CALCULATING THE MINIMUM NUMBER OF PLATFORMS REQUIRED FOR A RAILWAY STATION using Sorting technique

IDEA :All the events that the trains are arriving and departuring are sorted by time.

LOGIC :Total platforms at any time can be obtained by subtracting total departures from total arrivals by that time. Therefore,Minimum Platforms needed on railway station = Maximum platforms needed at any time.

Following the below steps to write a code for the given problem statement.

STEP 1: Firstly,sorting the arrival and departure times of trains into 2 arrays i.e arr(arrival time's array),dep(depature time's array).

STEP 2: Creating two pointers i=1 and j=0 and a variable to store ans(answer) and current count of the plat (platforms).

STEP 3: Running a while loop where i<n and j<n (where n is the size of the array)& comparing the ith element of arrival array and jth element of departure array.

STEP 4: If the arrival time is less than or equal to departure then one more platform is needed so increasing the count, i.e plat += 1 and incrementing i i.e i += 1

STEP 5: If the arrival time is greater than departure then one less platform is needed so decreasing the count, i.e., plat -= 1 and incrementing j i.e j += 1.

STEP 6:Finally Updating the ans(answer), i.e. ans = max(ans, plat).

In [2]:
```
# Program to find minimum number of platformsrequired on a railway station.
# Returns minimum number of platforms required.


def findPlatform(arr, dep, n): #function named 'findPlatform' that finds the min no. o

    # Sorting arrival and departure arrays.
    arr.sort()
    dep.sort()

    # plat indicates number of platforms needed at a time.
    plat = 1
    result = 1
    i = 1       # i is the index pointer of array arrival i.e arr.
    j = 0       # j is the index pointer of array departure i.e dep.

    # Similar to merge in merge sort, to process all events in sorted order.
    while (i < n and j < n):     # n is the total number elements in arr/dep.

        # If the next event in sorted order is arrival then incrementing count of plat
        if (arr[i] <= dep[j]):

            plat += 1
```

```
            i += 1

        # Else decrementing count of platforms needed.
        elif (arr[i] > dep[j]):

            plat -= 1
            j += 1

       # Updating the result.
        if (plat > result):
            result = plat

    return result

# Driver code


arr = [900, 940, 950, 1100, 1500, 1800]
dep = [910, 1200, 1120, 1130, 1900, 2000]
n = len(arr)

print("Minimum Number of Platforms Required = ",
    findPlatform(arr, dep, n))
```

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

Explanation: There are at-most three trains at a time (time between 9:40 to 12:00)

Given test case 1 passed.

.

Similarly testing the given testcase 2

In [3]:
```
# Program to find minimum number of platformsrequired on a railway station.
# Returns minimum number of platforms required.


def findPlatform(arr, dep, n): #function named 'findPlatform' that finds the min no. o

    # Sorting arrival and departure arrays.
    arr.sort()
    dep.sort()

    # plat indicates number of platforms needed at a time.
    plat = 1
    result = 1
    i = 1       # i is the index pointer of array arrival i.e arr.
    j = 0       # j is the index pointer of array departure i.e dep.

    # Similar to merge in merge sort, to process all events in sorted order.
    while (i < n and j < n):     # n is the total number elements in arr/dep.

        # If the next event in sorted order is arrival then incrementing count of plat
        if (arr[i] <= dep[j]):

            plat += 1
            i += 1
```

```
        # Else decrementing count of platforms needed.
        elif (arr[i] > dep[j]):

            plat -= 1
            j += 1

       # Updating the result.
        if (plat > result):
            result = plat

    return result

# Driver code


arr = [900, 940 ]
dep = [910, 1200]
n = len(arr)

print("Minimum Number of Platforms Required = ",
    findPlatform(arr, dep, n))
```

```
Minimum Number of Platforms Required =  1
```

Explanation: Only one platform is needed.

Given test case 2 also passed.

Hence claiming that the above written code is accurate.