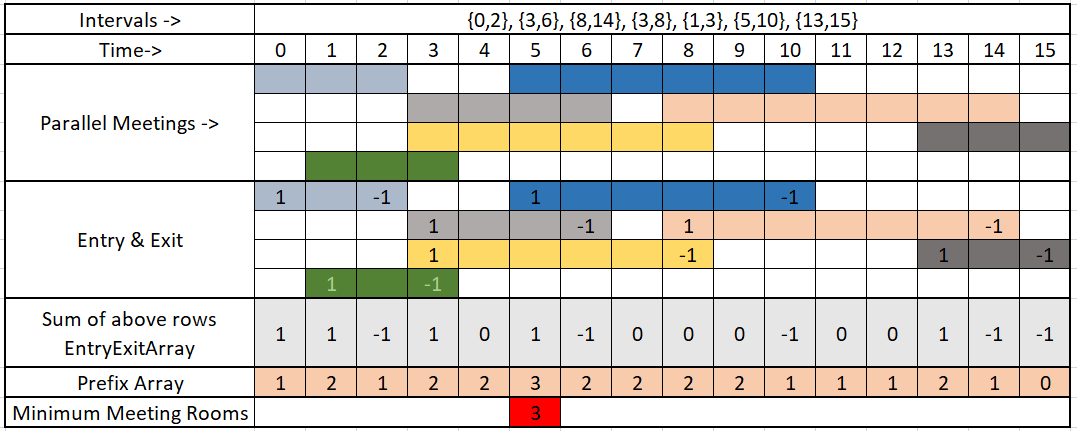
**[PROBLEM]** Given start and end time of meetings of a person, tell if there is any conflict in his meetings.

**[PROBLEM]** Given start and end time of meetings of a person, return a minimum set of intervals by merging all overlapping intervals.

**[PROBLEM]** Given start and end time of meetings of a person, inserting a new meeting, and then returning min set of intervals by merging all overlapping intervals.

**[PROBLEM]** Given start and end time of meetings of a person, Find max count of overlapping intervals at any time. Intervals = { {0,2}, {3,6}, {8,14}, {3,8}, {1,3}, {5,10}, {13,15} };

APPROACH 1:



|  |
| --- |
| struct Intervals {  int startTime;  int endTime;  };  int minimumMeetingRooms(*vector*<Intervals> intervals) {  int len = intervals.*size*();  *vector*<int> entryExitArray(24); // Hours.  for (auto i : intervals) {  entryExitArray[i.startTime] += 1;  entryExitArray[i.endTime] -= 1;  }  *vector*<int> prefixArray(24);  prefixArray[0] = entryExitArray[0];  int minRooms = prefixArray[0];  for (auto i = 1; i < 24; i++) {  prefixArray[i] = entryExitArray[i] + prefixArray[i - 1];  if (minRooms < prefixArray[i])  minRooms = prefixArray[i];  }  return minRooms;  }  int main() {  *vector*<Intervals> intervals = { {0,2}, {3,6}, {8,14}, {3,8}, {1,3}, {5,10}, {13,15} };  *cout* << minimumMeetingRooms(intervals) << *endl*;  return 0;  } |

APPROACH 2:

|  |
| --- |
| #include <iostream>  #include <vector>  #include <queue>  #include <algorithm>  using namespace *std*;  struct Interval {  int start, end;  Interval(int start, int end) {  this->start = start;  this->*end* = end;  }  };  int minMeetingRooms(*vector*<Interval> &intervals) {  if (intervals.*size*() == 0) return 0;  *sort*(intervals.*begin*(), intervals.*end*(), [](Interval a, Interval b) {  if (a.start < b.start) return true;  else return false;  });  *priority\_queue*<int, *vector*<int>, *greater*<int>> pq;  for (auto i : intervals) {  if (!pq.*empty*() && pq.*top*() <= i.start)  pq.*pop*();  pq.*push*(i.end);  }  return pq.*size*();  }  int main(void) {  *ios\_base*::*sync\_with\_stdio*(false);  *cin*.*tie*(nullptr);  *cout*.*tie*(nullptr);  *vector*<Interval> intervals = { {0,2}, {3,6}, {8,14}, {3,8}, {1,3}, {5,10}, {13,15} };  *cout* << minMeetingRooms(intervals) << "\n";  return 0;  } |