Minimum Number of Platforms Required for a Railway/Bus Station

Given arrival and departure times of all trains that reach a railway station, find the minimum number of platforms required for the railway station so that no train waits.
We are given two arrays which represent arrival and departure times of trains that stop

Examples:

```
Input:  arr[]  = {9:00,  9:40, 9:50,  11:00, 15:00, 18:00}
        dep[]  = {9:10, 12:00, 11:20, 11:30, 19:00, 20:00}
Output: 3
There are at-most three trains at a time (time between 11:00 to 11:20)
```

We strongly recommend that you click here and practice it, before moving on to the solution.

We need to find the maximum number of trains that are there on the given railway station at a time. A **Simple Solution** is to take every interval one by one and find the number of intervals that overlap with it. Keep track of maximum number of intervals that overlap with an interval. Finally return the maximum value. Time Complexity of this solution is $O(n^2)$.

We can solve the above problem **in O(nLogn) time**. The idea is to consider all evens in sorted order. Once we have all events in sorted order, we can trace the number of trains at any time keeping track of trains that have arrived, but not departed.

For example consider the above example.

```
    arr[]  = {9:00,  9:40, 9:50,  11:00, 15:00, 18:00}
    dep[]  = {9:10, 12:00, 11:20, 11:30, 19:00, 20:00}

All events sorted by time.
Total platforms at any time can be obtained by subtracting total
departures from total arrivals by that time.
 Time     Event Type     Total Platforms Needed at this Time
 9:00       Arrival                1
 9:10       Departure              0
 9:40       Arrival                1
 9:50       Arrival                2
 11:00      Arrival                3
 11:20      Departure              2
 11:30      Departure              1
 12:00      Departure              0
 15:00      Arrival                1
 18:00      Arrival                2
 19:00      Departure              1
 20:00      Departure              0

 Minimum Platforms needed on railway station = Maximum platforms
                                               needed at any time
                                             = 3
```

Following is C++ implementation of above approach. Note that the implementation doesn't create a single sorted list of all events, rather it individually sorts arr[] and dep[] arrays, and then uses merge process of merge sort to process them together as a single sorted array.

```
// Program to find minimum number of platforms required on a railway station
#include<iostream>
#include<algorithm>
using namespace std;

// Returns minimum number of platforms reqquired
int findPlatform(int arr[], int dep[], int n)
{
   // Sort arrival and departure arrays
   sort(arr, arr+n);
   sort(dep, dep+n);

   // plat_needed indicates number of platforms needed at a time
   int plat_needed = 1, result = 1;
   int i = 1, j = 0;

   // Similar to merge in merge sort to process all events in sorted order
   while (i < n && j < n)
   {
      // If next event in sorted order is arrival, increment count of
      // platforms needed
      if (arr[i] < dep[j])
      {
          plat_needed++;
          i++;
          if (plat_needed > result)  // Update result if needed
              result = plat_needed;
      }
      else // Else decrement count of platforms needed
      {
          plat_needed--;
          j++;
      }
   }

   return result;
}

// Driver program to test methods of graph class
int main()
{
    int arr[] = {900, 940, 950, 1100, 1500, 1800};
    int dep[] = {910, 1200, 1120, 1130, 1900, 2000};
    int n = sizeof(arr)/sizeof(arr[0]);
    cout << "Minimum Number of Platforms Required = "
         << findPlatform(arr, dep, n);
    return 0;
}
```

Run on IDE

Output:

```
Minimum Number of Platforms Required = 3
```

Algorithmic Paradigm: Dynamic Programming

Time Complexity: O(nLogn), assuming that a O(nLogn) sorting algorithm for sorting arr[] and dep[].